flowVS: Variance stabilization in flow cytometry (and microarrays)

Ariful Azad, Bartek Rajwa, Alex Pothen

October 26, 2021

Email: azad@iu.edu

Contents

1 Licensing **2**

2 Variance stabilization in flow cytometry **2**
2.1 Why variance stabilization might be needed 2
2.2 Our approach 2
2.3 Related work 2

3 Examples **3**
3.1 Healthy Data (HD) 3
3.2 Immune Tolerance Data (ITN) 6

4 Variance stabilization in microarray data **7**
4.1 Example 9

5 Sessioninfo **12**

1 Licensing

Under the Artistic License, you are free to use and redistribute this software for academic and personal use.

2 Variance stabilization in flow cytometry

2.1 Why variance stabilization might be needed

Scientists often compare cell populations (clusters of cells with similar marker expressions) to detect changes in populations across biological conditions. The between-population changes might help us to diagnose diseases, develop new drugs and understand the immune system in general. Comparing cell populations in conventional statistical framework (e.g., t-test, F-test, etc.) often requires variance homogeneity of the cell populations. Furthermore, algorithms for constructing meta-clustering and templates such as `flowMatch` [1,2, 4] and `FLAME` [10] can also use the homogeneity of clusters when creating homogeneous meta-clusters. Hence, within-population variance stabilization might be beneficial in between-population comparisons, which could enhance our effort in automating biological discovery based on flow cytometry.

2.2 Our approach

In this software package `flowVS` [3], we developed a variance stabilization (VS) method based on maximum likelihood (ML) estimation, which is built on top of a commonly used inverse hyperbolic since (asinh) transformation. The choice of asinh function is motivated by its success as a variance stabilizer for microarray data [6,9]. `flowVS` stabilizes the within-population variances separately for each fluorescence channel z across a collection of N samples. After transforming z by $\text{asinh}(z/c)$, where c is a normalization *cofactor*, `flowVS` identifies one-dimensional clusters (density peaks) in the transformed channel. Assume that a total of m 1-D clusters are identified from N samples with the i-th cluster having variance σ_i^2. Then the $\text{asinh}(z/c)$ transformation works as a variance stabilizer if the variances of the 1-D clusters are approximately equal, i.e., $\sigma_1^2 \sim \sigma_2^2 \sim ... \sim \sigma_m^2$. To evaluate the homogeneity of variance (also known as homoskedasticity), we use Bartlett's likelihood-ratio test [5]. From a wide range of cofactors, our algorithm selects one that minimizes Bartlett's test statistics, resulting in a transformation with the best possible VS. Note that, in contrast to other transformation approaches, our algorithm apply the same transformation to corresponding channels in every sample. `flowVS` is therefore an explicit VS method that stabilizes within-population variances in each channel by evaluating the homoskedasticity of clusters with a likelihood-ratio test.

The scope and limitations of flowVS are as follows:

- `flowVS` is a method for selecting parameters for transformation so that within-population variances are stabilized. Currently, one dimensional transformation is supported.
- `flowVS` stabilizes variance separately on each fluorescence channel. The same channel in all samples will be transformed with the same parameter.
- For each channel, `flowVS` stabilizes variance of a collection of flow cytometry samples. Variance can not be efficiently stabilized from a single sample.

2.3 Related work

Several packages are available in Bioconductor (http://www.bioconductor.org/) for transforming flow cytometry data. The `flowCore` package provides several transformation routines that transform data using logarithm, hyperlog, generalized Box-Cox, and biexponential (e.g., logicle and generalized arcsinh) functions. `flowCore` also provides several functions to estimate parameters of the transformations, for example, the `estimateLogicle` function estimates the parameters for logicle transformation. Current software packages estimate parameters of transformations in a data-driven manner to maximize the likelihood (`flowTrans` by Finak et al. [8]), to satisfy the normality (`flowScape` by Ray et al. [12]), and to comply with simulations (`FCSTrans` by Qian et al. [11]). `flowTrans` estimates transformation parameters for each sample by maximizing the likelihood of data being generated by a multivariate-normal distribution on the transformed scale.

`flowScape` optimizes the normalization factor of asinh transformation by the Jarque-Bera test of normality. `FCSTrans` selects the parameters of the linear, logarithm, and logicle transformations with an extensive set of simulations. However, normalizing data may not necessarily stabilize its variance, e.g., for a Poisson variable z, $\sqrt{z+3/8}$ is an approximate variance-stabilizer, whereas $z^{2/3}$ is a normalizer [7]. Therefore, we consider an approach built upon the well-known asinh transformation and estimate transformation parameters for explicitly stabilizing within-population variations.

3 Examples

3.1 Healthy Data (HD)

In the `flowVS` package, we have included a healthy donor (HD) dataset consisting of 12 samples from three healthy individuals, "A", "C", and "D". From each individual, the samples were drawn on two different days and two technical replicates were created from each sample (i.e., $3 \times 2 \times 2 = 12$ samples). Each HD sample was stained using labeled antibodies against CD45, CD3, CD4, CD8, and CD19 protein markers. Here, an HD sample "C_4_2" means that it is collected on day 4 from individual "C" and it is the second replicate on that day. We have identified lymphocytes in each sample of the HD dataset and apply the subsequent analysis on lymphocytes.

We stabilize within-population variances in each channel of the HD dataset. At first, we load the HD dataset from `flowVS` package and estimate the optimum cofactors for CD3 and CD4 channels. The optimum parameters are identified by the `estParamFlowVS` function. The function outputs the search intervals and Bartlett's statistics at the local optimum cofactor in each interval. The global optimum cofactor is computed from the local optimum cofactors.

```
library(flowVS) #load library
```

```
## Example 1: Healthy data from flowVS package
data(HD)
## identify optimum cofactor for CD3 and CD4 channels
cofactors = estParamFlowVS(HD[1:5],channels=c('CD3', 'CD4'))
```

```
## ========================================================================
## Channel  CD3  : Finding optimum cofactor for asinh transformation
## ========================================================================
##           cf range           opt cf    Bartlett's stat      time
## ========================================================================
## [     0.37,      1.00 ]       0.52         9450.42          3.10
## [     1.00,      2.72 ]       1.71         9457.66          2.17
## [     2.72,      7.39 ]       5.60         8172.71          1.62
## [     7.39,     20.09 ]      17.09         8355.33          1.99
## [    20.09,     54.60 ]      33.27         8284.71          1.75
## [    54.60,    148.41 ]     117.97         8219.30          1.99
## [   148.41,    403.43 ]     306.02         5324.25          1.47
## [   403.43,   1096.63 ]     430.90         5597.57          2.11
## [  1096.63,   2980.96 ]    2875.95         3353.36          2.07
## [  2980.96,   8103.08 ]    7064.65           93.60          2.00
## [  8103.08,  22026.47 ]    8879.01          433.64          1.59
##
##  Optimum cofactor for  CD3  :  7064.652
## ========================================================================
##
## ========================================================================
## Channel  CD4  : Finding optimum cofactor for asinh transformation
```

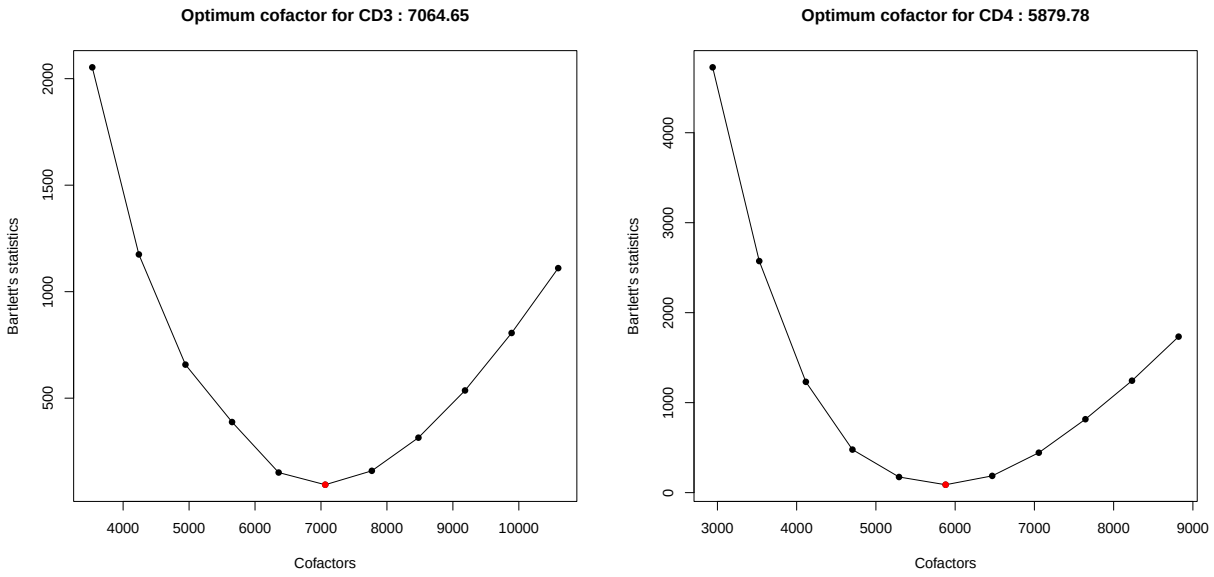


Figure 1: Transforming two fluorescence channels in the HD data. Bartlett's statistics (Y-axis) is computed from density peaks after data is transformed by different cofactors (X-axis). An optimum cofactor is obtained where Bartlett's statistics is minimum (indicated by red circles).

```
## ========================================================================
##            cf range            opt cf    Bartlett's stat      time
## ========================================================================
## [     0.37,      1.00 ]       0.40         4261.51          1.85
## [     1.00,      2.72 ]       1.10         4261.65          1.96
## [     2.72,      7.39 ]       2.89         4332.86          2.21
## [     7.39,     20.09 ]       8.11         4379.35          1.65
## [    20.09,     54.60 ]      21.28         4351.10          2.30
## [    54.60,    148.41 ]      57.83         4556.40          2.46
## [   148.41,    403.43 ]     389.22        44099.60          2.04
## [   403.43,   1096.63 ]    1058.00        24545.94          2.04
## [  1096.63,   2980.96 ]    2875.95         5068.62          1.95
## [  2980.96,   8103.08 ]    5879.78           87.73          1.41
## [  8103.08,  22026.47 ]    8879.01         1778.80          1.93
##
##  Optimum cofactor for  CD4  :  5879.785
## ========================================================================
```

After computing the optimum cofactors for the requested channels, we use `transFlowVS` function to actually transform the data by asinh transformation with the cofactors. The density plots show that the variance of populations are relatively stabilized after the transformation.

```
## transform CD3 and CD4 channels in all samples
HD.VS = transFlowVS(HD, channels=c('CD3', 'CD4'), cofactors)

## Transforming flowSet by asinh function with supplied cofactors.

## density plot (from flowViz package)
densityplot(~CD3+CD4, HD.VS, main="Transfromed CD3 and CD4 channels in HD data")
```

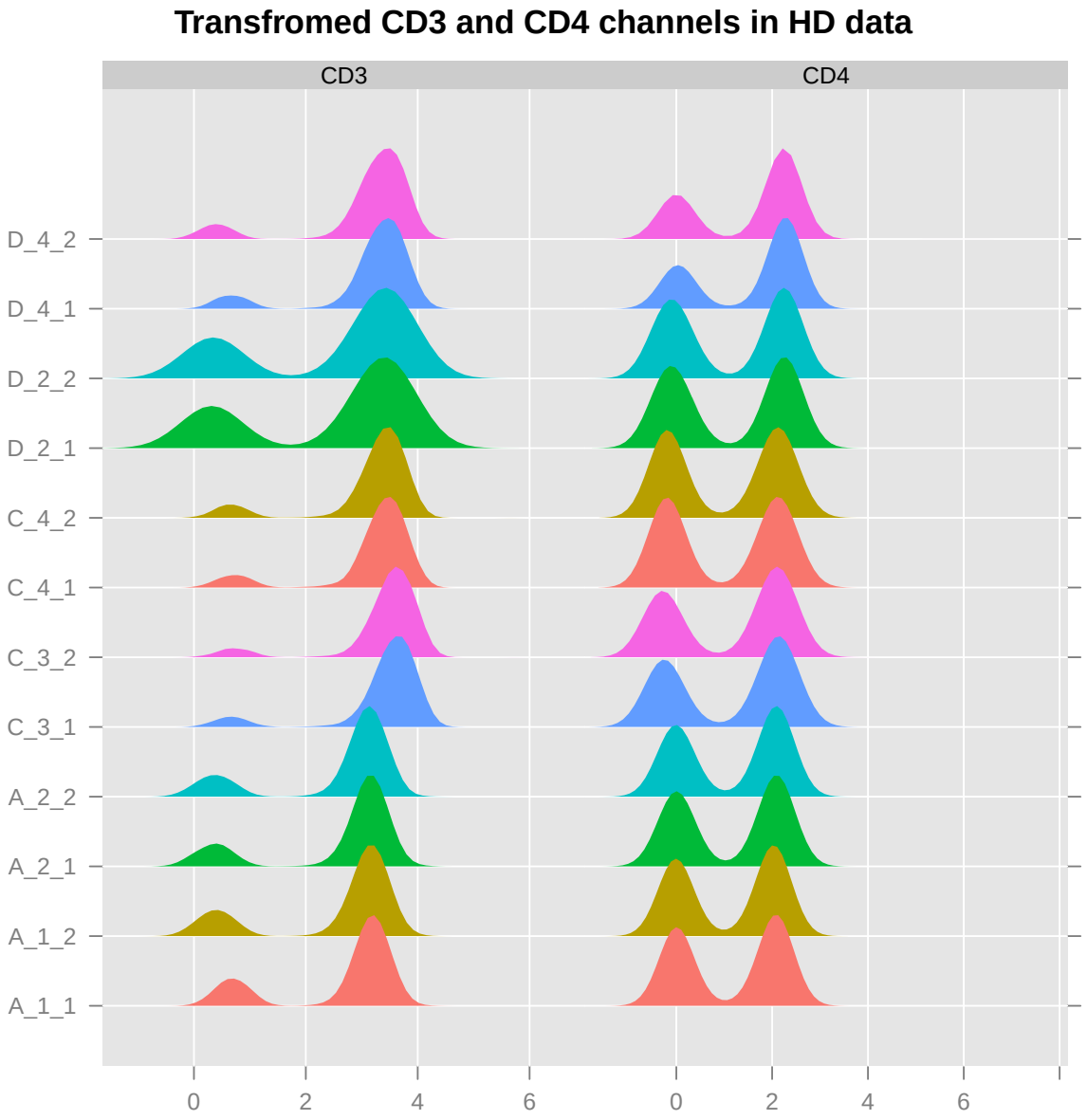


Figure 2: The density plots after the data is transformed by asins transformation with the optimum cofactors.

3.2 Immune Tolerance Data (ITN)

We use the Immune Tolerance Network (ITN) dataset from the `flowStats` package in Bioconductor to demonstrate the use of flowVS. The ITN dataset is collected from 15 patients. It includes 3 patient groups with 5 samples each. Each sample was stained using labeled antibodies against CD3, CD4, CD8, CD69 and HLADr. We identify lymphocytes in each sample of the ITN dataset beforehand by the `lymphs` function in `flowVS`.

```
## Example 2: ITN data from flowStats package
suppressMessages(library(flowStats))
data(ITN)
# identify lymphocytes
ITN.lymphs = fsApply(ITN,lymphs, list("FS"=c(200, 600),"SS"=c(0, 400)), "FSC", "SSC",FALSE)
## identify optimum cofactor for CD3 and CD4 channels
cofactors = estParamFlowVS(ITN.lymphs[1:5],channels=c('CD3', 'CD4'))

## ======================================================================
## Channel  CD3  : Finding optimum cofactor for asinh transformation
## ======================================================================
##           cf range           opt cf    Bartlett's stat      time
## ======================================================================
## [     0.37,      1.00 ]       0.83          518.03          1.62
## [     1.00,      2.72 ]       2.62          349.48          1.70
## [     2.72,      7.39 ]       3.58          324.14          1.35
## [     7.39,     20.09 ]      18.23          593.23          1.69
## [    20.09,     54.60 ]      22.01          470.14          1.39
## [    54.60,    148.41 ]      59.83         2966.10          1.65
## [   148.41,    403.43 ]     389.22          465.22          1.51
## [   403.43,   1096.63 ]     691.32          372.03          1.74
## [  1096.63,   2980.96 ]    1201.64          404.25          1.32
## [  2980.96,   8103.08 ]    3266.40          456.17          1.65
## [  8103.08,  22026.47 ]    8879.01          467.18          1.34
##
##  Optimum cofactor for  CD3  :  3.580238
## ======================================================================
##
## ======================================================================
## Channel  CD4  : Finding optimum cofactor for asinh transformation
## ======================================================================
##           cf range           opt cf    Bartlett's stat      time
## ======================================================================
## [     0.37,      1.00 ]       0.85         4592.92          1.64
## [     1.00,      2.72 ]       2.62         3979.68          1.31
## [     2.72,      7.39 ]       7.22         1820.98          2.14
## [     7.39,     20.09 ]      19.38          186.04          1.64
## [    20.09,     54.60 ]      23.97           97.07          1.31
## [    54.60,    148.41 ]      59.83          962.15          1.75
## [   148.41,    403.43 ]     162.62         3852.96          1.37
## [   403.43,   1096.63 ]     442.06         5076.06          1.78
## [  1096.63,   2980.96 ]    2091.30         5447.12          2.01
## [  2980.96,   8103.08 ]    3266.40         5459.34          1.44
## [  8103.08,  22026.47 ]    8879.01         5466.71          1.43
##
##  Optimum cofactor for  CD4  :  23.97044
## ======================================================================
```

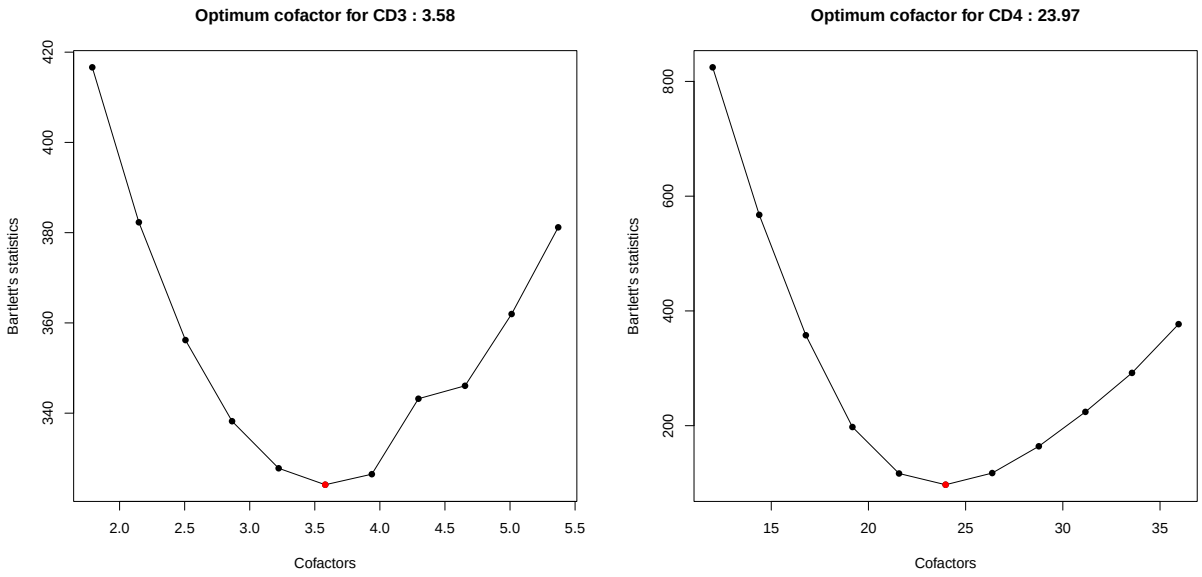


Figure 3: Transforming two fluorescence channels in the ITN data. Bartlett's statistics (Y-axis) is computed from density peaks after data is transformed by different cofactors (X-axis). An optimum cofactor is obtained where Bartlett's statistics is minimum (indicated by red circles).

After computing the optimum cofactors for the requested channels, we use `transFlowVS` function to actually transform the data by asinh transformation with the cofactors. The density plots show that the variance of populations are relatively stabilized after the transformation.

```
## transform CD4 channel in all samples
ITN.VS = transFlowVS(ITN.lymphs, channels=c('CD3', 'CD4'), cofactors)

## Transforming flowSet by asinh function with supplied cofactors.

## density plot (from flowViz package)
densityplot(~CD3+CD4, ITN.VS, main="Transfromed CD3 and CD4 channels in ITN data")
```

4 Variance stabilization in microarray data

The VS approach based on optimizing Bartlett's statistics can also be used to stabilize variance in microarray data. Assume that the expressions of m genes are measured from N samples in a microarray experiment. After transforming the data by the asinh function, the mean μ_i and variance σ_i^2 of the ith gene g_i are computed from the expressions of g_i in all samples. We then stabilizes the variances of the genes by transforming data using the asinh function with an optimum choice of cofactor. Unlike FC, a single cofactor is selected for all genes in microarrays.

The function `microVS` performs the variance stabilization in microarray data. This function transforms a microarray data matrix `z` by `asinh(z/c)` transformation where `c` is a normalizing cofactor. The cofactor is searched in the range `[cfLow, cfHigh]` and an optimum cofactor is obtained for which the transformed data is variance stabilized. The optimum cofactor is obtained by minimizing Bartlett's test statistics for homogeneity of variance.

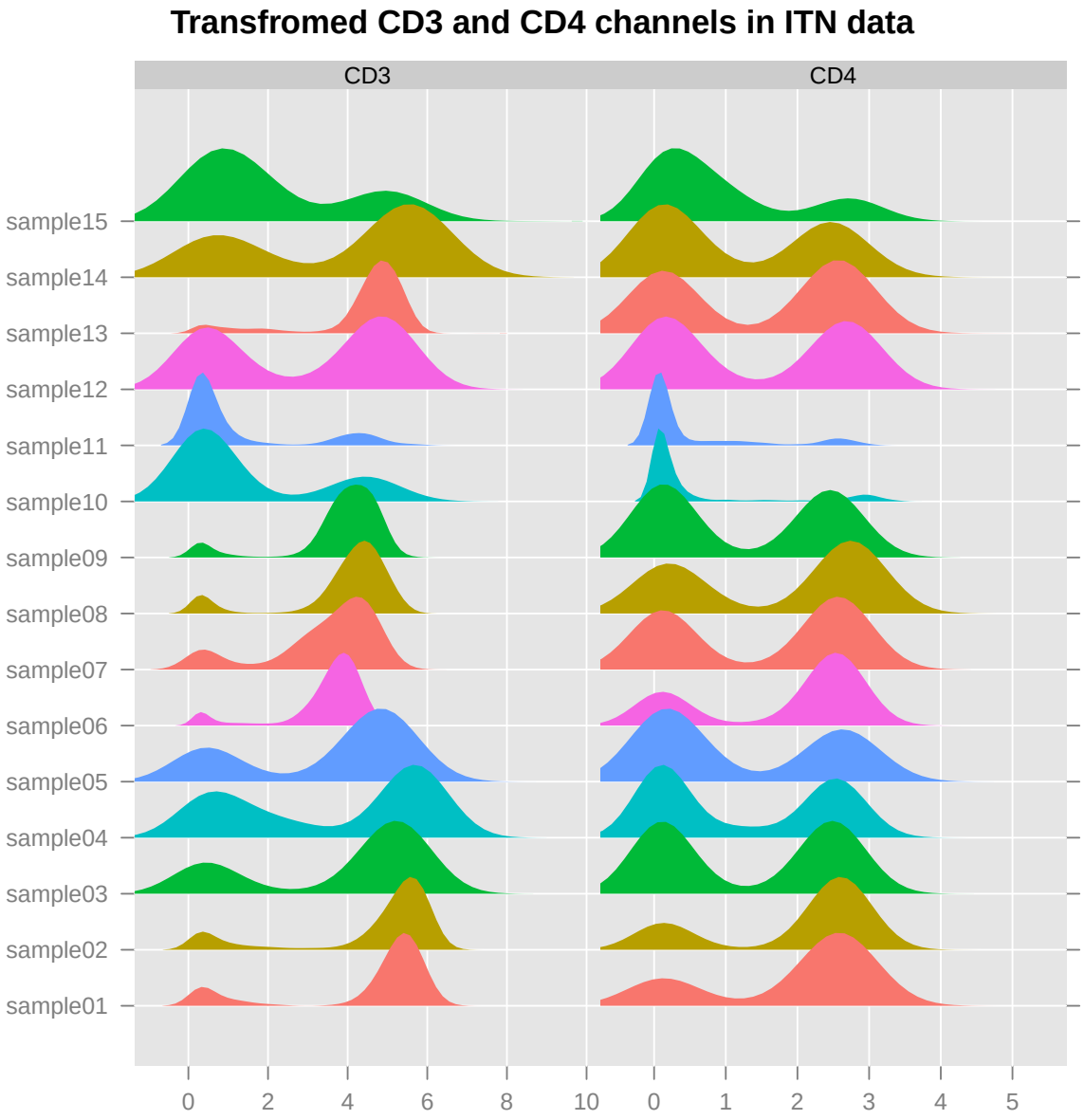


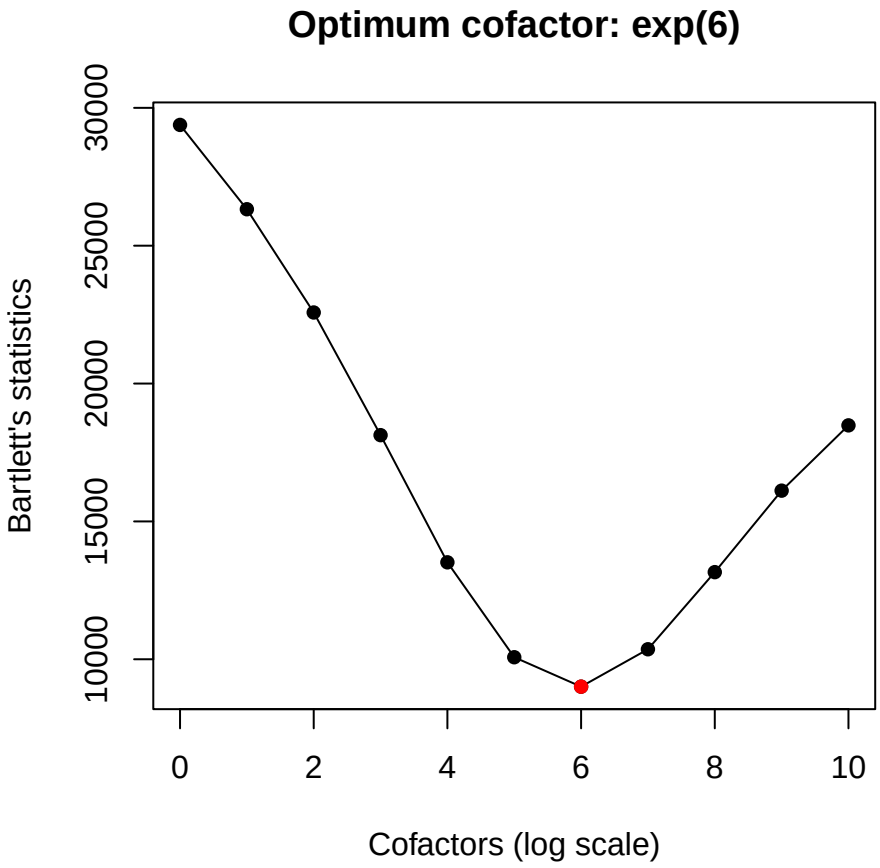
Figure 4: The density plots after the data is transformed by asins transformation with the optimum cofactors.

4.1 Example

We have applied the `microVS` to the publicly available Kidney microarray data provided by Huber et al. [9]. The Kidney data reports the expression of 8704 genes from two neighboring parts of a kidney tumor by using cDNA microarray technology. For different values of the cofactor, flowVS transforms the Kidney data with the asinh function and identifies the optimum cofactor by minimizing Bartlett's statistics. The figure below shows that a minimum value of Bartlett's statistics is obtained when the cofactor is set to exp(6) (~ 400). The optimum cofactor is then used with the asinh function to transform the Kidney data.

```
suppressMessages(library(vsn))
data(kidney)
kidney.microVS = microVS(exprs(kidney)) #variance stabilization

## ========================================================================
## Finding optimum cofactor for asinh transformation
## ========================================================================
## cofactor(log scale)     Bartlett's stat
## ========================================================================
##           0                   29380.95
##           1                   26322.83
##           2                   22577.36
##           3                   18128.08
##           4                   13515.48
##           5                   10073.42
##           6                    9005.62
##           7                   10362.98
##           8                   13159.53
##           9                   16115.28
##          10                   18483.08
##
##  Optimum cofactor : exp(6)
## ========================================================================
```



To take a closer look at the transformed data by `microVS`, we plot the variances of genes against the ranks of their means. For this purpose, we included a plotting function `plotMeanSd` that is modified from the `meanSdPlot` function in `vsn` package. We compare the performance of `microVS` and `vsn` in the figures below by using the `plotMeanSd` function. Here, the ranks of means distribute the data evenly along the x-axis and thus make it easy to visualize the homogeneity of variances. We also show the running median estimator of standard deviation by the red lines. Both `vsn` and `microVS` remove the mean-variance dependence because the red lines are approximately horizontal for both transformations. Hence, `flowVS` performs at least as well as a state-of-the-art approach developed for microarray data.

```
suppressMessages(library(vsn))
data(kidney)
kidney.vsn = vsn2(exprs(kidney)) #variance stabilization by vsn
plotMeanSd(kidney.microVS, main="Kidney data: VS by flowVS")
plotMeanSd(exprs(kidney.vsn), main="Kidney data: VS by vsn")
```

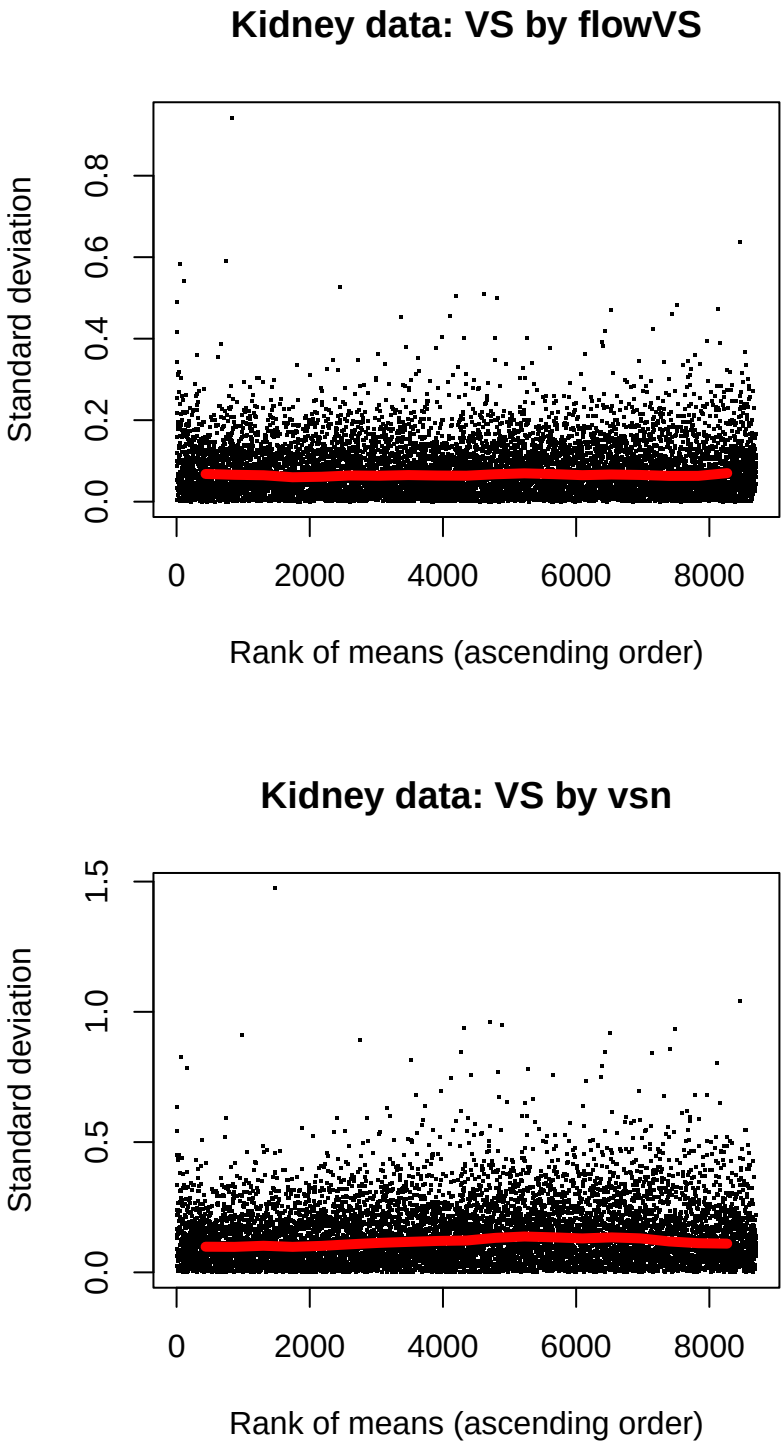


Figure 5: Variance stabilization of the Kidney microarray data by flowVs and vsn packages.

5 Sessioninfo

Here is the output of sessionInfo on the system on which this document was compiled:

```
toLatex(sessionInfo())
```

- R version 4.1.1 (2021-08-10), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 20.04.3 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.14-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.14-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Biobase 2.54.0, BiocGenerics 0.40.0, flowCore 2.6.0, flowStats 4.6.0, flowVS 1.26.0, flowViz 1.58.0, knitr 1.36, lattice 0.20-45, vsn 3.62.0
- Loaded via a namespace (and not attached): BiocManager 1.30.16, DBI 1.1.1, DEoptimR 1.0-9, IDPmisc 1.1.20, KernSmooth 2.23-20, MASS 7.3-54, Matrix 1.3-4, R6 2.5.1, RColorBrewer 1.1-2, RCurl 1.98-1.5, RProtoBufLib 2.6.0, Rcpp 1.0.7, RcppParallel 5.1.4, Rgraphviz 2.38.0, S4Vectors 0.32.0, XML 3.99-0.8, affy 1.72.0, affyio 1.64.0, assertthat 0.2.1, aws.s3 0.3.21, aws.signature 0.6.0, base64enc 0.1-3, bitops 1.0-7, cluster 2.1.2, colorspace 2.0-2, compiler 4.1.1, crayon 1.4.1, curl 4.3.2, cytolib 2.6.0, data.table 1.14.2, deSolve 1.30, digest 0.6.28, dplyr 1.0.7, ellipsis 0.3.2, evaluate 0.14, fansi 0.5.0, fda 5.4.0, fds 1.8, flowWorkspace 4.6.0, generics 0.1.1, ggplot2 3.3.5, glue 1.4.2, graph 1.72.0, grid 4.1.1, gtable 0.3.0, hdrcde 3.4, hexbin 1.28.2, highr 0.9, httr 1.4.2, jpeg 0.1-9, ks 1.13.2, latticeExtra 0.6-29, lifecycle 1.0.1, limma 3.50.0, magrittr 2.0.1, matrixStats 0.61.0, mclust 5.4.7, munsell 0.5.0, mvtnorm 1.1-3, ncdfFlow 2.40.0, pcaPP 1.9-74, pillar 1.6.4, pkgconfig 2.0.3, png 0.1-7, pracma 2.3.3, preprocessCore 1.56.0, purrr 0.3.4, rainbow 3.6, rlang 0.4.12, robustbase 0.93-9, rrcov 1.6-0, scales 1.1.1, splines 4.1.1, stats4 4.1.1, stringi 1.7.5, stringr 1.4.0, tibble 3.1.5, tidyselect 1.1.1, tools 4.1.1, utf8 1.2.2, vctrs 0.3.8, xfun 0.27, xml2 1.3.2, zlibbioc 1.40.0

References

[1] A. Azad, S. Pyne, and A. Pothen. Matching phosphorylation response patterns of antigen-receptor-stimulated T cells via flow cytometry. *BMC Bioinformatics*, 13(Suppl 2):S10, 2012.

[2] Ariful Azad, Arif Khan, Bartek Rajwa, Saumyadipta Pyne, and Alex Pothen. Classifying immunophenotypes with templates from flow cytometry. In *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics (ACM BCB)*, page 256. ACM, 2013.

[3] Ariful Azad, Bartek Rajwa, and Alex Pothen. flowvs: channel-specific variance stabilization in flow cytometry. *BMC bioinformatics*, 17(1):1–14, 2016.

[4] Ariful Azad, Bartek Rajwa, and Alex Pothen. Immunophenotype discovery, hierarchical organization, and template-based classification of flow cytometry samples. *Frontiers in oncology*, 6:188, 2016.

[5] M.S. Bartlett. Properties of sufficiency and statistical tests. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 160(901):268–282, 1937.

[6] Blythe P Durbin, Johanna S Hardin, Douglas M Hawkins, and David M Rocke. A variance-stabilizing transformation for gene-expression microarray data. *Bioinformatics*, 18(suppl 1):S105–S110, 2002.

[7] Bradley Efron. Transformation theory: How normal is a family of distributions? *The Annals of Statistics*, 10(2):323–339, 1982.

[8] G. Finak, J.M. Perez, A. Weng, and R. Gottardo. Optimizing transformations for automated, high throughput analysis of flow cytometry data. *BMC Bioinformatics*, 11(1):546, 2010.

[9] W. Huber, A. Von Heydebreck, H. Sültmann, A. Poustka, and M. Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18(suppl 1):S96–S104, 2002.

[10] S. Pyne, X. Hu, K. Wang, E. Rossin, T.I. Lin, L.M. Maier, C. Baecher-Allan, G.J. McLachlan, P. Tamayo, D.A. Hafler, et al. Automated high-dimensional flow cytometric data analysis. *Proceedings of the National Academy of Sciences*, 106(21):8519–8524, 2009.

[11] Yu Qian, Yue Liu, John Campbell, Elizabeth Thomson, Y Megan Kong, and Richard H Scheuermann. FCSTrans: An open source software system for fcs file conversion and data transformation. *Cytometry Part A*, 81(5):353–356, 2012.

[12] S. Ray and S. Pyne. A computational framework to emulate the human perspective in flow cytometric data analysis. *PLoS One*, 7(5):e35693, 2012.