Tools for detection of batch effect between experiments with an application to the comparison of batch effect correction methods

Guillaume Heger

24 May 2019

Introduction

I introduce here three methods for batch effect detection in merged experiments datasets. The first two ones are based on consideration on the samples while the third one is based on consideration on the genes.

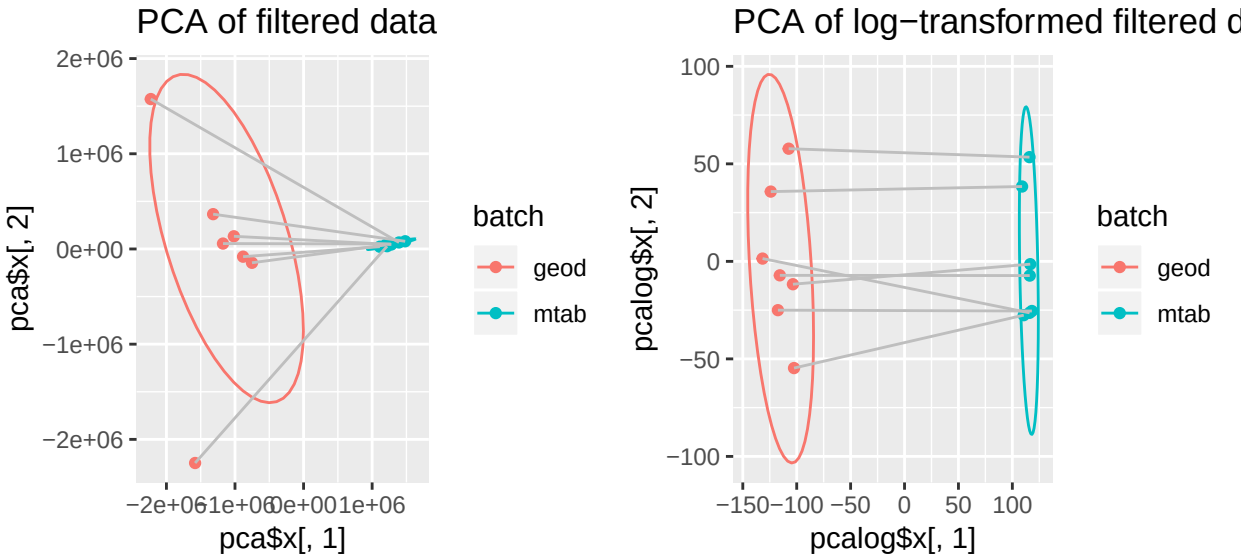
I worked here with two RNASeq Baseline datasets from the Mouse transcriptome, extracted from Expression Atlas. Their reference are GEOD74747 and MTAB4644.

In order to have two distributions which should theoretically be equal, I have only kept the common organism parts between the two experiments. A very simple filtering has been done then, omly extracting genes with 0 expression in one of the two experiments. Finally regarding the scale difference between the two datasets, I have log-transformed them to have a common scale between them. This may be subject of discussion.

```
geod74747<-get(load(
  'Mus musculus/~organism part/E-GEOD-74747-atlasExperimentSummary.Rdata'
))$rnaseq
mtab4644<-get(load(
  'Mus musculus/~organism part/E-MTAB-4644-atlasExperimentSummary.Rdata'
))$rnaseq

whole<-cbind(
  geod74747 %>% assays %>% use_series(counts),
  mtab4644 %>% assays %>% use_series(counts)
)
batch<-c(
  'geod' %>% rep(dim(geod74747)[2]),
  'mtab' %>% rep(dim(mtab4644)[2])
)
tissue<-c(
  geod74747$organism_part,
  mtab4644$organism_part
)
#keeping only common tissues
common.tissues<-intersect(
  tissue[batch=='geod'],
  tissue[batch=='mtab']
)
whole %<>% extract(,tissue %in% common.tissues)
batch %<>% extract(tissue %in% common.tissues)
tissue %<>% extract(tissue %in% common.tissues)

filter<-
  rowSums(whole[,batch=='geod']>0)>0 &
  rowSums(whole[,batch=='mtab']>0)>0
```

```
whole[filter,]->filtered

filtered %>% t %>% prcomp -> pca
filtered %>% log1p %>% t %>% prcomp -> pcalog
grid.arrange(
  ggplot(mapping=aes(x=pca$x[,1],y=pca$x[,2],colour=batch))+
    geom_point()+stat_ellipse()+
    geom_line(aes(group=tissue),colour='grey')+
    ggtitle("PCA of filtered data"),
  ggplot(mapping=aes(x=pcalog$x[,1],y=pcalog$x[,2],colour=batch))+
    geom_point()+stat_ellipse()+
    geom_line(aes(group=tissue),colour='grey')+
    ggtitle("PCA of log-transformed filtered data"),
  ncol=2
)
```

```
## Warning in MASS::cov.trob(data[, vars]): Probable convergence failure
```




```
#LOG TRANSFORMATION
filtered%<>%log1p
```

Sample-based detection methods

Guided PCA (gPCA)

The idea of gPCA comes from an article by Sarah Reese [A new statistic for identifying batch effects in high-throughput genomic data that uses guided principal component analysis.] I kept this fundamental idea and added some other features to it, among which is the notion of variance rank. In a nutshell, gPCA performs PCA on batchwise aggregated data (merged from all experiments) (aggregation can be sum or mean). The number of samples in the new dataset is the number of batches. Then we can project the non-aggregated data on the new "guided" principal components (gPCs). To see the importance of the "batch variable", one shall compare the parts of variance of the gPCs with the ones of the original principal components (PCs or eigengenes, i.e. from PCA on the merged dataset without batchwise aggregation). We can compute a δ statistic and its associated p-value (see [reference] for more information). This δ statistic represents somehow the part of variance of batch effect. We would like it to be small, or anyway with non-significant p-value,

which would mean that batch effect has as much effect as normal random noise.

The following call to the function `gPCA` shows several statistics:

```
filtered %>% gPCA(batch,nperm=1000,progress_bar = FALSE) -> gfiltered
```

```
## Computing PCA
## Computing gPCA
## part of variance from gPC1 : 0.756350096719459
## delta statistic : 0.999874798191682
## cumulative delta statistics :
## delta_1=0.999874798191682
##  delta_2=0.925202672295636
## variance ranks : 1
##  variance ranks : 13
## Estimating p-value
##
## p-value : 0
```

The first figure shown is the absolute part of variance of the first gPC, analogous to the parts of variance typically computed for classical PCA. Then we can see the δ statistic which is the ratio of the previous part of variance from gPC_1 by the part of variance from PC_1 :

$$\delta = \frac{\mathbb{V}gPC_1}{\mathbb{V}PC_1}$$

As PC1 is the one-dimensional axis which has most variance in the space of eigengenes, this ratio shall be less than 1. If δ is close to 1, it means that batch effect is responsible of a big part of variance in the dataset.

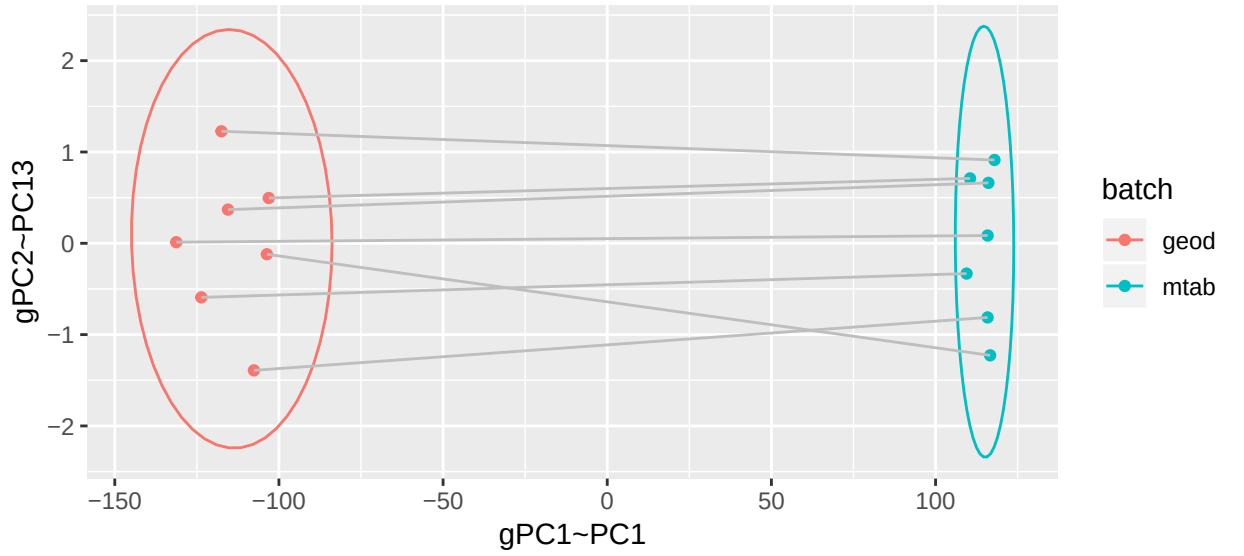
The following are cumulative δ statistics. Their number is the number of batches considered. δ_1 is exactly δ while δ_2 is the ratio between the cumulative part of variance of gPC_1 and gPC_2 and the cumulative part of variance of PC_1 and PC_2. In a general way :

$$\delta_k = \frac{\mathbb{V}gPC_1 + ... + \mathbb{V}gPC_k}{\mathbb{V}PC_1 + ... + \mathbb{V}PC_k}$$

Finally, the variance ranks are displayed. They position the parts of variance of the gPCs among the parts of variance of the PCs. The variance rank of gPC1 is the minimal number n such that PC_{n+1} has a smaller part of variance. Here the variance rank of gPC_1 is 1, which means that the batch effect creates variance in a comparable order of magnitude to PC_1.

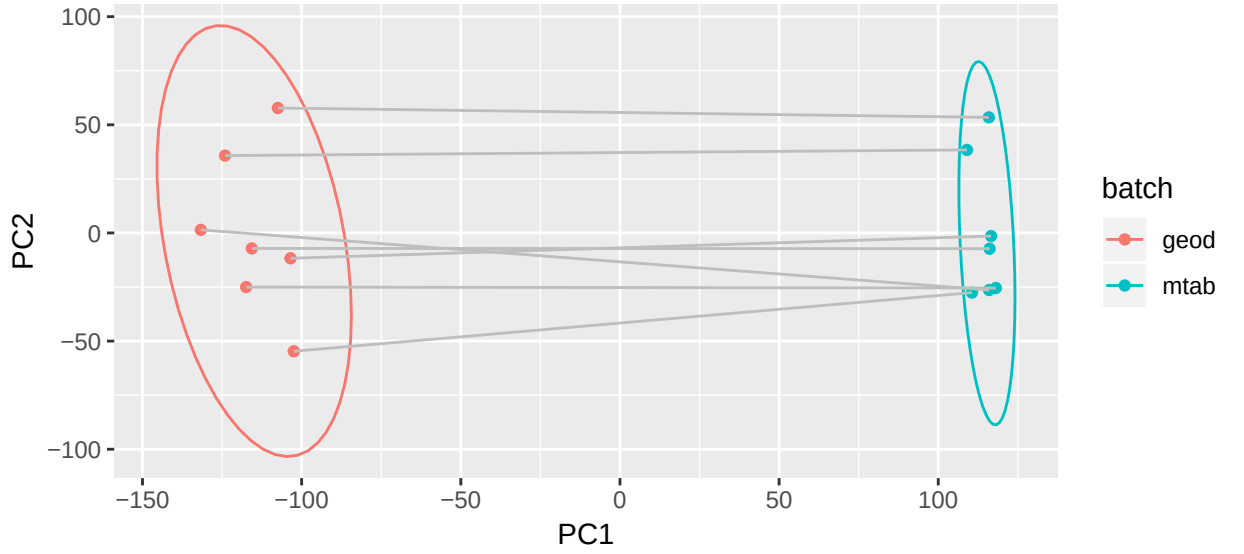
We can show all these results graphically. The plot below shows the first plan of gPCA (gPC_1 and $gPC2$). The grey lines link the samples extracted from the same tissue in the different experiments.

```
gfiltered %>% viz_gpca + geom_line(aes(group=tissue),colour='grey')
```



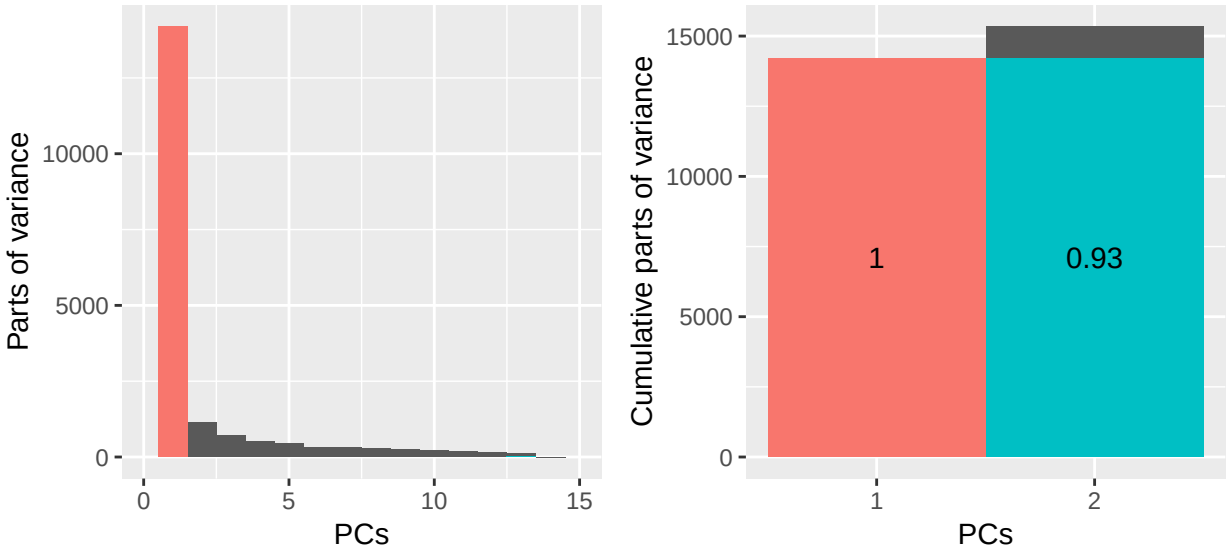
The function `gPCA` computes both gPCA and PCA, in order to compare them. Below is the first plan of PCA (PC_1 and PC_2).

```
gfiltered %>% viz_gpca(guided=FALSE) + geom_line(aes(group=tissue),colour='grey')
```



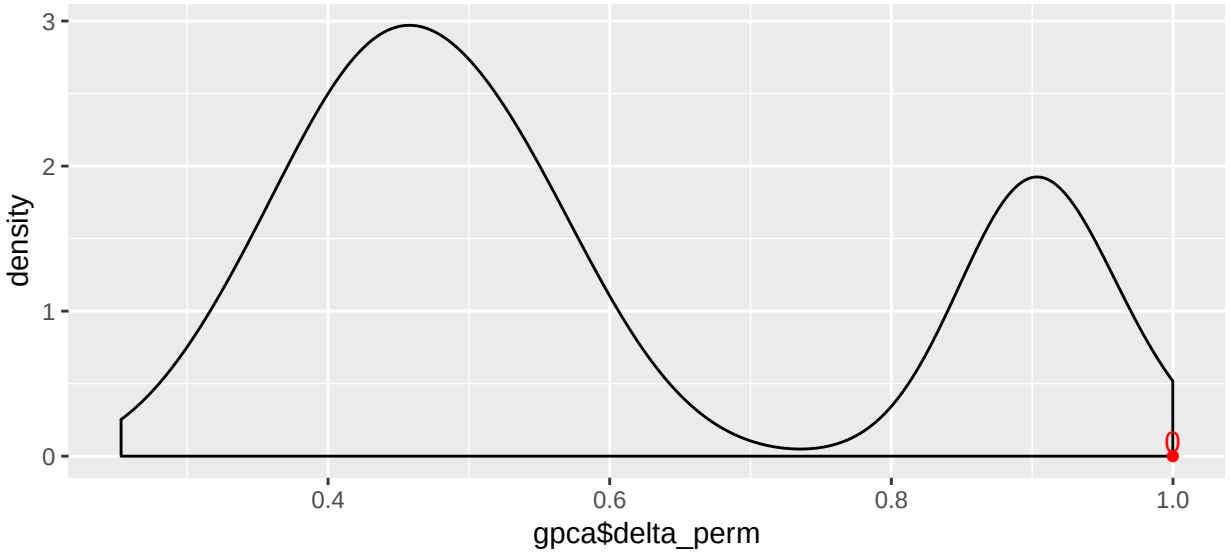
We can also visualise the cumulative parts of variance and cumulative δ statistics with their variance ranks shown on the variance profile of the dataset :

```
gfiltered %>% viz_gpca_contrib
```



Finally, we can show the p-value on a density plot. This density plot is estimated by permuting the batch labels in order to compare the effect of the real batches with whose of fake batches, assimilated to random noise.

```
gfiltered %>% viz_gpca_pvalue
```



Intra-class inertia index

Contrarily to the previous method, the intra-class inertia index takes into account the available biological information that differs between the samples of a dataset. In our example of the Mouse datasets, the experimental variable is the organism part.

A gene-based detection method

Previous methods for estimating the importance of batch effect between the experiments are based on consideration on the samples. Here I try to introduce a new kind of estimation based on consideration on

the genes and their relation with the eigengenes computed by PCA. This idea is driven by the fact that a biologist wants to know in general which genes are "important" in a dataset, that is to say which genes create most variance between the samples of an experiment. The basical idea of PCA is to find, not such genes, but eigengenes (which can be interpreted of summary of real genes with weights on each of them). The eigengenes are designed to be empirically decorrelated across the samples and to be ordered according to part of variance. Notably the first eigengene (PC1) is the summary of real genes (i.e. linear combination of real genes coordinates) that create most variance among all those possible summaries. As eigengenes are expressed with weights on genes, these weights allow to highlight some genes (for example, we can consider that the most discriminant genes in a dataset are those which have the highest weights (in absolute value) for the first eigengene).

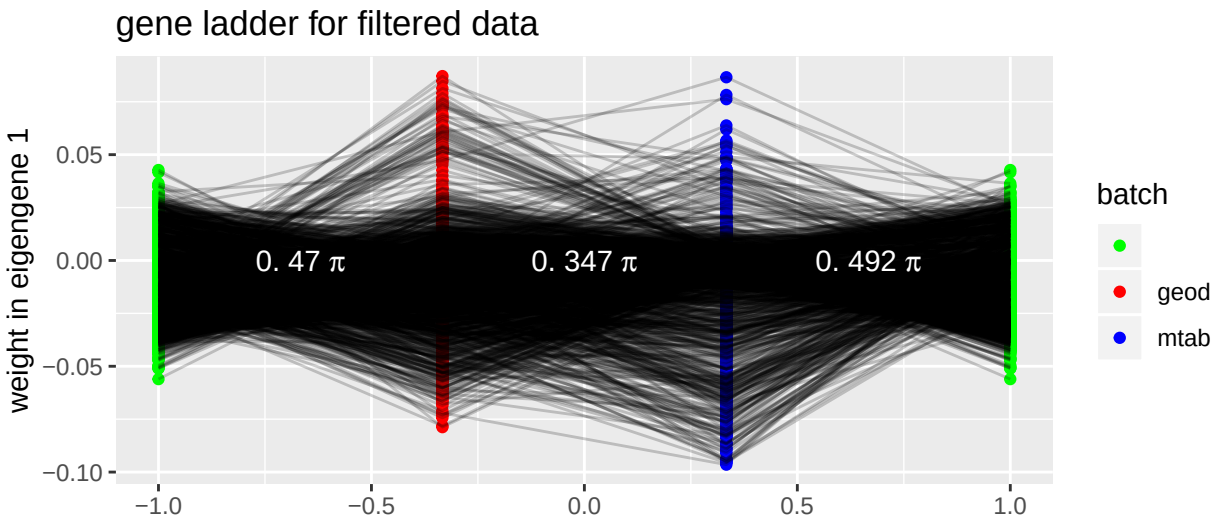
As PCA is a common way to visualise the results of an experiment, we can consider a new approach of the problem of batch effect. Actually to decide if there is batch effect between two experiments, one can wonder two questions :

- Do the two experiments highlight the same genes ?
- Does the merger of the two experiments highlight the same genes than each experiment taken separately ?

This can be rephrased in terms of weights of genes in the eigengenes of the three datasets : the first experiment, the second one, and their merger. In other words :

- Do the two experiments have the same eigengenes ?
- Does their merger have the same eigengenes than each experiment considered separately ?

As eigengenes represents geometrical directions in the space of genes, a good way to compare them is to calculate angles between them. Let's take our example of the two Mouse datasets : data has been filtered and log-transformed. We will calculate the angle between the first eigengene of the first dataset (denoted PC_1^1) and the first eigengene of the second dataset (denoted PC_1^2), as well as the angles between the PC1 of the merger (denoted PC_1^m) with PC_1^1 and PC_1^2. Thus we have before any correction :



We can see that the angles $\widehat{PC_1^1, PC_1^m}$ and $\widehat{PC_1^2, PC_1^m}$ are very close to the right angle $\frac{\pi}{2}$, which means that the batch effect completely changes the main directions of variance when we consider the merged dataset. Regarding the angle $\widehat{PC_1^1, PC_1^2}$ between the PC1 of the two original datasets, it is smaller (approximately $\frac{\pi}{3}$) but quite important though. It means that the two experiments contain different information.

Then we can apply BMC to this merged dataset. The angle $\widehat{PC_1^1, PC_1^2}$ doesn't change, as it can be expected,

because the only transformation is a shift, which doesn't change the variance of the dataset. However, the angles of the individual datasets with their merger have become much smaller (less than $\frac{\pi}{5}$). Indeed an important part of the variance due to batch effect has been removed with mean centering. Previously a big part of variance was simply explained by the gap between the two datasets (visible on figure... (PCA))

If we apply ComBat, it really seems that this angle measure is sensitive to correction, as all the 3 angles have decreased.

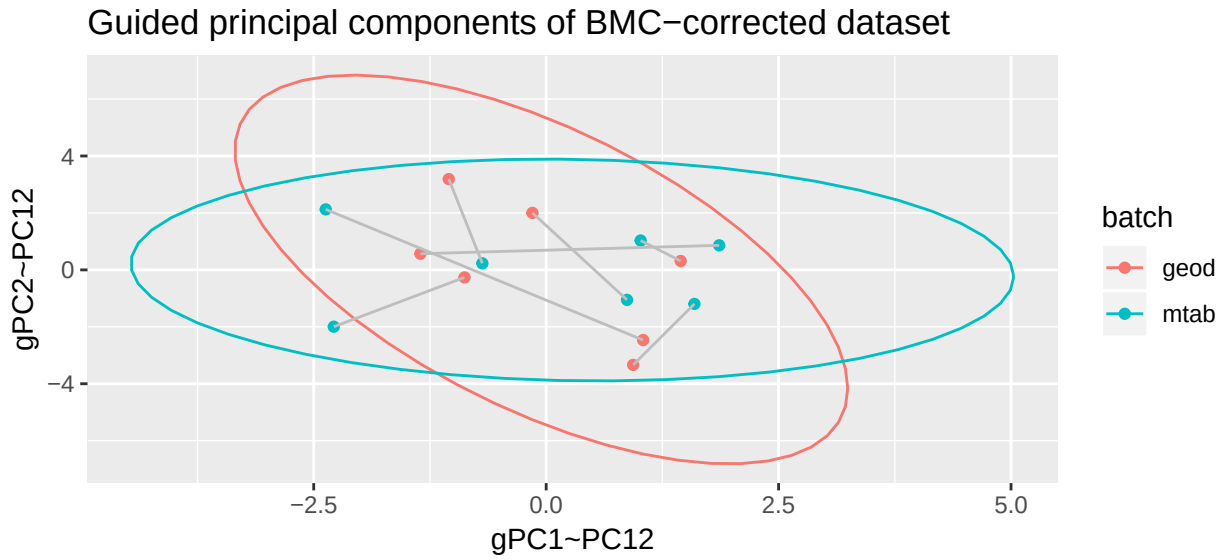
Comparison of different batch effect correction methods using these tools

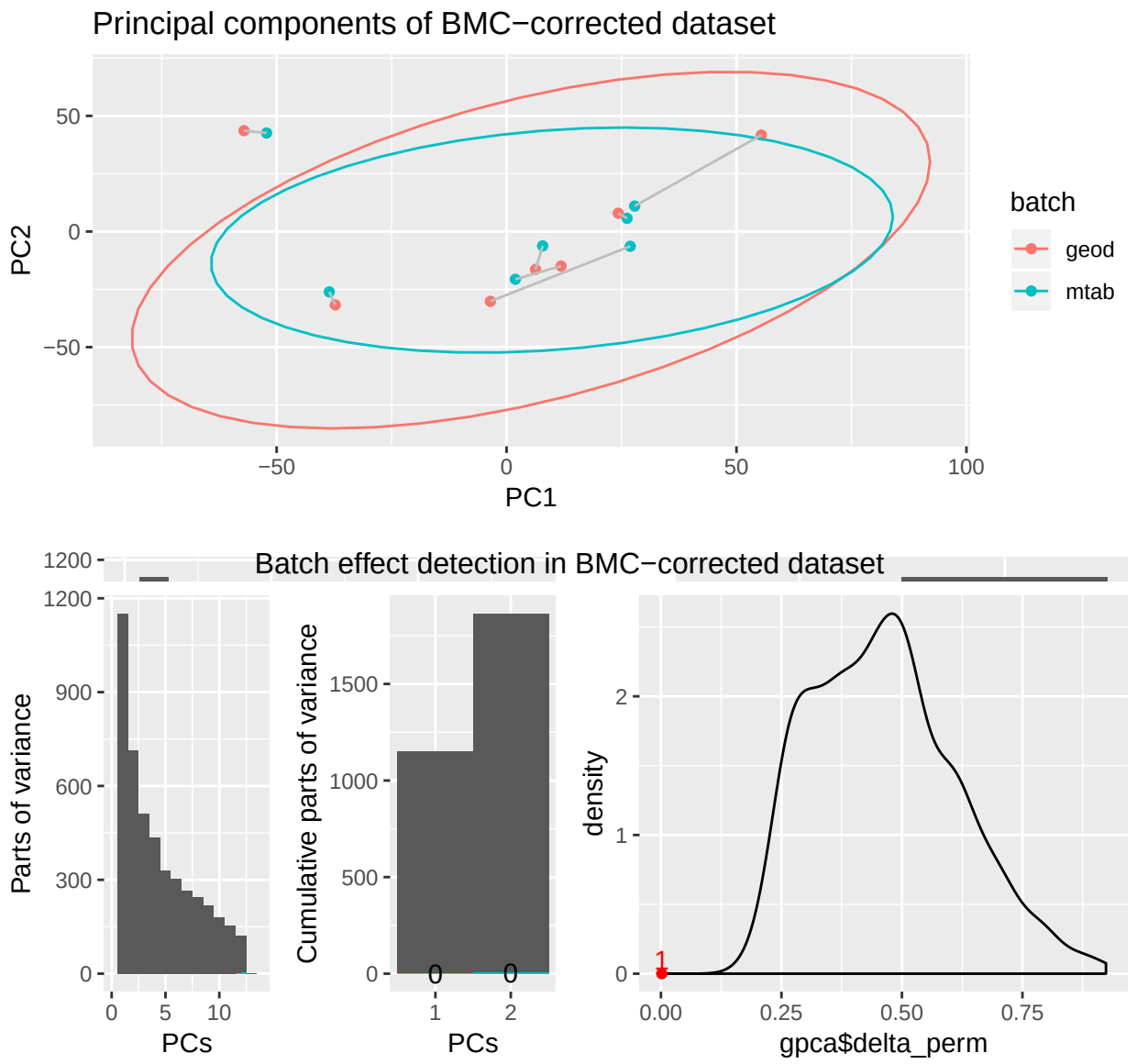
Using gPCA

Batch Mean Centering (BMC)

```
pam %>% gPCA(batch,nperm=1000) -> gpam
```

```
## Computing PCA
## Computing gPCA
## part of variance from gPC1 : 0.000446344314187227
## delta statistic : 0.00179648616443703
## cumulative delta statistics :
## delta_1=0.00179648616443703
##  delta_2=0.00295721300272122
## variance ranks : 12
##  variance ranks : 12
## Estimating p-value
##
## p-value : 1
```





BMC method has astonishing results if we analyse them using gPCA. It is actually normal as gPCA performs PCA on batchwise mean-aggregated data. Yet the two batches of the BMC corrected data have the same mean, by definition of BMC. Therefore gPCA is performed on one (aggregated) sample, which is theoretically not possible. The functions works though due to computing errors, but the result is worthless. gPCA is not applicable to BMC method.

However it is very easy to understand that BMC is a very simple method that tries to correct batch effect in the case of a systematic additive bias. Although it can remove a big part of batch effect, it is not a generic method as it needs the two distributions to be the same. However here it seems to be a good method as the samples from same tissue are close to each other.

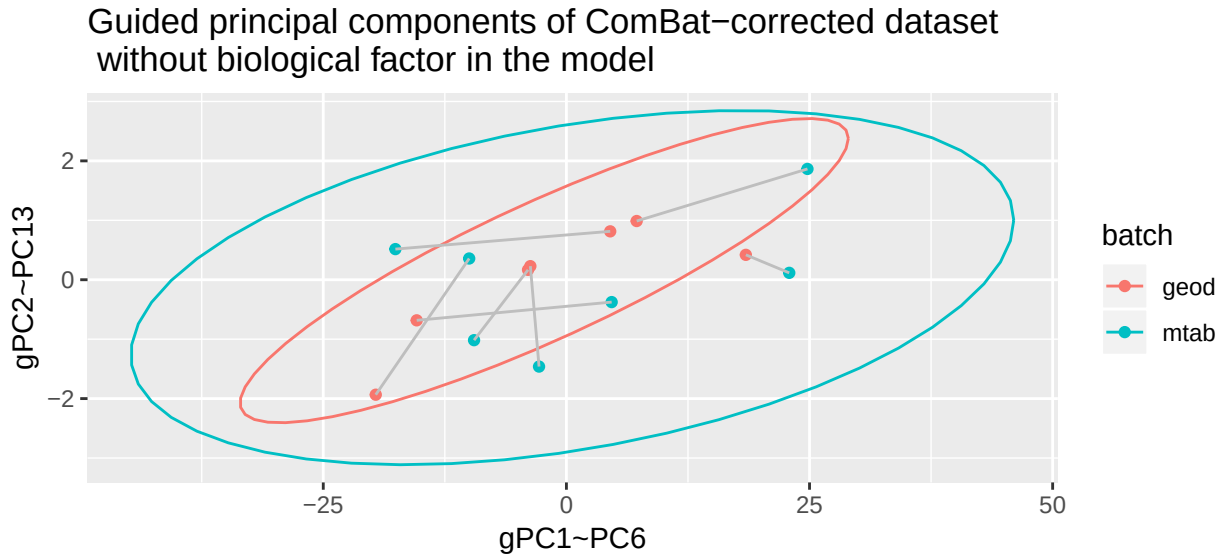
Empirical Bayes Method (ComBat)

Without biological factor in the model

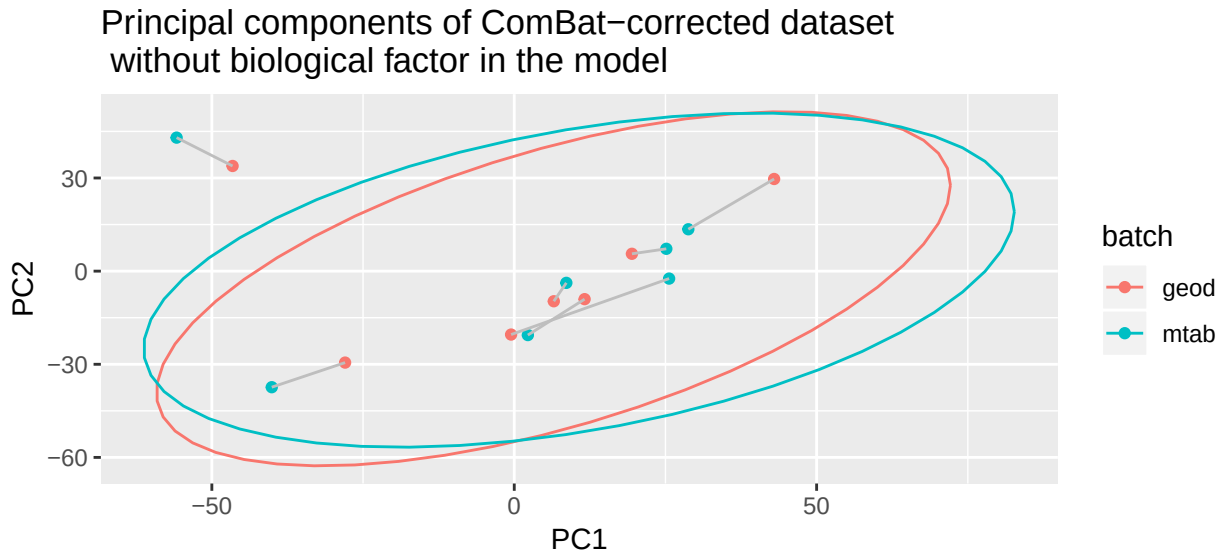
```
combat %>% gPCA(batch,nperm=1000) -> gcombat
```

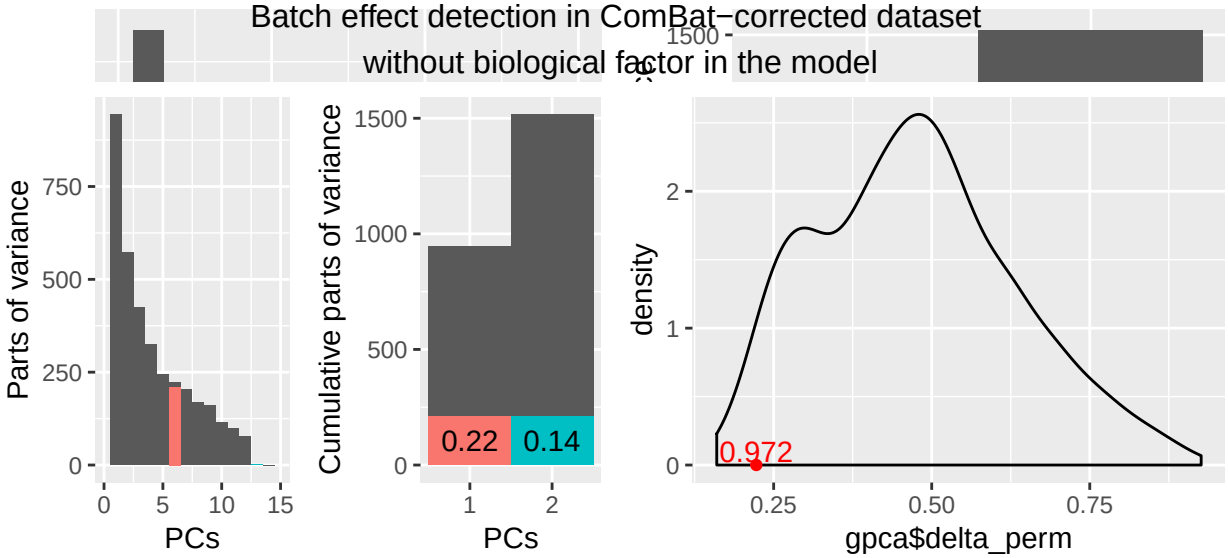
```
## Computing PCA
```

```
## Computing gPCA
## part of variance from gPC1 : 0.0591769845992101
## delta statistic : 0.22257039179507
## cumulative delta statistics :
## delta_1=0.22257039179507
##  delta_2=0.139229118871037
## variance ranks : 6
##  variance ranks : 13
## Estimating p-value
##
## p-value : 0.972
```




```
## Warning in MASS::cov.trob(data[, vars]): Probable convergence failure

## Warning in MASS::cov.trob(data[, vars]): Probable convergence failure
```



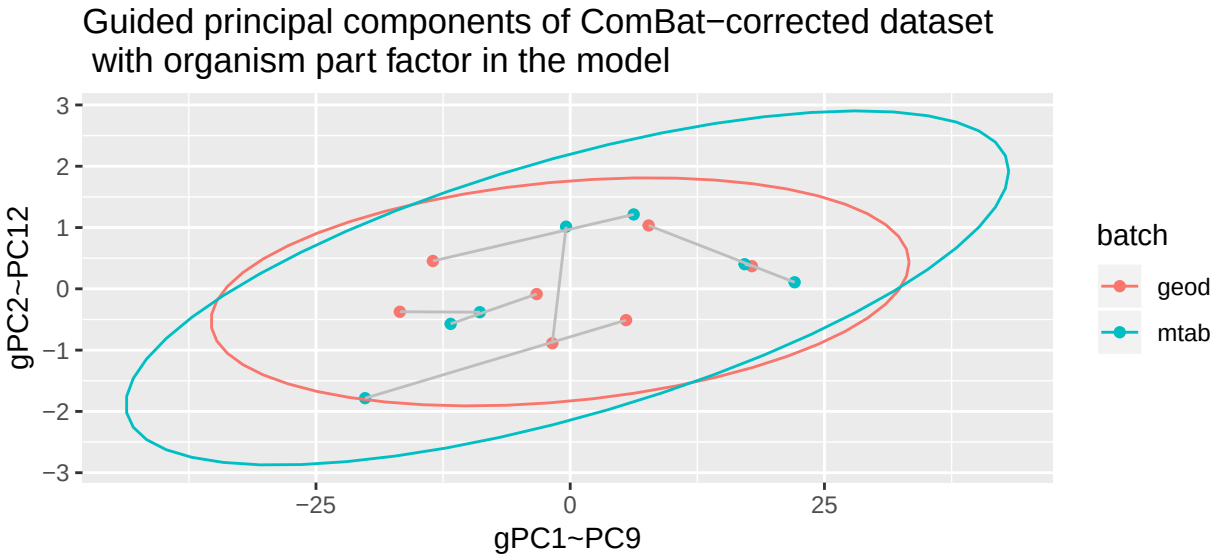


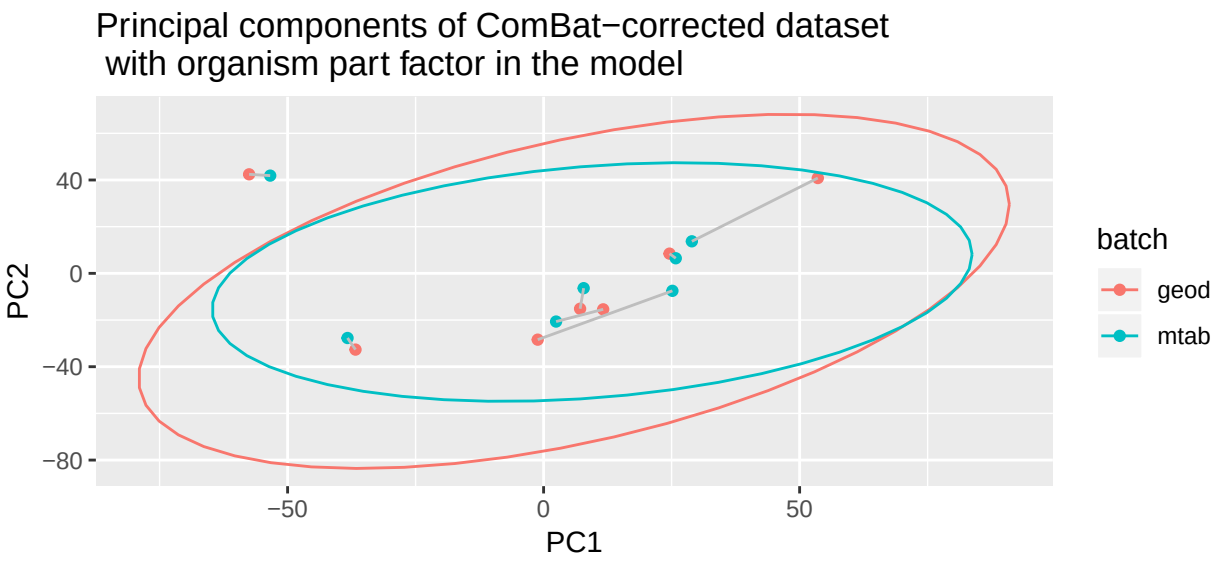
ComBat's gPCA score is very good and on the unguided PCA visualisation, samples seem to be clustered according to the organism part factor in most cases.

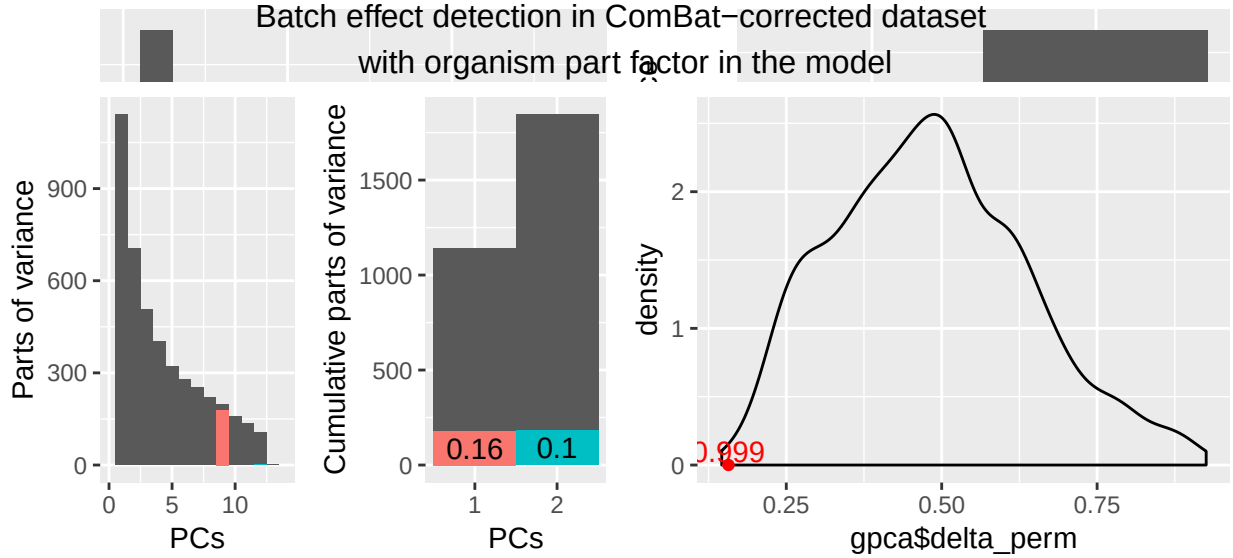
With biological factor in the model

```
ccombat %>% gPCA(batch,nperm=1000) -> gccombat
```

```
## Computing PCA
## Computing gPCA
## part of variance from gPC1 : 0.0405459642804096
## delta statistic : 0.157005269920255
## cumulative delta statistics :
## delta_1=0.157005269920255
##  delta_2=0.0974731232699797
## variance ranks : 9
##  variance ranks : 12
## Estimating p-value
##
## p-value : 0.999
```


Guided principal components of ComBat−corrected dataset
with organism part factor in the model
gPC2~PC12
gPC1~PC9
3
2
1
0
−1
−2
−3
−25
0
25
batch
geod
mtab


Principal components of ComBat−corrected dataset
with organism part factor in the model
PC2
PC1
40
0
−40
−80
−50
0
50
batch
geod
mtab

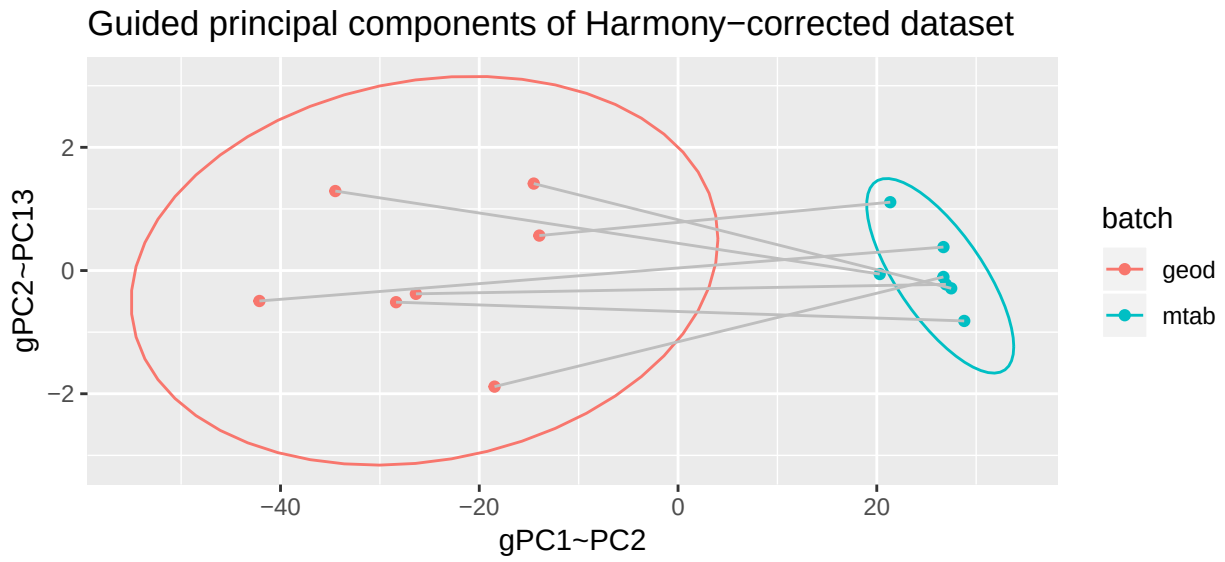


ComBat's gPCA score is even better when considering organism part as a factor in the model. Samples are still clustered according to organism part.

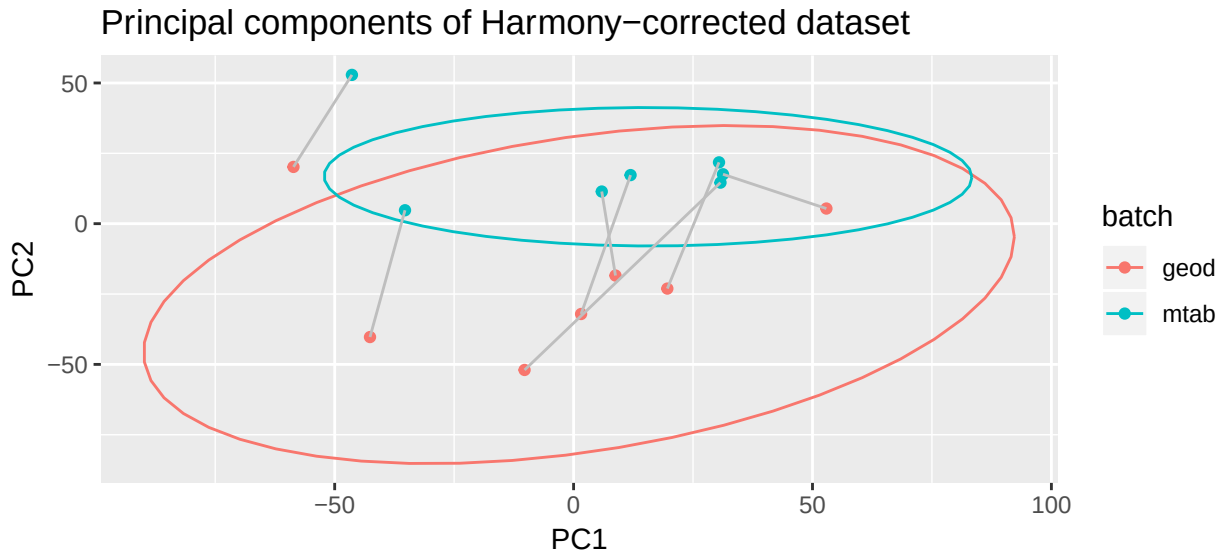
Harmony

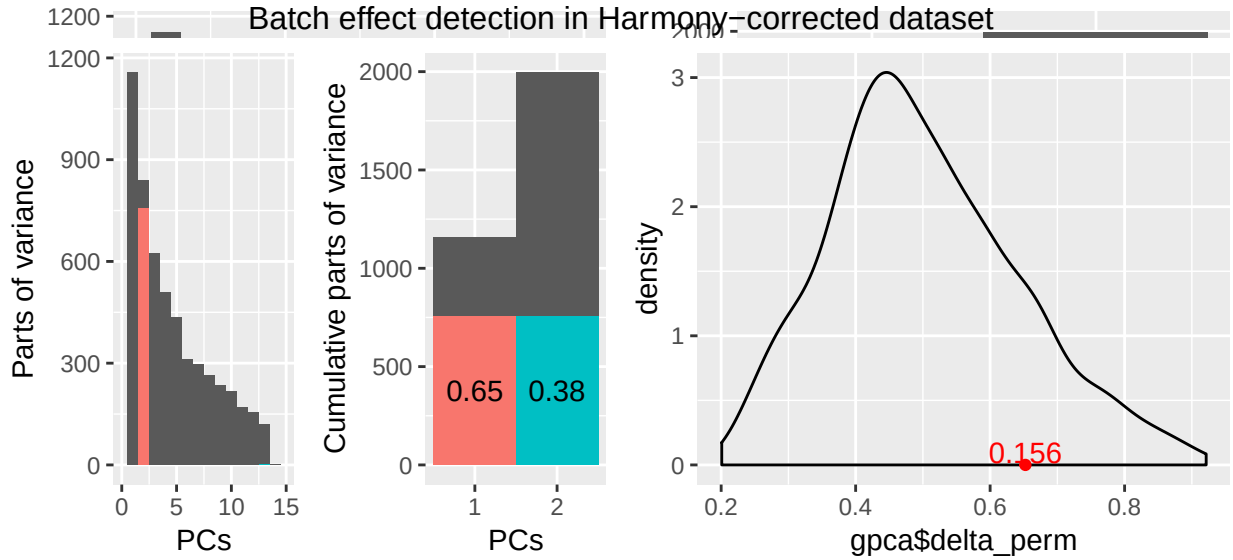
```
hm %>% gPCA(batch,nperm=1000)->ghm
```

```
## Computing PCA
## Computing gPCA
## part of variance from gPC1 : 0.141644391044073
## delta statistic : 0.652490168543371
## cumulative delta statistics :
## delta_1=0.652490168543371
##  delta_2=0.378693145383215
## variance ranks : 2
##  variance ranks : 13
## Estimating p-value
##
## p-value : 0.156
```




```
## Warning in MASS::cov.trob(data[, vars]): Probable convergence failure
```



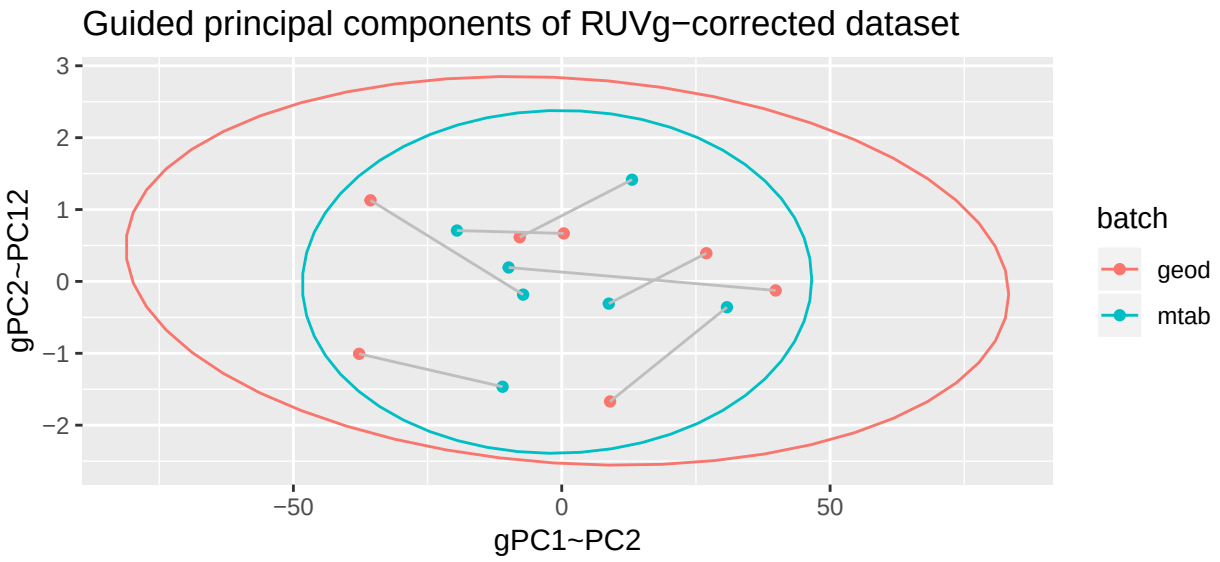


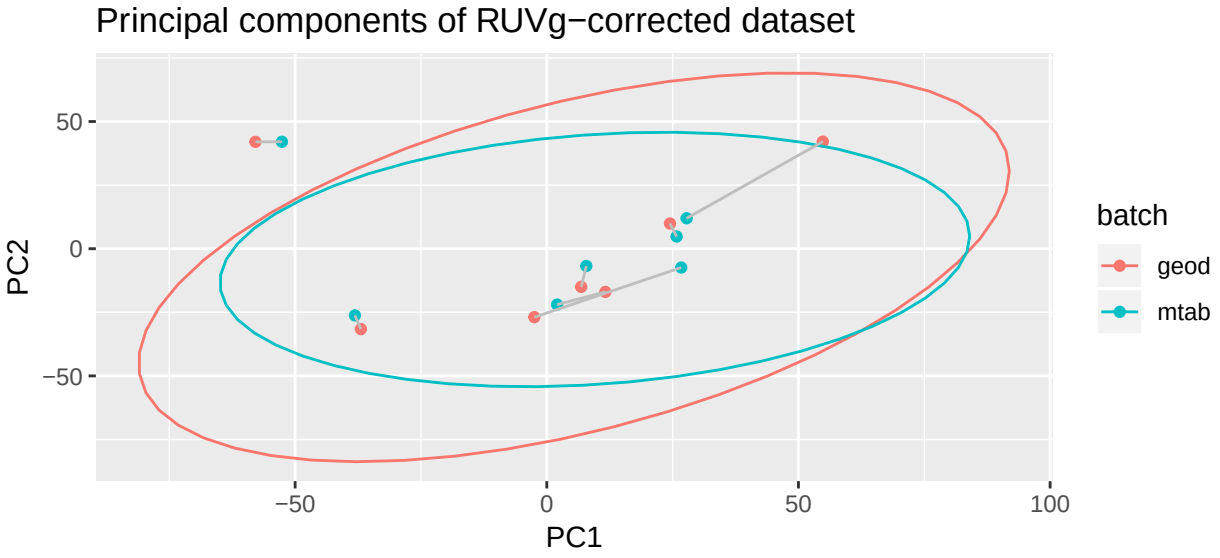
Harmony doesn't seem to work well here. Maybe we lack data for this method.

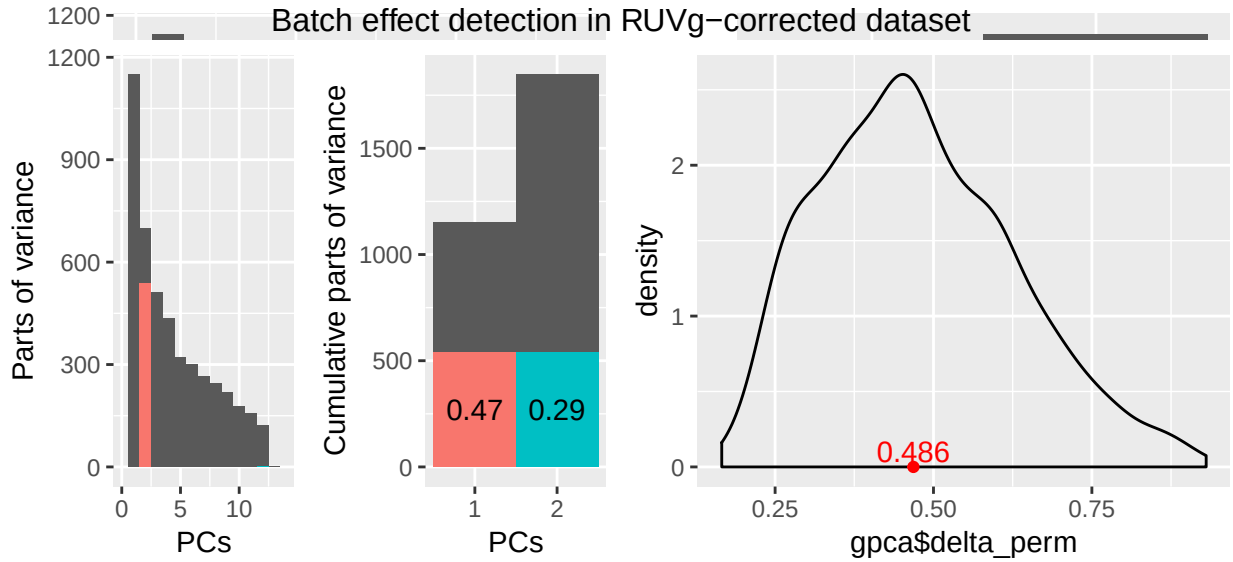
RUVg

```
ruvg$normalizedCounts %>% gPCA(batch,nperm=1000) -> gruvg
```

```
## Computing PCA
## Computing gPCA
## part of variance from gPC1 : 0.11712368715257
## delta statistic : 0.468299997314656
## cumulative delta statistics :
## delta_1=0.468299997314656
##  delta_2=0.291849572260883
## variance ranks : 2
##  variance ranks : 12
## Estimating p-value
##
## p-value : 0.486
```

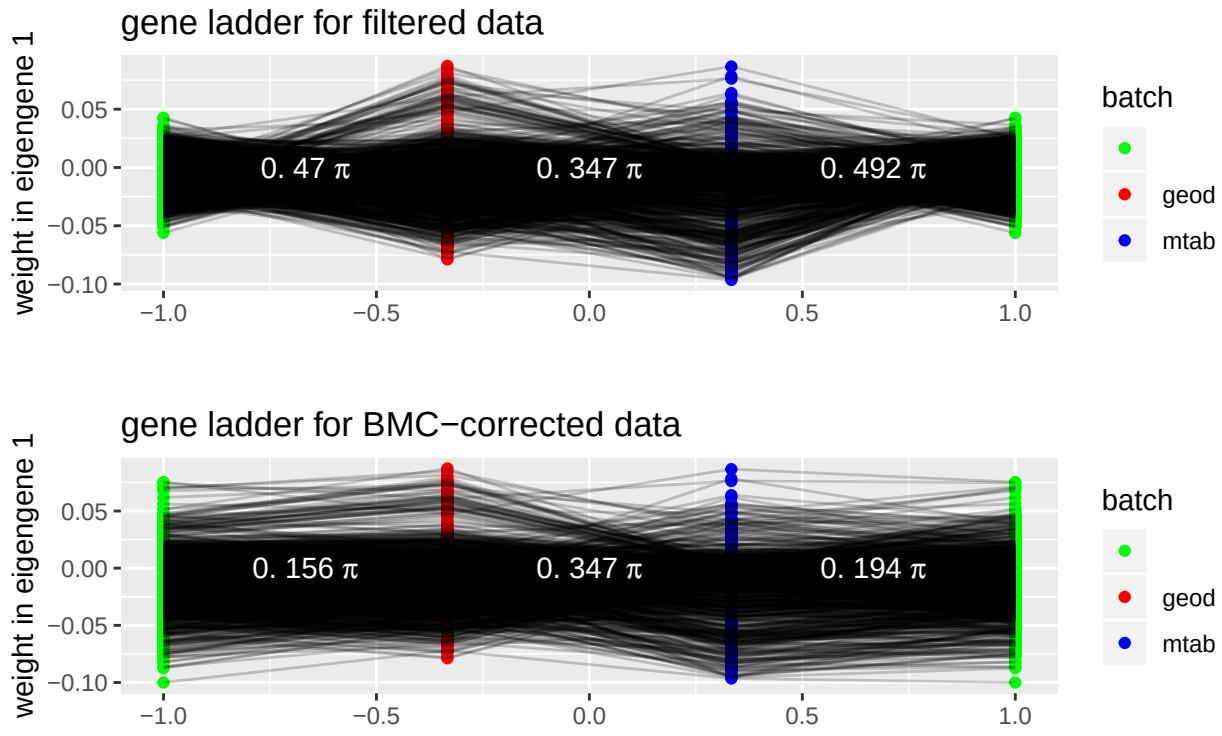

Guided principal components of RUVg−corrected dataset
gPC2~PC12
gPC1~PC2
batch
geod
mtab

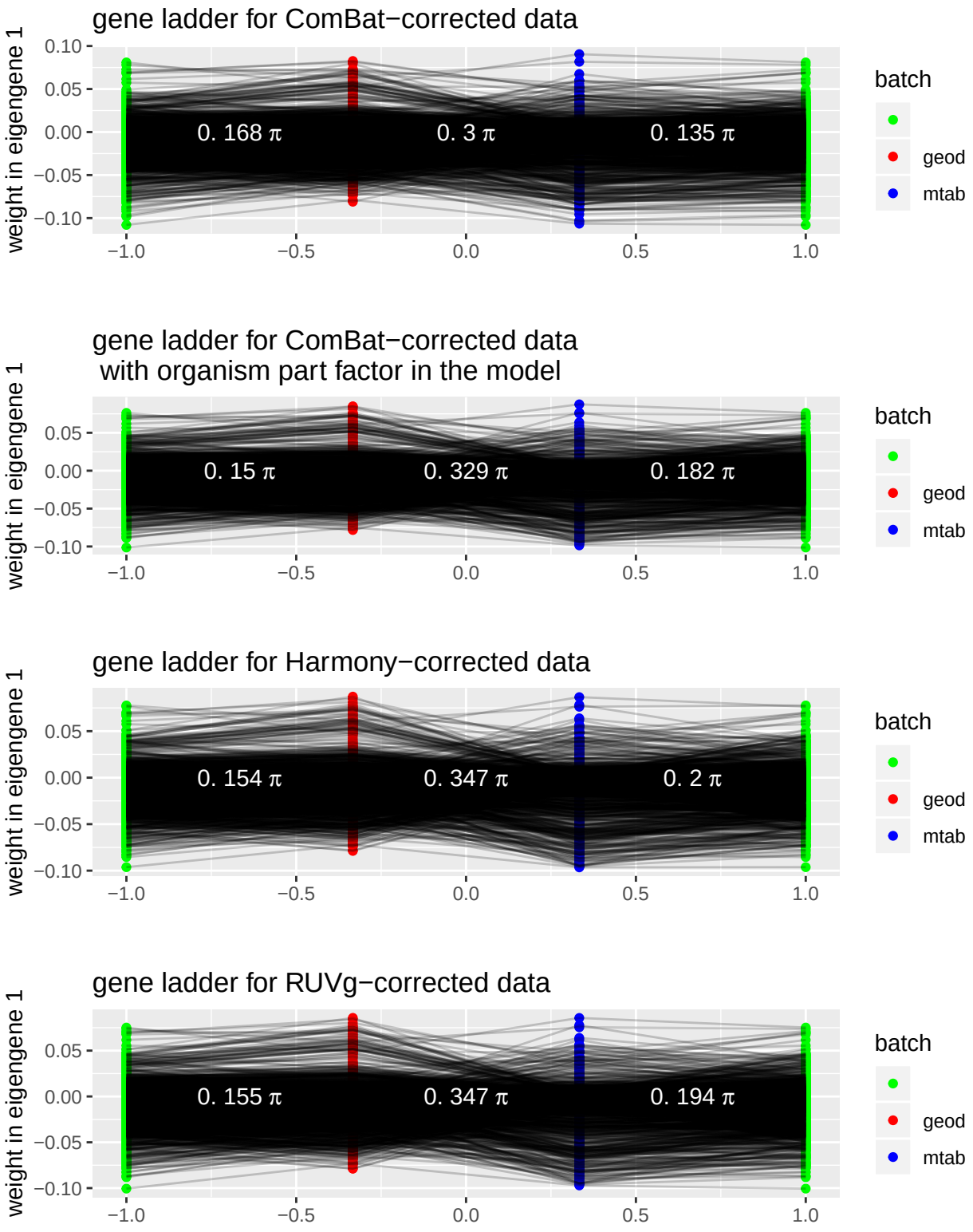

Principal components of RUVg−corrected dataset
PC2
PC1
batch
geod
mtab



RUVg has less impressive score than ComBat although it seems to cluster the samples according to organism part factor almost the same way as ComBat, at least on the first two principal components.

Using genemap





It appears in these graphics and angles that :

- this detection tool is sensitive to correction methods : the angles $\widehat{PC_1^1, PC_1^m}$ and $\widehat{PC_1^2, PC_1^m}$ become much smaller after correction, whatever the method used.

- the correction methods that were used don't seem to have an effect on the PCs of the separated dataset, except ComBat. Indeed the angle $\widehat{PC_1^1, PC_1^2}$ decreases slightly after ComBat correction, which means that ComBat try somehow to make the two experiments agree more...

One can wonder whether this last bullet point is good or not. Should the correction force the two datasets to agree on the important genes ? Isn't there a risk to lose biological information by this way ?

Conclusion

For the integration of such small datasets, ComBat seems to be a very good method for batch effect correction, which is not surprising as it uses a bayesian framework. Although this quality of bayesian framework could make