The PDNN model for the *affy* package

Laurent Gautier

October 22, 2008

1 Introduction

This package is our implementation of the PDNN model[**?**]. Whenever you use it, you can aknowledge it by quoting our published work:

```
> citation(package = "affypdnn")

  Nielsen HB, Gautier L, Knudsen S. Implementation of a gene expression
  index calculation method based on the PDNN model. Bioinformatics.
  2005 Mar 1;21(5):687-8. PMID: 15509605

A BibTeX entry for LaTeX users is

  @Article{,
    author = {Henrik Bjorn Nielsen and Laurent Gautier and Steen Knudsen},
    title = {Implementation of a gene expression index calculation method based on the PDNN m
    journal = {Bioinformatics},
    volume = {21},
    number = {5},
    year = {2005},
    pages = {687--688},
    publisher = {Oxford University Press},
    address = {Oxford, UK},
  }
```

This package is also briefly described in Chapter 'Preprocessing High-density Oligonucleotide Arrays' of the Bioconductor book.

The first thing to do is to attach the package to the current R session.

```
> library(affypdnn)
```

Upon executing this command, the package and its dependencies will be attached (which will cause few lines of text to appear on the console for your R session).

Throughout this presentation of the package, we will use 'Dilution' dataset. The reader can replace it with an arbitrary instance of class *AffyBatch*.

```
> library(affydata)
> data(Dilution)
> afbatch <- Dilution
```

We decomposed the model of Zhang *et al.* in such a way that it fits the simple framework for probe-level data processing implemented in the *affy* package:

1. Chip type-specific parameters are computed
2. Experiment-specific parameters are computed
3. Transform the PM probe signal using the PDNN model (two flavours: 'pdnn' and 'pdnnpredict').
4. Compute probeset-level expression indexes

2 Chip type-specific parameters

Chip type-specific parameters for *U95Av2* are included with the package, mostly to make some examples shorter to run:

```
> data(hgu95av2.pdnn.params)
> params.chiptype <- hgu95av2.pdnn.params
```

Here we are showing how to compute them.

Currently one needs an external data file, called 'energy data file'. This file contains parameters for all possible 16 dinucleotides (E_g and E_n), as well as other parameters (W_g and W_n).

These files were downloaded, and there is currently no implementation to compute these data in this R package. The files included within the package are:

```
> dir(system.file("exampleData", package = "affypdnn"))
```

```
[1] "pdnn-energy-parameter_hg-u133a.txt"  "pdnn-energy-parameter_hg-u95av2.txt"
[3] "pdnn-energy-parameter_mg-u74av2.txt"
```

The Dilution dataset is of chip-type *HGU95Av2*. We read the 'energy data file', then compute the parameters (note that the probe package is needed):

```
energy.file <- system.file("exampleData",
                           "pdnn-energy-parameter_hg-u95av2.txt",
                           package = "affypdnn")

params.chiptype <- pdnn.params.chiptype(energy.file,
                                        probes.pack = "hgu95av2probe")
```

3 Experiment-specific parameters

Parameters specific to an experiement, that is the probe-level values in a CEL file, can be computed easily:

```
params <- find.params.pdnn(afbatch, hgu95av2.pdnn.params)
```

4 Transform the PM probe-level signal

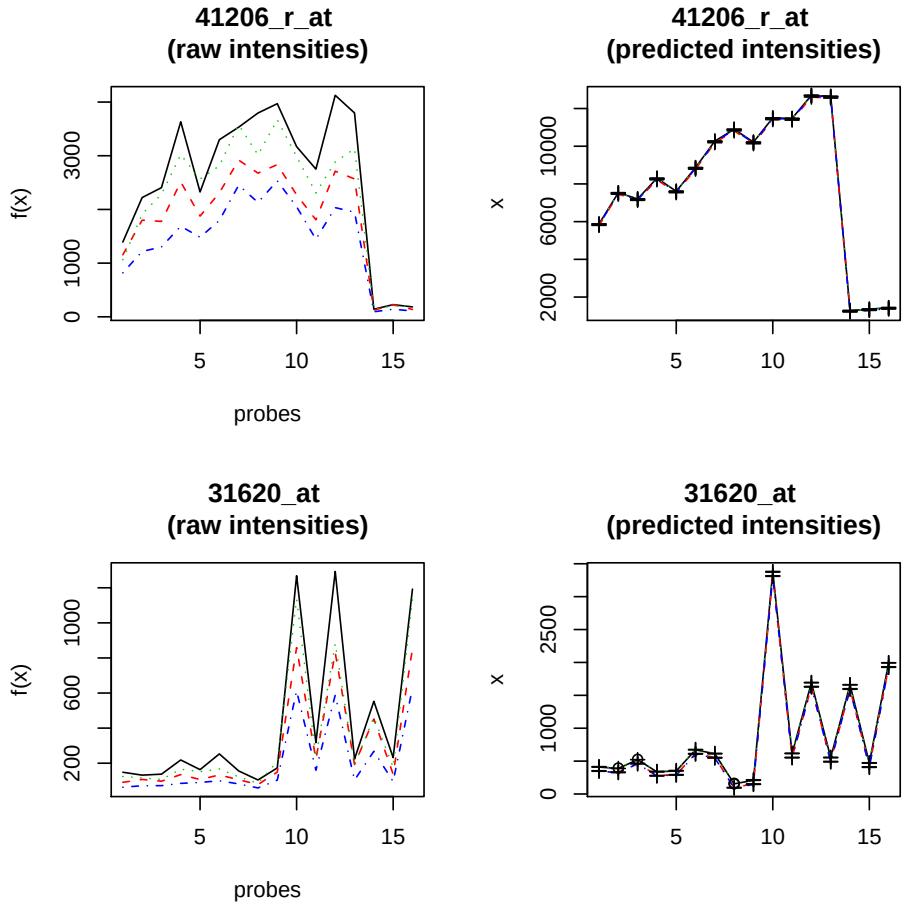
Here we arbitrarily pick two probesets:

```
> ppset.name <- c("41206_r_at", "31620_at")
> ppset <- probeset(afbatch, ppset.name)
```

Computing the transformed the PM probe-level signals is then just a matter of calling one of the functions:

- pmcorrect.pdnnpredict
- pmcorrect.pdnn

```
> par(mfrow = c(2, 2))
> for (i in 1:2) {
+     probes.pdnn <- pmcorrect.pdnnpredict(ppset[[i]], params,
+         params.chiptype = params.chiptype)
+     plot(ppset[[i]], main = paste(ppset.name[i], "\n(raw intensities)"))
+     matplotProbesPDNN(probes.pdnn, main = paste(ppset.name[i],
+         "\n(predicted intensities)"))
+ }
```



5 expressopdnn

Processing probe-level data can be done by using a modified[1] version of the function `expresso` in the *affy* package.

Like its *affy* counterpart, `expressopdnn` is a simple wrapper around a sequence of preprocessing steps. The example below shows a typical usage of it; the documentation page can be referred to for an exhaustive description of the parameters it accepts.

Here we take only ten probesets:

```
> ids <- ls(getCdfInfo(afbatch))[1:10]
> eset <- expressopdnn(afbatch, bg.correct = FALSE, normalize = FALSE,
+     findparams.param = list(params.chiptype = params.chiptype,
+         give.warnings = FALSE), summary.subset = ids)

PM/MM correction : pdnn
expression values: pdnn
```

[1]The design of the function `expresso` showed its limitations with the requirements for this one package, and a slightly modified version had to be written.

```
initializing data structure...done.
dealing with CEL 1 :
  step 1...done.
  step 2...done.
dealing with CEL 2 :
  step 1...done.
  step 2...done.
dealing with CEL 3 :
  step 1...done.
  step 2...done.
dealing with CEL 4 :
  step 1...done.
  step 2...done.
10 ids to be processed
|                    |
|####################|
```

One can note that we leave background correction and normalization aside, but it is obviously possible to mix-and-match with any such method available.