In summary, the pipeline requires three python scripts, the R package SCclust and index/mapping commands. Also people need to download their preferred reference genome in one directory filepath/ChromFa

After obtaining the mappable regions, all the following procedures till the input files for viewer are

implemented in R package SCclust.

**## build index, mask pseudoautosomal regions**

cd filepath/ChromFa  (the directory for reference genome chr\*.fa files)

python hg19.chrY.psr.py(hg38.chrY.psr.py)

bash bowtie.build.bash

**## Simulation for mappable regions**

**# parameters: read length, e.g. 100; genome, e.g. hg or hgdm**

mkdir k100

python generate.reads.py 100 hgdm| /filepath/bowtie-0.12.7/bowtie -S -t -v 0 -m 1 -f hgdm - |

python mappable.regions.py > /k100/mappable.regions.txt 2> /k100/mappable.regions.stderr &

**## prepare SAM files for each cell in one directory**

/filepath/cellSAMdir (Directory for SAM files of all single cells, e.g. example.rmdup.sam)

**################ R ############**

library(SCclust)

chromFa\_dir <- "/filepath/ChromFa"

k100\_dir <- "/filepath/ChromFa/k100**"**

**## determine bin boundaries**

bin\_boundaries(k100\_dir, bincount = 20000)

**## Compute GC content**

varbinGC(chromFa\_dir, k100\_dir, Nk = "20k")

**## compute bin counts**

SAM\_dir <- "/filepath/cellSAMdir"

cellname <- "example"

bin\_counts(SAM\_dir,k100\_dir, cellname, Nk = "20k")

**## GC normalization and segmentation for bin counts of each single cell**

bin\_mat <- read.table("/filepath/cellSAMdir/example.varbin.20k.txt", header = T, sep = "\t")

gc <- read.table(“/filepath/k100\_dir/varbin.gc.content.20k.txt”, header = T, sep = “\t”)

bin\_mat\_normalized <- gc\_one(bin\_mat, gc)

bin\_mat\_segmented <- cbs.segment\_one(bin\_mat\_normalized, alpha = 0.05, nperm = 1000, undo.SD = 1.0, min.width = 5, method = "multiplier", genome = "hg" , graphic = TRUE)

**## GC normalization and segmentation for all cells together into one combined table**

segfile <- cbs.segment\_all(SAM\_dir, Nk = "20k", gc, alpha = 0.05, nperm = 1000, undo.SD = 1.0, min.width = 5, method = "multiplier", genome = "hg")

seg.quantal <- segfile$seg.quantal

ratio.quantal <- segfile$ratio.quantal

**## pins, pinmat**

res1 <- preprocess\_segfile(seg.quantal, gc, eviltwins = c("CJA1024", "CJA1025"), ploidies = TRUE)

breakpoint\_table <- res1$breakpoint\_table

ploidies\_table <- res1$ploidies\_table

smear\_table <- findsmears(breakpoint\_table, smear = 1, keepboundaries = FALSE, mask\_XY = TRUE)

res2 <- findpins(breakpoint\_table, smear\_table)

pins <- res2$pins

pinmat <- res2$pinmat

cell\_names <- res2$cell\_names

**## hclust, clone identification**

res3 <- simFisher\_parallel(pins, pinmat, sim\_round = 500)

true\_fisherPV <- res3$true\_fisherPV

sim\_fisherPV <- res3$sim\_fisherPV

res4 <- fdr\_fisherPV(true\_fisherPV, sim\_fisherPV, cell\_names, lm\_max = 0.001, graphic = FALSE)

mat\_fdr <- res4$mat\_fdr

mat\_dist <- res4$mat\_dist

hc <- hclust\_tree(pinmat, mat\_fdr, mat\_dist, hc\_method = "average")

hc\_clone <- find\_clone(hc, fdr\_thresh = -2, share\_min = 0.85, n\_share = 3, bymax = TRUE,

                       climb\_from\_size = 2, climb\_to\_share = 3, graphic = FALSE)

**## subclone identification**

sub\_hc\_clone <- find\_subclone(hc\_clone, pinmat, pins, min\_node\_size = 6, sim\_round = 500,

                              lm\_max = 0.001, hc\_method = "average",base\_share = 3,

                              fdr\_thresh = -2, share\_min = 0.90, bymax = TRUE, climb\_from\_size = 2,

                              climb\_to\_share = 3, graphic = FALSE)

**## generate the output files for viewer**

output\_viewer(output\_file\_dir = "/filepath/viewerInput",

              seg.quantal, ratio.quantal, pins, pinmat, mat\_dist, hc\_clone,

              sub\_hc\_clone, subcloneTooBig = 0.8, smear = 1, study="GL9.2")