# yaqcaffy: Affymetrix expression GeneChips quality control and reproducibility with MAQC datasets

Laurent Gatto*

October 29, 2019

## Contents



*laurent.gatto@gmail.com

# 1 Introduction

Quality control is an important step in the analyses of microarray data. Indeed, poor quality arrays can can significant impact on subsequent analyses and conseqeuntly invalidate their interpretation. The Bioconductor project has several packages for quality control of microarray data, as listed under the *QualityControl* biocViews item.

The *yaqcaffy* package is part of the Bioconductor[1] project. It was written to automate the quality analysis of Affymetrix expression arrays and test in-house Human Whole Genome GeneChips array reproducibility against (a subset of) the Microarray Quality Consortium (MAQC) reference datasets. It is based on the *affy* and, in particular, *simpleaffy* packages, which do all the hard work. The *simpleaffy* package provides a variety of functions for high-level analysis of Affymetrix data as well as methods to assess some quality metrics of the arrays.

Since *yaqcaffy* is based on the *simpleaffy* (for example, it creates an `YAQCStats` object which is a subclass of *simpleaffy*'s `QCStats`), a basic understanding of the library, its vignette and the simpleaffy QC capabilities described in *QC and Affymetrix data*[2] is welcome.

# 2 The Affymetrix Quality Metrics

The **scale factor** (`scale.factors` slot[3]) is an array specific value that is used by Affymetrix software to adjust array intensities towards a user defined target value (default tgt=100 in *simpleaffy* and *yaqcaffy*) based on the (trimmed) mean array intensities. If there are no biases of labeling or hybridization across arrays, the highest value for the scale factor should be less than three times the smallest value.

The **background** and **noise averages** (`average.background`[3] and `average.noise` slots) assume that the hybridization occurred with the similar background and noise. Affymetrix suggests that arrays being compared

[1] `http://www.bioconductor.org/`

[2] `http://bioinf.picr.man.ac.uk/simpleaffy/QCandSimpleaffy.pdf`

[3] defined in the *simpleaffy*'s QCStats object

should ideally have comparable background and noise values.

The **percentage of present calls** (`percent.present`[3] slot) assumes that the number of probe sets called present relative to the total number of probe sets remains similar across arrays. Nevertheless, variability in the percentage of present calls might also represent biological variability.

The internal probe calls **AFFX-r2-Ec-bioB** (M', 3' ,5'), **bioC** (5', 3') and **bioD** (5', 3') (`morespikes` and `bio.calls` slots) are *E. coli* genes that are used as internal hybridization controls and must always be present (P)[4]. Furthermore, the overall signal AFFX-r2-Ec-bioB (All), AFFX-r2-Ec-bioC (All) and AFFX-r2-Ec-bioD (All) for these spikes are present in increasing concentration (1.5 pM, 5 pM and 25 pM for bioB, bioC and bioD respectively).

The ploy-A controls **AFFX-r2-Bs-Dap**, **AFFX-r2-Bs-Thr**, **AFFX-r2-Bs-Phe** and **AFFX-r2-Bs-Lys** (`morespikes` slot) are modified *B. subtilis* genes and should be called present at a decreasing intensity, to verify that there was no bias during the retro-transcription between highly expressed genes and low expressed genes. Note that the linearity for lys, phe and thr (dap is present at a much higher concentration) is affected by a double amplification.

Note that Affymetrix provides two sets of internal *bio* and *poly-A* controls. If we take as an example the bioB spike control, two similar probe sets IDs are present on some GeneChips: `AFFX-BioB-3_at` and `AFFX-r2-Ec-bioB-3_at`. These two probe sets target the same gene, but the individual probes are slightly shifted. The *r2* probe sets include less probes (11 for each control spike) than the older non-*r2* sets (20 probes per set). The *yaqcaffy* package uses the *r2* probe sets unless these are not available (as in older GeneChips).

The **GAPDH** and **$\beta$-Actin** 3'/5' signal ratios are RNA degradation controls (see slot `gcos.probes`). These values should generally be smaller then 3. Nevertheless, double amplification is known to have a significant impact on these two parameters.

More information regarding the Affymetrix internal controls can be found in the *GeneChip Expression Analysis* and *Data Analysis Fundamentals* manuals [5].

[4]Note that bioB is at the level of array sensitivity and might be absent (A) in less then 50% calls.

[5]`http://www.affymetrix.com/support/technical/manual/expression_manual.affx`

To assess the quality of the samples to analyses, we suggest that most qc metrics should lie within 2 standard deviations of one another across the entire set of arrays. We apply this rule to the above mentioned metrics. For the scale factor, we define the upper and lower limits as the $mean/2$ and $mean * 1.5$ respectively to stick to Affymetrix's three-fold rule.

# 3 The MAQC Reference Datasets

The Microarray Quality Consortium (MAQC) project [6]provides a set of reference datasets for a set of platforms (see *Summary of the MAQC Data Sets* [7]for more details). Regarding the Affymetrix platform (AFX prefix), a total of 120 Human Genome U133 Plus 2.0 GeneChips have been generated. Four different reference RNAs have been used: (A) 100% of Stratagene's *Universal Human Reference RNA*, (B) 100% of Ambion's Human Brain Reference RNA, (C) 75% of A and 25% of B and (D) 25% of A and 75% of B. Each reference has been repeated 5 times (noted _A1_ to _A5_) on six different test sites (noted _1_ to _6_). As an example, the .CEL result file for the first replicate of test site 2, for the reference ARN C is named `AFX_2_C1.CEL`.

These datasets are freely available and allow researchers, among other things, to compare the reproducibility of their own Human Genome U133 Plus 2.0 arrays with a set of high quality .CEL files. Nevertheless, using all the 30 available .CEL files (per reference RNA) is memory consuming and further reproducibility calculations time consuming. We randomly chose 6 .CEL file for each reference RNA, one for each test site as reference to compare the user's data to. These 6 .CEL files are distributed with the *MAQCsubsetAFX* package as associated data (respectively called `refA.RData`, `refB.RData`, `refC.RData` and `refD.RData`). These subsets are used to compute the Pearson correlation factors and draw scatterplots with the users data (see section 8).

[6]`http://www.fda.gov/ScienceResearch/BioinformaticsTools/MicroarrayQualityControlProject/default.htm`

[7]`http://www.fda.gov/downloads/ScienceResearch/BioinformaticsTools/MicroarrayQualityControlProject/UCM134500.pdf`

# 4 Data Classes defined in *yaqcaffy*

The main function of the package is `yaqc`, which is described in section 5. When calling this function with and `AffyBatch` object, (1) the data is normalised with the MAS5 algorithm, (2) the quality control probes are selected (an object of class `YaqcControlProbes` is instanciated), (3) the expression intensities and other quality metrics are extracted and (4) and `YaqcStats` object is created.

## 4.1 `YAQCStats` class

The `YAQCStats` class is the main class of the *yaqcaffy* pckage. It contains all the values of the quality metrics and is used to plot the quality plots. Since version 1.7 of the package, this class also contains two additional slots. The `objectVersion` stores the version of the library used to generate the `YAQCStats` object. The probe names used to compute the quality metrics are also explicitely stored and can be retrieved with the `getYaqcControlProbes` function (see section 4.2 for more details).

## 4.2 `YaqcControlProbes` class

This class is contains the names of the three main groups of control probes: the hybridization control probes (*bio* probes), the labeling control probes (the *spike* probes) and the degradation probes (used to compute the 3'/5' ratios). The three groups are contained in their own classes, namely `YaqcBioProbes`, `YaqcSpkProbes` and `YaqcDegProbes`.

The quality control probes that are used are selected automatically and appropriate warnings or errors are issued in several cases. These probes can also be explicitely selected by the user, as described in section 7.

# 5 Generating an `YAQCStats` Object

As an example, we will use *affydata*'s `Dilution` dataset. We will modify the raw probe intensities of the first sample to illustrate some of *yaqcaffy*'s functions below.

```
> library("yaqcaffy")
> library("affydata")
```

```
     Package    LibPath                                          Item
[1,] "affydata" "/home/biocbuild/bbs-3.10-bioc/R/library" "Dilution"
     Title
[1,] "AffyBatch instance Dilution"
```

```
> data(Dilution)
> ## probe intensities modification
> tmp <- exprs(Dilution)
> tmp[,1] <- tmp[,1]*2
> exprs(Dilution) <- tmp
```

The next step is the creation of the `YAQCStats` object that will hold the data that will subsequently be used to assess the quality of the arrays (see section 6). The `YAQCStats` object is a subclass of the `QCStats` object, defined in the *simpleaffy* package.

The function `yaqc` computes the following values that are used for quality assignment:

1. the scale factors, percent of present calls, average background and noise that are tested as described above;
2. the bioB, bioC and bioD calls;
3. the intensity values for the bioB, bioC, bioD and dap, lys, phe and thr probes, as computed by the Affymetrix GCOS software;
4. the intensity values for GAPDH and $\beta$-actin probes as computed by the Affymetrix GCOS software.

The newly created object can then be visualized as a `data frame` with the `show()` function.

```
> yqc <- yaqc(Dilution, verbose=TRUE)
```

```
MAS5 normalisation... done
Getting probe names... done
Extracting data... done
Generation YAQCStats object...
```

```
> show(yqc)
```

| | 20A | 20B | 10A |
|---|---|---|---|
| scale.factors | "0.446700673288299" | "1.26536267137974" | "1.14484301856915" |
| average.background | "188.506453894315" | "63.6385483408235" | "80.0943568071944" |
| average.noise | "5.99634613568846" | "2.05635393118791" | "2.41104721237696" |
| percent.present | "48.6732673267327" | "49.7029702970297" | "49.2514851485149" |
| b5 | "664.475559859501" | "529.820083206257" | "782.380763943258" |
| b3 | "3193.34776384752" | "2403.61326911566" | "3803.05793263888" |
| bm | "3405.73144678144" | "2848.81587661468" | "4099.20419046011" |
| c5 | "140.223227877387" | "102.335999707106" | "134.208345100585" |
| c3 | "252.436293058404" | "205.437770825853" | "292.390728738772" |
| d5 | "1.51769065090198" | "0.845418666059053" | "1.66675953810322" |
| d3 | "3877.1350169243" | "3394.51773110207" | "4730.57421072259" |
| dap5 | "70.7879945420667" | "77.6307445901373" | "109.999093608403" |
| dap3 | "47.623445854456" | "47.0670722097149" | "83.1313670261053" |
| dapm | "142.10873491128" | "157.471548891886" | "224.636335945386" |
| thr5 | "6.10352070691631" | "14.2456808606269" | "2.00185328136356" |
| thr3 | "3.24359573960641" | "1.69876702933264" | "3.8846865775984" |
| thrm | "9.21704036082898" | "4.98124090682066" | "4.50047460899867" |
| lys5 | "5.58330311506038" | "2.3329840013985" | "2.01530814419427" |
| lys3 | "4.80101892491885" | "5.19664179741684" | "7.43964195242549" |
| lysm | "9.18049463242503" | "5.52469246504483" | "10.1816741622856" |
| phe5 | "1.44519375614277" | "0.670665049720502" | "1.79541277725482" |
| phe3 | "7.08179780076799" | "6.16637709724212" | "8.83419969356382" |
| phem | "0.674714424858515" | "1.67990083326996" | "2.06622009441486" |
| act5 | "3422.54064237669" | "3043.59824738725" | "3852.89986182388" |
| act3 | "5545.0863924076" | "5016.28641670242" | "7047.69553069851" |
| actm | "5076.6312266744" | "4429.0480858254" | "6087.54358313079" |
| gap5 | "3276.2164923853" | "3112.48782158316" | "3658.51361772065" |
| gap3 | "4453.70276234178" | "3975.28201930944" | "4937.95681357743" |
| gapm | "4643.6097792756" | "4013.10843566896" | "3681.16271724012" |
| AFFX-BioB-5_at_call | "A" | "A" | "A" |
| AFFX-BioB-3_at_call | "A" | "A" | "A" |
| AFFX-BioB-M_at_call | "A" | "A" | "A" |
| AFFX-BioC-5_at_call | "P" | "P" | "P" |
| AFFX-BioC-3_at_call | "A" | "A" | "A" |
| AFFX-BioDn-5_at_call | "A" | "A" | "A" |
| AFFX-BioDn-3_at_call | "A" | "A" | "A" |
| | 10B | | |

```
scale.factors        "1.84540671835491"
average.background   "54.2582973752169"
average.noise        "1.53954177290118"
percent.present      "49.639603960396"
b5                   "868.048189730846"
b3                   "4651.34002639248"
bm                   "4708.37272584289"
c5                   "172.144437858794"
c3                   "400.137460869786"
d5                   "1.05867006723521"
d3                   "5815.90360991049"
dap5                 "133.380317244418"
dap3                 "82.203055600624"
dapm                 "231.292608780094"
thr5                 "9.39203341871421"
thr3                 "1.89381748154357"
thrm                 "1.90547577524204"
lys5                 "1.03588748317022"
lys3                 "3.09333383020266"
lysm                 "1.25190837897071"
phe5                 "1.0836801640999"
phe3                 "6.15585638526383"
phem                 "1.53941417497261"
act5                 "3508.18103953567"
act3                 "6690.4076575177"
actm                 "5538.67447202837"
gap5                 "3412.0483110118"
gap3                 "5074.631527531"
gapm                 "4693.68394083838"
AFFX-BioB-5_at_call  "A"
AFFX-BioB-3_at_call  "A"
AFFX-BioB-M_at_call  "A"
AFFX-BioC-5_at_call  "P"
AFFX-BioC-3_at_call  "A"
AFFX-BioDn-5_at_call "A"
AFFX-BioDn-3_at_call "A"
```

The version of the package that has been used to generate a given object

can be recovered with the `objectVersion` function.

```
> objectVersion(yqc)
```

```
[1] "1.46.0"
```

In the above examples, the data given as input is of class `AffyBatch` object. An `YAQCStats` object can also be created by providing an `ExpressionSet`, in which case some of the qc metrics cannot be computed: only the intensity values for the bioB, bioC, bioD and dap, lys, phe and thr probes and GAPDH and $\beta$-actin probes are used.

# 6 Quality Control Analysis

The quality metrics in the `YAQCStats` object can be plotted out to allow an easy and rapid overview, as shown on figure 1:

- the scale factors for the different arrays are plotted with the upper and lower limits as a dotchart;
- boxplots for the average background and noise, the percentage of present calls and GAPDH and $\beta$-actin $\frac{3'}{5'}$ ratios.
- boxplots of the control probes *biob*, *bioc*, *biod* and *dap*, *thr*, *phe*, *lys* intensities respectively

The mean (longdashed line), upper and lower 2 standard deviations (dotted lines) are also plotted on the graphs. The upper and lower limits may however not appear when they are outside of the boxplot y-axis. For the internal probes, a grey rectangle represents the mean (middle segment) and the +/- 2 stdev range.

The outliers (i.e. the data points the lie outside the mean +/- 2 stdev) can be queried and listed for each qc metrics using the `getOutliers()` function. The arguments are the `YAQCStats` object and a string describing the metrics that should be queried. In the above example, we can see that the scale factors of the fourth samples (counting from the botton) is out of range and not even present on the dotchart. We can retrieve the name of the sample and its scale factor value by typing:

```
> getOutliers(yqc,"sfs")
```

```
> plot(yqc)
```

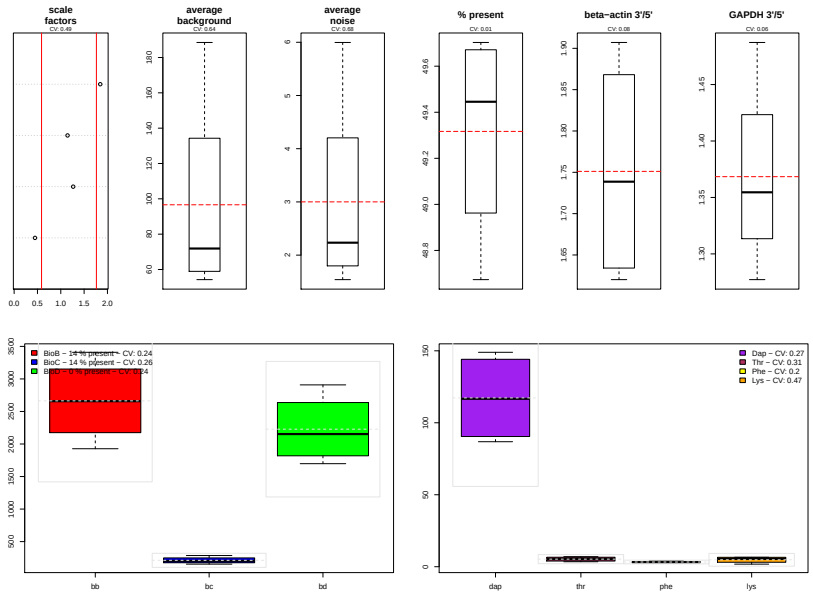



Figure 1: Graphical representation of the `YAQCStats` object.

```
      20A       10B
0.4467007 1.8454067
```

The qc metrics strings are respectively sfs, avbg, avns, pp, actin, gapdh, biob, bioc, biod, dap, thr, phe, lys (listed in their order of apperance on the qc plot). Individual plots can also be generated with the `which` argument: 'sfs' for the scale factor, 'avbg' and 'avns' for the average background and noise, 'pp' for the percentage of present calls, 'gapdh' and 'actin' for the GAPDH and $\beta$-actin ratios, 'bio' for the hybridization controls and 'spikes' for the retro-transciption spiked controls. In addition, the coefficient of variation is calculated for each qc metric and indicated on the qc plot. The outliers can be summerized in a data frame calling the `summary()` function on a `YAQCStats` object.

It is also possible to combine two `YAQCStats` object into one with the `merge()` function. To illustration this function, we will use the `arrays()` function that outputs the arrays names of the `YAQCStats` provided as parameter.

```
> yqc2 <- yaqc(Dilution[, 2:3])
> arrays(yqc)

[1] "20A" "20B" "10A" "10B"

> arrays(yqc2)

[1] "20B" "10A"

> yqc3 <- merge(yqc, yqc2)
> arrays(yqc3)

[1] "20A" "20B" "10A" "10B" "20B" "10A"
```

# 7 Generating ones own Control Probes Object

As already mentionned, the control probes are selected automatically. The selection is done based on patterns in the Affymetrix probe names. When

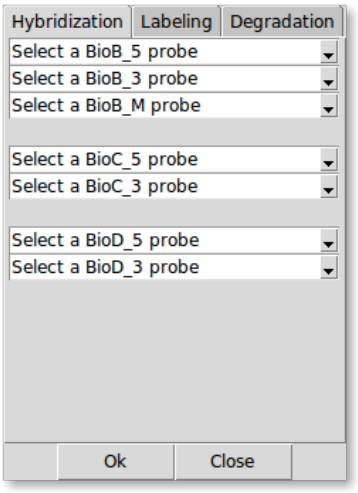



Figure 2: QC probes selection window.

possible, the *r2* probes are used. If these are not available (for instance in older arrays), the non-*r2* are used and a warning message is issued.

Sometimes, several probes can match a given pattern. In this case, a warning is issued but only the first probe is retained. All the probes that matched the pattern are given as part as the warning message. If the first probe is not the best one or if it does not match with the other probes of the group, the user is invited to created his/her own control probes.

If no probes match the pattern, an error is issued and the function exits.

The probes can be selected trough a graphical user interface that is started with the (probeSelectionInterface) function. This function requires a `AffyBatch` (or `ExpressionSet`) object as parameter. An additionnal logical parameter `filter` (which is by default set to TRUE) controls if the control probes can be selected from all the probes on the GeneChip or if a pre-filtering is done. This filtering removes the probes that are explicitely non-control features and groups the hybridization and labeling probes accordingly.

A tabbed window (see figure 2) opens up. Note that the probe selection is saved as an `yaqcControlProbes` object in the global environment once the window is closed. It is names *yaqcControlProbes* by default. The name of the output can be set with the `returnVar` parameter. If an object named *yaqccontrolProbes* alredy exists, the warning is issued in a new window and the user can cancel the operation to avoid to overwrite it existing object.

The tabs correspond to the three classes of control probes that can be defined. The *hybridization* tab has 7 drop-down menues, for the three BioB probes (5', M and 3'), the two BioC probes (5' and 3') and the two BioD probes (5' and 3'). The *labeling* tab has 12 drop-down menues, for the dap, thr, phe and lys 3', M and 5' probes respectively. The *degradation* tab has 6 drop-down menues, for the beta-actin and GAPDH 3', M and 5' probes respectively.

For some arrays, other probe sets than beta-actin and GAPDH are used for degradation control. These other genes will be present in the list (even when filtering is used).

Once the probes are selected, an `YaqcControlProbes` object (named as defined by returnVar, see above) is saved in the global environment when pressing `Ok`. No object is saved if the `Close` is pressed.

Note that the validity of any `YaqcControlProbes` object is checked before it is generated. Among the validity requirements of this class, there are constrains on the probe names, which must not contain white spaces. As such, all the probes must be properly selected from the drop-down menues. If not, an 'Error in validObject(.Object)' will be issued.

This object can then be used to generate an `YAQCStats` object as descrined in section 5, by adding it as a parameter to the `yaqc` function as shown below.

```
> yqc <- yaqc(myAffyData, myYaqcControlProbes = yaqcControlProbes)
```

# 8 Human genome U133 Plus 2.0 Reproducibility

To illustrate this section, we will compare the first array of the RNA B reference dataset (`AFX_1_B1.CEL`) to the RNA A reference dataset[8].

```
> library(MAQCsubsetAFX)
> data(refB)
> d <- refB[,1]
> sampleNames(d)
```

[8]Note that the reproducibility statistics will *de facto* be low, as the conditions to be compared are different.

```
[1] "AFX_1_B1.CEL"
```

We will compare this CEL file to the `refA` dataset using the `reprodPlot` function. The name of the `AffyBatch` object to be tested is given as first argument and the reference data is specified as a character provided as second parameter (respectively `"refA"`, `"refB"`, `"refC"` or `"refD"`). The reference dataset is automatically loaded and merged with the user's `AffyBatch` object, normalized and results are plotted. The intensities used for the statistics are normalized using the RMA algorithm implemented in the *affy* package (`normalize="rma"`, default). It is also possible the use GCRMA (as implemented in the *gcrma* package, `normalize="gcrma"`), MAS5 (as implemented in *affy*, `normalize="mas5"`)) or no normalization (`normalize="none"`).

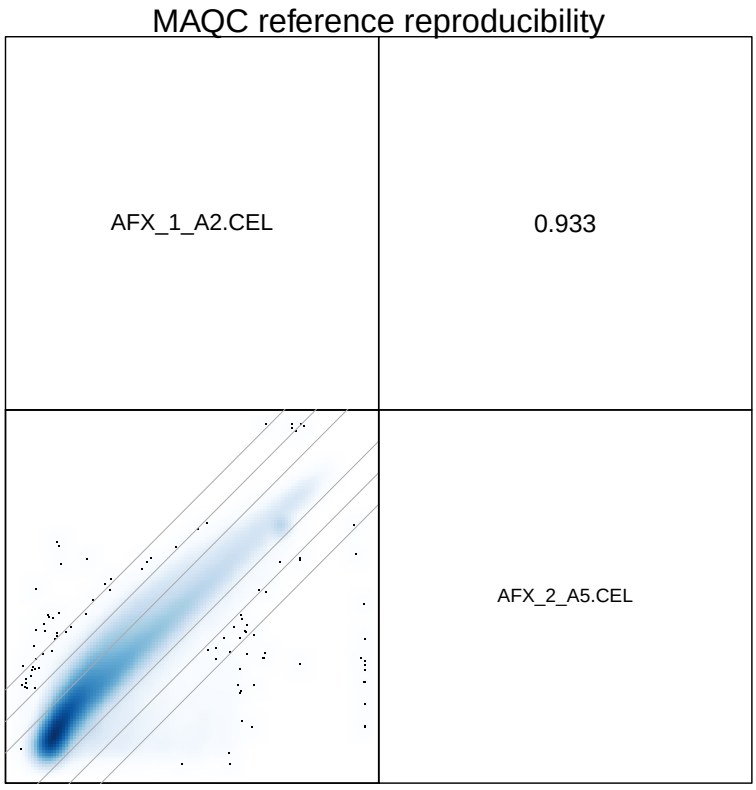
The `reprodPlot` function draws a 6 by 6 matrix showing scatterplots (below the diagonal) and the Pearson correlation factors (above the diagonal) for all comparisons. The sample names are given on the diagonal. The gray lines on the scatterplots represent respectively 2, 4 and 8 fold change differences.

```
> reprodPlot(d, "refA", normalize="rma")
```

The figure below is an example of the `reprodPlot` for 2 unnormalized samples [9].

```
> reprodPlot(d, "test", normalize="none")
```

[9] This `test` plot is used instead of the 6 by 6 plot to reduce time and size requirements to build the vignette.




# 9 Acknowledgements

This package has initially been developped at DNAVision in collaboration with Jean-Francois Laes.

# 10 Session information

- R version 3.6.1 (2019-07-05), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`

- Running under: `Ubuntu 18.04.3 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.10-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.10-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.46.0, BiocGenerics 0.32.0, MAQCsubsetAFX 1.23.0, affy 1.64.0, affydata 1.33.0, gcrma 2.58.0, genefilter 1.68.0, hgu95av2cdf 2.18.0, simpleaffy 2.62.0, yaqcaffy 1.46.0
- Loaded via a namespace (and not attached): AnnotationDbi 1.48.0, BiocManager 1.30.9, Biostrings 2.54.0, DBI 1.0.0, IRanges 2.20.0, KernSmooth 2.23-16, Matrix 1.2-17, RCurl 1.95-4.12, RSQLite 2.1.2, Rcpp 1.0.2, S4Vectors 0.24.0, XML 3.98-1.20, XVector 0.26.0, affyio 1.56.0, annotate 1.64.0, backports 1.1.5, bit 1.1-14, bit64 0.9-7, bitops 1.0-6, blob 1.2.0, compiler 3.6.1, crayon 1.3.4, digest 0.6.22, grid 3.6.1, lattice 0.20-38, memoise 1.1.0, pillar 1.4.2, pkgconfig 2.0.3, preprocessCore 1.48.0, rlang 0.4.1, splines 3.6.1, stats4 3.6.1, survival 2.44-1.1, tibble 2.1.3, tools 3.6.1, vctrs 0.2.0, xtable 1.8-4, zeallot 0.1.0, zlibbioc 1.32.0