

# RI Correction Extra

*Álvaro Cuadros-Inostroza*<sup>1</sup>

<sup>1</sup>Max Planck Institute for Molecular Plant Physiology, Potsdam, Germany

**August 5, 2025**

## **Abstract**

This vignette explains how to transform retention time (RT) to retention index (RI) when the RI markers are not spiked in the samples. The same principle can be applied to other uncommon scenarios.

## **Package**

TargetSearch 2.11.1

Report issues on <https://github.com/acinostroza/TargetSearch/issues>

## Contents

1	Motivation . . . . .	2
2	Retention Index correction. . . . .	2
3	Session info . . . . .	5

## 1 Motivation

The [TargetSearch](#) package assumes that the retention index markers (RIM), such as n-alkanes or FAMES (fatty acid methyl esters), are injected together with the biological samples. A different approach often used is to inject the RIMs separately, alternating between RIMs and biological samples. For example, a GC-MS run may look like the list shown in Table 1.

Measurement Order	Sample Type
1	Alkanes
2	Biological
3	Biological
4	Biological
5	Biological
6	Alkanes
7	Biological
8	Biological
9	Biological
10	Biological
11	Alkanes
12	Biological
13	Biological
14	Biological
15	Biological

**Table 1:** An example of a GC-MS run. *Alkanes* samples are retention index markers (RIMs).

In the example (Table 1), samples 1, 6, 11 are RIMs and the rest are biological samples. The assumption is that the retention time shifts between consecutive runs are not significant, so sample #1 is used to correct samples #2-5, sample #6 corrects samples #7-10, and so on. This document shows how to perform RI correction in such case with [TargetSearch](#), using the available chromatograms of the [TargetSearchData](#) package.

## 2 Retention Index correction

First, we load the required packages

```
library(TargetSearch)
library(TargetSearchData)
```

## RI Correction Extra

Specify the directory where the CDF files are. In this example we will use the CDF files of [TargetSearchData](#) package, which can be retrieved by the function `tsd_data_path()`.

```
cdfPath <- tsd_data_path()
dir(cdfPath, pattern="cdf$")

## [1] "7235eg04.cdf" "7235eg06.cdf" "7235eg07.cdf" "7235eg08.cdf"
## [5] "7235eg09.cdf" "7235eg11.cdf" "7235eg12.cdf" "7235eg15.cdf"
## [9] "7235eg20.cdf" "7235eg21.cdf" "7235eg22.cdf" "7235eg25.cdf"
## [13] "7235eg26.cdf" "7235eg30.cdf" "7235eg32.cdf"
```

Create a `tsSample` object using all chromatograms from the previous directory. Also set the RI path to the current directory. Use the R commands `setwd()` and `getwd()` to set/get the working directory.

```
samples.all <- ImportSamplesFromDir(cdfPath)
RIpath(samples.all) <- "."
```

Import retention index marker times limits. We will use the ones defined in [TargetSearchData](#) (the function `tsd_file_path()` gets the full path of the file).

```
rimLimits <- ImportFameSettings(tsd_file_path("rimLimits.txt"))
rimLimits

## An object of class "tsRim"
## Slot "limits":
##           LowerLimit UpperLimit
## RI.Marker 1         230         280
## RI.Marker 2         290         340
## RI.Marker 3         350         400
##
## Slot "standard":
## [1] 262320 323120 381020
##
## Slot "mass":
## [1] 87
```

Run the retention index correction methods. Here we use an intensity threshold of 50, the peak detection method is "ppc", and the time window width is  $2 * 15 + 1 = 31$  scan points (equal to 1.5 seconds in this example).

*Note: We skip conversion to CDF-4 files for illustration purposes. In a normal workflow you would set `writeCDF4path=TRUE` (the default).*

```
RImatrix <- RImatrix(samples.all, rimLimits, writeCDF4path=FALSE,
                     Window=15, pp.method="ppc", IntThreshold=50)
RImatrix

##           7235eg04 7235eg06 7235eg07 7235eg08 7235eg09 7235eg11 7235eg12
## RI.Marker 1   252.11 252.009 251.578 252.46 252.11 252.275 251.871
## RI.Marker 2   311.16 311.059 310.978 311.36 311.26 311.325 311.171
## RI.Marker 3   369.36 368.409 368.528 369.61 368.81 369.575 368.671
##           7235eg15 7235eg20 7235eg21 7235eg22 7235eg25 7235eg26 7235eg30
## RI.Marker 1  252.025 252.091 252.006 252.391 251.675 251.76 252.341
## RI.Marker 2  311.125 311.141 311.106 311.291 311.025 310.96 311.241
```

## RI Correction Extra

```
## RI.Marker 3 368.525 368.641 368.556 368.741 368.525 369.41 369.291
## 7235eg32
## RI.Marker 1 252.31
## RI.Marker 2 311.11
## RI.Marker 3 368.96
## attr("intensity")
## 7235eg04 7235eg06 7235eg07 7235eg08 7235eg09 7235eg11 7235eg12
## RI.Marker 1 205703 228186 188436 208359 183305 199893 175446
## RI.Marker 2 193708 221839 187412 190901 176739 188420 176675
## RI.Marker 3 199179 276981 211007 180635 198076 192659 191039
## 7235eg15 7235eg20 7235eg21 7235eg22 7235eg25 7235eg26 7235eg30
## RI.Marker 1 191280 213006 209952 212226 187063 191619 208937
## RI.Marker 2 185737 207126 204190 191877 199010 179361 205072
## RI.Marker 3 217234 208710 213369 224804 195209 172622 212263
## 7235eg32
## RI.Marker 1 181206
## RI.Marker 2 149373
## RI.Marker 3 182616
## attr("mass")
## [1] 87
```

Now you should make sure that the retention times (RT) of the RIMs (not the biological samples!!) are correct by using a chromatogram visualization tool such as LECO Pegasus, AMDIS, etc. Because there are no RIMs injected in the biological samples, the RTs of the RIMs found in `RImatrix` will make no sense, so you don't need to check them.

Up to now we have run the usual [TargetSearch](#) workflow which you can find in the main vignette. To correct the RI of the biological samples, the following procedure can be used.

First, we need a logical vector indicating which samples are RIMs and which biological. For example, you could create a vector like this.

```
isRIMarker <- rep(FALSE, length(samples.all))
isRIMarker[c(1, 6, 11)] <- TRUE
```

where `isRIMarker` is a logical vector of length equal to the number of samples in which `TRUE` signals that the respective sample is a RIM and `FALSE` otherwise. Note that here we have set components 1, 6, 11 to `TRUE` just like the example in Table 1.

Then we have to copy the RIM columns of `RImatrix` to their respective biological sample columns. In other words, copy column 1 to columns 2, 3, 4, 5; column 6 to columns 7, 8, 9, 10; and so on. There are many ways to achieve that, here I show two examples.

```
RImatrix2 <- RImatrix
RImatrix2[, 2:5] <- RImatrix[, 1]
RImatrix2[, 7:10] <- RImatrix[, 6]
RImatrix2[, 12:15] <- RImatrix[, 11]
```

This code snippet is a straight forward method, but has the disadvantage that the column indexes must be filled manually. A more general approach could be.

```
RImatrix2 <- RImatrix
z <- cumsum(as.numeric(isRIMarker))
```

## RI Correction Extra

```
for(i in unique(z))
  RImatrix2[, z==i] <- RImatrix[, z==i][,1]
RImatrix2

##           7235eg04 7235eg06 7235eg07 7235eg08 7235eg09 7235eg11 7235eg12
## RI.Marker 1    252.11   252.11   252.11   252.11   252.11   252.275   252.275
## RI.Marker 2    311.16   311.16   311.16   311.16   311.16   311.325   311.325
## RI.Marker 3    369.36   369.36   369.36   369.36   369.36   369.575   369.575
##           7235eg15 7235eg20 7235eg21 7235eg22 7235eg25 7235eg26 7235eg30
## RI.Marker 1    252.275   252.275   252.275   252.391   252.391   252.391   252.391
## RI.Marker 2    311.325   311.325   311.325   311.291   311.291   311.291   311.291
## RI.Marker 3    369.575   369.575   369.575   368.741   368.741   368.741   368.741
##           7235eg32
## RI.Marker 1    252.391
## RI.Marker 2    311.291
## RI.Marker 3    368.741
## attr(,"intensity")
##           7235eg04 7235eg06 7235eg07 7235eg08 7235eg09 7235eg11 7235eg12
## RI.Marker 1    205703   228186   188436   208359   183305   199893   175446
## RI.Marker 2    193708   221839   187412   190901   176739   188420   176675
## RI.Marker 3    199179   276981   211007   180635   198076   192659   191039
##           7235eg15 7235eg20 7235eg21 7235eg22 7235eg25 7235eg26 7235eg30
## RI.Marker 1    191280   213006   209952   212226   187063   191619   208937
## RI.Marker 2    185737   207126   204190   191877   199010   179361   205072
## RI.Marker 3    217234   208710   213369   224804   195209   172622   212263
##           7235eg32
## RI.Marker 1    181206
## RI.Marker 2    149373
## RI.Marker 3    182616
## attr(,"mass")
## [1] 87
```

After the `RImatrix2` object is corrected, the RI files of the biological samples must be fixed as well.

```
fixRI(samples.all, rimLimits, RImatrix2, which(!isRIMarker))
```

Finally, we remove the standards, since we don't need them anymore.

```
samples <- samples.all[!isRIMarker]
RImatrix <- RImatrix2[, !isRIMarker]
```

After that, we can continue we the normal [TargetSearch](#) workflow.

A script with all the commands used in this document is located in the 'doc' directory of the [TargetSearch](#) package.

## 3 Session info

Output of `sessionInfo()` on the system on which this document was compiled.

- R version 4.5.1 (2025-06-13 ucrt), x86\_64-w64-mingw32

## RI Correction Extra

- Locale: LC\_COLLATE=C, LC\_CTYPE=English\_United\_States.utf8, LC\_MONETARY=English\_United\_States.utf8, LC\_NUMERIC=C, LC\_TIME=English\_United\_States.utf8
- Time zone: America/New\_York
- TZcode source: internal
- Running under: Windows Server 2022 x64 (build 20348)
- Matrix products: default
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: TargetSearch 2.11.1, TargetSearchData 1.47.0
- Loaded via a namespace (and not attached): BiocManager 1.30.26, BiocStyle 2.37.1, assertthat 0.2.1, cli 3.6.5, compiler 4.5.1, digest 0.6.37, evaluate 1.0.4, fastmap 1.2.0, highr 0.11, htmltools 0.5.8.1, knitr 1.50, ncd4 1.24, rlang 1.1.6, rmarkdown 2.29, tools 4.5.1, xfun 0.52, yaml 2.3.10