Variance normalization

```
library(Seurat)
library(ggplot2)
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
source("1.new_variGenes.R")
source("2.various_res.R")

malObj <- readRDS("../tmp/crc_smc.malignantcells.Rds")
head(malObj@meta.data); nrow(malObj@meta.data) # 17347
```

```
##                                     nCount_RNA nFeature_RNA         Library
## AAACCTGCATACGCCG-1-PM-PS-0001-T-A1      35998         4823 PM-PS-0001-T-A1
## AAACCTGGTCGCATAT-1-PM-PS-0001-T-A1      31383         5252 PM-PS-0001-T-A1
## AAACCTGTCCCTTGCA-1-PM-PS-0001-T-A1       7302         1713 PM-PS-0001-T-A1
## AAACGGGAGGGAAACA-1-PM-PS-0001-T-A1       3759         1233 PM-PS-0001-T-A1
## AAACGGGGTATAGGTA-1-PM-PS-0001-T-A1      23097         3874 PM-PS-0001-T-A1
## AAAGATGAGGCCGAAT-1-PM-PS-0001-T-A1      14860         3282 PM-PS-0001-T-A1
##                                     Patient  Sample Cell_subtype
## AAACCTGCATACGCCG-1-PM-PS-0001-T-A1   SMC01 SMC01-T         CMS2
## AAACCTGGTCGCATAT-1-PM-PS-0001-T-A1   SMC01 SMC01-T         CMS2
## AAACCTGTCCCTTGCA-1-PM-PS-0001-T-A1   SMC01 SMC01-T         CMS2
## AAACGGGAGGGAAACA-1-PM-PS-0001-T-A1   SMC01 SMC01-T         CMS2
## AAACGGGGTATAGGTA-1-PM-PS-0001-T-A1   SMC01 SMC01-T         CMS2
## AAAGATGAGGCCGAAT-1-PM-PS-0001-T-A1   SMC01 SMC01-T         CMS2

## [1] 17347
```

```
Idents(malObj) <- "Patient"

# filter minor libraries (less than 0.1% of total cells)
trim <- names(which(summary(malObj@meta.data$Patient) < sum(nrow(malObj@meta.data))*0.001))
print (paste(c(trim, " (n=", summary(malObj@meta.data$Patient)[trim], ") ", "will be removed"), collaps
```

```
## [1] "SMC05 (n=13) will be removed"
```

```
malObj <- subset(malObj, idents = setdiff(levels(Idents(malObj)), trim))
malObj@meta.data <- droplevels(malObj@meta.data)
head(malObj@meta.data); nrow(malObj@meta.data) # 17334
```

```
##                                     nCount_RNA nFeature_RNA         Library
## AAACCTGCATACGCCG-1-PM-PS-0001-T-A1      35998         4823 PM-PS-0001-T-A1
## AAACCTGGTCGCATAT-1-PM-PS-0001-T-A1      31383         5252 PM-PS-0001-T-A1
## AAACCTGTCCCTTGCA-1-PM-PS-0001-T-A1       7302         1713 PM-PS-0001-T-A1
## AAACGGGAGGGAAACA-1-PM-PS-0001-T-A1       3759         1233 PM-PS-0001-T-A1
## AAACGGGGTATAGGTA-1-PM-PS-0001-T-A1      23097         3874 PM-PS-0001-T-A1
## AAAGATGAGGCCGAAT-1-PM-PS-0001-T-A1      14860         3282 PM-PS-0001-T-A1
##                                     Patient  Sample Cell_subtype
## AAACCTGCATACGCCG-1-PM-PS-0001-T-A1   SMC01 SMC01-T         CMS2
## AAACCTGGTCGCATAT-1-PM-PS-0001-T-A1   SMC01 SMC01-T         CMS2
## AAACCTGTCCCTTGCA-1-PM-PS-0001-T-A1   SMC01 SMC01-T         CMS2
## AAACGGGAGGGAAACA-1-PM-PS-0001-T-A1   SMC01 SMC01-T         CMS2
## AAACGGGGTATAGGTA-1-PM-PS-0001-T-A1   SMC01 SMC01-T         CMS2
## AAAGATGAGGCCGAAT-1-PM-PS-0001-T-A1   SMC01 SMC01-T         CMS2
```

```
## [1] 17334
```

```
### 1. Variance normalization ###
malObj <- newVariGenes(malObj, "./", "Patient", c("Patient", "nCount_RNA"))
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -2.2418
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.30103
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 1.0708e-15
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -1.9685
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.30103
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 1.0255e-14
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -2.4232

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.30103

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 9.8107e-15

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -2.2028

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.30103

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 1.455e-14

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -2.2095

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.30103

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 4.1653e-15

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -2.1973

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.30103

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 8.9809e-15

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -2.3988

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.30103

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 4.4466e-15

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -2.5038

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.30103
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 2.0285e-14

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : pseudoinverse used at -2.1973

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : neighborhood radius 0.30103

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : reciprocal condition number 1.2513e-14

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 75 rows containing missing values (geom_point).

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 75 rows containing missing values (geom_point).

## [1] "regress: Patient, nCount_RNA"

## PC_ 1
## Positive:  SDCBP2-ENSG00000125775.15-7, TSPAN1-ENSG00000117472.10-4, MUC13-ENSG00000173702.7-8, CEAC
##     IFI27-ENSG00000165949.12-7, TM4SF1-ENSG00000169908.12-6, TMPRSS2-ENSG00000184012.12-9, CDA-ENSG0
##     HLA-C-ENSG00000204525.16-4, SLC40A1-ENSG00000138449.11-5, SLC2A1-ENSG00000117394.23-8, ISG15-ENS
## Negative:  BIRC5-ENSG00000089685.15-7, HMGB2-ENSG00000164104.12-7, PTTG1-ENSG00000164611.13-5, TUBA1
##     ZWINT-ENSG00000122952.17-7, CENPF-ENSG00000117724.13-4, TK1-ENSG00000167900.12-5, CDK1-ENSG00000
##     OLFM4-ENSG00000102837.7-4, MT1E-ENSG00000169715.15-5, REG1A-ENSG00000115386.6-4, TUBA4A-ENSG0000
## PC_ 2
## Positive:  JUN-ENSG00000177606.7-4, IER2-ENSG00000160888.7-4, EGR1-ENSG00000120738.8-4, FOS-ENSG0000
##     SOX4-ENSG00000124766.7-5, DNAJA1-ENSG00000086061.16-4, HSPA6-ENSG00000173110.8-5, HEXIM1-ENSG000
##     ADM-ENSG00000148926.10-5, CITED2-ENSG00000164442.10-6, GADD45G-ENSG00000130222.11-4, RGS16-ENSG0
## Negative:  S100A11-ENSG00000163191.6-5, TFF1-ENSG00000160182.3-5, TSPAN1-ENSG00000117472.10-4, FABP1
##     IFI27-ENSG00000165949.12-7, ADIRF-ENSG00000148671.14-5, LCN2-ENSG00000148346.12-4, LGALS4-ENSG00
##     OPTN-ENSG00000123240.17-7, SLC40A1-ENSG00000138449.11-5, PLA2G2A-ENSG00000188257.11-5, S100A2-EN
## PC_ 3
## Positive:  GDF15-ENSG00000130513.6-5, SOX4-ENSG00000124766.7-5, LEFTY1-ENSG00000243709.1-7, JUN-ENSG
##     FOS-ENSG00000170345.10-5, KCNQ1OT1-ENSG00000269821.1-6, GADD45B-ENSG00000099860.9-7, TUBA1A-ENSG
##     DLL4-ENSG00000128917.8-5, SPINK4-ENSG00000122711.9-4, TSPYL2-ENSG00000184205.14-5, DYRK1B-ENSG00
## Negative:  UBE2C-ENSG00000175063.17-4, HMGB2-ENSG00000164104.12-7, TPX2-ENSG00000088325.16-4, BIRC5-
##     RRM2-ENSG00000171848.15-6, TK1-ENSG00000167900.12-5, TUBB-ENSG00000196230.13-4, CENPF-ENSG000001
##     SAA1-ENSG00000173432.12-5, CDA-ENSG00000158825.6-4, LCN2-ENSG00000148346.12-4, PI3-ENSG000001241
## PC_ 4
## Positive:  ISG15-ENSG00000187608.10-8, IFIT3-ENSG00000119917.14-4, PLAAT4-ENSG00000133321.11-5, PARP
##     SAMHD1-ENSG00000101347.10-7, TYMP-ENSG00000025708.14-7, RNF213-ENSG00000173821.19-6, CMPK2-ENSG0
##     RTP4-ENSG00000136514.3-4, SAA1-ENSG00000173432.12-5, TNFSF10-ENSG00000121858.11-5, ANXA6-ENSG000
## Negative:  FABP1-ENSG00000163586.10-5, EMP1-ENSG00000134531.10-6, TM4SF1-ENSG00000169908.12-6, NDRG1
##     ADM-ENSG00000148926.10-5, ERO1A-ENSG00000197930.13-6, PPP1R15A-ENSG00000087074.8-5, KRT20-ENSG000
##     MIR22HG-ENSG00000186594.14-6, AGR2-ENSG00000106541.12-5, LAMC2-ENSG00000058085.15-5, ZFAND2A-ENS
## PC_ 5
## Positive:  LGALS4-ENSG00000171747.9-5, NDRG1-ENSG00000104419.16-10, AGR2-ENSG00000106541.12-5, FABP1
##     TFF1-ENSG00000160182.3-5, HILPDA-ENSG00000135245.10-5, MIR210HG-ENSG00000247095.3-5, WFDC2-ENSG0
##     TSPAN1-ENSG00000117472.10-4, RASD1-ENSG00000108551.5-5, MT-TL1-ENSG00000209082.1, HSPA6-ENSG0000
## Negative:  CXCL1-ENSG00000163739.5-4, CXCL8-ENSG00000169429.11-5, NFKBIA-ENSG00000100906.11-4, TM4SF
##     TNF-ENSG00000232810.4-4, ICAM1-ENSG00000090339.9-4, LCN2-ENSG00000148346.12-4, CCN1-ENSG00000142
##     PI3-ENSG00000124102.5-4, S100A2-ENSG00000196754.13-6, NFKBIZ-ENSG00000144802.11-5, ATF3-ENSG0000

## Warning: Removed 99790 rows containing missing values (geom_point).
```

```r
pvals <- data.frame(malObj@reductions$pca@jackstraw$overall.p.values)
pcs_use <- pvals[pvals$Score > 0.001, "PC"][1]-1


### 2. Clustering ###
malObj <- various_res(malObj, pcNum = pcs_use, scaleFactor = 10, rangeMax = 2)
```

```
## Computing nearest neighbor graph

## Computing SNN

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9184
## Number of communities: 7
## Elapsed time: 3 seconds

## 1 singletons identified. 6 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8789
## Number of communities: 8
## Elapsed time: 2 seconds
## 1 singletons identified. 7 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8439
## Number of communities: 9
## Elapsed time: 3 seconds

## 1 singletons identified. 8 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8234
## Number of communities: 12
## Elapsed time: 2 seconds
## 1 singletons identified. 11 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8064
## Number of communities: 13
## Elapsed time: 3 seconds

## 1 singletons identified. 12 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6
## Calculating cluster 7
```

```
## Calculating cluster 8
```

```
## Calculating cluster 9
```

```
## Calculating cluster 10
```

```
## Calculating cluster 11
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7913
## Number of communities: 13
## Elapsed time: 3 seconds
```

```
## 1 singletons identified. 12 final clusters.
```

```
## Calculating cluster 0
```

```
## Calculating cluster 1
```

```
## Calculating cluster 2
```

```
## Calculating cluster 3
```

```
## Calculating cluster 4
```

```
## Calculating cluster 5
```

```
## Calculating cluster 6
```

```
## Calculating cluster 7
```

```
## Calculating cluster 8
```

```
## Calculating cluster 9
```

```
## Calculating cluster 10
```

```
## Calculating cluster 11
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7767
## Number of communities: 13
## Elapsed time: 3 seconds

## 1 singletons identified. 12 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10

## Calculating cluster 11

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7621
## Number of communities: 15
## Elapsed time: 3 seconds

## 1 singletons identified. 14 final clusters.

## Calculating cluster 0
## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10

## Calculating cluster 11

## Calculating cluster 12

## Calculating cluster 13

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7489
## Number of communities: 15
## Elapsed time: 3 seconds

## 1 singletons identified. 14 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5
## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10

## Calculating cluster 11

## Calculating cluster 12

## Calculating cluster 13

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7351
## Number of communities: 16
## Elapsed time: 3 seconds

## 1 singletons identified. 15 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10
## Calculating cluster 11

## Calculating cluster 12

## Calculating cluster 13

## Calculating cluster 14

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7255
## Number of communities: 18
## Elapsed time: 3 seconds

## 1 singletons identified. 17 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10

## Calculating cluster 11

## Calculating cluster 12

## Calculating cluster 13

## Calculating cluster 14
## Calculating cluster 15

## Calculating cluster 16

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7174
## Number of communities: 19
## Elapsed time: 3 seconds

## 1 singletons identified. 18 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10

## Calculating cluster 11

## Calculating cluster 12

## Calculating cluster 13

## Calculating cluster 14

## Calculating cluster 15

## Calculating cluster 16
## Calculating cluster 17
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7095
## Number of communities: 20
## Elapsed time: 3 seconds
```

```
## 1 singletons identified. 19 final clusters.
```

```
## Calculating cluster 0
```

```
## Calculating cluster 1
```

```
## Calculating cluster 2
```

```
## Calculating cluster 3
```

```
## Calculating cluster 4
```

```
## Calculating cluster 5
```

```
## Calculating cluster 6
```

```
## Calculating cluster 7
```

```
## Calculating cluster 8
```

```
## Calculating cluster 9
```

```
## Calculating cluster 10
```

```
## Calculating cluster 11
```

```
## Calculating cluster 12
```

```
## Calculating cluster 13
```

```
## Calculating cluster 14
```

```
## Calculating cluster 15
```

```
## Calculating cluster 16
```

```
## Calculating cluster 17
## Calculating cluster 18
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.7016
## Number of communities: 21
## Elapsed time: 3 seconds
```

```
## 1 singletons identified. 20 final clusters.
```

```
## Calculating cluster 0
```

```
## Calculating cluster 1
```

```
## Calculating cluster 2
```

```
## Calculating cluster 3
```

```
## Calculating cluster 4
```

```
## Calculating cluster 5
```

```
## Calculating cluster 6
```

```
## Calculating cluster 7
```

```
## Calculating cluster 8
```

```
## Calculating cluster 9
```

```
## Calculating cluster 10
```

```
## Calculating cluster 11
```

```
## Calculating cluster 12
```

```
## Calculating cluster 13
```

```
## Calculating cluster 14
```

```
## Calculating cluster 15
```

```
## Calculating cluster 16
```

```
## Calculating cluster 17
## Calculating cluster 18

## Calculating cluster 19

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.6942
## Number of communities: 21
## Elapsed time: 3 seconds

## 1 singletons identified. 20 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10

## Calculating cluster 11

## Calculating cluster 12

## Calculating cluster 13

## Calculating cluster 14

## Calculating cluster 15

## Calculating cluster 16
## Calculating cluster 17

## Calculating cluster 18

## Calculating cluster 19

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.6863
## Number of communities: 22
## Elapsed time: 3 seconds

## 1 singletons identified. 21 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10

## Calculating cluster 11

## Calculating cluster 12

## Calculating cluster 13

## Calculating cluster 14

## Calculating cluster 15
## Calculating cluster 16

## Calculating cluster 17

## Calculating cluster 18

## Calculating cluster 19

## Calculating cluster 20

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.6796
## Number of communities: 22
## Elapsed time: 4 seconds

## 1 singletons identified. 21 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10

## Calculating cluster 11

## Calculating cluster 12

## Calculating cluster 13
## Calculating cluster 14
```

```
## Calculating cluster 15
```

```
## Calculating cluster 16
```

```
## Calculating cluster 17
```

```
## Calculating cluster 18
```

```
## Calculating cluster 19
```

```
## Calculating cluster 20
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.6729
## Number of communities: 22
## Elapsed time: 3 seconds
```

```
## 1 singletons identified. 21 final clusters.
```

```
## Calculating cluster 0
```

```
## Calculating cluster 1
```

```
## Calculating cluster 2
```

```
## Calculating cluster 3
```

```
## Calculating cluster 4
```

```
## Calculating cluster 5
```

```
## Calculating cluster 6
```

```
## Calculating cluster 7
```

```
## Calculating cluster 8
```

```
## Calculating cluster 9
```

```
## Calculating cluster 10
```

```
## Calculating cluster 11
## Calculating cluster 12

## Calculating cluster 13

## Calculating cluster 14

## Calculating cluster 15

## Calculating cluster 16

## Calculating cluster 17

## Calculating cluster 18

## Calculating cluster 19

## Calculating cluster 20

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.6666
## Number of communities: 23
## Elapsed time: 3 seconds

## 1 singletons identified. 22 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6

## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9
## Calculating cluster 10

## Calculating cluster 11

## Calculating cluster 12

## Calculating cluster 13

## Calculating cluster 14

## Calculating cluster 15

## Calculating cluster 16

## Calculating cluster 17

## Calculating cluster 18

## Calculating cluster 19

## Calculating cluster 20

## Calculating cluster 21

## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 17334
## Number of edges: 539295
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.6593
## Number of communities: 26
## Elapsed time: 3 seconds

## 1 singletons identified. 25 final clusters.

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

## Calculating cluster 4

## Calculating cluster 5

## Calculating cluster 6
## Calculating cluster 7

## Calculating cluster 8

## Calculating cluster 9

## Calculating cluster 10

## Calculating cluster 11

## Calculating cluster 12

## Calculating cluster 13

## Calculating cluster 14

## Calculating cluster 15

## Calculating cluster 16

## Calculating cluster 17

## Calculating cluster 18

## Calculating cluster 19

## Calculating cluster 20

## Calculating cluster 21

## Calculating cluster 22

## Calculating cluster 23

## Calculating cluster 24
```

```
saveRDS(malObj, "../tmp/crc_smc.malignantcells.newHVGs.Rds")
```