Acquired Somatic Lesions Associated with Prostate Adenocarcinoma in a Cohort of Kenyan Patients

Supplement To:

"Manuscript Name Goes Here"

Stephen M. Kelly[1], Philip D. Anderson[1,*], Rose N. Njoroge[2], George Gachara[3], Mungai P. Ngugi[4], Janet Majanja[5], Wallace Bulimo[6], Lucy W. Muchiri[7], M. Christopher Baqir[1], George A.O. Magogha[4], and Sarki A. Abdulkadir[2,†]

[1] Department of Biological Sciences, Salisbury University, Salisbury MD
[2] Department of Urology, Northwestern University Feinberg School of Medicine, Chicago IL
[3] Department of Medical Laboratory Sciences, Kenyatta University, Nairobi Kenya
[4] Department of Surgery, College of Health Sciences, University of Nairobi, Nairobi Kenya
[5] Center for Virus Research, Kenya Medical Research Institute, Nairobi Kenya
[6] Department of Biochemistry, University of Nairobi, Nairobi Kenya
[7] Department of Human Pathology, College of Health Sciences, University of Nairobi, Nairobi Kenya

[*] To whom correspondence should be addressed. Tel: 1+410-548-5788. Fax: 312-503-0386. Email: pdanderson@salisbury.edu
[†] To whom correspondence should be addressed. Tel: 1+312-503-5032. Fax: 312-503-0386. Email: sarki.abdulkadir@northwestern.edu

Abstract

Prostate cancer is the most common non-cutaneous malignancy in American men, and African Americans have not only the highest burden of this disease, but also the worst clinical outcomes. These differences have persisted despite improvements to treatment access, suggesting that differences in the number or types of cancer-promoting mutations may play a role in the disparity. Unfortunately, admixture in the African-American population makes this hypothesis difficult to test in African-American men. Ideally, analysis of prostate tumors in indigenous African populations and comparison of the finding with results obtained by other investigators from analysis of Caucasian, African American and Asian tumors should shed light on this question.

In this report, we performed exome capture sequencing (Exome-Seq) to identify genetic mutations driving prostate tumorigenesis in a cohort of Kenyan patients. In this initial exploratory study, we sequenced prostate cancers and matched blood DNA from four patients at the Kenyatta National Hospital, the University of Nairobi, Kenya. We applied computational techniques to identify tumor-driving mutations in cancer that are not present in the normal DNA from the blood sample. Finally, we compared mutations in the African cohort to published European and African-American cohorts to determine the differences and similarities in the tumor genomes.

This study has helped to shed light on the reasons for poor treatment outcomes among men of African descent, and enrich our understanding of the number and diversity of normal biological pathways that become corrupted in prostate cancer.

1 Post-Excavator Analysis

Sample data has been processed with EXCAVATOR in order to find copy number variants (CNVs); areas of the genome that have undergone duplication or deletion. Each gene is usually present twice in diploid organisms, once on each copy of a chromosome. Therefore, two copies can be considered the 'default' copy number of a gene, while one or zero would indicate a deletion and three copies or more would indicate a duplication. Changes in the number of copies of a gene have been associated with cancer. Results are found in `/home/kellys/AfPC/EXCAVATOR_Package_v2.2/EXCAVATOR/OutputFolder/Results/SAMPLENAME/FastCallResults/SAMPLENAME.txt`, where SAMPLENAME is one of the following:

- KENPCA001
- KENPCA002
- KENPCA003
- KENPCA004
- KENPCA006
- KENPCA007

The sample KENPCA002 yielded no results, and so it has been removed from further processing. The sample KENPCA005 was also excluded after EXCAVATOR found no significant peaks between the blood and tumor sample pairs. A sample of EXCAVATOR FastCall results can be seen below. The file path has been broken up to same space on the page.

```
FILEPATH1="/home/kellys/AfPC/EXCAVATOR_Package_v2.2/EXCAVATOR/OutputFolder/Results/"
FILEPATH2="KENPCA001_Tumor/FastCallResults_KENPCA001_Tumor.txt"
head -n 8 $FILEPATH1$FILEPATH2

## Chromosome Start End Segment CNF CN Call ProbCall
## chr1 990220 1027507 0.786627396572122 3.45007421582464 3 1 0.874648265441447
## chr1 11824301 11849543 0.941386908211988 3.84074693421662 4 1 0.560213623943189
## chr1 26799099 26856529 1.06447775082655 4.18282532095703 4 2 0.842497676798431
## chr1 27657441 27672025 -1.24240292301556 0.845336164735924 1 -2 0.87949445166902
## chr1 36771912 36793791 0.734326471460584 3.3272412389592 3 1 0.911941542237978
## chr1 72747926 75805357 -0.527811670876362 1.38721204369663 1 -1 0.984353191223197
## chr1 76397943 98349455 -0.559727564074479 1.35686053034806 1 -1 0.986505443445259
```

Our EXCAVATOR output consists of eight columns, of which the following are relevant for our analysis:

- `Chromosome`: the chromosome on which the CNV is located
- `Start`: start position of the CNV
- `End`: end position of the CNV
- `CN`: copy number value, indicating the number of copies of the CNV present
- `Call`: copy number state

The `Call` value will be used in our analysis. This value represents the copy number state of the CNV entry, and is reported as one of four values:

- `-2`: 2-copy deletion
- `-1`: 1-copy deletion
- `1`: 1-copy duplication
- `2`: multiple-copy duplication

A `Call` value of `0` is not reported, since `0` would represent no duplications or deletions in a section of the chromosome. It is important to note that the EXCAVATOR results report CNV's as a range of chromosomal coordinates. These coordinates must be annotated to determine which genes are present within these loci.

1.1 Annotation of CNV loci with Bedtools

EXCAVATOR saves results in the form of a .TXT file, giving the chromosome number, chromosomal coordinates, copy number, and probability of each CNV found. In order to determine the genes present within the chromosomal coordinates reported by EXCAVATOR, we must annotate the results with gene symbols and other identifying information. For this purpose, we will be using `bedtools`. The `bedtools` tool set is able to use simple BED files to perform genomic analysis. In particular, we will use the `intersectBed` tool to locate the intersect between the chromosomal coordinates in our sample, and those of a reference genome. First, we must convert the EXCAVATOR output, in .TXT format, into .BED files. The .BED file format in its most basal form consists of three text columns with tab separators. In order, the columns are as follows: chromosome number, chromosome start location, chromosome end location. Our EXCAVATOR results already contain these values, so we only need to strip out the extraneous information and output the files as `SAMPLENAME.BED`. This will be performed in R by loading each sample's EXCAVATOR results into a list. Using a `for` loop, each object in the list will be written out into a table in the BED format.

```
Path<-/home/kellys/AfPC/EXCAVATOR_Package_v2.2/EXCAVATOR/OutputFolder/Results/
vec<-list.files(Path,recursive=T)[c(1,5,7,9,11)]
vec<-paste(Path,vec,sep=) #file path for EXCAVATOR results

BED_list<-lapply(X=vec,FUN=read.delim,header=F,skip=1) #load the EXCAVATOR data into a list
names(BED_list)<-gsub(pattern=FastCallResults_,replacement=, #fix names of the objects in the list.
                      x=gsub(pattern=_Tumor.txt,replacement=,x=basename(vec)))
# HEY! LISTEN! using nested gsub()s makes it easier to replace two patterns at once without an intermediary placeho

for(i in 1:length(BED_list)){ #remove extraneous data
  BED_list[[i]]<-BED_list[[i]][,-c(4:8)]
  write.table(x=BED_list[[i]],file=paste(names(BED_list)[i],".bed",sep=""),
              sep = "\t",quote=F,row.names=F,col.names=F)
}
rm(vec,Path,BED_list,i)
```

We can preview the BED file output with the following bash terminal command:

```
head -n 5 /home/kellys/AfPC/KENPCA001.bed

## chr1 990220 1027507
## chr1 11824301 11849543
## chr1 26799099 26856529
## chr1 27657441 27672025
## chr1 36771912 36793791
```

Next, a reference genome must be created. For this purpose, we will be using the `org.Hs.eg.db` R package which is a part of Bioconductor. This human genome database will be queried in R for annotation information for all genes, and the results will be output in the BED format, including three extra columns for gene symbol, score, and chromosomal strand. While score information is irrelevant in this case, the BED format requires that the field is filled in before strand information can be included. We will fill this column with a value of 0 for compatibility. DNA strand information will be infered by the positive or negative value of the chromosomal coordinates reported by `org.Hs.eg.db` and saved into the appropriate column of the BED file.

```
suppressPackageStartupMessages(require(org.Hs.eg.db))
# import the human genome DB
x <- org.Hs.egREFSEQ
mapped_genes <- mappedkeys(x)
# pick out the desired genome information ; ignore the Warning message
DB<-select(org.Hs.eg.db, mapped_genes, keytype="ENTREZID",columns=c(CHR,CHRLOC,CHRLOCEND,SYMBOL))
```

```
## Warning in .generateExtraRows(tab, keys, jointype): 'select' resulted in 1:many mapping
between keys and return rows
```

```
DB<-subset(DB,subset=(DB[[CHRLOC]]!=NA&DB[[CHRLOCEND]]!=NA) ) # remove genes lacking chromosomal coordinates
DB<-subset(DB,select=(-CHRLOCCHR) ) # remove this extraneous column
DB$CHR<-paste("chr",DB$CHR,sep="") # adjust the chromosome number labels
DB<-DB[,-1] #remove the ENTREZID column
DB$SCORE<-rep(x = 0,times=nrow(DB)) # fill the SCORE column with zeroes
DB$STRAND<-ifelse(DB$CHRLOC<0,"-","+") # report strand information
DB$CHRLOC<-abs(DB$CHRLOC) # return absolute value of chromosomal coordinates
DB$CHRLOCEND<-abs(DB$CHRLOCEND)

# preview the subsetted org.Hs.eg.db database within R
head(DB)

##     CHR    CHRLOC  CHRLOCEND SYMBOL SCORE STRAND
## 1 chr19  58858172  58864865   A1BG     0      -
## 2  chr8  18248755  18258723   NAT2     0      +
## 3 chr20  43248163  43280376    ADA     0      -
## 4 chr18  25530930  25757445   CDH2     0      -
## 5  chr1 243651535 244006584   AKT3     0      -
## 6  chr1 243663021 244006584   AKT3     0      -

# write out the BED file of the databse
write.table(x=DB,file="org_Hs_egREFSEQ.bed",sep = "\t",quote=F,row.names=F,col.names=F)
rm(DB,mapped_genes,x)
```

The reference genome BED file can be previewed on the command line:

```
head -n 5 /home/kellys/AfPC/org_Hs_egREFSEQ.bed

## chr19 58858172 58864865 A1BG 0 -
## chr8 18248755 18258723 NAT2 0 +
## chr20 43248163 43280376 ADA 0 -
## chr18 25530930 25757445 CDH2 0 -
## chr1 243651535 244006584 AKT3 0 -
```

Now that we have converted our samples and reference genome into BED files, we will use them in the `intersectBed` program. This program will compare our samples to the reference genome and return genes that are found in the chromosomal regions listed. It is important to note that the standard usage of `intersectBed` takes the unknown sample as the first argument and the reference genome as the second argument. For our purposes, we will enter the reference genome first and the sample second. Entering the files in either order yields the same intersected chromosomal coordinatesi n the output. However, the output includes the fields of the file entered in the first argument, allowing us to preserve the gene symbol information from the reference genome BED file by listing it first. This greatly streamlines our workflow.

We will copy the .BED files to a new directory called `BED` before processing them. If the `BED` directory already exists, it will be deleted. A directory called `BED_intersected_genes` will be created for the `intersectBed` results.

```
cd /home/kellys/AfPC
ls *.bed
rm -r BED
mkdir BED
mkdir BED_intersected_genes
```

```
FILES="*.bed"
for f in $FILES; do cp $f /home/kellys/AfPC/BED/$f; done
```

The .BED files will be proccessed with `intersectBed` and output to the `BED_intersected_genes` directory.

```
cd /home/kellys/AfPC/BED
OUTPATH="/home/kellys/AfPC/BED_intersected_genes/"
FILES="KENPCA*"
INTERSECT="/usr/local/bedtools2-master/bin//intersectBed"
for f in $FILES; do echo "$f"; $INTERSECT -a org_Hs_egREFSEQ.bed -b "$f" > $OUTPATH$f; done

## KENPCA001.bed
## KENPCA003.bed
## KENPCA004.bed
## KENPCA006.bed
## KENPCA007.bed
```

The intersections between the sample coordinates and the human genome have been saved into the `BED_intersected_genes` directory. For the next part of our analysis, we will pass the list of gene symbols found in the intersection for each sample through Webgestalt. To make this more convenient, we will load each intersect into R, and output a .TXT file that contains only a list of the gene symbols. These will be placed in the `WebGestalt_inputs` directory, since this data will be considered the input for WebGestalt analysis.

```
system("mkdir /home/kellys/AfPC/WebGestalt_inputs/") # make the directory if it doesnt already exist
FILEPATH<-/home/kellys/AfPC/BED_intersected_genes/ # intersected genes
OUTPATH<-/home/kellys/AfPC/WebGestalt_inputs/ #  folder for WebGestalt inputs
for(i in dir(FILEPATH)){
 BEDintersect<-read.delim(file=paste(FILEPATH,i,sep=""),header=F) # read in the intersected genes
 tmpNAME<-gsub(pattern = ".bed",replacement = "",x = gsub("-RESULTS","",i)) # fix the sample name
 write.table(x=BEDintersect[4],file=paste(OUTPATH,tmpNAME,".txt",sep=""), # write out the list of genes
            quote=F,row.names=F,col.names=F,sep="") # need these parameters to format correctly
 }
rm(i,tmpNAME,FILEPATH,OUTPATH,BEDintersect) #cleaning house
```

These results will now be processed with WebGestalt's website-based analysis tools. This step cannot be directly reproduced here. The following parameters will be used for WebGestalt Enrichment Analysis:

- Enrichment Analysis
- Disease Association
- hsapiens_genome
- Stat Model: Hypergeometric
- Multiple Test Adjust: BH

- Significance Level: 0.01
- Min. Number of Genes for a category: 4

Disease Association analysis returned a series of categories of diseases into which genes were place. The following disease categories have been chosen for further analysis:

- Cancer or viral infections
- Adenocarcinoma
- Adenoma
- Neoplasms

WebGestalt results were saved into .TSV files which have been collected into the directory `WebGestalt_results`, which is further divided into subdirectories labeled with their sample ID. We will load each .TSV file into R, remove extraneous information, and search for genes that we will later include in a Circos Plot for each sample. For our Circos plots, we will use the first three genes identified on each chromosome. A fresh copy of the `org.Hs.eg.db` human genome database will be loaded for this analysis.

```
suppressPackageStartupMessages(require(org.Hs.eg.db))
x <- org.Hs.egREFSEQ
mapped_genes <- mappedkeys(x)
DB<-select(org.Hs.eg.db, mapped_genes, keytype="ENTREZID",columns=c(CHR,CHRLOC,CHRLOCEND,SYMBOL))

## Warning in .generateExtraRows(tab, keys, jointype): 'select' resulted in 1:many mapping
between keys and return rows

## FIND SOME WAY TO MASK THE STUPID ERROR MESSAGE ABOUT THE 1:MANY MAPPINGS...
rm(mapped_genes,x)
```

We will now use a `for` loop to iteratively load each sample .TSV file provided by WebGestalt, remove extraneous information, and output several R objects containing properly formatted data to be used in Circos Plots.

```
# WebGestalt_results directory manually created outside of R code

DIRECTORY<-"/home/kellys/AfPC/WebGestalt_results"
SAMPLE<-dir(DIRECTORY) #[1] "KENPCA001" "KENPCA003" "KENPCA004" "KENPCA006" "KENPCA007"

#our diseases of interest
ChosenDiseases<-c("Adenocarcinoma","Adenoma","cancer or viral infections","Neoplasms")

## the BED_list object needs to exist for the alternate gene selection codes that are listed below
## this object was already created in a previous chunk, consider keeping that copy around
## instead of re-creating it here
# BED_list<-lapply(X=vec,FUN=read.delim,header=F,skip=1, #load the EXCAVATOR data into a list
#                  col.names=c(CHR,CHRLOC,CHRLOCEND,"Segment","CNF","CN","Call","ProbCall"))
#
# names(BED_list)<-gsub(pattern=FastCallResults_,replacement=, #fix names of the objects in the list.
#                       x=gsub(pattern=_Tumor.txt,replacement=,x=basename(vec)))
#
#



#### These steps process the WebGestalt results
```

```
for(a in 1:length(dir(DIRECTORY))){
  FILE<-dir(paste(DIRECTORY,SAMPLE[a],sep="/")) # WebGestalt output file
  FILEPATH<-paste(DIRECTORY,SAMPLE[a],FILE,sep="/") # full path to the file
  print(FILEPATH)
 TMP<-read.delim(file =FILEPATH,header = F,sep = "\t",fill = T,skip=10) # load the WebGestalt .TSV
  TMP<-TMP[-grep("C=",TMP$V1,value=F),] #remove the category scores first

  for(i in 1:nrow(TMP)){ # replace the NAs with the disease category
    if(is.na(TMP$V2[i])){
      TMP$V2[i]<-TMP$V2[i-1]
    }
  }

  TMP<-TMP[-grep("DB_ID:",TMP$V3,value=F),] #remove the disease category scores

  TMP2<-data.frame() # placeholder data frame

  for(i in 1:length(ChosenDiseases)){  # Find ONLY results that match the diseases of interest
    TMP2<-rbind(TMP2,TMP[grep(ChosenDiseases[i],TMP$V2,fixed=T,value=F),][c(2,1,5)])
    # use grep here because it returns ALL types of Neoplasm; there are many
  }
  TMP2<-TMP2[!duplicated(TMP2[,c(2,3)]),] # remove duplicate genes in columns 2 and 3
  colnames(TMP2)<-c("DISEASE","SYMBOL","ENTREZID") #fix the column names
 DB2<-DB[match(TMP2$ENTREZID,DB$ENTREZID),] # find sample gene ID matches in the reference Database

  DB2<-DB2[,c("CHR","CHRLOC","CHRLOCEND","SYMBOL")] # just use these columns for the plot
  DB2$CHRLOC<-abs(DB2$CHRLOC)  # use the abs() of the location; strand doesnt matter here
  DB2$CHRLOCEND<-abs(DB2$CHRLOCEND)
 write.table(DB2,file = paste("/home/kellys/AfPC/",SAMPLE[a],"_Selected_Genes_For_Plot.TSV",sep=""),
              quote = F,sep = "\t",row.names = F,col.names = T)


  mygenes<-data.frame() # placeholder

############################################################   ###############################
############################## Under Construction... ###################

  # alternate way to select genes for plotting

  # return the first gene for each CNV
  # for each CNV in the corresponding BED_list entry...
  # select the Chromosome Start value...
 # search the DB2 (Hs.org.db) under the **corresponding Chromosome** for the Chromosome location value closest to
 # the number can be greater than the Start value but cannot be less...? Lets go with that condition for right now
 ### for some reason ^^ this did not work in our first example test w/ KENPCA001, but worked using the opposite app
  ### e.g. DB2 start value <= EXCAVATOR start value... ???

  # return the first gene that matches this, rbind into mygenes

  ##################
  # trying to write code that implements the above procedure:
  # these codes can be run but will not produce useful output until they are mature
  ##########
  # Load up the original EXCAVATOR OUTPUT ; this is already done at the beginning of this chunk

#  BED_list[[SAMPLE[a]]]$CHR<-gsub(pattern = "chr", # need to fix the Chromosome values here to match DB2
#                                          replacement = "",
#                                          x = BED_list[[SAMPLE[a]]]$CHR)
#
#   # Load up the intersected genes; have ChrLOCs & Gene Names already.... may be easier to search
#
#
#   INTERSECTGENES<-/home/kellys/AfPC/BED_intersected_genes/ # intersected genes
# # for(FILE in dir(INTERSECTGENES)){
#  BEDintersect<-read.delim(file=paste(INTERSECTGENES,FILE,sep=""),header=F) # read in the intersected genes
#  tmpNAME<-gsub(pattern = ".bed",replacement = "",x = gsub("-RESULTS","",FILE)) # fix the sample name
#  write.table(x=BEDintersect[4],file=paste(OUTPATH,tmpNAME,".txt",sep=""), # write out the list of genes
#            quote=F,row.names=F,col.names=F,sep="") # need these parameters to format correctly
#  }
#  for(i in 1:nrow(BED_list[[SAMPLE[a]]])){ # for each CNV entry.. which is the first gene in the reference databa
#
#     ## the following codes were tested to try to implement this ^^^
#     ## these codes are not finished yet...
#
#   DB2[which(DB2[[CHRLOC]]<=BED_list[[SAMPLE[a]]][[CHRLOC]][i] & DB2[[CHR]]==BED_list[[SAMPLE[a]]][["CHR"]][
#
#
#     findInterval(x = DB2,
#            vec = BED_list[[SAMPLE[a]]][which(BED_list[[SAMPLE[a]]]$CHR==BED_list[[SAMPLE[a]]]$CHR[i]),]
#
#
#                  within(BED_list[[SAMPLE[a]]],
#                                    BED_list[[SAMPLE[a]]]$CHR==BED_list[[SAMPLE[a]]]$CHR[i])
#                                    BED_list[[SAMPLE[a]]][CHRLOC]


#     with(DB2, CHR==BED_list[[SAMPLE[a]]]["CHR"])# BED_list[[SAMPLE[a]]]["CHR"])
#     DB2<-DB[match(TMP2$ENTREZID,DB$ENTREZID),] # find sample gene ID matches in the reference Database
#     DB2[match(BED_list[[SAMPLE[a]]]$CHR[i],DB2$CHR),] # returns only 1st entry...
#    D B2[DB2[[CHR]]==BED_list[[SAMPLE[a]]]$CHR[i] & ,]
#     head(BED_list[[SAMPLE[a]]])
#     head(DB2)

    # PROBLEMS
    # match, %in%, only returns first found value
    # grep returns ALL found matches; grep for "1" also returns "11" and "21"
   ## there is a solution to this on StackOverflow that includes non-character end-line symbols in the search patte
   # do the Intersect gene coordinates line up with the CNVs? With the Reference genome chr. start values?

#   }

###########################  ^^^^^^^^ working on the alternate gene selection mechanism ^^^^^^^^^^ #############
#################################################################################
##########################

# The following code sections work and simply return the return three genes per chromosome
```

```
  for(i in 1:length(unique(DB2[[CHR]]))){
    if(nrow(DB2[DB2$CHR==unique(DB2[[CHR]])[i],])<3){ # if fewer than 3 genes in the CHR
      LENGTH<-nrow(DB2[DB2$CHR==unique(DB2[[CHR]])[i],])
    mygenes<-rbind(mygenes,DB2[DB2$CHR==unique(DB2[[CHR]])[i],][1:LENGTH,]) # add the genes to the list
      } else {
        LENGTH<-3 # take only 3 genes in the CHR
     mygenes<-rbind(mygenes,DB2[DB2$CHR==unique(DB2[[CHR]])[i],][1:LENGTH,]) # add the genes to the list
      }
  }
  assign(paste0(SAMPLE[a]),mygenes) #make an R object named after the sample to hold the data
  rm(mygenes,TMP,TMP2,DB2,FILE,FILEPATH,LENGTH)
}
```

```
## [1] "/home/kellys/AfPC/WebGestalt_results/KENPCA001/final_disease_geneset_file_1424281770.tsv"
## [1] "/home/kellys/AfPC/WebGestalt_results/KENPCA003/final_disease_geneset_file_1424281880.tsv"
## [1] "/home/kellys/AfPC/WebGestalt_results/KENPCA004/final_disease_geneset_file_1424289302.tsv"
## [1] "/home/kellys/AfPC/WebGestalt_results/KENPCA006/final_disease_geneset_file_1424289360.tsv"
## [1] "/home/kellys/AfPC/WebGestalt_results/KENPCA007/final_disease_geneset_file_1424289421.tsv"
```

```
rm(ChosenDiseases,DIRECTORY,SAMPLE,a,i,DB)

# list the objects we have created
ls() # [1] "KENPCA001" "KENPCA003" "KENPCA004" "KENPCA006" "KENPCA007"
```

```
## [1] "KENPCA001" "KENPCA003" "KENPCA004" "KENPCA006" "KENPCA007"
```

```
## ^^^ THIS ONLY MAKES SENSE IF THE rm() COMMANDS THROUGHOUT THE DOCUMENT HAVE BEEN RUN
## THESE COMMANDS MAY BE COMMENTED OUT FOR DEBUGGING PURPOSES
## SO FIX THIS IN THE FINAL VERSION OF THE DOCUMENT; THIS WILL MAKE IT MORE CLEAR
## TO THE READER  WHAT OUR DATA STRUCTURE WITHIN R IS LIKE

# preview one of the objects
print(ls()[1]) # [1] "KENPCA001"
```

```
## [1] "KENPCA001"
```

```
head(get(ls()[1]))
```

```
##       CHR   CHRLOC CHRLOCEND SYMBOL
## 11157   1 78470636  78482995 DNAJB4
## 21693   1 86046444  86049648  CYR61
## 17995   1 95699711  95712781  RWDD3
## 14086  16 50775961  50835846   CYLD
## 14809  16 69743304  69760533   NQO1
## 40319  16 68771195  68869444   CDH1
```

```
#       CHR   CHRLOC CHRLOCEND SYMBOL
# 11157   1 78470636  78482995 DNAJB4
# 21693   1 86046444  86049648  CYR61
# 17995   1 95699711  95712781  RWDD3
# 14086  16 50775961  50835846   CYLD
# 14809  16 69743304  69760533   NQO1
# 40319  16 68771195  68869444   CDH1

## AS PER ABOVE, THESE LINES ONLY WORK IF THE rm() COMMANDS HAVE CLEARED OUT ALL OTHER OBJECTS
## THE INTENT HERE IS TO SHOW THE READER WHAT OUR OUTPUT OBJECTS LOOK LIKE,
## HOW THEY ARE NAMED WITHIN R, AND HOW THE DATA IS FORMATTED
## FOR EASE OF REPLICATION AND REPRODUCIBILITY BY OTHERS
```

2 Generate Circos Plot

Write goes here. See v3 of the document for the writeup and stuff

2.1 Write CNV Plot Code

This section was written by `baqirm`. We expanded the capabilities of RCircos to work with CNV data by adapting the existing `RCircos.Heatmap.Plot` function. This function takes as arguments the data frames created by importing Excavatoroutput to R:

```
##### This chunk sets up the custom function that creates the Circos Plot for CNVs
##### thanks to baqirm for writing this one

# heatmap.data = the object holding the sample EXCAVATOR data
# data.col = the column of the EXCAVATOR data to plot
# nrow(heatmap.data) = the number of CNVs in the sample
# colors, plotted CNV values have been adjusted from original
# Now plotting the Call value from EXCAVATOR CNV results
# only showing the 2,1 copy deletions, and 1,>1 copy amplifications ; "Copy Number State"
## other details are irrelevant
# consider removing reference genomes completely!
# look into code controling the colored lines outside each chromosome
# consider finding a way to increase the gap between chromosomes for more clarity

# in order to understand how this function is working, start with the
# argument you dont understand and follow it backwards through its creation

suppressPackageStartupMessages(require(RCircos))
RCircos.CNV.Plot<-function(heatmap.data, data.col, track.num, side) {
  RCircos.Cyto      <- RCircos.Get.Plot.Ideogram()
  RCircos.Pos       <- RCircos.Get.Plot.Positions()
 RCircos.Par     <- RCircos.Get.Plot.Parameters() # all the params of the initialized Core components
 heatmap.data    <- RCircos.Get.Plot.Data(heatmap.data, "plot") # adds plotting locations to the EXCAVATOR CNVs
 heatmap.locations <- as.numeric(heatmap.data[, ncol(heatmap.data)]) # just the plotting locations from above
  start             <- heatmap.locations - RCircos.Par$heatmap.width/2
  end               <- heatmap.locations + RCircos.Par$heatmap.width/2
  data.chroms       <- as.character(heatmap.data[, 1]) # vector containing chromosome names
  chromosomes       <- unique(data.chroms) # the unique vector of chromosome names
  cyto.chroms      <- as.character(RCircos.Cyto$Chromosome) # length 862 of chromosome names (?)
  for (i in 1:length(chromosomes))  { # 1 to 24
        cyto.rows <- which(cyto.chroms == chromosomes[i])
        locations <- as.numeric(RCircos.Cyto$Location[cyto.rows])
        chr.start <- min(locations) - RCircos.Cyto$Unit[cyto.rows[1]]
        chr.end   <- max(locations)
        data.rows <- which(data.chroms == chromosomes[i])
        start[data.rows[start[data.rows] < chr.start]] <- chr.start
        end[data.rows[end[data.rows] > chr.end]]       <- chr.end
  }
  locations <- RCircos.Track.Positions(side, track.num)
  out.pos   <- locations[1]
  in.pos    <- locations[2]
  chroms    <- unique(RCircos.Cyto$Chromosome)
 for (i in 1:length(chroms)) { # 1 thru 24. I think this just builds the borders of the chromosomes.
    the.chr   <- RCircos.Cyto[RCircos.Cyto$Chromosome == chroms[i],]
    the.start <- the.chr$Location[1] - the.chr$Unit[1] + 1
    the.end   <- the.chr$Location[nrow(the.chr)]
   polygon.x <- c(RCircos.Pos[the.start:the.end, 1] * out.pos, RCircos.Pos[the.end:the.start, 1] * in.pos)
   polygon.y <- c(RCircos.Pos[the.start:the.end, 2] * out.pos, RCircos.Pos[the.end:the.start, 2] * in.pos)
    polygon(polygon.x, polygon.y, col = "white")#was azure2
  }
  for (a.point in 1:nrow(heatmap.data) )  { # iterate through each CNV in the list
    if ( heatmap.data[a.point,data.col]==-2 ) { # evaluate the Copy Number State
      assign(cell.color,purple1)
      } else if ( heatmap.data[a.point,data.col]==-1) {
        assign(cell.color,royalblue)
        } else if ( heatmap.data[a.point,data.col]==1) {
          assign(cell.color,orange)
          } else if ( heatmap.data[a.point,data.col]==2) {
            assign(cell.color,red)
          } else assign(cell.color,white)
    the.start <- start[a.point]
    the.end <- end[a.point]
   polygon.x <- c(RCircos.Pos[the.start:the.end, 1] * out.pos, RCircos.Pos[the.end:the.start, 1] * in.pos)
   polygon.y <- c(RCircos.Pos[the.start:the.end, 2] * out.pos, RCircos.Pos[the.end:the.start, 2] * in.pos)
    polygon(polygon.x, polygon.y, col = cell.color, border = NA)
  }
}
```

2.2 Create RCircos Plot

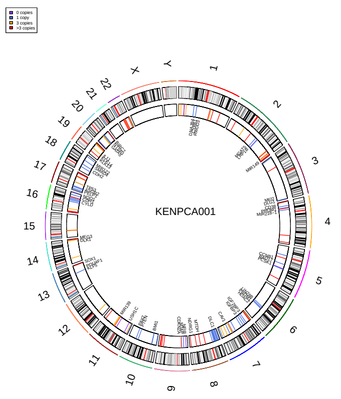
This code builds the Circos plot showing the areas of the genome affected by CNVs in our samples. The actual CNV intervals are probably larger than those shown, because we sequenced only the exomes.

```
suppressPackageStartupMessages(require(RCircos)) # get the citation information on this
data(UCSC.HG19.Human.CytoBandIdeogram) # and write the write up on this ??
cyto.info<-UCSC.HG19.Human.CytoBandIdeogram # see v3 documentatioon on this ??

Path<-/home/kellys/AfPC/EXCAVATOR_Package_v2.2/EXCAVATOR/OutputFolder/Results/
vec<-list.files(Path,recursive=T)[c(1,5,7,9,11)] # These are the ones we want to use
vec<-paste(Path,vec,sep=) #file path for EXCAVATOR results

DIRECTORY<-"/home/kellys/AfPC/WebGestalt_results"
SAMPLE<-dir(DIRECTORY)

for(i in c(1,2,4,5)){
  print(SAMPLE[i])
  pdf(file=paste0("/home/kellys/AfPC/CircosPlot_A_",SAMPLE[i],".PDF"),width=6,height=7)
 RCircos.Set.Core.Components(cyto.info=cyto.info,chr.exclude=NULL,tracks.inside=3,tracks.outside=0)
  RCircos.Set.Plot.Area()
  RCircos.Chromosome.Ideogram.Plot() # THIS START MAKING A PLOT
  # here is the connector codes, uneditted from original document
  #RCircos.Gene.Connector.Plot2(genomic.data=CircosGenes,track.num=1,side=out)
#RCircos.Gene.Connector.Plot(genomic.data=CircosGenes,track.num=1,side=in)
  RCircos.Gene.Name.Plot(gene.data=get(SAMPLE[i]),name.col=4,track.num=2,side=in)
 tmp<-read.delim(vec[i]) #  Excavator FastCall Results .TXT, this makes the heatmap data for the sample
RCircos.CNV.Plot(heatmap.data=tmp,data.col=7,track.num=1,side=in )
  legend(topleft,
```



(a) KENPCA001

```
          legend=c(0 copies,
                   1 copy,
                   3 copies,
                   >3 copies),
          cex=0.4,
          fill=c(purple1,royalblue,orange,red),border=black)
  legend("center",legend=SAMPLE[i],bty = "n")
  dev.off()
}

## [1] "KENPCA001"
##
## RCircos.Core.Components initialized.
## Type ?RCircos.Reset.Plot.Parameters to see how to modify the core components.
##
## Maximum lables for  chr19 is 3 . Extra ones are ignored.
##
## [1] "KENPCA003"
##
## RCircos.Core.Components initialized.
## Type ?RCircos.Reset.Plot.Parameters to see how to modify the core components.
##
##
## [1] "KENPCA006"
##
## RCircos.Core.Components initialized.
## Type ?RCircos.Reset.Plot.Parameters to see how to modify the core components.
##
## Maximum lables for  chr21 is 3 . Extra ones are ignored.
##
## [1] "KENPCA007"
##
## RCircos.Core.Components initialized.
## Type ?RCircos.Reset.Plot.Parameters to see how to modify the core components.
## Some chromosomes are in genomic data only and have been removed.
```

3 System and Session Information

```
system(uname -srv,intern=T)

## [1] "Linux 3.13.0-43-generic #72~precise1-Ubuntu SMP Tue Dec 9 12:14:18 UTC 2014"

sessionInfo()

## R version 3.1.2 (2014-10-31)
## Platform: x86_64-pc-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8 LC_NUMERIC=C         LC_TIME=C
##  [4] LC_COLLATE=C         LC_MONETARY=C        LC_MESSAGES=C
##  [7] LC_PAPER=C           LC_NAME=C            LC_ADDRESS=C
## [10] LC_TELEPHONE=C       LC_MEASUREMENT=C     LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats4    stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] RCircos_1.1.2        org.Hs.eg.db_3.0.0   RSQLite_1.0.0
##  [4] DBI_0.3.1            AnnotationDbi_1.28.1 GenomeInfoDb_1.2.4
##  [7] IRanges_2.0.1        S4Vectors_0.4.0      Biobase_2.26.0
## [10] BiocGenerics_0.12.1  knitr_1.9
##
## loaded via a namespace (and not attached):
## [1] evaluate_0.5.5 formatR_1.0    highr_0.4      stringr_0.6.2
## [5] tools_3.1.2
```