[Removing hidden batch effects] [http://master.bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html#removing-hidden-batch-effects]

[批次效应][Tackling the widespread and critical impact of batch effects in high-throughput data]就是在同一条件下部分测量数据表现出了不同的趋势(qualitatively different behavior), 同时该差异趋势和生物学或变量无关(unrelated to the biological or scientific variables). 例如, 相同实验的不同部分发生在的不同操作时间, 或者由不同的实验人员负责, 使用了不同的试剂批号, 芯片或设备.

用于生物研究的许多技术, 包括高通量测序, microarrays, bead chips, mass spectrometers等, 通过专业人员操作以实现准确的测量结果. 当实验过程中的这些条件发生了改变, 许多测量得到量化值将会受到生物学和非生物学因素影响. **批次效应的发生是由于实验室条件, 试剂批号和人员的不同所导致的.**

标准化(normalization)是用于针对个体样本朝着整体方向调整的分析技术, 因此个体之间可以适当的比较. 标准化步骤是基因表达分析的标准流程.

但是标准化过程不会去除批次差异, 批次效应影响特殊的基因集, 同时可能以不同的方式影响不同的基因. 在一些情况下, 由于批次和其他技术效应破坏了标准化的假设思想, 这些标准化过程可能甚至加剧了高通量检测的批次效应.

去批次效应方法

Empirical Bayes method(ComBat)

Singular-value decomposition(SVD)

Distance weighted discrimination(DWD)

sva

Surrogate Variable Analysis

sva包用于去除高通量测序中批次效应和其他非目的性变化. sva针对高维度数据集识别并构建代理变量(surrogate variable)。代理变量直接和高维度数据共变化构建(covariated constructed, 例如, 基因表达/RNA测序/甲基化/脑成像数据), 然后用于随后分析来调整未知的, 非模型的, 或隐藏的噪音.

sva通过三种方式去除人工效应(artifacts): 针对高通量实验未知来源的变化识别并评估代理变量; 直接使用ComBat去除批次效应; 通过已知的质控探针去除批次效应.

Setting up the data

首先恰当设置数据和构建模型矩阵. 该数据矩阵应包含features(genes, transcripts, voxels)行和样本列信息, 是典型的基因表达分析计数矩阵. sva假设存在需要考虑的两种类型变量: adjustment 变量和 interested 变量. 例如基因表达研究中, 设置cancer vs. control为感兴趣变量, 而患者年量, 性别, 数据产生时间等为调整变量.

需要构建两个矩阵, 'full model'和'null model', 'null model‘ 为包含所有调整变量但不含感兴趣变量的矩阵, 'full model'为包含感兴趣变量和调整变量的矩阵. 其假设思想为, 分析感兴趣变量和基因表达的相关性, 同时使用调整标量进行调整. 使用`model.matrix`构建模型矩阵.

Setting up the data from an ExpressionSet

膀胱癌数据存于表达数据集- a Bioconductor object used for storing gene expression data. 获得表型数据

```
pheno <- pData(bladderEset)
```

```
> pheno
           sample  outcome batch cancer
GSM71019.CEL      1   Normal     3 Normal
GSM71020.CEL      2   Normal     2 Normal
GSM71021.CEL      3   Normal     2 Normal
GSM71022.CEL      4   Normal     3 Normal
GSM71023.CEL      5   Normal     3 Normal
GSM71024.CEL      6   Normal     3 Normal
GSM71025.CEL      7   Normal     2 Normal
GSM71026.CEL      8   Normal     2 Normal
GSM71028.CEL      9 sTCC+CIS     5 Cancer
GSM71029.CEL     10 sTCC-CIS     2 Cancer
GSM71030.CEL     11 sTCC-CIS     5 Cancer
GSM71031.CEL     12 sTCC-CIS     2 Cancer
```

同时获得表达数据集的表达数据

```
edata <- exprs(bladderEset)
```

```
> head(edata)
         GSM71019.CEL GSM71020.CEL GSM71021.CEL GSM71022.CEL GSM71023.CEL GSM71024.CEL
1007_s_at    10.115170     8.628044     8.779235     9.248569    10.256841    10.023133
1053_at       5.345168     5.063598     5.113116     5.179410     5.181383     5.248418
117_at        6.348024     6.663625     6.465892     6.116422     5.980457     5.796155
121_at        8.901739     9.439977     9.540738     9.254368     8.798086     8.002870
1255_g_at     3.967672     4.466027     4.144885     4.189338     4.078509     3.919740
1294_at       7.775183     7.110154     7.248430     7.017220     7.896419     7.944676
         GSM71025.CEL GSM71026.CEL GSM71028.CEL GSM71029.CEL GSM71030.CEL GSM71031.CEL
1007_s_at    9.108034     8.735616     9.803271    10.168602     8.420904    10.194269
1053_at      5.252312     5.220931     5.595771     5.025180     5.909075     5.307699
117_at       6.414849     6.846798     5.841478     6.352257     5.967566     6.171909
121_at       9.093704     9.263386     7.789240     9.834564     7.844720     7.862812
1255_g_at    4.402590     4.173666     3.590649     4.338196     3.952783     3.985398
1294_at      7.469767     7.281925     7.367814     7.825735     7.401377     8.252998
         GSM71032.CEL GSM71033.CEL GSM71034.CEL GSM71035.CEL GSM71036.CEL GSM71037.CEL
```

构建全矩阵'full model', 包含感兴趣变量和调整变量

```
mod <- model.matrix(~as.factor(cancer), data=pheno)
```

针对pheno中的cancer变量构建矩阵

```
> head(mod)
           (Intercept) as.factor(cancer)Cancer as.factor(cancer)Normal
GSM71019.CEL           1                      0                       1
GSM71020.CEL           1                      0                       1
GSM71021.CEL           1                      0                       1
GSM71022.CEL           1                      0                       1
GSM71023.CEL           1                      0                       1
GSM71024.CEL           1                      0                       1
>
```

空矩阵'null model'仅包含调整参数. 因为不校正分析过程中的其他变量, 所以仅包截距项

```
mod0 <- model.matrix(~1, data=pheno)
```

```
> head(mod0)
            (Intercept)
GSM71019.CEL           1
GSM71020.CEL           1
GSM71021.CEL           1
GSM71022.CEL           1
GSM71023.CEL           1
GSM71024.CEL           1
```

使用sva函数评估批次和其他人为效应

Applying the sva function to estimate batch and other artifacts

首相识别潜在的需要评估的因子数目(如果不设置`n.sv`参数, 将会评估因子数目)

`num.sv`用于评估包含于差异分析模型中的代理变量数量, 默认过程是基于排序过程(a permutation procedure), 同时还包含一个渐进式过程(an interface to the asymptotic approach)

```
n.sv <- num.sv(edata, mod, method="leek")
```

```
n.sv 2
```

使用sva函数评估代替变量. `sva`用于评估来自microarray数据的人工影响, `svaseq`用于评估来自RNA-seq count-based的人工影响(sva, svaseq, this function is the implementation of the iteratively re-weighted least squares approach for estimating surrogate variables), `ComBat`用于去除来自microarry的已知批次效应, `fsva`用于针对prediction problems去除批次效应

```
> svobj <- sva(edata, mod, mod0, n.sv=n.sv)
Number of significant surrogate variables is:  2
Iteration (out of 5 ):1  2  3  4  5
```

输出

```
> names(svobj)
[1] "sv"        "pprob.gam" "pprob.b"   "n.sv"
```

`sv` 评估得到的代替变量, 每列一个

`pprob.gam` 每个基因被heterogeneity所影响的后验概率(posterior probabilities), 也就是每个基因和一个或多个潜在变量相关的后验概率

`pprob.b` 每个基因被mod影响的后验概率(posterior probabilities), 也就是每个基因与感兴趣变量相关的后验概率

`n.sv` 显著性代替变量数目

Adjusting for surrogate variables using the f.pvalue function

`f.pvalue`可用于计算数据矩阵每一行的参数的F-test p-values(parametric F-test p-values). 该例中, 对应计算矩阵22283行的参数的F-test p-values. F-test比较模型`mod`和`mod0`, 两模型必须为nested模型, 因此`mod0`中的所有变量必须出现在`mod`中.

首先计算涉及到cancer status的差异表达的F-test p-values, 不针对代理变量进行调整. Adjust them for multiple testing, 计算显著性Q-value小于0.05的数目.

`f.pavlue`通过简单线性代数计算由mod和mod0组成的nested模型每行的f检验值. 其中mod0列必须为mod列的子集.

```
> head(mod0)
            (Intercept)
GSM71019.CEL           1
GSM71020.CEL           1
GSM71021.CEL           1
GSM71022.CEL           1
GSM71023.CEL           1
GSM71024.CEL           1
> head(mod)
            (Intercept) as.factor(cancer)Cancer as.factor(cancer)Normal
GSM71019.CEL           1                       0                      1
GSM71020.CEL           1                       0                      1
GSM71021.CEL           1                       0                      1
GSM71022.CEL           1                       0                      1
GSM71023.CEL           1                       0                      1
```

```
pValues <- f.pvalue(edata, mod, mod0)
```

```
qvalues <- p.adjust(pValues, method="BH")
```

```
> table(qValues < 0.05)

FALSE  TRUE
 7090 15193
```

可见存在相当高的人工差异. 调整代理变量, 执行相同分析

首先在mod和mod0中包含代理变量, 因为我们向调整代理变量, 因此作为调整变量的值必须包含在两模型中, 计算p值和q值

```
modSv <- cbind(mod, svobj$sv)
```

```
mod0Sv <- cbind(mod0, svobj$sv)
```

```
pValuesSv <- f.pvalue(edata,modSv,mod0Sv)
```

```
qvaluesSv <- p.adjust(pValuesSv, method="BH")
```

```
> table(qValuesSv < 0.05)

FALSE  TRUE
 7470 14813
```

Now these are the adjusted P-values and Q-values accounting for surrogagte variables

---

~~**Adjusting for surrogate variables using the limma package**~~

首先, 包含代理变量拟合线性模型

```
fit <- lmFit(edata, modSv)
```

然后, 使用limma函数执行常规分析. 计算针对cancer的差异表达.

计算cancer/normal项目比对. 在比对时不包含代理变量, 因为它们仅用于校正分析

`contrast.matrix <- cbind("C1"=c(-1,1,0,rep(0,svobj$n.sv)), "C2"=c(0,-1,1,rep(0,svobj$n.sv)))`

`fitContrasts <- contrasts.fit(fit, contrast.matrix)`

然后使用`eBayes`计算统计检验

`eb <- eBayes(fitContrasts)`

`topTableF(eb, adjust="BH")`

Applying the ComBat function to adjust for known batches

`ComBat`使用经验贝叶斯模型调整来自microarray的已知的批次效应(an empirical Bayesian framework)数据, 同时可以通过设置`par.prior=FALSE`选项使用noparametric empirical Bayesian adjustments. 此外, 通过设置`prior.plots=TRUE`选型给出参数评估图, 其中.黑色表针对经验批次相应密度的核心评估, 红色为参数评估, 通过该图可以确保评估的合理性.

因此, 首先得知道数据中的批次变量

`batch <- pheno$batch`

和`sva`一样, 构建包含调整变量(adjustment variables)模型矩阵(just as with sva, we then need to create a model matrix for the adjsutment variables, including the variable fo interest). 这里在构建模型矩阵时不需要包含批次, 该批次将在包含在后续`ComBat`函数中. 该例子中不存在其他的调整变量, 因此简单拟合截距项

`modcombat <- model.matrix(~1, data=pheno)`

注意, 调整变量将在`ComBat`函数中处理. 如果想要校正含有p个不同水平的分类变量, 就需要制定p-1个indicator变量(for this covariate)

推荐使用`model.matrix`函数设置, 对于连续型调整变量, 在模型矩阵中提供包含该共变量(covariate values)的向量

这里使用parametric empirical Bayesian adjustments

`ComBat` 针对已知的批次共变量调整批次效应, 输入数据为干净的标准化后的数据, 输出为校正后批次效应的表达矩阵

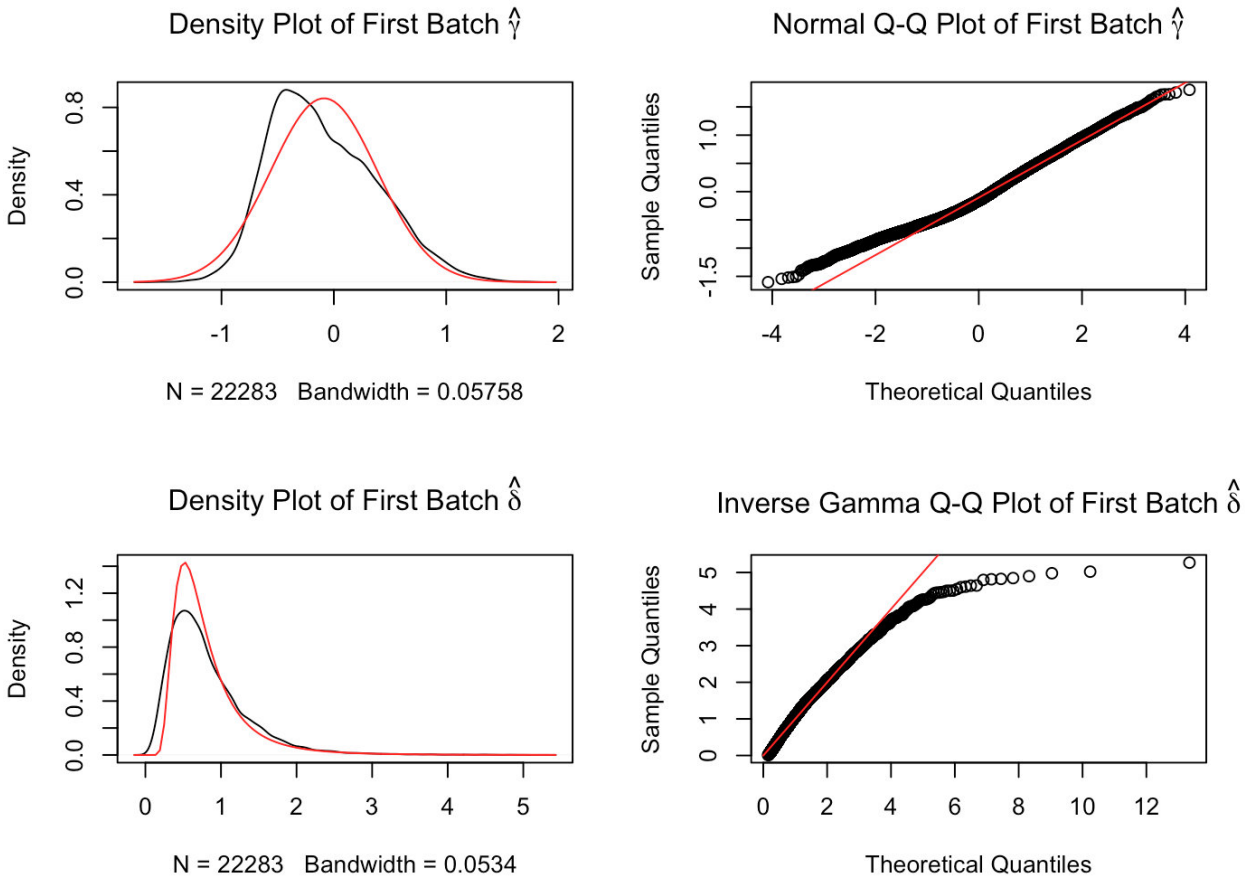
`combat_edata <- ComBat(data=edata, batch=batch, mod=modcombat, par.prior=TRUE, prior.plots=TRUE, mean.only=TRUE)`

```
> combat_edata <- ComBat(dat=edata, batch=batch,
+                        mod=modcombat,par.prior=T,
+                        prior.plots = T)
Found5batches
Adjusting for0covariate(s) or covariate level(s)
Standardizing Data across genes
Fitting L/S model and finding priors
Finding parametric adjustments
Adjusting the Data
```

返回一个和原数据集相同维度的表达矩阵, 该表达矩阵已经调整了批次效应. 同样直接执行显著性分析

`pValuesComBat <- f.pvalue(combat_edata, mod, mod0)`

```
qValuesComBat <- p.adjust(pValuesComBat, method="BH")
```



黑色表针对经验批次相应密度的核心评估, 红色为参数评估, 通过该图可以确保评估的合理性

同时这里添加了`mean.only=TRUE`参数, 仅用于不同批次间的批次效应均值(默认时调整均值和变异, mean and variance). 当实际会出现适当的批次效应时, 使用该选项`mean.only=TUER`, 或在由于生物学原因在不同的批次间缺失存在差异时使用

最后, 参数`ref.batch`允许用户选择一个批次作为其他批次调整的参考. 非参考批次的均值和变异哦度灰根据参考批次进行调整. 该参数在当一个批次比较大或质量更好时有用

~~**Removing known batch effects with a linear model**~~

通过`f.pvalue`函数直接调整批次效应. 在bladder cancer例子中, 其中一个已知的变量为批次变量. 该变量能被包含在mod和mod0模型中, 然后使用`f.pvalue`函数直接检出差异表达, 该过程为`ComBat`简化版本

```
modBatch <- model.matrix(~as.factor(cancer) + as. factor(batch), data=pheno)
```

```
mod0Batch <- model.matrix(~as.factor(batch), data=pheno)
```

```
pValuesBatch <- f.pvalue(edata, modBatch, mod0Batch)
```

```
qValuesBatch <- p.adjust(pValuesBatch, method="BH")
```

Surrogate variables versus direct adjustment

`sva`的目的在于去除所有非目的性差异来源, 保证比对是来自`mod`中的主要变量. 用于发现在不同组中一致性识别的特征, 去除所有共有的潜在差异来源.

在一些情况下, 潜在的差异可能是重要的生物学变异来源. 假如分析的目标在于识别一个或多个亚组中的杂合性, `sva`函数将不适用. 例如, 假设期待肿瘤样本代表两个不同的亚型, 但是存于未知的亚组. 假如, 这些亚组对表达影响很高, 那么其中一个或多个评估的代理变量可能和该亚组高度相关.

相反, 直接调整仅去除已知的批次变量效应. 使用该过程时所有潜在的生物学变异将保留在数据中. 换言之, 假如样本来自不同的环境, 那么该效应依然存在数据中. 假如来自环境, 实验等重要的杂合性来源没有去除, 将会增加假阳性率.

~~**Applying the fsva function to remove batch effects for prediction**~~

例如在使用microarray做种群水平的表达差异分析时, 有些情况是用于预测. 这时候, 数据集一般由一个训练集和检测集组成(a training set and a test set). 对于训练集中的每一个样本, outcome/class是已知的, 但是潜在来源的变异是未知的. 对于检测集中的样本, outcomt/class或者潜在变异都是未知的.

'Frozen' 代理变量分析可用于去除检测集中的潜在变异.

```
set.seed(12345)
```

```
trainIndicator <- sample(1:57, size=30, replace=FALSE)
```

```
testIndicator <- (1:57)[-trainIndicator]
```

```
trainData <- edata[, trainIndicator]
```

```
testData <- edata[, testIndicator]
```

```
trainPheno <- pheno[trainIndicator,]
```

```
testPheno <- pheno[testIndicator,]
```

使用`pamr`包在训练数据中训练预测模型, 在检测数据中训练检测模型

```
> mydata <- list(x=trainData,y=trainPheno$cancer)
> mytrain <- pamr.train(mydata)
1234567891011121314151617181920212223242526272829 30
> help(pamr.train)
> table(pamr.predict(mytrain, testData,threshold = 2), testPhene$cancer)

         Biopsy Cancer Normal
  Biopsy      4      1      2
  Cancer      0     15      1
  Normal      0      3      1
```

使用`sva`函数计算训练集的代理变量

```
> trainMod <- model.matrix(~cancer, data=trainPheno)
> trainMod0 <- model.matrix(~1, data=trainPheno)
> trainSv <- sva(trainData, trainMod, trainMod0)
Number of significant surrogate variables is:  6
Iteration (out of 5 ):1  2  3  4  5
```

使用`fsva`函数调整训练数据和检测数据. 训练集使用计算的代理变量调整, 检测集使用'frozen'代理变量算法调整.

```
> fsvaobj <- fsva(trainData, trainMod, trainSv, testData)
> mydataSv <- list(x=fsvaobj$db, y=trainPheno$cancer)
> mytrainSv <- pamr.train(mydataSv)
1234567891011121314151617181920212223242526272829 30
> table(pamr.predict(mytrainSv,fsvaobj$new,threshold = 1),
+       testPhene$cancer)

         Biopsy Cancer Normal
  Biopsy      4      0      0
  Cancer      0     18      1
  Normal      0      1      3
>
```

---

sva for sequencing (svaseq)

起初, 我们使用识别函数用于来自接近对称的/连续的数据. 而对测序数据而言, 是由计数表示的, 因此更恰当的模型可能要使用log函数. 因此, 第一步, 我们将使用log(g + c)来转换基因表达数据, c为一个小的正数, 这里设置为1

首先过滤掉低计数基因, 识别潜在的control基因

```
> c <- c(1,2,3,4,5,6,7,8,9)
> length(c>5)
[1] 9
> length(c[c>5])
[1] 4
```

```
library(zebrafishRNASeq)
```

```
data(zfGenes)
```

```
filter <- apply(zfGenes, 1, function(x)length(x[x>5])>=2)
```

```
fitlered <- zfGenes[filter,]
```

```
genes <- rownames(filtered)[grep["^ENS",rownames(filtered)]]
```

```
controls <- grepl("^ERCC",rownames(filtered))
```

```
group <- as.factor(rep(c("Ctl","Trt"),each =3))
```

```
dat0 <- as.matrix(filtered)
```

使用`svaseq`函数评估潜在的因子. 该例子中由于样本数目较少(n=6), 设置`n.sv=1`. 但一般可用`svaseq`去评估潜在因子数目

```
## Set null and alternative models(ignore batch)
```

```
mod1 <- model.matrix(~group)
```

```
mod0 <- cbind(mod1[,1])
```

```
svaseq <- svaseq(dat0, mod1, mod0, n.sv=1)$sv
```

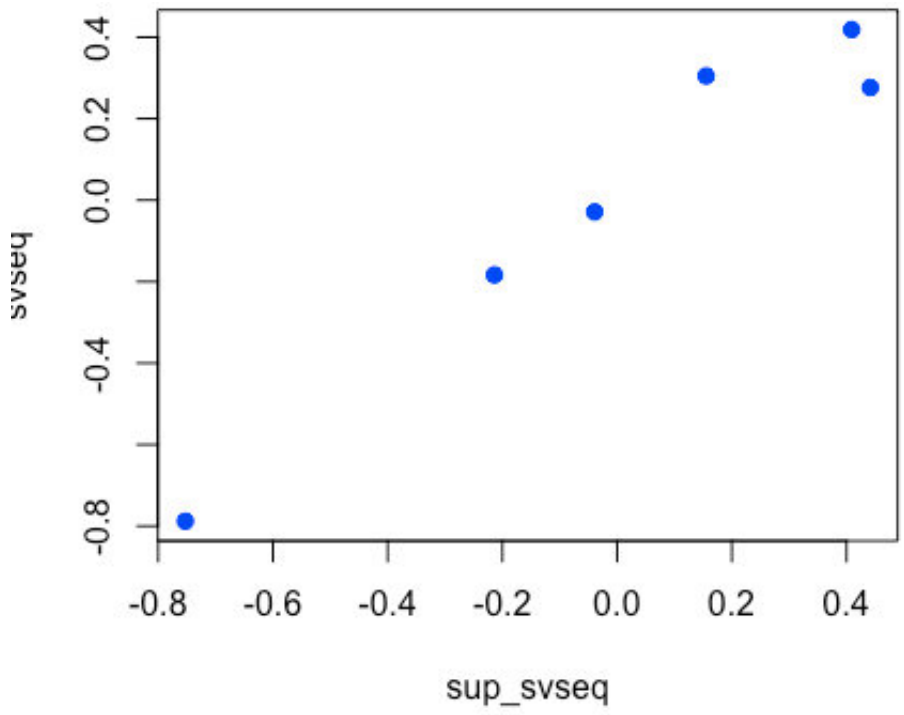
```
> head(dat0)
                  Ctl1 Ctl3 Ctl5 Trt9 Trt11 Trt13
ENSDARG00000000001  304  129  339  102    16   617
ENSDARG00000000002  605  637  406   82   230  1245
ENSDARG00000000018  391  235  217  554   451   565
ENSDARG00000000019 2979 4729 7002 7309  9395  3349
ENSDARG00000000068   89  356   41  149    45    44
ENSDARG00000000069  312  184  844  269   513   243
> group
[1] Ctl Ctl Ctl Trt Trt Trt
Levels: Ctl Trt
> svseq <- svaseq(dat0, mod1, mod0,n.sv=1)$sv
Number of significant surrogate variables is:  1
Iteration (out of 5 ):1  2  3  4  5
```

`plot(svseq, pch=19, col="blue")`

最初, 我们引入一个算法来评估仅被未知的人为因素所影响的基因. 随后, Gagnon-Bartsch和其同事使用一些技术上的或者实验上的指控探针来识别仅被人为因素所影响的基因. 因此, 监督型sva使用已知的探针来评估代理变量.

`sup_svseq <- svaseq(dat0, mod1, mod0, controls=controls,n.sv=1)$sv`

`plot(sup_svseq,svseq,pch=19,col="blue")`



这里传递了`controls`参数, 为一个0和1的向量值, 代表被批次影响而不被条件所影响的基因.

[DESeq2 & sva] [http://master.bioconductor.org/packages/release/workflows/vignettes/rnaseqGene/inst/doc/rnaseqGene.html]

RNA-seq workflow: gene-level exploratory analysis and differential expression

8. Removing hidden batch effects

假设不知道有多少细胞系用于实验, 仅知道处理组使用了地塞米松处理(dexamethasone). 因此不同细胞系所带了的计数影响为潜在的非目的性的变异, 可能影响数据集中的多个或所有基因.

```
library(sva)
```

获得标准化技术矩阵, 同时满足每行所有样本基因数目均值大于1

由于不知道潜在的细胞系信息, 因此构建一个`full model`矩阵包含地塞米松变量(dex)和一个`null model`矩阵仅包含截距项. 最后, 假设我们想要评估两个代理变量.

```
dat <- counts(dds, normalized=TRUE)
```

```
idx <- rowMeans(dat) > 1
```

```
dat <- dat[idx, ]
```

```
mod <- model.matrix(~dex, colData(dds))
```

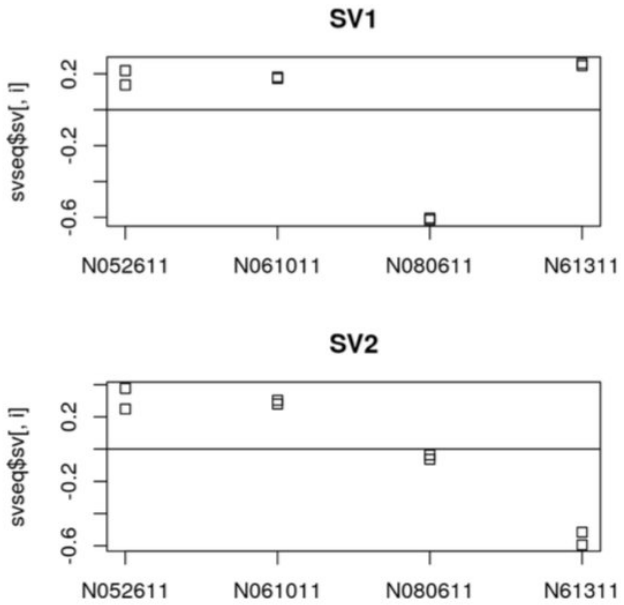
```
mod0 <- model.matrix(~1, colData(dds))
```

```
svseq <- svaseq(dat, mod, mod0, n.sv=2)
```

由于已知细胞系数目,查看SVA方法去除批次效应的程序

```
par(mfrow=c(2,1), mar=c(3,5,3,1))
```

```
for(i in 1:2){stripchart(svseq$sv[,i] ~ dds$cell, vertical=TRUE, main=paste0("SV",i)); abline(h=0)}
```



使用SVA根据代理变量去除批次相应, 向DESeqDataSet对象列添加这两个代理变量, 然后加入模型设计中(design)

```
ddssva <- dds
```

```
ddssva$SV1 <- svseq$sv[,1]
```

```
ddssva$SV2 <- svseq$sv[,2]
```

```
design(ddssva) <- ~SV1 + SV2 +dex
```

然后使用DESeq生成含有代理变量调整后的结果