

tweeDEseq: analysis of RNA-seq data using the Poisson-Tweedie family of distributions

Mikel Esnaola¹ Robert Castelo^{2,3}
mesnaola@creal.cat robert.castelo@upf.edu

Juan Ramón González^{1,2,3}
jrgonzalez@creal.cat

October 17, 2016

¹Centre for Research in Environmental Epidemiology (CREAL), Barcelona, Spain

<http://www.creal.cat/jrgonzalez/software.htm>

²Dept. of Experimental and Health Sciences, Research Program on Biomedical Informatics (GRIB),
Universitat Pompeu Fabra, Barcelona, Spain

³Hospital del Mar Research Institute (IMIM), Barcelona, Spain

1 Getting started

This document gives an overview of the R package **tweeDEseq**, which provides statistical procedures for testing differential expression in RNA-seq count data. In fact, these procedures could be applied, in principle, to any kind of count data, other than RNA-seq. The **tweeDEseq** package offers a function for normalizing count data which actually calls other functions for that purpose from the **edgeR** package. For this reason, it is necessary to have installed the **edgeR** package as well, although it is not necessary to explicitly load it onto the session. Another package necessary for running this vignette is the **tweeDEseqCountData** package which contains data to illustrate the analyses and which is employed in the article introducing **tweeDEseq** by ?. Therefore, we should start the R session with loading these libraries as follows:

```
> library(tweeDEseq)
> library(tweeDEseqCountData)
```

We will start loading into the workspace the data corresponding to the initial table of raw counts of the RNA-seq experiment from ? on lymphoblastoid cell lines derived from 69 unrelated nigerian individuals as well as a vector of gender labels for each sample matching the sample order in the table of counts:

```
> data(pickrell)
> countsNigerian <- exprs(pickrell.eset)
> dim(countsNigerian)

[1] 52580    69

> countsNigerian[1:5, 1:5]
```

	NA18486	NA18498	NA18499	NA18501	NA18502
ENSG00000000003	0	0	0	0	0
ENSG00000000005	0	0	0	0	0
ENSG00000000419	22	105	40	55	67
ENSG00000000457	22	100	107	53	72
ENSG00000000460	5	23	10	18	15

```
> genderNigerian <- pData(pickrell.eset)[,"gender"]
> head(genderNigerian)
```

```
[1] male   male   female male   female male
Levels: female male
```

```
> table(genderNigerian)
```

```
genderNigerian
female   male
    40     29
```

2 Normalization and filtering

We proceed to normalize this initial table of raw counts in order to try to remove any technical biases that might be affecting the data. The `tweedEseq` package relies for this purpose on part of the functionality provided by the `edgeR` package (comprising RNA composition adjustment by TMM (?) and quantile-to-quantile count adjustment (?) produced by the `equalizeLibSizes()` from `edgeR`) and offers one function (`normalizeCounts()`) that makes the appropriate calls to `edgeR` to normalize these data.

```
> countsNigerianNorm <- normalizeCounts(countsNigerian, genderNigerian)
```

```
> dim(countsNigerianNorm)
```

```
[1] 10231    69
```

If more control is needed in this step, the user should directly employ the corresponding `edgeR` functions. Next, we can filter out genes with very low expression using the function `filterCounts()` whose default parameters remove those genes with less than 5 counts per million in all samples but one.

```
> countsNigerianNorm <- filterCounts(countsNigerianNorm)
```

```
> dim(countsNigerianNorm)
```

```
[1] 10231    69
```

3 The Poisson-Tweedie family of distributions to model RNA-seq count data

The package `tweedEseq` uses the Poisson-Tweedie (PT) family of distributions as the statistical model for count data. PT distributions have been studied by several authors(????) and unify several count data distributions (see **Fig. 1** in El-Shaarawi *et al.*, 2011) such as Poisson, negative

binomial, Poisson-Inverse Gaussian, Pólya-Aeppli or Neyman type A. These distributions can model different scenarios as, for instance, a RNA-seq expression profile with a wide dynamic range leading to a heavy tail in the distribution.

Following ?, let $Y \sim \text{PT}(a, b, c)$ be a PT random variable with vector of parameters $\theta = (a, b, c)^T$ defined in the domain

$$\Theta = (-\infty, 1] \times (0, +\infty) \times [0, 1]. \quad (1)$$

For the sake of interpretability, we reparametrize $\theta = (a, b, c)$ to $\theta = (\mu, \phi, a)$, where μ denotes the mean, $\phi = \sigma^2/\mu$ is the dispersion index (σ^2 is the variance), and a the shape parameter that is employed to define some count data distributions that are particular cases of PT such as Poisson or Negative Binomial. The relationship between both parameterizations is the following:

$$c = \frac{\phi - 1}{\phi - a}, \quad b = \frac{\mu(1 - a)^{(1-a)}}{(\phi - 1)(d - a)^{-a}}. \quad (2)$$

Under this parametrization, the shape parameter determines the specific count data distribution being employed. For instance $a = 1$ corresponds to Poisson and $a = 0$ corresponds to negative binomial. We can estimate the parameter vector θ by maximum likelihood through the function `mlePoissonTweedie()` as follows:

```
> set.seed(123)
> y <- rnbinom(1000, mu=8, size=1/0.2)
> thetahat <- mlePoissonTweedie(y)
> getParam(thetahat)

      mu      D      a
8.0680000 2.6620193 0.0405573
```

where here we have simulated 1000 random observations from a negative binomial distribution and the last call to `getParam()` allows us to extract the $\hat{\theta}$ vector from the object return by `mlePoissonTweedie()`.

4 Goodness of fit to a count data distribution

The PT distribution allows one to test for the goodness of fit to a particular count data distribution defined by a specific value of the PT shape parameter a . For this purpose, the function `testShapePT()` allows us to test the goodness of fit to, for instance, the widely used negative binomial distribution (i.e., $H_0 : a = 0$) as illustrated here with the previously estimated vector $\hat{\theta}$:

```
> testShapePT(thetahat, a=0)

$statistic
[1] 0.01471032

$pvalue
[1] 0.9034644
```

These functions are called from another one called `gofTest()` which can perform for us a goodness-of-fit for every gene in the rows of a given matrix of counts, and will return the corresponding χ^2_1 statistics. Since calculating this for the entire gene set would take too long for building quickly

this vignette we are going to work on a subset of genes formed by human genes with documented sex-specific expression, a sampled subset of 25 human housekeeping genes and the secretin (*SCT*) gene which encodes for an endocrine hormone peptide in chromosome 11 that controls secretions in the duodenum. The gender-related and housekeeping gene lists form part from the previously loaded experimental data package `tweeDEseqCountData` and can be loaded as follows:

```
> data(genderGenes)
> data(hkGenes)
> length(XiEgenes)

[1] 63

> length(msYgenes)

[1] 32

> length(hkGenes)

[1] 669
```

The list of genes with documented sex-specific expression was built by first selecting genes in chromosome X that escape X-inactivation (?) and genes in the male-specific region of the Y chromosome (?), and then filtering out those that do not occur in the initial table of counts with 52580 Ensembl genes. The list of housekeeping genes was retrieved from the literature(?) and then also filtered to keep only those genes that form part of the initial table of counts. The selection is finally done as follows:

```
> set.seed(123)
> someHKgenes <- sample(hkGenes, size=25)
> geneSubset <- unique(c("ENSG00000070031",
+                         intersect(rownames(countsNigerianNorm),
+                         c(someHKgenes, msYgenes, XiEgenes))))
> length(geneSubset)

[1] 43
```

Now, we calculate the goodness of fit to a negative binomial distribution for each of these 43 genes using the function `gofTest()`:

```
> chi2gof <- gofTest(countsNigerianNorm[geneSubset, ], a=0)
```

and we can examine the result by means of the quantile-quantile plots produced with the function `qqchisq()` and shown in Figure ??, which indicates that more than a 50% of the genes show a substantial discrepancy with the respect to the negative binomial distribution.

```
> par(mfrow=c(1,2), mar=c(4, 5, 3, 4))
> qq <- qqchisq(chi2gof, main="Chi2 Q-Q Plot", ylim = c(0, 60))
> qq <- qqchisq(chi2gof, normal=TRUE, ylim = c(-5, 7))
```

This indicates that different genes may require different count data distributions but, in fact, this can be also observed for different sample groups within the same gene. Figure ?? illustrates such a case with the secretin (*SCT*) gene (Ensembl ID ENSG00000070031) when looking separately to male and female samples. This figure is produced with the following code that calls the function `compareCountDist()` which helps in comparing an empirical distribution of counts against the Poisson, the negative binomial and the corresponding estimated PT distribution.

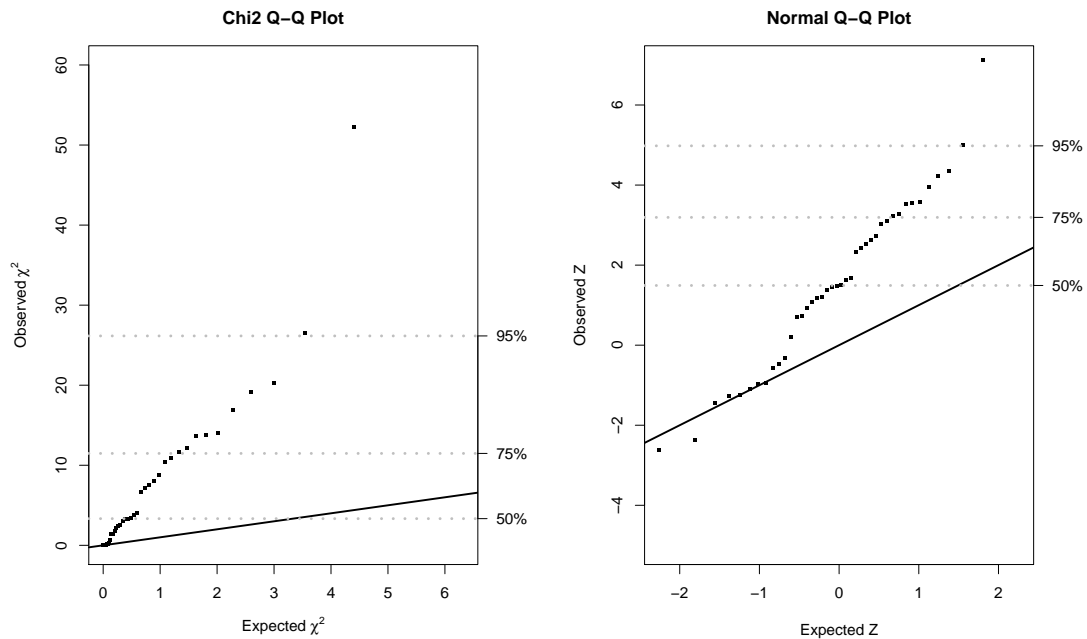


Figure 1. Goodness of fit to a binomial distribution. On the left a quantile-quantile plot of the χ^2 statistic employed to assess the goodness-of-fit of the RNA-seq data to a negative binomial distribution is shown. More than a 50% of the genes have expression profiles that depart substantially from the negative binomial distribution. On the right we have the same data but χ^2 statistics are transformed into standard normal z-scores to improve visibility of the lower quantiles.

```
> par(mfrow=c(1,2), mar=c(4, 5, 3, 2))
> xf <- unlist(countsNigerianNorm["ENSG00000070031", genderNigerian=="female"])
> compareCountDist(xf, main="Female samples")
> xm <- unlist(countsNigerianNorm["ENSG00000070031", genderNigerian=="male"])
> compareCountDist(xm, main="Male samples")
```

What Figure ?? reflects can be also easily seen by just looking to the actual counts and identifying the female sample that produces the heavy tail on the distribution

```
> sort(xf)
```

NA19137	NA18855	NA19127	NA19159	NA19225	NA19257	NA18508	NA18511	NA18858	NA18912
1	2	2	2	2	2	3	3	3	3
NA19102	NA19114	NA19116	NA18505	NA18909	NA19099	NA19140	NA19222	NA18517	NA18520
3	3	3	4	4	4	4	4	5	5
NA19131	NA19143	NA18523	NA18861	NA18870	NA19108	NA19152	NA19201	NA18499	NA19093
5	5	6	6	6	6	6	6	8	9
NA19147	NA18852	NA19209	NA18502	NA19238	NA19190	NA19204	NA18916	NA19172	NA19193
9	11	11	12	12	16	17	21	37	196

```
> sort(xm)
```



Figure 2. Empirical cumulative distribution function (CDF) of counts (black dots), calculated separately from male and female samples, for the secretin gene (*SCT*) and estimated CDF (solid lines) of Poisson-Tweedie distributions with shape parameter fixed to $a = 1$ corresponding to a Poisson (green), $a = 0$ corresponding to a negative binomial (blue) and with the value of a estimated from data too (red). The legend shows the values of the a parameter and the P value of the likelihood ratio test on whether the expression profile follows a negative binomial distribution ($H_0 : a = 0$). We can observe that for both, male and female samples, the Poisson distribution is not adequate and that the negative binomial distribution is not adequate for female samples.

NA18507	NA19128	NA19138	NA19160	NA19210	NA18510	NA19192	NA18498	NA18519	NA18522
1	1	1	1	1	2	2	3	3	3
NA18856	NA18862	NA18871	NA19239	NA18486	NA18853	NA18913	NA19153	NA19171	NA19098
3	3	3	3	4	4	4	4	4	6
NA18516	NA19101	NA18501	NA19130	NA18504	NA19200	NA19203	NA19119	NA19144	
7	7	8	8	9	11	15	18	30	

and realize that by just removing that sample, the large overexpression in females just vanishes:

```
> xf[which.max(xf)]
NA19193
196
> 2^{log2(mean(xf))-log2(mean(xm))}
[1] 2.003402
> 2^{log2(mean(xf[-which.max(xf)])) - log2(mean(xm))}
```

```
[1] 1.192384
```

This illustrates a case in which Poisson and negative binomial distributions may be too restrictive to account for the biological variability that extensively-replicated RNA-seq experiments can reveal in count data.

5 Testing for differential expression

In order to illustrate the accuracy of `tweeDEseq` for detecting DE genes in a extensively-replicated RNA-seq experiment we have compared the expression profiles between males and females from the population of 69 unrelated Nigerian individuals(?).

The `tweeDEseq` package contains a function to test for differential expression between two different conditions using a score based test: the `tweeDE()` function. This function takes as input a matrix of counts with genes on the rows and samples on the columns.

An important feature of the `tweeDE()` function is that it allows to use multiple processors in the computing process. This is done by loading first the `multicore` package and specifying the number of cores to be used with the `mc.cores` argument in the call to `tweeDE()`.

```
> resPT <- tweeDE(countsNigerianNorm[geneSubset, ], group = genderNigerian)
```

The function `tweeDE()` returns a *data.frame* object of class *tweeDE* which can be examined with the `print()`:

```
> print(resPT)
```

Comparison of groups: male - female
Showing top 6 genes ranked by P-value
Minimum absolute log2 fold-change of 0
Maximum adjusted P-value of 1

	overallMean	female	male	log2fc	stat	pval
ENSG00000129824	186.942	3.275	440.276	7.070769	16.118299	1.897624e-58
ENSG00000154620	16.043	2.900	34.172	3.558707	11.319513	1.050535e-29
ENSG00000099749	31.464	3.550	69.966	4.300753	10.664342	1.494538e-26
ENSG00000198692	18.551	3.775	38.931	3.366372	9.717809	2.531693e-22
ENSG00000157828	7.507	1.875	15.276	3.026291	7.968633	1.604387e-15
ENSG00000006757	152.101	196.175	91.310	-1.103291	6.820705	9.059494e-12
	pval.adjust					
ENSG00000129824	8.159783e-57					
ENSG00000154620	2.258651e-28					
ENSG00000099749	2.142171e-25					
ENSG00000198692	2.721570e-21					
ENSG00000157828	1.379773e-14					
ENSG00000006757	6.492637e-11					

which will show us by default the top 6 genes ranked by *P*-value including information on the magnitude of the fold-change in log2 scale (`log2fc`), overall mean expression in counts (`overallMean`), mean expression in counts for each sample group, raw *P*-value (`pval`) and the Benjamini-Hochberg (FDR) adjusted *P*-value (`pval.adjust`).

The same `print()` function allows us to call differentially expressed a subset of gene meeting cutoffs on the minimum magnitude of the fold-change and maximum FDR and store that subset in a *data.frame* object by using the appropriate arguments as follows:

```
> deGenes <- print(resPT, n=Inf, log2fc=log2(1.5), pval.adjust=0.05, print=FALSE)
> dim(deGenes)
```

```
[1] 9 7
```

We can further enrich the output with information like the symbol and description of the gene by using the annotation information stored as a *data.frame* in the experimental data package *tweeDEseqCountData* as follows:

```
> data(annotEnsembl63)
> head(annotEnsembl63)
```

	Symbol	Chr	Start	End	EntrezID
ENSG00000252775	U7	5	133913821	133913880	<NA>
ENSG00000207459	U6	5	133970529	133970635	<NA>
ENSG00000252899	U7	5	133997420	133997479	<NA>
ENSG00000201298	U6	5	134036862	134036968	<NA>
ENSG00000222266	U6	5	134051173	134051272	<NA>
ENSG00000222924	U6	5	137405044	137405147	<NA>

	Description	Length
ENSG00000252775	U7 small nuclear RNA	[Source:RFAM;Acc:RF00066] NA
ENSG00000207459	U6 spliceosomal RNA	[Source:RFAM;Acc:RF00026] NA
ENSG00000252899	U7 small nuclear RNA	[Source:RFAM;Acc:RF00066] NA
ENSG00000201298	U6 spliceosomal RNA	[Source:RFAM;Acc:RF00026] NA
ENSG00000222266	U6 spliceosomal RNA	[Source:RFAM;Acc:RF00026] NA
ENSG00000222924	U6 spliceosomal RNA	[Source:RFAM;Acc:RF00026] NA

	GCcontent
ENSG00000252775	NA
ENSG00000207459	NA
ENSG00000252899	NA
ENSG00000201298	NA
ENSG00000222266	NA
ENSG00000222924	NA

```
> deGenes <- merge(deGenes, annotEnsembl63, by="row.names", sort=FALSE)
```

and select certain columns to build Table ?? using the *xtable* package (code not shown but available in the source of the vignette).

6 Visualizing the results

An informative way to visualize the results of a differential expression analysis is by means of MA and Volcano plots, which we can easily obtain through the *tweeDEseq* package functions *MAPlot()* and *Vpplot()*, respectively as follows. Their result is shown in Figure ??.

```
> deGenes <- rownames(print(resPT, n=Inf, log2fc=log2(1.5), pval.adjust=0.05, print=FALSE))
> length(deGenes)
```

```
[1] 9
```


Table 1. Differentially expressed genes between female and male Nigerian individuals found by *tweedEseq*.

Symbol	Chr	log2fc	pval.adjust	Description
RPS4Y1	Y	7.07	8.16E-57	ribosomal protein S4, Y-linked 1
TMSB4Y	Y	3.56	2.26E-28	thymosin beta 4, Y-linked
CYorf15A	Y	4.30	2.14E-25	chromosome Y open reading frame 15A
EIF1AY	Y	3.37	2.72E-21	eukaryotic translation initiation factor 1A, Y-linked
RPS4Y2	Y	3.03	1.38E-14	ribosomal protein S4, Y-linked 2
PNPLA4	X	-1.10	6.49E-11	patatin-like phospholipase domain containing 4
STS	X	-0.99	1.21E-04	steroid sulfatase (microsomal), isozyme S
HDHD1	X	-0.77	4.56E-04	haloacid dehalogenase-like hydrolase domain containing 1
UTY	Y	0.63	4.46E-02	ubiquitously transcribed tetratricopeptide repeat gene, Y-linked

```
> hl <- list(list(genes=msYgenes, pch=21, col="blue", bg="blue", cex=0.7),
+           list(genes=XiEgenes, pch=21, col="skyblue", bg="skyblue", cex=0.7),
+           list(genes=deGenes, pch=1, col="red", lwd=2, cex=1.5)
+           )
> par(mfrow=c(1,2), mar=c(4, 5, 3, 2))
> MAplot(resPT, cex=0.7, log2fc.cutoff=log2(1.5), highlight=hl, main="MA-plot")
> Vplot(resPT, cex=0.7, highlight=hl, log2fc.cutoff=log2(1.5),
+       pval.adjust.cutoff=0.05, main="Volcano plot")
```

7 Assessing differential expression calling accuracy

Finally, the accuracy of the differential expression analysis illustrated here can be assessed by comparing our list of differentially expressed genes with the list of genes with documented sex-specific expression by means of a Fisher's exact test.

```
> genderGenes <- c(msYgenes[msYgenes %in% rownames(resPT)],
+                 XiEgenes[XiEgenes %in% rownames(resPT)])
> N <- nrow(resPT)
> m <- length(genderGenes)
> n <- length(deGenes)
> k <- length(intersect(deGenes, genderGenes))
> t <- array(c(k,n-k,m-k,N+k-n-m), dim=c(2,2), dimnames=list(SEX=c("in", "out"), DE=c("yes", "no")))
> t

      DE
SEX   yes no
in    9  18
out   0  16

> fisher.test(t, alternative="greater")
```

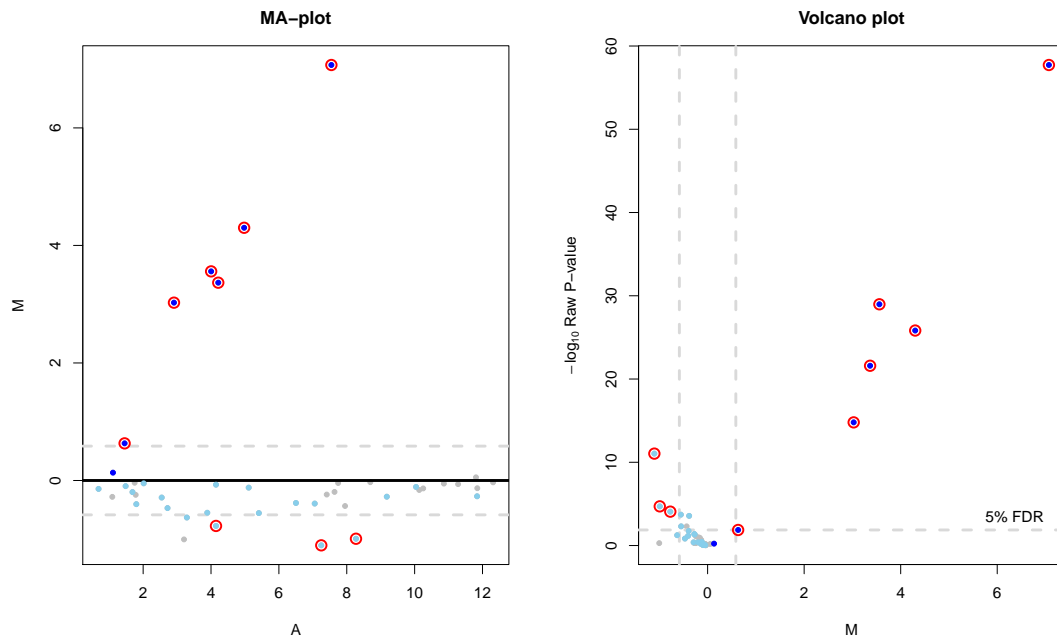


Figure 3. Differential expression analysis for a subset of genes between male and female lymphoblastoid cell lines. On the left a MA-plot shows the magnitude of the fold-change of every gene as function of its average normalized expression level. No expression-level dependent biases can be observed in the data. On the right a volcano plot shows the raw P value of every gene for the null hypothesis of no differential expression, calculated by tweedEseq, as function of its fold-change. In both plots, red circles indicate differentially expressed genes defined by the cutoffs depicted with horizontal and vertical dashed lines. Light blue dots denote genes from the male-specific region (?) of chromosome Y (MSY) and dark blue dots denote genes from Xi that escape chromosome inactivation (XiE) in female samples (?).

Fisher's Exact Test for Count Data

```
data: t
p-value = 0.008311
alternative hypothesis: true odds ratio is greater than 1
95 percent confidence interval:
 1.884678      Inf
sample estimates:
odds ratio
      Inf
```

8 Fitting generalized linear models (GLM)

The tweedEseq package also allows to fit generalized linear models for a response variable following the Poisson-Tweedie family of distributions and several covariates. This can be done using

the `glmPT()` function. For instance, we can fit a model taking the secretin (*SCT*) gene as response and gender as covariate:

```
> mod <- glmPT(countsNigerianNorm["ENSG00000070031",] ~ genderNigerian)
> mod

Call:  glmPT(formula = countsNigerianNorm["ENSG00000070031", ] ~ genderNigerian)
```

Coefficients:

(Intercept)	genderNigerianmale
2.2544	-0.2412

Poisson-Tweedie parameters:

c	a
0.997	0.8253

```
> summary(mod)
```

Call:

```
glmPT(formula = countsNigerianNorm["ENSG00000070031", ] ~ genderNigerian)
```

Coefficients:

	Estimate	Std.Error	t value	Pr(> t)
(Intercept)	2.2544	0.1918	11.7553	< 2e-16 ***
genderNigerianmale	-0.2412	0.1622	-1.4874	0.93154

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Poisson-Tweedie parameters

	Estimate	Std.Error
c	0.997	0.0019
a	0.8253	0.0427

The resulting model can also be used to test differential expression. This can be done using the `anova()` method, which tests whether the model is significantly better than the null one.

```
> anova(mod)
```

```
[1] 0.1435284
```

The `tweeDEglm()` allows testing several genes at the same time. This function also allows using multiple processors in the computing process. Following with the example in section 5, we apply it to the same subset of genes and use the gender as covariate:

```
> resPTglm <- tweeDEglm( ~ genderNigerian, counts = countsNigerianNorm[geneSubset,])
```

`tweeDEglm()` returns a `data.frame` with the *AIC* (Akaike Information Criterion) for the fitted and null models as well as the original and adjusted p-values resulting from the test between both models. In order to visualize the top significant genes we run the following command.

```
> head(resPTglm[sort(resPTglm$pval.adjust, index.return = TRUE)$ix,])
```

	AICfull	AICnull	pval	pval.adjust
ENSG00000129824	553.0737	759.2850	3.373761e-47	1.450717e-45
ENSG00000198692	412.7887	537.5054	2.142812e-29	4.607046e-28
ENSG00000154620	402.8749	518.1261	2.529103e-27	3.625047e-26
ENSG00000099749	481.4012	591.2299	3.895582e-26	4.187751e-25
ENSG00000157828	358.1338	421.8267	5.269581e-16	4.531839e-15
ENSG00000006757	764.8961	801.0417	6.565947e-10	4.705595e-09

If we compare these results with those obtained by the `tweeDE()` function we observe that both methods place the same genes at the top of the most significant list. This is not surprising as, though the statistical tests are not identical, the underlying distributional assumptions are the same. In fact, `tweeDEglm()` detects all the genes captured by `tweeDE()`.

8.1 Incorporating CQN offsets

Package `cqn` (?, available at Bioconductor) performs conditional quantile normalization in order to remove possibly existing bias arising from differences in GC content or gene lengths. The method returns a series of offsets which can be incorporated into `tweeDEseq` via the `tweeDEglm` or `glmPT` function.

For instance, suppose the result of the `cqn` normalization is stored in an object called `cqn.subset`¹. The normalizing offsets are stored as a matrix at `cqn.subset$offset`. They can be incorporated into the model using the 'offset' argument.

```
> tweeDEglm(~ genderNigerian, counts = countsNigerianNorm[geneSubset,],
+ offset = cqn.subset$offset)
```

9 Session info

```
> sessionInfo()
```

```
R version 3.3.1 (2016-06-21)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)
```

```
locale:
```

```
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
attached base packages:
```

```
[1] parallel stats graphics grDevices utils datasets methods
[8] base
```

```
other attached packages:
```

```
[1] xtable_1.8-2 tweeDEseqCountData_1.11.0
[3] Biobase_2.34.0 BiocGenerics_0.20.0
[5] tweeDEseq_1.20.0
```

```
loaded via a namespace (and not attached):
```

¹For more information about how to normalize RNA-seq count data using the `cqn` package, please refer to the package vignette available at <http://www.bioconductor.org/packages/release/bioc/html/cqn.html>

```
[1] edgeR_3.16.0    MASS_7.3-45     cqn_1.20.0      limma_3.30.0
[5] tools_3.3.1     splines_3.3.1   grid_3.3.1      locfit_1.5-9.1
[9] nor1mix_1.2-2   lattice_0.20-34
```