```
rm(list=ls())
cat('\014')
t0<-proc.time()
require(dendextend)
require(data.table)
require(stringdist)
require(parallel)
require(foreach)
require(doParallel)


# Load strings
setwd('/Users/bambrose/Dropbox/GitHub/knowledge-survival')
load('hd1.RData')
#hd<-sample(hd,1000)
sl<-lapply(hd,function(x) unlist(labels(x))) # list of string sets
save(sl,file='sl.RData')
rm(hd)
sll<-sapply(sl,length)
ncom<-sll*(sll-1)/2 # eventual number of comparisons
tcom<-sum(ncom)
cat('Total number of comparisons is ',format(tcom,big.interval=3,big.mark=','),'.',sep='')


sdt.f<-function(strl=sl,ix=i){
        # Define column inspectors
        flist<-list(
                py=function(x) grep('^[0-9]{4}$',x)
                ,v=function(x) grep('^V([0-9]+|[IVXLC]+)$',x) # numeric or roman
                ,ch=function(x) grep('^PCHR?([0-9]+|[IVXLC]+)$',x) # numeric or roman
                ,p=function(x) grep('^P([A-Z]{0,3}[0-9]+|[IVXLC]+)[A-Z]?$',x) # numeric or roman
        )
        # initalize set
        nam<-c('au','py','so','m','d','v','p','ch','t')
        y<-data.table(matrix(NA_character_,ncol=length(nam)-1,nrow=length(strl[[ix]])))
        y[,'t':=NA_real_]
        setnames(y,nam)
        for(j in 1:length(strl[[ix]])) {
                pyi<-NULL
                s<-strsplit(strl[[ix]][[j]],split=', ')[[1]]
                m<-regexpr(' ((16)|(17)|(18)|(19)|(20))[0-9]{2}$',s)
                wm<-which(m>-1)
                if(length(wm)&all(!grepl('^[0-9]{4}$',s))) { # if year is included at end of source field and not
otherwise extracted, assume publication year
                        ns<-regmatches(s,m,invert=T)
                        ns[[wm]][2]<-sub(' ','',regmatches(s,m))
                        ns[[wm]]<-rev(ns[[wm]])
                        s<-unlist(ns) # split so and switch year and source to match common au-py-so format
                }
                for(k in names(flist)){ # something is wrong, source gets treated as author
                        si<-flist[[k]](s)
                        if(!length(s[si])) next
                        if(k=='py') pyi<-si
                        y[j,(k):=s[si]]
                        s<-s[-si]
                }
                if(!is.null(pyi)&pyi!=1) {
                        y[j,'au':=paste(s[1:(pyi-1)])]
                        s<-s[-(1:(pyi-1))]
                }
                if(!length(s)) next
                s<-paste(s,collapse=', ')
                if(grepl(' [0-1][0-9][0-3][0-9]$',s)) { # for periodicals with month/day code
                        nc<-nchar(s)
                        y[j,'d':=substr(s,nc-1,nc)]
                        y[j,'m':=substr(s,nc-3,nc-2)]

y[j,'t':=as.numeric(strptime(apply(y[j,list(py,m,d)],1,paste,collapse=''),format='%Y%m%d'))/24/60/60]
                        s<-sub(' +$','',sub('(.+) +[0-1][0-9][0-3][0-9]','\\1',s))
                }
                if(grepl(', P\\[',s)){ # for P including multiple pages
                        s<-strsplit(s,', P\\[')[[1]]
                        y[j,'p':=paste('P[',s[2],sep='')]
                        s<-s[1]
                }
                y[j,'so':=s]
        }
        y[,k:=ix]
        y[,ij:=1:nrow(y)]
        y
```

```
}
sdt.f(ix=1)

t1<-proc.time()
cl <- makeCluster(detectCores() )
registerDoParallel(cl, cores = detectCores() )
sdt <- foreach(i = 1:length(sl),.packages = c("data.table"),.inorder=F) %dopar% {
sdt.f(ix=i)
}
stopCluster(cl)
t2<-proc.time()
round((t2-t1)/60,3)

sdt<-rbindlist(sdt)
sdt[,py:=as.integer(py)]
sdt[!is.na(v),nv:=as.integer(gsub('[A-Z]','',v))]
suppressWarnings(sdt[!is.na(p),np:=sapply(
                strsplit(p,'(, )|\\[|\\]')
                ,function(x) {
                        y<-as.integer(na.omit(as.integer(gsub('[A-Z]',"",x))))
                        z<-as.integer(na.omit(as.roman(sub('P','',x))))
                        x<-c(y,z)
                        if(!length(x)) x<-NA_integer_
                        unique(x)
                }
)])
suppressWarnings(sdt[!is.na(ch),nch:=unname(sapply(gsub('^PCHR?','',ch)
        ,function(x) ifelse(
                is.na(as.integer(x))
                ,as.integer(as.roman(x))
                ,as.integer(x)
        )
))])

setkey(sdt,k,ij)
save(sdt,file='sdt.RData')

### Features ###
setwd('/Users/bambrose/Dropbox/GitHub/knowledge-survival')
load('sdt.RData')
load('sl.RData')
sll<-sapply(sl,length)
library(data.table)
library(stringdist)
ncom<-sll*(sll-1)/2 # eventual number of comparisons

t3<-proc.time()
# index of which table and which pair of rows for which to calculate features
comps<-data.table(do.call(rbind,lapply(sll,function(x) do.call(rbind,combn(1:x,m=2,simplify=F)))))
setnames(comps,c('i','j'))
comps[,'k':=rep(1:length(sl),ncom)]
setcolorder(comps,c(3,1,2))
setkey(comps,k)

# measure features
#rm(list=ls()[!ls()%in%c('sl','sll','sdt','t0','t1','t2','t3','ncom','tcom','comps')])
ix.f<-function(k,ij) k[ij]
comps[,jw:=stringdist(mapply(FUN=ix.f,k=sl[comps$k],ij=comps$i),mapply(FUN=ix.f,k=sl[comps$k],ij=comps$j),method='jw',p

comps[,cpau:=as.integer(grepl('^\\*',sdt[list(comps$k,comps$i),au])&grepl('^\\*',sdt[list(comps$k,comps$j),au]))]
comps[is.na(sdt[list(comps$k,comps$i),au])|is.na(sdt[list(comps$k,comps$j),au]),cpau:=NA]
comps[,jwau:=stringdist(sdt[list(comps$k,comps$i),au],sdt[list(comps$k,comps$j),au],method='jw',p=.1)]
sl[sample(unique(comps$k[is.na(comps$jwau)]),3)]
comps[,dfpy:=abs(sdt[list(comps$k,comps$i),py]-sdt[list(comps$k,comps$j),py])]
comps[,dfpy2:=dfpy^2]
comps[,jwso:=stringdist(sdt[list(comps$k,comps$i),so],sdt[list(comps$k,comps$j),so],method='jw',p=.1)]
sl[sample(unique(comps$k[is.na(comps$jwso)]),3)]
comps[,dfv:=abs(sdt[list(comps$k,comps$i),nv]-sdt[list(comps$k,comps$j),nv])]
sl[sample(unique(comps$k[is.na(comps$dfv)]),3)]
comps[,mfv:=apply(data.frame(sdt[list(comps$k,comps$i),nv],sdt[list(comps$k,comps$j),nv]),1,FUN=mean)]
comps[,dfp:=mapply(FUN=
function(x,y) {
        if(is.null(x)|is.null(y)) {u<-NA} else {u<-min(abs(unlist(lapply(x,function(z) z-unlist(y)))))}
        u
}
,x=sdt[list(comps$k,comps$i),np]
,y=sdt[list(comps$k,comps$j),np])]
sl[sample(unique(comps$k[is.na(comps$dfp)]),3)]
comps[,mfp:=mapply(FUN=
function(x,y) {
        if(is.null(x)|is.null(y)) {u<-NA} else {u<-round(min(unlist(lapply(x,function(z)
```

```
mean(c(z,min(unlist(y))))))))}
        u
}
,x=sdt[list(comps$k,comps$i),np]
,y=sdt[list(comps$k,comps$j),np])]
comps[,dfch:=abs(sdt[list(comps$k,comps$i),nch]-sdt[list(comps$k,comps$j),nch])]
sl[sample(unique(comps$k[is.na(comps$dfch)]),3)]
comps[,dft:=abs(sdt[list(comps$k,comps$i),t]-sdt[list(comps$k,comps$j),t])]
sl[sample(unique(comps$k[is.na(comps$dft)]),3)]
comps[,pyxvxp:=mapply(FUN=function(w,x,y,z) prod(na.omit(c(w,x,y,z))),w=dfpy,x=dfv,y=dfp,z=dfch)]
comps[,prob:=1/.N,by=k]
t4<-proc.time()
round((t4-t3)/60,3)

save(comps,file='comps.RData')

### which NA combos to model ###
setwd('/Users/bambrose/Dropbox/GitHub/knowledge-survival')
t5<-proc.time()
load('sdt.RData')
load('comps.RData')
library(data.table)

compl<-comps[,list(
        jwau=factor(!is.na(jwau),levels=c('FALSE','TRUE'))
        ,dfpy=factor(!is.na(dfpy),levels=c('FALSE','TRUE'))
        ,jwso=factor(!is.na(jwso),levels=c('FALSE','TRUE'))
        ,dfv=factor(!is.na(dfv),levels=c('FALSE','TRUE'))
        ,dfp=factor(!is.na(dfp),levels=c('FALSE','TRUE'))
        ,dfch=factor(!is.na(dfch),levels=c('FALSE','TRUE'))
        ,dft=factor(!is.na(dft),levels=c('FALSE','TRUE'))
)] #complete data

t<-table(compl)
w<-which(t>0,arr.ind=T)
tw<-w-1
t<-cbind(tw,freq=t[w],'perc'=round(prop.table(t)[w]*100,3))
t<-t[order(t[,'freq'],decreasing=T),]
t<-suppressWarnings(data.frame(t,cumul=cumsum(t[,'perc'])))
row.names(t)<-NULL
t
t[do.call(order,c(t[!names(t)%in%c('freq','perc','cumul')],decreasing=T)),]

lt<-t(!!t[,!colnames(t)%in%c('freq','perc','cumul')])
t5<-proc.time()
round((t5-t4)/60,3)

lcompl<-data.frame(t(compl))
comps[,miss:=sapply(lcompl,function(x) which(!apply(as.logical(x)-lt,2,any)))]
rm(lcompl)
t6<-proc.time()
round((t6-t5)/60,3)

setkey(comps,k)
setkey(comps,miss)

samp.batch<-function(x){
for(h in x){ #
        samp<-comps[list(h),list(k,i,j,miss)]
        samp<-samp[sample(1:nrow(samp),3,replace=T)]
        print(samp)
        cat('\n')
        for(g in 1:nrow(samp)){
        print(sdt[list(k=c(rep(samp$k[g],2)),ij=unlist(samp[g,list(i,j)])),list(au,py,so,m,d,v,p,ch,k,ij)])
        cat('\n')
        }
}}
samp.batch(c(17,19)) #1:nrow(t)

## results of analyzing: /Users/bambrose/Dropbox/Summer 2015/Diss/ML_partitions.rtf
setkey(comps,miss)
comps[list(1,13),batch:='A']
comps[list(2,15),batch:='B']
comps[list(3),batch:='C']
comps[list(4,20,21),batch:='D']
comps[list(5),batch:='E']
comps[list(7,8),batch:='F']
comps[list(6,18),batch:='G']
comps[list(9,17),batch:='H']
comps[list(10),batch:='I']
comps[list(12,16,14),batch:='J']
```

```
comps[list(11,22),batch:='K']

wtab<-comps[,list(w=sum(prob)),by=batch]
wtab<-wtab[order(wtab$w,decreasing=T),]
wtab<-wtab[c((1:nrow(wtab))[-which(is.na(wtab$batch))],which(is.na(wtab$batch))),]
wtab$cumul<-round(cumsum(prop.table(wtab$w))*100,2)
wtab
comps[,batch:=factor(comps$batch)]

load('sl.RData')
ix.f<-function(k,ij) k[ij]
comps[,str:=paste(mapply(FUN=ix.f,k=sl[comps$k],ij=comps$i),mapply(FUN=ix.f,k=sl[comps$k],ij=comps$j),sep='\r')]

setkey(comps,batch)
save(comps,file='comps.RData')

### SuperLearner
### Machine Learning
if(F){
        install.packages('devtools')
        library('devtools')
        install_github('hadley/stringr')
        install_github('ecpolley/Superlearner')
        install_github('ledell/subsemble')
}
setwd('/Users/bambrose/Dropbox/GitHub/knowledge-survival')
library(data.table)
load('comps.RData')
frame<-comps[c('A','B','C','D','E')] # sampling frame, 97.34% of data
rej<-comps[!c('A','B','C','D','E')] # rejected for limited info

frame[,train:=F]
frame[,match:=NA]
frame[,ix:=1:nrow(frame)]
samp.f<-function(x,y){
        set.seed(12345)
        sample(x,500,prob=y)
}
samp<-frame[,samp.f(x=ix,y=prob),by=batch]$V1
frame[samp,train:=T]
save(samp,file='samp.RData')

frame[,test:=F]
samp.f<-function(x,y){
        set.seed(54321)
        sample(x,100,prob=y)
}
samp.test<-frame[,samp.f(x=ix,y=prob),by=batch]$V1
frame[samp.test,test:=T]
save(samp.test,file='samp.test.RData')
save(frame,file='frame.RData')

### Define sample for hand coding
write.csv(frame[frame$train,],file='frame.train.csv',na='',row.names=F) # convert to .xls to protect hand coding
write.csv(frame[frame$test,],file='frame.test.csv',na='',row.names=F) # convert to .xls to protect hand coding

### Import hand codes
library(data.table)
hand<-data.table(read.csv('frame.train.csv'))
hand[,match:=as.integer(!is.na(match))]
hand[,mean(match),by=batch]
setkey(hand,batch)

fitter<-function(b,hand){
        require(data.table)
        require(subsemble)
        require(cvAUC)
A<-hand[b]
cols<-apply(A,2,FUN=function(x) all(!is.na(x)))
cols<-cols[!names(cols)%in%c('k','i','j','prob','miss','batch','train','ix','str')&cols]
A<-A[,names(cols),with=F]
if(any(duplicated(t(A)))) A<-A[,!duplicated(t(A)),with=F]


fit<-subsemble(
        x=A[,!'match',with=F]
        ,y=A$match
        ,family='binomial'
        ,parallel='multicore'
        ,learner=c("SL.randomForest","SL.glmnet")
        ,metalearner="SL.glm"
```

```
        ,subsets=1
)
auc <- AUC(predictions = fit$pred, labels = A$match)
print(hand)
print(b)
print(auc)  # Test set AUC is: 0.937
ret<-list(fit,auc)
}

t7<-proc.time()
lb<-list()
for(i in levels(hand$batch)) lb[[i]]<-fitter(i,hand)
t8<-proc.time()
round((t8-t7)/60,3)
save(lb,file='lb.RData')
```