affy: Custom Processing Methods (HowTo)

Laurent

April 28, 2026

Contents

1 Introduction

This document describes briefly how to customize the affy package by adding one's own processing methods. The types of processing methods are background correction, normalization, perfect match correction and summary expression value computation. We tried our best to make this as easy as we could, but we are aware that it is far from being perfect. We are still working on things to improve them. Hopefully this document should let you extend the package with supplementary processing methods easily.

As usual, loading the package in your R session is required.

```
R> library(affy) ##load the affy package
```

2 How-to

For each processing step, labels for the methods known to the package are stored in variables.

```
> normalize.AffyBatch.methods()
```

```
[1] "constant"         "contrasts"        "invariantset"     "loess"
[5] "methods"          "qspline"          "quantiles"        "quantiles.robust"
```

```
> bgcorrect.methods()
```

variable for labels	naming convention
bgcorrect.methods	bg.correct.<label>
normalize.AffyBatch.methods	normalize.AffyBatch.<label>
pmcorrect.methods	pmcorrect.<label>
express.summary.stat.methods	generateExprset.methods.<label>

Table 1: Summary table for the processing methods.

step	argument(s)	returns
background correction	`AffyBatch`	`AffyBatch`
normalization	`AffyBatch`	`AffyBatch`
pm correction	`ProbeSet`	a `matrix` of corrected PM values (one probe per row, one column per chip).
expression value	a `matrix` of PM values	a list of two elements `exprs` and `se.exprs`

Table 2: Summary table for the processing methods.

```
[1] "bg.correct" "mas"        "none"       "rma"
```

```
> pmcorrect.methods()
```

```
[1] "mas"        "methods"    "pmonly"     "subtractmm"
```

```
> express.summary.stat.methods()
```

```
[1] "avgdiff"      "liwong"       "mas"          "medianpolish" "playerout"
```

We would recommend the use of the method `normalize.methods` to access the list of available normalization methods (as a scheme for normalization methods that would go beyond 'affy' is thought).

```
> library(affydata)
> data(Dilution)
> normalize.methods(Dilution)
```

```
[1] "constant"         "contrasts"        "invariantset"     "loess"
[5] "methods"          "qspline"          "quantiles"        "quantiles.robust"
```

For each processing step, a naming convention exists between the method label and the function name in `R` (see table 1). Each processing methods should be passed objects (and return objects) corresponding to the processing step (see table 2).

Practically, this means that to add a method one has to

1. create an appropriate method with a name satisfying the convention.
2. register the method by adding the label name to the corresponding variable.

3 Examples

As an example we show how to add two simple new methods. The first one does *pm* correction. The method subtract *mm* values to the *pm* values, except when the result is negative. In this case the *pm* value remains unchanged.

We create a function using the label name `subtractmmsometimes`.

```
> pmcorrect.subtractmmsometimes <- function(object) {
+
+   ## subtract mm
+   mm.subtracted <- pm(object) - mm(object)
+
+   ## find which ones are unwanted and fix them
+   invalid <- which(mm.subtracted <= 0)
+   mm.subtracted[invalid] <- pm(object)[invalid]
+
+   return(mm.subtracted)
+ }
```

Once the method defined, we just register the *label name* in the corresponding variable.

```
> upDate.pmcorrect.methods(c(pmcorrect.methods(), "subtractmmsometimes"))
```

The second new method intends to be an robust alternative to the summary expression value computation `avgdiff`. The idea is to use the function `huber` of the package `MASS`.

```
> huber <- function (y, k = 1.5, tol = 1e-06) {
+     y <- y[!is.na(y)]
+     n <- length(y)
+     mu <- median(y)
+     s <- mad(y)
+     if (s == 0)
+         stop("cannot estimate scale: MAD is zero for this sample")
+     repeat {
+         yy <- pmin(pmax(mu - k * s, y), mu + k * s)
+         mu1 <- sum(yy)/n
+         if (abs(mu - mu1) < tol * s)
+             break
+         mu <- mu1
+     }
+     list(mu = mu, s = s)
+ }
```

This method returns P.J. Huber's location estimate with MAD scale. You think this is what you want to compute the summary expression value from the probe intensities. What is needed to have as a processing method is a simple wrapper:

```
> computeExprVal.huber <- function(probes) {
+   res <- apply(probes, 2, huber)
+   mu <- unlist(lapply(res, function(x) x$mu))
+   s <- unlist(lapply(res, function(x) x$s))
+   return(list(exprs=mu, se.exprs=s))
+ }
> upDate.generateExprSet.methods(c(generateExprSet.methods(), "huber"))
```

From now the package is aware of the two new methods... in this session.

The code for the methods included in the package can be informative if you plan to develop methods. An example that demonstrates how a normalization method can be added is given by the function `normalize.AffyBatch.vsn` in the package *vsn*; see its help file.