The gPCA Package for Identifying Batch Effects in High-Throughput Genomic Data

Sarah Reese*

July 31, 2013

Batch effects are commonly observed systematic non-biological variation between groups of samples due to experimental artifacts, such as processing date, lab, or technician. Combining samples from multiple batches can cause the true biological variation in a high-throughput experiment to be obscured by variation due to batch.

1 Guided Principal Components Analysis

Guided principal components analysis (gPCA) is an extension of principal components analysis (PCA) that replaces the data $\mathbf{X}$ matrix in the singular value decomposition (SVD) of PCA with $\mathbf{Y}'\mathbf{X}$ such that

$$\mathbf{Y}'\mathbf{X} = \mathbf{UDV}'$$

where $\mathbf{Y}$ is an $n \times b$ indicator matrix where n denotes sample and b denotes batch. For $k = 1, \ldots, b$ batches, each is comprised of n_k observations such that $\sum_{k=1}^{b} n_k = n$. The indicator matrix consists of b blocks with n_k rows for $k = 1, \ldots, b$, and k columns where, for each block,

$$\mathbf{Y}_k = \begin{cases} \mathbf{1} & \text{if } k = b \\ \mathbf{0} & \text{otherwise .} \end{cases}$$

Performing SVD on $\mathbf{Y}'\mathbf{X}$ results in a $b \times b$ batch loadings matrix $\mathbf{U}$ and a $p \times p$ probe loadings matrix $\mathbf{V}$. Large singular values (the diagonal elements of the $q \times q$ matrix $\mathbf{D}$ where $q = \min(n, p)$) imply that the batch is important for the corresponding principal component. gPCA guides the SVD to look for batch effects in the data based on the batch indicator matrix $\mathbf{Y}$, which can be defined to indicate any type of potential batch effect, such as time of hybridization, plate, or other experimental artifact.

In Reese et al. [6], we proposed a test statistic δ that quantifies the proportion of variance due to batch effects in experimental genomic data. The proportion of total variance due to batch is taken to be the ratio of the variance of the first principal component from gPCA to the variance of the first principal component from unguided PCA

$$\delta = \frac{\text{var}(\mathbf{XV}_{g_1})}{\text{var}(\mathbf{XV}_{u_1})}$$

*reesese@vcu.edu

where g indicates gPCA and u indicates unguided PCA. $\mathbf{V}$ is the matrix of probe loadings resulting from gPCA or PCA, respectively. Large values of δ (values near 1) imply that the batch effect is large.

To determine whether δ is significantly larger than would be expected by chance, a p-value is estimated using a permutation distribution created by permuting the batch vector $M = 1000$ times so that δ_{p_m} is computed for $m = 1, \ldots, M$ where p indicates the permutation. Here δ_{p_m} is the proportion of the total variance due to the first principal component from the m^{th} permutation from gPCA to the total variance due to the first principal component from the m^{th} permutation from unguided PCA. A one-sided p-value (testing $H_0 : \delta_{p_m} = \delta$ versus $H_1 : \delta_{p_m} > \delta$) is estimated as the proportion of times the observed δ was in the extreme tail of the permutation distribution

$$p\text{-value} = \frac{\sum_{m=1}^{M} (\delta_{p_m} > \delta)}{M}.$$

For more details on gPCA see Reese et al. [6].

2 R Package

The `gPCA` package includes four example data sets, the `gPCA.batchdetect()` function that produces the δ statistic and corresponding p-value, and additional visualization functions.

2.1 Data

Four data sets are included in the `gPCA` package, three simulated data sets and one case study data set. The case study data (`data(caseDat)`) contains copy number variation data with $n = 500$ observations and $p = 1000$ features that were retained after a variance filter was applied.

The simulated data represents copy number data under three scenarios: (1) feature data (here, feature denotes probe) with no phenotypic variable (`data(nopheDat)`); (2) feature data with a high variance phenotypic variable (`data(highpheDat)`); and (3) feature data with a low variance phenotypic variable (`data(lowpheDat)`). The feature data were generated independently from a multivariate normal distribution with 1000 features and 90 observations. Data with two batches and two phenotypes were simulated. Batch mean vectors $\boldsymbol{\mu}_{b_1} = \mathbf{0}$ and $\boldsymbol{\mu}_{b_2} = \mathbf{1}$ and batch variance $\sigma_b^2\mathbf{I}$ where $\sigma_b^2 = 0.5$ were used to simulate the data. The proportion of features affected by batch was `bprop` $= 0.01$ for the no phenotype scenario and `bprop` $= 0.05$ for the high and low variance phenotype scenarios.

For the scenarios with phenotypic effects, the proportion of features affected by phenotype was `pprop` $= 0.1$. The phenotypic mean vectors were $\boldsymbol{\mu}_{p_1} = \mathbf{0}$ and $\boldsymbol{\mu}_{p_2} = \mathbf{1}$ and the phenotypic variance was $\sigma_p^2\mathbf{I}$ where $\sigma_p^2 = 2$ for the high variance phenotype scenario and $\sigma_p^2 = 0.2$ for the low variance phenotype scenario. Reese et al. [6] provides an in depth description of the data simulations.

For all four data sets, the first column of the data frame containing the data contains the `batch` vector which indicates batch for the n observations. The rest of the data frame contains the uncentered feature data.

2.2 Application

The δ statistic, corresponding p-value from the permutation test, and various other measures are output by the `gPCA.batchdetect()` function. The syntax for this function is

```
> out<-gPCA.batchdetect(x=data,batch=batch,center=FALSE,
+          scaleY=FALSE,filt=NULL,nperm=1000,seed=NULL)
```

where `x` is the $n \times p$ matrix of feature data $\mathbf{X}$, `batch` is a length n vector indicating batch which is used to calculate the $\mathbf{Y}$ matrix for gPCA. The option `center` is a logical indicating whether or not `data` is centered where `center=TRUE` if the data `x` is already centered. `scaleY` is a logical indicating whether the batch indicator matrix `Y` is to be scaled by the batch sample size n_k. `nperm` indicates how many permutations will be used for calculating the permutation test statistic (defaults to 1000), `filt` gives the number of features to retain when applying a variance-based filter to the data (defaults to `NULL` indicating no filter applied), and `seed` sets `set.seed(seed)`. Note that `x` must be complete data (i.e. contain no missing values) and the class of `x` must be `"matrix"`. The function, when run actively, will ask if mean-value imputation should be performed for any missing values, but when run passively will cause an error.

The `gPCA.batchdetect()` function outputs the value of the statistic δ, the associated p-value, the batch vector `batch`, the M values of δ_p resulting from the permutation test, the proportion of variance associated with the first principal component from unguided (`PCu`) and guided (`PCg`) PCA, as well as the cumulative variance associated with all n principal components resulting from unguided PCA (`cumulative.var.x`) and the cumulative variance associated with all b principal components resulting from gPCA (`cumulative.var.g`).

The `gPCA` package also has three functions to visualize the data. The function `gDist` produces a density plot of the δ_p values output by the `gPCA.batchdetect` function. The function `PCplot` produces principal component plots of either the unguided or guided principal components and allows for either directly comparing the first two principal components, or comparing the first `npcs` principal components. Finally, the function `CumulativeVarPlot` produces a plot of the cumulative variance from guided or unguided PCA.

```
> gDist(out)
> PCplot(out,ug="guided",type="1v2")
> PCplot(out,ug="guided",type="comp",npcs=3)
> CumulativeVarPlot(out,ug="unguided",col="blue")
```

3 Example

We will discuss a brief example using `caseDat` data from the `gPCA` package. We first load the data `caseDat` and assign the first column to `batch`. The rest of the data frame is the feature data, so we assign that to `dat` and re-classify it as a matrix. Since the `caseDat` feature data is already centered, we set `center=TRUE`. The value of the test statistic δ and the corresponding p-value are easily printed and the percent of total variation that is explained by batch is calculated.

```
> data(caseDat)
> batch<-caseDat$batch
> data<-caseDat$data
> out<-gPCA.batchdetect(x=data,batch=batch,center=TRUE)
> out$delta ; out$p.val

[1] 0.5723698

[1] "<0.001"

> ((out$varPCg1-out$varPCu1)/out$varPCg1)*100

[1] 96.29305
```

We can also plot the distribution of the δ_p values from the permutation test and see where our test statistic δ (represented by the red dashed line) falls in comparison (Figure 1). Plots of the first versus the second principal components from gPCA can be plotted (Figure 2) as well as a sample of the first few principal comparisons (Figure 3).

4 Conclusion

The `gPCA` package provides functionality to test for batch effects in high-throughput genomic data using the function `gPCA.batchdetect()`. The ability to detect batch effects in genomic data allows further batch correction procedures such as batch mean-centering [7], distance weighted discrimination (DWD) [1, 2, 3, 5], or empirical Bayes [4], to be employed to attempt to remove the unwanted variation due to batch effects. However, correcting for batch when there is no significant batch effect may result in removing biological variation instead of the systematic non-biological variation due to batch. This package provides the ability to perform a test to detect batch effects.

5 Session Info

```
> sessionInfo()

R version 3.0.1 (2013-05-16)
Platform: x86_64-apple-darwin10.8.0 (64-bit)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
```

```
> gDist(out)
```

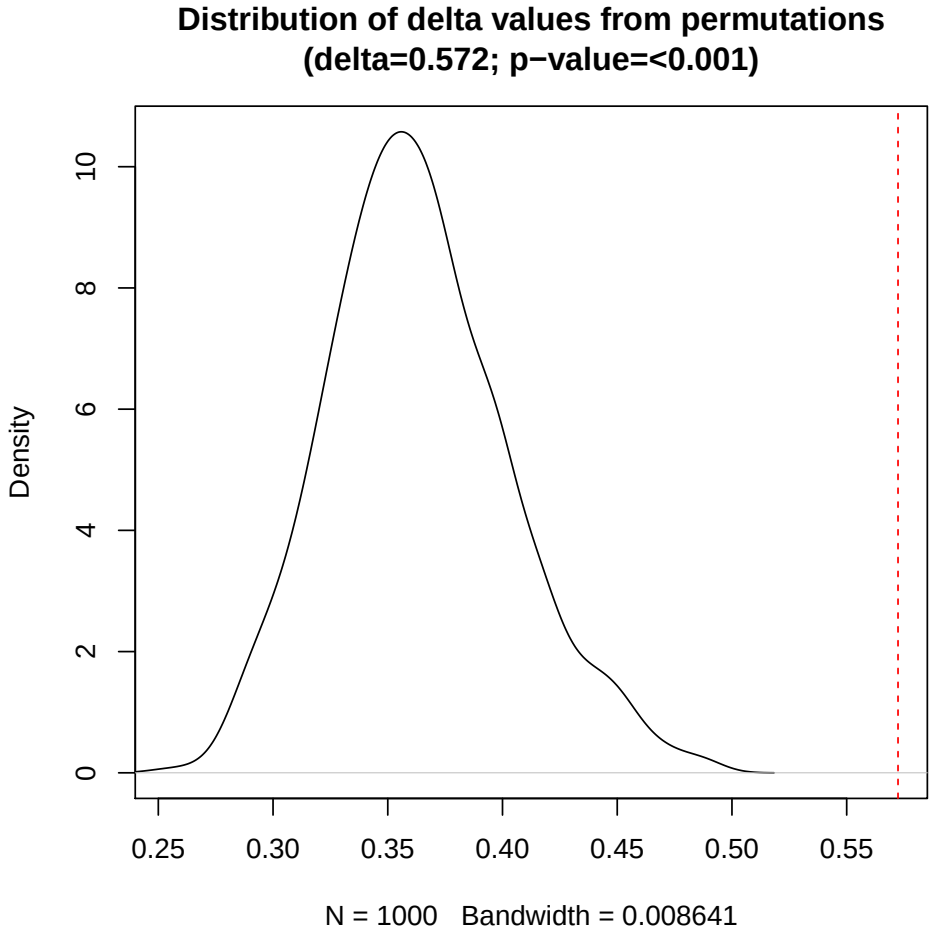

Figure 1: Distribution plot of δ_p values

```
> par(mai=c(0.8,0.8,0.1,0.1),cex=0.8)
> PCplot(out,ug="guided",type="1v2")
```

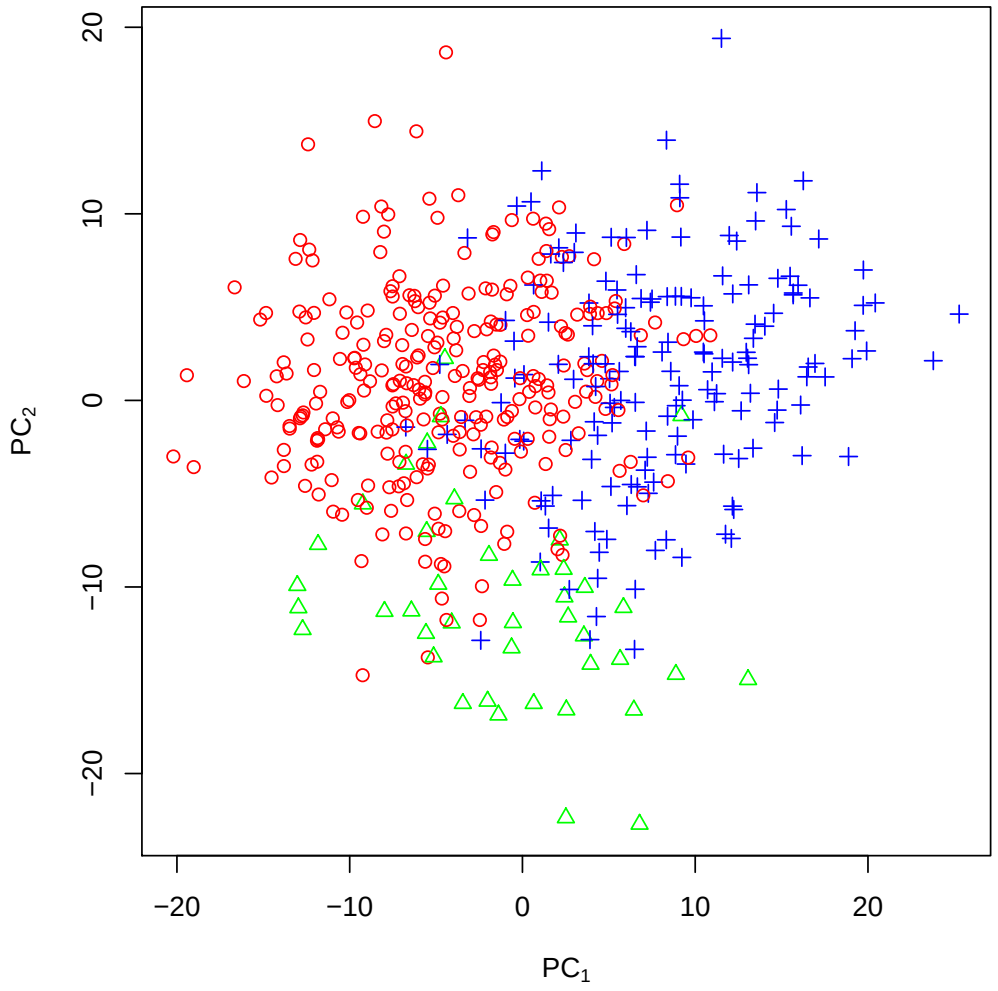

Figure 2: Principal components plot of first two principal components from gPCA

```
> par(mai=c(0.65,0.65,0.1,0.1),cex=0.8)
> PCplot(out,ug="guided",type="comp",npcs=3)
```

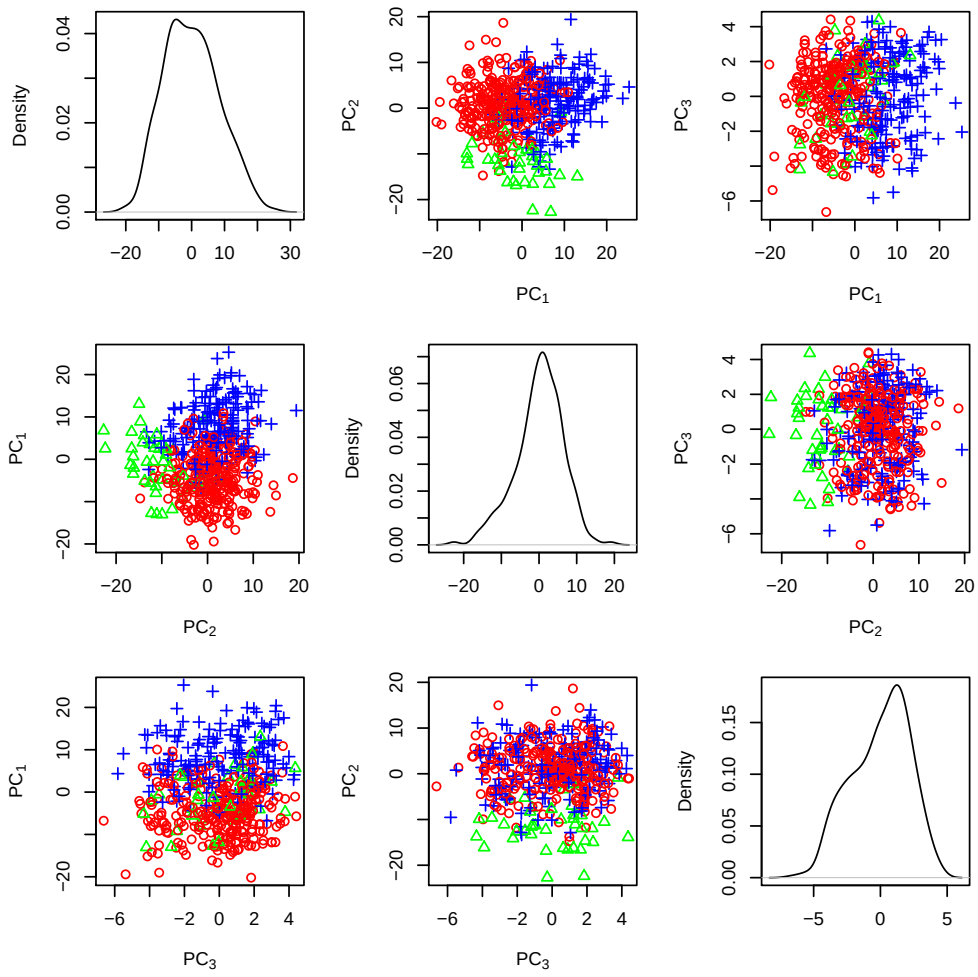

Figure 3: Principal components plots of the first three principal components with density plots of the principal components on the diagonal.

```
[1] gPCA_1.0

loaded via a namespace (and not attached):
[1] tools_3.0.1
```

References

[1] Benito, M., Parker, J., Du, Q., Wu, J., Xiang, D., P., C. M., and Marron, J. S. Adjustment of systematic microarray data biases. *Bioinformatics*, 20(1):105–114, 2004.

[2] Huang, H., Lu, X., Liu, Y., Haaland, P., and Marron, J. R/DWD: distance-weighted discrimination for classification, visualization and batch adjustment. *Bioinformatics*, 28(8):1182–1183, 2012.

[3] Huang, H., Liu, Y., Du, Y., Perou, C. M., Hayes, D. N., Todd, M. J., and Marron, J. S. Multiclass distance weighted discrimination. *Journal of Computational and Graphical Statistics*, 2012.

[4] Johnson, W. E., Li, C., and Rabinovic, A. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics*, 8(1):118–127, 2007.

[5] Marron, J. S. and Todd, M. Distance weighted discrimination. Aug. 2002.

[6] Reese, S. E., Archer, K. J., Therneau, T. M., Atkinson, E. J., Vachon, C. M., de Andrade, M., Kocher, J. A., and Eckel-Passow, J. E. A new statistic for identifying batch effects in high-throughput genomic data that uses guided principal components analysis. *Bioinformatics*, under review.

[7] Sims, A. H., Smethurst, G. J., Hey, Y., Okoniewski, M. J., Pepper, S. D., Howell, A., Miller, C. J., and Clarke, R. B. The removal of multiplicative, systematic bias allows integration of breast cancer gene expression datasets – improving meta-analysis and prediction of prognosis. *BMC Medical Genomics*, 1(42), 2008.