Annotate human TF clones

July 11, 2016

1 Settings

- This analysis was run on gale-cluster-8.

```
registerDoMC(32)
getDoParWorkers()

## [1] 32
```

```
# PANTHER protein families
pthr_class = fread('/gale/netapp/home/shhuang/data/PANTHER/hmm_classifications/9.0/PANTHER9.0_

## Warning in fread("/gale/netapp/home/shhuang/data/PANTHER/hmm_classifications/9.0/PANTHER9.0_
Bumped column 7 to type character on data row 20, field contains 'Heterotrimeric G-protein
signaling pathway-rod outer segment phototransduction#P00028>Ggamma#P00751;Inflammation
mediated by chemokine and cytokine signaling pathway#P00031>Gbetagamma#P00836;Nicotine
pharmacodynamics pathway#P06587>GNG#P06590;Heterotrimeric G-protein signaling pathway-Gq
alpha and Go alpha mediated pathway#P00027>Ggamma#P00726;Wnt signaling pathway#P00057>Ggamma#P0
hormone receptor pathway#P06664>Ggamma#P06754;Cortocotropin releasing factor receptor
signaling pathway#P04380>G-Protein#P04458;Heterotrimeric G-protein signaling pathway-Gi
alpha and Gs alpha mediated pathway#P00026>Ggamma#P00721'.  Coercing previously read
values in this column from logical, integer or numeric back to character which may
not be lossless; e.g., if '00' and '000' occurred before they will now be just '0',
and there may be inconsistencies with treatment of ',,' and ',NA,' too (if they occurred
in this  [...  truncated]

setnames(pthr_class,c("PTHR_ID","PTHR_Name","MF","BP","CC","Class","Pathway"))

# TF class annotation by TFClass database (Wingender et al NAR 2013)
tf_class = fread('/gale/netapp/home/shhuang/data/human/TFClass/TFClass_ontologies_20160725/TFCl

## various plasmid collections available at ASU, July 2016 (DNASU.org)
# ORFeome collection v8.1
orfeome_table = fread('/gale/netapp/home/shhuang/data/DNASU/Human_ORFeome_v8.1/Clones-LF.txt')
setnames(orfeome_table,c("Clone ID","Gene ID","Gene Symbol","Gene Name","Reference Sequence Ge
         c("CloneID","GeneID","GeneSymbol","GeneName","GenbankAccession","GI"))
```

```
# breat cancer 1000
bc1000_table = fread('/gale/netapp/home/shhuang/data/DNASU/Human_BC1000/BC1000_pDNRdual_complet
setnames(bc1000_table,c("Clone ID","Gene ID"),c("CloneID","GeneID"))
# human expression plasmids
hExpr_table = fread('/gale/netapp/home/shhuang/data/DNASU/Human_expression/hGenes_expr_complete
setnames(hExpr_table,c("Clone ID","Gene ID"),c("CloneID","GeneID"))
# human TF
hTF_table = fread('/gale/netapp/home/shhuang/data/DNASU/Human_TF/hTF_lenti_clones.tsv')
setnames(hTF_table,c("Clone ID","Gene ID"),c("CloneID","GeneID"))
# PSI:Biology-MR plasmids
psibiol_table = fread('/gale/netapp/home/shhuang/data/DNASU/PSI_biology/PSI_biology_clones.tsv
setnames(psibiol_table,c("Clone ID","Gene ID"),c("CloneID","GeneID"))
# PSI-2 plasmids
psi_table = fread('/gale/netapp/home/shhuang/data/DNASU/PSI/PSI_clones.txt')
```

```
## Warning in fread("/gale/netapp/home/shhuang/data/DNASU/PSI/PSI_clones.txt"): Bumped
column 3 to type character on data row 403, field contains 'TM0026'. Coercing previously
read values in this column from logical, integer or numeric back to character which
may not be lossless; e.g., if '00' and '000' occurred before they will now be just
'0', and there may be inconsistencies with treatment of ',,' and ',NA,' too (if they
occurred in this column before the bump). If this matters please rerun and set 'colClasses'
to 'character' for this column. Please note that column type detection uses the first
5 rows, the middle 5 rows and the last 5 rows, so hopefully this message should be
very rare. If reporting to datatable-help, please rerun and include the output from
verbose=TRUE.
## Warning in fread("/gale/netapp/home/shhuang/data/DNASU/PSI/PSI_clones.txt"): Bumped
column 7 to type character on data row 17483, field contains 'BC000613'. Coercing
previously read values in this column from logical, integer or numeric back to character
which may not be lossless; e.g., if '00' and '000' occurred before they will now be
just '0', and there may be inconsistencies with treatment of ',,' and ',NA,' too (if
they occurred in this column before the bump). If this matters please rerun and set
'colClasses' to 'character' for this column. Please note that column type detection
uses the first 5 rows, the middle 5 rows and the last 5 rows, so hopefully this message
should be very rare. If reporting to datatable-help, please rerun and include the
output from verbose=TRUE.

##
Read 76.0% of 65802 rows
Read 65802 rows and 16 (of 16) columns from 0.060 GB file in 00:00:03
```

```
setnames(psi_table,c("Clone ID","Gene ID"),c("CloneID","GeneID"))

# ENCODE DREAM challenge TFs
enc_tfs = fread('/gale/netapp/home/shhuang/devel/dap_hs/metadata/encode_dream_tfs_2016-06-21.t
                header=FALSE)
enc_h1_tfs = fread('/gale/netapp/home/shhuang/devel/dap_hs/metadata/encode_dream_tfs_H1_2016-0
                   header=FALSE)
```

```
# clones batch 1 sent by ASU, June 2016
hTF_clones = fread(file.path(PROJ_DEVEL_PATH,'metadata','hTFclones-LF.txt'))
```

```
## Warning in fread(file.path(PROJ_DEVEL_PATH, "metadata", "hTFclones-LF.txt")): Bumped
column 4 to type character on data row 6, field contains '100130086//100506164'. Coercing
previously read values in this column from logical, integer or numeric back to character
which may not be lossless; e.g., if '00' and '000' occurred before they will now be
just '0', and there may be inconsistencies with treatment of ',,' and ',NA,' too (if
they occurred in this column before the bump). If this matters please rerun and set
'colClasses' to 'character' for this column. Please note that column type detection
uses the first 5 rows, the middle 5 rows and the last 5 rows, so hopefully this message
should be very rare. If reporting to datatable-help, please rerun and include the
output from verbose=TRUE.
```

```
hTF_clones_split = separate_rows(hTF_clones,GeneID,sep='//')
```

```
## Warning: failed to assign NativeSymbolInfo for env since env is already defined
in the 'lazyeval' namespace
```

```
ensembl84 = useMart(host='mar2016.archive.ensembl.org', biomart='ENSEMBL_MART_ENSEMBL', dataset
```

```
# annotation of hTF clones using GeneID
hTF_eg_ens = getBM(attributes=c("ensembl_gene_id","entrezgene","description"),
                   filters="entrezgene",values=hTF_clones_split[,GeneID],
                   mart=ensembl84,uniqueRows=TRUE)
hTF_eg_ens = within(hTF_eg_ens, { entrezgene=as.character(entrezgene) })
```

```
# annotation of hTF clones using RefSeq
hTF_rs_ens = getBM(attributes=c("ensembl_gene_id","refseq_peptide","description"),
                   filters="refseq_peptide",values=hTF_clones[,RefSeq_Acc],
                   mart=ensembl84,uniqueRows=TRUE)
#hTF_rs_ens = within(hTF_rs_ens, { entrezgene=as.character(entrezgene) })
```

```
# with GeneID to Ensembl gene mapping
hTF_annot1 = merge(hTF_clones_split[,list(DNASU_CloneID,RefSeq_Acc,GeneID,GeneSymbol)],hTF_eg_e
# with RefSeq to Ensembl gene mapping
hTF_annot2 = merge(hTF_clones_split[,list(DNASU_CloneID,RefSeq_Acc,GeneID,GeneSymbol)],hTF_rs_e
x# with GeneID and/or Refseq to Ensembl gene mapping
```

```
## Error in eval(expr, envir, enclos): object 'x' not found
```

```
hTF_annot3 = merge(hTF_annot1,hTF_annot2,
                   by=c("ensembl_gene_id","GeneID","RefSeq_Acc","GeneSymbol","description"),
                   all=TRUE,suffixes=c(".eg",".rs"),
                   )
```

```
# primary is mapping by EntrezGene; if not, use mapping by RefSeq
hTF_annot3 = hTF_annot3[,DNASU_CloneID:=ifelse(is.na(DNASU_CloneID.eg),DNASU_CloneID.rs,DNASU_C
# condensed to unique; this has Ensembl ID mappings for all the clones!!!
hTF_annot4 = unique(hTF_annot3[,list(DNASU_CloneID,ensembl_gene_id,GeneID,RefSeq_Acc,GeneSymbol
```

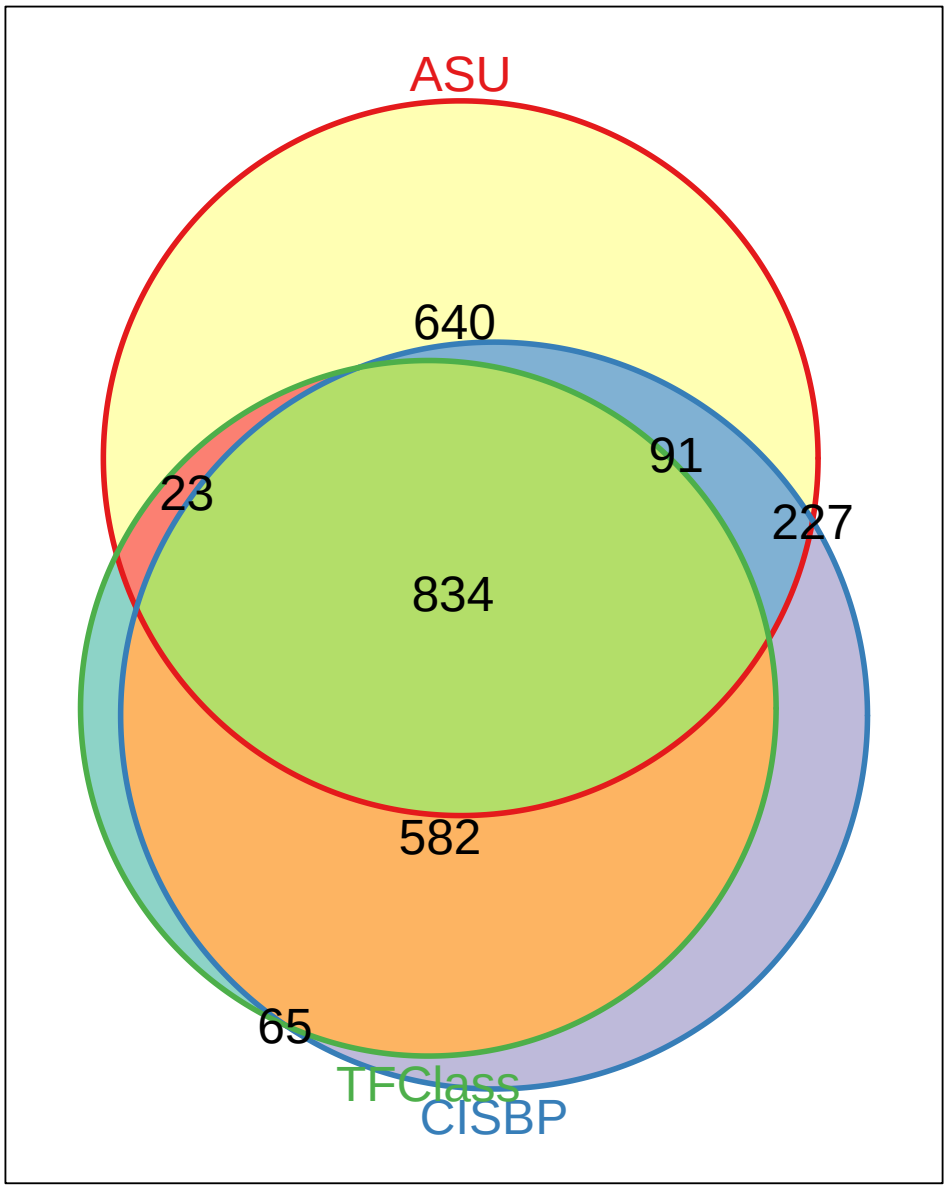
```
# add CIS-BP annotation by Ensembl Gene ID, description field is NA if TF is not in the hTF cl
hTF_annot5 = merge(hTF_annot4,cisbp_tf_info1,
                   by.x="ensembl_gene_id",by.y="DBID",
                   all.x=TRUE)
# this should be true, since all clones were mapped to ensembl_gene_id
setequal(hTF_clones[,DNASU_CloneID],hTF_annot5[,DNASU_CloneID])

## [1] TRUE

# How many clones do not have CIS-BP annotation?
hTF_annot5[,list(Family_Name=unique(Family_Name)),by="DNASU_CloneID"][,.N,by=is.na(Family_Name)

##    is.na   N
## 1:  TRUE 537
## 2: FALSE 890
```

```
# add CIS-BP annotation by Ensembl Gene ID, description field is NA if TF is not in the hTF cl
hTF_annot52 = merge(hTF_annot4,cisbp_tf_info1,
                   by.x="ensembl_gene_id",by.y="DBID",
                   all=TRUE)
# add TFClass annotation by Ensembl Gene ID, description field is NA if TF is not in the hTF c
hTF_annot53 = merge(hTF_annot52,tf_class,
                    by.x="ensembl_gene_id",by.y="ENSEMBL",
                    all=TRUE)

# consolidate CIS-BP and TFClass annotations
hTF_annot72 = ddply(hTF_annot53,c("DNASU_CloneID","GeneID","RefSeq_Acc","GeneSymbol"),function
    idx = which(!is.na(df[,'TF_Name'])) # TRUE==has CIS-BP annotation
    if (length(idx)==0) { # does not have CIS-BP annotation
        idx = which(is.na(df[,'TF_Name']) & !is.na(df[,'Class']))
        if (length(idx)==0) { # no annoation from either CIS-BP or TFClass, just get the empty
            idx = which(is.na(df[,'TF_Name']))[1]
        }
    }
    unique(df[idx,])
})

# get description for CISBP or TFClass genes that do not have overlap with ASU clones
print(length(subset(hTF_annot72,is.na(description))[,'ensembl_gene_id'])) # number of genes do

## [1] 818

print(length(unique(subset(hTF_annot72,is.na(description))[,'ensembl_gene_id']))) # number of

## [1] 809

hTF_annot72_desc = getBM(attributes=c("ensembl_gene_id","description"),
                         filters="ensembl_gene_id",values=subset(hTF_annot72,is.na(description)
```

```
                          mart=ensembl84)
hTF_annot72 = merge(hTF_annot72,hTF_annot72_desc,by.x="ensembl_gene_id",by.y="ensembl_gene_id"
                    all.x=TRUE,suffixes=c('.ASUCloneT','.ASUCloneF'))

# how many genes not overlap with ASU clones are still not annotated?
print(length(subset(hTF_annot72,is.na(description.ASUCloneT) & is.na(description.ASUCloneF))[,

## [1] 102

print(length(unique(subset(hTF_annot72,is.na(description.ASUCloneT) & is.na(description.ASUClon

## [1] 102

hTF_annot72 = hTF_annot72[with(hTF_annot72,order(is.na(DNASU_CloneID),is.na(TF_Name),is.na(Clas

write.table(hTF_annot72,paste0(rdata_prefix,'all_annot.txt'),sep='\t',quote=FALSE,row.names=FAI
```

2 Venn diagrams of overlap

```
tf_annot_list = list('ASU'=unique(hTF_annot4[,ensembl_gene_id]),
                     'CISBP'=unique(cisbp_tf_info1[,DBID]),
                     'TFClass'=unique(tf_class[,ENSEMBL]))
Vtf = Venn(tf_annot_list)

plot(Vtf)
```



3 TF overlap with all ASU ORFeome, based on Entrez ID

```
# current version
#ensembl = useMart("ensembl",dataset='hsapiens_gene_ensembl')
up = UniProt.ws(taxId=9606)
# archives
ensembl_h37 = useMart(host='grch37.ensembl.org', biomart='ENSEMBL_MART_ENSEMBL', dataset='hsapi

ens_ids = cisbp_tf_info1[,DBID]
# annotation of CIS-BP TFs using Ensembl gene ID
cisbp_ens = getBM(attributes=c("ensembl_gene_id","entrezgene","refseq_peptide"),
```

```
                  filters="ensembl_gene_id",values=ens_ids,mart=ensembl_h37)
cisbp_ens = within(cisbp_ens,{ entrezgene=as.character(entrezgene) })

cisbp_add = merge(cisbp_tf_info1,cisbp_ens,by.x="DBID",by.y="ensembl_gene_id",all.x=TRUE) # un
# mar 2016 (84): 119
# GRCh37: 99
# dec 2015: 119
# dec 2014: 119
# dec 2013: 99

# map those unmapped by ensemble gene id by UniProt
# did not get any UniProt IDs mapped
#cisbp_up = select(up,keys=cisbp_add[is.na(entrezgene),DBID],columns=c("ENTREZ_GENE"),keytype=
#cisbp_up = select(org.Hs.eg.db,keys=cisbp_add[is.na(entrezgene),DBID],columns=c("ENTREZID"),k
cisbp_comb = cisbp_add

# did we preserve all the genes in the orignal table
setequal(cisbp_tf_info1[,DBID],cisbp_comb[,DBID])

## [1] TRUE

# how many genes are unmapped?
cisbp_comb[is.na(entrezgene),]

##                 DBID    TF_Name      Family_Name     DBDs entrezgene
##   1:      BAD93120.1       CGBP             CxxC  zf-CXXC         NA
##   2:      DUX1_HUMAN       DUX1      Homeodomain Homeobox         NA
##   3:      DUX3_HUMAN       DUX3      Homeodomain Homeobox         NA
##   4: ENSG00000031544      NR2E3 Nuclear receptor    zf-C4         NA
##   5: ENSG00000105663 AD000671.1             CxxC  zf-CXXC         NA
##  ---
## 109:     ZN735_HUMAN    ZNF735P          C2H2 ZF  zf-C2H2         NA
## 110:     ZN75C_HUMAN     ZNF75C          C2H2 ZF  zf-C2H2         NA
## 111:     ZN806_HUMAN     ZNF806          C2H2 ZF  zf-C2H2         NA
## 112:     ZN883_HUMAN     ZNF883          C2H2 ZF  zf-C2H2         NA
## 113:     ZNF73_HUMAN      ZNF73          C2H2 ZF  zf-C2H2         NA
##      refseq_peptide
##   1:             NA
##   2:             NA
##   3:             NA
##   4:             NA
##   5:             NA
##  ---
## 109:             NA
## 110:             NA
## 111:             NA
## 112:             NA
## 113:             NA
```

```
tf_class_ens = getBM(attributes=c("ensembl_gene_id","entrezgene"),
                     filters="ensembl_gene_id",values=tf_class[,ENSEMBL],mart=ensembl_h37)
tf_class_add = merge(tf_class,tf_class_ens,by.x="ENSEMBL",by.y="ensembl_gene_id",all.x=TRUE) #
# number unmapped
# jul 2016 (current version): 77:
# mar 2016: 77
# dec 2015: 75
# jul 2015: 75
# may 2015: 76
# mar 2015: 100
# dec 2014: 81
# oct 2014: 79
# aug 2014: 69
# GRCh37 (feb 2014): 52
# dec 2013: 52
# sep 2013: 51
# may 2012: 58
# may 2009: 704

tf_class_up = select(up,keys=tf_class_add[is.na(entrezgene),UNIPROT_HUMAN],columns=c("ENTREZ_GE

## Getting mapping data for Q7RTU5 ...  and P_ENTREZGENEID
## 'select()' returned 1:many mapping between keys and columns

tf_class_add2 = merge(tf_class,tf_class_up,by.x='UNIPROT_HUMAN',by.y='UNIPROTKB') # only get th

tf_class_cols = colnames(tf_class)
# combining
tf_class_comb = rbind(tf_class_add[!is.na(entrezgene),c(tf_class_cols,"entrezgene"),with=FALSE]
                      tf_class_add2[,entrezgene:=ENTREZ_GENE][,c(tf_class_cols,"entrezgene"),wi

# did we preserve all the genes in the orignal table
setequal(tf_class[,ENSEMBL],tf_class_comb[,ENSEMBL])

## [1] TRUE

# how many genes are unmapped?
tf_class_comb[is.na(entrezgene),]

##             ENSEMBL Superclass Class Family Subfamily    TRANSFAC UNIPROT
##  1:                           3   3.1  3.1.4   3.1.4.2                  NA
##  2:                           2   2.3  2.3.3   2.3.3.0                  NA
##  3: ENSG00000204532           2   2.3  2.3.3   2.3.3.5 PR000717273      NA
##  4:                           3   3.1  3.1.4   3.1.4.2                  NA
##  5: ENSG00000196081           2   2.3  2.3.3   2.3.3.0                  NA
##  6:                           2   2.3  2.3.3   2.3.3.7 PR000743170      NA
##  7:                           2   2.3  2.3.3   2.3.3.7                  NA
##  8: ENSG00000243660           2   2.3  2.3.2   2.3.2.2                  NA
```

```
##  9:                          2   2.3  2.3.3   2.3.3.0 PR000743178      NA
## 10:                          3   3.1  3.1.3   3.1.3.7 PR000016175      NA
## 11:                          4   4.1  4.1.6   4.1.6.2                  NA
## 12:                          1   1.2  1.2.6   1.2.6.5 PR000016403      NA
## 13:                          2   2.3  2.3.3   2.3.3.0                  NA
## 14:                          2   2.3  2.3.3   2.3.3.0 PR000699215      NA
## 15: ENSG00000235608          3   3.1  3.1.2  3.1.2.13 PR000681996      NA
## 16:                          2   2.3  2.3.3  2.3.3.18 PR000677833      NA
## 17: ENSG00000180438          3   3.1  3.1.3  3.1.3.26                  NA
## 18:                          2   2.3  2.3.3  2.3.3.18                  NA
## 19:                          2   2.3  2.3.2   2.3.2.2 PR000676407      NA
## 20:                          3   3.1  3.1.2  3.1.2.12 PR000677790      NA
## 21:                          3   3.1  3.1.3   3.1.3.7                  NA
## 22: ENSG00000160229          2   2.3  2.3.3   2.3.3.0 PR000677848      NA
## 23: ENSG00000214189          2   2.3  2.3.4   2.3.4.0 PR000677792      NA
## 24:                          1   1.2  1.2.2   1.2.2.2 PR000676776      NA
## 25: ENSG00000197701          2   2.3  2.3.3  2.3.3.77 PR000676948      NA
## 26: ENSG00000203326          2   2.3  2.3.3  2.3.3.27 PR000678173      NA
## 27:                          3   3.1  3.1.2  3.1.2.12 PR000678398      NA
## 28:                          2   2.3  2.3.3   2.3.3.6 PR000676789      NA
## 29:                          2   2.3  2.3.2   2.3.2.4 PR000677018      NA
## 30:                          2   2.3  2.3.3   2.3.3.0 PR000678396      NA
##             ENSEMBL Superclass Class Family Subfamily    TRANSFAC UNIPROT
##     UNIPROT_HUMAN UNIPROT_MOUSE UNIPROT_RAT entrezgene
##  1:        A6NDR6                                  NA
##  2:        A6NDX5                                  NA
##  3:        A6NGD5                                  NA
##  4:        A8K0S8                                  NA
##  5:        A8MTY0                                  NA
##  6:        A8MVS1                                  NA
##  7:        A8MWA4                                  NA
##  8:        B1APH4                                  NA
##  9:        O43830                                  NA
## 10:        O75505                                  NA
## 11:        P0CB48                                  NA
## 12:        P12525                                  NA
## 13:        P52743                                  NA
## 14:        Q14591        P15620      Q5FVP4        NA
## 15:        Q15270        G3UXB3      M0R5R8        NA
## 16:        Q15929                                  NA
## 17:        Q17RH7                                  NA
## 18:        Q49A33                                  NA
## 19:        Q5EBM4                                  NA
## 20:        Q6NSW7                                  NA
## 21:        Q6RFH8                                  NA
## 22:        Q6ZN08                                  NA
## 23:        Q6ZQV5                                  NA
```

```
## 24:        Q7RTU5        M0QWB7      F1LUI6         NA
## 25:        Q8IYB9                                   NA
## 26:        Q8N782                                   NA
## 27:        Q8N7R0                                   NA
## 28:        Q92670                                   NA
## 29:        Q9H963                                   NA
## 30:        Q9NSJ1                                   NA
##     UNIPROT_HUMAN UNIPROT_MOUSE UNIPROT_RAT entrezgene

#y = select(org.Hs.eg.db,keys=tf_class_add[is.na(entrezgene),UNIPROT_HUMAN],columns=c("ENTREZI
#z = select(org.Hs.eg.db,keys=tf_class[,ENSEMBL],columns=c("ENTREZID"),keytype="ENSEMBL")
#z_add = merge(tf_class,z,by.x="ENSEMBL",by.y="ENSEMBL",all.x=TRUE)
#zz = select(org.Hs.eg.db,keys=z_add[is.na(ENTREZID),UNIPROT_HUMAN],columns=c("ENTREZID"),keyt
```

```
par(mfrow=c(2,2))
orfeome_venn_list = list('ASUsent'=unique(hTF_annot4[,GeneID]),
                         'TFClass+CISBP'=unique(c(tf_class_comb[!is.na(entrezgene),entrezgene],
                                                  cisbp_comb[!is.na(entrezgene),entrezgene])),
                         'ORFeome'=unique(orfeome_table[!is.na(GeneID),GeneID]))

Vtf2 = Venn(orfeome_venn_list)
plot(Vtf2)

other_clones_list = list('hTF_lenti'=unique(hTF_table[,GeneID]),
                              'hExpr'=unique(hExpr_table[,GeneID]),
                              'BC1000'=unique(bc1000_table[,GeneID]),
                              'PSIBiol'=unique(psibiol_table[,GeneID]),
                              'PSI'=unique(psi_table[,GeneID]))

for (i in seq_along(other_clones_list)) {
    li = other_clones_list[i]
    vl = c(orfeome_venn_list,li)
    print(names(vl))
    Vtf3 = Venn(vl)
    plot(Vtf3,type="ellipses")
}
```

```
## [1] "ASUsent"       "TFClass+CISBP" "ORFeome"       "hTF_lenti"
```

```
## Warning in compute.E4(V, doWeights):  Cant do a weighted E4
```

```
## [1] "ASUsent"       "TFClass+CISBP" "ORFeome"       "hExpr"
```

```
## Warning in compute.E4(V, doWeights):  Cant do a weighted E4
```

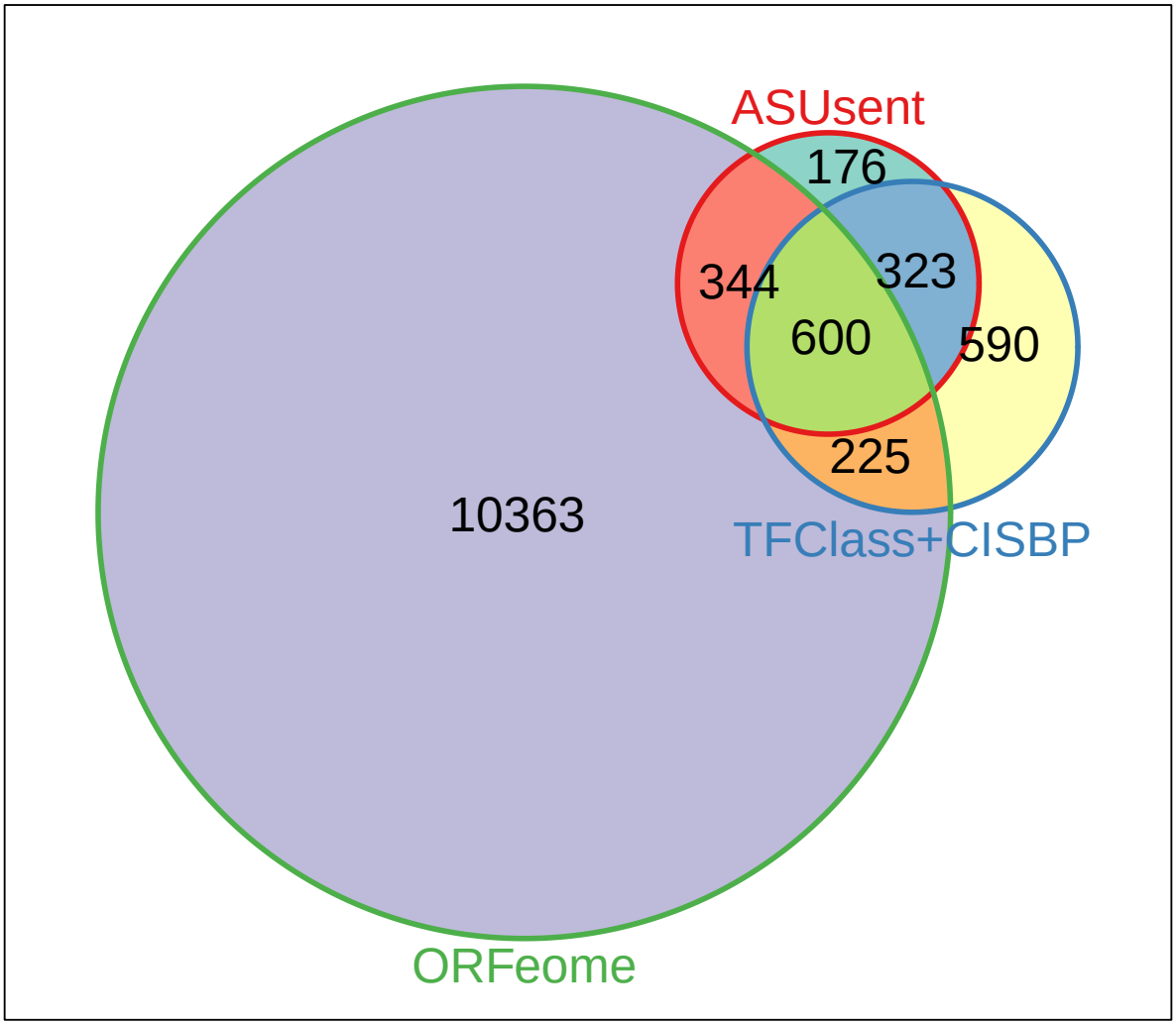
```
## [1] "ASUsent"       "TFClass+CISBP" "ORFeome"       "BC1000"
```

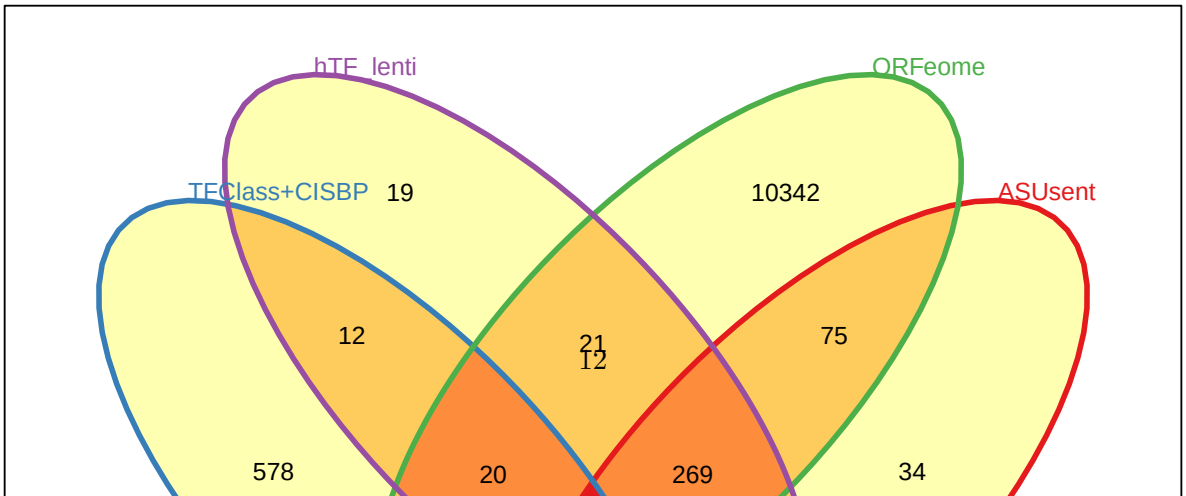
```
## Warning in compute.E4(V, doWeights):  Cant do a weighted E4
```

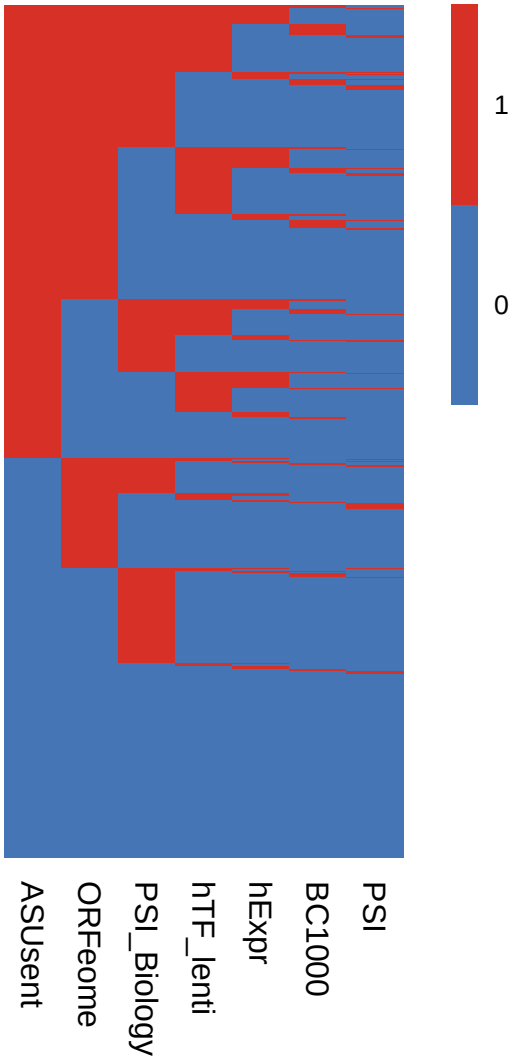
```
## [1] "ASUsent"       "TFClass+CISBP" "ORFeome"       "PSIBiol"

## Warning in compute.E4(V, doWeights):  Cant do a weighted E4

## [1] "ASUsent"       "TFClass+CISBP" "ORFeome"       "PSI"

## Warning in compute.E4(V, doWeights):  Cant do a weighted E4
```

The clones sent by ASU ("ASUsent" covers about 70% of the TFs in TFClass+CISBP annotation. Another 200 will be covered by the ORFeome collection that they have promised to send (looks like most are homeoboxes and zinc fingers). The PSI:Biology set is the collection that will increase the coverage the most (180 genes; most are zinc fingers).

```
tftotal_eg = unique(c(tf_class_comb[!is.na(entrezgene),entrezgene],cisbp_comb[!is.na(entrezgene
tftotal_eg2 = unique(c(tf_class_ens[,'entrezgene'],cisbp_ens[,'entrezgene']))

tftotal_mat = data.table(entrezgene=tftotal_eg,
                         ASUsent=tftotal_eg %in% hTF_annot4[,GeneID],
                         ORFeome=tftotal_eg %in% orfeome_table[,GeneID],
                         hTF_lenti=tftotal_eg %in% hTF_table[,GeneID],
                         hExpr=tftotal_eg %in% hExpr_table[,GeneID],
                         BC1000=tftotal_eg %in% bc1000_table[,GeneID],
                         PSI_Biology=tftotal_eg %in% psibiol_table[,GeneID],
                         PSI=tftotal_eg %in% psi_table[,GeneID])

tftotal_mat = tftotal_mat[with(tftotal_mat, order(-ASUsent,-ORFeome,-PSI_Biology,-hTF_lenti,-hE

#pdf(paste0(prefix.string,'TFClass_DNASU.pdf'),width=3,height=6)
aheatmap(data.matrix(tftotal_mat[,list(ASUsent,ORFeome,PSI_Biology,hTF_lenti,hExpr,BC1000,PSI)]
         Colv=NA,Rowv=NA,labRow=NA)
#dev.off()
```


ASUsent
176
344
323
600
590
225
10363
TFClass+CISBP
ORFeome


hTF_lenti
ORFeome
TFClass+CISBP
19
10342
ASUsent
12
21
12
75
578
20
269
34




```
tftotal_psi = tftotal_mat[entrezgene %in% tftotal_mat[!ASUsent & !ORFeome & PSI_Biology,entrezg

tftotal_psi_annot = getBM(attributes=c("entrezgene","hgnc_symbol","description"),
                          filters="entrezgene",values=tftotal_psi[,entrezgene],
                          mart=ensembl_h37)

psibiol_tf = psibiol_table[GeneID %in% tftotal_psi[,entrezgene],]

write.table(psibiol_tf,
            paste0(rdata_prefix,'psibiol_tf.txt'),sep='\t',quote=FALSE,col.names=TRUE,row.names
```

4 Find the 200+ clones to request from ASU

Goal is to get as many as possible, based on mappings from Ensembl Gene ID, Entrez Gene ID, RefSeq peptide and UniProt

```
orfeome_tftotal = orfeome_table[,GeneID_TF:=GeneID %in% tftotal_eg]

# get Gene Symbol and alias
tftotal_symb = select(org.Hs.eg.db,keys=tftotal_eg,
                      columns=c("SYMBOL","ALIAS"),keytype="ENTREZID")

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

orfeome_tftotal = orfeome_tftotal[,GeneSymbol_TF:=GeneSymbol %in% na.omit(tftotal_symb[,'ALIAS

orfeome_tftotal = orfeome_tftotal[,`:=`(June2016=CloneID %in% hTF_clones[,DNASU_CloneID],
                                        June2016G=GeneID %in% hTF_clones[,GeneID])]

# all the ORFeome TF clones that we haven't got yet
orfeome_miss1 = orfeome_tftotal[(GeneID_TF==TRUE | GeneSymbol_TF==TRUE) & June2016==FALSE,]

write.table(orfeome_miss1,paste0(rdata_prefix,'ORFeome8_1_addTF1.tsv'),
            sep='\t',quote=FALSE,row.names=FALSE,col.names=TRUE)

# all the ORFeome TF clones for which the TF genes we haven't got yet
orfeome_miss2 = orfeome_tftotal[(GeneID_TF==TRUE | GeneSymbol_TF==TRUE) & June2016G==FALSE & Ju

write.table(orfeome_miss2,paste0(rdata_prefix,'ORFeome8_1_addTF2.tsv'),
            sep='\t',quote=FALSE,row.names=FALSE,col.names=TRUE)

write.table(orfeome_miss2[,!c("GeneID_TF","GeneSymbol_TF","June2016","June2016G"),with=FALSE],
            paste0(rdata_prefix,'ORFeome8_1_addTF3.tsv'),
            sep='\t',quote=FALSE,row.names=FALSE,col.names=TRUE)
```

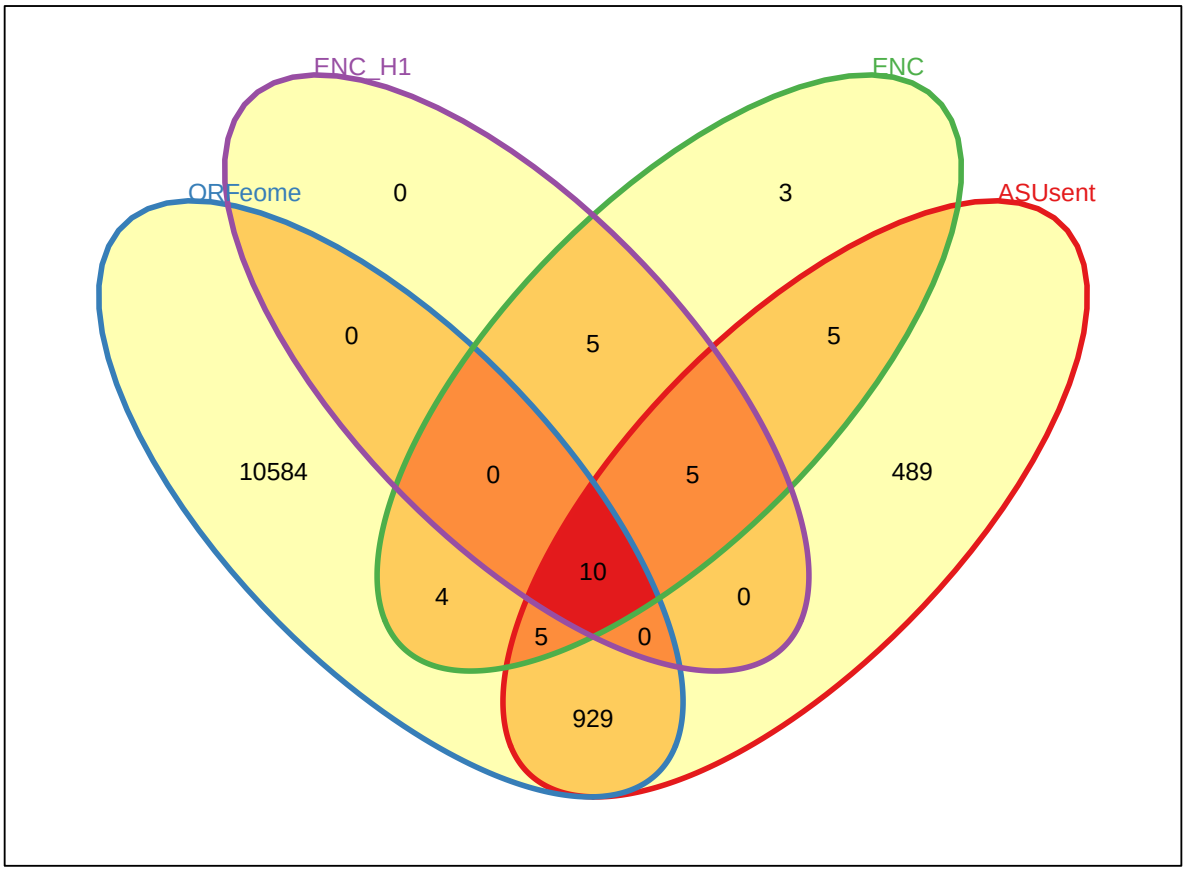
5 Overlap with ENCODE DREAM challenge TFs

```
enc_ens = getBM(attributes=c("entrezgene","hgnc_symbol","description"),
                filters="hgnc_symbol",values=enc_tfs[[1]],
                mart=ensembl_h37)
enc_ens = within(enc_ens,{ entrezgene=as.character(entrezgene)})
enc_ens = merge(enc_tfs,enc_ens,by.x="V1",by.y="hgnc_symbol",all.x=TRUE)
print(dim(enc_ens))

## [1] 37  3

enc_ens = enc_ens[,H1:=V1 %in% enc_h1_tfs[[1]]]
enc_ens[,.N,by="H1"]
```

```
##       H1  N
## 1: FALSE 17
## 2:  TRUE 20
```

```
enc_venn_list = list('ASUsent'=unique(hTF_annot4[,GeneID]),
                     'ORFeome'=unique(orfeome_table[!is.na(GeneID),GeneID]),
                     'ENC'=unique(enc_ens[,entrezgene]),
                     'ENC_H1'=unique(enc_ens[H1==TRUE,entrezgene]))

Vtf4 = Venn(enc_venn_list)
plot(Vtf4,type="ellipses")
```

```
## Warning in compute.E4(V, doWeights):  Cant do a weighted E4
```




```
enctf_mat = data.frame(H1=enc_ens[,H1],
                       ASUsent=enc_ens[,entrezgene] %in% hTF_annot4[,GeneID],
                       ORFeome=enc_ens[,entrezgene] %in% orfeome_table[,GeneID]
                       )
rownames(enctf_mat) = enc_ens[,V1]

enctf_mat = enctf_mat[with(enctf_mat,order(-H1,-ASUsent,-ORFeome)),]

nmf.options(grid.patch=TRUE)# this gets rid of the first blank page in the PDF
pdf(paste0(prefix.string,'ENC_TFs.pdf'),width=2,height=4)
aheatmap(data.matrix(enctf_mat),cellwidth=5,cellheight=5,
         Colv=NA,Rowv=NA,border_color="grey50")
```

```
dev.off()

## pdf
##   2
```

6 Session info

```
sessionInfo()

## R version 3.2.2 (2015-08-14)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: CentOS release 6.8 (Final)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] RColorBrewer_1.1-2    UniProt.ws_2.10.4     RCurl_1.95-4.7
##  [4] bitops_1.0-6          RSQLite_1.0.0         DBI_0.4-1
##  [7] NMF_0.22              Biobase_2.31.0        BiocGenerics_0.17.1
## [10] cluster_2.0.3         rngtools_1.2.4        pkgmaker_0.25.8
## [13] registry_0.3          Vennerable_3.1.0.9000 R.utils_2.2.0
## [16] R.oo_1.19.0           R.methodsS3_1.7.0     tidyr_0.5.1
## [19] stringr_1.0.0         plyr_1.8.3            ggplot2_2.1.0
## [22] doMC_1.3.4            iterators_1.0.8       foreach_1.4.3
## [25] data.table_1.9.6      biomaRt_2.26.1        knitr_1.12.3
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.3          highr_0.5.1          formatR_1.2.1
##  [4] tools_3.2.2          dendextend_1.1.8     digest_0.6.9
##  [7] gridBase_0.4-7       evaluate_0.8         tibble_1.1
## [10] gtable_0.2.0         lattice_0.20-33      graph_1.48.0
## [13] dplyr_0.5.0          S4Vectors_0.8.11     IRanges_2.4.6
## [16] stats4_3.2.2         grid_3.2.2           reshape_0.8.5
## [19] R6_2.1.2             AnnotationDbi_1.32.3 XML_3.98-1.3
## [22] RBGL_1.46.0          reshape2_1.4.1       whisker_0.3-2
```

```
## [25] magrittr_1.5         scales_0.4.0         codetools_0.2-14
## [28] assertthat_0.1       xtable_1.8-2         colorspace_1.2-6
## [31] stringi_1.0-1        doParallel_1.0.10    lazyeval_0.2.0
## [34] munsell_0.4.3        chron_2.3-47
```