CMScaller: an R package for consensus molecular subtyping of colorectal cancer pre-clinical models

11 juni 2020

Peter W. Eide[1,2,3], Jarle Bruun[1,2], Ragnhild A. Lothe[1,2,3], Anita Sveen[1,2]

[1] Department of Molecular Oncology, Institute for Cancer Research and [2] K.G.Jebsen Colorectal Cancer Research Centre, Oslo University Hospital, Oslo, NO-0424, Norway[3] Institute for Clinical Medicine, University of Oslo, Oslo, N-0318, Norway

- contact: peteid@rr-research.no or ansvee@rr-research.no
- Date: 2020-06-11
- package: CMScaller 0.99.2

1 Introduction

Colorectal cancers (CRCs) can be divided into four gene expression-based biologically distinct consensus molecular subtypes (CMS) [1]. This classification provides prognostic stratification of the patients and presents a potential basis for stratified treatment. The original CMS classifier is dependent on gene expression signals from the immune and stromal compartments and often fails to identify the poor-prognostic CMS4 mesenchymal group in immortalized cell lines and patient-derived organoids. CMScaller uses cancer cell-intrinsic, subtype-specific gene expression markers as features for *Nearest Template Prediction* [2].

2 Input data

`CMScaller` provides robust *cross platform and sample-type* performance given a balanced, homogeneous dataset of >40 unique samples.[3, 4] For less than ~40 samples, sampling variance (by-chance subtype depletion/enrichment) is a concern. Similarly, selection, *e.g.* excluding microsatellite instable (MSI) samples or including only aggressive cancers, would break an underlying assumption and bias the resulting predictions [5].

3 Quick start

3.1 Installation and dependencies

The following packages are required in order to run examples in this vignette.

- Bioconductor[6]: *Biobase*, *limma*

In addition, *edgeR* is needed for specific RNA-sequencing normalization methods and *parallel*, *snow* for `ntp` parallelization.

```
# dependencies: run if not already installed
source("https://bioconductor.org/biocLite.R")
biocLite(c("Biobase", "limma"))
# proper repository to be fixed for publication
install.packages("pathToPackageFile/CMScaller_0.99.2.tar.gz", repos=NULL)
```

3.2 CMS classification

`CMScaller` function requires an expression matrix or ExpressionSet as input (`emat`). Gene names in `rownames(emat)` must be NCBI Entrez, Ensembl or HGNC symbol identifiers. For gene symbols or Ensembl identifiers, parameter `rowNames` must be set to `symbol` or `ensg`, respectively. The code chunk below demonstrates how to perform classification using TCGA primary colorectal cancer example data [7].

- microarray data input should be pre-processed and normalized (often log_2transformed)[1].
- RNA-sequencing counts/RSEM values could be used directly by setting `RNAseq=TRUE` which activates quantile normalization and log_2transform.

[1] Hoshida[2] does not explicitly state whether input should be log_2transformed or not and examples in paper include both. Such transformation reduces the weight of genes with large deviations and will affect results at the margins.

```
library(Biobase) # if input is ExpressionSet
library(CMScaller)
# get RNA-seq counts from TCGA example data
counts <- exprs(crcTCGAsubset)
head(counts[,1:2])
##        TCGA-4N-A93T-01A-11R TCGA-4T-AA8H-01A-11R
## 1                       354                   28
## 29974                   208                  238
## 54715                    21                    1
## 87769                   767                  404
## 144568                    1                   50
## 2                      3088                 1823
# prediction and gene set analysis
par(mfrow=c(1,2))
res <- CMScaller(emat=counts, RNAseq=TRUE, FDR=0.05)
## performing log2-transform and quantile normalization...
## cosine correlation distance
## 7/530 templates features not in emat, discarded
## 92 samples; 4 classes; 81-232 features/class
## serial processing; 1000 permutation(s)...
## predicted samples/class (FDR<0.05)
##
## CMS1 CMS2 CMS3 CMS4 <NA>
##   14   23   15   28   12
## 12/92 samples set to NA
cam <- CMSgsa(emat=counts, class=res$prediction,RNAseq=TRUE)
## 12 samples with class or batch NA's excluded
```

```
# comparison with true class
table(pred=res$prediction, true=crcTCGAsubset$CMS)
##       true
## pred   CMS1 CMS2 CMS3 CMS4
##   CMS1    8    0    3    0
```

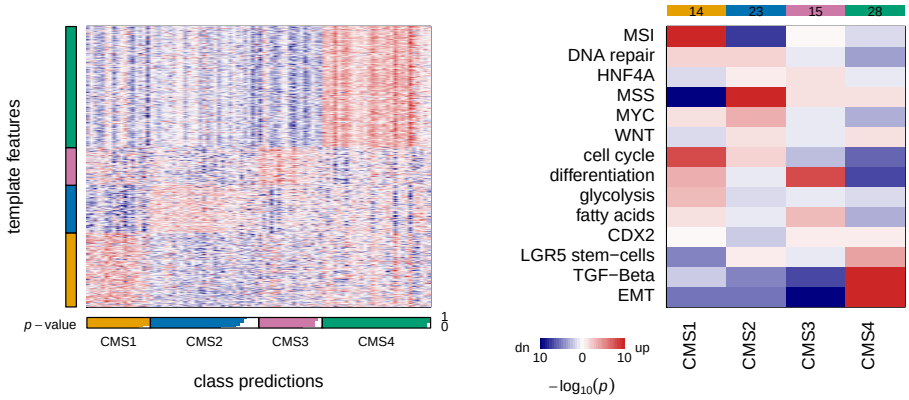


Figure 1: CMScaller graphic output
Left heatmap shows relative expression for template genes. Samples (columns) are ordered according to class predictions and confidence. The height of the white bars below gives the unadjusted prediction p-values. Genes (rows) are ordered according to class template. Heatmap color saturation indicates magnitude while red and blue indicate genes up and down relative to the sample mean. Right heatmap shows results for Camera gene set analysis. Heatmap color saturation indicates statistical significance and red and blue indicates direction of change.

```
##   CMS2    0   20    0    0
##   CMS3    0    1    6    0
##   CMS4    0    2    0   19
head(res, n=3)
##                     prediction d.CMS1 d.CMS2 d.CMS3 d.CMS4 p.value    FDR
## TCGA-4N-A93T-01A-11R       <NA>   0.76   0.70   0.72   0.89   0.907 0.9273
## TCGA-4T-AA8H-01A-11R       CMS3   0.76   0.69   0.62   0.89   0.001 0.0014
## TCGA-5M-AAT4-01A-11R       <NA>   0.74   0.69   0.73   0.83   0.794 0.8303
```

- `rownames(res)` equals `colnames(emat)`
- class predictions with `NA` for samples with adjusted-p-value $>$ threshold
- templates distances
- prediction p-values[2]
- prediction FDR-adjusted p-values

[2] lowest possible estimate of the p-value is 1/permutations

4 Package details

`CMScaller` is basically a wrapper function for `ntp`. Similarly, `CMSgsa` just provides some presets for `subCamera`.

4.1 Preparing custom templates

Templates consists of sets of subtype-specific marker genes. `subDEG` performs *limma* differential expression analysis for identification of such markers. Below, is an example on how to prepare custom templates based on a training set with known class labels. `doVoom=TRUE` enables voom transformation - required for proper *limma* modeling of RNA-sequencing counts [8].

```
emat <- crcTCGAsubset
cms <- emat$CMS.Syn
train <- sample(seq_along(cms), size=length(cms)/(2))
```

```
deg <- subDEG(emat[,train], class=cms[train], doVoom=TRUE)
## 16 samples with class or batch NA's excluded
templates <- ntpMakeTemplates(deg, resDEG=TRUE, topN=50)
templates$symbol <- fromTo(templates$probe)
tail(templates,n=3)
##     probe class symbol
## 198 22808  CMS4   MRAS
## 199  4130  CMS4  MAP1A
## 200  2687  CMS4   GGT5
```

4.2 Gene Set Analysis

`subCamera` provides gene set analysis and visualization and is a wrapper functions for `camera` in the *limma* package. `camera` controls for intra-set gene-wise correlations in order to reduce false-positive rate while retaining statistical power [9, 10]. `CMSgsa` provides preset gene sets to `subCamera`.

```
# increase left margins to accommodate gene set names
par.old <- par()
par(mfrow=c(1,1), mar=par.old$mar+c(0,4,0,0))
subCamera(counts, cms, geneList=geneSets.CRC, doVoom=TRUE)
## 29 samples with class or batch NA's excluded
```

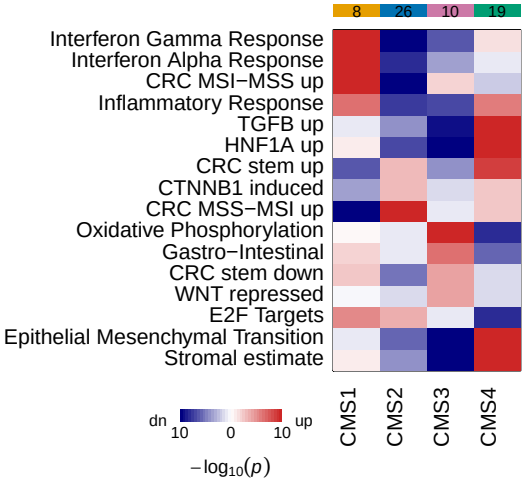


Figure 2: **Gene Set Analysis (GSA) shows that CMS are biologically distinct**

```
# restore margins
par(mar=par.old$mar)
```

4.3 PCA Analysis

`subPCA` and `plotPC` provide convenient wrapper functions for `prcomp`. `subPCA` perform PCA on the input data and plot the resulting low-dimensional embedding with samples colored according to either a continuous covariate (*.e.g.* expression of gene of interest) or group (such as CMS). `plotPC` visualizes the most important variables.

```
# increase left margins to accommodate gene set names
par(mfrow=c(1,2))
p <- subPCA(emat = crcTCGAsubset, class = crcTCGAsubset$CMS.Syn,
            normMethod = "quantile", pch=16, frame=FALSE)
plotPC(p, n=6, entrez=TRUE)
```

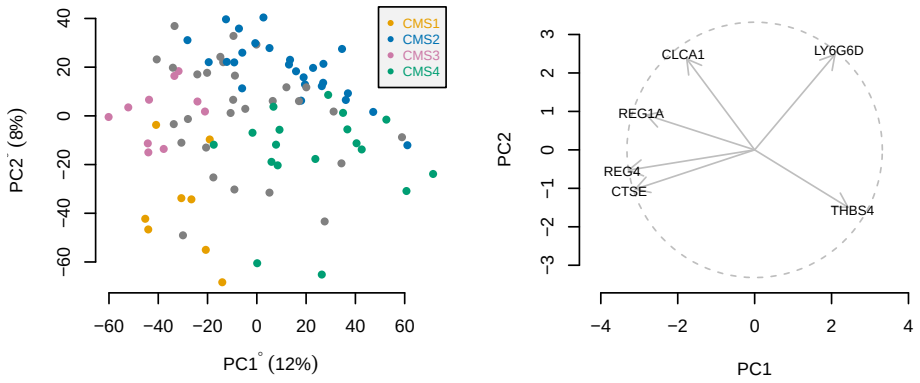


Figure 3: Principal component analysis (PCA) and CMS
First two principal components seperates CMS (left) with CMS4 characgterized by high levels of THBS4 and low levels of CLCA1 (right).

4.4 Nearest Template Prediction

`ntp` matches `templates$probe` against `rownames(emat)`. Missing features and features with `NA/NaN`'s are ignored in the prediction. `emat` should be row-wise centered and scaled.

```
# loads included emat, scales and centers
emat <- crcTCGAsubset
emat_sc <- ematAdjust(emat, normMethod="quantile")
head(emat_sc[,1:2])
##        TCGA-4N-A93T-01A-11R TCGA-4T-AA8H-01A-11R
## 1                    4.2733              1.0e+00
## 29974               -0.0046              5.1e-01
## 54715                1.2930              2.4e-04
## 87769                1.9734              1.5e+00
## 144568               0.2322              2.5e+00
## 2                   -1.5560             -1.7e+00
```

`ntp` function requires an expression matrix and templates. Since prediction confidence is estimated from permutations, strict p-value reproducibility requires `set.seed`.

```
# test set prediction
res <- ntp(emat_sc[,-train], templates, nPerm=1000)
res <- subSetNA(res, pValue=.1)
## 6/46 samples set to NA
table(pred=res$prediction, true=cms[-train])
##       true
## pred   CMS1 CMS2 CMS3 CMS4
##   CMS1    3    0    1    0
##   CMS2    0    9    0    0
```

```
##   CMS3    1    0    3    0
##   CMS4    2    0    0   10
head(res)
##                      prediction d.CMS1 d.CMS2 d.CMS3 d.CMS4 p.value    FDR
## TCGA-5M-AAT4-01A-11R       <NA>   0.68   0.69   0.70   0.86   0.775 0.7752
## TCGA-A6-5667-01A-21R       CMS2   0.77   0.50   0.84   0.68   0.001 0.0014
## TCGA-A6-6137-01A-11R       CMS2   0.75   0.55   0.70   0.82   0.001 0.0014
## TCGA-A6-6140-01A-11R       CMS2   0.77   0.44   0.78   0.86   0.001 0.0014
## TCGA-A6-6651-01A-21R       CMS4   0.72   0.66   0.85   0.39   0.001 0.0014
## TCGA-A6-6780-01A-11R       <NA>   0.65   0.84   0.71   0.76   0.131 0.1468
```

`ntp` output is a `data.frame` with $3 + K$ columns where K is the number of classes. Rows represent columns in input `emat`.

- `rownames(res)` equals `colnames(emat)`
- class predictions with `levels(res$prediction)` equaling `levels(templats$class)`
- templates distances (defaults to cosine correlation distance)
- prediction p-values
- prediction FDR-adjusted p-values

`subSetNA` function resets predictions with p-value or FDR above some arbitrary threshold to `NA`.

5 Nearest Template Prediction

Nearest template prediction (NTP) was proposed as a classification algorithm by Yujin Hoshida and published in *PLoS ONE* in 2010 [2]. It aims to provide robust single-sample class prediction for high-dimensional, noisy gene expression data. In brief, first, for each subclass, *a template*, a list of genes coherently upregulated is determined. Then, for each sample, *the distance* to each template is calculated and class is assigned based on the smallest distance. Finally, prediction confidence is assessed based on the distance of the null-distribution, estimated from *permutation tests* (feature permutation). The default distance metric selected by Hoshida was a cosine similarity-derived distance (see below). When applied to a reasonably well-balanced homogeneous dataset, row-wise centering and scaling is performed (gene means and standard deviations $\mu = 0$, $\sigma = 1$). In case of single-sample prediction, feature-wise means and standard deviations from a previous sample set are used to perform sample-wise scaling and centering. The key advantages of the NTP algorithm are conceptual simplicity, biological plausibility, ease of implementation and robustness.

Formally, N samples with expression values for P genes divided into K different classes.

- $\mathbf{X}_{[P,N]}$ centered and scaled expression matrix where column vector $x_{[P]}$ is the expression for sample n.
- M is a list of K vectors where each element m is a set of marker features with higher expression in samples belonging to class k as compared to remaining samples. m's may be of uneven length, but are typically «P
- $\mathbf{Y}_{[P,K]}$ template matrix where $y_{[P]} = [p \in m]$ for class k (0 if not marker, 1 otherwise).

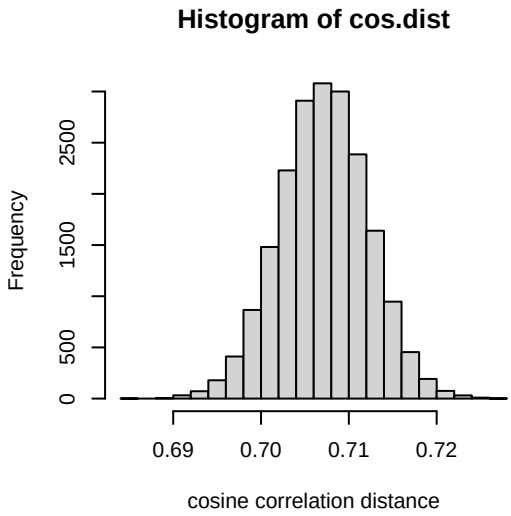
For the sample and template vectors x and y, a proper distance metric, $d_{x,y}$ for the similarity function $f(x, y)$ is given by $d = \sqrt{\frac{1}{2}(1 - f(x, y))}$ [11]. Here f is either cosine, Kendall, Pearson or Spearman correlation. Cosine similarity, the angle between two Euclidean vectors

is given by

$$f(x,y) = \cos(\theta) = \frac{\sum xy}{\sqrt{\sum x^2}\sqrt{\sum y^2}}$$

The following code chunks demonstrate NTP in code.

```
# random centered/scaled expression matrix and templates
set.seed(42)
N <- 5000;P <- 5000;K <- 4;nPerm <- 1000;n <- 1
X <- matrix(rnorm(P*N, mean=0, sd=1), ncol=N)
Y <- matrix(rbinom(P*K, size=1, prob=.01), ncol=K)
# sample-template correlations (implemented in corCosine)
cos.sim <- crossprod(X,Y) / outer(
               sqrt(apply(X, 2, crossprod)),
               sqrt(apply(Y, 2, crossprod)))
# sample-template distances (vectorized)
simToDist <- function(cos.sim) sqrt(1/2 * (1-cos.sim))
cos.dist <- simToDist(cos.sim)
hist(cos.dist, xlab="cosine correlation distance")
```



For centered, scaled and uncorrelated data, the cosine correlation distance is

$$\sqrt{0.5 \times (1-0)} \approx 0.707$$

Resulting distances are ranked among distances of permutated samples and used to estimate prediction confidence. The lowest possible p-value estimate is therefore $1/permutations$

```
# estimate prediction confidence
pred.class <- apply(cos.dist, 1, which.min)
pred.dists <- apply(cos.dist, 1, min)
null.dist <- replicate(nPerm, min(simToDist(corCosine(sample(X[,n]), Y))))
p <- rank(c(pred.dists[n], null.dist))[1]/(length(null.dist))
```

Code and plot below illustrate the uniform p-value distribution for centered and scaled uncorrelated input[3].

[3] present NTP implementation provides more conservative p-value estimates than Hoshida[2].

```
# rearrange matrix and templates for ntp input
rownames(X) <- make.names(seq_len(P))
templates <- lapply(seq_len(K), function(k) rownames(X)[Y[,k]==1])
names(templates) <- paste("k", seq_len(K))
templates <- ntpMakeTemplates(templates, resDEG=FALSE)
# permutations set to 100 to reduce processing for vignette
par(mfrow=c(1,2))
res <- ntp(X, templates, nCores=1L, nPerm=100, doPlot=TRUE)
# expect uniform distribution
hist(res$p.value, main ="", xlab="prediction p-values")
```

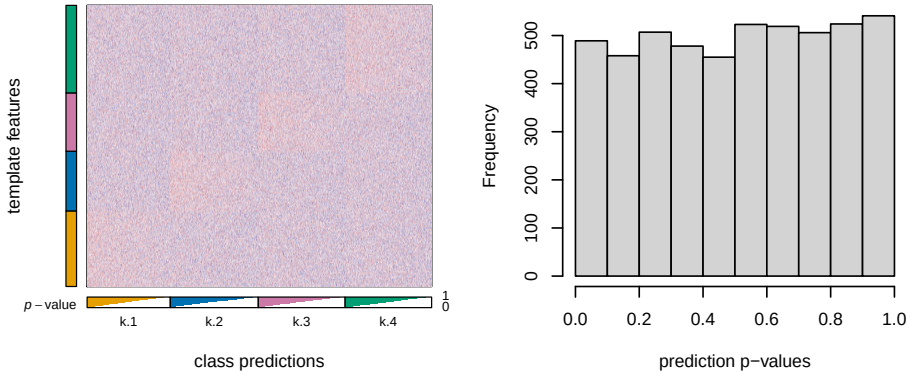


Figure 4: NTP results for random data
Left heatmap shows expression for template genes for random data with rows and columns sorted according to class. Right histogram shows the expected uniform p-value distribution for the random data.

6 Notes

- Default qualitative color palette is now from [12].
- See `news(package="CMScaller")` for additional details.

7 Session

- R version 4.0.1 (2020-06-06), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=nb_NO.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=nb_NO.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=nb_NO.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=nb_NO.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 20.04 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3`
- LAPACK: `/usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.48.0, BiocGenerics 0.34.0, BiocStyle 2.16.0, CMScaller 0.99.2
- Loaded via a namespace (and not attached): BiocManager 1.30.10, R6 2.4.1, Rcpp 1.0.4.6, assertthat 0.2.1, backports 1.1.7, bookdown 0.19, callr 3.4.3, cli 2.0.2, compiler 4.0.1, crayon 1.3.4, desc 1.2.0, devtools 2.3.0, digest 0.6.25, ellipsis 0.3.1, evaluate 0.14, fansi 0.4.1, fs 1.4.1, glue 1.4.1, htmltools 0.4.0, knitr 1.28, limma 3.44.1, magrittr 1.5, memoise 1.1.0, pkgbuild 1.0.8, pkgload 1.1.0, plotrix 3.7-8, prettyunits 1.1.1, processx 3.4.2, ps 1.3.3, purrr 0.3.4, remotes 2.1.1, rlang 0.4.6, rmarkdown 2.2, roxygen2 7.1.0, rprojroot 1.3-2, rstudioapi 0.11, sessioninfo 1.1.1, stringi 1.4.6, stringr 1.4.0, testthat 2.3.2, tools 4.0.1, usethis 1.6.1, withr 2.2.0, xfun 0.14, xml2 1.3.2, yaml 2.2.1

References

1. Guinney J, Dienstmann R, Wang X, Reyniès A de, Schlicker A, Soneson C, Marisa L, Roepman P, Nyamundanda G, Angelino P, Bot BM, Morris JS, Simon IM, Gerster S, Fessler E, Melo FDSE, Missiaglia E, Ramay H, Barras D, Homicsko K, Maru D, Manyam GC, Broom B, Boige V, Perez-Villamil B, Laderas T, Salazar R, Gray JW, Hanahan D, Tabernero J, et al.: **The consensus molecular subtypes of colorectal cancer**. *Nat Med* 2015, **21**:1350–1356.

2. Hoshida Y: **Nearest Template Prediction: A Single-Sample-Based Flexible Class Prediction with Confidence Assessment**. *PLoS One* 2010, **5**:e15543.

3. Eide PW, Bruun J, Lothe RA, Sveen A: **CMScaller: An r package for consensus molecular subtyping of colorectal cancer pre-clinical models**. *Sci Rep* 2017, **7**:16618.

4. Sveen A, Bruun J, Eide PW, Eilertsen IA, Ramirez L, Murumägi A, Arjama M, Danielsen SA, Kryeziu K, Elez E, Tabernero J, Guinney J, Palmer HG, Nesbakken A, Kallioniemi O, Dienstmann R, Lothe RA: **Colorectal cancer consensus molecular subtypes translated to preclinical models uncover potentially targetable cancer cell dependencies**. *Clin Cancer Res* 2018, **24**:794–806.

5. Zhao X, Rødland EA, Tibshirani R, Plevritis S: **Molecular subtyping for clinically defined breast cancer subgroups**. *Breast Cancer Res* 2015, **17**.

6. Huber W, Carey VJ, Gentleman R, Anders S, Carlson M, Carvalho BS, Bravo HC, Davis S, Gatto L, Girke T, Gottardo R, Hahne F, Hansen KD, Irizarry RA, Lawrence M, Love MI, MacDonald J, Obenchain V, Oleś AK, Pagès H, Reyes A, Shannon P, Smyth GK, Tenenbaum D, Waldron L, Morgan M: **Orchestrating high-throughput genomic analysis with Bioconductor**. *Nat Meth* 2015, **12**:115–121.

7. TCGA: **Comprehensive molecular characterization of human colon and rectal cancer**. *Nature* 2012, **487**:330–337.

8. Law CW, Chen Y, Shi W, Smyth GK: **Voom: Precision weights unlock linear model analysis tools for RNA-seq read counts**. *Genome Biology* 2014, **15**:R29.

9. Wu D, Smyth GK: **Camera: A competitive gene set test accounting for inter-gene correlation**. *Nucl Acids Res* 2012, **40**:e133.

10. Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK: **Limma powers differential expression analyses for RNA-sequencing and microarray studies**. *Nucl Acids Res* 2015, **43**:e47.

11. Dongen S van, Enright AJ: **Metric distances derived from cosine similarity and Pearson and Spearman correlations**. *arXiv:12083145 [cs, stat]* 2012.

12. **Color universal design (CUD) - - how to make figures and presentations that are friendly to colorblind people** [http://jfly.iam.u-tokyo.ac.jp/color/]