Package "maskBAD"

Michael Dannemann, Michael Lachmann, Anna Lorenc

March 31, 2012

Abstract

Package "maskBAD" allows to users to identify, inspect and remove probes for which binding affinity differs between two groups of samples. It works for Affymetrix 3' IVT and whole transcript (exon and gene) arrays. When there are SNPs, transcript isoforms or cross hybrydizing transcripts specific to only one of compared groups, probes' binding affinities may differ between groups. Such probes bias estimates of gene expression, introduce false positives in differential gene expression analysis and reduce the power to detect real expression differences. Hence they should be identified and removed from the data at preprocessing stage. For details about impact of BAD (binding affinity difference) probes on differential gene expression estimates, implemented detection method and its power to improve data quality, please consult our paper Dannemann, Lorenc et. al: "The effects of probe binding affinity differences on gene expression measurements and how to deal with them", Bioinformatics, 2009 ([1]).

1 Introduction

The "maskBAD" package implements functions to detect and remove probes with different binding affinity (BAD) in Affymetrix array expression data. Identification and removal of BAD probes removes spurious gene expression differences and helps recover true ones. BAD probes are prevalent in comparisons of genetically distinct samples, like belonging to different strains or species, but systematic qualitative differences in transcriptome might introduce them also when samples differ by treatment, health status or tissue type. Masking therefore should be a routine step in data preprocessing. In this package we introduce functions that allow to identify, inspect and remove BAD probes and show how it can be integrated in a standard gene expression analysis pipeline.

2 Identification of BAD probes

Detection of BAD (binding affinity difference) probes is done on an `AffyBatch` object, usually prepared with `ReadAffy()` from package *affy*. We'll use 100 random probe sets from human and chimpanzee expression dataset ([2]), each with 10 individuals, available with the package.

```
> ## data available by loading data(AffyBatch)
> newAffyBatch

AffyBatch object
size of arrays=640x640 features (22 kb)
cdf=newCdf (100 affyids)
number of samples=20
number of genes=100
annotation=hgu95av2
notes=

> exmask <- mask(newAffyBatch,ind=rep(1:2,each=10),verbose=FALSE,useExpr=F)
```

The scoring of probes is performed with the function `mask` and is based on the algorithm presented in Dannemann et. al ([1]). Briefly, for a given probe, its signal is proportional to the amount of RNA in the sample and its binding affinity. When one target is measured with several probes (a probe set), as on Affymetrix arrays, the probes' signals are correlated for each sample. BAD probes correlation differs between groups, hence comparison of probes' pairwise correlations between those groups allows to identify of BAD probes. Function `mask()` detects probes differing in binding affinity between groups of samples defined by argument `ind`. The masking algorithm workes on a gene/probe set basis as the expression signal for the same transcript should be correlated among probes. Therefore, to identify BAD probes use probe sets defined at transcript/gene level.
The method works properly only for probe sets with signal above background in both groups. By default (`use.expr=TRUE` option) only probe sets with `mas5calls()` "P" in at least 90% of samples from both groups are considered in BAD detection. Another desired set of probe sets might be specified by the parameter `exprlist`.

2.1 Evaluation of the mask and choice of cutoff

For human-chimpanzee, with assumption of sequence divergence 1%, we'll expect naively ∼22% of probes comprise a SNP. To filter out lower 22% of quality scores, we have to put a cutoff at 0.029.

```
> quantile(exmask[[1]]$quality.score,0.22)

       22%
0.02922448
```

Manual inspection of borderline cases with `plotProbe()` might help to decide about the cutoff. It plots signal intensity of a probe against all other probes from the same probe set.

```
> ## add rownames containing x and y coordinates of each probe
> rownames(exmask[[1]]) <-apply(exmask[[1]][,c("x","y")],1,
+                               function(x)paste(x[1],x[2],sep="."))
> ## filtering for probes around the choosen cutoff
> probesToSee = rownames(exmask[[1]][exmask[[1]]$quality.score<0.03
+   & exmask[[1]]$quality.score >0.028,])
> ## select a random probe and plot it against
> ## all other probes of its probe set
> plotProbe(affy=newAffyBatch,probeset=as.character(exmask[[1]][ probesToSee[1],"probeset"]),
+           probeXY=probesToSee[1],scan=TRUE,ind=rep(1:2,each=10),
+           exmask=exmask$probes,names=FALSE)
```

2.2 Including external data

Mask might be calibrated with a subset of probes with known BAD status (called later on external mask), for example probes with target sequences known in both compared groups. External mask contains `x` and `y` coordinates and probe set assignement for the probes, along with their BAD status coded as 0/1 (BAD probe/not BAD probe). For a given cutoff, compare BAD status according to expression-based mask and given by external mask. Here we use the object sequenceMask - the information whether the probe sequence is affected by a SNP between the groups (0) or not (1) .

```
> head(exmask$probes)

          x   y quality.score probeset
399.559 399 559   0.139886115  1000_at
544.185 544 185   0.485398831  1000_at
530.505 530 505   0.704649407  1000_at
```

```
617.349 617 349   0.008627547  1000_at
459.489 459 489   0.072944350  1000_at
408.545 408 545   0.342924723  1000_at
```

```
> head(sequenceMask)
```

```
       x   y   affy id match
340  291 575 1195_s_at     0
788  617 349   1000_at     0
1134   6 591   1024_at     0
1624  15 427 1015_s_at     0
2305  26 515   1270_at     0
2563  31 541 1015_s_at     0
```

```
> rownames(sequenceMask) <- paste(sequenceMask$x,sequenceMask$y,sep="_")
> rownames(exmask$probes)<- paste(exmask$probes$x,exmask$probes$y,sep="_")
> cutoff=0.029
> table((exmask$probes[rownames(sequenceMask),"quality.score"]>cutoff)+0,
+       sequenceMask$match)
```

```
     0   1
  0  93 151
  1  79 859
```

Here, 952 probes are concordant between masks, whereas 79 probes with known SNP are not detected by expression-based mask.

With `overlapExprExtMasks()` expression mask may be compared with any other designation of BAD probes along a range of cutoffs. Type 1 (fraction of external mask not BAD probes, identified as BAD by expression mask) and type 2 (fraction of external mask BAD probes, not detected as BAD in expression mask) errors are plotted with their binomial confidence intervals.

```
> head(exmask$probes[,1:3])
```

```
          x   y quality.score
399_559 399 559   0.139886115
544_185 544 185   0.485398831
530_505 530 505   0.704649407
617_349 617 349   0.008627547
459_489 459 489   0.072944350
408_545 408 545   0.342924723
```

```
> head(sequenceMask[,c(1,2,4)])
```

```
          x   y match
291_575 291 575     0
617_349 617 349     0
6_591     6 591     0
15_427   15 427     0
26_515   26 515     0
31_541   31 541     0
```

The function `overlapExprExtMasks()` applied on our example data, produces a plot like shown in Figure 1.

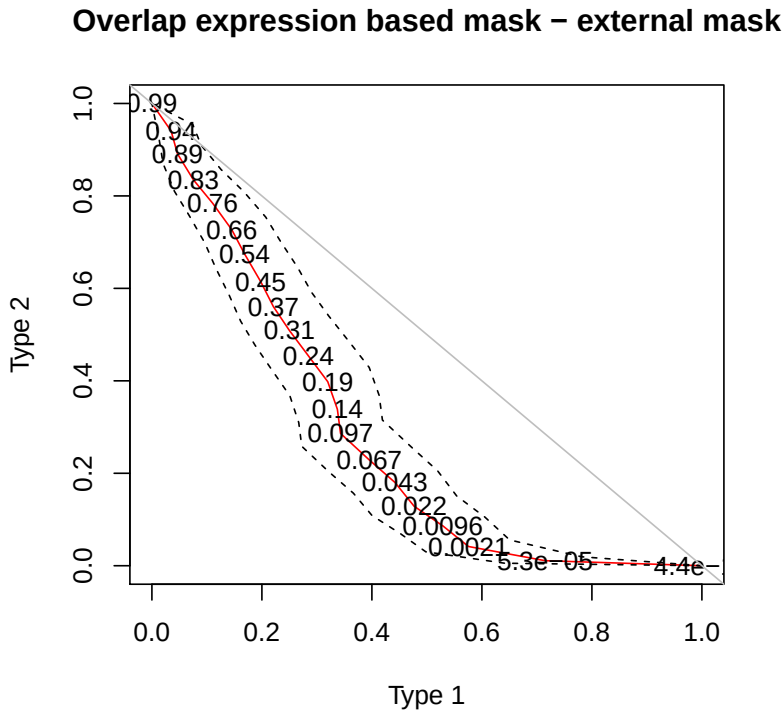

Figure 1: Overlap of expression-based mask and external mask (red line), with binomial confidence intervals (dashed lines). Several cutoff levels are marked on the plot.

```
> overlapExSeq <- overlapExprExtMasks(exmask$probes[,1:3],
+                                     sequenceMask[,c(1,2,4)],verbose=FALSE,
+                                     cutoffs=quantile(exmask$probes[,3],seq(0,1,0.05)))
```

The cutoff depends on the goal of masking. SFPs detection needs tighter control of type 2 than type 1 error - low cutoff values. Removal of differential expression bias needs higher cutoff. The distribution of quality scores for probes declared in external mask as BAD and not BAD may be compared with Wilcoxon rank sum test and Kolomogornov-Smirnow test for several cutoff levels. When those distributions do not differ (high p-values on 2), cutoff is too high.

```
> overlapTests <- overlapExprExtMasks(exmask$probes[,1:3],verbose=FALSE,
+                                     sequenceMask[,c(1,2,4)],
+                                     wilcox.ks=T,sample=100)
```

When no information about sequence divergence between the groups is available, a cutoff might be suggested by the distribution of quality scores for the mask. On Figure 3, an excess of probes with low quality scores is obvious. Assuming a uniform distribution of quality scores for probes not different between groups, a cutoff should be set so that the remaining distribution is uniform - like.

3 Removal of BAD probes and estimation of expression values

3.1 Removal of BAD probes

To remove probes from `affybatch`, use `prepareMaskedAffybatch()`. It produces an `affybatch` object (without the masked probes) and an according `CDF` environment. When one probe appears in several probe sets (as is a case for some custom annotations) and it is detected as BAD in one probe set only, it will be nevertheless removed from all probe sets it appears. With `cdftable` argument, `prepareMaskedAffybatch()` allows also to remove any list of probes.

```
> affyBatchAfterMasking <-
+   prepareMaskedAffybatch(affy=newAffyBatch,exmask=exmask$probes,
+                          cutoff=quantile(exmask[[1]]$quality.score,0.22))

Inferring mismatch positions from perfect match
Renaming columns:  probeset x y as probeset px py
Regular run: more than  NA  probe per grouping
Made list.
Making env
finished making CDF!

> newAffyBatch

AffyBatch object
size of arrays=640x640 features (22 kb)
cdf=newCdf (100 affyids)
number of samples=20
number of genes=100
annotation=hgu95av2
notes=

> affyBatchAfterMasking
```

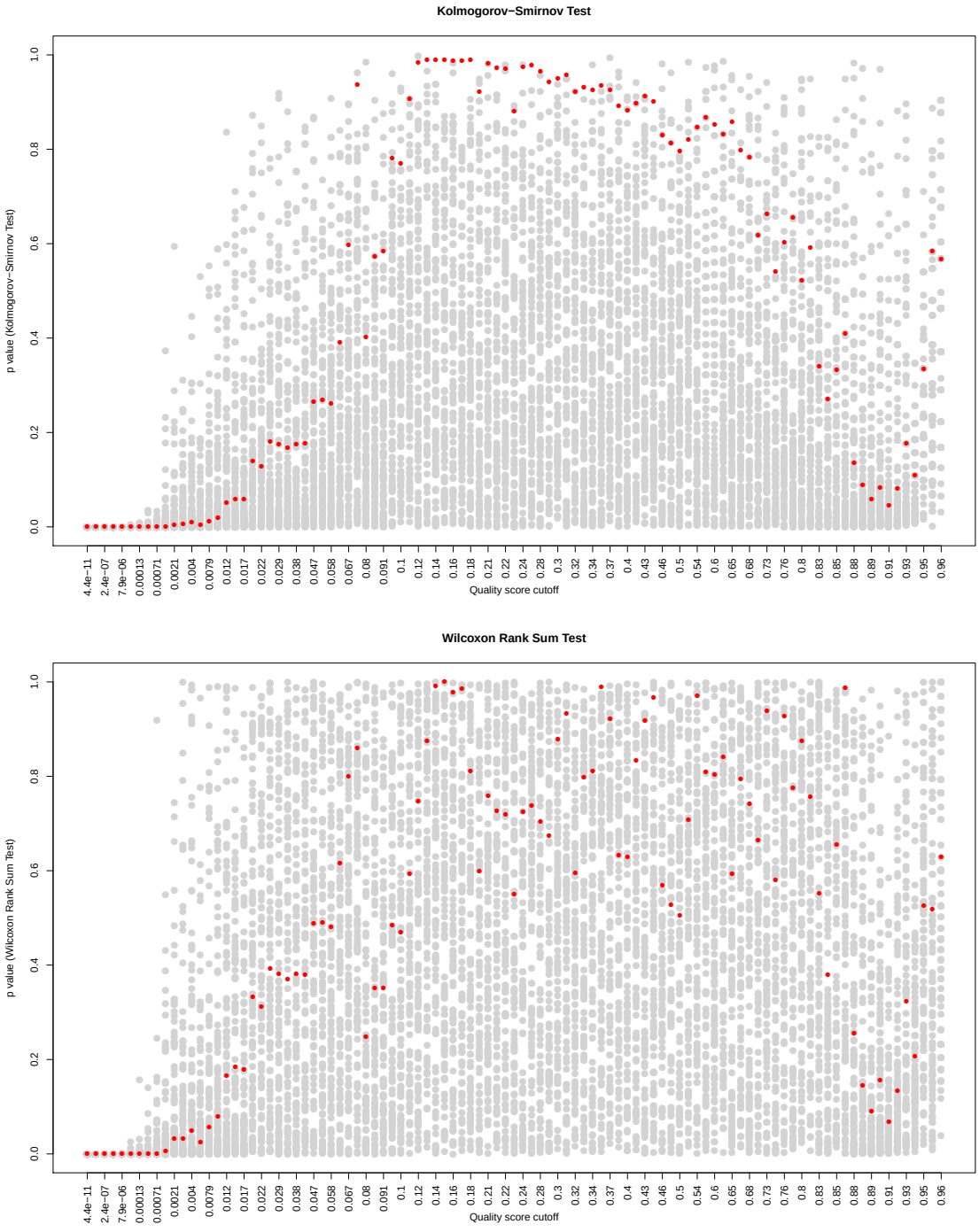

Figure 2: Kolmogorow-Smirnov-Test (upper panel) and Wilcoxon-Rank-Sum-Test (lower panel) comparing distributions of quality scores of BAD and not BAD probes. For low cutoffs, two distributions are different (red - actual tests; gray - tests on bootstraped data, to estimate variance).

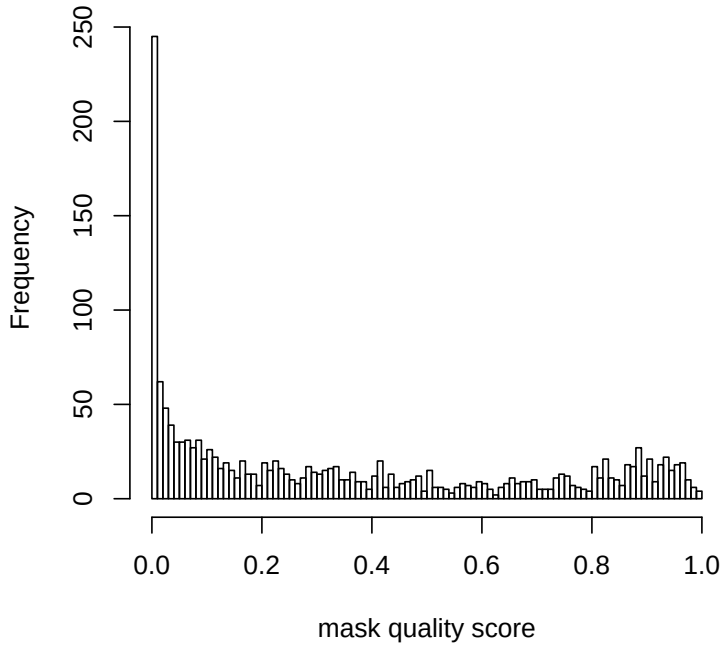

Figure 3: Probes' quality scores.

```
$newAffyBatch
AffyBatch object
size of arrays=640x640 features (22 kb)
cdf=new_cdf (100 affyids)
number of samples=20
number of genes=100
annotation=hgu95av2
notes=

$newCdf
<environment: 0x85c6200>
```

Note changed number of genes (probe sets) in masked `affybatch`.

3.2 Expression estimation with *rma*

For the new `affybatch` expression values might be estimated as usual. New `affybatch` object has a masked `cdf` declared, instead of a standard one.

```
> cdfName(affyBatchAfterMasking[[1]])

[1] "new_cdf"
```

Therefore it is important to save new `cdf` for further use and make it available in the workspace (with a name matching `cdfName` slot of masked `affybatch`) whenever a function to estimate expression is called.

```
> new_cdf=affyBatchAfterMasking[[2]]
> head(exprs(rma(affyBatchAfterMasking[[1]])))

Background correcting
Normalizing
Calculating Expression
          a_c1a.CEL a_c1b.CEL a_c1c.CEL a_c1d.CEL a_c1e.CEL a_c1f.CEL a_c2a.CEL
1000_at    8.519381  8.239144  8.698188  8.426627  8.213712  8.631725  8.702705
1001_at    4.926438  4.926092  4.976691  5.100733  5.071768  4.901918  4.899042
1002_f_at  3.994510  3.957382  4.015966  4.146419  4.027846  4.028062  3.938472
1003_s_at  6.156051  6.242018  6.284055  6.408501  6.484892  6.239529  5.993723
1004_at    5.491737  5.487617  5.606666  5.861494  5.653364  5.564533  5.545361
1005_at    6.606239  6.125861  6.718311  6.343611  6.174270  6.464771  5.276121
          a_c2b.CEL a_c2c.CEL a_c2d.CEL a_h1a.CEL a_h1b.CEL a_h1c.CEL a_h1d.CEL
1000_at    8.147788  8.531981  8.406286  8.617236  8.265444  8.821593  8.607421
1001_at    5.082391  5.025381  5.166789  4.935324  4.966834  4.984837  5.063387
1002_f_at  4.085803  3.981629  3.984972  4.155437  4.396664  4.026082  4.319549
1003_s_at  6.178582  6.249877  6.329526  6.465071  6.704101  6.283548  6.401511
1004_at    5.559185  5.647424  5.666993  5.935005  5.918048  5.894887  5.806115
1005_at    5.237686  5.070708  5.382761  4.831896  5.298492  5.073494  4.665798
          a_h1e.CEL a_h1f.CEL a_h2a.CEL a_h2b.CEL a_h2c.CEL a_h2d.CEL
1000_at    8.344701  8.542945  8.607421  8.156907  8.882432  8.567411
1001_at    5.308487  4.971850  4.911928  4.815596  4.822864  5.187153
1002_f_at  4.511715  4.240136  3.977734  4.016085  3.964241  4.185346
1003_s_at  6.747904  6.560276  6.206139  6.275659  6.210961  6.311143
1004_at    6.409215  6.004868  5.643354  5.551346  5.601732  5.835371
1005_at    5.680619  5.317797  5.036664  5.414302  4.848811  4.793108
```

3.3 Expression estimation with *gcrma*

Normalization and expression estimates with `gcrma()` require probe affinities. Those should be computed with standard `cdf`. Then both `affybatch` object and probe affinities object should be filtered with the mask.

```
> library(gcrma)
> library(hgu95av2probe)
> affy.affinities=compute.affinities("hgu95av2",verbose=FALSE)

> affy.affinities.m=prepareMaskedAffybatch(affy=affy.affinities,
+   exmask=exmask$probes,cdfName="MaskedAffinitesCdf")

Inferring mismatch positions from perfect match
Renaming columns:  probeset x y as probeset px py
Regular run: more than  NA  probe per grouping
Made list.
Making env
finished making CDF!
```

To calculate expression values, just use both masked objects (make sure that a `cdf` with name matching `cdf` slot of masked `affyBatch` is accessible!).

```
> head(exprs(gcrma(affyBatchAfterMasking[[1]],affy.affinities.m)))

Adjusting for optical effect...................Done.
Normalizing
```

```
Calculating Expression
          a_c1a.CEL a_c1b.CEL a_c1c.CEL a_c1d.CEL a_c1e.CEL a_c1f.CEL a_c2a.CEL
1000_at    8.634916  8.358381  8.796348  8.549656  8.423818  8.733339  8.771972
1001_at    5.846012  5.845099  5.895825  5.982687  5.989225  5.821090  5.813759
1002_f_at  5.249379  5.206399  5.273889  5.380924  5.271458  5.265842  5.193669
1003_s_at  6.663292  6.720045  6.750878  6.852030  6.921729  6.700775  6.548312
1004_at    6.181949  6.167532  6.273633  6.458531  6.284642  6.262856  6.222565
1005_at    6.984931  6.588212  7.071891  6.788985  6.807228  6.902829  6.077073
          a_c2b.CEL a_c2c.CEL a_c2d.CEL a_h1a.CEL a_h1b.CEL a_h1c.CEL a_h1d.CEL
1000_at    8.394478  8.649110  8.534103  8.713059  8.384302  8.905719  8.721932
1001_at    5.982168  5.928388  6.013410  5.856875  5.855905  5.881186  5.956344
1002_f_at  5.323003  5.253987  5.225674  5.386190  5.636114  5.265202  5.553035
1003_s_at  6.667764  6.724276  6.773942  6.892581  7.056563  6.739612  6.906102
1004_at    6.213381  6.283493  6.307053  6.507925  6.546897  6.487002  6.475916
1005_at    6.046712  5.928464  6.171188  5.813974  6.096748  5.925406  5.679128
          a_h1e.CEL a_h1f.CEL a_h2a.CEL a_h2b.CEL a_h2c.CEL a_h2d.CEL
1000_at    8.467408  8.657795  8.721932  8.343586  8.929597  8.677108
1001_at    6.232725  5.881665  5.825770  5.759558  5.768511  6.070740
1002_f_at  5.603963  5.516864  5.233100  5.253987  5.213435  5.418177
1003_s_at  7.192853  6.996384  6.675245  6.745539  6.720009  6.769744
1004_at    6.895957  6.569566  6.299328  6.119514  6.234432  6.440966
1005_at    6.509646  6.098351  5.886391  6.187611  5.805249  5.720689
```

3.4 Exon and gene arrays

Exon and gene arrays require setting `useExpr=FALSE`, as those arrays do not contain MM probes and `mas5calls()` - based definition of expressed genes is not applicable. As computing time increases with number of probe sets, we recommend to use a subset of probe sets (specified with `exprlist`) for exon arrays.

References

[1] Michael Dannemann, Anna Lorenc, Ines Hellmann, Philipp Khaitovich, and Michael Lachmann. The effects of probe binding affinity differences on gene expression measurements and how to deal with them. *Bioinformatics (Oxford, England)*, 25(21):2772–2779, November 2009.

[2] Philipp Khaitovich, Bjoern Muetzel, Xinwei She, Michael Lachmann, Ines Hellmann, Janko Dietzsch, Stephan Steigele, Hong-Hai H. Do, Gunter Weiss, Wolfgang Enard, Florian Heissig, Thomas Arendt, Kay Nieselt-Struwe, Evan E. Eichler, and Svante Pääbo. Regional patterns of gene expression in human and chimpanzee brains. *Genome Research*, 14(8):1462–1473, August 2004.