

# PECA: Probe-level Expression Change Averaging

Tomi Suomi

April 16, 2015

`tomi.suomi@utu.fi`

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Example</b>	<b>2</b>

## 1 Introduction

PECA determines differential gene/protein expression using directly the probe/peptide-level measurements from Affymetrix gene expression microarrays or proteomic datasets, instead of the common practice of using precalculated gene/protein-level values. An expression change between two groups of samples is first calculated for each measured probe/peptide. The gene/protein-level expression changes are then defined as medians over the probe/peptide-level changes. This is illustrated in fig 1. For more details about the probe-level expression change averaging (PECA) procedure, see Elo et al. (2005), Laajala et al. (2009) and Suomi et al.

PECA calculates the probe/peptide-level expression changes using the ordinary or modified t-statistic. The ordinary t-statistic is calculated using the function `rowttests` in the Bioconductor `genefilter` package. The modified t-statistic is calculated using the linear modeling approach in the Bioconductor `limma` package. Both paired and unpaired tests are supported.

The significance of an expression change is determined based on the analytical p-value of the gene-level test statistic. Unadjusted p-values are reported along with the corresponding p-values looked up from beta distribution taking into account the number of probes/peptide per gene/protein. The quality control and filtering of the data (e.g. based on low intensity or probe specificity) is left to the user.

## 2 Example

We start by loading PECA and spike-in example data.

```
> library(PECA)
> library(SpikeIn)
> data(SpikeIn133)
```

We subset the original spike-in for our purposes, including two groups with three replicates.

```
> data <- SpikeIn133[,c(1,15,29,2,16,30)]
```

Calculate p-values using PECA.

```
> peca_results <- PECA_AffyBatch(normalize="true", affy=data)
```

Calculate p-values using MAS5.

```
> library(multtest)
> mas5 <- mas5(data)
```

```
background correction: mas
PM/MM correction : mas
expression values: mas
```

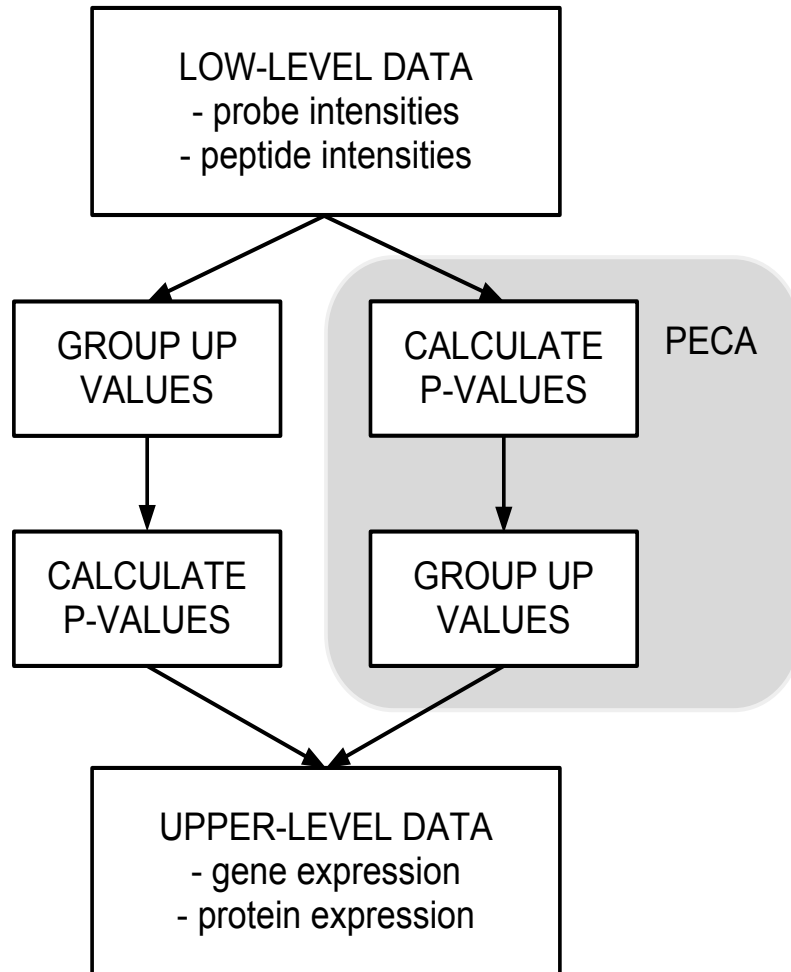


Figure 1: Probe-level expression change averaging

background correcting...done.

22300 ids to be processed

```
|           |  
|#####|
```

```
> groups=c(1,1,1,0,0,0)  
> stats=mt.teststat(exprs(mas5), groups, test="t")  
> mas5_results=2*(1-pnorm(abs(stats)))
```

Calculate p-values using RMA.

```
> rma <- rma(data)
```

Background correcting  
Normalizing  
Calculating Expression

```
> groups=c(1,1,1,0,0,0)  
> stats=mt.teststat(exprs(rma), groups, test="t")  
> rma_results=2*(1-pnorm(abs(stats)))
```

Combine and preview results.

```
> results <- cbind(peca=peca_results[, "p"], mas5=mas5_results, rma=rma_results, spike=0)  
> rownames(results) <- rownames(peca_results)  
> results[,4][rownames(results) %in% colnames(pData(data))] <- 1  
> head(results)
```

	peca	mas5	rma	spike
1007_s_at	0.4185863	0.6620376	0.1720956	0
1053_at	0.9974262	0.1478299	0.8738921	0
117_at	0.9998561	0.6182159	0.5277216	0
121_at	1.0000000	0.5922030	0.4554895	0
1255_g_at	0.9999160	0.9854084	0.4150493	0
1294_at	0.9181849	0.9342097	0.7798362	0

ROC Curves from these results show that PECA performs well at low false positive rates, which in practice is of relevance to the user.

```
> library(ROCR)  
> results_roc <- results  
> results_roc[,1:3] <- 1-results_roc[,1:3]  
> pred.peca <- prediction(results_roc[,1], results_roc[,4])  
> pred.mas5 <- prediction(results_roc[,2], results_roc[,4])  
> pred.rma <- prediction(results_roc[,3], results_roc[,4])  
> perf.peca <- performance(pred.peca, "tpr", "fpr")  
> perf.mas5 <- performance(pred.mas5, "tpr", "fpr")  
> perf.rma <- performance(pred.rma, "tpr", "fpr")
```

```
> plot(perf.mas5, lwd=3, col="green")
> plot(perf.rma, lwd=3, col="blue", add=TRUE)
> plot(perf.pecas, lwd=3, col="red", add=TRUE)
> legend(0.7,0.2,c('PECA', 'RMA', 'MAS'), col=c('red','blue','green'), lwd=3)
```

