Initiation à l'utilisation de l'environnement Bioconductor. Exemples sur des données Affymetrix.

Denis Puthier

15 novembre 2011

Laboratoire INSERM TAGC/ERM206, Parc Scientifique de Luminy case 928,13288 MARSEILLE cedex 09, FRANCE.
http://biologie.univ-mrs.fr/view-data.php?id=245

Table des matières

1 Objectifs du cours **2**

2 Notes à propos du projet Bioconductor **2**

3 Chargement des données **3**

4 L'objet AffyBatch **4**

5 Accéder au contenu d'un objet AffyBatch. **5**
5.1 Principe 5
5.2 Exemple : le slot assayData 5

6 Lecture des données de phénotypes **6**

7 Indexation d'un objet affyBatch **6**

8 Caractéristiques physiques de la puce HGU133a **6**

9 Outils graphiques de la librairie affy. **7**
9.1 Image du microarray 7
9.2 Les probesets 7

10 Contrôle qualité des données brutes **9**

11 Statistiques descriptives **9**
11.1 AffyRNAdeg 9

12 Quels sont les gènes qui s'expriment ? **9**

13 Normalisation des données **11**

14 L'objet ExpressionSet **11**

15 Contrôle qualité sur les données normalisées **12**
15.1 Boxplot 12
15.2 Relative Log Expression (RLE) 12
15.3 Graphique MvsA ("MA plot") 12

16 Annotation des sondes **14**

17 Sortie disque **16**

1 Objectifs du cours

Introduire le fonctionnement des librairies du projet Bioconductor.
Normaliser et effectuer le contrôle qualité de puces Affymetrix.

2 Notes à propos du projet Bioconductor

Le projet Bioconductor a été lancé en 2001. L'objectif du projet Bioconductor est d'offrir à la communauté scientifique (biologistes, bio-informaticiens, statisticiens) un palette d'outils ouverts, bien documentés, intégrés et contrôlés, permettant l'analyse des données biologiques toujours plus volumineuses. Bioconductor est largement orienté vers l'analyse de données de microarrays et propose depuis quelques années un certain nombre d'outils permettant d'analyser des données de séquençage.
La force de Bioconductor est de standardiser les méthodes d'importation et d'analyse tout en offrant une certaine versatilité et une palette d'outils très large. Cette standardisation des méthodes permet aussi la reproductibilité des expériences puisque qu'il devient très facile de re-analyser des données à partir du code. On s'oriente, alors, peu à peu vers un véritable cahier de manipulation informatisé que l'on pourrait soumettre en tant que matériel et méthode associé à une publication.
Les librairies que l'on trouve dans bioconductor sont extrêmement nombreuses ($\sim$ 500). La librairie principale est "Biobase", qui contient le descriptif des classes de base de Bioconductor. Pour cette raison, Biobase est nécessaire au fonctionnement de certaines des

librairies de Bioconductor. Parmi les nombreuses librairies on peut citer celles dédiées à :

- L'analyse de données de microarrays de type Affymetrix : Affy, simpleaffy, gcrma, exonmap, ...
- L'annotation des sondes : Annotate et les librairies d'annotation (hgu133a.db, hgu133aprobe),...
- Le filtrage des données de microarrays : Genefilter.
- La recherche de gènes différentiellement exprimés : siggenes, RankProd, Multtest, ...
- L'interprétation biologique des données : GO, Gostats, goCluster, geneplotter,...
- L'analyse de données issues de méthodes HTS : ShortRead, Rsamtools, SRAdb,...
- L'analyse de graphes : graph, Rgraphviz,...
- La cytométrie en flux : flowCore, flowViz,...
- La protéomique : isobar, xcms,...
- L'analyse de cultures cellulaires à haut débit : cellHTS...
- L'analyse d'images : EBImage...
- ...

On peut à tout moment obtenir de l'aide sur les librairies a l'aide de la fonction `openVignette`.

3 Chargement des données

Nous utiliserons ici un sous-ensemble des échantillons issus de l'expérience GSE13425 stockée dans la base de données publique Gene Expression Omnibus (GEO). Les auteurs s'intéressent, dans cette expérience, à la caractérisation moléculaires des leucémies aiguës lymphoblastiques (Acute Lymphoblastic leukaemia, ALL) qui correspondent à la prolifération anormale, dans la moelle osseuse, d'un clone cellulaire, issu de la lignée lymphocytaire, et bloqué à un stade précis de différenciation. Les données ont été produites sur des microarrays Affymetrix de type HGU133A (Affymetrix Human Genome U133A Array). L'ensemble des informations concernant cette plate-forme est disponible sous l'identifiant "GPL96" sur le site de GEO (Gene expression Omnibus). Prenez quelques minutes pour vous renseigner sur ce type de microarrays.

Téléchargez les données (ici) puis décompressez le fichier.
Stockez les fichiers *.gz dans votre repertoire de travail.

- Ouvrez un terminal.
- Placez vous dans le repertoire contenant les données.
- Lancez R.
- Chargez la librairie affy

- Chargez les données dans un objet que vous nommerez `affy` à l'aide de la commande `ReadAffy`.

Si on appelle l'objet `affy`, il renvoie les informations suivantes :

```
> affy

AffyBatch object
size of arrays=712x712 features (21 kb)
cdf=HG-U133A (22283 affyids)
number of samples=13
number of genes=22283
annotation=hgu133a
notes=
```

- Combien de sondes ("features") contient la biopuces. A combien de jeux de sondes (probeset/genes) cela correspond-il ?
- Quel est la classe de l'objet que vous avez créé ?
- Regardez l'aide sur cet classe. Quels méthodes sont associées à cet classe ?

4 L'objet AffyBatch

Cet objet contient les champs suivants :

```
> slotNames(affy)

 [1] "cdfName"           "nrow"              "ncol"
 [4] "assayData"         "phenoData"         "featureData"
 [7] "experimentData"    "annotation"        "protocolData"
[10] ".__classVersion__"
```

Les champs contenus dans cet objet sont de natures très variées car il peut englober l'ensemble des données relatives à l'expérience.

- La matrice d'expression avec les échantillons en colonne et les gènes en ligne (objet `assayData`).
- les informations sur les sondes (objet `featureData`).
- Les phenotypes des échantillons (champ nommé `phenoData` et contenant un objet de type `AnnotatedDataFrame`).
- Le nom de l'experimentateur, son laboratoire d'appartenance, le descriptif de l'expérience... Champ nommé `experimentData` et contenant un objet de type `MIAME`.

- Le nombre physique de lignes et de colonnes dans la puce (objets `nrow` et `ncol`).
- Le nom de la plate-forme (champ `cdfName` contenant un vecteur de chaîne de caractères de taille 1).

5 Accéder au contenu d'un objet AffyBatch.

5.1 Principe

Vous pouvez accéder à un champs en utilisant l'opérateur @ ou la méthode associée, ici `cdfName()`. Ceci est vrai pour tous les objets de type S4 très fréquents dans Bioconductor.

```
> affy@cdfName
> cdfName(affy)
> isS4(affy)
```

5.2 Exemple : le slot assayData

C'est dans ce champ que se trouve les données d'expression. Ce champ correspond à un objet assez particulier puisqu'il s'agit d'un environnement. Un environnement est une partie réservée de la mémoire qui contient des éléments auxquels on peut accéder par leur noms. De manière générale, on ne s'occupera pas trop de ce détail et on utilisera tout simplement la méthode exprs() ou intensity() pour avoir accès aux valeurs d'expression.

```
> is(affy@assayData)

[1] "environment" "refObject"   "AssayData"

> m <- exprs(affy)
> head(m, 3)

  GSM338681.CEL.gz GSM338691.CEL.gz GSM338737.CEL.gz GSM338781.CEL.gz
1             83.0             64.3             69.3             51.5
2           6357.0           5504.0           4791.5           5001.5
3             99.5             67.5             76.3             56.0
  GSM338795.CEL.gz GSM338799.CEL.gz GSM338803.CEL.gz GSM338806.CEL.gz
1             79.0            207.3             63.3             66.5
2           4180.8           6626.8           5669.8           4814.5
3             76.8            209.3             92.3             66.5
  GSM338809.CEL.gz GSM338810.CEL.gz GSM338846.CEL.gz GSM338849.CEL.gz
1             89.3             82.3            109.5             72.5
2           5344.5           4934.3           5632.5           4624.0
3             96.3             73.8            107.0             79.3
```

```
  GSM338855.CEL.gz
1            110.8
2           6304.0
3            129.0
```

```
> rm(m)
```

6 Lecture des données de phénotypes

Par défaut la fonction ReadAffy ne lit pas les données phenotypiques associées à l'expérience. On peut les charger à l'aide de la fonction `read.AnnotatedDataFrame`. Cette fonction renvoie un objet de type de `AnnotatedDataFrame`. Etant donnée que le champ `phenoData` de l'objet `affy` est lui même un objet de type `AnnotatedDataFrame`, il suffit de lui assigner le résultat de cette fonction.

```
> link <- "ftp://tagc.univ-mrs.fr/public/Tagc/Denis/phenoData_sub.txt"
> pheno <- read.AnnotatedDataFrame(link)
> is(pheno)
> is(affy@phenoData)
> affy@phenoData <- pheno
> phenoData(affy)
> head(pData(affy))
> p <- pData(affy)
> fix(p)
```

7 Indexation d'un objet affyBatch

L'opérateur d'indexation "[" (qui est en fait une fonction), est défini dans le code source de la librairie affy. Cette définition stipule qu'il renvoie un objet de type affyBatch contenant l'ensemble des informations demandées. Si l'utilisateur demande $affy[, 1:2]$, l'objet renvoyé sera du même type (AffyBatch) mais contiendra des champs dont le contenu aura été mis à jour. Dans l'exemple suivant, selectionner deux microarrays permet aussi de selectionner des données d'expression ainsi que les données phénotypiques correspondantes.

```
> affy2 <- affy[, 1:2]
> pData(affy2)
```

8 Caractéristiques physiques de la puce HGU133a

- Vérifiez que le nombre de lignes dans la matrice d'expression correspond bien au nombre physique de cellules.

9 Outils graphiques de la librairie affy.

9.1 Image du microarray

- Visualisez les niveaux d'expression des cellules de la première GeneChip (`affy[,1]`) à l'aide de la commande image() (figures 1).

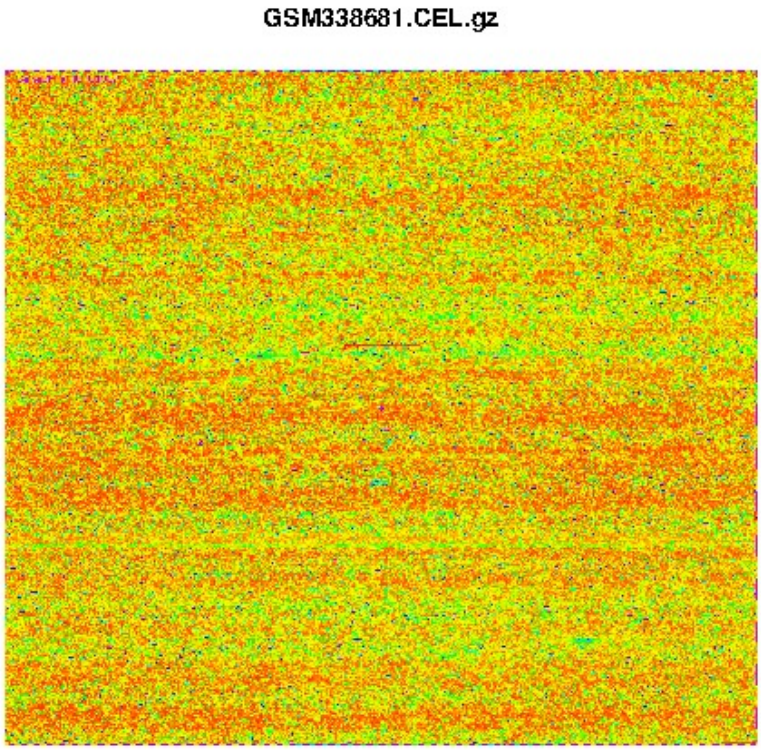

FIGURE 1 – Image avec code couleur d'une geneChip Affymetrix.

9.2 Les probesets

Les noms des probesets de la puce sont accessibles via la fonction `geneNames`.

```
> gn <- geneNames(affy)
> gn[1:10]
```

```
[1] "1007_s_at" "1053_at"   "117_at"    "121_at"    "1255_g_at" "1294_at"
[7] "1316_at"   "1320_at"   "1405_i_at" "1431_at"
```

La méthode `probeset` permet, à partir d'un identifiant affymetrix, d'extraire les valeurs d'intensité des sondes correspondantes.

- Créez un objet nommé `pS` qui contient les valeurs pour le probeSet dont l'identifiant est "209380_s_at".
- Utilisez la fonction `barplot.ProbeSet` pour représenter graphiquement ce probeSet (attention `probeset` renvoie une liste d'objets `ProbeSet` et barplot. ProbeSet prend comme argument un objet `ProbeSet`, pas une liste...). Le résultat (pour la puce 1) est donné en figure 2.

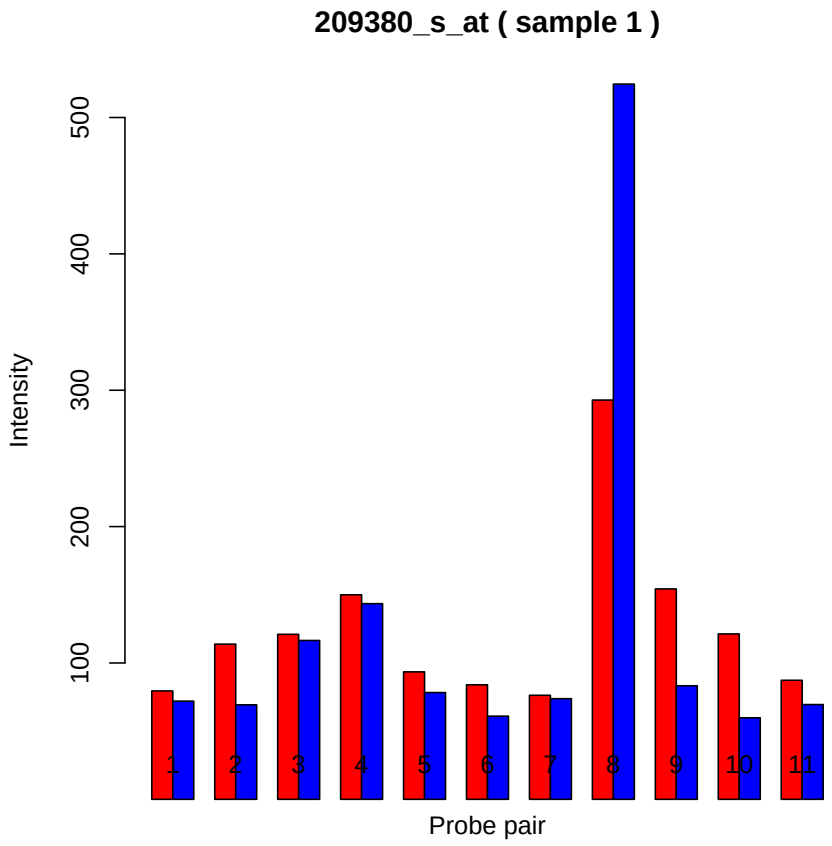

FIGURE 2 – Les valeurs PM et MM pour le probeset "209380_s_at".

Les séquences des sondes "Perfect-Match" peuvent être obtenues via la librairie `hgu133aprobe`.

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("hgu133aprobe")

> seq <- hgu133aprobe[hgu133aprobe$Probe.Set.Name == "209642_at",
+     ]
> seq
```

- Rendez vous sur le site de l'UCSC. En utilisant l'algorithme BLAT, verifiez que la première séquence se trouve bien dans le gène BUB1 qui correspond au jeu de sonde "209642_at".

10 Contrôle qualité des données brutes

11 Statistiques descriptives

Créez un objet `affyLog2` qui contiendra les données d'intensité en logarithme base 2 :

```
> affyLog2 <- log2(exprs(affy))
```

Appliquez à l'objet `affyLog2` :

- la fonction plotDensity pour visualiser la distribution des données de chacunes des biopuces.
- la fonction boxplot pour visualiser les valeur du 1er, 2ème et 3ème quartile.
- La fonction *cor* afin de calculer la matrice de corrélation échantillon-échantillon. Représentez le résultat avec la fonction levelplot (librairie `lattice`).

11.1 AffyRNAdeg

Sur les puces affymetrix, l'analyse des images, est parfois peu informative. La librairie affy propose un autre critère de qualité via la fonction `AffyRNAdeg`.

- Testez cette fonction et analysez le résultat (figure 3).

12 Quels sont les gènes qui s'expriment ?

Il peut être intéressant de se concentrer sur des gènes dont on sait qu'ils donnent un signal supérieur au bruit de fond au moins dans un certain nombre d'échantillons. La méthode de référence d'Affymetrix permet de deriver pour chaque valeur d'expression une valeur "A"

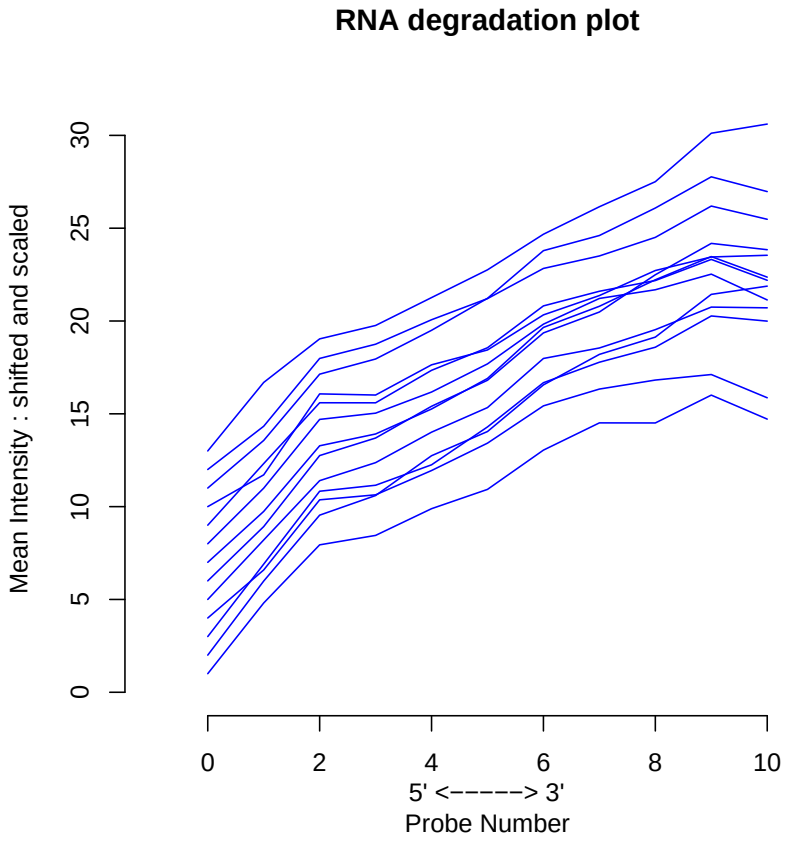

FIGURE 3 – Contrôle qualité des puces affymetrix via la fonction `AffyRNAdeg`.

(Absent), "M" (Marginal), ou "P" (Present). Cette méthode, qui est basé sur la comparaison des signaux emis par les Perfect Matchs et les Mismatchs, est implémenté dans la fonction `mas5calls`.

- Appliquez cette fonction à l'objet contenant les données brutes.
- Faites un diagramme à barres affichant pour chaque microarray le pourcentage de gènes détectés comme "Présents".

13 Normalisation des données

Il existe plusieurs algorithmes de normalisation (rma, PLIER, mas5, ...). Nous utiliserons la fonction rma (`rma`) pour normaliser les données. Elle applique l'algorithme de normalisation par les quantiles puis calcul pour chaque probeSet une valeur d'expression à partir des signaux émis par les sondes correspondantes. Seules les données de "Perfect-Match" sont utilisées. Les valeurs générées par rma sont en logarithme base 2.

```
> eset <- rma(affy)

Background correcting
Normalizing
Calculating Expression
```

- Quel type d'objet obtenez vous ? Quels sont les champs contenus dans cet objet ?
- Regardez l'aide sur cet objet.
- En utilisant la fonction *scatter.smooth* de la librairie geneplotter (installez la), faites un nuage de point pour comparer les valeurs normalisées des deux premiers microarrays.

14 L'objet ExpressionSet

Une grande partie des librairies de Bioconductor converge vers un objet de type ExpressionSet. Cette objet est assez simple et assez semblable à l'objet affyBatch. Il est conçu pour stocker des données normalisées issues de technologies diverses.

```
> is(eset)

[1] "ExpressionSet"    "eSet"             "VersionedBiobase" "Versioned"

> slotNames(eset)

[1] "assayData"         "phenoData"         "featureData"
[4] "experimentData"    "annotation"        "protocolData"
[7] ".__classVersion__"
```

15 Contrôle qualité sur les données normalisées

15.1 Boxplot

- Affichez le boxplot correspondant aux données normalisées.

15.2 Relative Log Expression (RLE)

Les valeurs de RLE sont obtenues en calculant, pour chaque valeur, sa différence à la médiane de la ligne correspondante. On considère que la plupart des probesets ne varient pas entre les conditions biologiques. Pour une biopuce on s'attend donc à obtenir des RLE moyens proches de zéro.

- Produisez les données de RLE et représentez les sous la forme d'une boite à moustache.

15.3 Graphique MvsA ("MA plot")

Dans l'analyse de données de biopuces, on utilise communément le "MA plot". On tente par cette représentation de mettre en évidence des biais se traduisant par une dépendance entre la valeur de ratio d'un gène et l'expression moyenne de celui-ci (e.g ; seules les gènes très forts sont induits).
Afin d'effectuer ce diagramme pour le microarray 1, nous générons tout d'abord un pseudo-microarray `ref` qui est très représentatif de l'expérience puiqu'il contient les valeurs médianes d'intensités de chacun des probesets :

```
> ref <- rowMedians(exprs(eset))
```

- Calculer $A_{1..n}$ et $M_{1..n}$ pour les échantillons 1 et *ref* sachant que pour un gène g avec des intensités $I_{g,1}$ et $I_{g,ref}$ le calcule s'effectue de la manière suivante :

$$A = (I_{g,1} + I_{g,ref})/2$$
$$M = I_{g,1} - I_{g,ref}$$

- Faites un nuage de points (figure 1) (A en abscisse, M en ordonnée) 4. Représentez les droites d'équations $M = 0$, $M = 1$ et $M = -1$ avec la fonction `abline`.
- A l'aide de la fonction *text*, affichez, sur le graphique, les noms des probesets pour lesquelles une valeur absolue de M supérieure à 4 est observée.
- Effectuez la régression locale de M par A en créant un objet à l'aide de la fonction *lowess*. Représentez le résultat sur le graphique MA à l'aide de la fonction `lines`.

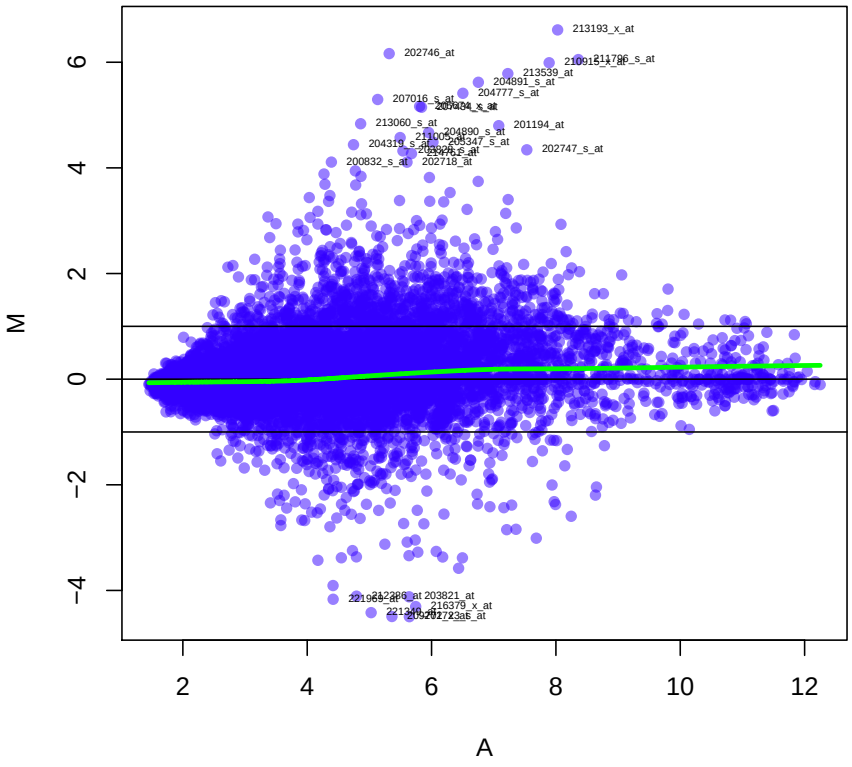

FIGURE 4 – MA plot pour l'échantillons 1.

16 Annotation des sondes

Dans Bioconductor, l'annotation associée aux sondes (qui est dynamique par définition) est contenue dans des librairies d'annotation (ex : la librairie hgu133a.db, hgu95av2.db,...). A chaque type de puce affymetrix correspond une annotation. Ici, il faut installer et charger l'annotation hgu133a.

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("hgu133a.db")
> library(hgu133a.db)
```

Cette librairie donne accès à diverses annotations que l'on peut lister via a fonction hgu133a.

```
> hgu133a()
```

```
Quality control information for hgu133a:


This package has the following mappings:

hgu133aACCNUM has 22283 mapped keys (of 22283 keys)
hgu133aALIAS2PROBE has 53200 mapped keys (of 110538 keys)
hgu133aCHR has 20267 mapped keys (of 22283 keys)
hgu133aCHRLENGTHS has 93 mapped keys (of 93 keys)
hgu133aCHRLOC has 20066 mapped keys (of 22283 keys)
hgu133aCHRLOCEND has 20066 mapped keys (of 22283 keys)
hgu133aENSEMBL has 19742 mapped keys (of 22283 keys)
hgu133aENSEMBL2PROBE has 12921 mapped keys (of 19887 keys)
hgu133aENTREZID has 20273 mapped keys (of 22283 keys)
hgu133aENZYME has 3002 mapped keys (of 22283 keys)
hgu133aENZYME2PROBE has 869 mapped keys (of 936 keys)
hgu133aGENENAME has 20273 mapped keys (of 22283 keys)
hgu133aGO has 19270 mapped keys (of 22283 keys)
hgu133aGO2ALLPROBES has 12901 mapped keys (of 13360 keys)
hgu133aGO2PROBE has 9648 mapped keys (of 10161 keys)
hgu133aMAP has 20229 mapped keys (of 22283 keys)
hgu133aOMIM has 16682 mapped keys (of 22283 keys)
hgu133aPATH has 7585 mapped keys (of 22283 keys)
hgu133aPATH2PROBE has 214 mapped keys (of 214 keys)
hgu133aPFAM has 20157 mapped keys (of 22283 keys)
hgu133aPMID has 20197 mapped keys (of 22283 keys)
```

```
hgu133aPMID2PROBE has 266196 mapped keys (of 283543 keys)
hgu133aPROSITE has 20157 mapped keys (of 22283 keys)
hgu133aREFSEQ has 20182 mapped keys (of 22283 keys)
hgu133aSYMBOL has 20273 mapped keys (of 22283 keys)
hgu133aUNIGENE has 20232 mapped keys (of 22283 keys)
hgu133aUNIPROT has 19681 mapped keys (of 22283 keys)


Additional Information about this package:

DB schema: HUMANCHIP_DB
DB schema version: 2.1
Organism: Homo sapiens
Date for NCBI data: 2010-Sep7
Date for GO data: 20100904
Date for KEGG data: 2010-Sep7
Date for Golden Path data: 2010-Mar22
Date for IPI data: 2010-Aug19
Date for Ensembl data: 2010-Aug5
```

On peut utiliser la syntaxe suivante pour récupérer les symboles de gènes de la puce hgu133a.

```
> gn <- geneNames(affy)
> head(gn)

[1] "1007_s_at" "1053_at"   "117_at"    "121_at"    "1255_g_at" "1294_at"

> gn[1]

[1] "1007_s_at"

> mget(gn[1:4], env = hgu133aSYMBOL)

$`1007_s_at`
[1] "DDR1"

$`1053_at`
[1] "RFC2"

$`117_at`
[1] "HSPA6"
```

```
$`121_at`
[1] "PAX8"
```

Créez un objet `m` qui contiendra la matrice d'expression normalisée. En utilisant la fonction `paste` modifiez les noms des lignes afin que chacune contienne le nom du probeset ainsi que le symbole du gène correspondant (utilisez comme séparateur le caractère "|").

17 Sortie disque

On peut sauvegarder tout l'objet à l'aide de la fonction `save` ou la matrice d'expression (`write.table`) pour poursuivre l'analyse ultérieurement. Ce que nous ferons au prochain cours...

```
> save(eset, file = "eset.Rdata")
> write.table(exprs(eset), "GSE3254_norm.txt", sep = "t", col.names = NA,
+     quote = F)
```