Prétraitement de données ^{1}H-NMR avec PepsNMR - Human Serum dataset

B. Govaerts & M. Martin

May 17, 2018,10:40

Contents

Description des données et but

"In the HSerum dataset, a blood sample was collected for 4 different donors. For each sample, 8 sub-samples were measured across 8 days with one sub-sample of each donor per day and permutations according to a latin hypercube sampling method. The total number of available FID signals is then 4*8 = 32. Data were acquired with a 500MHz Bruker Advance spectrometer using a CPMG relaxation-editing sequence with pre-saturation. Spectra are labelled as: Jx-Dy-1D-Tx where x is the day of measurement, y is the donor label, 1D means that the spectra is a one day spectra and Tz is the order of the measure of within the day."

La source de variance principale dans ces données est donc le donneur. Le prétraitement devrait donc idéalement donner une matrice de spectres qui maximise la variabilité entre les spectres des différents donneurs (variance "between") et minimise la variance entre spectres au sein d'un même donneur car ce sont uniquement des répétitions de mesures (variance "within").

Paramètrage du prétraitement et de l'affichage des étapes de prétraitements

```
## [1] "Group_HS.csv" "HumanSerum"
```

```
## group
## D1 D2 D3 D4
##  8  8  8  8
```

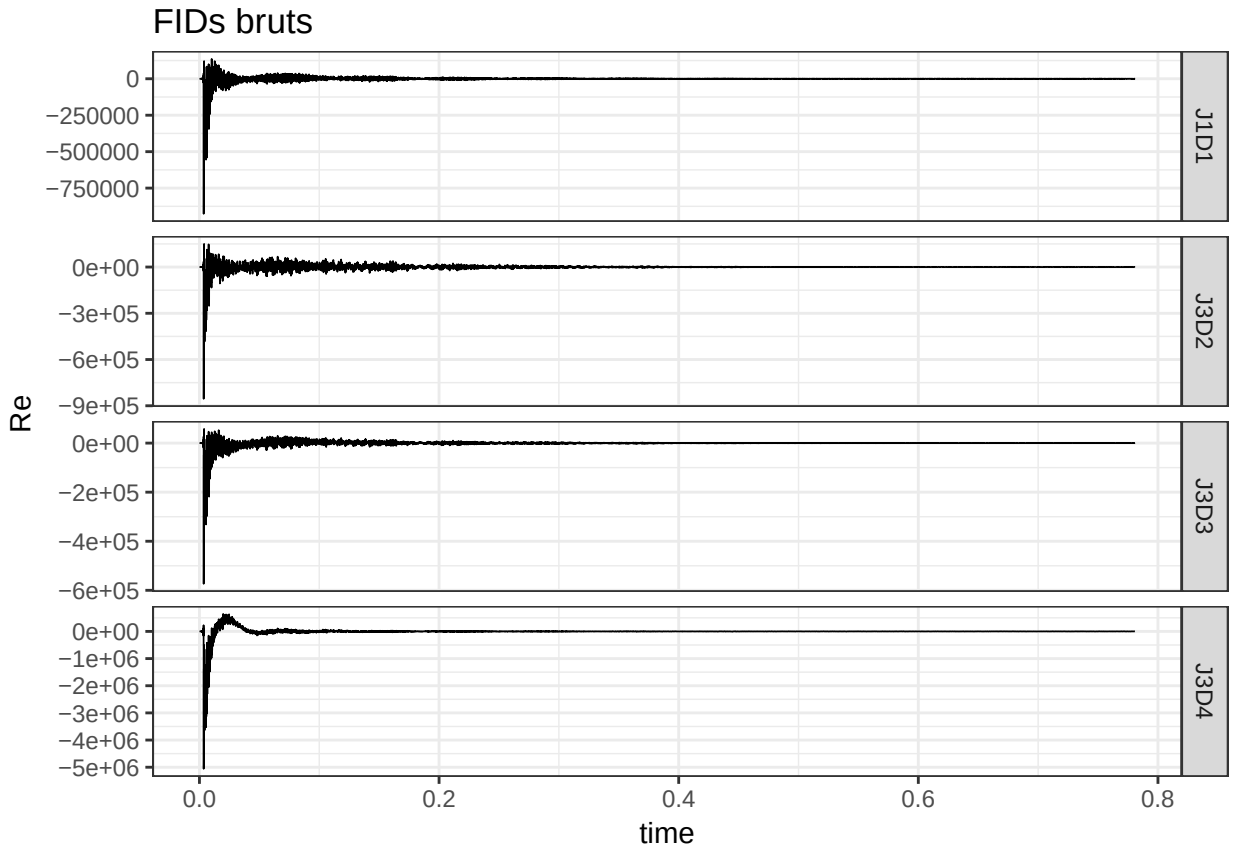
```
## group_num
## 1 2 3 4
## 8 8 8 8
```

Lecture des FID

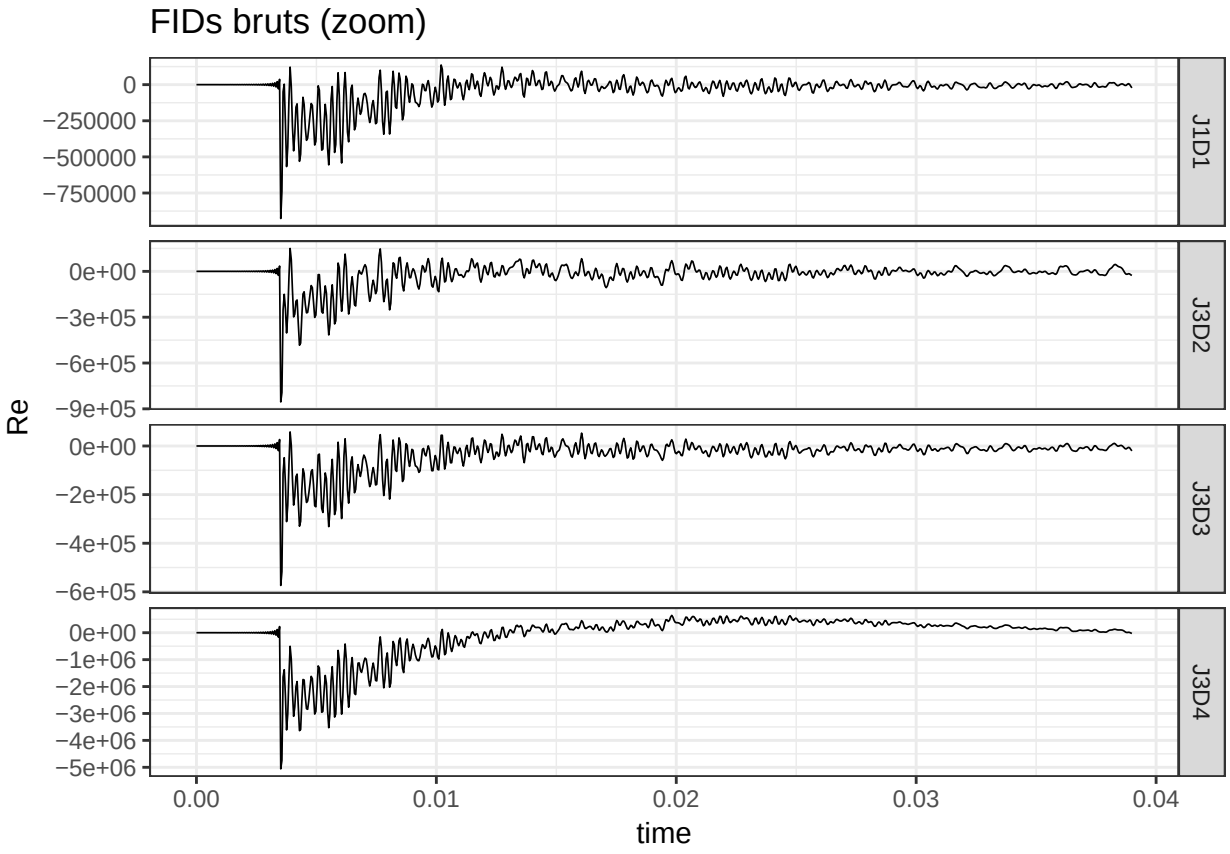
```
# Lecture des Fid
fidList <- ReadFids(path = file.path(data.path, dataname), subdirs = FALSE)
```

```
## Begin ReadFids
## dim Fid_data:  32 32768
## IDs:  J1-D1-1D-T1 J3-D2-1D-T8 J3-D3-1D-T9 J3-D4-1D-T14 J4-D1-1D-T4 J4-D2-1D-T7 J4-D3-1D-T10 J4-D4-1D
## non-unique IDs? 0
## End ReadFids
## It lasted 0.556 s user time, 0.075 s system time and 0.684 s elapsed time.
```

```
# créer la matrice spectrale
Fid_data <- fidList[["Fid_data"]]

# créer la matrice d'infos sur les paramètres d'acquisition
Fid_info <- fidList[["Fid_info"]]

# réarranger les noms de ligne dans Fid_data et Fid_info pour les
# rendre plus lisibles
rownames(Fid_data) <- substr(rownames(Fid_data),1,5)
rownames(Fid_data) <- sub("-","", rownames(Fid_data))
rownames(Fid_info) <- rownames(Fid_data)


# Représentation graphique des FID
if(DrawFid == T) Draw(Fid_data[WhichSpectra,Fid_window], type.draw = "signal",
                      num.stacked = NumStack, main="FIDs bruts", xlab = "time")
```




```
# Représentation graphique des FID agrandis (1/20eme de la fenêtre spectrale)
if(DrawFid==T) Draw(Fid_data[WhichSpectra, round(Fid_window/20)],
                    type.draw = "signal", num.stacked = NumStack,
                    main = "FIDs bruts (zoom)", xlab = "time")
```



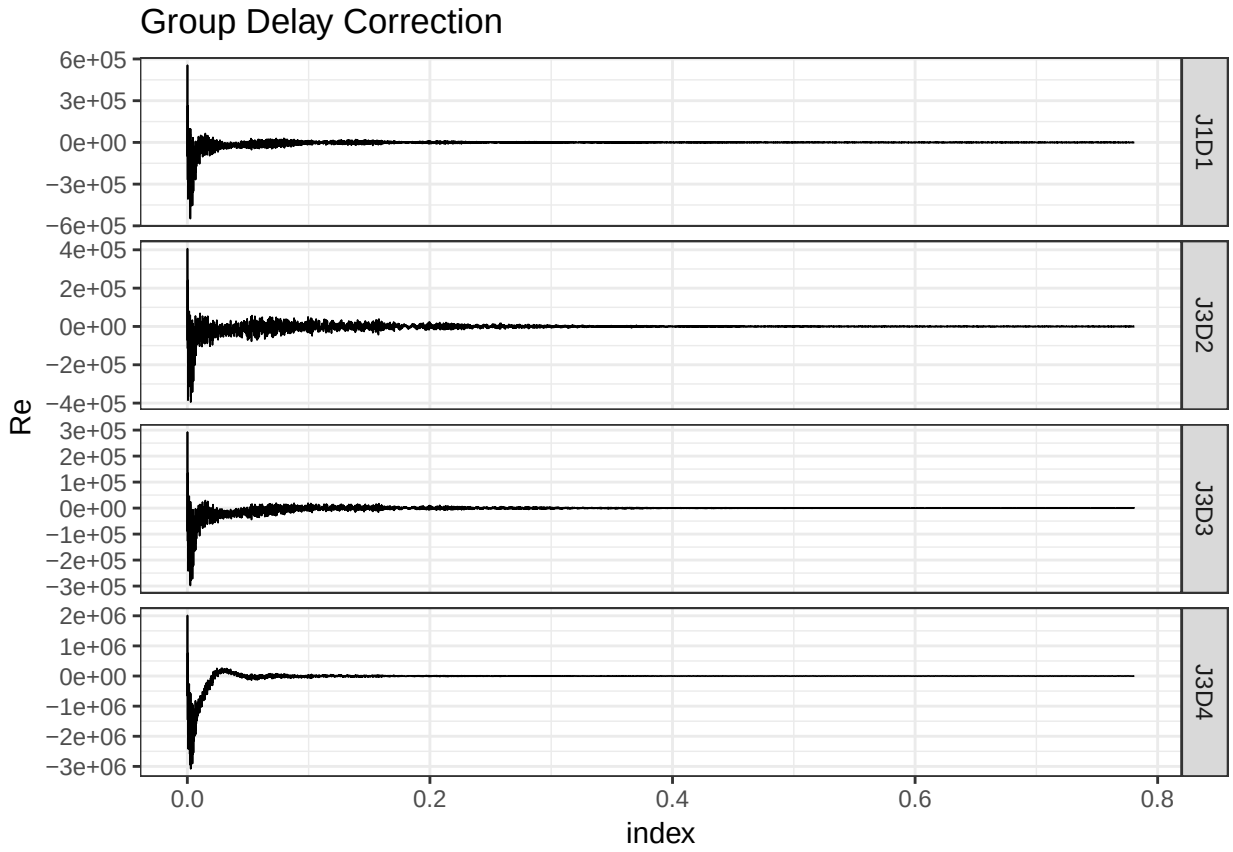
Prétraitement du FID

Correction du Group Delay

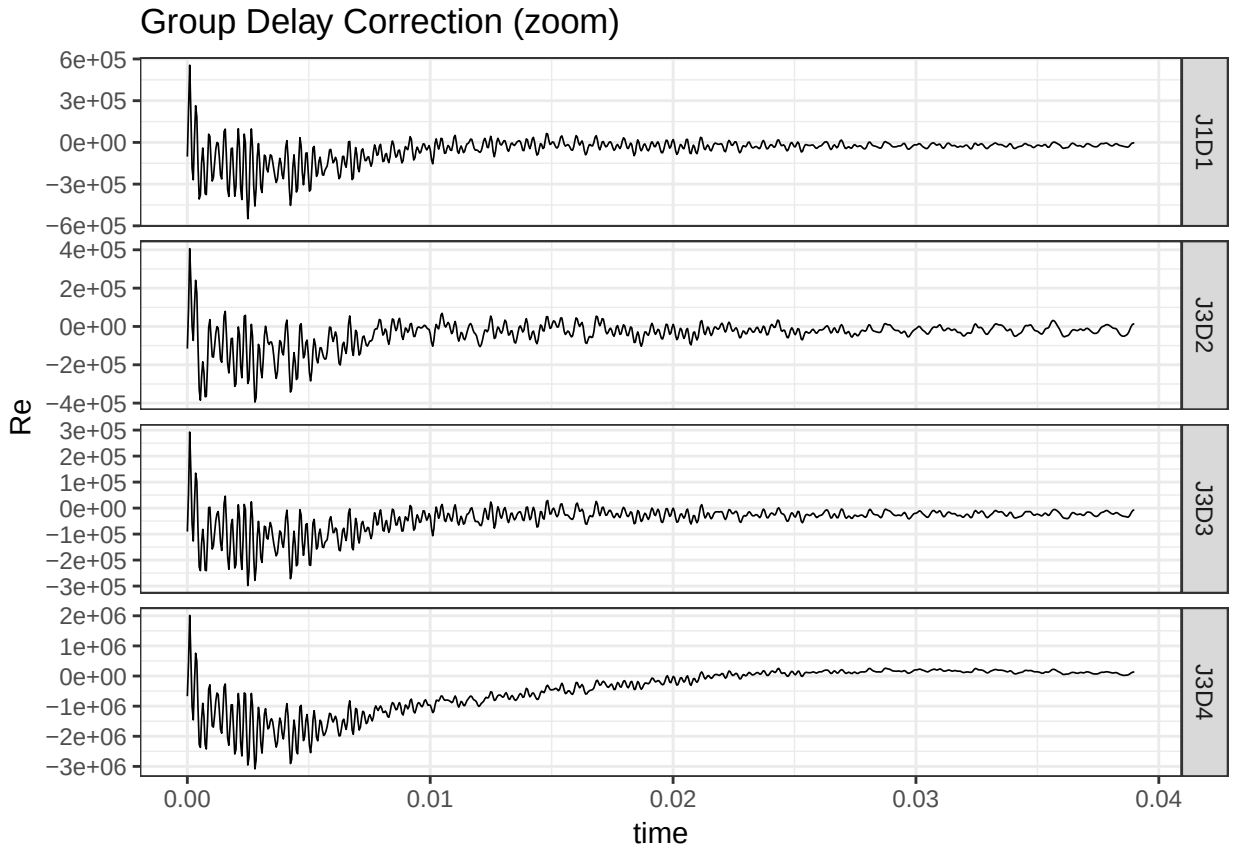
```
# GroupDelayCorrection
Fid_data_GDC <- GroupDelayCorrection(Fid_data, Fid_info)
```

```
## Begin GroupDelayCorrection
## End GroupDelayCorrection
## It lasted 0.651 s user time, 0.061 s system time and 0.719 s elapsed time.
```

```
# Représentation graphique des FID
if(DrawFid==T) Draw(Fid_data_GDC[WhichSpectra, Fid_window], type.draw = "signal",
                    num.stacked = NumStack, main = "Group Delay Correction")
```




```
# Représentation graphique des FID agrandis (1/20eme de la fenêtre spectrale)
if(DrawFid==T) Draw(Fid_data_GDC[WhichSpectra, round(Fid_window/20)],
                    type.draw = "signal",num.stacked = NumStack,
                    main = "Group Delay Correction (zoom)", xlab = "time")
```



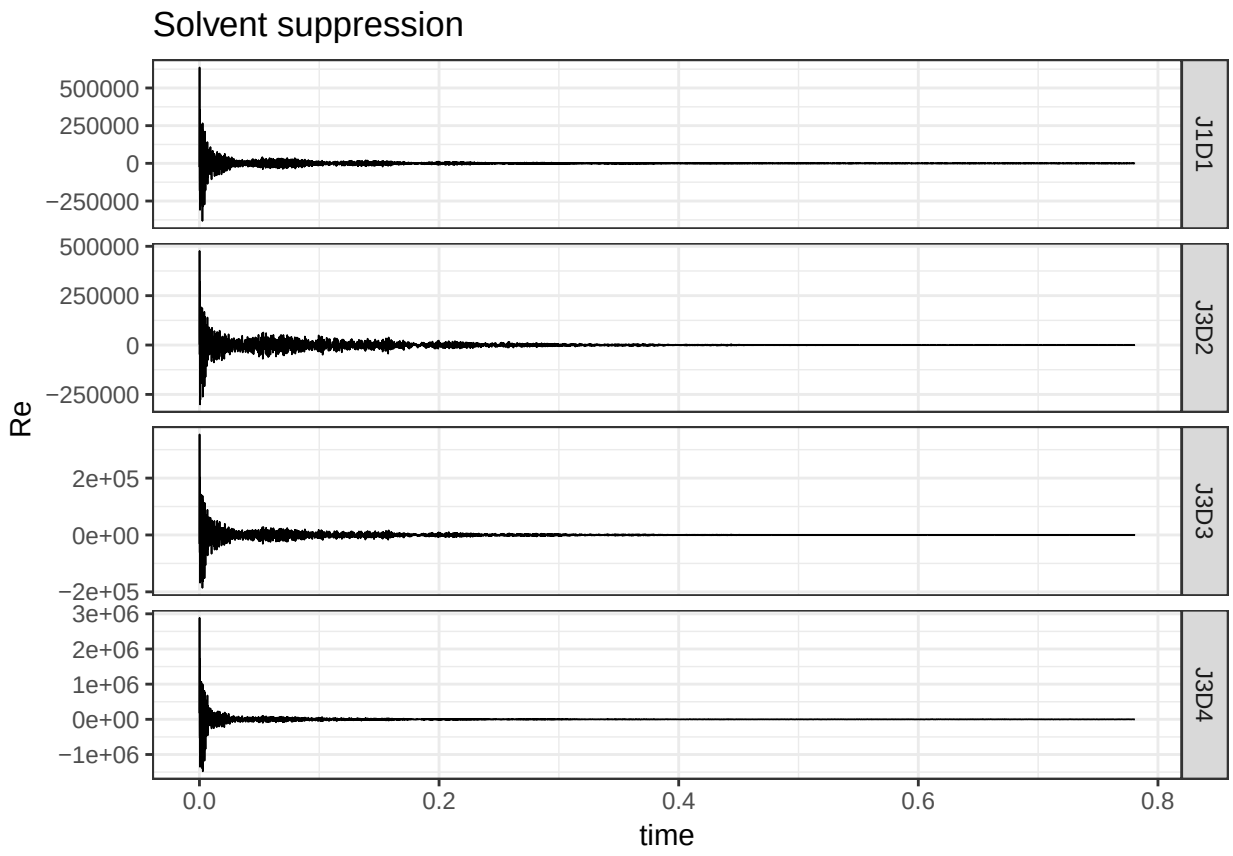
Solvent Suppression

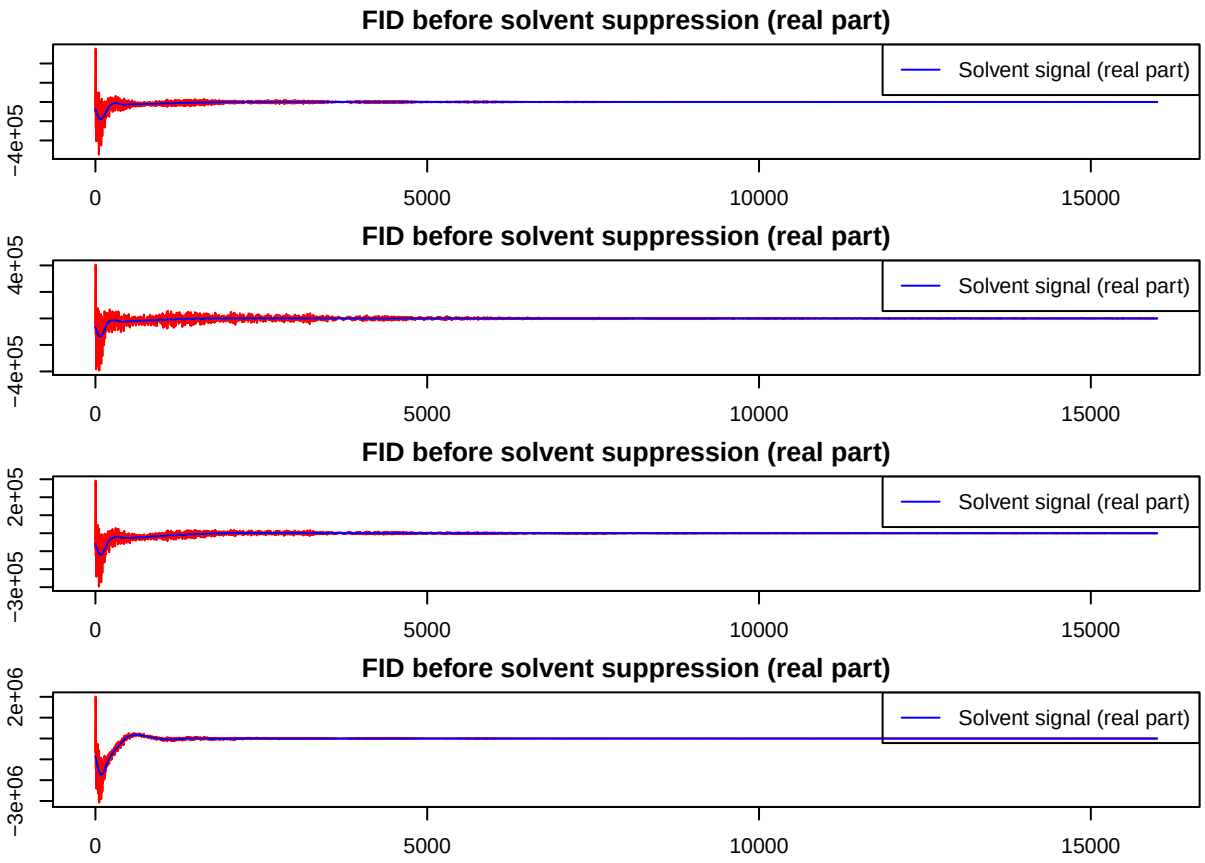
```
# Solvent Suppression
res_solventSuppres <- SolventSuppression(Fid_data_GDC, returnSolvent = TRUE)
```

```
## Begin SolventSuppression
## End SolventSuppression
## It lasted 0.41 s user time, 0.13 s system time and 0.547 s elapsed time.
```

```
# Extraire la matrice spectrale
Fid_data_SS <- res_solventSuppres$Fid_data

# Extraire la partie réelle du signal du solvent estimé
SolventRe <- Re(res_solventSuppres$SolventRe)

# Représentation graphique des FID
if(DrawFid==T){
  Draw(Fid_data_SS[WhichSpectra,Fid_window], type.draw = "signal",
       num.stacked = NumStack, main = "Solvent suppression", xlab = "time")

  par(mfrow=c(4,1), mar=c(2,2,2,2))
   for (i in WhichSpectra)  {
     plot(Re(Fid_data_GDC[i,Fid_window]), type = "l", col = "red",
          main = "FID before solvent suppression (real part)", xlab = "index")
     lines(SolventRe[i,Fid_window], type = "l",col = "blue")
     legend("topright", legend = c("Solvent signal (real part)"),
```

```
            col="blue", lty=1)
  }

}
```



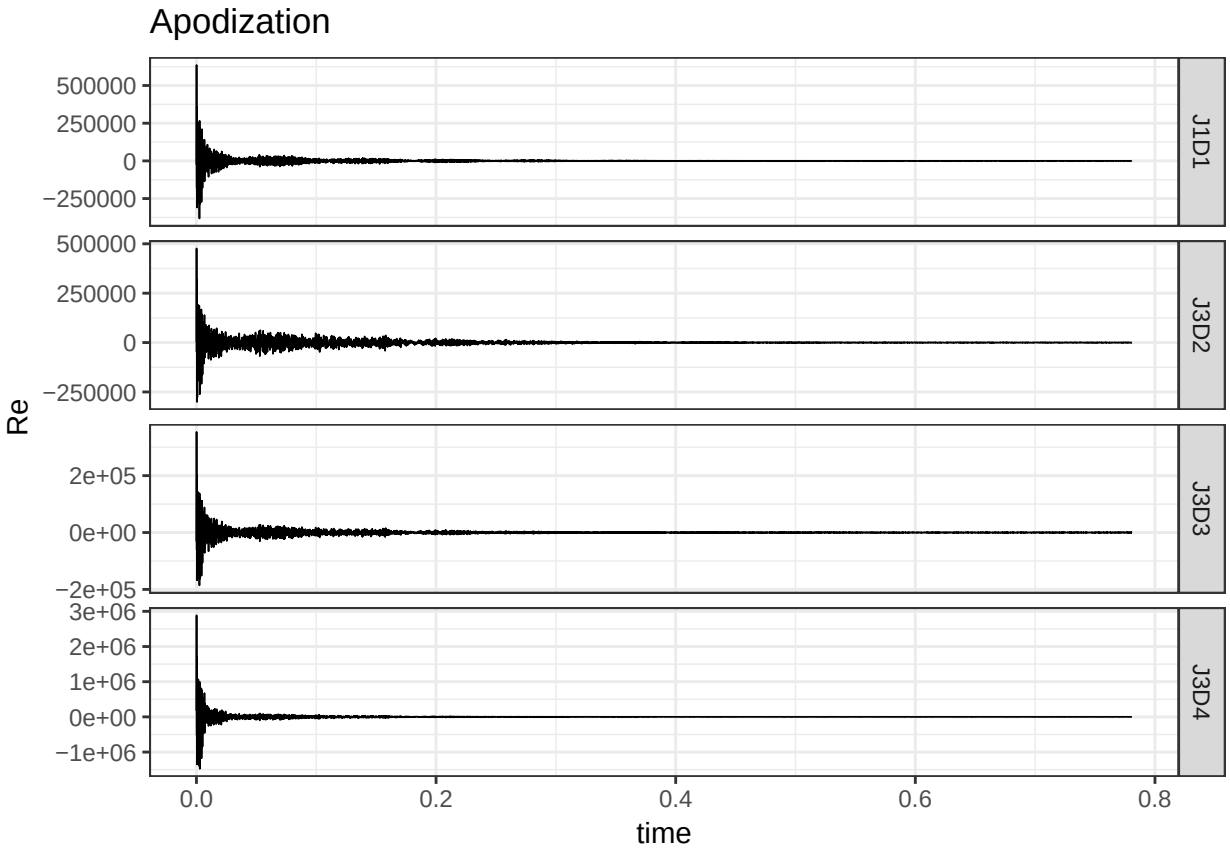


Apodization

```
# Apodization
Fid_data_Apod <- Apodization(Fid_data_SS, Fid_info)
```

```
## Begin Apodization
## End Apodization
## It lasted 0.023 s user time, 0.014 s system time and 0.037 s elapsed time.
```

```
# Représentation graphique des FID
if(DrawFid==T) Draw(Fid_data_Apod[WhichSpectra, Fid_window], type.draw = "signal",
                    num.stacked = NumStack, main = "Apodization", xlab = "time")
```

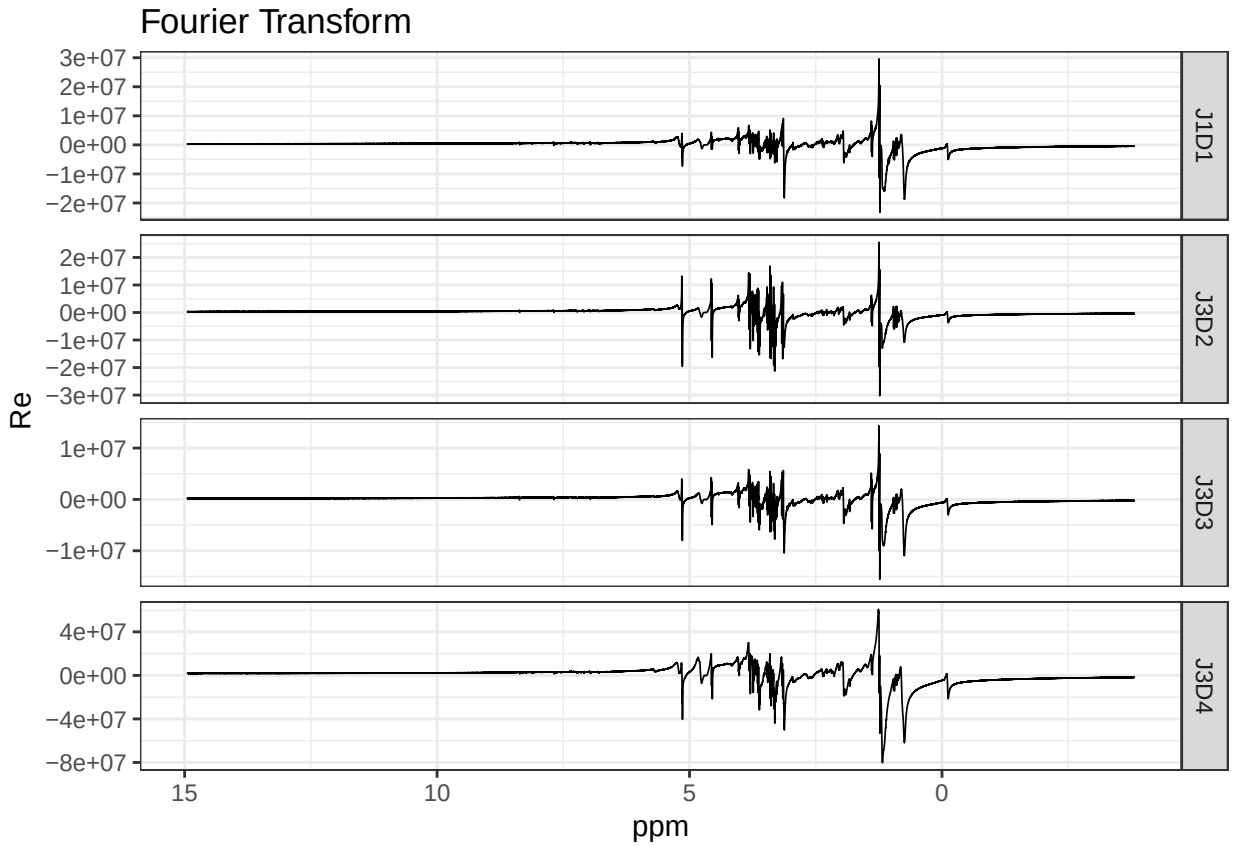


Transformée de Fourier

```
# FourierTransform
Spectrum_data_FT <- FourierTransform(Fid_data_Apod, Fid_info)
```

```
## Begin FourierTransform
## End FourierTransform
## It lasted 0.339 s user time, 0.076 s system time and 0.531 s elapsed time.
```

```
# Représentation graphique des FID
if(DrawSpectra==T) Draw(Spectrum_data_FT[WhichSpectra,Raw_Spec_window],
                        type.draw = "signal", num.stacked = NumStack,
                        main="Fourier Transform", xlab = "ppm")
```



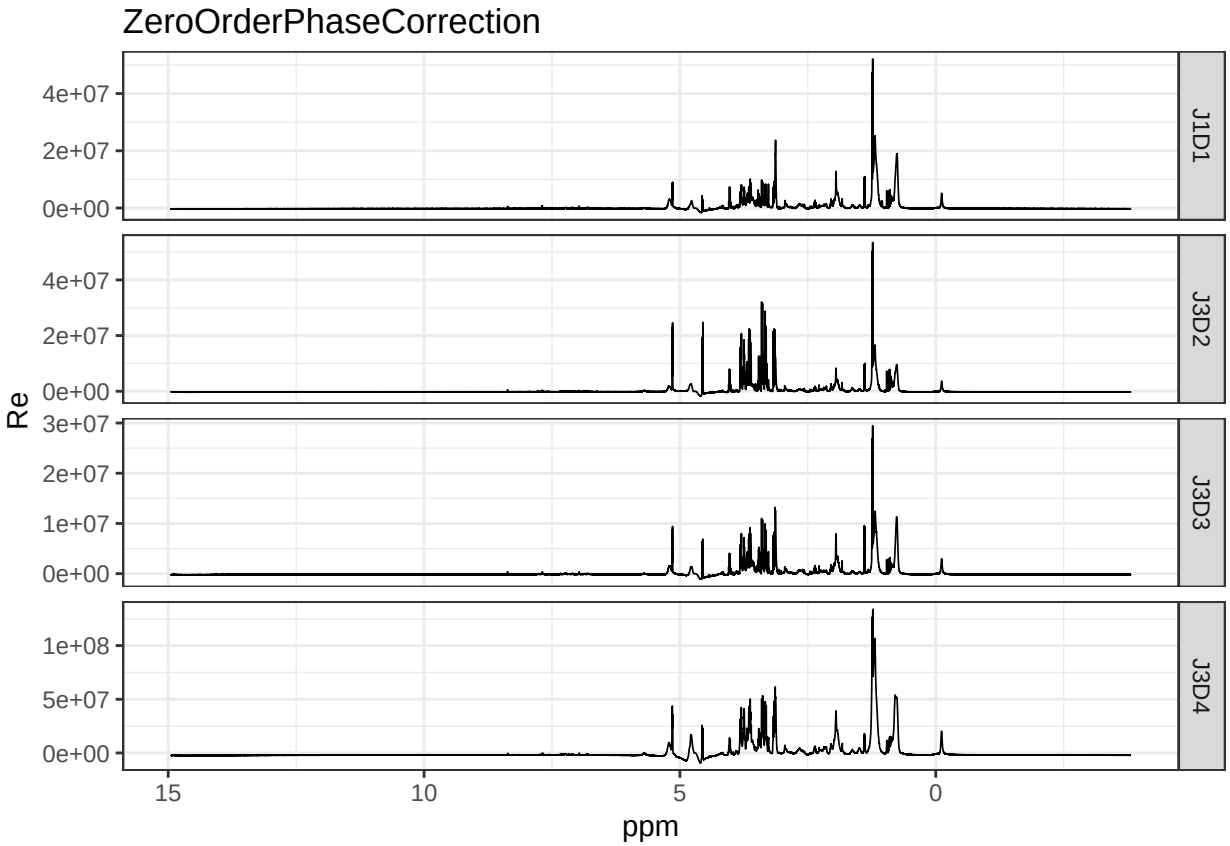
Prétraitements des spectres après la Transformée de Fourier

Correction de phase d'ordre 0

```
# ZeroOrderPhaseCorrection
Spectrum_data_ZOPC <- ZeroOrderPhaseCorrection(Spectrum_data_FT)
```

```
## Begin ZeroOrderPhaseCorrection
## End ZeroOrderPhaseCorrection
## It lasted 0.719 s user time, 0.299 s system time and 1.031 s elapsed time.
```

```
# Représentation graphique des spectres
if(DrawSpectra==T)  Draw(Spectrum_data_ZOPC[WhichSpectra,Raw_Spec_window],
                         type.draw = "signal",  num.stacked = NumStack,
                         main = "ZeroOrderPhaseCorrection", xlab = "ppm")
```

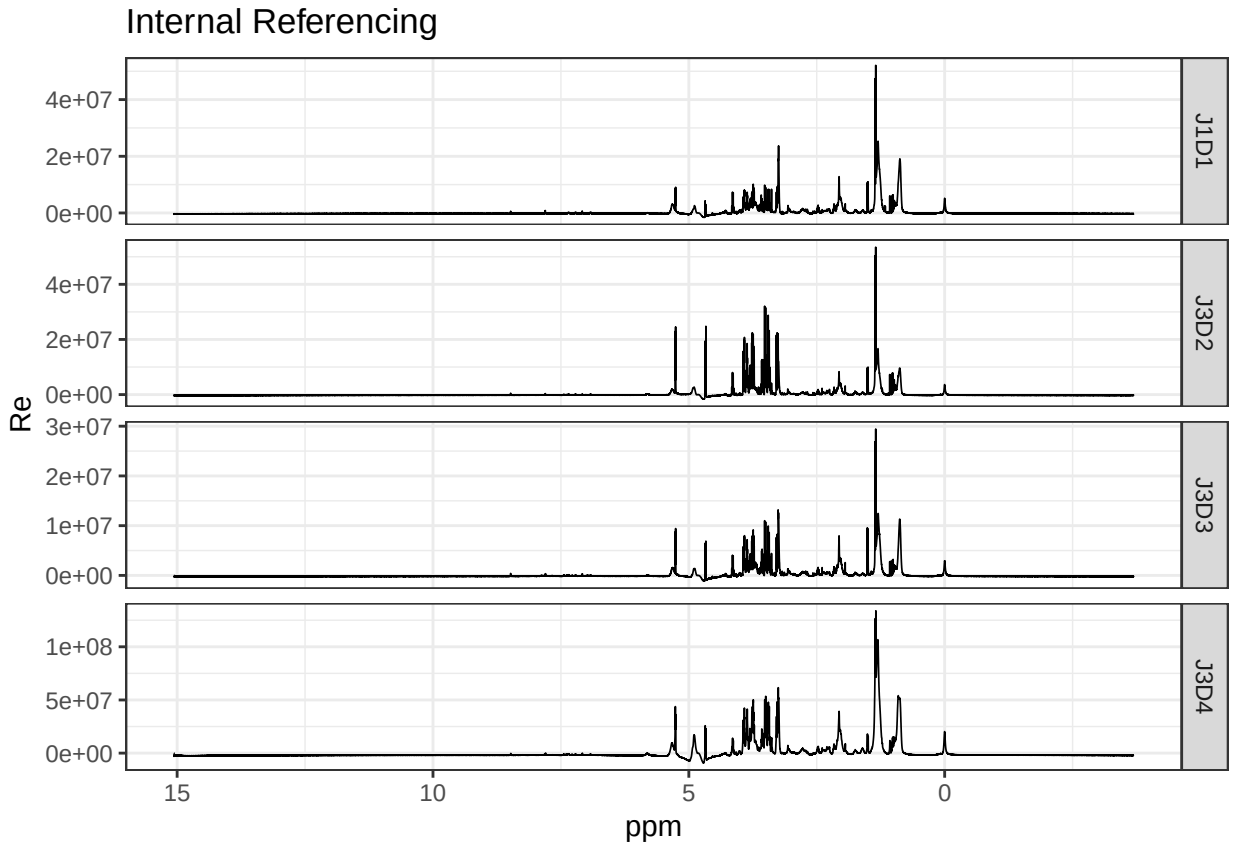


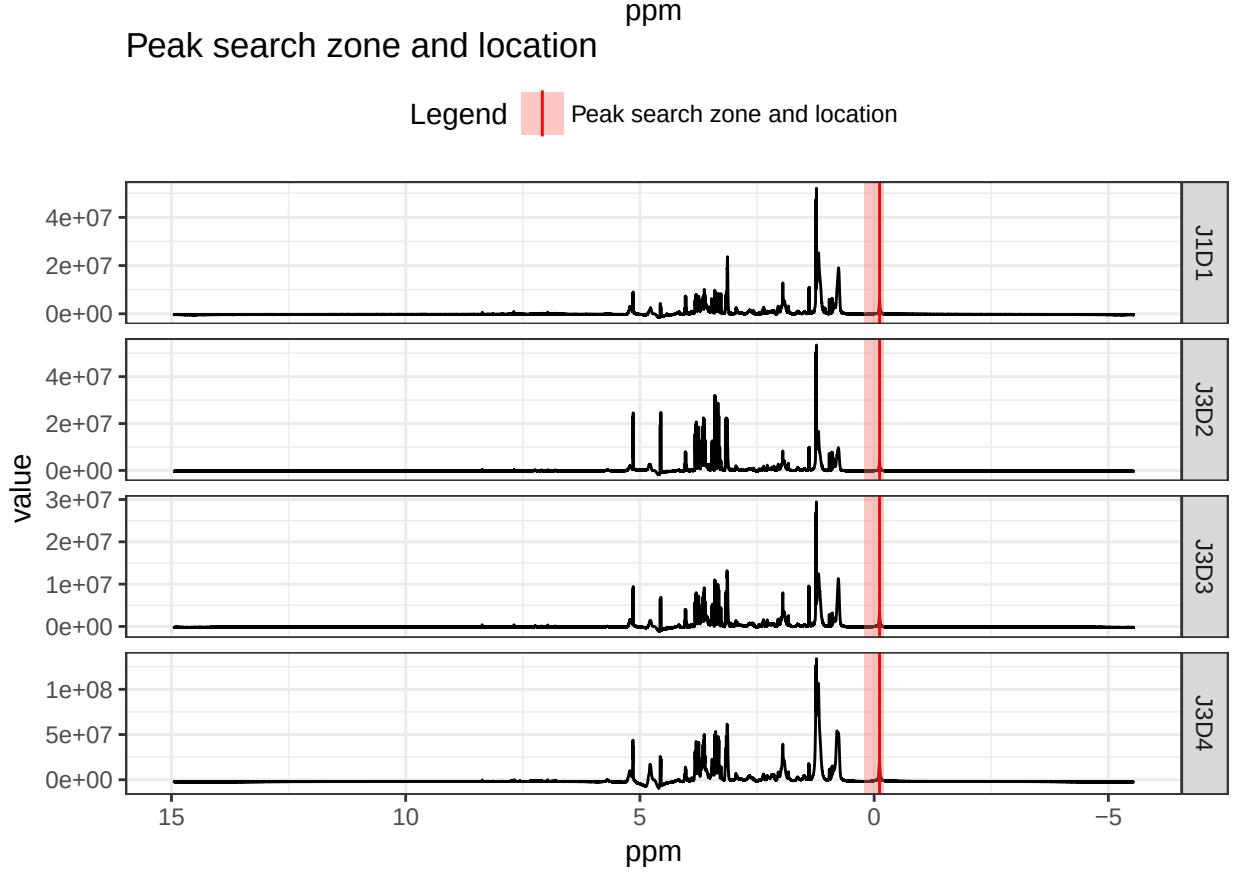
Alignement par rapport au pic de référence (le TMSP)

```
# InternalReferencing
res_IR <- InternalReferencing(Spectrum_data_ZOPC, Fid_info,
                              rowindex_graph = WhichSpectra)
```

```
## Begin InternalReferencing
## End InternalReferencing
## It lasted 0.261 s user time, 0.065 s system time and 0.333 s elapsed time.
```

```
# Extraire la matrice spectrale
Spectrum_data_IR <- res_IR$Spectrum_data



# Représentation graphique des spectres
if(DrawSpectra == T)  {
  Draw(Spectrum_data_IR[WhichSpectra,Raw_Spec_window],
       type.draw = "signal", num.stacked = NumStack,
       main = "Internal Referencing", xlab = "ppm")

  res_IR$plots[[1]] # graphe donné en sortie de la fonction InternalReferencing
}
```


Internal Referencing
Re
4e+07
2e+07
0e+00
3e+07
1e+07
1e+08
5e+07
J1D1
J3D2
J3D3
J3D4
15
10
5
0
ppm

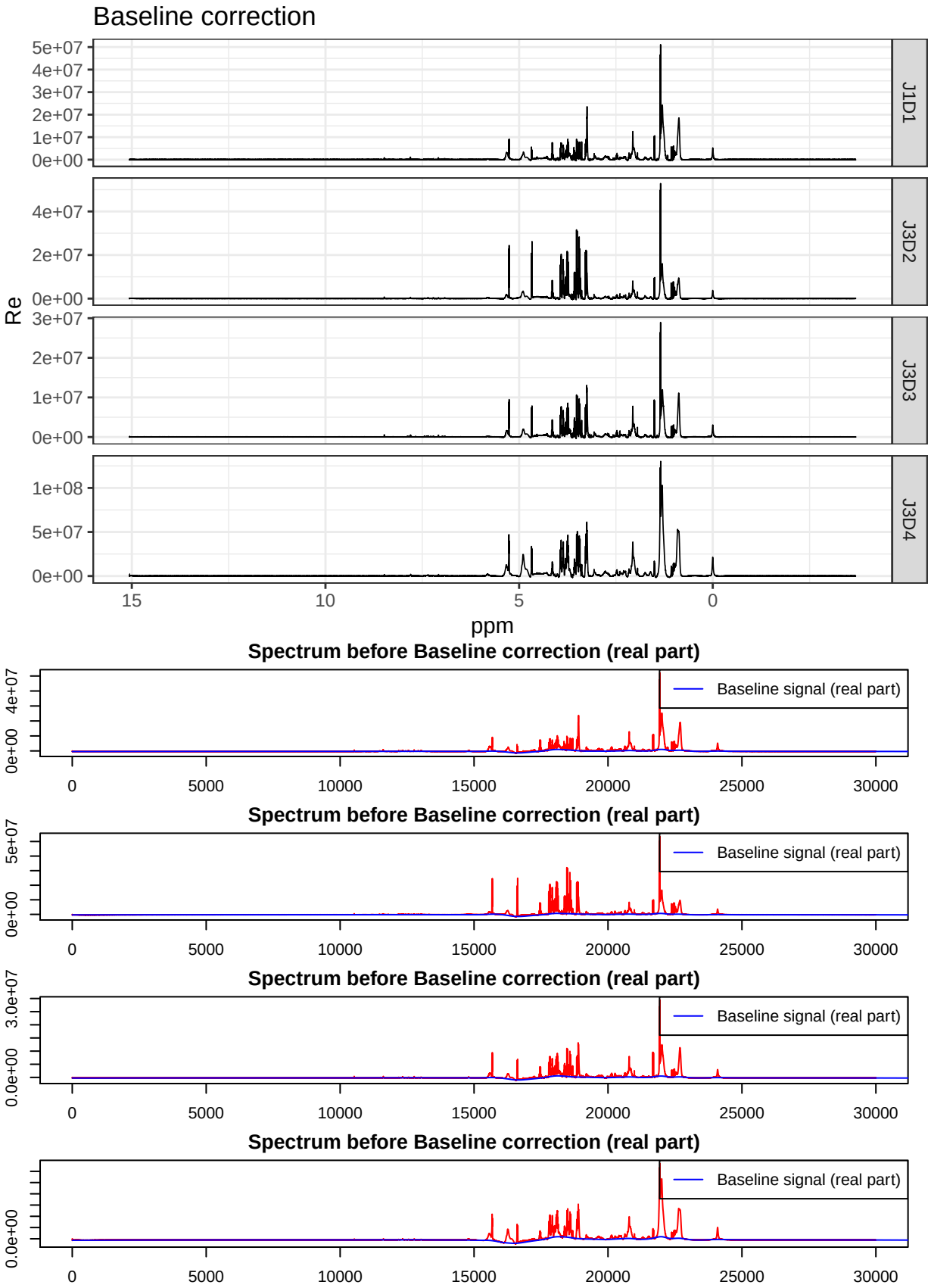

Peak search zone and location
Legend
Peak search zone and location
value
4e+07
2e+07
0e+00
3e+07
1e+07
1e+08
5e+07
J1D1
J3D2
J3D3
J3D4
15
10
5
0
−5
ppm

Baseline correction

```
res_BC <- BaselineCorrection(Spectrum_data_IR, lambda.bc = 1e+08,
                             p.bc = 0.01, returnBaseline = TRUE)
```

```
## Begin BaselineCorrection
## End BaselineCorrection
## It lasted 0.818 s user time, 0.464 s system time and 1.339 s elapsed time.
```

```
# Extraire la matrice spectrale
Spectrum_data_BC <- res_BC$Spectrum_data

# Extraire les Baselines estimées
Baseline <- t(res_BC$Baseline)

# Représentation graphique des spectres
if(DrawSpectra==T)  {
  Draw(Spectrum_data_BC[WhichSpectra, Raw_Spec_window],
                          type.draw = "signal", num.stacked = NumStack,
                          main="Baseline correction", xlab = "ppm")

  par(mfrow=c(4,1), mar=c(2,2,2,2))
   for (i in 1:length(WhichSpectra))  {
     plot(Re(Spectrum_data_IR[i,Raw_Spec_window]), type = "l", col = "red",
          main = "Spectrum before Baseline correction (real part)",
          xlab = "index")
     lines(Baseline[i,], type = "l", col = "blue")
     legend("topright", legend = c("Baseline signal (real part)"),
            col="blue", lty=1)
   }
}
```

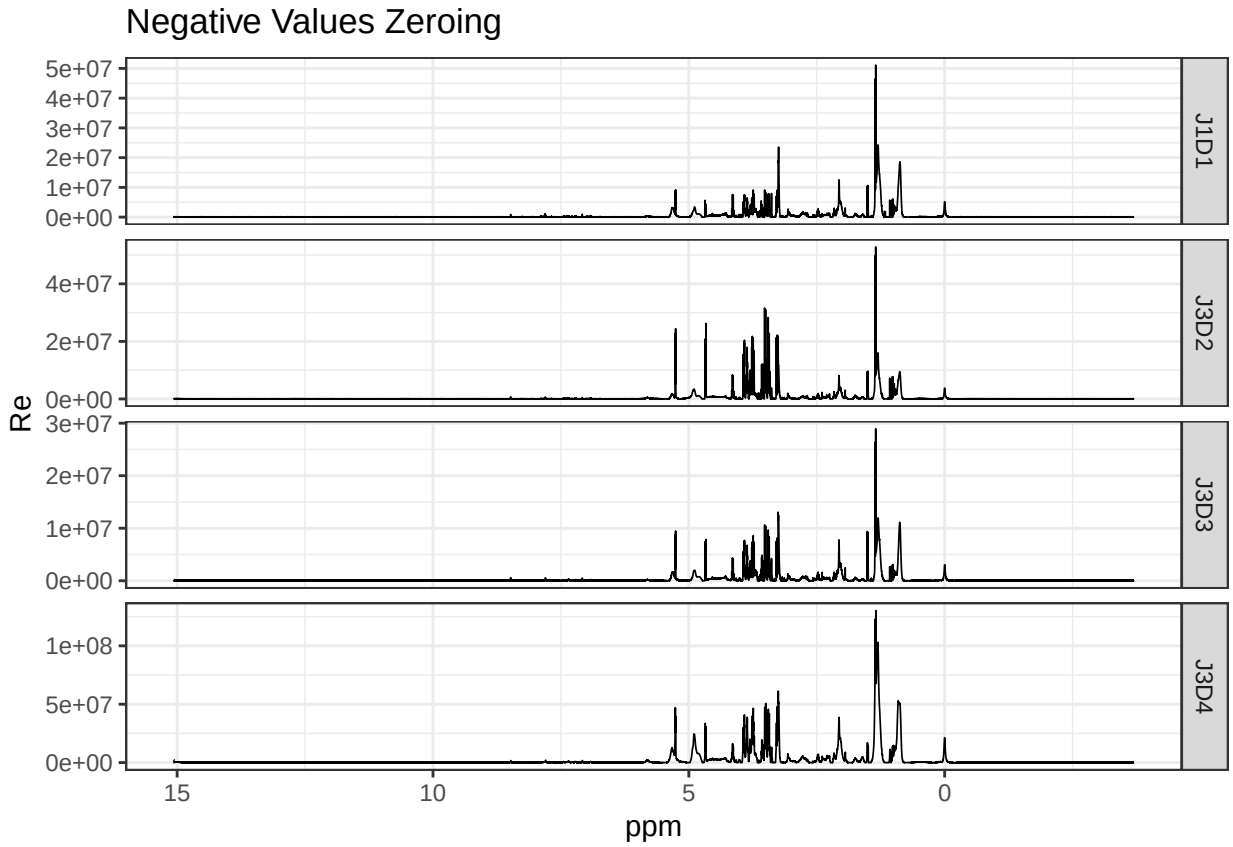

Baseline correction
J1D1
J3D2
J3D3
J3D4
Re
ppm
Spectrum before Baseline correction (real part)
Baseline signal (real part)

Suppression des valeurs négatives

```
# NegativeValuesZeroing
Spectrum_data_NVZ <- NegativeValuesZeroing(Spectrum_data_BC)
```

```
## Begin NegativeValuesZeroing
## End NegativeValuesZeroing
## It lasted 0.009 s user time, 0.006 s system time and 0.016 s elapsed time.
```

```
# Représentation graphique des spectres
if(DrawSpectra==T)  Draw(Spectrum_data_NVZ[WhichSpectra, Raw_Spec_window],
                         type.draw = "signal", num.stacked = NumStack,
                         main = "Negative Values Zeroing", xlab = "ppm")
```

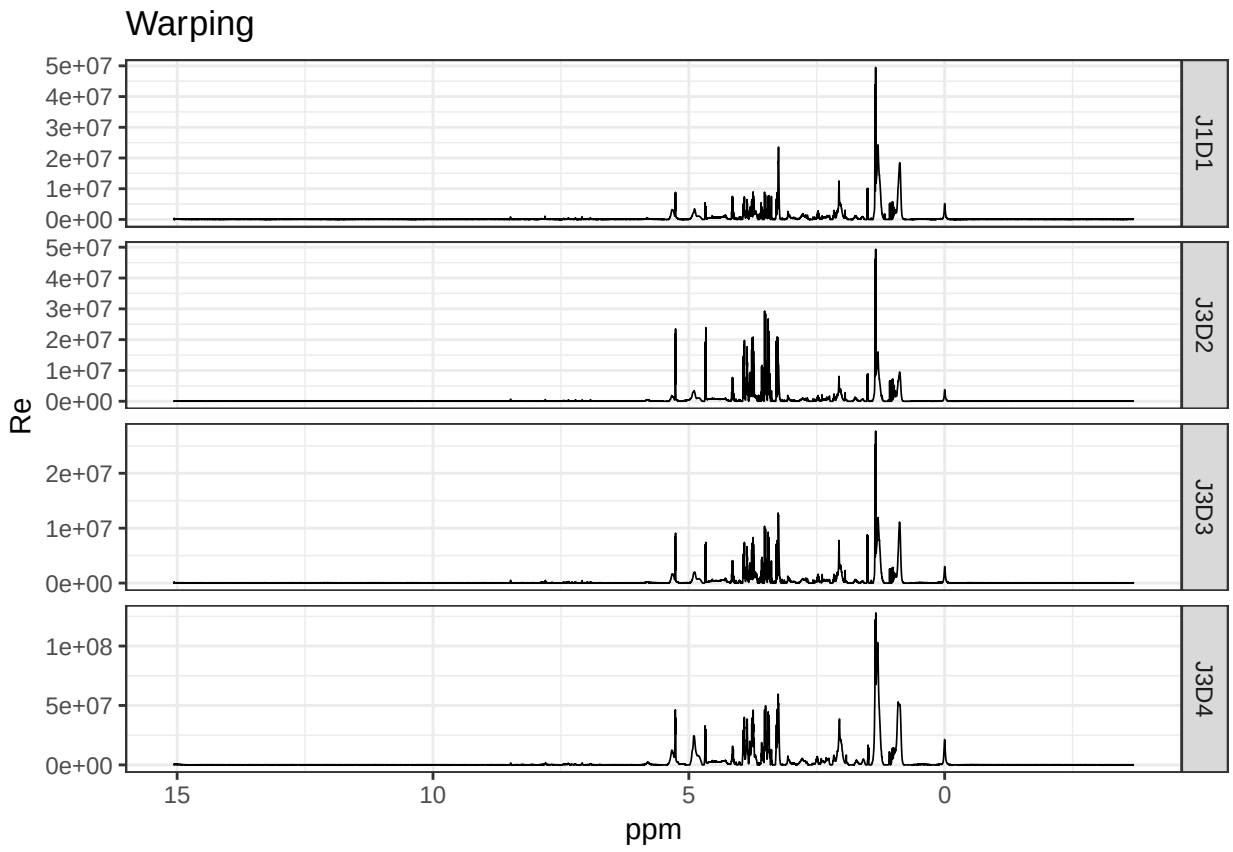


Alignement - Warping

```
if (do_Warping){
  # Warping
  Spectrum_data_beforeWW <- Warping(Spectrum_data_NVZ,
                                    reference.choice = reference.choice)

  # Représentation graphique des spectres
  if(DrawSpectra==T)  Draw(Spectrum_data_beforeWW[WhichSpectra, Raw_Spec_window],
                           type.draw = "signal",  num.stacked = NumStack,
                           main = "Warping", xlab = "ppm")
```

```
} else {Spectrum_data_beforeWW <- Spectrum_data_NVZ}
```

```
## Begin Warping
## Begin Normalization
## End Normalization
## It lasted 0.03 s user time, 0.001 s system time and 0.031 s elapsed time.
## End Warping
## It lasted 54.691 s user time, 14.12 s system time and 70.459 s elapsed time.
```



Window selection

```
# Window selection
 Spectrum_data_WS <- WindowSelection(Spectrum_data_beforeWW,
                                     from.ws = 10, to.ws = 0.2)
```

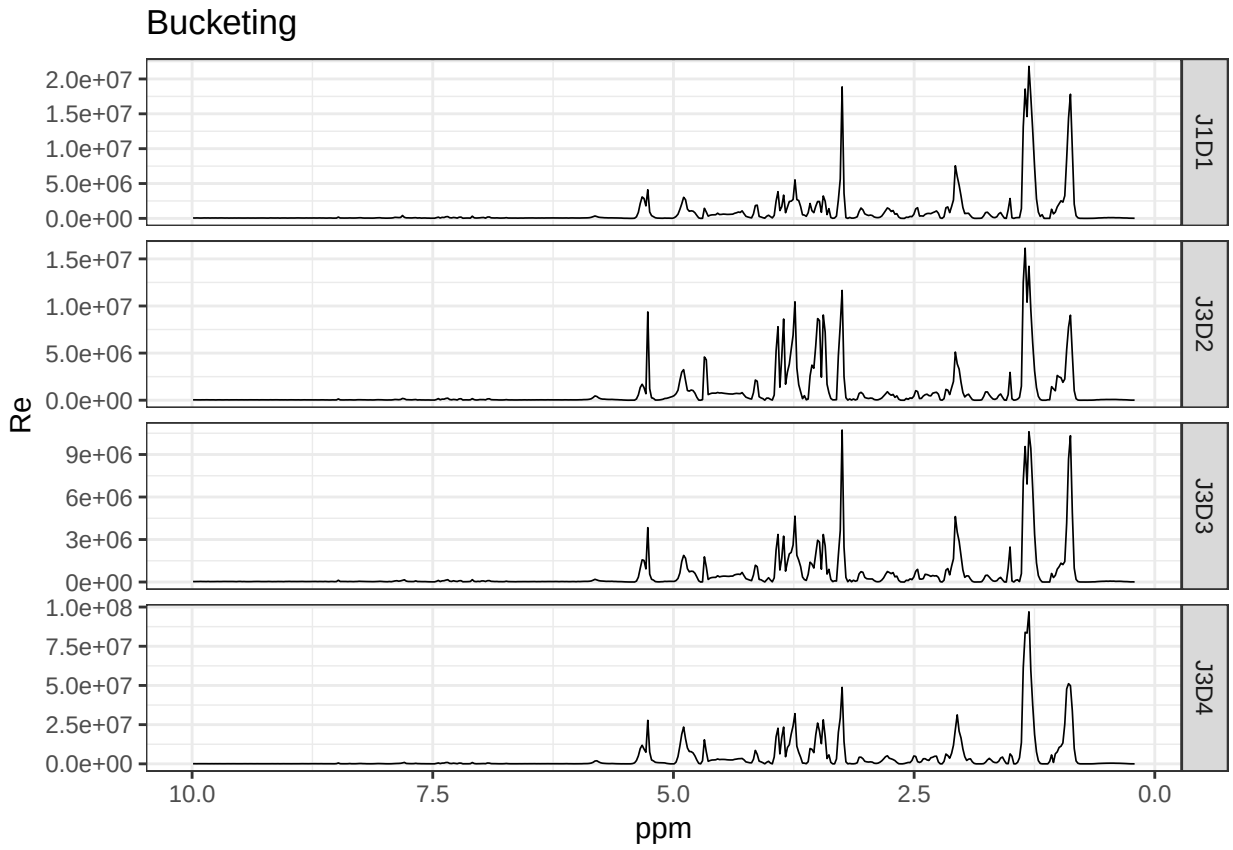
```
## Begin WindowSelection
## End WindowSelection
## It lasted 0.015 s user time, 0.007 s system time and 0.023 s elapsed time.
```

Bucketing

```
# Bucketing avec le Window selection intégré
Spectrum_data_B <- Bucketing(Spectrum_data_WS, intmeth = "t", mb = mb)
```

```
## Begin Bucketing
## PPM range of the bucketted spectral matrix: 9.99967 0.21967
## PPM width between 2 buckets: 0.01960
## End Bucketing
## It lasted 0.121 s user time, 0.013 s system time and 0.135 s elapsed time.
```

```
# Représentation graphique des spectres
if(DrawSpectra==T) Draw(Spectrum_data_B[WhichSpectra,],
                        type.draw = "signal", num.stacked = NumStack,
                        main = "Bucketing", xlab = "ppm")
```

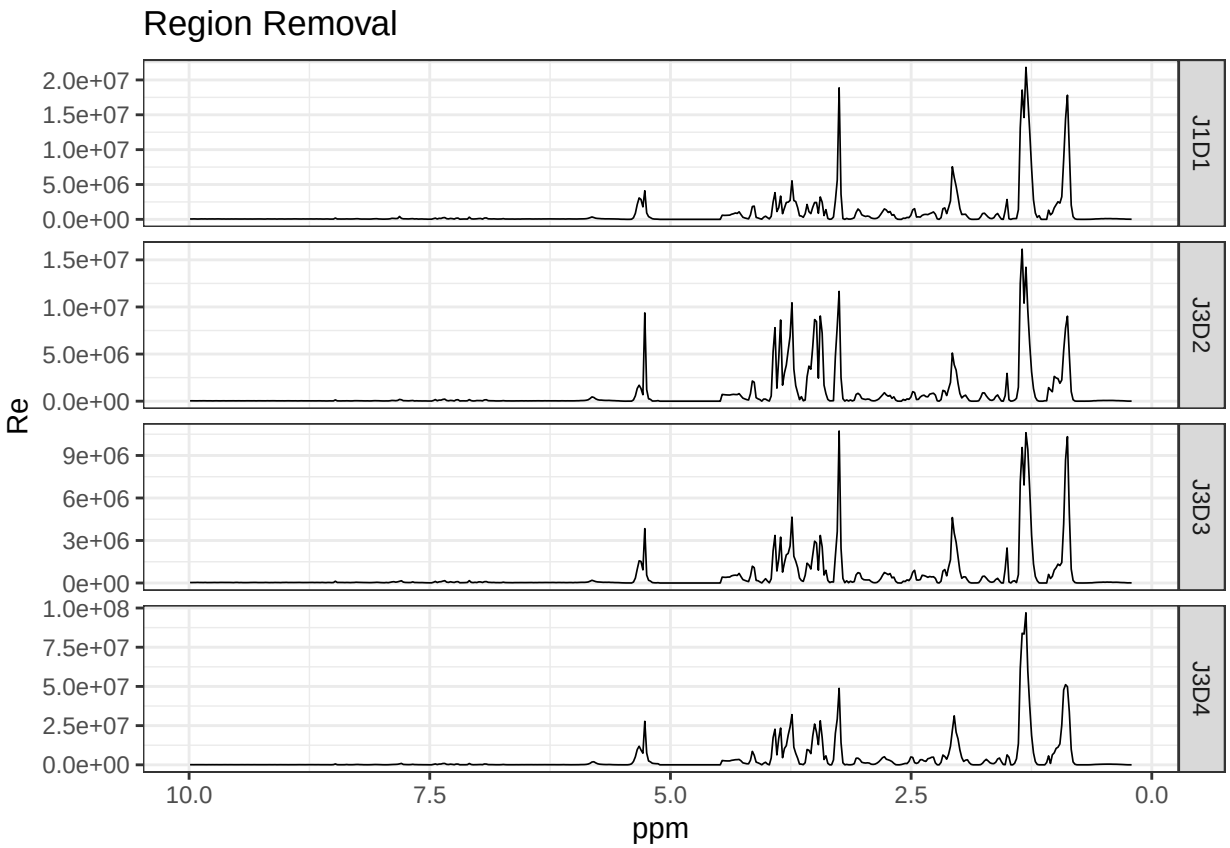


Suppression des régions non informatives

```
# RegionRemoval
Spectrum_data_RR <-  RegionRemoval(Spectrum_data_B, typeofspectra = typeofspectra)
```

```
## Begin RegionRemoval
## End RegionRemoval
## It lasted 0 s user time, 0 s system time and 0 s elapsed time.
```

```
# Représentation graphique des spectres
if(DrawSpectra==T) Draw(Spectrum_data_RR[WhichSpectra,], type.draw = "signal",
                        num.stacked = NumStack, main= "Region Removal", xlab = "ppm")
```



Agregation de zones

```
# ZoneAggregation
if(do_ZoneAggreg == TRUE) {
Spectrum_data_beforeNorm <- ZoneAggregation(Spectrum_data_RR,
                                fromto.za = fromto.za)

# Représentation graphique des spectres
if(DrawSpectra==T) Draw(Spectrum_data[WhichSpectra,], type.draw = "signal",
                        num.stacked = NumStack, main = "Zone Aggregation", xlab = "ppm")

} else {Spectrum_data_beforeNorm <- Spectrum_data_RR}
```

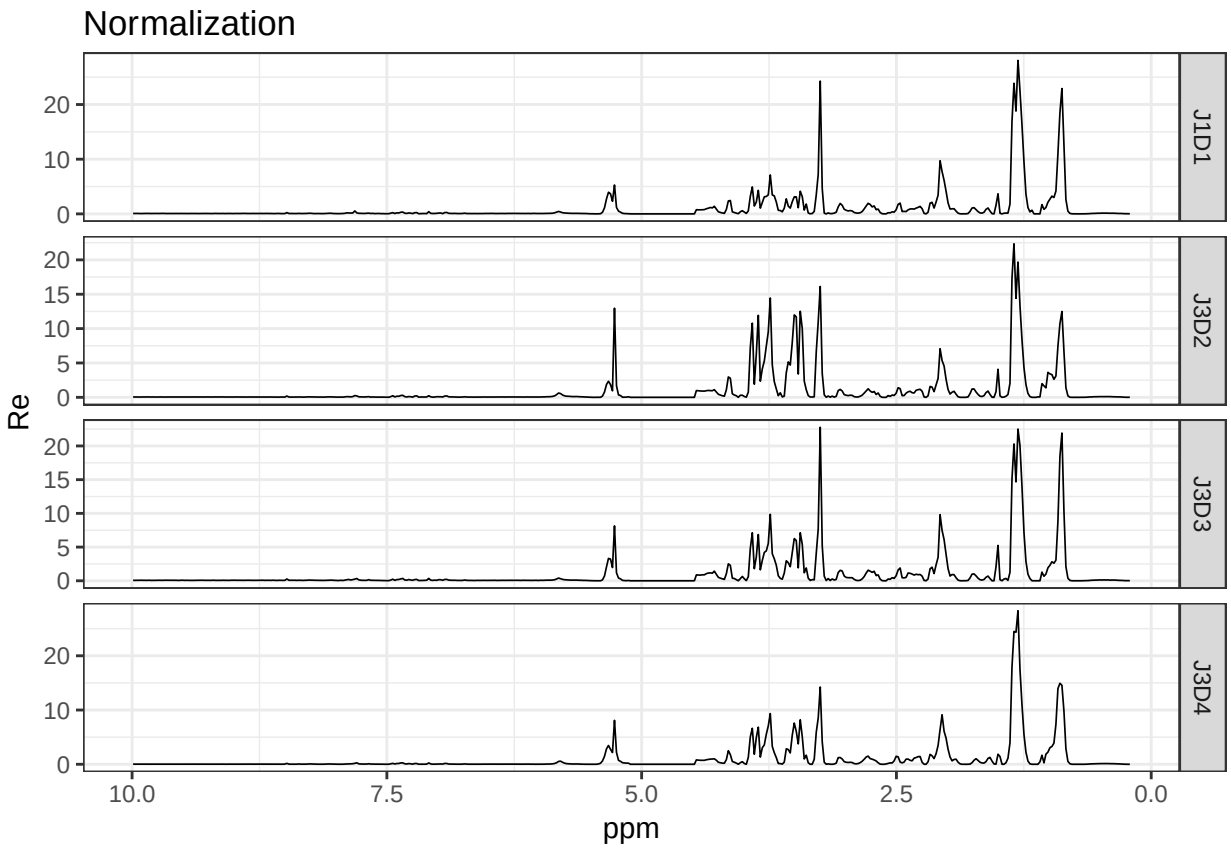
Normalisation

```
# Normalization
Spectrum_data_N <- Normalization(Spectrum_data_beforeNorm, type.norm = type.norm)
```

```
## Begin Normalization
## End Normalization
## It lasted 0.001 s user time, 0 s system time and 0 s elapsed time.
```

```
# Représentation graphique des spectres
if(DrawSpectra==T) Draw(Spectrum_data_N[WhichSpectra,], type.draw = "signal",
```

```
                    num.stacked = NumStack, main = "Normalization", xlab = "ppm")
```



Exportation des résultats

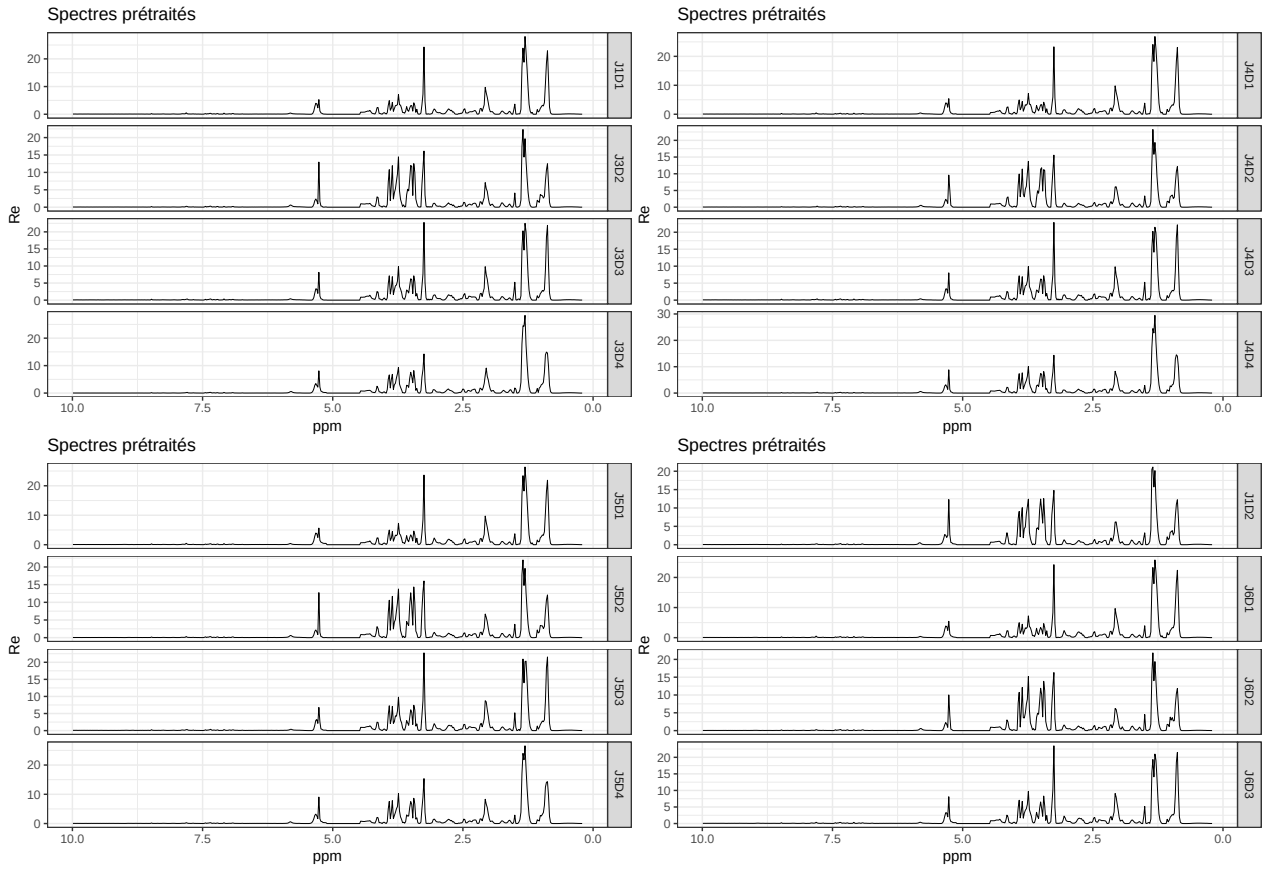
```
## Sauvegarde des spectres traités en fichiers .Rdata (format R) et .csv

# extraire la partie réelle de la matrice spectrale
Re_Spectrum_data <- Re(Spectrum_data_N)

if (save_RData == TRUE) {
  save(Re_Spectrum_data, group_HS, Fid_info,
       file=file.path(out.path, paste0(dataname,".RData")))
}

if (export_csv == TRUE) {
  # separateur de champs: ";" et séparateur décimal: "."
  utils::write.table(Re_Spectrum_data,
                     file = file.path(out.path,paste0(dataname,"_Spectra.csv")),
                     sep=";", dec=".", row.names = TRUE, col.names = NA)
  utils::write.table(Fid_info,
                     file = file.path(out.path, paste0(dataname,"_FidInfo.csv")),
                     sep=" ;", dec=".")
}
```

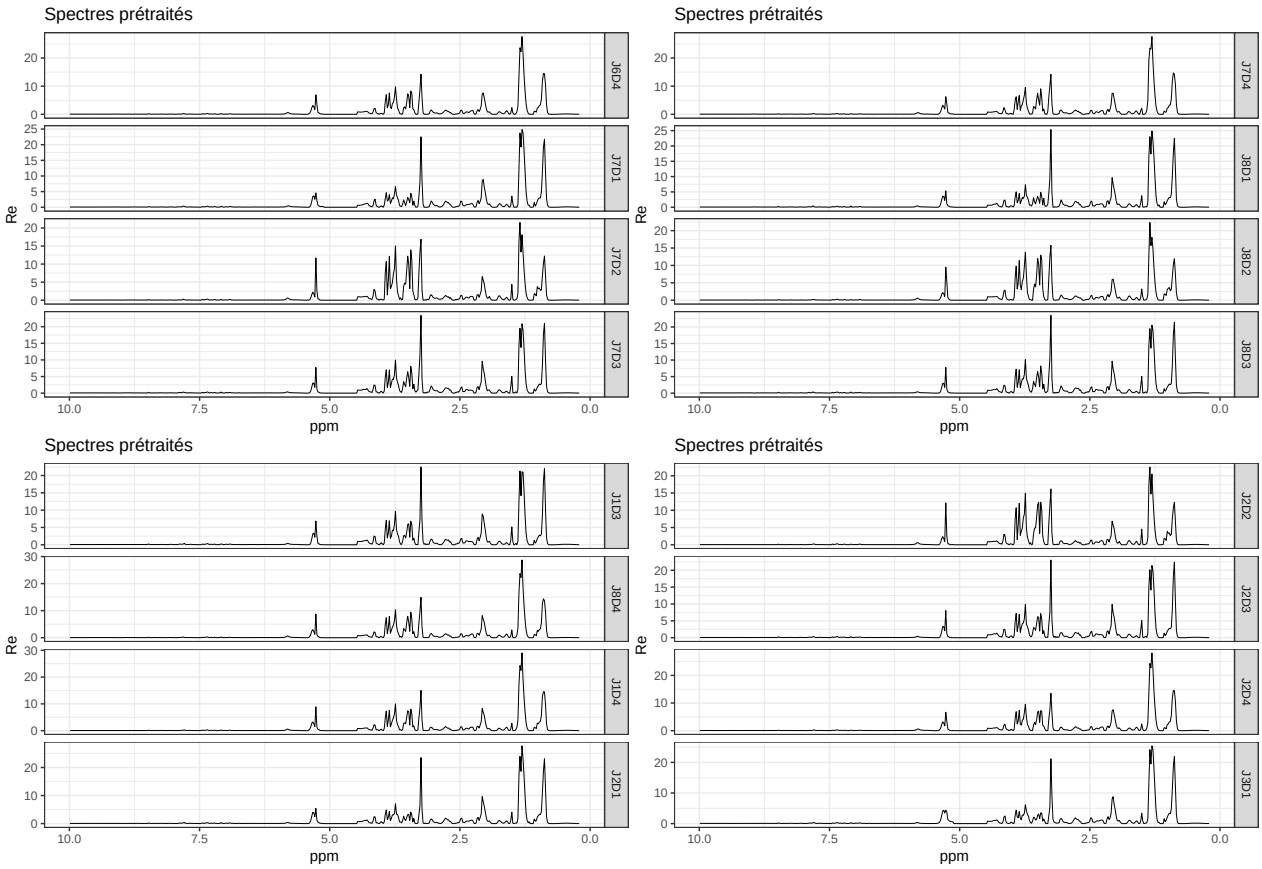
Utilisation de la fonction PreprocessingChain

```
?PreprocessingChain
res_PreprocessingChain <- PreprocessingChain(Fid_data = NULL, Fid_info = NULL,
                                             data.path = file.path(data.path,
                                                                   dataname),
                         readFids = TRUE, groupDelayCorr = TRUE,
                         solventSuppression = TRUE, apodization = TRUE,
                        fourierTransform = TRUE, zeroOrderPhaseCorr = TRUE,
                        internalReferencing = TRUE, baselineCorrection = TRUE,
                        negativeValues0 = TRUE, warping = TRUE,
                        windowSelection = TRUE, bucketing = TRUE,
                        regionRemoval = TRUE, zoneAggregation = FALSE,
                        normalization = TRUE,
                        export = FALSE, type.norm = "mean",
                        reference.choice = "before",
                        typeofspectra = "serum")
```

Visualisation des spectres finaux

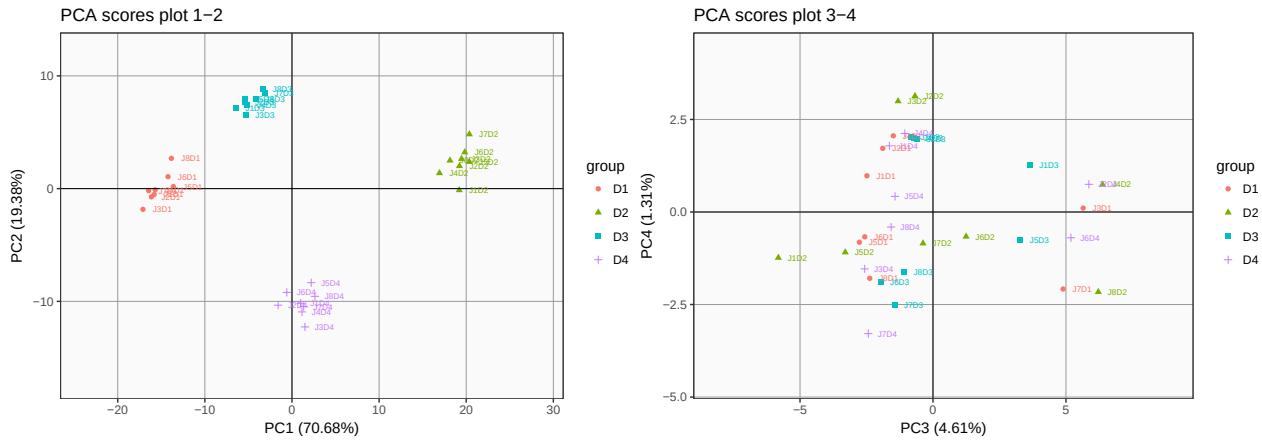
```
## Visualisation des spectres finaux
Draw(Re_Spectrum_data, type.draw = "signal", num.stacked = 4,
     main = "Spectres prétraités", xlab = "ppm")
```





PCA sur les spectres finaux

```
# PCA scores
Draw(Re_Spectrum_data, type.draw = "pca", type = "scores", Class = group,
     axes = c(1:2), main = "PCA scores plot 1-2")
Draw(Re_Spectrum_data, type.draw = "pca", type = "scores", Class = group,
     axes = c(3:4), main = "PCA scores plot 3-4")
```




```
# PCA loadings
Draw(Re_Spectrum_data, type.draw = "pca", type = "loadings", axes = c(1:2),
```

```
    main = "PCA loadings plot 1-2", xlab = "ppm")
```