Bioconductor BRAIN Package Vignette

Piotr Dittwald, Jürgen Claesen, Dirk Valkenborg

May 1, 2024

The R package BRAIN (**B**affling **R**ecursive **A**lgorithm for **I**sotopic distributio**N** calculations) provides a computation- and memory-efficient method to calculate the *aggregated isotopic distribution* of peptides and proteins.

1 Introduction

The *isotopic distribution* is an important, but often forgotten, concept in the field of biomolecular mass spectrometry. Yet, it is particularly useful for the interpretation of the complex patterns observed in mass spectral data. The isotopic distribution reflects the probabilities of occurrence of different isotopic variants of a molecule and is visualized in the mass spectrum by the relative heights of the series of peaks related to the molecule. Ignoring the small deviations of the masses from integer values, *aggregated isotopic variants* of a molecule, with masses differing approximately by 1Da, can be used to calculate the *aggregated isotopic distribution*. Prior knowledge about this distribution can be used to develop strategies for searching particular profiles in the spectra and, hence, for efficient processing of the spectral information. Computing the isotopic distribution for small molecules is relatively easy for small molecules, however the complexity of this computation increases drastically with the size of the molecule. Therefore, the calculation of the (aggregated) isotopic distribution for larger molecules can be sub-optimal or even problematic due to a combinatorial explosion of terms. We presented an alternate, computation- and memory-efficient method to calculate the probabilities of occurrence and exact center-masses of the aggregated isotopic distribution of a molecule in Claesen *et al.*, 2012.

2 Overview of the BRAIN package

This package has five functions:

1. `useBRAIN`, which computes the probabilities of aggregated isotopic variants and their center-masses for biomolecules composed out of carbon, hydrogen, oxygen, nitrogen and sulfur. Additionally the function returns the average mass of the biomolecule;

2. `calculateMonoisotopicMass`, which computes the theoretical monoisotopic mass of biomolecules composed out of carbon, hydrogen, oxygen, nitrogen and sulfur;

3. `calculateAverageMass`, which computes the theoretical average mass of biomolecules composed out of carbon, hydrogen, oxygen, nitrogen and sulfur;

4. `calculateIsotopicProbabilities`, which computes the isotopic probabilities of the aggregated isotopic variants for biomolecules composed out of carbon, hydrogen, oxygen, nitrogen and sulfur;

5. `calculateNrPeaks`, which computes heuristically the required number of consecutive aggregated isotopic variants (starting from the monoisotopic mass), with a minimum of five;

6. `getAtomsFromSeq`, which creates atomic composition from a given aminoacid sequence;

7. `useBRAIN2`, which computes the probabilities of aggregated isotopic variants for biomolecules composed out of carbon, hydrogen, oxygen, nitrogen and sulfur. This function allows for heuristics aimed for time savings in computations.

3 Aggregated isotopic distribution calculation

Here, we will show how to use the package with angiotensine II as an example. Its atomic composition is $C_{50}H_{71}N_{13}O_{12}$.

3.1 Theoretical monoisotopic mass

The theoretical monoisotopic mass of a biomolecule with atomic composition $C_vH_wN_xO_yS_z$ is calculated as follows:

$$Monoisotopic\,mass = vM_{C_{12}} + wM_{H_1} + xM_{N_{14}} + yM_{O_{16}} + zM_{S_{32}} \quad (1)$$

We can calculate the monoisotopic mass for angiotensine II with the BRAIN-package as follows:

```
> #angiotensineII
> angiotensineII <- list(C=50,H=71,N=13,O=12)
> monoisotopicMassAngiotensineII <- calculateMonoisotopicMass(aC = angiotensineII)
> monoisotopicMassAngiotensineII

[1] 1045.535

> #human dynein heavy chain
> humandynein <- list(C=23832,H=37816,N=6528,O=7031,S=170)
> monoisotopicMassHumandynein <- calculateMonoisotopicMass(aC = humandynein)
> monoisotopicMassHumandynein

[1] 533403.5
```

3.2 Theoretical average mass

The theoretical average mass of the same biomolecule is calculated as follows:

$$\begin{aligned} \textit{Average mass} &= vM_{C_{12}} \times P_{C_{12}} + vM_{C_{13}} \times P_{C_{13}} \\ &+ wM_{H_1} \times P_{H_1} + wM_{H_2} \times P_{H_2} \\ &+ xM_{N_{14}} \times P_{N_{14}} + xM_{N_{15}} \times P_{N_{15}} \\ &+ yM_{O_{16}} \times P_{O_{16}} + yM_{O_{17}} \times P_{O_{17}} + yM_{O_{18}} \times P_{O_{18}} \\ &+ zM_{S_{32}} \times P_{S_{32}} + zM_{S_{33}} \times P_{S_{33}} + zM_{S_{34}} \times P_{S_{34}} + zM_{S_{36}} \times P_{S_{36}} \end{aligned}$$

We can calculate the average mass for angiotensine II with the BRAIN-package with:

```
> #angiotensineII
> averageMassAngiotensineII <- calculateAverageMass(aC = angiotensineII)
> averageMassAngiotensineII

[1] 1046.181

> #humandynein
> averageMassHumandynein <- calculateAverageMass(aC = humandynein)
> averageMassHumandynein

[1] 533735.2
```

3.3 Number of requested aggregated isotopic variants

The calculation of the aggregated isotopic distribution can be stopped when the required number of aggregated isotopic variants has been reached. The latter number can be heuristically obtained. For this purpose, we have added the function *calculateNrPeaks*, which uses following rule of thumb: the difference between the theoretical monoisotopic mass and the theoretical average mass is computed and multiplied by two. Subsequently, the obtained number is rounded to the nearest integer greater than or equal to the multiplied difference. For small molecules, the minimal number of requested variants is five.

```
> #angiotensineII
> nrPeaksAngiotensineII <- calculateNrPeaks(aC = angiotensineII)
> nrPeaksAngiotensineII

[1] 5

> #human dynein heavy chain
> nrPeaksHumandynein <- calculateNrPeaks(aC = humandynein)
> nrPeaksHumandynein

[1] 664
```

3.4 Isotopic probabilities of the aggregated isotopic variants

The function *calculateIsotopicProbabilities* returns the isotopic probabilities of the aggregated isotopic variants for molecules with carbon, hydrogen, oxygen, nitrogen and sulfur as atomic building blocks. The function will stop computing as soon as the required number of aggregated variants has been reached (default), when the user-defined coverage has been reached, or when the computed occurrence probabilities become smaller than the defined /textbfabundantEstim.

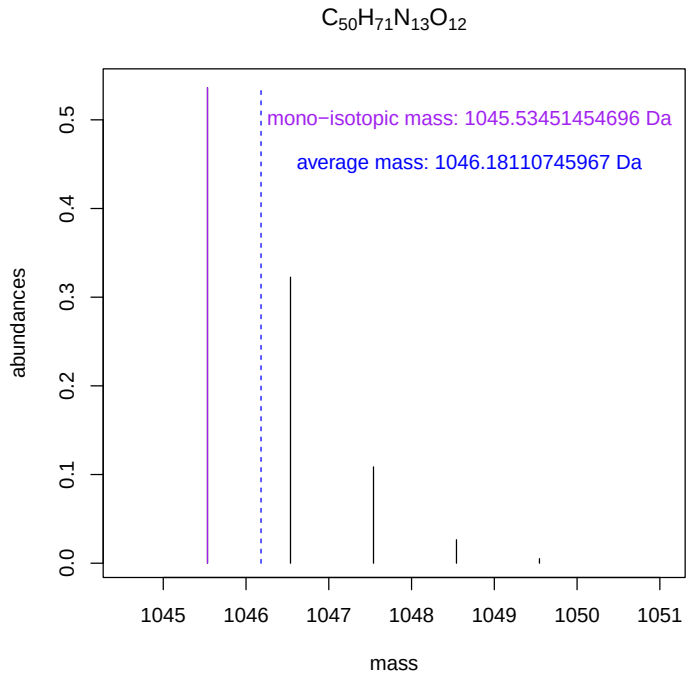
```
> #angiotensineII
>
> #with default options
> prob1 <- calculateIsotopicProbabilities(aC = angiotensineII)
> print(length(prob1))

[1] 5

> #with user defined number of requested aggregated isotopic variants
> prob2 <- calculateIsotopicProbabilities(aC = angiotensineII, nrPeaks=20)
> print(length(prob2))

[1] 20

> #with user-defined coverage as stopping criterium
> prob3 <- calculateIsotopicProbabilities(aC = angiotensineII,
+ stopOption = "coverage", coverage = 0.99)
> print(length(prob3))

[1] 4

> #with user-defined abundantEstim as stopping criterium
> prob4 <- calculateIsotopicProbabilities(aC = angiotensineII,
+ stopOption = "abundantEstim", abundantEstim = 10)
> print(length(prob4))

[1] 5

> #human dynein heavy chain
> prob1 <- calculateIsotopicProbabilities(aC = humandynein)
> print(length(prob1))

[1] 664

> prob2 <- calculateIsotopicProbabilities(aC = humandynein, nrPeaks=300)
> print(length(prob2))

[1] 300

> prob3 <- calculateIsotopicProbabilities(aC = humandynein,
+ stopOption = "coverage", coverage = 0.99)
> print(length(prob3))
```

```
[1] 378

> prob4 <- calculateIsotopicProbabilities(aC = humandynein,
+ stopOption = "abundantEstim", abundantEstim = 150)
> print(length(prob4))

[1] 481
```

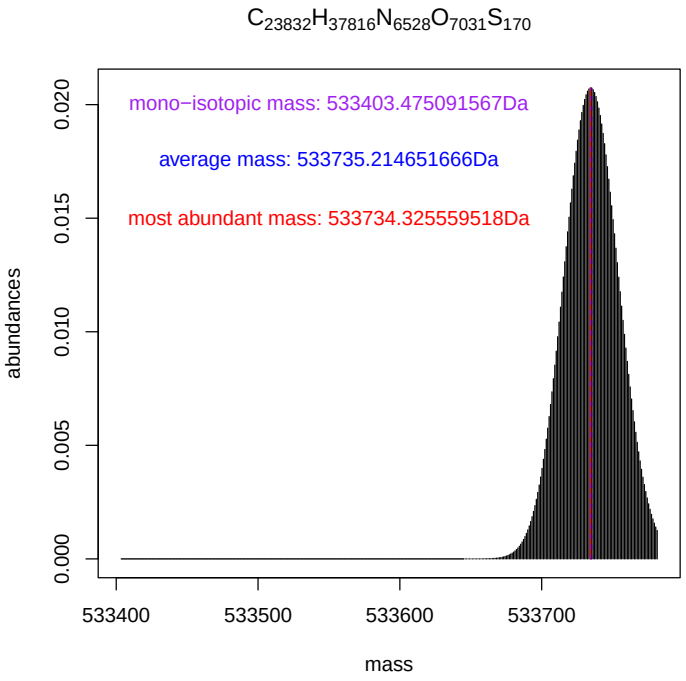
3.5 Global function

The functions described above are incorporated in the global function *useBRAIN*.

```
> #angiotensineII
> headr <- expression(paste(C[50], H[71], N[13],O[12]))
> #with default options
> res <- useBRAIN(aC= angiotensineII)
> plot(res$masses,res$isoDistr,xlab="mass",ylab="abundances", type="h",
+ xlim=c(min(res$masses)-1,max(res$masses)+1.5))
> title(headr)
> labelMono <- paste("mono-isotopic mass:", res$monoisotopicMass, "Da", sep=" ")
> labelAvg <- paste("average mass:", res$avgMass, "Da", sep=" ")
> text(x=1048.7, y=0.5, labelMono, col="purple")
> text(x=1048.7, y=0.45, labelAvg, col="blue")
> lines(x=rep(res$monoisotopicMass[1],2),y=c(0,res$isoDistr[1]), col = "purple")
> lines(x=rep(res$avgMass,2),y=c(0,max(res$isoDistr)), col = "blue", lty=2)
```



```
> #human dynein heavy chain
> headr <- expression(paste(C[23832], H[37816], N[6528],O[7031], S[170]))
> res <- useBRAIN(aC=humandynein, stopOption="coverage", coverage=0.99)
> plot(res$masses,res$isoDistr,xlab="mass",ylab="abundances", type="h",
+ xlim=c(min(res$masses)-1,max(res$masses)+1))
> title(headr)
> labelMono <- paste("mono-isotopic mass: ", res$monoisotopicMass, "Da", sep="")
> labelAvg <- paste("average mass: ", res$avgMass, "Da", sep="")
> mostAbundant <- res$masses[which.max(res$isoDistr)]
> labelAbundant <- paste("most abundant mass: ", mostAbundant, "Da", sep="")
> text(x=533550, y=0.02, labelMono, col="purple")
> text(x=533550, y=0.0175, labelAvg, col="blue")
> text(x=533550, y=0.015, labelAbundant, col="red")
> lines(x=rep(res$avgMass,2),y=c(0,max(res$isoDistr)), col = "blue", lty=2)
> lines(x=rep(mostAbundant,2),y=c(0,max(res$isoDistr)), col = "red")
```



Another stopping criteria as for the function *calculateIsotopicDistribution* are available (corresponding plots not shown in this document).

```
> #with user defined number of requested aggregated isotopic variants
> res <- useBRAIN(aC = angiotensineII, nrPeaks = 20)
> plot(res$masses,res$isoDistr,xlab="mass",ylab="abundances", type="h",
+ xlim=c(min(res$masses)-1,max(res$masses)+1))
> title(headr)
> #with user defined coverage as stopping criterium
> res <- useBRAIN(aC = angiotensineII, stopOption = "coverage", coverage = 0.99)
> plot(res$masses,res$isoDistr,xlab="mass",ylab="abundances", type="h",
```

```
+ xlim=c(min(res$masses)-1,max(res$masses)+1))
> title(headr)
> #with user defined abundantEstim as stopping criterium
> res <- useBRAIN(aC = angiotensineII, stopOption = "abundantEstim", abundantEstim = 10)
> plot(res$masses,res$isoDistr,xlab="mass",ylab="abundances", type="h",
+ xlim=c(min(res$masses)-1,max(res$masses)+1))
> title(headr)
```

4 High-throughput calculation of the aggregated isotopic distribution and their exact center-masses

We used the Uniprot database as a case example for the high-throughput calculations. The data was downloaded at 28.11.2011 (release 2011_11) for query: `organism:"Homo sapiens (Human) [9606]" AND keyword:"Complete proteome [181]"` in UniProtKB (`http://www.uniprot.org/uniprot/?query=organism:9606+keyword:181`).

We downloaded data in a tab-delimited format and considered both reviewed (20,245) (UniProtKB/Swiss-Prot) and unreviewed (37,802) (UniProtKB/TrEMBL) entries. We only used the availably sequence information in this example.

```
> tabUniprot <- read.table('data/uniprot.tab', sep="\t", header=TRUE)
> tabUniprot$Sequence <- gsub(" ", "", tabUniprot$Sequence)
> howMany <- nrow(tabUniprot)
```

We will process only those proteins which are solely composed out of the 20 natural amino acides, i.e. those which return TRUE values from the test below (other proteins are ignored).

```
> AMINOS <- c("A", "R", "N", "D", "C", "E", "Q", "G", "H", "I", "L", "K",
+ "M", "F", "P", "S", "T", "W", "Y", "V")
> !is.na(sum(match(seqVector, AMINOS)))
```

To be able to use the function `useBRAIN` from the BRAIN package we need to change the amino acid sequences into chemical formulas containing the numbers of Carbon, Hydrogen, Nitrogen, Oxygen and Sulphur atoms. This may be done using the function `getAtomsFromSeq` changing the amino acid string into the list with corresponding atomic composition. Finally we run the following script to obtain `dfUniprot` data frame.

```
> nrPeaks = 1000
> howMany <- nrow(tabUniprot)
> dfUniprot <- data.frame()
> for (idx in 1:howMany){
+   seq <- as.character(tabUniprot[idx,"Sequence"])
+   seqVector <- strsplit(seq, split="")[[1]]
+   if (!is.na(sum(match(seqVector, AMINOS)))){
+   aC <- getAtomsFromSeq(seq)
+   res <- useBRAIN(aC = aC, nrPeaks = nrPeaks, stopOption = "abundantEstim",
+ abundantEstim = 10)
```

```
+   isoDistr <- res$isoDistr
+   masses <- res$masses
+   maxIdx <- which.max(isoDistr)
+   mostAbundantPeakMass <- masses[maxIdx]
+   monoMass <- res$monoisotopicMass
+   dfAtomicComp <- data.frame(C=aC[1], H=aC[2], N=aC[3], O=aC[4], S=aC[5])
+   singleDfUniprot <- data.frame(dfAtomicComp, monoMass=monoMass, maxIdx=maxIdx,
+ mostAbundantPeakMass=mostAbundantPeakMass
+   )
+   if (!is.na(monoMass)){ #for huge atomic configuration numerical problems may occur
+     dfUniprot <- rbind(dfUniprot, singleDfUniprot)
+   }
+   }
+ }
> write.table(unique(dfUniprot), "data/uniprotBRAIN.txt")
```

Execution of this code needed around 80 minutes on PC with two Intel(R) Core(TM)2 2.40GHz CPUs.

5 Predicting monoisotopic mass from most abundant peak mass

We obtain the data frame with the processed human proteins from Uniprot database (the processing procedure was described in Section 4). We may also check the number of rows.

```
> uniprotBRAIN <- read.table(system.file("extdata", "uniprotBRAIN.txt", package="BRAIN"))
> nrow(uniprotBRAIN)
```

```
[1] 57930
```

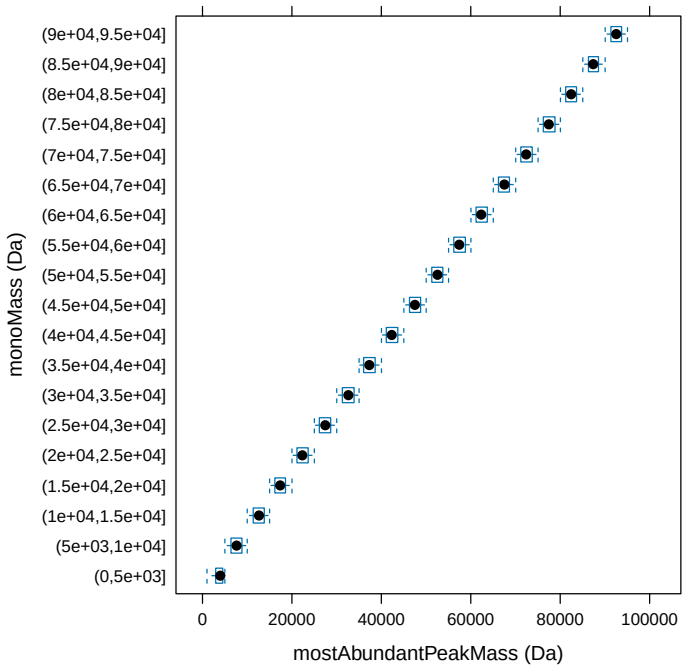
We only consider proteins with monoisotopic mass lower than 10^5 Da.

```
> uniprot10to5 <- uniprotBRAIN[uniprotBRAIN$monoMass < 10^5,]
> nrow(uniprot10to5)
```

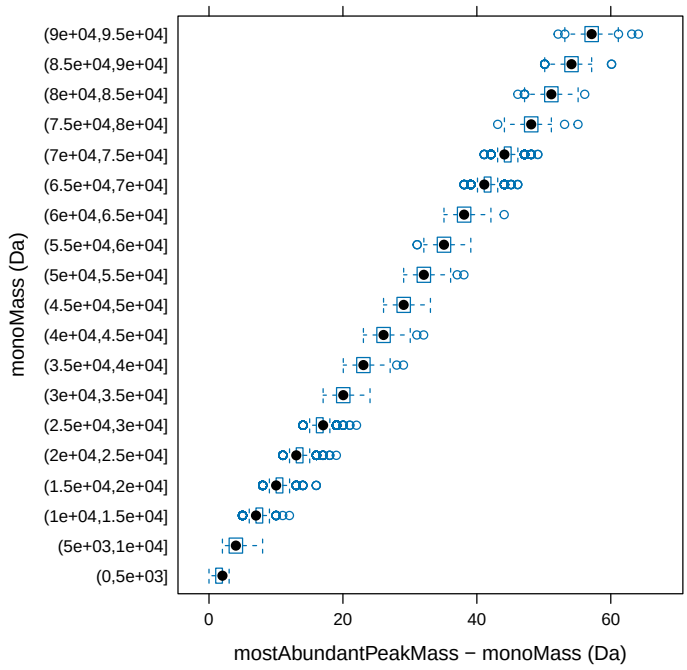
```
[1] 52589
```

We can observe a linear relationship between most abundant peak mass and monoisotopic mass for the considered data.

```
> library(lattice)
> mm <- uniprot10to5$monoMass
> mmMin <- floor(min(mm)) - 1
> mmMax <- ceiling(max(mm)) + 1
> sq <- seq(from=0, to=mmMax, by=5000)
> bw <- bwplot(cut(monoMass, sq) ~ mostAbundantPeakMass, data=uniprot10to5, ylab="monoMass
> plot(bw)
```




```
> bw2 <- bwplot(cut(monoMass, sq) ~ (mostAbundantPeakMass - monoMass),
+ data=uniprot10to5,  ylab="monoMass (Da)", xlab="mostAbundantPeakMass - monoMass (Da)")
> plot(bw2)
```



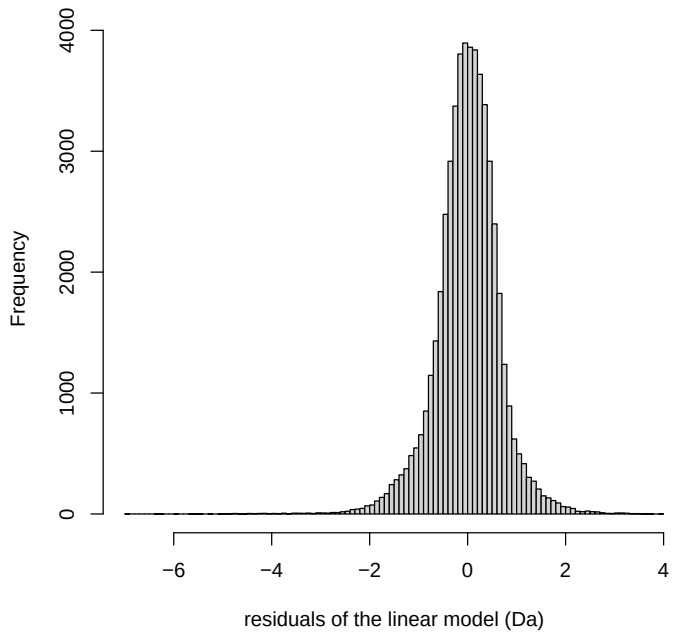
Following obtained linear relationship we build the linear model for predicting monoisotopic mass just by knowing most abundant peak mass.

```
> lmod <- lm(monoMass ~ mostAbundantPeakMass, data=uniprot10to5)
> summary(lmod)

Call:
lm(formula = monoMass ~ mostAbundantPeakMass, data = uniprot10to5)

Residuals:
    Min      1Q  Median      3Q     Max
-6.9328 -0.3419  0.0207  0.3780  3.9032

Coefficients:
                     Estimate Std. Error   t value Pr(>|t|)
(Intercept)          4.820e-01  4.929e-03 9.778e+01   <2e-16 ***
mostAbundantPeakMass 9.994e-01  1.220e-07 8.191e+06   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6556 on 52587 degrees of freedom
Multiple R-squared:      1,        Adjusted R-squared:      1
F-statistic: 6.71e+13 on 1 and 52587 DF,  p-value: < 2.2e-16
```

We can then further analyze the model using histogram which shows the residuals.

```
> icptLm <- lmod$coefficients[1]
> cffLm <- lmod$coefficients[2]
```

```
> expected <- icptLm + cffLm * uniprot10to5$mostAbundantPeakMass
> residuals <- uniprot10to5$monoMass - expected
> hist <- hist(residuals, seq(floor(min(residuals)),ceiling(max(residuals)), by=0.1),
+ main="", xlab="residuals of the linear model (Da)")
> plot(hist)
```



6 BRAIN 2.0 heuristics

As mentioned in *Dittwald P., Valkenborg D. BRAIN 2.0: time and memory complexity improvements in the algorithm for calculating the isotope distribution. JASMS, 2014*, it is possible to use several heuristics to make faster computations of isotopic distribution.

6.1 RCL: Recurrence of Constant Length

The recurrence used to compute isotopic distribution might be shortened by using *approxParam*, with value $>= 5$ should be enough. This limits the number of multiplications, without much loss in accuracy. Let us see the examples:

```
> benchmarkRCL <- function(aC, nrPeaks){
+   print("Benchmarking correctness of approximations of results:")
+   resNoRCL <-  useBRAIN2(aC = aC, nrPeaks = nrPeaks)$iso
+   resRCL   <-  useBRAIN2(aC = aC, nrPeaks = nrPeaks, approxParam = 10)$iso
+   print(c('max error: ', max(abs(resNoRCL - resRCL))))
+   print(c('>> max result no RCL:  ', max(resNoRCL)))
```

```
+   print(c('>> max result with RCL:', max(resRCL)))
+   print("\nBenchmarking time performance")
+   print("No RCL: 10 replications")
+   print(system.time(
+         replicate(10, useBRAIN2(aC = aC, nrPeaks = nrPeaks)$iso ) ))
+   print("With RCL: 10 replications")
+   print(system.time(
+         replicate(
+           10, useBRAIN2(aC = aC, nrPeaks = nrPeaks, approxParam = 10)$iso )))
+
+
+ }
> #angiotensineII
> benchmarkRCL(aC = angiotensineII, nrPeaks = 50)

[1] "Benchmarking correctness of approximations of results:"
[1] "max error: "          "1.34686315023989e-16"
[1] ">> max result no RCL:  " "0.536240764521433"
[1] ">> max result with RCL:" "0.536240764521433"
[1] "\nBenchmarking time performance"
[1] "No RCL: 10 replications"
   user  system elapsed
   0.01    0.00    0.01
[1] "With RCL: 10 replications"
   user  system elapsed
   0.01    0.00    0.01

> #humandynein
> benchmarkRCL(aC = humandynein, nrPeaks = 1000)

[1] "Benchmarking correctness of approximations of results:"
[1] "max error: "          "5.35225290823615e-09"
[1] ">> max result no RCL:  " "0.020735257894218"
[1] ">> max result with RCL:" "0.0207352623437652"
[1] "\nBenchmarking time performance"
[1] "No RCL: 10 replications"
   user  system elapsed
   0.18    0.03    0.22
[1] "With RCL: 10 replications"
   user  system elapsed
   0.02    0.02    0.03
```

6.2 LSP: Late Starting Point

By using *approxStart* parameter, we can narrow down computations to the peak indices arrount most abundant ones. This option changes absolute values of isotopic distribution, but is aimed to approximate peak ratios. Note, the approximation needs some burn-in period to retrieve good isotopic ratios. The exemplary rule-of-thumb suggested in *BRAIN 2.0* paper is

$$\lceil (\log_{10}(monoisotopicMass)) \rceil + 5$$

Also, LSP option is useful for heavy molecules, were most abundant peak is much shifted from the monoisotopic peak.

```
> benchmarkLSP <- function(aC, startIdx, endIdx){
+   print("Benchmarking correctness of approximations of results:")
+   burn_in = 11 ##inspired by BRAIN 2.0 paper
+   resNoLSP <-  useBRAIN2(aC = aC, nrPeaks = endIdx)$iso
+   approxStart <- ifelse(1 <= startIdx-burn_in+1, startIdx-burn_in+1, 1)
+   resLSP   <-  useBRAIN2(aC = aC, nrPeaks = endIdx, approxStart = approxStart)$iso
+   print(c('length(resNoLSP): ', length(resNoLSP)))
+   print(c('length(resLSP): ', length(resLSP)))
+   resNoLSP_truncate = resNoLSP[startIdx:endIdx]
+   resLSP_truncate = resLSP[burn_in:length(resLSP)] ###remove peaks from burn_in region
+   print(c('Note, absolute values might vary'))
+   print(c('>> max result no RCL:  ', max(resNoLSP_truncate)))
+   print(c('>> max result with RCL:  ', max(resLSP_truncate)))
+   ratiosNoLSP = resNoLSP_truncate/max(resNoLSP_truncate)
+   ratiosLSP = resLSP_truncate/max(resLSP_truncate)
+   print(c('max error in interesting region: ', max(abs(ratiosNoLSP - ratiosLSP))))
+   print(c('>> top 3 ratios no RCL:  ',
+         sort(ratiosNoLSP, decreasing = TRUE)[1:3]))##we assume at least 3 values
+   print(c('>> top 3 ratios with RCL:',
+         sort(ratiosLSP, decreasing = TRUE)[1:3]))##we assume at least 3 values
+   print("\nBenchmarking time performance")
+   print("No RCL: 10 replications")
+   print(system.time( replicate(10, useBRAIN2(aC = aC, nrPeaks = endIdx)$iso )))
+   print("With RCL: 10 replications")
+   print(system.time(
+         replicate(
+          10, useBRAIN2(aC=aC, nrPeaks= endIdx, approxStart=approxStart)$iso )))
+
+
+ }
> #humandynein
> benchmarkLSP(aC = humandynein, startIdx = 210, endIdx = 450)

[1] "Benchmarking correctness of approximations of results:"
[1] "length(resNoLSP): " "450"
[1] "length(resLSP): " "251"
[1] "Note, absolute values might vary"
[1] ">> max result no RCL:  " "0.020735257894218"
[1] ">> max result with RCL:  " "804496485233.005"
[1] "max error in interesting region: " "7.7715611723761e-16"
[1] ">> top 3 ratios no RCL:  " "1"
[3] "0.999420825083151"         "0.997877678565631"
[1] ">> top 3 ratios with RCL:" "1"
[3] "0.999420825083151"         "0.997877678565631"
[1] "\nBenchmarking time performance"
[1] "No RCL: 10 replications"
   user  system elapsed
   0.06    0.02    0.08
```

```
[1] "With RCL: 10 replications"
   user  system elapsed
   0.03    0.00    0.04
```