comparing counting windows from different stages

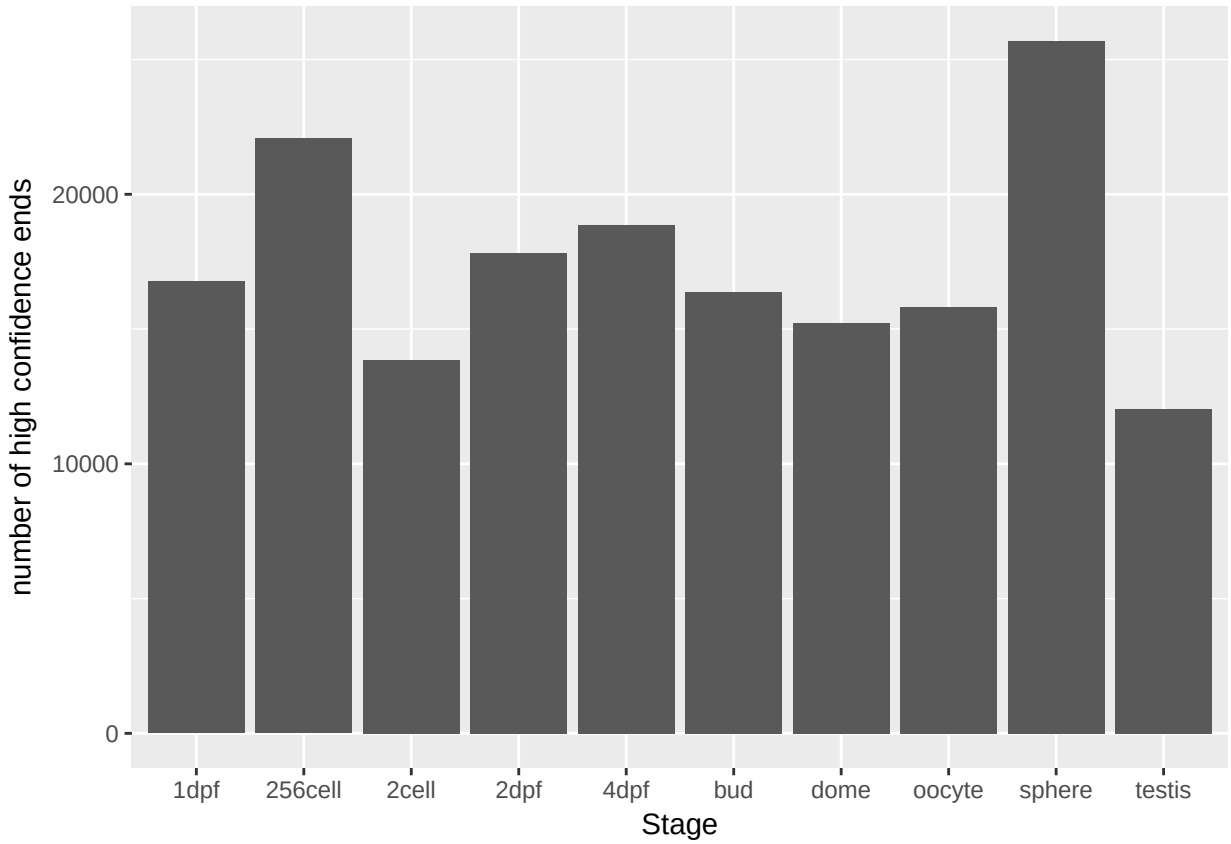
Pooja Bhat

July 19, 2017

R Markdown

```
## Warning: package 'dplyr' was built under R version 3.4.1
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
## ------------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------------
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
## Loading required package: stats4
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
## The following objects are masked from 'package:dplyr':
##
##     combine, intersect, setdiff, union
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

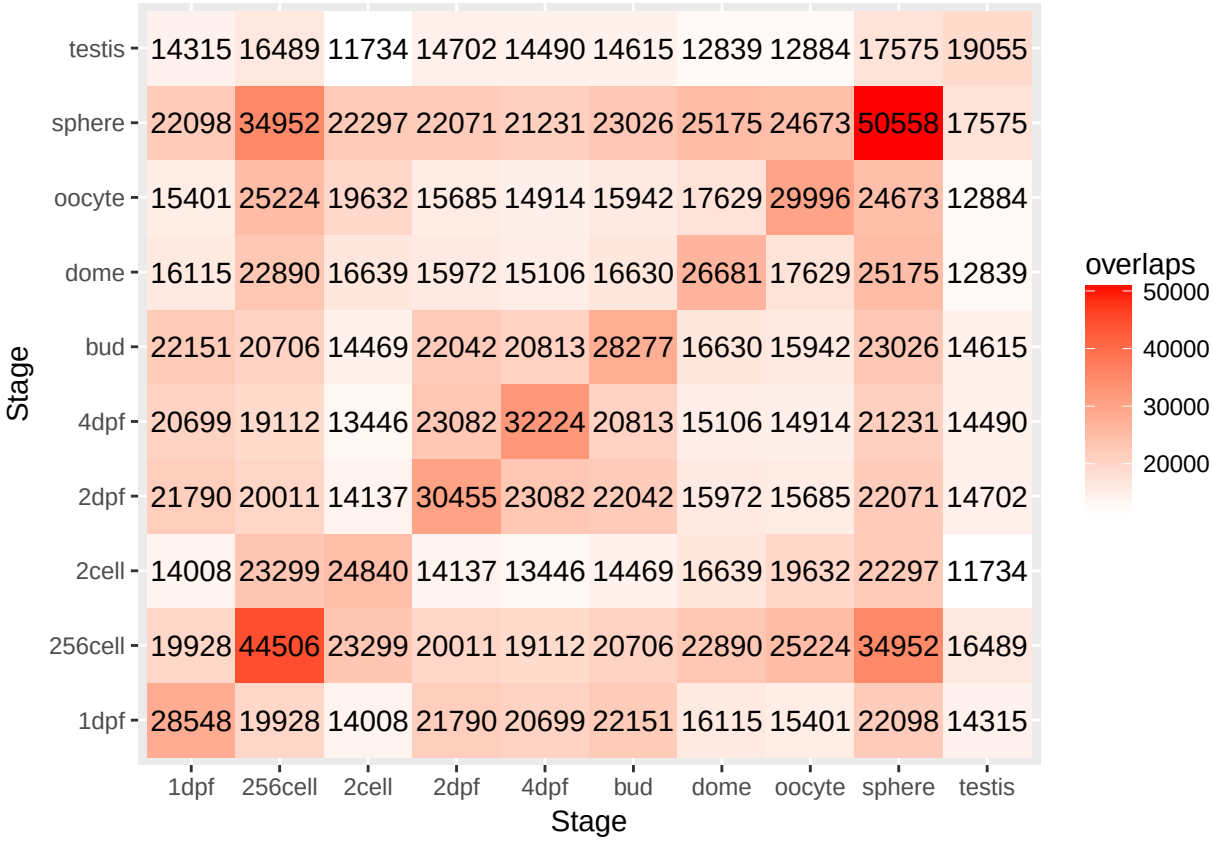
```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, cbind, colMeans,
##     colnames, colSums, do.call, duplicated, eval, evalq, Filter,
##     Find, get, grep, grepl, intersect, is.unsorted, lapply,
##     lengths, Map, mapply, match, mget, order, paste, pmax,
##     pmax.int, pmin, pmin.int, Position, rank, rbind, Reduce,
##     rowMeans, rownames, rowSums, sapply, setdiff, sort, table,
##     tapply, union, unique, unsplit, which, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following object is masked from 'package:plyr':
##
##     rename

## The following objects are masked from 'package:dplyr':
##
##     first, rename

## The following object is masked from 'package:base':
##
##     expand.grid

## Loading required package: IRanges

##
## Attaching package: 'IRanges'

## The following object is masked from 'package:plyr':
##
##     desc

## The following objects are masked from 'package:dplyr':
##
##     collapse, desc, slice

## Loading required package: GenomeInfoDb

##
## Attaching package: 'reshape'

## The following objects are masked from 'package:S4Vectors':
##
##     expand, rename

## The following objects are masked from 'package:plyr':
##
##     rename, round_any

## The following object is masked from 'package:dplyr':
##
##     rename
```

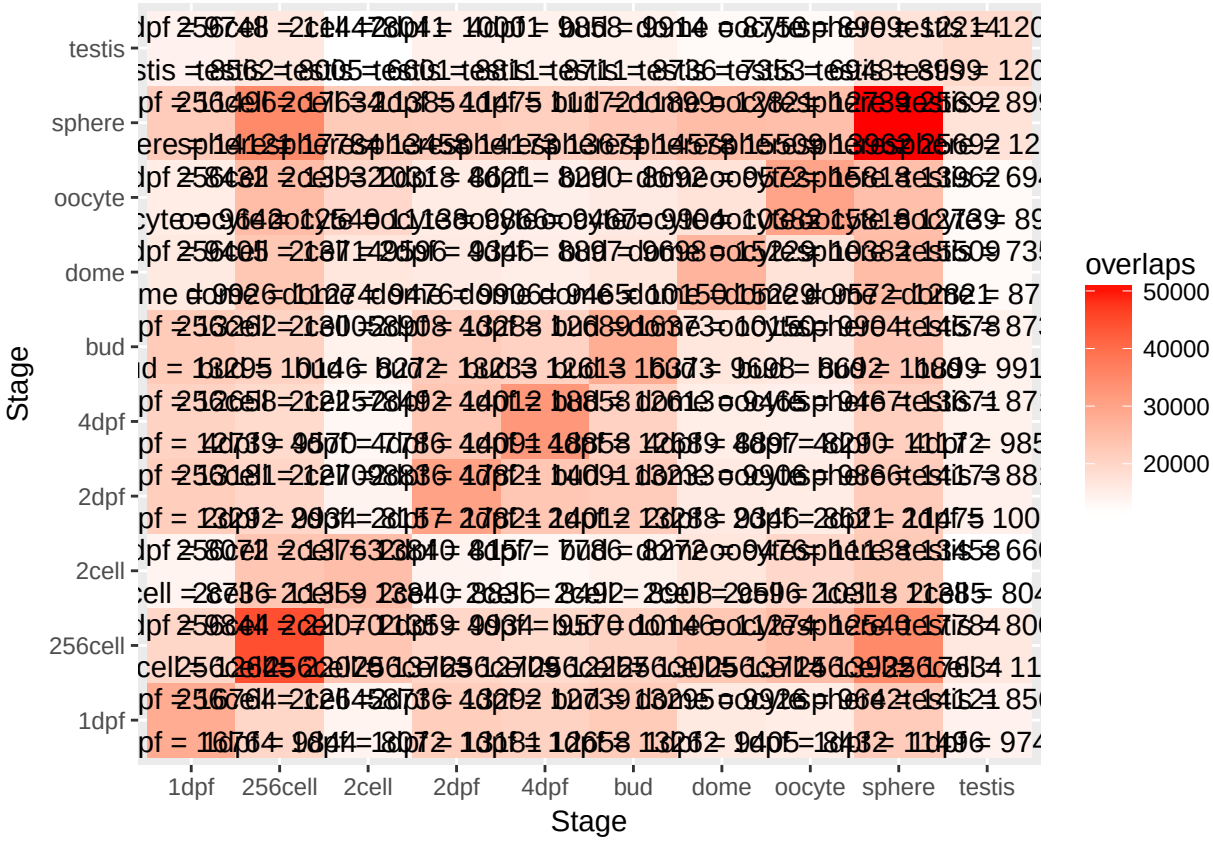

number of high confidence ends
20000
10000
0
1dpf
256cell
2cell
2dpf
4dpf
bud
dome
oocyte
sphere
testis
Stage

```
## pdf
##   2
```


testis 4676 3141 2854 4885 4849 4865 3505 2821 3951 12035
sphere 4193 6249 3619 4306 4035 4792 4274 4275 25692 3951
oocyte 2794 4216 3479 2939 2771 3076 2996 15818 4275 2821
dome 3523 3599 3043 3545 3325 3799 15229 2996 4274 3505
bud 6622 3808 3054 6660 5996 16373 3799 3076 4792 4865
4dpf 5708 3238 2738 6662 18858 5996 3325 2771 4035 4849
2dpf 6135 3470 2909 17821 6662 6660 3545 2939 4306 4885
2cell 2818 3824 13840 2909 2738 3054 3043 3479 3619 2854
256cell 3364 22070 3824 3470 3238 3808 3599 4216 6249 3141
1dpf 16764 3364 2818 6135 5708 6622 3523 2794 4193 4676
Stage
1dpf 256cell 2cell 2dpf 4dpf bud dome oocyte sphere testis
Stage
intersection
25000
20000
15000
10000
5000

```
## pdf
##   2
```

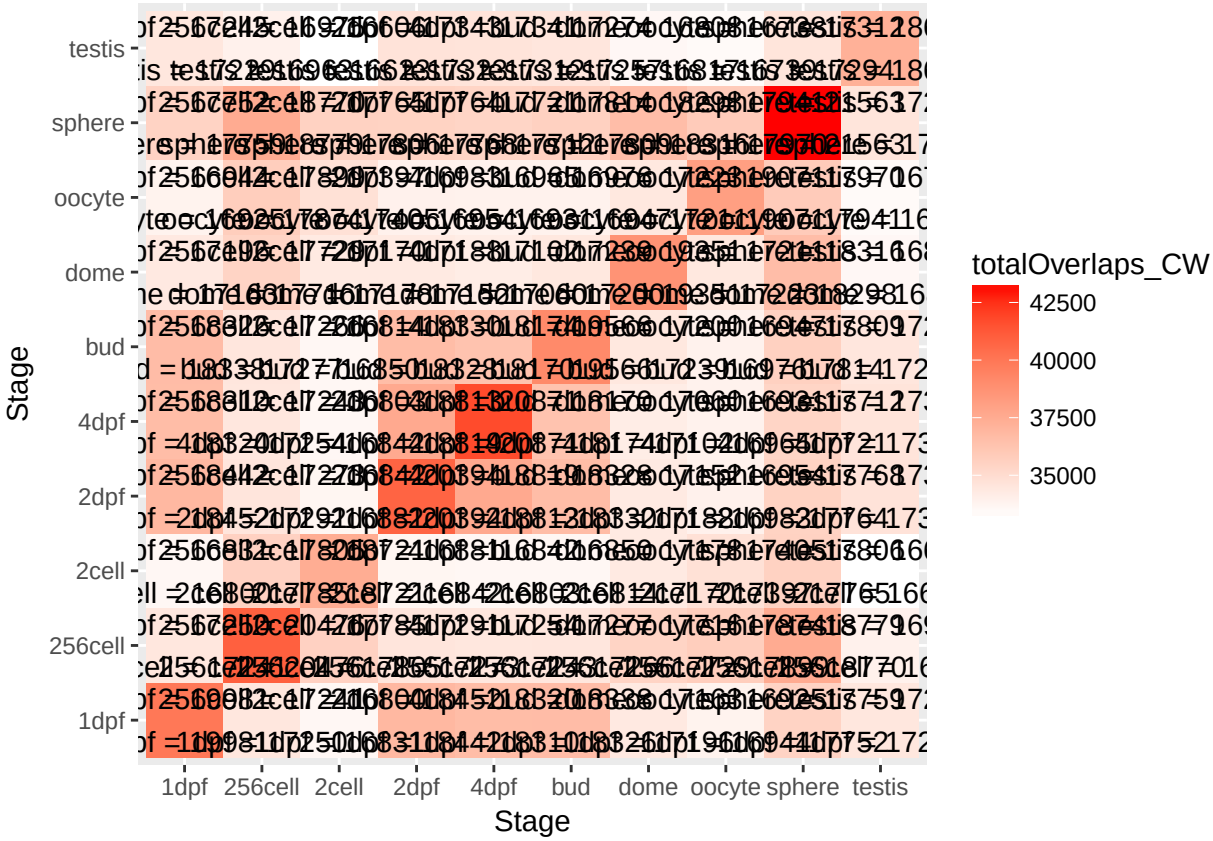

testis 14315 16489 11734 14702 14490 14615 12839 12884 17575 19055
sphere 22098 34952 22297 22071 21231 23026 25175 24673 50558 17575
oocyte 15401 25224 19632 15685 14914 15942 17629 29996 24673 12884
dome 16115 22890 16639 15972 15106 16630 26681 17629 25175 12839
bud 22151 20706 14469 22042 20813 28277 16630 15942 23026 14615
4dpf 20699 19112 13446 23082 32224 20813 15106 14914 21231 14490
2dpf 21790 20011 14137 30455 23082 22042 15972 15685 22071 14702
2cell 14008 23299 24840 14137 13446 14469 16639 19632 22297 11734
256cell 19928 44506 23299 20011 19112 20706 22890 25224 34952 16489
1dpf 28548 19928 14008 21790 20699 22151 16115 15401 22098 14315
Stage
1dpf 256cell 2cell 2dpf 4dpf bud dome oocyte sphere testis
Stage
overlaps
50000
40000
30000
20000


testis
sphere
oocyte
dome
bud
4dpf
2dpf
2cell
256cell
1dpf
Stage
1dpf 256cell 2cell 2dpf 4dpf bud dome oocyte sphere testis
Stage
overlaps
50000
40000
30000
20000

```
## pdf
```

```
##   2
```

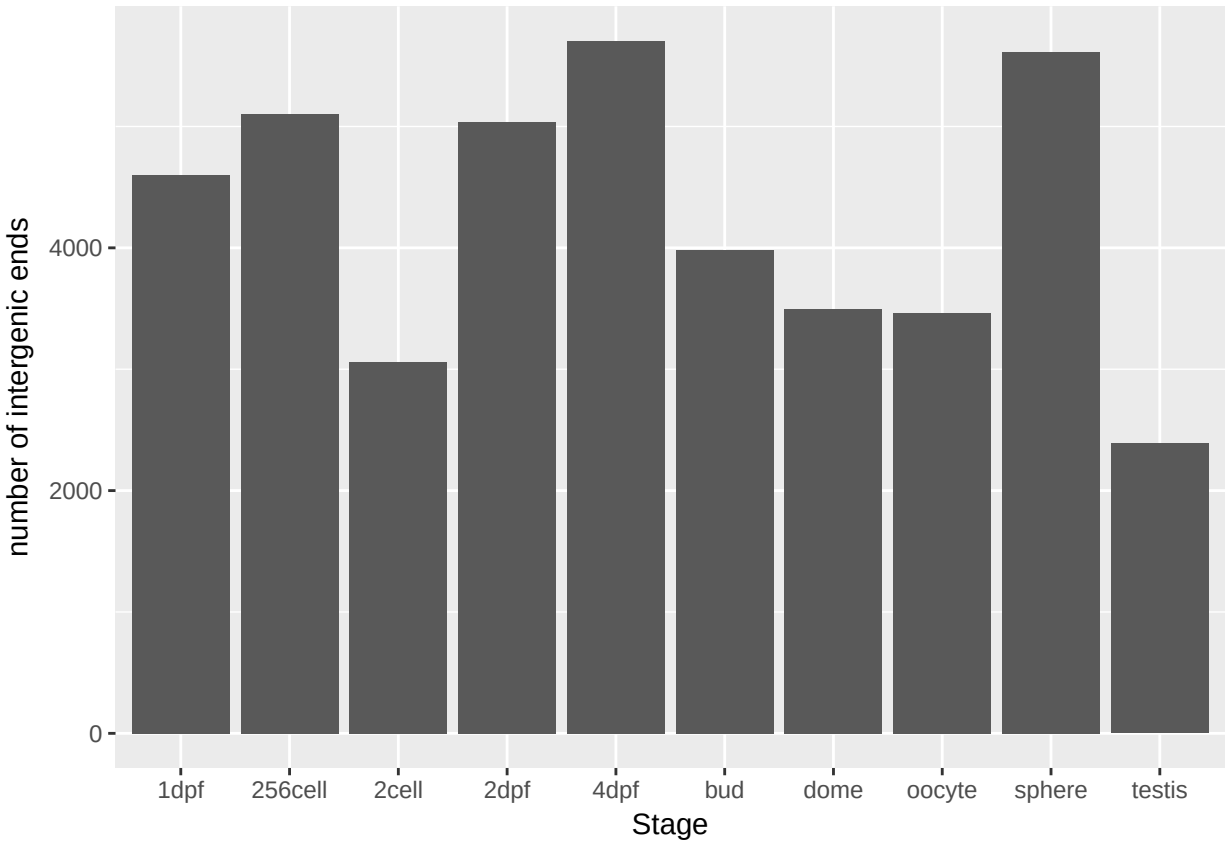
```
completePath_countingWindows = paste0(completePath,"allAnnotations.bed")

allCountingWindows = lapply(completePath_countingWindows,function(x) read.delim(x,stringsAsFactors = F,
names(allCountingWindows) = stages
allCombinations$queryOverlaps_CW = NA
allCombinations$subjectOverlaps_CW = NA
allCombinations$querySubject_CW = NA
for(i in 1:nrow(allCombinations)){

       sample1 =  allCountingWindows[which(stages == allCombinations[i,1])]
       sample2 =  allCountingWindows[which(stages == allCombinations[i,2])]
     granges_1 = makeGRangesFromDataFrame(df = sample1,keep.extra.columns = T,seqnames.field = "V1",sta

  granges_2 = makeGRangesFromDataFrame(df = sample2,keep.extra.columns = T,seqnames.field = "V1",start.

     sample1_overlap = sample1[[1]][queryHits(findOverlaps(granges_1,granges_2)),]
     sample1_overlap= sample1_overlap[!duplicated(sample1_overlap),]
     sample2_overlap = sample2[[1]][subjectHits(findOverlaps(granges_1,granges_2)),]
     sample2_overlap= sample2_overlap[!duplicated(sample2_overlap),]
     allCombinations$queryOverlaps_CW[i] = nrow(sample1_overlap)
     allCombinations$subjectOverlaps_CW[i] = nrow(sample2_overlap)
     allCombinations$querySubject_CW[i] = paste(paste(names(sample1),"=",nrow(sample1_overlap)),paste(n
  }

allCombinations$totalOverlaps_CW = allCombinations$queryOverlaps_CW + allCombinations$subjectOverlaps_C

overlapCountingWindows = ggplot(allCombinations, aes(Var1, Var2)) + geom_tile(aes(fill = totalOverlaps_

print(overlapCountingWindows)
```



So there is a high degree of overlap in the final counting windows.

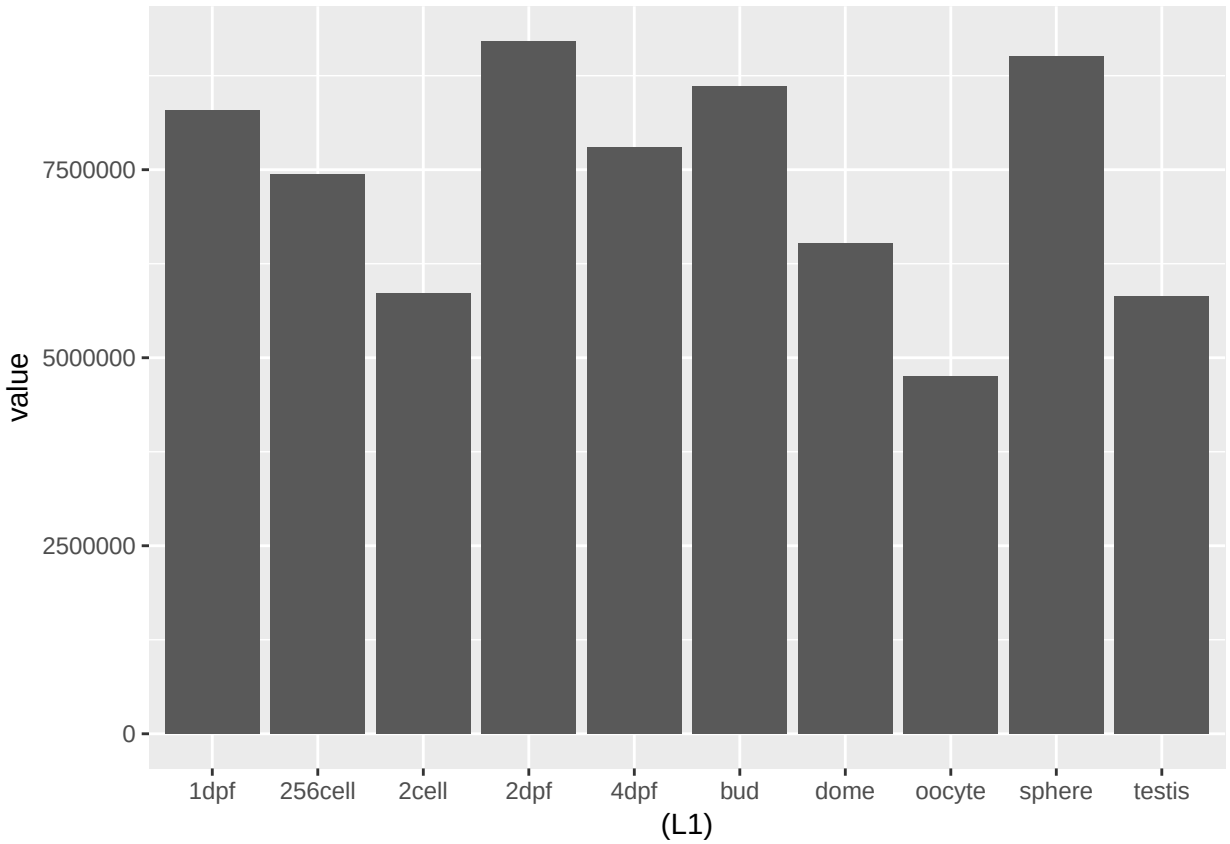
I wanted to enxt compare the high confidence intergenic ends that we identify in the different stages. This is still very exploratory and I want to see the deviation in the data we have.

```
completePath_intergenicEnds = paste0(completePath,"onlyIntergenic_90percent_n100.bed")
intergenicEnds = lapply(completePath_intergenicEnds,function(x) read.delim(x,stringsAsFactors = F,heade
names(intergenicEnds) = stages



library(reshape)
nuMberOfIntergenicEnds = melt(lapply(intergenicEnds,nrow))
colnames(nuMberOfIntergenicEnds) = c("nuMberOfIntergenicEnds","stage")
write.table(nuMberOfIntergenicEnds,"/Volumes/groups/ameres/Pooja/Projects/zebrafishAnnotation/zebrafish_

library(ggplot2)
q  = ggplot(nuMberOfIntergenicEnds,aes(x=stage,y=nuMberOfIntergenicEnds)) + geom_bar(stat = "identity")
print(q)
```



looks like there is a relaitonship between the number of ends and the number of intergenic ends. This could just be a function of the number of polyA reads used in the samples or the sequencing depth. So the initial threshold to identify priming sites will probably have to be set differently for different samples.

```
path_preprocessing = paste0(path_allStages,stages,"/output/polyAmapping_allTimepoints/logs/")
preProcessingFile=c()
for(i in 1:length(path_preprocessing)){
preProcessingFile =  c(preProcessingFile,paste0(path_preprocessing[i],list.files(path_preprocessing[i],p

}

preProcessingStats = lapply(preProcessingFile,function(x) read.table(x,stringsAsFactors = F))
names(preProcessingStats) = stages

preProcessingStats_split = lapply(preProcessingStats,function(x) strsplit(x$V1,":",T))
preProcessingStats_split = lapply(preProcessingStats_split,function(x) lapply(x,function(y) y[2]))

preProcessingStats_split_melt = melt(preProcessingStats_split)
preProcessingStats_split_melt$value = as.numeric(as.character(preProcessingStats_split_melt$value))
sampleNames = c("initialFile","adapterTrimmed","fivePrimeTrimming","polyAcontaining","finalFile")
preProcessingStats_split_melt$sample = sampleNames
preProcessingStats_split_melt$stage = rep(stages,each = 5)
library(dplyr)
filalFile = preProcessingStats_split_melt %>% filter(sample=="finalFile")
ggplot(filalFile,aes(x=(L1),y=value)) + geom_bar(stat="identity")
```



It looks like the number of ends we identify is based directly on the polyA readthrough. So we must either :

1. Normalize the cutoff to the number of polyA reads.
2. Consider even 1 read as indication of a priming site.