DE analysis for the 2016-17 cohort

EJC

Nov 2020, based on code from June 2020 + May 2019

Setup

The Babraham compute cluster does not contain a global tex installation, so a local tex is added to $PATH to allow knitting to pdf.

```
Sys.setenv(PATH=paste(Sys.getenv("PATH"),
                      "/bi/home/carre/texlive/2017/bin/x86_64-linux/",sep=":"))
```

```
library(dplyr)
library(SingleCellExperiment)
library(scater)
library(scran)
library(org.Hs.eg.db)

load(file = "../cohort_2016_17/data/SCE_QC_pass_finalised.RData")
```

```
########
# Get day 42 cells
# Aggregate by PID + UMAP clusters
# Use edgeR to calculate differentially expressed genes within each UMAP cluster for young vs old.
##########


summed <- sce[,sce$day == "d42"] %>%
  aggregateAcrossCells(.,
                       id=DataFrame(
                                 cluster=.$clusters,
                                 age=.$age, samples = .$PID))

summed
```

```
## class: SingleCellExperiment
## dim: 58051 90
## metadata(0):
## assays(1): counts
## rownames(58051): ENSG00000223972 ENSG00000227232 ... ENSG00000277475
##   ENSG00000268674
## rowData names(0):
## colnames: NULL
## colData names(48): lane i5 ... age samples
```

```
## reducedDimNames(2): PCA UMAP
## spikeNames(0):
## altExpNames(0):
```

```
#
# This combines several colData columns.
# This may be useful for index sort data (eg average cell width or for MFIs)
# But I have not explored exactly what data manipulation is taking place.
# Therefore treat the colData with extreme caution.
#

library(edgeR)
### Loop for all clusters / labels

de.results.d42.YvsO <- list()

for (i in levels(factor(summed$clusters))) {
  current <- summed[,i==summed$clusters]
  y <- DGEList(counts(current), samples=colData(current))

  discarded <- isOutlier(colSums(counts(current)), log=TRUE, type="lower")
  y <- y[,!discarded]
  y <- y[filterByExpr(y, min.count = 1, min.total.count =10, min.prop = 0.1),]
  y <- calcNormFactors(y)

  design <- try(
    model.matrix(~  factor(age), y$samples),
    silent=TRUE
  )
  if (is(design, "try-error") ||
      qr(design)$rank==nrow(design) ||
      qr(design)$rank < ncol(design))
  {
    # Skipping labels without contrasts or without
    # enough residual d.f. to estimate the dispersion.
    next
  }

  y <- estimateDisp(y, design)
  fit <- glmQLFit(y, design)
  res <- glmQLFTest(fit, coef=ncol(design))
  de.results.d42.YvsO[[i]] <- res
}

#
summaries.d42.YvsO <- lapply(de.results.d42.YvsO, FUN=function(x) summary(decideTests(x, adjust.method =


sum.tab.d42.YvsO <- do.call(rbind, summaries.d42.YvsO)
sum.tab.d42.YvsO
```

```
##   Down NotSig Up
## 1    0   7378  0
## 2    0   6977  0
```

```
## 3    0   3761  0
## 4    0   2854  0
## 5    0   8756  0
```

```
########
# Get day 0 cells
# Aggregate by PID + UMAP clusters
# Use edgeR to calculate differentially expressed genes within each UMAP cluster for young vs old.
##########


summed <- sce[,sce$day == "d0"] %>%
  aggregateAcrossCells(.,
                       id=DataFrame(
                         cluster=.$clusters,
                         age=.$age, samples = .$PID))

summed
```

```
## class: SingleCellExperiment
## dim: 58051 59
## metadata(0):
## assays(1): counts
## rownames(58051): ENSG00000223972 ENSG00000227232 ... ENSG00000277475
##   ENSG00000268674
## rowData names(0):
## colnames: NULL
## colData names(48): lane i5 ... age samples
## reducedDimNames(2): PCA UMAP
## spikeNames(0):
## altExpNames(0):
```

```
#
# This combines several colData columns.
# This may be useful for index sort data (eg average cell width or for MFIs)
# But I have not explored exactly what data manipulation is taking place.
# Therefore treat the colData with extreme caution.
#

library(edgeR)
### Loop for all clusters / labels

de.results.d0.YvsO <- list()

for (i in levels(factor(summed$clusters))) {
  current <- summed[,i==summed$clusters]
  y <- DGEList(counts(current), samples=colData(current))

  discarded <- isOutlier(colSums(counts(current)), log=TRUE, type="lower")
  y <- y[,!discarded]
  y <- y[filterByExpr(y, min.count = 1, min.total.count =10, min.prop = 0.1),]
  y <- calcNormFactors(y)
```

```
  design <- try(
    model.matrix(~  factor(age), y$samples),
    silent=TRUE
  )
  if (is(design, "try-error") ||
      qr(design)$rank==nrow(design) ||
      qr(design)$rank < ncol(design))
  {
    # Skipping labels without contrasts or without
    # enough residual d.f. to estimate the dispersion.
    next
  }

  y <- estimateDisp(y, design)
  fit <- glmQLFit(y, design)
  res <- glmQLFTest(fit, coef=ncol(design))
  de.results.d0.YvsO[[i]] <- res
}

#
summaries.d0.YvsO <- lapply(de.results.d0.YvsO, FUN=function(x) summary(decideTests(x, adjust.method =


sum.tab.d0.YvsO <- do.call(rbind, summaries.d0.YvsO)
sum.tab.d0.YvsO
```

```
##   Down NotSig Up
## 1    0    620  0
## 2    0    645  0
## 3    0   1268  0
## 4    0    733  0
## 5    0   2619  0
```

```
########
# Get young cells
# Aggregate by PID + UMAP clusters
# Use edgeR to calculate differentially expressed genes within each UMAP cluster for young d0 vs young
##########


summed <- sce[,sce$age == "young"] %>%
  aggregateAcrossCells(.,
                       id=DataFrame(
                         cluster=.$clusters,
                         day=.$day, samples = .$PID))

summed
```

```
## class: SingleCellExperiment
## dim: 58051 71
## metadata(0):
## assays(1): counts
## rownames(58051): ENSG00000223972 ENSG00000227232 ... ENSG00000277475
```

```
##   ENSG00000268674
## rowData names(0):
## colnames: NULL
## colData names(48): lane i5 ... day samples
## reducedDimNames(2): PCA UMAP
## spikeNames(0):
## altExpNames(0):
```

```
#
# This combines several colData columns.
# This may be useful for index sort data (eg average cell width or for MFIs)
# But I have not explored exactly what data manipulation is taking place.
# Therefore treat the colData with extreme caution.
#

library(edgeR)
### Loop for all clusters / labels

de.results.Y.d0vsd42 <- list()

for (i in levels(factor(summed$clusters))) {
  current <- summed[,i==summed$clusters]
  y <- DGEList(counts(current), samples=colData(current))

  discarded <- isOutlier(colSums(counts(current)), log=TRUE, type="lower")
  y <- y[,!discarded]
  y <- y[filterByExpr(y, min.count = 1, min.total.count =10, min.prop = 0.1),]
  y <- calcNormFactors(y)

  design <- try(
    model.matrix(~  factor(day), y$samples),
    silent=TRUE
  )
  if (is(design, "try-error") ||
      qr(design)$rank==nrow(design) ||
      qr(design)$rank < ncol(design))
  {
    # Skipping labels without contrasts or without
    # enough residual d.f. to estimate the dispersion.
    next
  }

  y <- estimateDisp(y, design)
  fit <- glmQLFit(y, design)
  res <- glmQLFTest(fit, coef=ncol(design))
  de.results.Y.d0vsd42[[i]] <- res
}

#
summaries.Y.d0vsd42 <- lapply(de.results.Y.d0vsd42, FUN=function(x) summary(decideTests(x, adjust.metho
```

```
sum.tab.Y.d0vsd42 <- do.call(rbind, summaries.Y.d0vsd42)
```

```
########
# Get old cells
# Aggregate by PID + UMAP clusters
# Use edgeR to calculate differentially expressed genes within each UMAP cluster for old d0 vs old d42.
##########


summed <- sce[,sce$age == "old"] %>%
  aggregateAcrossCells(.,
                       id=DataFrame(
                         cluster=.$clusters,
                         day=.$day, samples = .$PID))

summed
```

```
## class: SingleCellExperiment
## dim: 58051 78
## metadata(0):
## assays(1): counts
## rownames(58051): ENSG00000223972 ENSG00000227232 ... ENSG00000277475
##   ENSG00000268674
## rowData names(0):
## colnames: NULL
## colData names(48): lane i5 ... day samples
## reducedDimNames(2): PCA UMAP
## spikeNames(0):
## altExpNames(0):
```

```
#
# This combines several colData columns.
# This may be useful for index sort data (eg average cell width or for MFIs)
# But I have not explored exactly what data manipulation is taking place.
# Therefore treat the colData with extreme caution.
#

library(edgeR)
### Loop for all clusters / labels

de.results.O.d0vsd42 <- list()

for (i in levels(factor(summed$clusters))) {
  current <- summed[,i==summed$clusters]
  y <- DGEList(counts(current), samples=colData(current))

  discarded <- isOutlier(colSums(counts(current)), log=TRUE, type="lower")
  y <- y[,!discarded]
  y <- y[filterByExpr(y, min.count = 1, min.total.count =10, min.prop = 0.1),]
  y <- calcNormFactors(y)

  design <- try(
    model.matrix(~  factor(day), y$samples),
    silent=TRUE
  )
```

```
  if (is(design, "try-error") ||
      qr(design)$rank==nrow(design) ||
      qr(design)$rank < ncol(design))
  {
    # Skipping labels without contrasts or without
    # enough residual d.f. to estimate the dispersion.
    next
  }

  y <- estimateDisp(y, design)
  fit <- glmQLFit(y, design)
  res <- glmQLFTest(fit, coef=ncol(design))
  de.results.0.d0vsd42[[i]] <- res
}

#
summaries.0.d0vsd42 <- lapply(de.results.0.d0vsd42, FUN=function(x) summary(decideTests(x, adjust.metho


sum.tab.0.d0vsd42 <- do.call(rbind, summaries.0.d0vsd42)


########
# Get all cells
# Aggregate by PID, day + UMAP clusters
# Use edgeR to calculate differentially expressed genes within each UMAP cluster for day0 and day 42 (i
##########


summed <- sce %>%
  aggregateAcrossCells(.,
                       id=DataFrame(
                                 cluster=.$clusters,
                                 day=.$day, samples = .$PID))

summed
```

```
## class: SingleCellExperiment
## dim: 58051 149
## metadata(0):
## assays(1): counts
## rownames(58051): ENSG00000223972 ENSG00000227232 ... ENSG00000277475
##   ENSG00000268674
## rowData names(0):
## colnames: NULL
## colData names(48): lane i5 ... day samples
## reducedDimNames(2): PCA UMAP
## spikeNames(0):
## altExpNames(0):
```

```
#
# This combines several colData columns.
# This may be useful for index sort data (eg average cell width or for MFIs)
```

```
# But I have not explored exactly what data manipulation is taking place.
# Therefore treat the colData with extreme caution.
#

library(edgeR)
### Loop for all clusters / labels

de.results.d42vsd0 <- list()

for (i in levels(factor(summed$clusters))) {
  current <- summed[,i==summed$clusters]
  y <- DGEList(counts(current), samples=colData(current))

  discarded <- isOutlier(colSums(counts(current)), log=TRUE, type="lower")
  y <- y[,!discarded]
  y <- y[filterByExpr(y, min.count = 1, min.total.count =10, min.prop = 0.1),]
  y <- calcNormFactors(y)

  design <- try(
    model.matrix(~  factor(day), y$samples),
    silent=TRUE
  )
  if (is(design, "try-error") ||
      qr(design)$rank==nrow(design) ||
      qr(design)$rank < ncol(design))
  {
    # Skipping labels without contrasts or without
    # enough residual d.f. to estimate the dispersion.
    next
  }

  y <- estimateDisp(y, design)
  fit <- glmQLFit(y, design)
  res <- glmQLFTest(fit, coef=ncol(design))
  de.results.d42vsd0[[i]] <- res
}

#
summaries.d42vsd0 <- lapply(de.results.d42vsd0, FUN=function(x) summary(decideTests(x, adjust.method =


sum.tab.d42vsd0 <- do.call(rbind, summaries.d42vsd0)
sum.tab.d42vsd0
```

```
##   Down NotSig Up
## 1    0   7784 74
## 2    0   8485  0
## 3    0   6168  0
## 4    0   4852  6
## 5    0   9606  5
```

```
### Overall:
sum.tab.d42vsd0
```

```
##   Down NotSig Up
## 1    0   7784 74
## 2    0   8485  0
## 3    0   6168  0
## 4    0   4852  6
## 5    0   9606  5
```

```
sum.tab.Y.d0vsd42
```

```
##   Down NotSig Up
## 1    0   2462  0
## 2    0   4198  0
## 3    0   2549  0
## 4    0   1541  0
## 5    0   6094  0
```

```
sum.tab.0.d0vsd42
```

```
##   Down NotSig Up
## 1    0   2787  0
## 2    0   3035  0
## 3    0   2661  0
## 4    0   1497  0
## 5    0   5633  0
```

```
sum.tab.d0.Yvs0
```

```
##   Down NotSig Up
## 1    0    620  0
## 2    0    645  0
## 3    0   1268  0
## 4    0    733  0
## 5    0   2619  0
```

```
sum.tab.d42.Yvs0
```

```
##   Down NotSig Up
## 1    0   7378  0
## 2    0   6977  0
## 3    0   3761  0
## 4    0   2854  0
## 5    0   8756  0
```

```
library(org.Hs.eg.db)
lapply(de.results.d42vsd0, function(z) topTags(z,n = 100, p.value = 0.05)) %>%
  lapply(., function(x) {
    if(nrow(x) >0){
      mapIds(org.Hs.eg.db, keys=rownames(x),
       keytype="ENSEMBL", column="SYMBOL", multiVals = "first")}
})
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
## 'select()' returned 1:1 mapping between keys and columns
## 'select()' returned 1:1 mapping between keys and columns

## $`1`
## ENSG00000095485 ENSG00000162819 ENSG00000135845 ENSG00000110697 ENSG00000176438
##       "CWF19L1"          "BROX"          "PIGC"       "PITPNM1"         "SYNE3"
## ENSG00000176903 ENSG00000144579 ENSG00000145246 ENSG00000116704 ENSG00000163635
##         "PNMA1"        "CTDSP1"        "ATP10D"       "SLC35D1"         "ATXN7"
## ENSG00000105866 ENSG00000189007 ENSG00000186001 ENSG00000116863 ENSG00000135801
##           "SP4"         "ADAT2"         "LRCH3"       "ADPRHL2"         "TAF5L"
## ENSG00000082512 ENSG00000018408 ENSG00000089048 ENSG00000158555 ENSG00000123143
##         "TRAF5"         "WWTR1"          "ESF1"         "GDPD5"          "PKN1"
## ENSG00000204120 ENSG00000170266 ENSG00000142252 ENSG00000131844 ENSG00000122884
##        "GIGYF2"          "GLB1"        "GEMIN7"         "MCCC2"         "P4HA1"
## ENSG00000135686 ENSG00000108344 ENSG00000115738 ENSG00000142102 ENSG00000146282
##        "KLHL36"         "PSMD3"          "ID2"         "PGGHG"         "RARS2"
## ENSG00000005194 ENSG00000181666 ENSG00000166004 ENSG00000149591 ENSG00000198089
##       "CIAPIN1"        "ZNF875"        "CEP295"         "TAGLN"          "SFI1"
## ENSG00000214425 ENSG00000165609 ENSG00000112110 ENSG00000112983 ENSG00000163110
##              NA         "NUDT5"        "MRPL18"          "BRD8"        "PDLIM5"
## ENSG00000140931 ENSG00000180098 ENSG00000067057 ENSG00000176986 ENSG00000226015
##         "CMTM3"      "TRNAU1AP"          "PFKP"        "SEC24C"              NA
## ENSG00000169764 ENSG00000122390 ENSG00000107341 ENSG00000144231 ENSG00000162341
##          "UGP2"         "NAA60"        "UBE2R2"        "POLR2D"         "TPCN2"
## ENSG00000178082 ENSG00000133028 ENSG00000138785 ENSG00000109534 ENSG00000129675
##              NA          "SCO1"        "INTS12"          "GAR1"       "ARHGEF6"
## ENSG00000213020 ENSG00000079134 ENSG00000168924 ENSG00000128438 ENSG00000197114
##        "ZNF611"         "THOC1"         "LETM1"              NA         "ZGPAT"
## ENSG00000134899 ENSG00000166454 ENSG00000134285 ENSG00000110660 ENSG00000236675
##         "ERCC5"         "ATMIN"        "FKBP11"       "SLC35F2"              NA
## ENSG00000119688 ENSG00000023228 ENSG00000215441 ENSG00000102445 ENSG00000104973
##         "ABCD4"        "NDUFS1"              NA        "RUBCNL"         "MED25"
## ENSG00000178105 ENSG00000117984 ENSG00000171130 ENSG00000143458
##         "DDX10"          "CTSD"      "ATP6V0E2"        "GABPB2"
##
## $`2`
## NULL
##
## $`3`
## NULL
##
## $`4`
## ENSG00000228217 ENSG00000160326 ENSG00000108094 ENSG00000173786 ENSG00000112941
##              NA        "SLC2A6"          "CUL2"           "CNP"        "TENT4A"
## ENSG00000115419
##           "GLS"
##
## $`5`
## ENSG00000146909 ENSG00000130818 ENSG00000130347 ENSG00000154222 ENSG00000143457
##          "NOM1"        "ZNF426"       "RTN4IP1"        "CC2D1B"       "GOLPH3L"
```