CODEX2: Full-spectrum copy number variation detection by high-throughput DNA sequencing

Yuchao Jiang

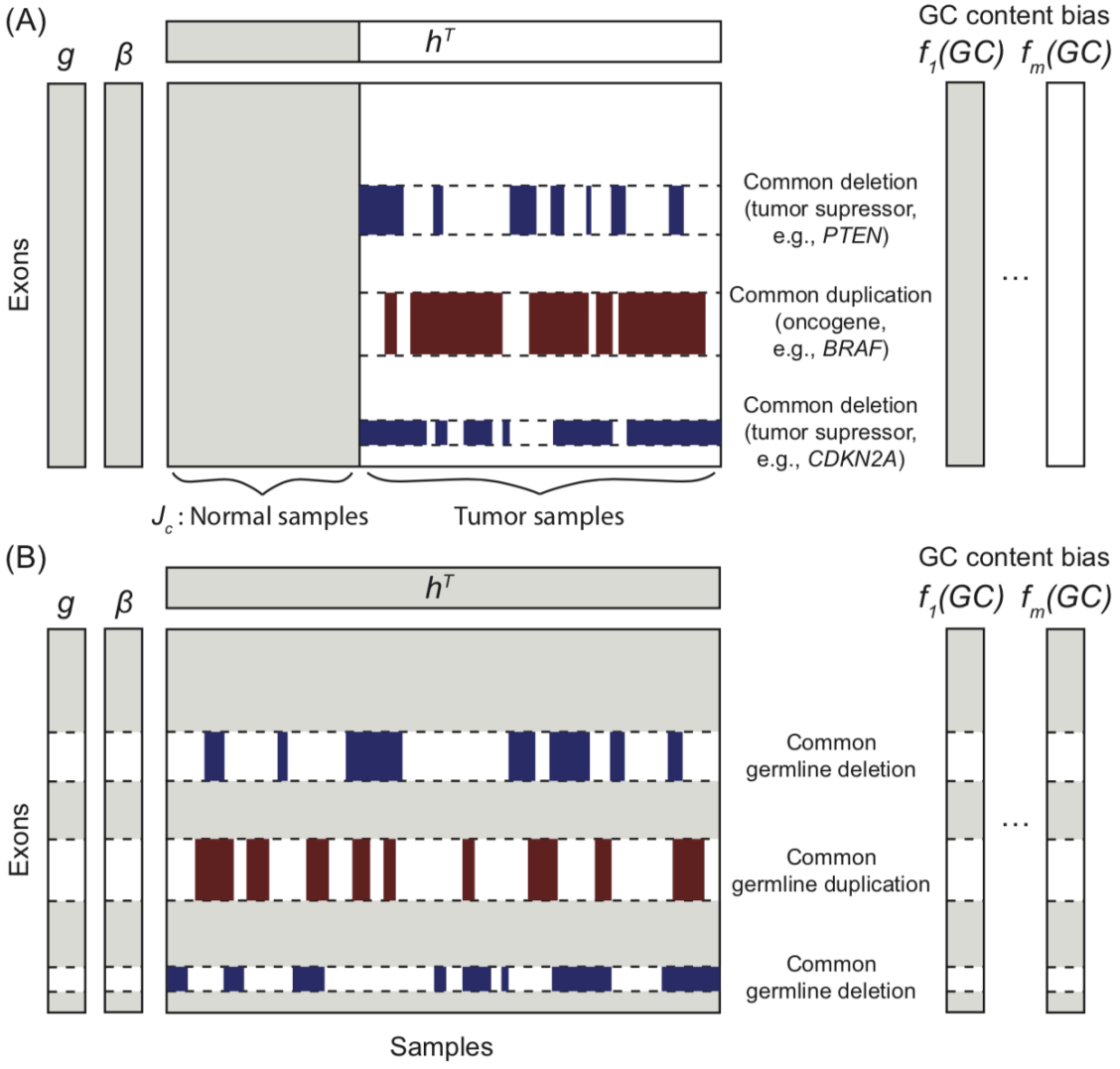
2018-07-04

Abstract

High-throughput DNA sequencing enables detection of copy number variations (CNVs) on the genome-wide scale with finer resolution compared to array-based methods, but suffers from biases and artifacts that lead to false discoveries and low sensitivity. We describe CODEX2, a statistical framework for full-spectrum CNV profiling that is sensitive for variants with both common and rare population frequencies and that is applicable to study designs with and without negative control samples. We demonstrate and evaluate CODEX2 on whole-exome and targeted sequencing data, where biases are the most prominent. CODEX2 outperforms existing methods and, in particular, significantly improves sensitivity for common CNVs.

Analysis overview

The figure below illustrates the two experimental designs for which CODEX2 can be applied: (i) case-control design with a group of negative control samples, where the goal is to detect CNVs disproportionately present in the 'cases' versus the 'controls'; and (ii) detection of all CNVs present in all samples design, such as in the Exome Aggregation Consortium. The key innovation in CODEX2 is the usage of negative control genome regions in a genome-wide latent factor model for sample- and position-specific background correction, and the utilization of negative control samples, under a case-control design, to further improve background bias estimation under this model. The negative control genome regions defined by CODEX2 are regions that do not harbor common CNVs, but that are still allowed to harbor rare CNVs, and can be constructed from existing studies or learned from data.



Installation

```
source("https://bioconductor.org/biocLite.R")
biocLite("WES.1KG.WUGSC")

devtools::install_github("yuchaojiang/CODEX2/package")
```

Pre-computation and quality control

Pre-processing

This step is to get directories of .bam files, read in exon target positions from .bed files, and get sample names. The direct input of CODEX2 include: *bamdir*, which is a vector indicating the directories of all .bam files; *sampname*, which is a column vector with row entries of sample names; *bedFile*, which indicates the directory of the .bed file (WES bait file, no header, sorted by start and end positions); and *chr*, which specifies the chromosome. CODEX2 processes the entire genome chromosome by chromosome; make sure the chromosome formats are consistent between the .bed and the .bam files.

```
library(CODEX2)
library(WES.1KG.WUGSC) # Load Toy data from the 1000 Genomes Project.
dirPath <- system.file("extdata", package = "WES.1KG.WUGSC")
bamFile <- list.files(dirPath, pattern = '*.bam$')
bamdir <- file.path(dirPath, bamFile)
sampname <- substr(bamFile,1,7)
bedFile <- file.path(dirPath, "chr22_400_to_500.bed")
bambedObj <- getbambed(bamdir = bamdir, bedFile = bedFile,
                       sampname = sampname, projectname = "CODEX2_demo")
bamdir <- bambedObj$bamdir; sampname <- bambedObj$sampname
ref <- bambedObj$ref; projectname <- bambedObj$projectname
```

Getting GC content and mappability

Obtain GC content and mappability for each exon/target/window. Mappability is calculated from the hg19 mappability from ENCODE (link (http://rohsdb.cmb.usc.edu/GBshape/cgi-bin/hgFileUi?db=hg19&g=wgEncodeMapability)). Mappability for hg38 is lifted over from hg19 (link (http://hgdownload.cse.ucsc.edu/goldenPath/hg19/liftOver/)).

```
gc <- getgc(ref, genome = BSgenome.Hsapiens.UCSC.hg19)
mapp <- getmapp(ref, genome = BSgenome.Hsapiens.UCSC.hg19)
values(ref) <- cbind(values(ref), DataFrame(gc, mapp))
```

Getting raw read depth

Read depth matrix, as well as read lengths across all samples, will be returned. This will need to be generated for each chromosome.

```
coverageObj <- getcoverage(bambedObj, mapqthres = 20)
Y <- coverageObj$Y
write.csv(Y, file = paste(projectname, '_coverage.csv', sep=''), quote = FALSE)
head(Y[,1:5])
```

```
##                     NA06994 NA10847 NA11840 NA12249 NA12716
## 22:21345867-21346168       5       9       7      11      11
## 22:21346453-21346708      52      53      54      56      70
## 22:21347033-21347243      11      15       8      12       8
## 22:21347901-21348093      12      19      21      19      23
## 22:21348163-21348608      25      22      39      31      27
## 22:21348797-21349066      57      61      66      65      64
```

Quality control

Take a sample-wise and exon-wise quality control procedure on the depth of coverage matrix.

```
qcObj <- qc(Y, sampname, ref, cov_thresh = c(20, 4000),
            length_thresh = c(20, 2000), mapp_thresh = 0.9,
            gc_thresh = c(20, 80))
Y_qc <- qcObj$Y_qc; sampname_qc <- qcObj$sampname_qc
ref_qc <- qcObj$ref_qc; qcmat <- qcObj$qcmat; gc_qc <- ref_qc$gc
write.table(qcmat, file = paste(projectname, '_qcmat', '.txt', sep=''),
            sep = '\t', quote = FALSE, row.names = FALSE)
```

Running CODEX2

For demonstration purpose, we in silico spiked in CNVs spanning exon 1580 - 1620 with a population frequency 40%. There are altogether 90 samples, 36 of which have the heterozygous deletion. The toy dataset is stored as part of the CODEX2 R-package.

```
# Load pre-stored data.
Y_qc <- Y_qc_demo
ref_qc <- ref_qc_demo
gc_qc <- ref_qc$gc
```

Estimate library size factor based on genome-wide read depth after QC. It is important, especially in cancer genomics, to calculate library size factor using genome-wide data, as chromosomal read depth can be attentuated by large copy number aberrations.

```
Y.nonzero <- Y_qc[apply(Y_qc, 1, function(x){!any(x==0)}),]
pseudo.sample <- apply(Y.nonzero,1,function(x){prod(x)^(1/length(x))})
N <- apply(apply(Y.nonzero, 2, function(x){x/pseudo.sample}), 2, median)
```

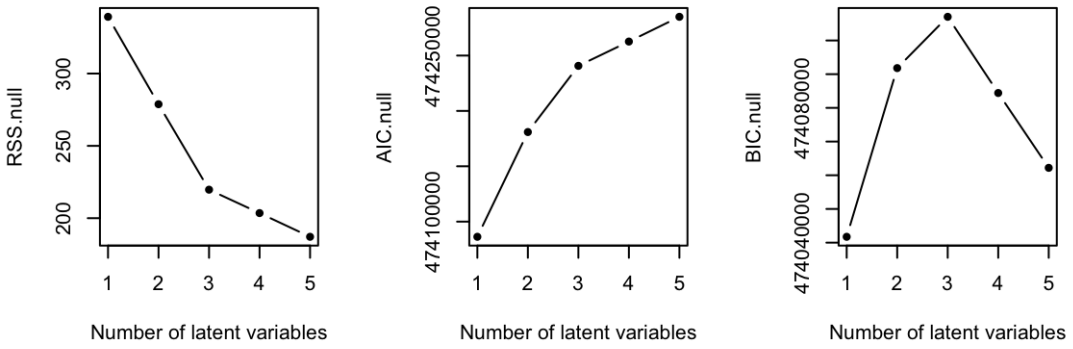
Running CODEX2 without specifying negative control samples/regions

Y_qc and gc_qc can be obtained from the sequencing bam files using the code in the previous section.

```
chr <- 20 # This can be run for one chromosome or multiple chromosomes
chr.index <- which(seqnames(ref_qc)==chr)
normObj.null <- normalize_null(Y_qc = Y_qc[chr.index,],
                               gc_qc = gc_qc[chr.index],
                               K = 1:5, N = N)
Yhat.null <- normObj.null$Yhat
AIC.null <- normObj.null$AIC; BIC.null <- normObj.null$BIC
RSS.null <- normObj.null$RSS
```

Choose the number of latent Poisson factors. BIC is used as the model selection metric by default.

```
choiceofK(AIC.null, BIC.null, RSS.null, K = 1:5 , filename = "codex2_null_choiceofK.pdf")
```



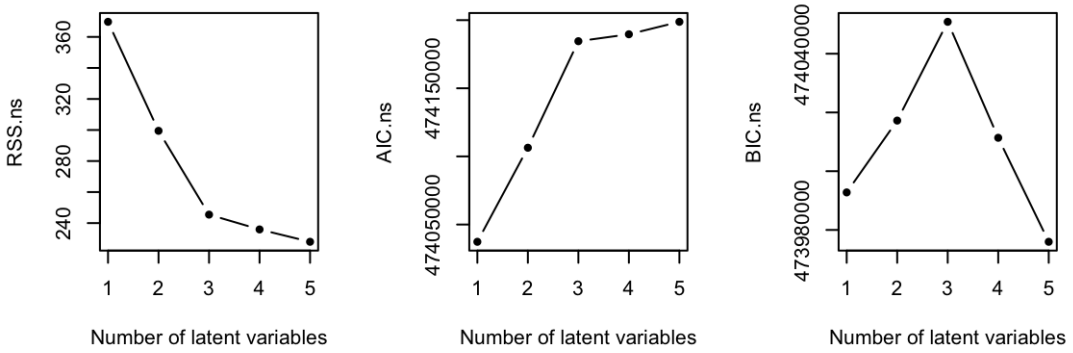
Running CODEX2 with negative control samples

For the case-control scenario, the normal sample index is known (samples without spike-in signals).

```
# Below are pre-computed demo dataset, stored as part of the CODEX2 R-package.
normObj <- normalize_codex2_ns(Y_qc = Y_qc[chr.index,],
                               gc_qc = gc_qc[chr.index],
                               K = 1:5, norm_index = norm_index_demo,
                               N = N)
Yhat.ns <- normObj$Yhat; fGC.hat.ns <- normObj$fGC.hat;
beta.hat.ns <- normObj$beta.hat; g.hat.ns <- normObj$g.hat; h.hat.ns <- normObj$h.hat
AIC.ns <- normObj$AIC; BIC.ns <- normObj$BIC; RSS.ns <- normObj$RSS
```

Choose the number of latent Poisson factors. BIC is used as the model selection metric by default.

```
choiceofK(AIC.ns, BIC.ns, RSS.ns, K = 1:5 , filename = "codex2_ns_choiceofK.pdf")
```



Running CODEX2 with negative control regions

We can empirically identify common CNV regions by a first-pass CODEX run: For exons residing in common CNV regions, the s.d. of normalized z-scores (using normalize_null) across all samples will be large. This can also be provided by the user as known, e.g., from existing database (DGV or dbVar) or knowledge (tumor supressors or oncogenes with recurrent CNA changes).

```
cnv_index <- 1580:1620
normObj <- normalize_codex2_nr(Y_qc = Y_qc[chr.index,], gc_qc = gc_qc[chr.index],
                               K = 1:5, cnv_index = cnv_index,
                               N = N)
Yhat.nr <- normObj$Yhat; fGC.hat.nr <- normObj$fGC.hat;
beta.hat.nr <- normObj$beta.hat; g.hat.nr <- normObj$g.hat; h.hat.nr <- normObj$h.hat
AIC.nr <- normObj$AIC; BIC.nr <- normObj$BIC; RSS.nr <- normObj$RSS
```

Running segmentation by CODEX2

We offer two versions of segmentation procedures: poisson-likelihood based recursive segmentation (recommended) and hidden Markov model (not recommended). For CODEX2 with negative control regions, simply change Yhat.ns to Yhat.nr in the code below.

```
finalcall.CBS <- segmentCBS(Y_qc[chr.index,],  # recommended
                            Yhat.ns, optK = which.max(BIC.ns),
                            K = 1:5,
                            sampname_qc = paste('sample',1:ncol(Y_qc),sep=''),
                            ref_qc = ranges(ref_qc)[chr.index],
                            chr = chr, lmax = 400, mode = "integer")

finalcall.HMM <- segmentHMM(Y_qc[chr.index,],  # not recommended
                            Yhat.ns, optK = which.max(BIC.ns),
                            K = 1:5,
                            sampname_qc = paste('sample',1:ncol(Y_qc),sep=''),
                            ref_qc = ranges(ref_qc)[chr.index],
                            chr = chr, mode = "integer")
```

Post-segmentation pruning and filtering are recommended based on CNV length (filter1), length per exon (filter2), likelihood ratio (filter3), and number of exons (filter4).

```
filter1 <- finalcall.CBS$length_kb<=200
filter2 <- finalcall.CBS$length_kb/(finalcall.CBS$ed_exon-finalcall.CBS$st_exon+1)<50
finalcall.CBS.filter <- finalcall.CBS[filter1 & filter2, ]

filter3 <- finalcall.CBS.filter$lratio>40
filter4 <- (finalcall.CBS.filter$ed_exon-finalcall.CBS.filter$st_exon)>1
finalcall.CBS.filter=finalcall.CBS.filter[filter3|filter4,]
```