

# Dropout-based Feature Selection with M3Drop and NBumi

Tallulah Andrews

January 9, 2026

## Introduction

Feature selection is a key step in the analysis of single-cell RNASeq data. Feature selection aims to identify and remove genes whose expression is dominated by technical noise, thus reducing the effect of batch-effects and other technical confounders while also reducing the curse of dimensionality.

A common way to select features is to identify genes that follow a different expression distribution than a set of control genes. These control genes may be spike-ins but more commonly one assumes that the majority of genes don't differ from control genes and use all genes to approximate the expression pattern of expression of controls.

Often the variance relative to the mean expression is used to characterize the expression distribution of a gene, and feature selection proceeds by identifying highly variable genes. In contrast, this package provides two methods which characterize the expression distribution of a gene using the dropout-rate and mean expression. This takes advantage of the fact that at least 50% of entries in a single-cell RNASeq expression matrix are dropouts, and that dropout rate is less sensitive to sampling errors than variance.

## M3Drop

Single-cell RNA sequencing is able to quantify the whole transcriptome from the small amount of RNA present in individual cells. However, a consequence of reverse-transcribing and amplifying small quantities of RNA is a large number of dropouts, genes with zero expression in particular cells. The frequency of dropout events is strongly non-linearly related to the measured expression levels of the respective genes. M3Drop posits that these dropouts are due to failures of reverse transcription, a simple enzyme reaction, thus should be modelled using the Michaelis-Menten equation as follows:

$$P_i = 1 - \frac{S_i}{S_i + K}$$

Where  $P_i$  is the proportion of cells where gene  $i$  dropouts out,  $S_i$  is the mean expression of gene  $i$  and  $K$  is the Michaelis constant. This model fits observed single-cell RNASeq data which have been sequenced to near saturation (i.e. >100,000 reads per cell).

Note: This model works well for Smartseq2 and other single-cell datasets that do not use unique molecular identifiers (UMIs). For 10X Chromium data or other UMI tagged data see the NBumi model below.

We'll be using a portion of the Deng et al. (2014) dataset in this example. You can download the R-package containing this data (M3DExampleData) from Bioconductor using biocLite().

```
> library(M3Drop)
> library(M3DExampleData)
```

## Prepare Data

The first step is to perform quality control on cells. Any existing tools can be used for this. Here we will simply remove cells with fewer than 2000 detected genes.

```
> counts <- Mmus_example_list$data
> labels <- Mmus_example_list$labels
> total_features <- colSums(counts >= 0)
> counts <- counts[,total_features >= 2000]
> labels <- labels[total_features >= 2000]
```

The second is to extract and correctly format data for M3Drop. To support quality control tools in various other single-cell R packages, we have provided a function to extract appropriate data from objects used by scater, monocle, and Seurat. This function also removes undetected genes, and if necessary applies a counts-per-million normalization to raw count data.

```
> norm <- M3DropConvertData(counts, is.counts=TRUE)
```

```
[1] "Removing 18 undetected genes."
```

M3Drop requires a normalized but not log-transformed expression matrix, thus the above function can optionally de-log a log2-normalized expression matrix from any normalization method.

```
> norm <- M3DropConvertData(log2(norm+1), is.log=TRUE, pseudocount=1)
```

```
[1] "Removing 0 undetected genes."
```

Any normalization method which preserves dropouts (i.e zeros) is compatible with M3Drop.

## Feature Selection

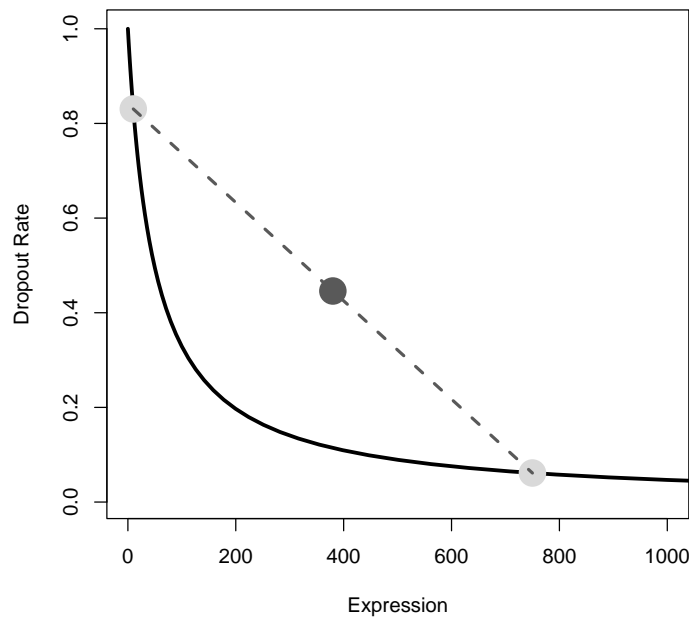
Since the Michaelis-Menten equation is convex, averaging across a mixed population forces differentially expressed genes to be shifted to the right of the Michaelis-Menten curve (Figure 1).

```
> K <- 49
> S_sim <- 10^seq(from=-3, to=4, by=0.05)
> MM <- 1-S_sim/(K+S_sim)
> plot(S_sim, MM, type="l", lwd=3, xlab="Expression", ylab="Dropout Rate",
```

```

+       xlim=c(1,1000))
> S1 <- 10; P1 <- 1-S1/(K+S1);
> S2 <- 750; P2 <- 1-S2/(K+S2);
> points(c(S1,S2), c(P1,P2), pch=16, col="grey85", cex=3)
> lines(c(S1, S2), c(P1,P2), lwd=2.5, lty=2, col="grey35")
> mix <- 0.5;
> points(S1*mix+S2*(1-mix), P1*mix+P2*(1-mix), pch=16, col="grey35", cex=3)

```



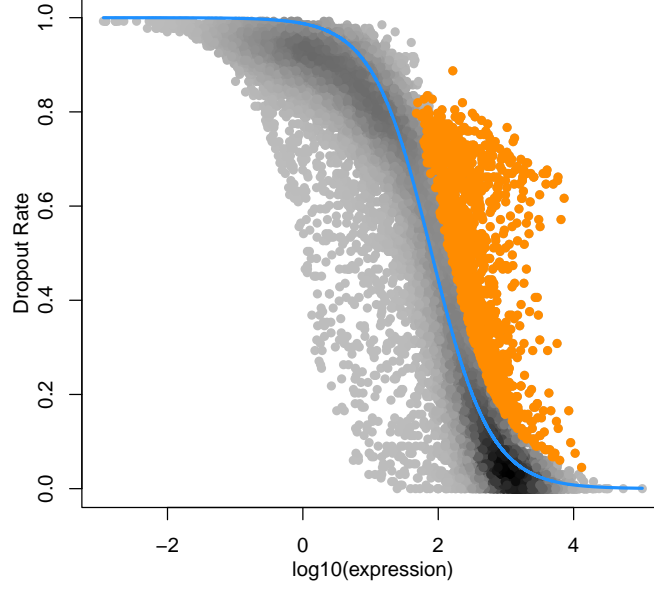
**Figure 1:** Michaelis-Menten is convex which leads to DE genes being outliers to the right/above the curve.

Feature selection for DE genes are identified by comparing the local  $K$  calculated for a specific gene to the global  $K$  fitted across all genes using a Z-test followed by multiple-testing correction. Here we find 1,248 DE genes at 1% FDR.

```

> M3Drop_genes <- M3DropFeatureSelection(norm, mt_method="fdr", mt_threshold=0.01)

```



**Figure 2:** Differentially expressed genes at 1% FDR (purple).

## NBumi

The Michaelis-Menten equation fits full-transcript single-cell RNASeq data well, but often struggles to fit data tagged with unique molecular identifiers (UMIs). This is a result of UMI datasets typically not being sequenced to saturation, thus many dropouts are a result of low sequencing coverage rather than a failure of reverse transcription.

To account for zeros resulting from insufficient sequencing depth, the M3Drop package includes a depth-adjusted negative binomial model (DANB). DANB models each observation as a negative binomial distribution with mean proportional to the mean expression of the respective gene and the relative sequencing depth of the respective cell. The dispersion parameter of the negative binomial is fit to the variance of each gene. The equations for calculating these parameters are:

Observation specific means:

$$\mu_{ij} = \frac{\sum_i t_{ij} * \sum_j t_{ij}}{\sum_{ij} t_{ij}}$$

Gene specific dispersion (solved for  $r_i$ ):

$$\sum_j (t_{ij} - \mu_{ij})^2 = \sum_j (\mu_{ij} + r_i \mu_{ij}^2)$$

Where  $\mu_{ij}$  is the observation specific mean for the negative binomial for the observed molecule counts,  $t_{ij}$  of the  $i$ th gene in the  $j$ th of cell and  $r_i$  is the dispersion parameter (the "size" parameter of R's `nbinom` functions) for gene  $i$ .

Functions relating to the DANB model are tagged with the "NBumi" prefix. We will continue using the example data despite it not using UMIs for demonstration purposes. Similar to M3Drop we have provided a function to extract the relevant expression matrix from objects used by popular single-cell RNASeq analysis packages : `scater`, `monocle`, and `Seurat`. This is a separate function because DANB requires raw counts as opposed to the normalized expression matrix used by M3Drop.

```
> count_mat <- NBumiConvertData(Mmus_example_list$data, is.counts=TRUE)

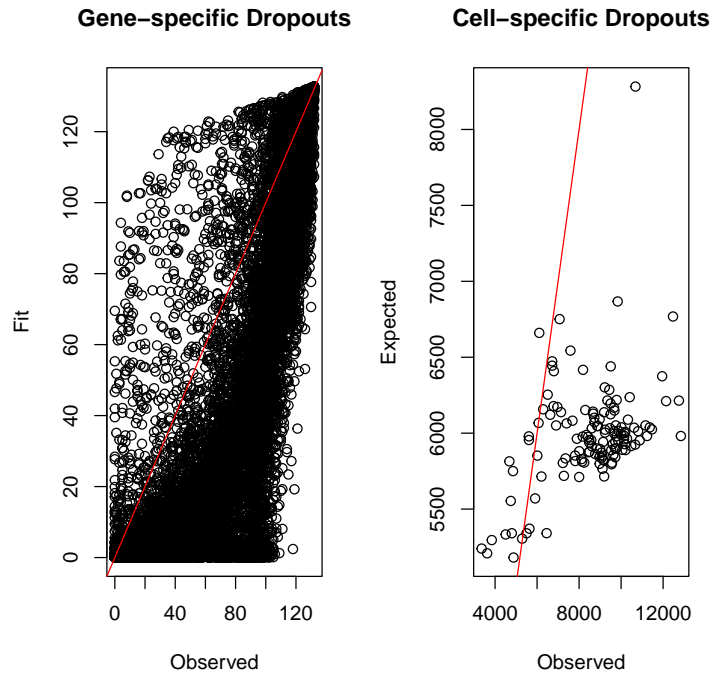
[1] "Removing 18 undetected genes."
```

This function can also de-log and de-normalize a provided expression matrix, but raw counts are preferable if they are available.

Next we fit the DANB model to this count matrix, and check it fits the data:

```
> DANB_fit <- NBumiFitModel(count_mat)
> # Smoothed gene-specific variances
> par(mfrow=c(1,2))
> stats <- NBumiCheckFitFS(count_mat, DANB_fit)
> print(c(stats$gene_error,stats$cell_error))

[1] 16981269 1313367011
```

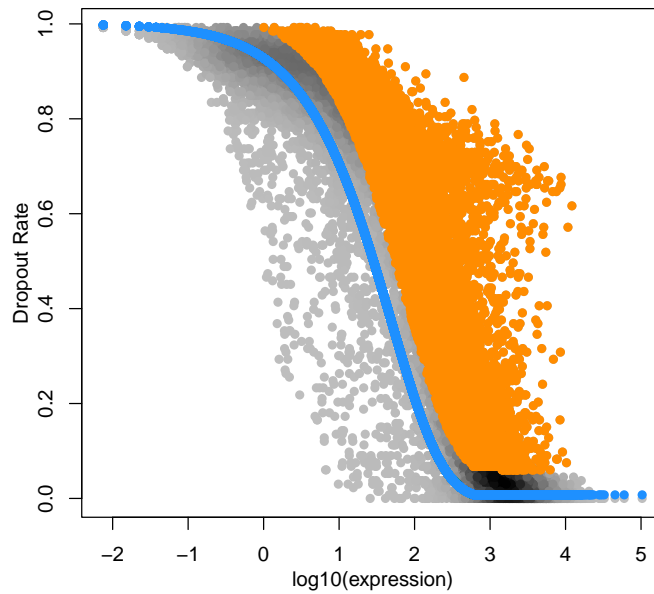


**Figure 3:** Fits of the depth-adjusted negative binomial.

Since this is a full-transcript dataset that we have converted from normalized values the model doesn't fit well as can be seen above. We will continue with this data for demonstration purposes only.

We use a binomial test to evaluate the significance of features under the DANB model:

```
> NBDropFS <- NBumiFeatureSelectionCombinedDrop(DANB_fit, method="fdr", qval.thres=0.01, s
```



**Figure 4:**  
Dropout-based feature selection using DANB

## Pearson Residuals

Pearson residuals have recently been proposed as an alternative normalization approach for UMI tagged single-cell RNAseq data. (<https://genomebiology.biomedcentral.com/articles/10.1186/s13059-021-02451-7>)

We have added two option for calculating Pearson residuals using the DANB model presented here:

```
> pearson1 <- NBumiPearsonResiduals(count_mat, DANB_fit)
> pearson2 <- NBumiPearsonResidualsApprox(count_mat, DANB_fit)
```

If you have not fit the DANB model to the data, you can run each of these functions directly on the count matrix alone, and the required model fitting will be performed as part of the function. In this case the NBumiPearsonResidualsApprox function will be much quicker since it does not fit the dispersion parameter of the DANB model, only the mean, and approximates the Pearson residuals as if the data was Poisson distributed:

$$p_{approx} = \frac{x_{ij} - mu_{ij}}{\sqrt{mu_{ij}}}$$

Where  $x_{ij}$  is the observed umi counts.

Whereas the NBumiPearsonResiduals function calculates the Pearson residuals using the full DANB model as:

$$p_{exact} = \frac{x_{ij} - mu_{ij}}{\sqrt{mu_{ij} + \frac{mu_{ij}^2}{r_i}}}$$

We do not perform any trimming or clipping of the residuals. To perform  $\sqrt{n}$  clipping as per Hafemeister and Satija, simply run the following additional steps:

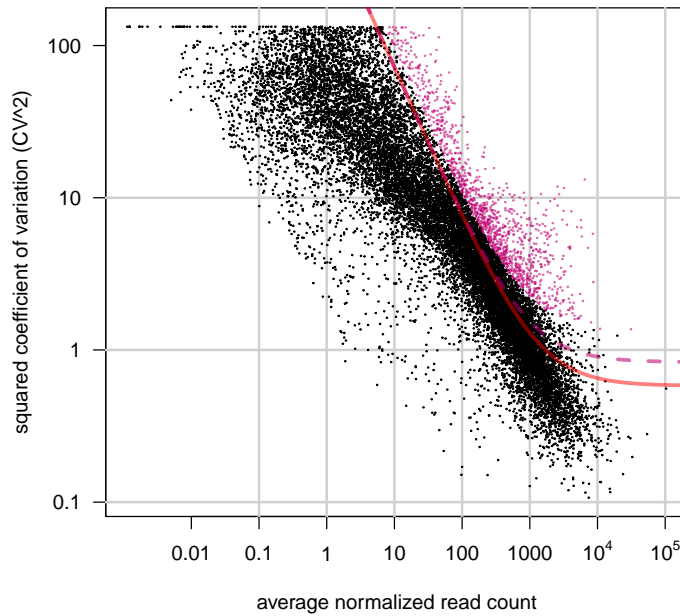
```
> pearson1[pearson1 > sqrt(ncol(count_mat))] <- sqrt(ncol(count_mat))
> pearson1[pearson1 < -sqrt(ncol(count_mat))] <- -1*sqrt(ncol(count_mat))
```

## Other Feature Selection Methods

For comparison purposes we have provided functions for other feature selection methods, including: BrenneckeGetVariableGenes from Brennecke et al. (2013) modified to optionally use all genes to fit the function between CV2 and mean expression. giniFS based on GiniClust (REF) pcaFS based on monocle (REF) using irlba and sparse-matrices. corFS which uses gene-gene correlations. and ConsensusFS which uses all the available feature selection methods and takes the average rank of genes across all methods.

Only the Brennecke highly variable gene method also provides a significance test and runs in linear time, thus is the only one we will demonstrate here.

```
> HVG <- BrenneckeGetVariableGenes(norm)
```



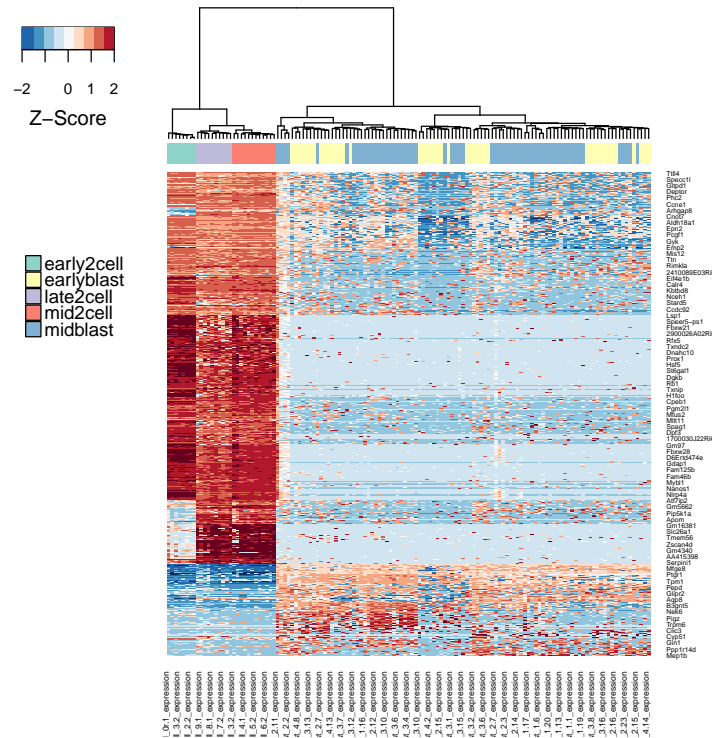
**Figure 5:**

Brennecke highly variable genes.

## Examining Features and Identifying Subpopulations of Cells

To check that the identified genes are truly differentially expressed we can plot the normalised expression of the genes across cells.

```
> heat_out <- M3DropExpressionHeatmap(M3Drop_genes$Gene, norm,
+                                     cell_labels = labels)
```



**Figure 6:** Heatmap of expression of M3Drop features.

The heatmap (Figure 6) shows that the genes identified by M3Drop are differentially expressed across timepoints. Furthermore, it shows that the blastocysts cluster into two different groups based on the expression of these genes. We can extract these groups and identify marker genes for them as follows:

```
> cell_populations <- M3DropGetHeatmapClusters(heat_out, k=4, type="cell")
> library("ROCR")
> marker_genes <- M3DropGetMarkers(norm, cell_populations)
```

The first function cuts the dendrogram from the heatmap to produce k clusters of cells. These labels are stored in cell\_populations. The second function tests all genes as marker genes for the provided clusters.

Marker genes are ranked by the area-under the ROC curve (AUC) for predicting the population with the highest expression of the gene from the other groups. Significance is calculated using a Wilcoxon-rank-sum test. Now we can examine the marker genes of the two clusters of blastocyst cells more closely.

```
> head(marker_genes[marker_genes$Group==4,],20)
```

	AUC	Group	pval
1600015I10Rik	1	4	1.322348e-26
4933411G11Rik	1	4	3.387992e-25
5033411D12Rik	1	4	7.122560e-26
A530032D15Rik	1	4	1.448636e-13
AA415398	1	4	3.972292e-20
AU015228	1	4	1.276920e-15

Abca13	1	4	1.001830e-15
Bex6	1	4	6.309915e-23
Card6	1	4	3.483869e-21
Dcdc2c	1	4	3.703031e-14
Dnajb9	1	4	1.055742e-13
Dub1a	1	4	1.400339e-21
Fbxo33	1	4	1.261562e-13
Fiz1	1	4	1.122001e-13
Gm10696	1	4	3.108289e-19
Gm10697 Tdpoz5	1	4	7.122560e-26
Gm11544	1	4	5.752433e-19
Gm13040 Gm13043 Gm13057	1	4	5.547325e-24
Gm13040 Gm13057	1	4	5.547325e-24
Gm13078	1	4	1.440336e-13

```
> marker_genes[rownames(marker_genes)=="Cdx2",]
```

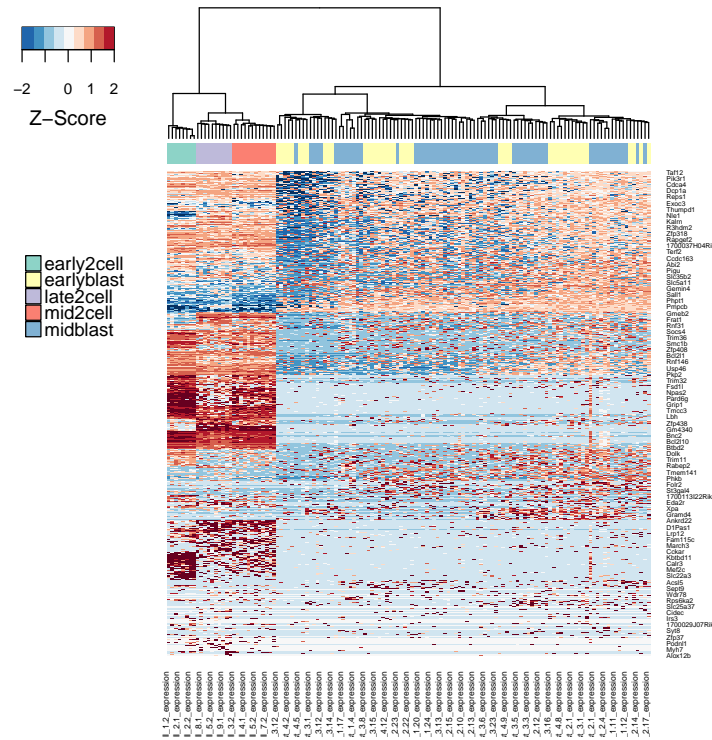
	AUC	Group	pval
Cdx2	0.8859793	1	1.286591e-12

This shows that the inner cell mass (ICM) marker Sox2 is one of the top 20 markers for group 4 and that the trophectoderm (TE) marker Cdx2 is a marker for group 3, suggesting these two clusters corespond to ICM and TE cells within each blastocyst.

## Comparing to Other Methods

The DANB features look similar to M3Drop though identifies more genes due to the simplified noise model it uses, which is appropriate for UMI-tagged data but too permissive for data with high amplification noise.

```
> heat_out <- M3DropExpressionHeatmap(NBDropFS$Gene, norm,
+                                     cell_labels = labels)
```

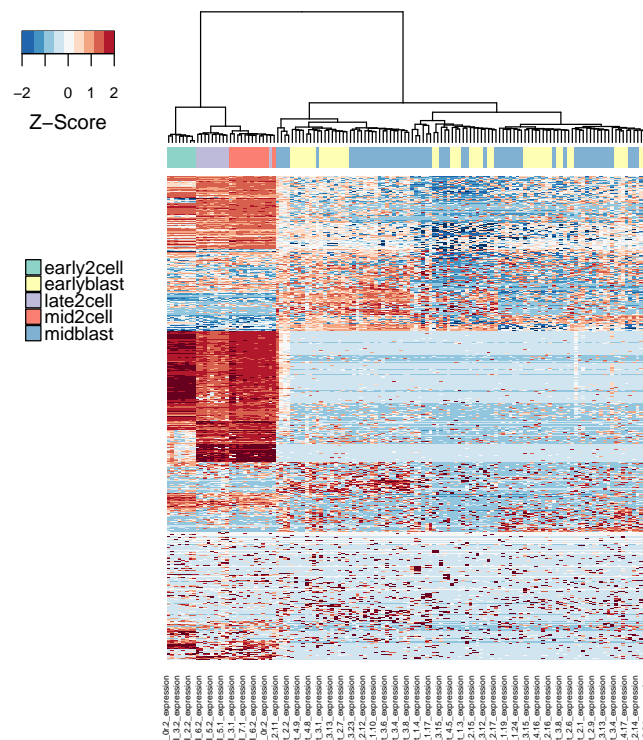


**Figure 7:** Heatmap of expression of DANB features.

The HVG method is more sensitive to lowly expressed genes. In addition, the quadratic model it uses frequently over estimates the expected variability of highly expressed genes. This is in contrast with M3Drop which recognizes the low information available for lowly expressed genes.

This difference can be seen by comparing the heatmaps for the respective genes. The highly variable genes contains many more genes exhibiting just noisy expression, whereas nearly all genes detected by M3Drop are clearly differentially expressed across the different cell populations.

```
> heat_out <- M3DropExpressionHeatmap(HVG, norm,
+                                     cell_labels = labels)
```



## References