

CNVassoc: Association analysis of CNV data

Juan R Gonzalez, Isaac Subirana

June 25, 2009

Center for Research in Environmental Epidemiology (CREAL)
Institut Municipal d'Investigació Mèdica (IMIM)

`jrgonzalez@creal.cat`
<http://www.creal.cat/jrgonzalez/software.htm>

Contents

1	Introduction	1
2	CNV from a single probe	2
2.1	The data	2
2.2	Inferring copy number status from signal	5
2.2.1	From univariate signal intensity	5
2.2.2	From other algorithms	6
2.3	Summarizing information	6
2.4	Measuring uncertainty in inferring copy number status	10
2.5	Assessing association between CNV and disease	10
2.5.1	Modelling association	11
2.5.2	Testing association	14
3	CNV from aCGH	17

1 Introduction

CNVassoc allows users to perform association analysis between CNVs and disease incorporating uncertainty of CNV genotype. This document pro-

vides an overview on the usage of the `CNVassoc` package. For more detailed information on the model and assumption please refer to article (2) and its supplementary material. We illustrate how to analyze CNV data by using some real data sets. The first data set belongs to a case-control study where peaks intensities of MLPA assays were obtained for two different genes. The second example corresponds to the Neve dataset (3) that is available at Bioconductor. The data consists on 50 CGH arrays of 1MB resolution for patients diagnosed with breast cancer. All datasets are available directly from the `CNVassoc` package.

Start by loading the package `CNVassoc`:

```
> library(CNVassoc)
```

```
use of mclust requires a license agreement  
see http://www.stat.washington.edu/mclust/license.txt
```

2 CNV from a single probe

2.1 The data

In order to illustrate how to assess association between CNV and disease, we use a data set including 360 cases and 291 controls. Data is to be published soon as described in (2). The data contains peaks intensities for two genes arising from a MLPA assay. Note that Illumina or Affymetrix data, where \log_2 ratios are available instead of peak intensities, can be analyzed in the same way as we are illustrating.

MLPA data set contains case control status as well as two simulated covariates (`quanti` and `cov`) that have been generated for illustrating purposes (e.g., association between a quantitative trait and CNV or how to adjust for covariates). To load the MLPA data just type

```
> data(dataMLPA)  
> head(dataMLPA)
```

	id	casco	Gene1	Gene2	PCR.Gene1	PCR.Gene2	quanti	cov
1	H238	1	0.51	0.5385080	wt	wt	-0.61	10.83
2	H238	1	0.45	0.6392029	wt	wt	-0.13	10.69
3	H239	1	0.00	0.4831572	del	wt	-0.57	9.63
4	H239	1	0.00	0.4640072	del	wt	-1.40	9.87
5	H276	1	0.00	0.0000000	del	del	0.83	10.25
6	H276	1	0.00	0.0000000	del	del	-2.07	10.40

First, let us see the distribution of the peak intensities for each of the two genes analyzed that is given in Figure 1. It can be obtained by executing

```
> pdf("./figures/fig1.pdf")
> par(mfrow = c(2, 2), mar = c(3, 4, 3, 1))
> hist(dataMLPA$Gene1, main = "gene 1 signal histogram", xlab = "",
+      ylab = "frequency")
> hist(dataMLPA$Gene2, main = "gene 2 signal histogram", xlab = "",
+      ylab = "frequency")
> par(xaxs = "i")
> plot(density(dataMLPA$Gene1), main = "gene 1 signal density function",
+      xlab = "", ylab = "density")
> plot(density(dataMLPA$Gene2), main = "gene 2 signal density function",
+      xlab = "", ylab = "density")
> dev.off()

null device
      1
```

Figure 1 shows the signals for gene 1 and gene 2. For both genes it is clear that there are 3 clusters corresponding to 0, 1 and 2 copies. However, the three peaks for gene 2 are not so well separated as those of gene 1 (e.g., underlying distributions are much more overlapped). This fact leads to more uncertainty when inferring the copy number status for each individual. This will be illustrated in the next section.

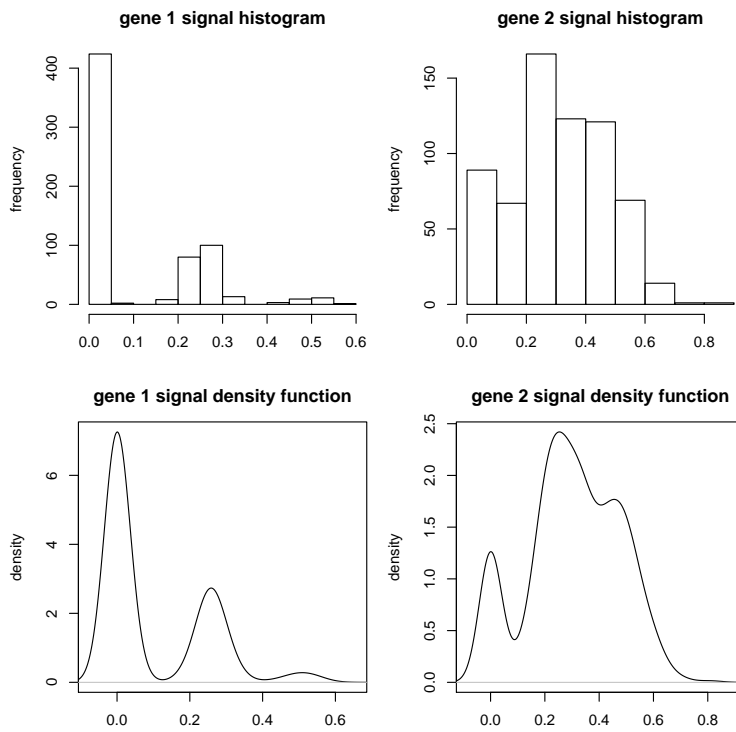


Figure 1: Signal distribution for gene 1 and gene 2

2.2 Inferring copy number status from signal

2.2.1 From univariate signal intensity

The `cnv` function is used to infer the copy number status for each subject using the quantitative signal for an individual probe. This signal can be obtained from any platform (MLPA, Illumina, ...).

This function assumes a normal mixture model like other authors have proposed in the context of aCGH (4; 5). It should be pointed out that in some instances, the intensity distributions (see Gene 1 in Figure 1) for a null allele is meant to be equal to 0. Due to experimental noise this intensity may show values that slightly deviate from this theoretical value. For these cases, the normal mixture model fails because the underlying distribution of individuals with 0 copies is not normal. In these situations we fit a modified mixture model (see (2) for further details).

Figure 1 shows two distinctly different scenarios. For the Gene 1 it is clear that there are three different status, whilst for the Gene 2 this is not.

In order to consider all these issues, the function `cnv` has different arguments. The calling for the Gene 1 can be done by executing

```
> CNV.1 <- cnv(x = dataMLPA$Gene1, threshold.0 = 0.06, num.class = 3,
+             mix.method = "mixdist")
```

The argument `threshold.0=0.06` indicates that individuals with peak intensities lower than 0.06 will have 0 copies. Since there are three underlying copy number status, we set the argument `num.class` equal to 3. The argument `mix.method` indicates the algorithm to be used to estimate the normal mixture model. "mixdist" uses a combination of a Newton-type method and EM algorithm implemented in the library `mixdist`, while "mclust" uses EM algorithm implemented in the library `Mclust`.

In the case of not knowing the exact number of components for the mixture model (as it could be in the case of Gene 2), the function uses the Bayesian Information Criteria (BIC) to select the number of components. This is performed when the argument `num.class` is missing. In this case the function estimates the mixture model for 2 up to 6 copy number status.

```
> CNV.2 <- cnv(x = dataMLPA$Gene2, threshold.0 = 0.01, mix.method = "mixdist")
> CNV.2
```

```
Inferred copy number variant by a quantitative signal
Method: function mix {package: mixdist}
```

```
Number of individuals: 651
Copies 0, 1, 2
Estimated means: 0, 0.2435, 0.4469
Estimated variances: 0, 0.0041, 0.0095
Estimated proportions: 0.1306, 0.4234, 0.446
Goodness-of-fit test: p-value= 0.4887659
```

-. Note: number of classes has been selected using the best BIC

As we can see, the best model has 3 copy number status. The result obtained by using BIC is what we expected because we already know that this gene has 0, 1 and 2 copies (see (2)).

2.2.2 From other algorithms

The result of applying `cnv` function is an object of class `cnv` that, among other things, contains the posterior probabilities matrix for each individual. This information is then used in the association analysis where the uncertainty is taken into account. Posterior probabilities from any other calling algorithms can also be encapsulated in a `cnv` object to be further used in the analysis.

To illustrate this, we will use the posterior probability matrix that has been computed when inferring copy number for gene 2 by using the normal mixture model. This information is saved as an attribute for an object of class `cnv`. Save this matrix in an object called `probs.2`.

```
> probs.2 <- attr(CNV.2, "probabilities")
```

Imagine that `probs.2` contains posterior probabilities obtained from any calling algorithm such as `CANARY` (from `PLINK`) or `GCHcall` (this will be further illustrated in Section 3). In this case, we create the object of class `cnv` that will be used in the association step by typing

```
> CNV.2probs <- cnv(probs.2)
```

2.3 Summarizing information

We have implemented two generic functions for an object of class `cnv`. The generic `print` function gives the results from inferred copy number status.

It includes the means, variances and proportions of copy number clusters as well as the p value corresponding to the goodness-of-fit test for the selected number of classes.

```
> CNV.1
```

```
Inferred copy number variant by a quantitative signal  
  Method: function mix {package: mixdist}
```

```
Number of individuals: 651  
Copies 0, 1, 2  
Estimated means: 0, 0.2543, 0.4958  
Estimated variances: 0, 9e-04, 0.0012  
Estimated proportions: 0.6544, 0.3087, 0.0369  
Goodness-of-fit test: p-value= 0.6615318
```

and for the gene 2

```
> CNV.2
```

```
Inferred copy number variant by a quantitative signal  
  Method: function mix {package: mixdist}
```

```
Number of individuals: 651  
Copies 0, 1, 2  
Estimated means: 0, 0.2435, 0.4469  
Estimated variances: 0, 0.0041, 0.0095  
Estimated proportions: 0.1306, 0.4234, 0.446  
Goodness-of-fit test: p-value= 0.4887659
```

-. Note: number of classes has been selected using the best BIC

This information is different when only posterior probabilities are given

```
> CNV.2probs
```

```
Copy number variant  
  Input data: called probabilities  
Number of individuals: 651  
Copies 0, 1, 2  
Estimated proportions: 0.1306, 0.4234, 0.446
```

Figure 2 shows the result of the generic plot function

```
> pdf("./figures/fig2a.pdf")
> plot(CNV.1, case.control = dataMLPA$casco, main = "gen 1")
> dev.off()

null device
      1

> pdf("./figures/fig2b.pdf")
> plot(CNV.2, case.control = dataMLPA$casco, main = "gen 2")
> dev.off()

null device
      1
```

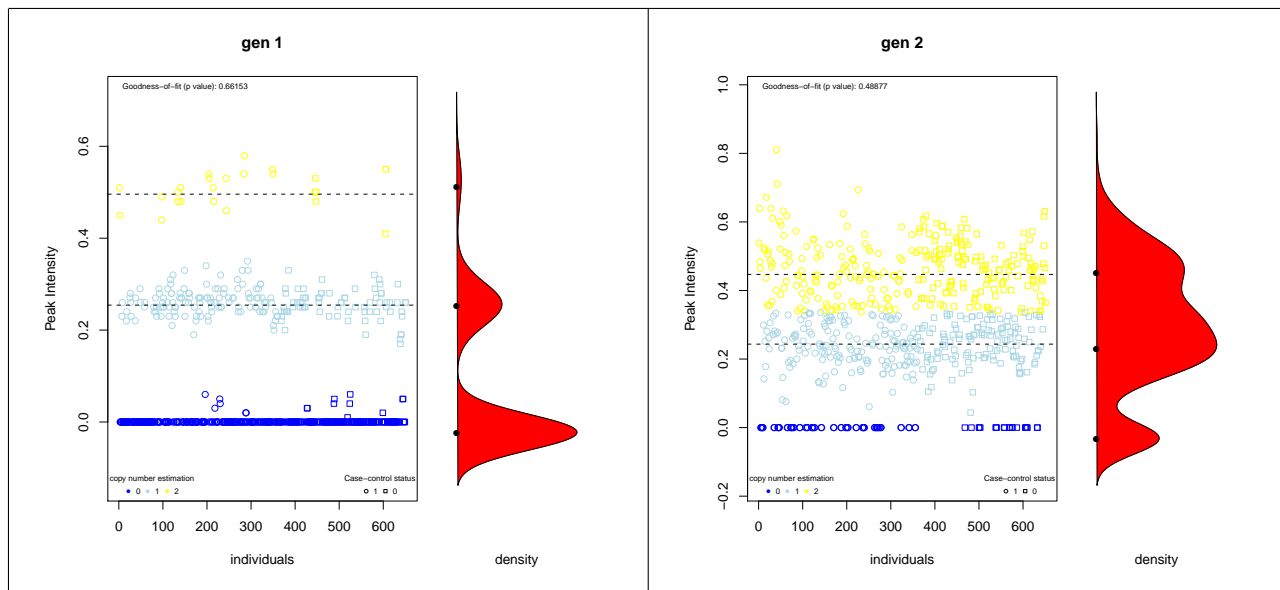


Figure 2: Signal distribution by case control, and inferred number of copies

In the figure 2 the signal is coloured by the inferred (most probable) copy number, distinguishing between the case and control status. This last option is specified by the argument `case.control`. On the right side of the plot, a density function of signal distribution is drawn. The p-value of goodness-of-fit test is the same as this described in Section 2.3. It indicates whether

the assumed normal mixture model (with a given number of components) is correct or not. Notice that for both genes the intensity data fits well to our the model (goodness-of-fit p-values > 0.1).

The action of `plot` when only posterior probabilities are available gives a different result (Figure 3). Two barplots are created for cases and controls (when argument `case.control` is used). Each one is splitted by the copy number frequency.

```
> pdf("./figures/fig3.pdf")  
> plot(CNV.2probs, case.control = dataMLPA$casco)  
> dev.off()
```

```
null device
```

```
1
```

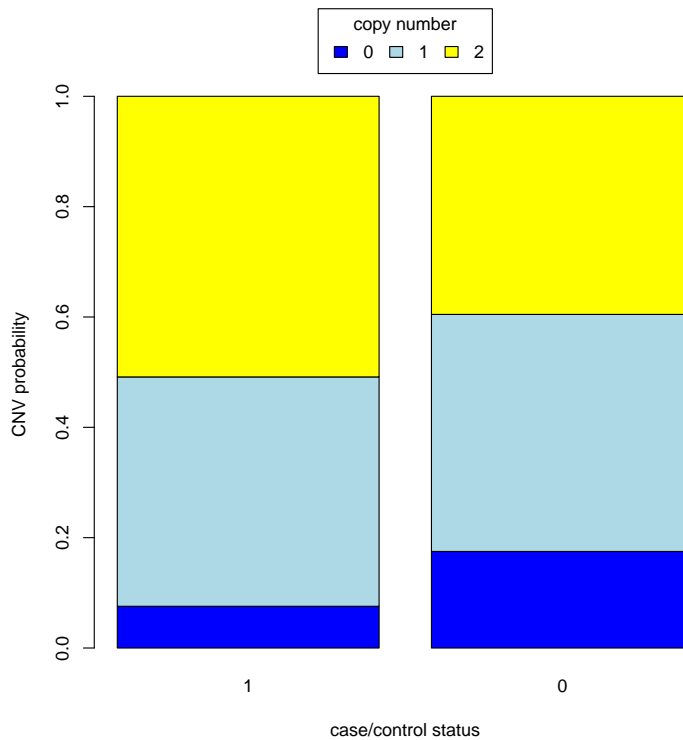


Figure 3: Estimated copy number frequencies for gene 1 and gene 2

2.4 Measuring uncertainty in inferring copy number status

The function `getQualityScore` uses the information from an object of class `cnv` to compute a value that indicates how much the underline copy number distribution (peaks intensities) are mixed or overlapped. The more separated these peaks are (less uncertainty), the larger the quality score is.

Two measures of uncertainty are implemented. The first one is the one defined in `CNVtools` package, and the second is the estimated probability of good classification (PBC). To choose PBC method type

```
> getQualityScore(CNV.1, type = "class")
--Probability of good classification: 0.9998963
> getQualityScore(CNV.2, type = "class")
--Probability of good classification: 0.9081534
```

And the measure defined in `CNVtools` package is obtained with:

```
> getQualityScore(CNV.1, type = "CNVtools")
--CNVtools Quality Score: 25.16956
> getQualityScore(CNV.2, type = "CNVtools")
--CNVtools Quality Score: 3.063055
```

It is clear that in gene 1 there is much less uncertainty, because the PBC is bigger than 99%, and the measure of `CNVtools` package is higher than 25 (`CNVtools` recommends a quality score of 4 or larger). This fact can also be seen in the Figure 2 where the underline copy number intensities distributions are very well separated. On the other hand, the PBC for gene 2 is 91.3% , and the `CNVtools` package value is about 3 indicating that more uncertainty is present.

2.5 Assessing association between CNV and disease

The function `CNVassoc` carries out association analysis between CNV and disease. This function incorporates calling uncertainty by using a latent class model as described in (2). The function can analyze both binary or quantitative traits. In the first case, it performs a linear regression and, in

the second, a logistic one. The regression model can be selected by using the argument `case.control`. Nonetheless, the program automatically detects whether or not a quantitative trait is analyzed and it is not necessary to be specified.

The function also allows the user to fit a model with additive or multiplicative effects of CNV. This can be set through the argument `model`. Possible values are "add" for additive effect or "mult" for multiplicative effect.

The function `CNVassoc` returns an object of class `CNVassoc`. This class of objects share some properties in common with objects of class `glm` like `coef` or `summary` among others

2.5.1 Modelling association

The effect of a given CNV on case/control status (`casco` variable) can be fitted by typing

```
> model1mul <- CNVassoc(casco ~ CNV.1, data = dataMLPA, model = "mul")
> model2mul <- CNVassoc(casco ~ CNV.2, data = dataMLPA, model = "mul")
```

By default, a short summary is printed (similar to `glm` objects)

```
> model1mul

Call: CNVassoc(formula = casco ~ CNV.1, data = dataMLPA, model = "mul")

Coefficients:
           CNV0           CNV1           CNV2
CNVmult  0.0281709  0.5185469  1.1003268

Number of estimated parameters: 3
Deviance: 883.021

> model2mul

Call: CNVassoc(formula = casco ~ CNV.2, data = dataMLPA, model = "mul")

Coefficients:
           CNV0           CNV1           CNV2
CNVmult  1.0520923  0.3095268 -0.0988672

Number of estimated parameters: 3
Deviance: 876.384
```

Note that the coefficients are a matrix with one row per variable and as many columns as copy number status. In this model, because there are no covariates and the CNV has a multiplicative effect, there is just one row (one intercept) and this is different among columns (copy number status).

By using the generic function `summary` we can obtain a more exhaustive output. In particular the odds ratio and its confidence intervals are printed as well as its p-value.

```
> summary(model1mul)
```

Call:

```
CNVassoc(formula = casco ~ CNV.1, data = dataMLPA, model = "mul")
```

Deviance: 883.0211

Number of parameters: 3

Coefficients:

	OR	lower.lim	upper.lim	SE	stat	pvalue
CNV0	1.0000					
CNV1	1.6329	1.1585	2.3016	0.1751	2.8003	0.005
CNV2	2.9217	1.1376	7.5038	0.4813	2.2278	0.026

(Dispersion parameter for binomial family taken to be 1)

Covariance between coefficients:

	CNV0	CNV1	CNV2
CNV0	0.0094	0.0000	0.0000
CNV1		0.0213	0.0000
CNV2			0.2222

```
> summary(model2mul)
```

Call:

```
CNVassoc(formula = casco ~ CNV.2, data = dataMLPA, model = "mul")
```

Deviance: 876.3842

Number of parameters: 3

Coefficients:

	OR	lower.lim	upper.lim	SE	stat	pvalue
--	----	-----------	-----------	----	------	--------

```

CNV0  1.0000
CNV1  0.4759    0.2738    0.8272    0.2821   -2.6323    0.008
CNV2  0.3163    0.1829    0.5470    0.2794   -4.1194   3.8e-05

```

(Dispersion parameter for binomial family taken to be 1)

Covariance between coefficients:

```

      CNV0    CNV1    CNV2
CNV0  0.0613  0.0000  0.0000
CNV1         0.0183 -0.0031
CNV2                0.0167

```

In the case of analyzing a quantitative trait or adjusting the association by other covariates, the same function can be used as following

```

> mod <- CNVassoc(quant_i ~ CNV.2 + cov, data = dataMLPA, model = "add",
+   emsteps = 10)
> mod

```

Call: CNVassoc(formula = quant_i ~ CNV.2 + cov, data = dataMLPA, model = "add",

Coefficients:

```

      CNV0      CNV1      CNV2
intercept -0.1401115 -0.1401115 -0.1401115
CNVadd    -0.0796270 -0.0796270 -0.0796270
cov        0.0241753  0.0241753  0.0241753

```

Number of estimated parameters: 4

Deviance: 1824.56

Notice that in this case, we use another extra argument called `emsteps`. This is necessary for computational reasons. After performing some preliminary steps using EM algorithm it makes it easier to maximize the likelihood function using Newton-Raphson procedure. In general, it is enough to perform few iterations (no more than 10). As usual, the model is then summarized by typing

```

> summary(mod)

```

```
Call:
CNVassoc(formula = quanti ~ CNV.2 + cov, data = dataMLPA, model = "add",
          emsteps = 10)
```

```
Deviance: 1824.560
Number of parameters: 4
```

```
Coefficients:
              beta lower.lim upper.lim      SE      stat pvalue
(Intercept) -0.14011 -0.90658  0.62635  0.39106 -0.35829  0.720
trend        -0.07963 -0.19765  0.03839  0.06022 -1.32237  0.186
cov           0.02418 -0.05069  0.09904  0.03820  0.63289  0.527
```

```
(Dispersion parameter for gaussian family taken to be 0.9650073 )
```

```
Covariance between coefficients:
```

```
          intercept CNVadd cov
intercept 0.1529   -0.0041 -0.0146
CNVadd           0.0036 -0.0001
cov                          0.0015
```

Notice that for quantitative traits we obtain mean differences instead of odds ratios.

2.5.2 Testing association

In the previous analysis we obtained p values corresponding to the comparison between every copy number status versus the reference one (zero copies). Nonetheless, we are normally interested in testing the overall effect of CNV on disease. Two statistical tests are implemented to perform such omnibus test: Wald test and likelihood ratio test (LRT). Both test are carried out by using `CNVtest` function which requires an object of class `CNVassoc` as the input. To specify the type of test, set the argument `type` to "Wald" or "LRT", respectively. For the gene 1,

```
> CNVtest(modelmul, type = "Wald")
```

```
----CNV Wald test----
```

```
Chi= 11.55957 (df= 2 ) , pvalue= 0.003089375
```

```
> CNVtest(model1mul, type = "LRT")

----CNV Likelihood Ratio Test----
Chi= 12.12937 (df= 2 ) , pvalue= 0.002323488
```

and for ghe gene 2,

```
> CNVtest(model2mul, type = "Wald")

----CNV Wald test----
Chi= 17.34114 (df= 2 ) , pvalue= 0.0001715614
```

```
> CNVtest(model2mul, type = "LRT")

----CNV Likelihood Ratio Test----
Chi= 18.76629 (df= 2 ) , pvalue= 8.413033e-05
```

Other generic functions like `logLik`, `coef`, `summary` or `update` can be used for an object of class `CNVassoc` to get more information.

For a multiplicative CNV effect model and for a binary traits, it is possible to change the reference category of copy number status. This can be done by using the argument `ref` when executing the `summary` function. For example, if we want to set individuals with one copy as the reference category just type:

```
> coef(summary(model1mul, ref = 2))
```

	OR	lower.lim	upper.lim	SE	stat	pvalue
CNV1	1.000000	NA	NA	NA	NA	NA
CNV0	0.612396	0.4344842	0.863159	0.1751153	-2.800304	0.005105448
CNV2	1.789220	0.6801587	4.706709	0.4934831	1.178926	0.238427714

The same kind of results can be obtained if we assume an additive effect of CNV on the trait. In this case we need to set the argument `model` equal to `"add"`

```
> model2add <- CNVassoc(casco ~ CNV.2, data = dataMLPA, model = "add")
> model2add
```

```
Call: CNVassoc(formula = casco ~ CNV.2, data = dataMLPA, model = "add")
```

Coefficients:

	CNV0	CNV1	CNV2
intercept	0.929636	0.929636	0.929636
CNVadd	-0.537912	-0.537912	-0.537912

Number of estimated parameters: 2
 Deviance: 877.07

Notice that under an additive CNV effect the structure of coefficients are different from the multiplicative CNV effect. Now there are two rows, one for intercept and the other one for the slope. (change of risk in increasing one copy). And this two values remain constant for every column (copy number status).

```
> summary(model2add)
```

Call:

```
CNVassoc(formula = casco ~ CNV.2, data = dataMLPA, model = "add")
```

Deviance: 877.0699

Number of parameters: 2

Coefficients:

	OR	lower.lim	upper.lim	SE	stat	pvalue
trend	0.5840	0.4529	0.7530	0.1297	-4.1477	3.36e-05

(Dispersion parameter for binomial family taken to be 1)

Covariance between coefficients:

	intercept	CNVadd
intercept	0.0372	-0.0228
CNVadd		0.0168

Finally, one might be interested in testing the additive effect. To do this, one can compare both additive and multiplicative models. It is straightforward to see that the additive model is a particular case of the multiplicative one, and therefore the first is nested in the second one.

To compare two nested models we use the generic function `anova` (NOTE: it is only implemented for comparing two fitted models with `CNVassoc` function).

```

> anova(model2mul, model2add)

--- Likelihood ratio test comparing 2 CNVassoc models:

Model 1 call: CNVassoc(formula = casco ~ CNV.2, data = dataMLPA, model = "mul")
Model 2 call: CNVassoc(formula = casco ~ CNV.2, data = dataMLPA, model = "add")

Chi= 0.6856273 (df= 1 ) p-value= 0.4076557

```

Note: the 2 models must be nested, and this function doesn't check this!

The likelihood ratio test is performed. In this case the p-value is not significant, indicating that an additive CNV effect can be assumed. In any case, one should consider the power of this test before making conclusions.

3 CNV from aCGH

The analysis of aCGH data requires additional steps to take into account, due to the dependency across probes and the fact that CNVs are not measured with a unique probe. Table 1 shows four steps we recommend for the analysis of this kind of data. First, posterior probabilities should be obtained with an algorithm that considers probe correlation. We use, in particular, the `CGHcall` R program which includes a mixture model to infer CNV status (5). Second, we build blocks/regions of consecutive clones with similar signatures. To perform this step the `CGHregions` R library was used (6). Third, the association between the CNV status of blocks and the trait is assessed by incorporating the uncertainty probabilities in `CNVassoc` function. And fourth, corrections for multiple comparisons must be performed. We use the Benjamini-Hochberg (BH) correction (1). This is a heuristic method that is robust against positive dependence and increasingly conservative as correlation increases.

We illustrate the methodology to the breast cancer data studied by Neve et al. (3). The data consists on CGH arrays of 1MB resolution and it is available from Bioconductor <http://www.bioconductor.org/>. The authors chose the 50 samples that could be matched to the name tokens of caArrayDB data (June 9th 2007). In this example the association between estrogen receptor positivity (dichotomous variable; 0: negative, 1: positive) and CNVs was tested. The original data set contained 2621 probes which

Table 1: Steps to assess association between CNVs and traits for aCGH

-
-
- Step 1.** Use any aCGH calling procedure that provides posterior probabilities (uncertainty) (`CGHcall`)
 - Step 2.** Build blocks/regions of consecutive probes with similar signatures (`CGHregions`)
 - Step 3.** Use the signature that occurs most in a block to perform association(`multiCNVassoc`)
 - Step 4.** Correct for multiple testing considering dependency among signatures (`getPvalBH`)
-
-

were reduced to 459 blocks after the application of `CGHcall` and `CGHregions` functions as we illustrate bellow.

The data is saved in an object called `NeveData`. This object is a list with two components. The first component corresponds to a dataframe containing 2621 rows and 54 columns with aCGH data (4 columns for the annotation and 50 log2ratio intensities). The second component is a vector with the phenotype analyzed (strogen receptor posistivity). The data can be loaded as usual

```
> data(NeveData)
> intensities <- NeveData$data
> pheno <- NeveData$pheno
```

The calling can be performed using `CGHcall` package by using the following instructions:

```
\dontrun{
#####
### chunk number 1: Class of aCGH data
#####
library(CGHcall)
Neve <- cghRaw(intensities)

#####
### chunk number 2: Preprocessing
#####
cghdata <- preprocess(Neve, maxmiss=30, nchrom=22)
```

```
#####
### chunk number 3: Normalization
#####
norm.cghdata <- normalize(cghdata, method="median", smoothOutliers=TRUE)

#####
### chunk number 4: Segmentation
#####
seg.cghdata <- segmentData(norm.cghdata, method="DNAcopy")

#####
### chunk number 5: Calling
#####
NeveCalled <- CGHcall(seg.cghdata)
}
```

This process takes about 20 minutes. To avoid waiting, we have saved the final object of class `cghCall` that can be loaded as

```
> data(NeveCalled)
```

We can then obtain the posterior probabilities. `CGHcall` function does not estimates the underlying number of copies for each segment but assigns the underlying status: loss, normal or gain. For each segment and for each individual we obtain three posterior probabilities corresponding to each of these three status. This is done by executing

```
> probs <- getProbs(NeveCalled)
```

This is a dataframe that looks like

```
> probs[1:5, 1:7]
```

	Clone	Chromo	BPstart	BPend	X600MPE	X600MPE.1	X600MPE.2
RP11-82D16	RP11-82D16	1	2008651	2008651	0.022	0.932	0.046
RP11-62M23	RP11-62M23	1	3367844	3367844	0.022	0.932	0.046
RP11-11105	RP11-11105	1	4261844	4261844	0.022	0.932	0.046
RMC01P070	RMC01P070	1	5918606	5918606	0.022	0.932	0.046
RP11-51B4	RP11-51B4	1	6068980	6068980	0.022	0.932	0.046

This table can be read as following. The probability that the individual X600MOE is normal for the signature RP11-82D16 is 0.932, while the probability of having a gain is 0.046 and 0.022 of having a loss.

We then determine the regions that are recurrent or common among samples. We use the `CGHregions` function that takes an object of class `cghCall` (e.g. object `NeveCalled` in our case). This algorithm reduces the initial table to a smaller matrix that contains regions rather than individual probes. The regions consist of consecutive clones with alike signatures (6). This can be done by executing

```
\dontrun{
library(CGHregions)
NeveRegions <- CGHregions(NeveCalled)
}
```

This process takes about 3 minutes. We have stored the result in the object `NeveRegions` that can be loaded as usual

```
> data(NeveRegions)
```

Now we have to get the posterior probabilities for each block/region. This can be done by typing

```
> probsRegions <- getProbsRegions(probs, NeveRegions, intensities)
```

Finally, the association analysis between each region and the strogen receptor positivity can be analyzed by using `multiCNVassoc` function. This function repeatedly calls `CNVassoc` and it returns the p-value of association for each block/region

```
> pvals <- multiCNVassoc(probsRegions, formula = "pheno~CNV", model = "mult",
+   num.copies = 0:2, cnv.tol = 0.01)
```

Notice that the arguments of `multiCNVassoc` function are the same as those of `CNVassoc`. In this example, we have set the argument `num.copies` equal to 0, 1, and 2 that corresponds to `loss`, `normal`, `gain` status used in the `CGHcall` function.

Multiple comparisons can be addressed by using Benjamini & Hochberg approach (1). The function `getPvalBH` produces the corrected p-values

```
> pvalsBH <- getPvalBH(pvals)
> head(pvalsBH)
```

	region	pval	pval.BH
1	319	2.891862e-06	0.001324473
2	318	1.633799e-05	0.002494267
3	320	1.576279e-05	0.002494267
4	316	8.998845e-05	0.010303677
5	9	2.865773e-04	0.011217002
6	298	2.027325e-04	0.011217002

Table 6 in (2) can be obtained by typing

```
> cumsum(table(cut(pvalsBH[, 2], c(-Inf, 1e-05, 1e-04, 0.001, 0.01,
+ 0.05))))
(-Inf,1e-05] (1e-05,0.0001] (0.0001,0.001] (0.001,0.01] (0.01,0.05]
              1              4              27              64              117
```

References

- [1] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J. Roy. Statist. Soc. Ser. B*, 57:289–300, 1995.
- [2] J. R. Gonzalez, I. Subirana G. Escaramis, S. Peraza, A. Caceres, X. Estivill, and Lluís Armengol. Accounting for uncertainty when assessing association between copy number and disease: a latent class model. *BMC Bioinformatics*, 2009, In press.
- [3] R. M. Neve, K. Chin, J. Fridlyand, J. Yeh, F. L. Baehner, T. Fevr, L. Clark, N. Bayani, J-P. Coppe, F. Tong, T. Speed, P. T. Spellman, S. DeVries, A. Lapuk, N. J. Wang, W-L. Kuo, J. L. Stilwell, D. Pinkel, D. G. Albertson, F. M. Waldman, F. McCormick, R. B. Dickson, M. D. Johnson, M. Lippman, S. Ethier, A. Gazdar, and J. W. Gray. A collection of breast cancer cell lines for the study of functionally distinct cancer subtypes. *Cancer Cell*, 10:515 – 527, 2006.
- [4] F. Picard, S. Robin, E. Lebarbier, and J.-J. Daudin. A segmentation/clustering model for the analysis of array CGH data. *Biometrics*, 63(3):758–766, 2007.
- [5] M. A. van de Wiel, K. I. Kim, S. J. Vosse, W. N. van Wieringen, S. M. Wilting, and B. Ylstra. CGHcall: calling aberrations for array CGH tumor profiles. *Bioinformatics*, 23(7):892–894, 2007.

- [6] M.A van de Wiel and W.N. van Wieringen. CGHregions: dimension reduction for array CGH data with minimal information loss. *Cancer Informatics*, 2:55–63, 2007.