CNVrd2: A package for measuring gene copy number, identifying SNPs tagging copy number variants, and detecting copy number polymorphic genomic regions

Hoang Tan Nguyen\textsuperscript{1,2}, Tony R Merriman\textsuperscript{1} and Michael A Black\textsuperscript{1}

\textsuperscript{1}Department of Biochemistry, University of Otago
\textsuperscript{2}Department of Mathematics and Statistics, University of Otago

October 17, 2016

Contents

1 Introduction 2

2 Getting started 3
  2.1 Measuring FCGR3B CN  ................................................................. 5
    2.1.1 CNVrd2 object ................................................................. 5
    2.1.2 Count reads in windows ......................................................... 5
    2.1.3 Segmentation ........................................................................... 6
    2.1.4 Obtain copy-number count ....................................................... 6
    2.1.5 Plots ....................................................................................... 11
  2.2 Identifying tag SNPs/INDELs for FCGR3B CNVs ............................... 11

3 Working with complex loci 13
  3.1 Measuring CCL3L1 CN ................................................................. 15
  3.2 Identifying tag-SNPs for CCL3L1 CNVs ........................................... 21

4 Identifying polymorphic regions 24

5 Note 27

6 Session information 27
Introduction

The CNVrd2 package (Nguyen et al., 2014) utilizes next-generation sequencing (NGS) data to measure human-gene copy number (CN) and identify single-nucleotide polymorphisms (SNPs), and insertions and deletions (INDELs) that are in linkage disequilibrium with a gene of interest. Typically, the data being used are low- or medium-coverage whole genome sequence (WGS) data from multiple individuals in a population. Such data comprise collections of sequence reads that have been aligned (or "mapped") to an appropriate reference genome. Changes in read depth (i.e., the number of reads aligned to a specific region of the genome) can indicate changes in DNA copy number in this region (i.e., deletions or duplications of specific portions of DNA). If this region encompasses a gene, then changes in copy number may also be reflected by changes in gene activity - such changes have been shown to be associated with altered risk of disease in human populations, and altered trait distributions in agricultural settings.

To measure gene CN, CNVrd2 firstly divides a region (usually at least 1Mb) flanking a gene of interest into constant-sized windows, and counts reads mapped in these windows. Next, these read-count windows are transformed and standardized. After that, the DNAcopy package (Venkatraman and Olshen, 2007) is used to join the per-window standardized counts into regions (or "segments") of similar values. The package then refines the segmentation step and outputs segmentation results, namely segmentation scores (SS), for each sample. A function in the CNVrd2 package is then used to group SSs into copy-number groups.

To calculate linkage disequilibrium (LD) between gene CNVs and SNPs/INDELs nearby, SNPs/INDELs are coded into numeric values (0, 1, 2) and Fisher’s Exact Test is used to assess associations between SNPs/INDELs and copy number. CNVrd2 is designed to identify SNPs/INDELs that can be used as a surrogate marker for CNVs, therefore multiple samples are needed to obtain reliable results. The package also uses distribution quantiles to identify highly polymorphic regions of the genome (within a collection of samples) and can identify regions with variable polymorphism between populations. The BAM format (Li et al., 2009) for aligned-NGS data and VCF format (Danecek et al., 2011) for structural variant information are used as the main forms of input for the package.

1CNVrd2 is an improved version of the pipeline CNVrd used to identify tagSNPs of FCGR3A/B CNV
Getting started

First, we load the package in our R session. Note that the rjags package (Plummer, 2013) requires the associated JAGS application to be installed.

```
library('CNVrd2')
```

```r
## Loading required package: VariantAnnotation
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##  clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##  clusterExport, clusterMap, parApply, parApplyLB, parLapply,
##  parLapplyLB, parRapply, parSapply, parSapplyLB
## The following objects are masked from 'package:stats':
##
##  IQR, mad, stabs
## The following objects are masked from 'package:base':
##
##  Filter, Find, Map, Position, Reduce, anyDuplicated, append,
##  as.data.frame, cbind, colnames, do.call, duplicated, eval,
##  evalq, get, grep, grepl, intersect, is.unsorted, lapply,
##  lengths, maply, match, mget, order, paste, pmax, pmax.int,
##  pmin, pmin.int, rank, rbind, rownames, sapply, setdiff,
##  sort, table, tapply, union, unique, unsplit, which,
##  which.max, which.min
## Loading required package: GenomeInfoDb
## Loading required package: S4Vectors
## Loading required package: stats4
##
## Attaching package: 'S4Vectors'
## The following objects are masked from 'package:base':
##
```

3
Working with BAM and VCF files.
The following section describes the workflow of the \textit{CNVrd2} package in reading BAM and VCF files into R. The 58 MXL-sample BAM files (chr1:161100000-162100000) were downloaded from the 1000 Genomes Project to measure copy number counts of \textit{FCGR3B} gene (chr1:161592986-161601753). Users can download the file \textit{MXLexample.zip} on 

\url{http://code.google.com/p/cnvrdfortagsnps/downloads/list} and unzip it into a directory.

Alternatively, to run the example without downloading the associated BAM files, users can skip to section 2.1.3 to load a pre-processed version of the same data.

**Measuring FCGR3B CN**

\textbf{CNVrd2 object}

We need to make an object of class \textit{CNVrd2} to define a region we want to investigate (regions sized $>1$Mb tend to work well - multiple genes can be included by specifying the start and end positions of each). Here, we choose 1000bp-constant windows. We also need to supply a directory that consists of BAM files including only mapped reads. Users who have \textbf{not downloaded the BAM files}, should skip to section 2.1.3.

```r
objectCNVrd2 <- new("CNVrd2", windows = 1000, chr = "chr1",
                  st = 161100001, en = 162100000,
                  dirBamFile = "BamMXL",
                  genes = c(161592986, 161601753),
                  geneNames = "3B")
```

\textbf{Count reads in windows}

Use the function \textit{countReadInWindow} to read the BAM files into R and count the number of reads in each of the windows.

```r
readCountMatrix <- countReadInWindow(Object = objectCNVrd2, correctGC = TRUE)
```

If GC-content correction is selected (\texttt{correctGC=TRUE}) then a reference genome must be supplied. The default reference genome is the human reference genome (UCSC version hg19). A full list of reference genomes available through Bioconductor can be obtained from:

\url{http://www.bioconductor.org/packages/release/bioc/html/BSgenome.html}
Segmentation

Use the function `segmentSamples` to segment and obtain segmentation scores for the `FCGR3B` gene (Figure 1):

```r
resultSegment <- segmentSamples(Object = objectCNVrd2, stdCntMatrix = readCountMatrix)
```

Instead of reading BAM files directly, we can use a matrix of read counts for the function `segmentSamples`. Here, we obtain a read-count matrix from data in the `CNVrd2` package.

```r
data(fcgr3bMXL)
readCountMatrix <- resultSegment$stdCntMatrix
readCountMatrix[1:2, 1:2]
objectCNVrd2 <- new("CNVrd2", windows = 1000, chr = "chr1",
                   st = 161100001, en = 162100000,
                   dirBamFile = "BamMXL",
                   genes = c(161592986, 161601753),
                   geneNames = "3B")
resultSegment <- segmentSamples(Object = objectCNVrd2, stdCntMatrix = readCountMatrix)
sS <- resultSegment$segmentationScores
hist(sS[, 1], 100, xlab = 'Segmentation score', main = '')
```

Obtain copy-number count

The data in Figure 1 suggest four distinct groups of segmentation scores, likely related to four different copy number genotypes. The function `groupCNVs` uses a normal mixture model to cluster SSs into groups. Unequal variances are assumed by default (`EV = FALSE`), however, if there are relatively few SS values in one group then we can set `EV = TRUE` (see the `groupCNVs` manual page for additional details).
Figure 1: FCGR3B segmentation score.
objectCluster <- new("clusteringCNVs",
    x = resultSegment$segmentationScores[, 1], k = 4, EV = TRUE)
#Cluster into 4 groups

copynumberGroups <- groupCNVs(Object = objectCluster)

## 7 iterations
## 4 components with Equal variances
## m: -1.30876666665667 -0.0258018450411359 0.9838838408971832 0.0847018121148
## p: 0.0517241379315355 0.689660686451199 0.206900745610526 0.051714430067392
## sigma: 0.146985856296588 0.146985856296588 0.146985856296588 0.146985856296588

Clustering results are shown in Figure 2, and the group assignments for the samples are contained in the allGroups object. For example, the NA19648 sample is assigned to the second group because the probability associated with membership of this group is higher than that of the other groups (nearly 1).

copynumberGroups$allGroups[1:3, ]

## Name Classification Group1
## NA19648.MXL.bam NA19648.MXL.bam 2 1.784492e-18
## NA19649.MXL.bam NA19649.MXL.bam 2 3.742821e-17
## NA19651.MXL.bam NA19651.MXL.bam 3 5.012828e-61

If we would like to force outliers into the lowest or highest CN genotype groups (e.g., dividing the data into three groups: deletions, normal CN, duplications) then we can use options rightLimit (Figure 3) or leftLimit or both.

#Set right limit = 1.5 to make values > 1.5 be into the largest group.
objectCluster <- new("clusteringCNVs",
    x = resultSegment$segmentationScores[, 1], k = 3, EV = TRUE)

copynumberGroups <- groupCNVs(Object = objectCluster, rightLimit = 1.5)
Figure 2: FCGR3B CN groups.
Figure 3: FCGR3B CN groups (rightLimit = 1.5).

## 5 iterations
## 3 components with Equal variances
## m: -1.30876666666667-0.02580733715714130.983841901058517
## p: 0.05454545454545460.7272728952233420.218181650231204
## sigma: 0.1254981622689890.1254981622689890.125498162268989
Plots

The function `plotCNVrd2` can plot multiple samples. Trace plots of some of the samples exhibiting duplications at the FCGR locus are shown in Figure 4. Here, based on information from the literature, we assume that a copy number of two is the most common CN genotype.

```r
allGroups <- copynumberGroups$allGroups
###Obtain names of duplicate samples
duplicatedSamples <- rownames(allGroups[allGroups[, 2] > 2,])
###Plot 6 duplicate samples
par(mfrow = c(3, 2))
for (ii in duplicatedSamples[1:6])
  plotCNVrd2(Object = objectCNVrd2,
             segmentObject = resultSegment,
             sampleName = ii)
```

Identifying tag SNPs/INDELS for FCGR3B CNVs

The function `calculateLDSNPandCNV` is used to calculate LD between CNVs and SNPs/INDELS. This function will read a VCF file into R and transform phased/unphased values (00, 01, 10, 11) into numeric values (0, 1, 2 or 0, 1). For a large VCF file (e.g., >= 1Mb), we generally use the option `nChunkForVcf=50` to break the file into 50 chunks for reading into R.

```r
##Obtain VCF-file information in CNVrd2 package
vcfFile <- system.file(package="CNVrd2", "extdata",
                       "chr1.161600000.161611000.vcf.gz")
##Make a data frame named sampleCNV including samples, CNs, population names
sampleCNV <- data.frame(copynumberGroups$allGroups[, c(1, 2)].rep("MXL", dim(copynumberGroups$allGroups)[1]),
                        rownames(sampleCNV) <- substr(sampleCNV[, 1], 1, 7)
sampleCNV[, 1] <- rownames(sampleCNV)
##The first column must be the sample names and some samples should be in the vcf file
tagSNPandINDELofMXL <- calculateLDSNPandCNV(sampleCNV = sampleCNV,
                                              vcfFile = vcfFile, cnvColumn = 2,
                                              population = "MXL", popColumn = 3,
                                              nChunkForVcf = 50)
```
Figure 4: MXL duplicated samples.
## Reading the VCF file 1:161600001-161611000 with 5 blocks each

## VCF file: 161600001 to 161602751
## VCF file: 161602752 to 161605500
## VCF file: 161605501 to 161608250
## VCF file: 161608251 to 161611000
## Calculating p and r² values for MXL population.

```r
head(tagSNPandINDELofMXL)
```

<table>
<thead>
<tr>
<th></th>
<th>1CN (n=3)</th>
<th>2CN (n=36)</th>
<th>3CN (n=15)</th>
<th>p.values</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs117435514</td>
<td>0.00</td>
<td>2.78</td>
<td>86.67</td>
<td>2.1e-09</td>
<td>0.81</td>
</tr>
<tr>
<td>rs185696163</td>
<td>0.00</td>
<td>2.78</td>
<td>86.67</td>
<td>2.1e-09</td>
<td>0.81</td>
</tr>
<tr>
<td>rs34015117</td>
<td>0.00</td>
<td>2.78</td>
<td>80.00</td>
<td>2.3e-08</td>
<td>0.77</td>
</tr>
<tr>
<td>rs76736176</td>
<td>0.00</td>
<td>2.78</td>
<td>66.67</td>
<td>1.5e-06</td>
<td>0.68</td>
</tr>
<tr>
<td>esv2661911</td>
<td>66.67</td>
<td>0.00</td>
<td>0.00</td>
<td>2.1e-03</td>
<td>-0.39</td>
</tr>
<tr>
<td>rs72704050</td>
<td>33.33</td>
<td>52.78</td>
<td>93.33</td>
<td>4.6e-03</td>
<td>0.34</td>
</tr>
</tbody>
</table>

```r
head(tagSNPandINDELofMXL)
```

<table>
<thead>
<tr>
<th></th>
<th>p.valuesAdjusted</th>
<th>r²</th>
<th>POP</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs117435514</td>
<td>1.3e-07</td>
<td>0.66</td>
<td>MXL</td>
</tr>
<tr>
<td>rs185696163</td>
<td>1.3e-07</td>
<td>0.66</td>
<td>MXL</td>
</tr>
<tr>
<td>rs34015117</td>
<td>9.6e-07</td>
<td>0.59</td>
<td>MXL</td>
</tr>
<tr>
<td>rs76736176</td>
<td>4.7e-05</td>
<td>0.46</td>
<td>MXL</td>
</tr>
<tr>
<td>esv2661911</td>
<td>5.4e-02</td>
<td>0.15</td>
<td>MXL</td>
</tr>
<tr>
<td>rs72704050</td>
<td>9.7e-02</td>
<td>0.11</td>
<td>MXL</td>
</tr>
</tbody>
</table>

From the results of the LD analysis, rs117435514 is the best tagSNP for duplications: 0%, 2.78% and 86.7% of deleted, normal and duplicated samples have this SNP (adjusted p-value = 7.1e-08, r² = 0.66).

### Working with complex loci

`CNVid2` can also be used to measure multiallelic copy-number polymorphisms. For loci having high CN, users should use the function `segmentSamplesUsingPopInformation` to adjust the seg-

...
mentation process across populations. An example of a gene exhibiting this type of complex CN polymorphism is CCL3L1. Below we use the package to measure CCL3L1 CN and identify tag SNPs/INDELs for CCL3L1 CNVs.

The data set used here includes 1,917 samples of five large populations European, East Asian, West African, South Asian ancestry and Americas with a total of 26 small populations as in the table below:

<table>
<thead>
<tr>
<th>Large Pop</th>
<th>Small Pop</th>
<th>Sample size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Americas</td>
<td>CLM</td>
<td>65</td>
</tr>
<tr>
<td>Americas</td>
<td>MXL</td>
<td>90</td>
</tr>
<tr>
<td>Americas</td>
<td>PEL</td>
<td>60</td>
</tr>
<tr>
<td>Americas</td>
<td>PUR</td>
<td>74</td>
</tr>
<tr>
<td>East Asian</td>
<td>CDX</td>
<td>88</td>
</tr>
<tr>
<td>East Asian</td>
<td>CHB</td>
<td>83</td>
</tr>
<tr>
<td>East Asian</td>
<td>CHS</td>
<td>104</td>
</tr>
<tr>
<td>East Asian</td>
<td>JPT</td>
<td>82</td>
</tr>
<tr>
<td>East Asian</td>
<td>KHV</td>
<td>78</td>
</tr>
<tr>
<td>European</td>
<td>CEU</td>
<td>96</td>
</tr>
<tr>
<td>European</td>
<td>FIN</td>
<td>78</td>
</tr>
<tr>
<td>European</td>
<td>GBR</td>
<td>77</td>
</tr>
<tr>
<td>European</td>
<td>IBS</td>
<td>77</td>
</tr>
<tr>
<td>European</td>
<td>TSI</td>
<td>100</td>
</tr>
<tr>
<td>South Asian</td>
<td>BEB</td>
<td>50</td>
</tr>
<tr>
<td>South Asian</td>
<td>GIH</td>
<td>81</td>
</tr>
<tr>
<td>South Asian</td>
<td>ITU</td>
<td>39</td>
</tr>
<tr>
<td>South Asian</td>
<td>PJL</td>
<td>37</td>
</tr>
<tr>
<td>South Asian</td>
<td>STU</td>
<td>49</td>
</tr>
<tr>
<td>African</td>
<td>ACB</td>
<td>74</td>
</tr>
<tr>
<td>African</td>
<td>ASW</td>
<td>50</td>
</tr>
<tr>
<td>African</td>
<td>ESN</td>
<td>64</td>
</tr>
<tr>
<td>African</td>
<td>GWD</td>
<td>105</td>
</tr>
<tr>
<td>African</td>
<td>LWK</td>
<td>90</td>
</tr>
<tr>
<td>African</td>
<td>MSL</td>
<td>68</td>
</tr>
<tr>
<td>African</td>
<td>YRI</td>
<td>89</td>
</tr>
</tbody>
</table>
Measuring CCL3L1 CN

The `ccl3l1data` data includes 1917 samples downloaded from the 1000 Genomes Project in October 2012 and March 2013, their corresponding populations, segmentation scores and CNs. The segmentation scores were obtained by using the function `segmentSamplesUsingPopInformation` for a 1Mb region (`chr17:33670000-34670000`) with 500bp-constant windows.

```r
# Load data into R:
data(ccl3l1data)
head(ccl3l1data)

## Name Pop SS CN
## 1 HG00096 GBR -0.6932975 1
## 2 HG00100 GBR -0.2308212 2
## 3 HG00103 GBR -0.3511010 2
## 4 HG00106 GBR -0.8012343 1
## 5 HG00108 GBR -0.4119802 2
## 6 HG00111 GBR -0.2879718 2

hist(ccl3l1data$SS, 100)
```

As can be seen in Figure 5, the data is multimodal and there are not clear clusters on the right. Therefore, we can use a single population which has clear clusters to obtain prior information for the clustering process into CN groups. Here, we used the large European-ancestry population to obtain prior information.

```r
xyEuro <- ccl3l1data[grep("CEU|TSI|IBS|GBR|FIN", ccl3l1data[, 2]), ]
yEuro <- xyEuro[, 3]
names(yEuro) <- xyEuro[, 1]
hist(yEuro, 100, xlab = '', main = '')
```

As can be seen from Figure 6, the European-ancestry data exhibit relatively clear clusters, allowing us to classify the samples into different CN groups.

**Note:** if we use the option `autoDetermineGroup = TRUE` in the function `groupByCN-
Figure 5: CCL3L1 segmentation score.
Figure 6: European-ancestry segmentation score.
NVs then the Bayesian information criterion (BIC) will be used to choose a suitable 
number of components (See Schwarz (1978)).

```r
# Clustering European segmentation

objectClusterEuroCCL3L1 <- new("clusteringCNVs", x = yEuro, k = 5)
europeanCCL3L1Groups <- groupCNVs(Object = objectClusterEuroCCL3L1)

# 7 iterations

## 5 components with Unequal variances

# m: 
-1.51402924373199 -0.871863641754189 -0.3596050190158570 0.1442265197789390 0.611056959363024

# p: 
0.0256947654760155 0.281378789950390 0.4465042675825080 0.1787783811870730 0.0676437955040135

# sigma: 
0.049855661235899970 0.1133002471605970 1.00886849220440 0.06154673291350 2.30716323036497
```

Next, we use these results to infer \textit{CCL3L1} CN in all populations. The following 
code collects information about the means, standard deviations and proportions of 
the mixture components from the European population.

```r
# Means

lambda0 <- as.numeric(europeanCCL3L1Groups$m)

# SD

sdEM <- as.numeric(europeanCCL3L1Groups$sigma)

# Proportions

pEM <- as.numeric(europeanCCL3L1Groups$p)
```

Take a look these results:

```
lambda0
## [1] -1.5140292  -0.8718636  -0.3596050   0.1442265   0.6110570

sdEM
## [1] 0.04985566 0.11330025 0.10088648 0.11546746 0.23071632
```
Figure 7: Clustering results of European-ancestry sample sets.
## Calculate the distances between groups

```r
for (ii in 2:5) {print(lambda0[ii] - lambda0[ii-1])}
```

## All segmentation scores

```r
ccl3l1X <- ccl3l1data$SS
names(ccl3l1X) <- as.character(ccl3l1data$Name)
range(ccl3l1X)
```

The information above is then used by the function `groupBayesianCNVs` to cluster the segmentation scores for the combined set of European-ancestry cohorts into different CN groups. There is a high value in the full SS set (Figure 5), which we eliminate in the following steps by setting `rightLimit = 4` so that it is automatically allocated into the highest CN group. Using the other values, combined with locus-specific information from the literature, we set the number of groups to be 10. All prior information was used in our work to obtain CN for CCL3L1 gene (the four group).

```r
## Set prior information:
#prior for the sd of the means of groups:
#5 was set for the third group = 2 CN
sd <- c(1, 1, 5, 1, 1)
ccl3l1X <- sort(ccl3l1X)
## Data
xData <- ccl3l1X
## Number of groups
```
nGroups <- 10
###prior for means of groups
lambda0 <- lambda0
###Prior for mixing proportions
alpha0 <- c(3, 29, 44, 18, 7, rep(2, nGroups - length(pEM) - 1))
##Prior for the distances between groups
distanceBetweenGroups = 0.485

sdEM = sdEM

The final (5th) group of the results of the European-ancestry sample sets has a large standard deviation as a result of the scattering of values on the right (Figure 6), therefore, we can set this value to equal the standard deviation of the fourth group to avoid an overly wide mixture component.

###Adjust standard deviation for the fifth group

Run the groupBayesianCNVs function to obtain CN groups.

```r
set.seed(123)
groupCCL3L1allPops <- groupBayesianCNVs(xData = xData, nGroups = nGroups,
                                        lambda0 = lambda0,
                                        sd0 = sdEM, alpha0 = alpha0,
                                        distanceBetweenGroups = distanceBetweenGroups,
                                        sdOftau = sd,
                                        rightLimit = 4)
```

These results would be similar to the results of the fourth column in the data of the package.

**Identifying tag-SNPs for CCL3L1 CNVs**

We can obtain obtain tag-SNPs/INDELs for multiple populations simultaneously. Below, we reuse the CCL3L1 data to obtain tag-SNPs for some populations.
Load VCF file into R and choose populations which we would like to find tagSNPs/IN-DELS.

```r
crownames(ccl3l1data) <- ccl3l1data[, 1]
```

## Obtain vcf-file information in CNVrd2

```r
vcfFileCCL3L1 <- system.file(package="CNVrd2", "extdata", "chr17.34800000.34830000.vcf.gz")
```

## Set populations we would like to identify tagSNPs

```r
allPops <- c("TSI", "CEU", "GBR", "FIN", "IBS")
```

## Identify tag SNPs/INDELs

```r
tagSNPandINDELofCCL3L1 <- calculateLDSNPandCNV(sampleCNV = ccl3l1data, vcfFile = vcfFileCCL3L1, cnvColumn = 4, population = allPops, popColumn = 2, nChunkForVcf = 5, chr = "17", st = 34800000, en = 34830000 )
```

## Reading the VCF file 17:34800000-34830000 with 5 blocks each

## VCF file: 34800000 to 34807500
## VCF file: 34807501 to 34815000
## VCF file: 34815001 to 34822500
## VCF file: 34822501 to 34830000

## Calculating p and r2 values for TSI population.
## Calculating p and r2 values for CEU population.
## Calculating p and r2 values for GBR population.
## Calculating p and r2 values for FIN population.
## Calculating p and r2 values for IBS population.

Take a quick look some significant results (multiple populations: the return value of `calculateLDSNPandCNV` is a list of populations).

```r
lapply(tagSNPandINDELofCCL3L1, head)
```
<table>
<thead>
<tr>
<th>p.values r</th>
<th>p.valuesAdjusted r 2 PEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs80644266</td>
<td>3.2e-12 -0.64 5.0e-20 0.40 TSI</td>
</tr>
<tr>
<td>rs132877402</td>
<td>6.7e-22 -0.63 6.0e-20 0.39 TSI</td>
</tr>
<tr>
<td>rs13287723</td>
<td>3.4e-11 -0.63 1.0e-19 0.39 TSI</td>
</tr>
<tr>
<td>rs8072769</td>
<td>5.3e-10 -0.61 1.0e-18 0.37 TSI</td>
</tr>
<tr>
<td>rs132877191</td>
<td>2.4e-10 -0.57 1.5e-18 0.34 TSI</td>
</tr>
<tr>
<td>rs132877182</td>
<td>1.4e-09 -0.57 3.0e-18 0.30 TSI</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0CN(n=2) 1CN(n=20) 2CN(n=39) 3CN(n=14) 4CN(n=3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs8072769</td>
<td>0 75 2.56 0.00 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs113435750</td>
<td>0 75 5.13 0.00 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs138153523</td>
<td>0 75 5.13 0.00 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs11316723</td>
<td>0 80 5.13 7.14 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs80644266</td>
<td>0 80 5.13 7.14 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs9911791</td>
<td>0 85 10.16 7.14 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5CN(n=1) p.values r 2 PEP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs8072769</td>
<td>0 7.6e-10 -0.61 1.4e-07 0.37 CEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs113435750</td>
<td>0 6.4e-09 -0.60 2.4e-07 0.35 CEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs138153523</td>
<td>0 6.4e-09 -0.60 2.4e-07 0.35 CEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs11316723</td>
<td>0 3.7e-09 -0.57 2.3e-07 0.33 CEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs80644266</td>
<td>0 3.7e-09 -0.57 2.3e-07 0.33 CEU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs9911791</td>
<td>0 8.8e-09 -0.58 2.7e-07 0.33 CEU</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1CN(n=18) 2CN(n=37) 3CN(n=11) 4CN(n=4) p.values r 2 PEP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs11316723</td>
<td>72.22 2.70 0 0 1.2e-08 -0.65</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs80644266</td>
<td>72.22 3.70 0 0 1.3e-08 -0.65</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs9911791</td>
<td>72.22 8.11 0 0 5.6e-07 -0.62</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs113877493</td>
<td>50.00 0.00 0 0 3.8e-06 -0.54</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs8072769</td>
<td>50.00 2.70 0 0 5.1e-05 -0.52</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs132877191</td>
<td>50.00 2.70 0 0 5.1e-05 -0.52</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5CN(n=1) p.values r 2 PEP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs11316723</td>
<td>1.9e-06 0.43 GBR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs80644266</td>
<td>1.9e-06 0.43 GBR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs9911791</td>
<td>3.4e-05 0.38 GBR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs138153523</td>
<td>1.8e-04 0.32 GBR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs8072769</td>
<td>1.6e-03 0.37 GBR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs132877191</td>
<td>1.6e-03 0.37 GBR</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0CN(n=2) 1CN(n=19) 2CN(n=28) 3CN(n=18) 4CN(n=6) p.values r 2 PEP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs113877493</td>
<td>100 73.68 7.14 0.00 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs6607368</td>
<td>100 84.21 21.43 5.56 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs8067765</td>
<td>100 63.16 7.14 0.00 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs60952743</td>
<td>100 63.16 7.14 0.00 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs8070238</td>
<td>100 63.16 7.14 0.00 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs8072238</td>
<td>100 63.16 7.14 0.00 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5CN(n=1) p.values r 2 PEP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs113877493</td>
<td>0 3.7e-09 -0.67 4.9e-07 0.45 FIN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs6607368</td>
<td>0 4.7e-08 -0.66 2.8e-06 0.43 FIN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs8067765</td>
<td>0 2.3e-07 -0.61 4.2e-06 0.38 FIN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs60952743</td>
<td>0 2.3e-07 -0.61 4.2e-06 0.38 FIN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs8070238</td>
<td>0 2.3e-07 -0.61 4.2e-06 0.38 FIN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs8072238</td>
<td>0 2.3e-07 -0.61 4.2e-06 0.38 FIN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1CN(n=2) 2CN(n=1) 3CN(n=1) 4CN(n=2) p.values r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs4796217</td>
<td>100 100 100 0 2.0e-01 -0.85</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs28856610</td>
<td>0 0 0 100 2.0e-01 0.85</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs116523258</td>
<td>0 0 100 2.0e-01 0.70</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs133254191</td>
<td>0 0 0 50 1.0e+00 0.54</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs4796216</td>
<td>50 100 100 100 1.0e+00 0.54</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs806320102</td>
<td>0 0 0 50 1.0e+00 0.54</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The output above provides evidence that rs113877493 may be a tagSNP for CCL3L1 deletions in the FIN (p = 1.1e-07, \( r^2 = 0.44 \)) and GBR (p = 2.6e-04, \( r^2 = 0.27 \)) populations.

```
## Notice: these results are in a list
for (ii in 1:length(allPops))
    write.table(tagSNPandINDELofCCL3L1[[ii]],
        paste("TagSNPforPop", ii, ".csv", sep = ""),
        quote = FALSE, sep = "\t")
```

Note: All TagSNP results for DEFB103A, CCL3L1 and FCGR3B CNV will be uploaded on https://github.com/hoangtn/CNVrd2

### Identifying polymorphic regions

CNVrd2 can also be used to identify CN polymorphic regions and the putative boundaries of these regions. We reuse the data from the FCGR3 locus to investigate the polymorphic region around the two genes.

```
fgr3PolymorphicRegion <- identifyPolymorphicRegion(Object = objectCNVrd2,
    segmentObject = resultSegment,
    plotLegend = FALSE)
```

```
## Calculate segmentation scores for polymorphic regions
```
To plot a small region around the gene, we use the function `plotPolymorphicRegion`. Users can change `typePlot` to only plot SDs or percentiles.

```r
glelim = c(161300000, 161800000). drawThresholds = TRUE. typePlot = "SD")
```

---

```
## IRanges object with 5 ranges and 0 metadata columns:
##
## start  end  width
##<integer> <integer> <integer>
##[1] 161392001 161394000 2000
##[2] 161424001 161426000 2000
```
Figure 8: CN polymorphic region at FCGR3 locus, represented by quantiles of the distribution of segmentation scores across samples.
Here, we are using standard deviations to identify polymorphic regions. Therefore, the putative boundaries of these regions rely on the parameter `sdThreshold`. Alternatively, users can use different percentiles to identify these regions. In that case, two parameters `quantileValue` and `thresholdForPolymorphicRegions` can be used to adjust the boundaries of regions. For example, here we can set small `sdThreshold` values to obtain only high-polymorphic regions (e.g., `CCL3L1`), but it can omit some medium-polymorphic regions (e.g., `FCGR3A/3B`). Figure 8 and 9 depicts two different thresholds resulting in different polymorphic regions.

In the function `identifyPolymorphicRegion`, if we would like to obtain only polymorphic regions which differentiate between populations (e.g., to detect evidence of selection) then we can use the option `VstTest=TRUE`. This option will calculate the Vst statistics (Redon et al., 2006). Users have to supply a vector which includes population information in `popName`. The returned putative boundaries will be the intersection of polymorphic regions and regions having `maxVst >= thresholdVST`.

**Note**

If we use the option `entireGene = TRUE` in the step `segmentation` then the pipeline will not refine the segmentation results (the results will be the same as the pipeline used in Nguyen et al. (2013)).

**Session information**
Figure 9: CN polymorphic region at FCGR3 locus, represented by quantiles of the distribution of segmentation scores across samples.
sessionInfo()

## R version 3.3.1 (2016-06-21)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.1 LTS
##
## locale:
## [1] LC_CTYPE=en_US.UTF-8  LC_NUMERIC=C
## [3] LC_TIME=en_US.UTF-8  LC_COLLATE=C
## [5] LC_MONETARY=en_US.UTF-8 LC_MESSAGES=en_US.UTF-8
## [7] LC_PAPER=en_US.UTF-8  LC_NAME=C
## [9] LC_ADDRESS=C  LC_TELEPHONE=C
##[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4 parallel stats graphics grDevices utils
## [7] datasets methods base
##
## other attached packages:
## [1] CNVrd2_1.12.0 gridExtra_2.2.1
## [3] ggplot2_2.1.0 rjags_4-6
## [5] coda_0.18-1 VariantAnnotation_1.20.0
## [7] Rsamtools_1.26.0 Biostrings_2.42.0
## [9] XVector_0.14.0 SummarizedExperiment_1.4.0
##[11] Biobase_2.34.0 GenomicRanges_1.26.0
##[13] GenomeInfoDb_1.10.0 IRanges_2.8.0
##[15] S4Vectors_0.12.0 BiocGenerics_0.20.0
##[17] knitr_1.14
##
## loaded via a namespace (and not attached):
## [1] Rcpp_0.12.7 plyr_1.8.4
## [3] formatR_1.4 highr_0.6
## [5] GenomicFeatures_1.26.0 bitops_1.0-6
## [7] tools_3.3.1 zlibbioc_1.20.0
## [9] digest_0.6.10 biomaRt_2.30.0
##[11] evaluate_0.10 RSQLite_1.0.0
## Packages

- gtable_0.2.0
- lattice_0.20-34
- BSgenome_1.42.0
- Matrix_1.2-7.1
- DBI_0.5-1
- rtracklayer_1.34.0
- stringr_1.1.0
- grid_3.3.1
- DNAcopy_1.48.0
- AnnotationDbi_1.36.0
- XML_3.98-1.4
- BiocParallel_1.8.0
- magrittr_1.5
- scales_0.4.0
- GenomicAlignments_1.10.0
- colorspace_1.2-7
- labeling_0.3
- stringi_1.1.2
- RCurl_1.95-4.8
- munsell_0.4.3

## References


