Untitled

Kijong Yi

201859

# TODO: Add comment  
#   
# Author: mg14  
###############################################################################  
  
library(Rsamtools)  
  
vcfPath <- '~/pcawg/final/final\_consensus\_12oct\_passonly/snv\_mnv'  
basePath <- '~/pcawg/dp/20170129\_dpclust\_finalConsensusCopynum\_levels\_a\_b\_c\_d'  
dpPath <- paste0('~/pcawg/final/consensus\_subclonal\_reconstruction\_20170325')  
#CANCERGENES <- read.table('~/pcawg/ref/cancer\_genes.txt')$V1  
purityPloidy <- read.table( '~/pcawg/final/consensus.20170218.purity.ploidy.txt', header=TRUE, row.names=1)  
#colnames(purityPloidy) <- c("purity","ploidy")  
cnPath <- paste0(basePath,'/4\_copynumber/')  
bbPath <- paste0(basePath,'/4\_copynumber/')  
  
allGender <- read.table('~/pcawg/gender/2016\_12\_09\_inferred\_sex\_all\_samples.txt', header=TRUE, sep='\t')  
allGender <- allGender[!duplicated(allGender$tumourid) & allGender$tumourid != 'tumourid',]  
#write.table(gender,'~/pcawg/gender/2016\_12\_09\_inferred\_sex\_all\_samples\_CORRECTED\_MG.txt', row.names=FALSE, col.names=TRUE, sep='\t', quote=FALSE)  
rownames(allGender) <- allGender$tumourid  
  
addTNC <- function(vcf){   
 r = "/lustre/scratch112/sanger/cgppipe/PanCancerReference/genome.fa.gz" #meta(header(v))["reference",]  
 if(!"TNC" %in% rownames(header(vcf)@header$INFO)){  
 tnc=scanFa(file=r, resize(granges(vcf), 3,fix="center"))  
 i = header(vcf)@header$INFO  
 exptData(vcf)$header@header$INFO <- rbind(i, DataFrame(Number=1,Type="String",Description="Trinucleotide context", row.names="TNC"))  
 info(vcf)$TNC <- as.character(tnc)  
 }  
 return(vcf)  
}  
  
dpFiles <- dir(dpPath, pattern="\_subclonal\_structure.txt", recursive=TRUE)  
  
bbFiles <- dir(bbPath, pattern="\_segments.txt", recursive=TRUE)  
  
wccTable <- read.table("~/pcawg/final/wcc\_consensus\_values\_9\_12.tsv", header=TRUE, sep='\t')  
d <- data.frame(cluster=wccTable$sc\_n, n\_ssms=wccTable$consensus\_mutation\_number, proportion = round(wccTable$consensus\_cluster\_cp,3))  
d[d$cluster==1,"proportion"] <- wccTable[d$cluster==1,"purity"]  
wccClusters <- split(d, wccTable$sid)  
wccPurity <- d[d$cluster==1,"proportion"]   
#plot(wccPurity, purityPloidy[,"purity"])  
  
loadClusters <- function(ID){  
 file <- paste0(dpPath,"/",grep(paste0(ID,"[[:punct:]]"), dpFiles, value=TRUE, perl=TRUE))  
 if(grepl(".gz", file))  
 file <- gzfile(file)  
 read.table(file, header=TRUE, sep="\t")  
}  
  
loadConsensusClusters <- function(ID){  
 file <- paste0(dpPath,"/",grep(paste0(ID,"[[:punct:]]"), dpFiles, value=TRUE, perl=TRUE))  
 if(grepl(".gz", file))  
 file <- gzfile(file)  
 read.table(file, header=TRUE, sep="\t")  
}  
  
consensusClustersToOld <- function(clusters){  
 data.frame(cluster=clusters$cluster, n\_ssms=clusters$n\_snvs, proportion=clusters$fraction\_total\_cells)  
}  
  
parseRegion <- function(regions){  
 chr <- regmatches(regions, regexpr("^.+(?=:)",regions,perl=TRUE))  
 start <- as.numeric(regmatches(regions, regexpr("(?<=:)[0-9]+(?=-)",regions,perl=TRUE)))  
 end <- as.numeric(regmatches(regions, regexpr("(?<=-)[0-9]+$",regions,perl=TRUE)))  
 data.frame(chr,start,end)  
}  
  
loadCn <- function(ID){  
 file <- paste0(cnPath, "/",ID,"\_timingsMle.txt")  
 tab <- read.table(file, header=TRUE, sep=" ")  
 reg <- parseRegion(tab$ascatId)  
 GRanges(tab$chr, IRanges(reg$start,reg$end), strand="\*", tab[-1])  
}  
  
loadBB <- function(ID){  
 t <- try({  
 file <- grep(paste0(ID,"[[:punct:]]"), dir(bbPath, pattern="segments.txt", recursive=TRUE, full.names=TRUE), value=TRUE)  
 if(grepl(".gz", file))  
 file <- gzfile(file)  
 tab <- read.table(file, header=TRUE, sep='\t')  
 GRanges(tab$chromosome, IRanges(tab$start, tab$end), strand="\*", tab[-3:-1])  
 })  
 if(class(t)=='try-error') GRanges(copy\_number=numeric(), major\_cn=numeric(), minor\_cn=numeric(), clonal\_frequency=numeric()) else t  
}  
  
loadConsensusCNA <- function(ID, purity=1, path="/lustre/scratch112/sanger/cgppipe/PanCancerDownloads/workspace/mg14/final/consensus.20170119.somatic.cna.annotated"){  
 file <- grep(paste0(ID,"[[:punct:]]"), dir(path, pattern="cna.annotated.txt", recursive=TRUE, full.names=TRUE), value=TRUE)  
 if(grepl(".gz", file))  
 file <- gzfile(file)  
 tab <- read.table(file, header=TRUE, sep='\t')  
 subclonalIndex <- !is.na(tab$total\_cn) & !is.na(tab$battenberg\_nMaj2\_A) & !is.na(tab$battenberg\_nMin2\_A) & !is.na(tab$battenberg\_frac2\_A) & (tab$battenberg\_nMaj1\_A == tab$major\_cn & tab$battenberg\_nMin1\_A == tab$minor\_cn | tab$battenberg\_nMaj2\_A == tab$major\_cn & tab$battenberg\_nMin2\_A == tab$minor\_cn)  
 ix <- c(1:nrow(tab), which(subclonalIndex))  
 gr <- GRanges(tab$chromosome, IRanges(tab$start, tab$end), strand="\*", clonal\_frequency=purity, tab[-3:-1])[ix]  
 if(any(subclonalIndex)){  
 gr$clonal\_frequency[which(subclonalIndex)] <- tab$battenberg\_frac1\_A[subclonalIndex] \* purity  
 gr$major\_cn[which(subclonalIndex)] <- tab$battenberg\_nMaj1\_A[subclonalIndex]  
 gr$minor\_cn[which(subclonalIndex)] <- tab$battenberg\_nMin1\_A[subclonalIndex]  
 gr$total\_cn[which(subclonalIndex)] <- tab$battenberg\_nMaj1\_A[subclonalIndex] + tab$battenberg\_nMin1\_A[subclonalIndex]  
 gr$clonal\_frequency[-(1:nrow(tab))] <- tab$battenberg\_frac2\_A[subclonalIndex] \* purity  
 gr$total\_cn[-(1:nrow(tab))] <- tab$battenberg\_nMaj2\_A[subclonalIndex] + tab$battenberg\_nMin2\_A[subclonalIndex]  
 gr$major\_cn[-(1:nrow(tab))] <- tab$battenberg\_nMaj2\_A[subclonalIndex]  
 gr$minor\_cn[-(1:nrow(tab))] <- tab$battenberg\_nMin2\_A[subclonalIndex]  
 }  
 seqlevels(gr) <- c(1:22, "X","Y")  
 sort(gr)  
}  
  
refFile = "/lustre/scratch112/sanger/cgppipe/PanCancerReference/genome.fa.gz" #meta(header(v))["reference",]  
refLengths <- scanFaIndex(file=refFile)  
chrOffset <- cumsum(c(0,as.numeric(width(refLengths))))  
names(chrOffset) <- c(seqlevels(refLengths), "NA")  
  
averagePloidy <- function(bb) {  
 c <- if(!is.null(bb$copy\_number)) bb$copy\_number else bb$total\_cn  
 sum(width(bb) \* c \* bb$clonal\_frequency, na.rm=TRUE) / sum(width(bb) \* bb$clonal\_frequency, na.rm=TRUE)  
}  
  
averageEvenPloidy <- function(bb){  
 sum(width(bb) \* (!bb$major\_cn %% 2 & !bb$minor\_cn %% 2) \* bb$clonal\_frequency, na.rm=TRUE) / sum(width(bb) \* bb$clonal\_frequency, na.rm=TRUE)  
}  
  
getTumorCounts <- function(vcf){  
 sapply(grep("(F|R).Z", names(geno(vcf)), value=TRUE), function(i) geno(vcf)[[i]][,"TUMOUR"])  
}  
  
getTumorDepth <- function(vcf){  
 if("t\_alt\_count" %in% colnames(info(vcf))){ ## consensus data, snv and indel  
 info(vcf)$t\_alt\_count + info(vcf)$t\_ref\_count  
 }else{ ## older data  
 if("FAZ" %in% rownames(geno(header(vcf)))){  
 rowSums(getTumorCounts(vcf))  
 }else{  
 geno(vcf)$DEP[,2]  
 }  
 }  
}  
  
getAltCount <- function(vcf){  
 if("t\_alt\_count" %in% colnames(info(vcf))){ ## consensus data, snv and indel  
 info(vcf)$t\_alt\_count  
 }else{ ## older formats  
 if("FAZ" %in% rownames(geno(header(vcf)))){ ## ie subs  
 c <- getTumorCounts(vcf)  
 t <- c[,1:4] + c[,5:8]  
 colnames(t) <- substring(colnames(t),2,2)  
 a <- as.character(unlist(alt(vcf)))  
 a[!a%in%c('A','T','C','G')] <- NA  
 sapply(seq\_along(a), function(i) if(is.na(a[i])) NA else t[i, a[i]])  
 }  
 else{ ## ie indel  
 #(geno(vcf)$PP + geno(vcf)$NP + geno(vcf)$PB + geno(vcf)$NB)[,"TUMOUR"]  
 geno(vcf)$MTR[,2]  
 }  
 }  
}  
  
  
  
mergeClusters <- function(clusters, deltaFreq=0.05){  
 if(nrow(clusters) <= 1) return(clusters)  
 h <- hclust(dist(clusters$proportion), members=clusters$n\_ssms)  
 ct <- cutree(h, h=deltaFreq)  
 cp <- as.matrix(cophenetic(h))  
 Reduce("rbind",lapply(unique(ct), function(i) {  
 n\_ssms <- sum(clusters$n\_ssms[ct==i])  
 w <- max(cp[ct==i,ct==i])  
 data.frame(new.cluster=i, n\_ssms=n\_ssms, proportion=sum(clusters$proportion[ct==i]\*clusters$n\_ssms[ct==i])/n\_ssms, width=w)  
 }  
 ))  
}  
  
  
removeSuperclones <- function(clusters, min.frac=0.1, delta.prop=0.1) {  
 m <- which(clusters$proportion == max(clusters$proportion[clusters$n\_ssms >= 0.1 \* sum(clusters$n\_ssms)]))  
 w <- clusters$proportion >= clusters$proportion[m] - delta.prop  
 if(sum(w)>1){  
 cl <- as.data.frame(rbind(if(any(!w)) clusters[!w,,drop=FALSE], if(any(w)) colSums(clusters[w,,drop=FALSE]\*(clusters[w,"n\_ssms"]/sum(clusters[w,"n\_ssms"])))))  
 cl[nrow(cl),"n\_ssms"] <- sum(clusters[w,"n\_ssms"])  
 clusters <- cl  
 }  
 return(clusters)  
}  
  
clustersFromBB <- function(bb){  
 w <- bb$clonal\_frequency == max(bb$clonal\_frequency, na.rm=TRUE) | bb$clonal\_frequency < 0.5 \* max(bb$clonal\_frequency, na.rm=TRUE)  
 t <- table(bb$clonal\_frequency[w])  
 cl <- data.frame(cluster=seq\_along(t), n\_ssms=as.numeric(t), proportion=as.numeric(names(t)))  
 mergeClusters(cl, deltaFreq=0.2)  
}  
  
  
  
probGenotype <- function(vcf){  
 dg <- factor(paste(unlist(info(vcf)$DG)), levels=c("NA",as.character(CANCERGENES)))  
 P <- pGainToTime(vcf)  
 G <- matrix(0, ncol=5, nrow=nlevels(dg), dimnames=list(levels(dg), c(colnames(P),"NA")))  
 t <- table(dg)  
 for(g in names(t[t>0]))  
 G[g,] <- c(colSums(P[dg==g,,drop=FALSE],na.rm=TRUE), "NA"=sum(is.na(P[dg==g,1])))  
 return(G)  
}  
  
probGenotypeTail <- function(vcf){  
 dg <- factor(paste(unlist(info(vcf)$DG)), levels=c("NA",as.character(CANCERGENES)))  
 P <- info(vcf)$pMutCNTail  
 G <- rep(NA, nlevels(dg))  
 names(G) <- levels(dg)  
 t <- table(dg)  
 for(g in names(t[t>0]))  
 G[g] <- mean(P[dg==g,drop=FALSE],na.rm=TRUE)  
 return(G)  
}  
  
getGenotype <- function(vcf, reclassify='missing', ...){  
 w <- c(TRUE,diff(start(vcf)) != 1)  
 cls <- classifyMutations(vcf, reclassify=reclassify)  
 t <- info(vcf)$TCN  
 if(is.null(t))  
 t <- info(vcf)$MinCN + info(vcf)$MajCN  
 hom <- factor(info(vcf)$MutCN==t, levels=c(TRUE,FALSE))  
 dg <- factor(unlist(info(vcf)$DG), levels=as.character(CANCERGENES))  
 table(gene=dg[w], class=cls[w], homozygous=hom[w], ...)  
}  
  
tryExceptNull <- function(x) if(class(x)=="try-error") GRanges() else x  
  
loadVcf <- function(ID){  
 file <- dir(vcfPath, pattern=paste0(ID, ".+somatic.snv\_mnv.TNC.vcf.bgz$"), full.names=TRUE)  
 pos <- loadPositions(ID)  
 vcf <- readVcf(file, genome="GRCh37") #, param=ScanVcfParam(which=pos))  
 f <- findOverlaps(pos, vcf, select="first")  
 vcf <- vcf[na.omit(f)]  
 vcf <- addDriver(vcf, CANCERGENES)  
 i = header(vcf)@header$INFO  
 exptData(vcf)$header@header$INFO <- rbind(i, DataFrame(Number=1,Type="Numeric",Description="DP cluster", row.names="DPC"))  
 i = header(vcf)@header$INFO  
 exptData(vcf)$header@header$INFO <- rbind(i, DataFrame(Number=1,Type="Numeric",Description="DP cluster probability", row.names="DPP"))  
 info(vcf)$DPC <- pos$cluster[!is.na(f)]  
 info(vcf)$DPP <- pos$likelihood[!is.na(f)]   
 vcf  
}  
  
isDeamination <- function(vcf) grepl("(A.CG)|(T..CG)", paste(as.character(unlist(alt(vcf))),vcf@info$TNC))  
isDeaminationNoUV <- function(vcf) grepl("(A.CG[C,T])|(T.[A,G]CG)", paste(as.character(unlist(alt(vcf))),vcf@info$TNC))  
  
  
testDriver <- function(vcf) sapply(info(vcf)$VC, function(x) if(length(x) ==0) FALSE else any(x %in% c('nonsense','missense','ess\_splice','frameshift','inframe','cds\_distrupted')))  
  
addDriver <- function(vcf, mutsigDrivers){  
 i = header(vcf)@header$INFO  
 exptData(vcf)$header@header$INFO <- rbind(i, DataFrame(Number=1,Type="String",Description="Driver gene", row.names="DG"))  
 if(nrow(vcf)==0){  
 info(vcf)$DG <- CharacterList()  
 return(vcf)  
 }  
   
 r <- paste(paste0("^",mutsigDrivers,"(?=\\|)"), collapse="|")  
 rowIdx <- grepl(r, info(vcf)$VD, perl=TRUE) & testDriver(vcf)  
 g <- sapply(info(vcf)$VD, function(x) sub("\\|.+","", x))  
 d <- ifelse(rowIdx,g,character(0))  
 #d[is.na(d)] <- character(0)  
 info(vcf)$DG <- as(d,"CharacterList")  
 vcf  
}  
  
loadAssignment <- function(ID){  
 file <- paste0(dpPath,"/",ID,"\_DPoutput\_1250iters\_250burnin/",ID,"\_DP\_and\_cluster\_info.txt")  
 read.table(file, header=TRUE, sep="\t")  
}  
  
addAssignment <- function(vcf, ID){  
 a <- loadAssignment(ID)  
 i = header(vcf)@header$INFO  
 exptData(vcf)$header@header$INFO <- rbind(i, DataFrame(Number=ncol(a),Type="Numeric",Description="DP probability", row.names="DPP"))  
 info(vcf)$DPP <- as.matrix(a)  
 vcf  
}  
  
loadPositions <- function(ID){  
 file <- gzfile(paste0(dpPath,"/",ID,"\_mutation\_assignments.txt.gz"))  
 tmp <- read.table(file, header=TRUE, sep="\t")  
 #GRanges(tmp$chr, IRanges(tmp$start+1, tmp$end), cluster=tmp$cluster, likelihood=tmp$likelihood)  
 GRanges(tmp$chr, IRanges(tmp$pos, width=1), cluster=tmp$cluster)  
}  
  
tncToPyrimidine <- function(vcf){  
 a <- unlist(alt(vcf))  
 r <- ref(vcf)  
 tnc <- DNAStringSet(info(vcf)$TNC)  
 rc <- grepl("A|G", r)  
 tnc[rc] <- reverseComplement(tnc[rc])  
 a[rc] <- reverseComplement(a[rc])  
 t <- paste0(substr(tnc,1,1), "[",substr(tnc,2,2), ">",a, "]", substr(tnc,3,3))  
 n <- c("A","C","G","T")  
 f <- paste0(rep(n, each=4), "[", rep(c("C","T"), each=96/2), ">", c(rep(c("A","G","T"), each=48/3), rep(c("A","C","G"), each=48/3)), "]", n)  
 factor(t, levels=f)  
}   
  
  
applyPigeonHole <- function(ID){  
 c <- loadClusters(ID)  
 p <- purityPloidy[ID,"purity"]  
 mcf <- c$proportion#\*p  
 l <- sapply(1:length(mcf), function(i) mcf[i] > pmax(0,1-mcf))  
 w <- which(l & upper.tri(l), arr.ind=TRUE)  
 cbind(c$cluster[w[,1]], c$cluster[w[,2]])  
}  
  
reduceToCoverRelations <- function(rel){  
 if(nrow(rel) ==0) return(rel)  
 n <- max(rel)  
 P <- matrix(FALSE,n,n)  
 for(i in 1:nrow(rel))  
 P[rel[i,1],rel[i,2]] <- TRUE  
 for(i in 1:n){  
 q <- list()  
 visit <- logical(n)  
 for(j in 1:n)  
 if(P[i,j])  
 q <- c(q,j)  
 while(length(q)>0){  
 j <- q[[1]]  
 q[[1]] <- NULL  
 for(k in 1:n){  
 if(P[j,k] & !visit[k]){  
 visit[k] <- TRUE  
 q <- c(q,k)  
 if(P[i,k])  
 P[i,k] <- FALSE  
 if(P[k,i])  
 stop("Error.")  
   
 }  
 }  
 }  
 }  
 which(P, arr.ind=TRUE)  
}  
  
cnWeights <- function(vcf){  
 t <- if(is.null(info(vcf)$TCN)) (info(vcf)$MajCN + info(vcf)$MinCN) else info(vcf)$TCN  
 info(vcf)$MutCN / t  
}  
  
branchLengths <- function(vcf, type=c("all","deamination")){  
 type <- match.arg(type)  
 if(type=="deamination")  
 w <- isDeamination(vcf)  
 else  
 w <- rep(TRUE, nrow(vcf))  
 cnw <- cnWeights(vcf)  
 u <- allClusters[[meta(header(vcf))$META["ID",1]]]$cluster  
 b <- sapply(u, function(c) sum(2\*cnw \* (info(vcf)$DPC==c & w), na.rm=TRUE))  
 #if(length(b)==0) b <- rep(0, length(u))  
 names(b) <- u  
 return(b)  
}  
  
avgWeights <- function(vcf, type=c("all","deamination")){  
 type <- match.arg(type)  
 if(type=="deamination")  
 w <- isDeamination(vcf)  
 else  
 w <- rep(TRUE, nrow(vcf))  
 cnw <- cnWeights(vcf)  
 mean(2\*cnw[w], na.rm=TRUE)  
}  
  
predictRealTime <- function(x, signatures=snmf$snmfFit$P[-1,]){  
 snmf$snmfFit$P[1,1] / snmf$weight \* snmf$nmSolve(x, signatures, maxIter=100)[1,]  
}  
  
#' ### Compute graphs (posets)  
toGraph <- function(edgelist, branch.length, edge.labels, node.labels=1:max(edgelist)){  
 g <- graph.edgelist(edgelist)  
 E(g)$weight <- branch.length  
 E(g)$name <- edge.labels  
 V(g)$name <- node.labels  
 return(g)   
}  
  
na.rm <- function(x) x[!is.na(x)]  
  
plotPoset <- function(g){  
 c <- colorRampPalette(brewer.pal(9, "Spectral"))(10)  
 plot(g, layout=layout.reingold.tilford(g), edge.label=E(g)$name, vertex.size = 25\*pmin(1,V(g)$size), edge.width=E(g)$weight/100)  
}  
  
#' Distance in poset   
posetDist <- function(g) {  
 e <- get.edgelist(g)  
 w <- c(0,E(g)$weight)  
 names(w) <- c("Germline", e[,2])  
 d <- shortest.paths(g, mode='out')  
 d <- d - rep(w[colnames(d)], each=length(w))/2  
 diag(d) <- NA  
 d  
}  
  
getMutationCluster <- function(allMutations, vcf){  
 m <- match(allMutations, unlist(info(vcf)$DG))  
 info(vcf)$DPC[m]  
}  
  
distAsRange <- function(g){  
 e <- get.edgelist(g)  
 w <- c(0,E(g)$weight)  
 names(w) <- c("Germline", e[,2])  
 d <- shortest.paths(g, mode='out')  
 y <- shift(IRanges(-w[colnames(d)],0), d["Germline", ])  
 names(y) <- paste0(colnames(d), ", genes=",E(g)$name[match(colnames(d), e[,2])])  
 y  
}  
  
  
nmSolve <- function(D, P, maxIter = 500, tol=1e-3) {  
 n <- nrow(D)  
 m <- ncol(D)  
 s <- ncol(P)  
 tP <- t(P)  
 rP <- rep(colSums(P), m)  
 D <- as.matrix(D)  
 E1 <- E2 <- matrix(runif(s \* m, 1e-3, 1), ncol = m)  
 err <- 2\*tol  
   
 iter <- 1  
 while (iter < maxIter & err > tol) {  
 E1 <- E2  
 E2 <- E1 \* (tP %\*% (D/(P %\*% (E1 + .Machine$double.eps))))/rP  
 iter <- iter + 1  
 err <- mean(abs(E2 - E1)/(E1+.Machine$double.eps), na.rm=TRUE)  
 if(iter %% 100 == 0) cat(round(-log10(err)))  
 }  
 cat("\n")  
 if(iter == maxIter) warning(paste("No convergence after",iter, "iterations."))  
 E2  
}  
  
wnmSolve <- function(D, P, weights = rep(0, ncol(P)), maxIter = 500, tol=1e-3) {  
 D <- as.matrix(D)  
 D <- rbind(D, matrix(weights, ncol=ncol(D), nrow=ncol(P)))  
 P <- rbind(P, diag(rep(1,ncol(P))))  
 n <- nrow(D)  
 m <- ncol(D)  
 s <- ncol(P)  
 tP <- t(P)  
 rP <- rep(colSums(P), m)  
 E1 <- E2 <- matrix(runif(s \* m, 1e-3, 1), ncol = m)  
 err <- 2\*tol  
   
 iter <- 1  
 while (iter < maxIter & err > tol) {  
 E1 <- E2  
 E2 <- E1 \* (tP %\*% (D/(P %\*% (E1 + .Machine$double.eps))))/rP  
 iter <- iter + 1  
 err <- mean(abs(E2 - E1)/(E1+.Machine$double.eps), na.rm=TRUE)  
 if(iter %% 100 == 0) cat(round(-log10(err)))  
 }  
 cat("\n")  
 if(iter == maxIter) warning(paste("No convergence after",iter, "iterations."))  
 E2  
}  
  
  
  
wgdTest <- function(vcf){  
 id <- meta(header(vcf))$META["ID",1]  
 bb <- allBB[[id]]  
 ix <- which(bb$copy\_number==4 & bb$minor\_cn==2)  
 v <- vcf[vcf %over% bb[ix]]  
 #w <- sum(as.numeric(width(reduce(bb[ix]))))  
 t <- table(info(v)$MutCN, info(v)$TCN, as.character(seqnames(v)), info(v)$DPC)  
}  
  
#' Power  
power <- function(f,n, theta=6.3, err=1e-4) if(any(is.na(c(f,n,theta,err)))) NA else sum((log10(dbinom(0:n, n, 0:n/n) / dbinom(0:n,n,err)) > theta) \* dbinom(0:n,n,f))  
  
testIndel <- function(vcf) sapply(info(vcf)$VC, function(x) if(length(x) ==0) FALSE else any(x %in% c('frameshift','inframe','ess\_splice','SO:0001577:complex\_change\_in\_transcript', 'SO:0001582:initiator\_codon\_change', 'splice\_region')))  
  
asum <- function(x, dim) apply(x, setdiff(seq\_along(dim(x)), dim), sum)  
  
#' official driver file  
#d <- lapply(2:3, function(sheet) xlsx::read.xlsx2(file="~/pcawg/ref/TableS2\_driver\_point\_mutations\_annotation.xlsx", sheetIndex=sheet, colIndex=1:22, stringsAsFactors=FALSE, na.char="NaN"))  
#colnames(d[[2]]) <- colnames(d[[1]])  
#drivers <- do.call("rbind",d)  
#drivers[drivers=="NaN" | drivers==""] <- NA  
#drivers <- as.data.frame(sapply(drivers, function(x) if(all(!is.na(as.numeric(x[!is.na(x)])))) as.numeric(x) else x, simplify=FALSE))  
finalData <- read.table("~/pcawg/final/ref/release\_may2016.v1.4.tsv", header=TRUE, sep="\t")  
#r <- gsub("-","",drivers$ref)  
#i <- grepl("-",drivers$ref) | grepl("-",drivers$alt) #drivers$mut\_type=="indel" # need to fix indels  
#r[i] <- paste0("N",r[i])  
#a <- gsub("-","",drivers$alt)  
#a[i] <- paste0("N",a[i])  
#p <- drivers$pos  
#p[i & !grepl("-", drivers$ref)] <- p[i & !grepl("-", drivers$ref)]-1  
#m <- sapply(levels(drivers$sample), function(x) grep(x, finalData$sanger\_variant\_calling\_file\_name\_prefix))  
#finalDrivers <- VRanges(seqnames = drivers$chr, ranges=IRanges(p, width = width(r)), ref=DNAStringSet(r), alt=DNAStringSet(a), sampleNames = finalData$icgc\_donor\_id[m[drivers$sample]])  
#mcols(finalDrivers) <- cbind(sample=drivers$sample, samples=finalData$sanger\_variant\_calling\_file\_name\_prefix[m[drivers$sample]], ID= drivers$gene\_id, drivers[,8:22], mut\_type=ifelse(i, "indel","snv"))  
#save(finalDrivers, file="~/pcawg/ref/TableS2\_driver\_point\_mutations\_annotation.RData")  
load(file="~/pcawg/ref/TableS2\_driver\_point\_mutations\_annotation.RData")  
CANCERGENES <- levels(finalDrivers$ID)  
  
matchDrivers <- function(vcf, finalDrivers) {  
 ID <- meta(header(vcf))$META["ID",1]  
 d <- finalDrivers[grep(ID, finalDrivers$samples)]  
 g <- factor(rep(NA,nrow(vcf)), levels = levels(d$ID))  
 if(length(d)==0)  
 return(g)  
 overlaps <- findOverlaps(vcf, d)  
 g[queryHits(overlaps)] <- d$ID[subjectHits(overlaps)]  
 return(g)  
}  
  
addFinalDriver <- function(vcf, finalDrivers){  
 i = header(vcf)@header$INFO  
 exptData(vcf)$header@header$INFO <- rbind(i, DataFrame(Number=1,Type="String",Description="Driver mutation", row.names="DG"))  
 info(vcf)$DG <- factor(rep(NA,nrow(vcf)), levels = levels(finalDrivers$ID))  
 if(nrow(vcf)==0)  
 return(vcf)  
 g <- matchDrivers(vcf = vcf, finalDrivers = finalDrivers)  
 info(vcf)$DG <- g  
 return(vcf)  
}  
  
  
clinicalData <- read.table("../ref/pcawg\_donor\_clinical\_August2016\_v9.tsv", header=TRUE, sep="\t", comment="", quote="")  
  
load("../ref/Sarcs\_ages.RDa")  
for(x in Sarcs\_age)  
 clinicalData$donor\_age\_at\_diagnosis[match(as.character(x$icgc\_donor\_id), as.character(clinicalData$icgc\_donor\_id))] <- as.numeric(x$Age)  
rm(Sarcs\_age)  
specimenData <- read.table("../ref/pcawg\_specimen\_histology\_August2016\_v9.tsv", header=TRUE, sep="\t", comment="", quote="")  
specimenData$histology\_abbreviation <- sub("CA$","Ca",specimenData$histology\_abbreviation)  
  
s <- strsplit(as.character(finalData$sanger\_variant\_calling\_file\_name\_prefix),",")  
sample2donor <- as.character(finalData$icgc\_donor\_id[unlist(sapply(seq\_along(s), function(i) rep(i, length(s[[i]]))))])  
names(sample2donor) <- unlist(s)  
  
s <- strsplit(as.character(finalData$sanger\_variant\_calling\_file\_name\_prefix),",")  
sample2icgc <- as.character(finalData$tumor\_wgs\_icgc\_sample\_id[unlist(sapply(seq\_along(s), function(i) rep(i, length(s[[i]]))))])  
names(sample2icgc) <- unlist(s)  
  
  
donor2type <- factor(specimenData$histology\_abbreviation, levels=c(sort(unique(specimenData$histology\_abbreviation))[-1], ""))  
names(donor2type) <- specimenData$icgc\_donor\_id  
levels(donor2type)[levels(donor2type)==""] <- "Other/NA"  
  
  
t <- read.table("../ref/tumour\_subtype\_consolidation\_map.tsv - Unique List of Tumour Types\_August.tsv", sep='\t', header=TRUE, comment="")  
c <- as.character(t$`Color..RGB.code.`)  
names(c) <- sub("CA$","Ca",t$`Abbreviation`)  
c <- c[c != "" & !duplicated(names(c))]  
tissueColors <- c(table(donor2type))\*NA  
tissueColors[names(c)] <- c  
tissueColors["Lymph-CLL"] <- "#F4A35D"  
  
tissueBorder <- c("white","black")[names(tissueColors) %in% c("Lung-SCC","Lung-AdenoCa")+1]  
names(tissueBorder) <- names(tissueColors)  
  
tissueLines <- tissueColors  
tissueLines[names(tissueColors) %in% c("Lung-SCC","Lung-AdenoCa")] <- "black"  
  
tissueLty <- c(1,1)[names(tissueColors) %in% c("Lung-SCC","Lung-AdenoCa")+1]  
names(tissueLty) <- names(tissueColors)  
tissueLty["Lung-SCC"] <- 5  
tissueLty["Lung-AdenoCa"] <- 4  
  
tissueCex <- tissueLty  
tissueCex[grep("Lung", names(tissueCex))] <- 0.8  
  
averageHom <- function(bb){  
 sum(width(bb) \* (bb$minor\_cn == 0) \* bb$clonal\_frequency, na.rm=TRUE) / sum(width(bb) \* bb$clonal\_frequency, na.rm=TRUE)  
}  
  
.classWgd <- function(ploidy, hom) 2.9 -2\*hom <= ploidy  
  
classWgd <- function(bb) .classWgd(averagePloidy(bb), averageHom(bb))  
  
plotBB <- function(bb, ylim=c(0,max(max(bb$total\_cn, na.rm=TRUE))), col=RColorBrewer::brewer.pal(4,"Set2"), type=c("lines","bars"), legend=TRUE, lty.grid=1, col.grid="grey", xaxt=TRUE){  
 type <- match.arg(type)  
 s <- c(1:22, "X","Y")  
 l <- as.numeric(width(refLengths[seqnames(refLengths) %in% s]))  
 names(l) <- s  
 plot(NA,NA, ylab="Copy number",xlab="",xlim=c(0,sum(l)), ylim=ylim, xaxt="n")  
 c <- cumsum(l)  
 axis(side=1, at=c(0,c), labels=rep('', length(l)+1))  
 if(xaxt) mtext(side=1, at= cumsum(l) - l/2, text=names(l), line=1)  
 #abline(v=c, lty=3)  
 if(type=="lines"){  
 x0 <- start(bb) + cumsum(l)[as.character(seqnames(bb))] - l[as.character(seqnames(bb))]  
 x1 <- end(bb) + cumsum(l)[as.character(seqnames(bb))] - l[as.character(seqnames(bb))]  
 lwd <- 5\* bb$clonal\_frequency / max(bb$clonal\_frequency)  
 segments(x0=x0, bb$major\_cn ,x1, bb$major\_cn, col=col[1], lwd=lwd)  
 segments(x0=x0, bb$minor\_cn -.125,x1, bb$minor\_cn-.125, col=col[2], lwd=lwd)  
 segments(x0=x0, bb$total\_cn+.125,x1, bb$total\_cn+.125, col=1, lwd=lwd)  
 # cv <- coverage(bb)  
 # cv <- cv[s[s%in%names(cv)]]  
 # par(xpd=NA)  
 # for(n in names(cv)){  
 # cc <- cv[[n]]  
 # segments(start(cc) + cumsum(l)[n] - l[n] ,-runValue(cc)/2,end(cc)+ cumsum(l)[n] - l[n], -runValue(cc)/2, col=4)  
 # }  
 }else{  
 ub <- unique(bb)  
 f <- findOverlaps(ub,bb)  
 m <- t(model.matrix( ~ 0 + factor(queryHits(f))))  
 ub$total\_cn <- m %\*% mg14::na.zero(bb$total\_cn \* bb$clonal\_frequency) / max(bb$clonal\_frequency)  
 ub$major\_cn <- m %\*% mg14::na.zero(bb$major\_cn \* bb$clonal\_frequency) / max(bb$clonal\_frequency)  
 ub$minor\_cn <- m %\*% mg14::na.zero(bb$minor\_cn \* bb$clonal\_frequency) / max(bb$clonal\_frequency)  
 ub$clonal\_frequency <- max(bb$clonal\_frequency)  
 x0 <- start(ub) + cumsum(l)[as.character(seqnames(ub))] - l[as.character(seqnames(ub))]  
 x1 <- end(ub) + cumsum(l)[as.character(seqnames(ub))] - l[as.character(seqnames(ub))]  
 rect(x0,0,x1, ub$major\_cn, col=col[2], lwd=NA)  
 rect(x0,ub$major\_cn,x1, ub$total\_cn, col=col[1], lwd=NA)  
 abline(h = 1:floor(ylim[2]), lty=lty.grid, col=col.grid)  
 }  
 abline(v = chrOffset[1:25], lty=lty.grid, col=col.grid)  
 if(xaxt) mtext(side=1, line=1, at=chrOffset[1:24] + diff(chrOffset[1:25])/2, text=names(chrOffset[1:24]))  
 if(legend){  
 if(type=="lines") legend("topleft", c("Total CN","Major CN","Minor CN"), col=c("black", col[1:2]), lty=1, lwd=2, bg='white')  
 else legend("topleft", c("Major CN","Minor CN"), fill=col[1:2], bg='white')  
 }  
}  
  
plotVcf <- function(vcf, bb, clusters, col = RColorBrewer::brewer.pal(9, "Set1")[c(3,4,2,1,9)], ID = meta(header(vcf))[[1]]["ID",1], IS\_WGD=classWgd(bb), NO\_CLUSTER=FALSE, title=TRUE, legend=TRUE, lty.grid=1, col.grid="grey", xaxt=TRUE, pch=16, pch.out=pch, cex=0.66) {  
 cls <- factor(paste(as.character(info(vcf)$CLS)), levels = c(levels(info(vcf)$CLS), "NA"))  
 plot(start(vcf) + chrOffset[as.character(seqnames(vcf))], getAltCount(vcf)/getTumorDepth(vcf),col=col[cls], xlab='', ylab="VAF", pch=ifelse(info(vcf)$pMutCNTail < 0.025 | info(vcf)$pMutCNTail > 0.975, pch.out , pch), ylim=c(0,1), xlim=c(0,chrOffset["MT"]), xaxt="n", cex=cex)  
 if(title){  
 title(main=paste0(ID,", ", donor2type[sample2donor[ID]], "\nploidy=",round(averagePloidy(bb),2), ", hom=",round(averageHom(bb),2), if(IS\_WGD) ", WGD" else "", if(NO\_CLUSTER) ", (No clusters available)" else(paste0(", clusters=(",paste(round(clusters$proportion, 2), collapse="; "),")"))), font.main=1, line=1, cex.main=1)  
 }   
 abline(v = chrOffset[1:25], lty=lty.grid, col=col.grid)  
 if(xaxt) mtext(side=1, line=1, at=chrOffset[1:24] + diff(chrOffset[1:25])/2, text=names(chrOffset[1:24]))  
 for(i in which(!sapply(bb$timing\_param, is.null))) {  
 s <- start(bb)[i]  
 e <- end(bb)[i]  
 x <- chrOffset[as.character(seqnames(bb)[i])]  
 y <- bb$timing\_param[[i]][,"f"]  
 l <- bb$timing\_param[[i]][,"pi.s"] \* bb$timing\_param[[i]][,"P.m.sX"]  
 if(any(is.na(c(s,e,x,y,l)))) next  
 segments(s+x,y,e+x,y, lwd=l\*4+.1)  
 #text(x=(s+e)/2 +x, y=y, paste(signif(bb$timing\_param[[i]][,"m"],2),signif(bb$timing\_param[[i]][,"cfi"]/purityPloidy[meta(header(vv))["ID",1],"purity"],2), sep=":"), pos=3, cex=0.5)  
 }  
 if(legend) legend("topleft", pch=19, col=col, legend=paste(as.numeric(table(cls)), levels(cls)), bg='white')  
}  
  
timeToBeta <- function(time){  
 mu <- time[,1]  
 #if(any(is.na(time))) return(c(NA,NA))  
 mu <- pmax(1e-3, pmin(1 - 1e-3, mu))  
 v <- (0.5 \* (pmax(mu,time[,3])-pmin(mu,time[,2])))^2  
 alpha <- mu \* (mu \* (1-mu) / v - 1)  
 beta <- (1-mu) \* (mu \* (1-mu) / v - 1)  
 return(cbind(alpha, beta))  
}  
  
plotTiming <- function(bb, time=mcols(bb)[,c("type","time","time.lo","time.up")], col=paste0(RColorBrewer::brewer.pal(5,"Set2")[c(3:5)],"88"), legend=TRUE, col.grid='grey', lty.grid=1){  
 plot(NA,NA, xlab='', ylab="Time [mutations]", ylim=c(0,1), xlim=c(0,chrOffset["MT"]), xaxt="n")  
 try({  
 bb <- bb[!is.na(bb$time)]  
 s <- start(bb)  
 e <- end(bb)  
 x <- chrOffset[as.character(seqnames(bb))]  
 y <- time[,2]  
 rect(s+x,time[,3],e+x,time[,4], border=NA, col=col[time[,1]], angle = ifelse(bb$time.star=="\*" | is.na(bb$time.star),45,135), density=ifelse(bb$time.star == "\*\*\*", -1, 72))  
 segments(s+x,y,e+x,y)  
 }, silent=FALSE)  
 abline(v = chrOffset[1:25], lty=lty.grid, col=col.grid)  
 s <- c(1:22, "X","Y")  
 l <- as.numeric(width(refLengths[seqnames(refLengths) %in% s]))  
 names(l) <- s  
 c <- cumsum(l)  
 axis(side=1, at=c(0,c), labels=rep('', length(l)+1))  
 mtext(side=1, line=1, at=chrOffset[1:24] + diff(chrOffset[1:25])/2, text=names(chrOffset[1:24]))  
 if(legend) legend("topleft", levels(time[,1]), fill=col, bg="white")  
}  
  
source("../modules/MutationTime.R/MutationTime.R")  
  
findMainCluster <- function(bb, min.dist=0.05){  
 w <- which(bb$n.snv\_mnv > 20 & !is.na(bb$time))  
 # d <- dist(bb$time[w])  
 # ci <- weighted.mean((bb$time.up - bb$time.lo)[w], width(bb)[w])  
 # h <- hclust(d, method='average', members=bb$n.snv\_mnv[w])  
 # c <- cutree(h, h=ci)  
 # ww <- c==which.max(table(c))  
 # weighted.mean(bb$time[w][ww], 1/((bb$time.up - bb$time.lo + min.dist)[w][ww]), na.rm=TRUE)  
 s <- seq(0,1,0.01)  
 l2 <- pmin(bb$time.lo, bb$time - min.dist)[w]  
 u2 <- pmax(bb$time.up, bb$time + min.dist)[w]  
 l1 <- (l2 + bb$time[w])/2  
 u1 <- (u2+ bb$time[w])/2  
 wd <- as.numeric(width(bb)[w])  
 o <- sapply(s, function(i) sum(wd \* ( (l2 <= i & u2 >=i) + (l1 <= i & u1 >= i))))  
 s[which.max(o)]  
}  
  
fractionGenomeWgdCompatible <- function(bb, min.dist=0.05){  
 m <- findMainCluster(bb)  
 l <- pmin(bb$time.lo, bb$time - min.dist)  
 u <- pmax(bb$time.up, bb$time + min.dist)  
 w <- which(l <= m & u >= m)  
 avgCi <- weighted.mean(bb$time.up- bb$time.lo, width(bb), na.rm=TRUE)  
 sd.wgd <- sqrt(weighted.mean((bb$time[w] - m)^2, width(bb)[w], na.rm=TRUE))  
 sd.all <- sqrt(weighted.mean((bb$time - m)^2, width(bb), na.rm=TRUE))  
 c(nt.wgd=sum(as.numeric(width(bb))[w]), nt.total=sum(as.numeric(width(bb))[!is.na(bb$time)]), time.wgd=m, n.wgd=length(w), n.all = sum(!is.na(bb$time)), chr.wgd = length(unique(seqnames(bb)[w])), chr.all = length(unique(seqnames(bb)[!is.na(bb$time)])), sd.wgd=sd.wgd, avg.ci=avgCi, sd.all=sd.all)   
}  
  
flattenBB <- function(bb){  
 u <- unique(bb)  
 d <- duplicated(bb)  
 mcols(u) <- mcols(u)[1:7]  
 u$total\_cn\_2 <- u$major\_cn\_2 <- u$minor\_cn\_2 <- as.integer(NA)  
 u$clonal\_frequency\_2 <- as.numeric(0)  
 if(any(d)){  
 s <- bb[d]  
 f <- findOverlaps(s, u, select='first')  
 mcols(u)[f, c("total\_cn\_2","major\_cn\_2","minor\_cn\_2","clonal\_frequency\_2")] <- mcols(s)[, c("total\_cn","major\_cn","minor\_cn","clonal\_frequency")]  
 }  
 u  
}  
  
reduceBB <- function(bb){  
 b <- split(bb, do.call("paste", mcols(bb)[c("clonal\_frequency","major\_cn","minor\_cn")]))  
 r <- reduce(b)  
 s <- sort(unlist(r))  
 d <- DataFrame(t(sapply(strsplit(names(s), " "), as.numeric)))  
 names(d) <- c("clonal\_frequency","major\_cn","minor\_cn")#names(mcols(bb))  
 mcols(s) <- d  
 names(s) <- NULL  
 u <- unique(bb)  
 f <- findOverlaps(s, u)  
 t <- table(subjectHits(f), queryHits(f))  
 s$n.snv\_mnv <- u$n.snv\_mnv %\*% as.matrix(t)  
 s$total\_cn <- s$major\_cn + s$minor\_cn  
 s$timing\_param <- vector(mode="list", length=length(s))  
 s$timing\_param[subjectHits(f)] <- u$timing\_param[queryHits(f)]  
 return(s)  
}  
  
plotSample <- function(w, vcf = finalSnv[[w]], bb = finalBB[[w]], title=w) {  
 p <- par()  
 stackTime <- function(bb, t=seq(0,1,0.01)){  
 u <- unique(bb)  
 w <- as.numeric(width(u))  
 f <- function(x) pmin(pmax(x,0.01),0.99)  
 ut <- f((0.5\*5+u$time \* u$n.snv\_mnv)/(5+u$n.snv\_mnv))  
 uu <- f(u$time.up)  
 ul <- f(u$time.lo)  
 diff(car::logit(f(t))) \* rowSums(sapply(which(!is.na(ut)), function(i) w[i]\*dnorm(car::logit(t[-1] - diff(t)/2), mean=car::logit(ut[i]), sd= (car::logit(uu[i]) - car::logit(ul[i]) + 0.05)/4)))#(t <= u$time.up[i] & t >= u$time.lo[i])))  
 #rowSums(sapply(which(!is.na(ut)), function(i) w[i]\*(t <= u$time.up[i] & t >= u$time.lo[i])))  
 }  
 layout(matrix(1:3, ncol=1), height=c(4,1.2,3.5))  
 par(mar=c(0.5,3,0.5,0.5), mgp=c(2,0.25,0), bty="L", las=2, tcl=-0.25, cex=1)  
 plotVcf(vcf, bb, finalClusters[[w]], title=FALSE, legend=FALSE, col.grid='white', xaxt=FALSE, cex=0.33)  
 mtext(line=-1, side=3, title, las=1)  
 plotBB(bb, ylim=c(0,5), legend=FALSE, type='bar', col.grid='white', col=c("lightgrey", "darkgrey"), xaxt=FALSE)  
 par(mar=c(3,3,0.5,0.5))  
 plotTiming(bb, legend=FALSE, col.grid=NA)  
 s <- stackTime(bb)  
 g <- colorRampPalette(RColorBrewer::brewer.pal(4,"Set1")[c(3,2,4)])(100)  
 segments(x0=chrOffset["MT"] ,y0=seq(0,1,l=100),x1=chrOffset["MT"] + s/max(s) \* 1e8, col=g, lend=3)  
 #print(w)  
 par(p[setdiff(names(p), c("cin","cra","csi","cxy","din","page"))])  
}  
  
t <- read.table("../ref/release\_may2016.v1.1.TiN\_\_donor.TiNsorted.20Jul2016.tsv", header=TRUE, sep="\t")  
TiN <- t$TiN\_donor  
names(TiN) <- t$icgc\_donor\_id