COPYKAT: Inference of genomic copy number and subclonal structure of human tumors from high-throughput single cell RNA data

Author: Ruli Gao rgao@houstonmethodist.org

First version: December 31, 2019 Updated: April 28, 2020

Description

A major challenge for single cell RNA sequencing of human tumors is to distinguish cancer cells from stromal cell types, as well as the presence of multiple tumor subclones. COPYKAT is a computational tool using integrative Bayesian approaches to identify genome-wide aneuploidy at 5MB resolution in single cells to separate tumor cells from normal cells, and tumor subclones using high-throughput sc-RNAseq data. The underlying logic of calculating DNA copy numbers from RNAseq data is that gene expression levels of many adjacent genes can be dosed by genomic DNA copy numbers in that region. Copycat estimated copy numbers can achieve a high concordance (80%) with whole genome DNA sequenced copy numbers. The rationale for prediction tumor/normal cell states is that aneuploidy is common in human cancer. Cells with extensive genome-wide copy number aberrations (aneuploidy) are considered as tumor cells, wherease stromal normal cells often have diploid or pseudo-diploid copy number profiles. As such, this prediction strategy will not work with diploid tumors or tumors with aneuploidy close to 2. In this vignettes, I will take you go through the process of calculating single cell copy numbers, predict aneuploid/diploid cells, and infer tumor cell subpopulations from single cells RNAseq data.

Step 1: installation

Installing copycat from GitHub

```
library(devtools)
install_github("navinlabcode/copykat")
```

An example raw matrix is attached with this package named, exp.rawdata.

To test the package with this dataset, simply issue this line of code in R/Rstudio:

```
copykat.test <- copykat(rawmat=exp.rawdata, sam.name=“test”)
```

Step 2: prepare readcount input

The only one direct input that you need to prepare to run copycat is the raw gene expression matrix, with gene ids in rows and cell names in columns. the gene ids can be gene symbol or ensemble id. You do not need to worry about the genomic positions of each genes, copycat will do the annotation. The matrix values are often the count of unique molecular identifier (UMI) from nowadays high througput single cell RNAseq data. The early generation of scRNAseq data may be summarized as TPM values or total read counts, which should also work. I will show you an example of generating input from 10Xgenomics cellranger output.

An example to generate input from 10X genomics cellranger V3 output

```r
library(Seurat)
raw <- Read10X(data.dir = data.path.to.cellranger.outs)
raw <- CreateSeuratObject(counts = raw, project = "copycat.test", min.cells = 0,
      min.features = 0)
exp.rawdata <- as.matrix(raw@assays$RNA@counts)
```

you can save the matrix for future use.

```r
write.table(exp.rawdata, file="exp.rawdata.txt", sep="\t", quote = FALSE, row.names = TRUE)
```

In this vignette, I will take this example raw UMI count matrix to demonstrate the workflow.

Step 3: run copykat

Now I have prepared the only one input, raw UMI count matrix, I am ready to run copycat. The default gene ids in cellranger output is gene symbol, so I put “Symbol” or “S”. To filter out cells, I require at least 5 genes in each chromosome to calculate DNA copy numbers. I can tune down this ngene.chr=1 to keep as many cells as possible, however I think using at least 5 genes to represent one chromosome is not very stringent. To filter out genes, I can tune parameters to keep only genes that are expressed LOW.DR to UP.DR of cells. I put default LOW.DR=0.05, UP.DR=0.2. I can tune down these values to keep more genes in the analysis. I need to make sure that LOW.DR is smaller than UP.DR.

I ask copycat to take at least 25 genes per segments, suggested range is 15-150 genes per bin. KS.cut is the segmentation parameter, ranging from 0 to 1. Increasing KS.cut decreases sensitivity. ie. less segments.

Here I do parallel computation by setting n.cores = 4. I also give the sample.name by sam.name=“test”. One struggling and intesting observation is that none of one clustering method could fit all datasets.

In this version, I add a distance parameters include “euclidean” distannce and correlational distance, ie “pearson” and “spearman”. In general, corretional distances tend to favor noise data; enclidean distance tend to favor data with larger CNV segments. I observed that 10X cellranger V3 tends to have better clustering results with corelational distance.

```r
library(copykat)
copykat.test <- copykat(rawmat=exp.rawdata, id.type="S", ngene.chr=5, win.size=25,
        KS.cut=0.2, sam.name="test", distance="euclidean", n.cores=4)
#> [1] "test copykat v0"
#> [1] "step1: read and filter data ..."
#> [1] "33694 genes, 302 cells in raw data"
#> [1] "12156 genes past LOW.DR filtering"
#> [1] "step 2: annotations gene coordinates ..."
#> [1] "start annotation ..."
#> [1] "step 3: smoothing data with dlm ..."
#> [1] "step 4: measuring baselines ..."
#> number of iterations= 325
#> number of iterations= 710
#> number of iterations= 227
#> number of iterations= 826
#> number of iterations= 429
#> number of iterations= 762
#> [1] "step 5: segmentation..."
#> [1] "step 6: convert to genomic bins..."
#> [1] "step 7: adjust baseline ..."
#> [1] "step 8: final prediction ..."
#> [1] "step 9: saving results..."
#> [1] "step 10: ploting heatmap ..."
#> Time difference of 1.505151 mins
```

It might take a while to run a dataset with more than 10,000 single cells. It is suggested to run large dataset in terminal using “Rscript”, instead of running copycat in interactive mode in R or Rstudio. I usually run Rscript run_copycat.R in sever and taking either 10X output or raw UMI count matrix as input using the args.

After this step, copycat aumatically save the calculated copy number matrix, the heatmap and tumor/normal prediction results in your working directly. You can also extract from the object:

```
pred.test <- data.frame(copykat.test$prediction)
CNA.test <- data.frame(copykat.test$CNAmat)
```

Step 4: navigate prediction results

Now let’s look at the prediction results. Predicted aneuploid cells are inferred as tumor cells; diploid cells are stromal normal cells.

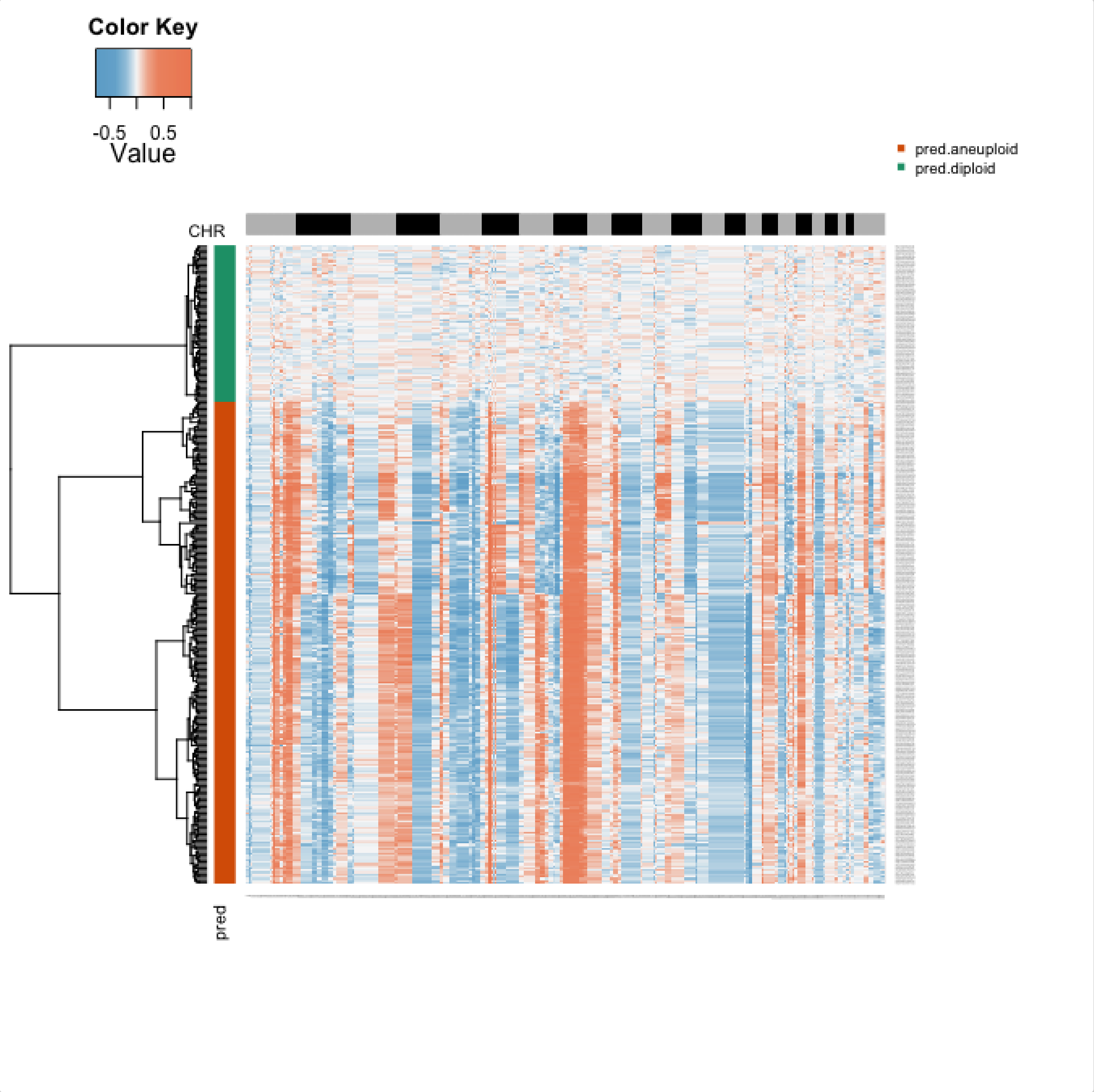
Please note that filtered cells are not included in the result. I am debatinng whether should I include all cells in the results.

```
head(pred.test)
#>                        cell.names copykat.pred
#> AAACCTGCACCTTGTC AAACCTGCACCTTGTC    aneuploid
#> AAACGGGAGTCCTCCT AAACGGGAGTCCTCCT      diploid
#> AAACGGGTCCAGAGGA AAACGGGTCCAGAGGA    aneuploid
#> AAAGATGCAGTTTACG AAAGATGCAGTTTACG    aneuploid
#> AAAGCAACAGGAATGC AAAGCAACAGGAATGC    aneuploid
#> AAAGCAATCGGAATCT AAAGCAATCGGAATCT    aneuploid
```

The first 3 columns in the CNA matrix are the genomic coordinates. Rows are 220KB bins in genomic orders.

```
head(CNA.test[ , 1:5])
#>   chrom chrompos  abspos AAACCTGCACCTTGTC AAACGGGAGTCCTCCT
#> 1     1  1042457 1042457      -0.02957907       0.06251932
#> 2     1  1265484 1265484      -0.02957907       0.06251932
#> 3     1  1519859 1519859      -0.02957907       0.06251932
#> 4     1  1826619 1826619      -0.02957907       0.06251932
#> 5     1  2058465 2058465      -0.02957907       0.06251932
#> 6     1  2280372 2280372      -0.02957907       0.06251932
```

Copykat also generate a heatmap plot for estimated copy numbers. Rows are single cells; columns are 220kb bins in genomic order.



Step 5: define subpopulations of aneuploid tumor cells

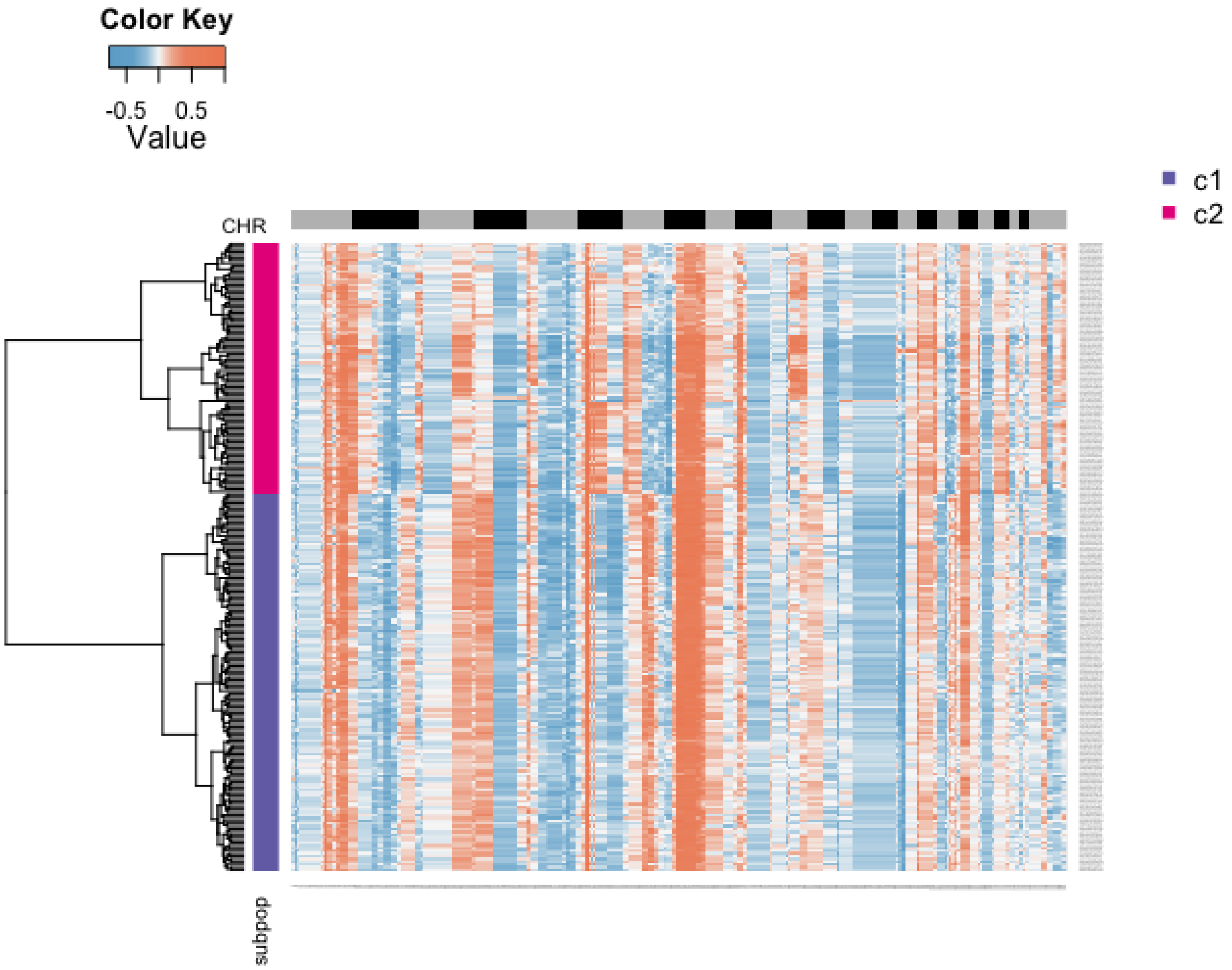
I observed both diploid and aneuploid cells. Next step is to extract aneuploid cells that are considered as tumor cells in aneuploid tumors to define two copy number subpopulations of single tumor cells.

```
tumor.cells <- pred.test$cell.names[which(pred.test$copykat.pred=="aneuploid")]
tumor.mat <- CNA.test[, which(colnames(CNA.test) %in% tumor.cells)]
hcc <- hclust(parallelDist::parDist(t(tumor.mat),threads =4, method = "euclidean"), method = 
        "ward.D2")
hc.umap <- cutree(hcc,2)

rbPal6 <- colorRampPalette(RColorBrewer::brewer.pal(n = 8, name = "Dark2")[3:4])
subpop <- rbPal6(2)[as.numeric(factor(hc.umap))]
cells <- rbind(subpop,subpop)
```

```
heatmap.3(t(tumor.mat),dendrogram="r", distfun = function(x) parallelDist::parDist(x,threads
        =4, method = "euclidean"), hclustfun = function(x) hclust(x, method="ward.D2"),
            ColSideColors=chr1,RowSideColors=cells,Colv=NA, Rowv=TRUE,
            notecol="black",col=my_palette,breaks=col_breaks, key=TRUE,
            keysize=1, density.info="none", trace="none",
            cexRow=0.1,cexCol=0.1,cex.main=1,cex.lab=0.1,
            symm=F,symkey=F,symbreaks=T,cex=1, cex.main=4, margins=c(10,10))

  legend("topright", c("c1","c2"), pch=15,col=RColorBrewer::brewer.pal(n = 8, name = "Dark2")
        [3:4], cex=0.9, bty='n')
```



Now I have defined two subpopulations with major subclonal differences, I can move forward to compare their gene expression profiles, and evaluate gene dosage effects of subclonal copy number changes.

Thank you.

Enjoy.

2020 New Year Eve