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

Hoang Tan Nguyen[1,2], Tony R Merriman[1] and Michael A Black[1]

[1]Department of Biochemistry, University of Otago
[2]Department of Mathematics and Statistics, University of Otago

September 15, 2013

Contents

1 Introduction

The *CNVrd2* package [1] (Nguyen *et~al.*, 2013) 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 disquilibrium (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.

[1] CNVrd2 is an improved version of the pipeline *CNVrd* used to identify tagSNPs of *FCGR3A/B* CNV

2 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")

## Loading required package: DNAcopy
```

Working with BAM and VCF files.

The following section describes the workflow of the *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 *FCGR3B* gene (chr1:161592986-161601753). Users can download the aligned data in the file *MXLexample.zip* from

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 verison of the same data.

2.1 Measuring FCGR3B copy number

2.1.1 CNVrd2 object

We need to make an object of class *CNVrd2* to define a region we want to investigate (regions sized > 1Mb 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 **not downloaded the BAM files**, should skip to section 2.1.3

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

2.1.2 Count reads in windows

Use the function *countReadInWindow* to read the BAM files into R and count the number of reads in each of the windows.

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

If GC-content correcion is selected (*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:

http://www.bioconductor.org/packages/release/bioc/html/BSgenome.html

2.1.3 Segmentation

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

```
## Obtain segmentation scores
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.

```
## Load data into R
data(fcgr3bMXL)
## Reload readCountMatrix
readCountMatrix <- resultSegment$stdCntMatrix
## Take a quick look the data
readCountMatrix[1:2, 1:2]
## Make a CNVrd2 object
objectCNVrd2 <- new("CNVrd2", windows = 1000, chr = "chr1", st = 161100001,
    en = 162100000, dirBamFile = "BamMXL", genes = c(161592986, 161601753),
    geneNames = "3B")
## Obtain segmentation scores
resultSegment <- segmentSamples(Object = objectCNVrd2, stdCntMatrix = readCountMatrix)
## View these segmentation results
sS <- resultSegment$segmentationScores
hist(sS[, 1], 100, xlab = "Segmentation score", main = "")
```

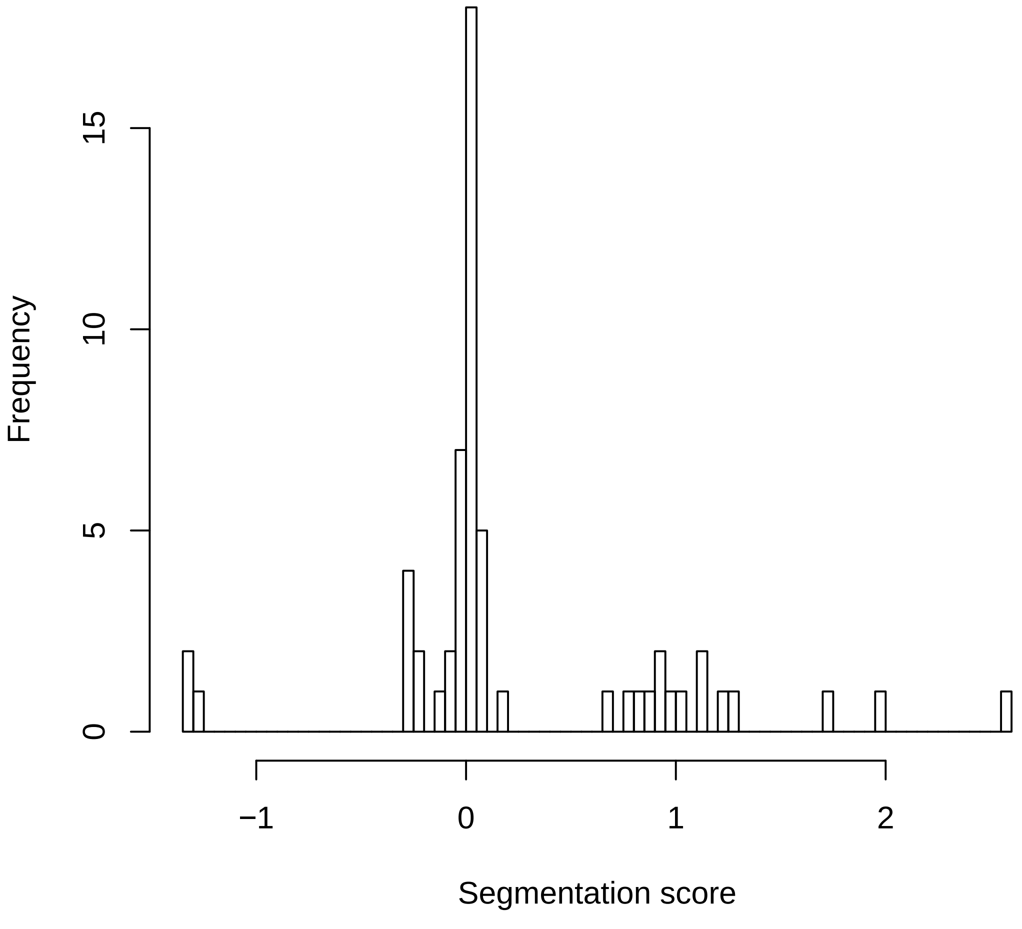


Figure 1: FCGR3B segmentation score.

2.1.4 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).

```
objectCluster <- new("clusteringCNVs", x = resultSegment$segmentationScores[,
    1], k = 4, EV = TRUE)
# Cluster into 4 groups
copynumberGroups <- groupCNVs(Object = objectCluster)
```

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    Group2
## NA19648.MXL.bam NA19648.MXL.bam              2 1.784e-18 1.000e+00
## NA19649.MXL.bam NA19649.MXL.bam              2 3.743e-17 1.000e+00
## NA19651.MXL.bam NA19651.MXL.bam              3 5.013e-61 1.541e-13
##                    Group3    Group4   score
## NA19648.MXL.bam 2.472e-11 3.368e-46 -0.0195
## NA19649.MXL.bam 2.250e-12 2.248e-48 -0.0710
## NA19651.MXL.bam 1.000e+00 4.647e-10  1.1393
```

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)
```

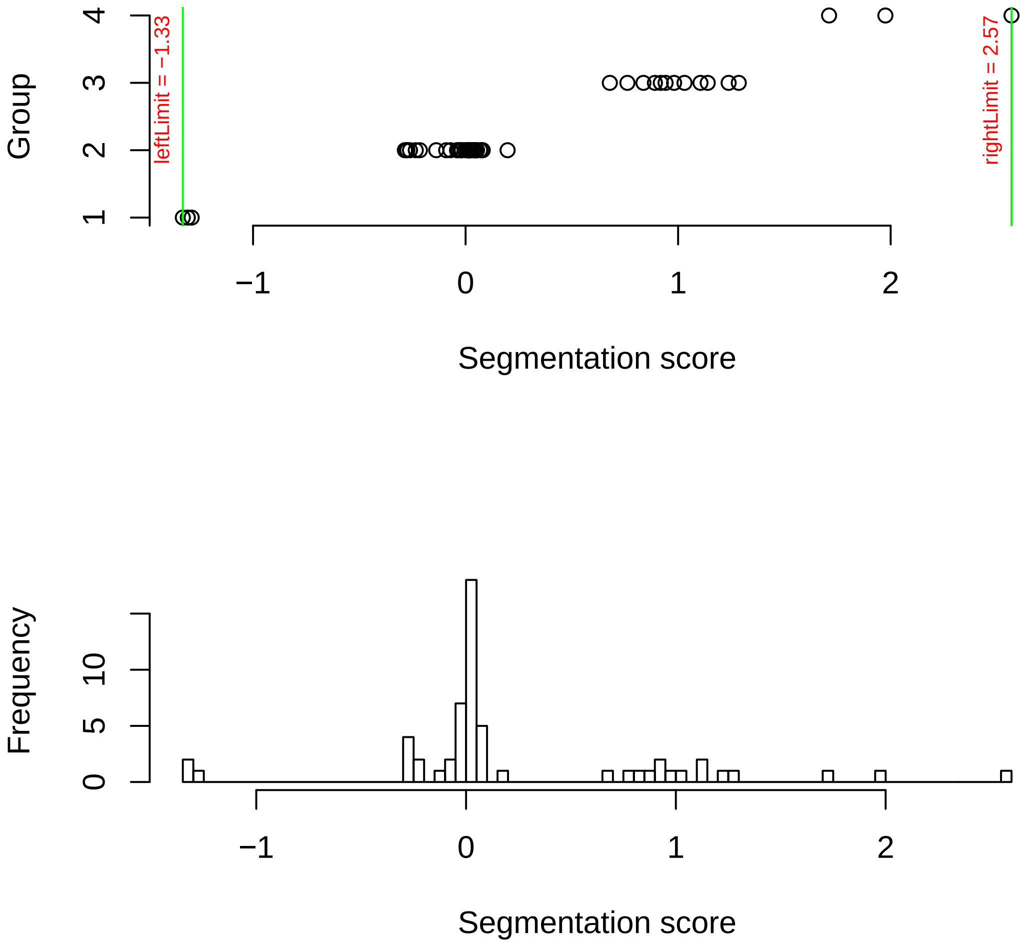


Figure 2: FCGR3B CN groups.

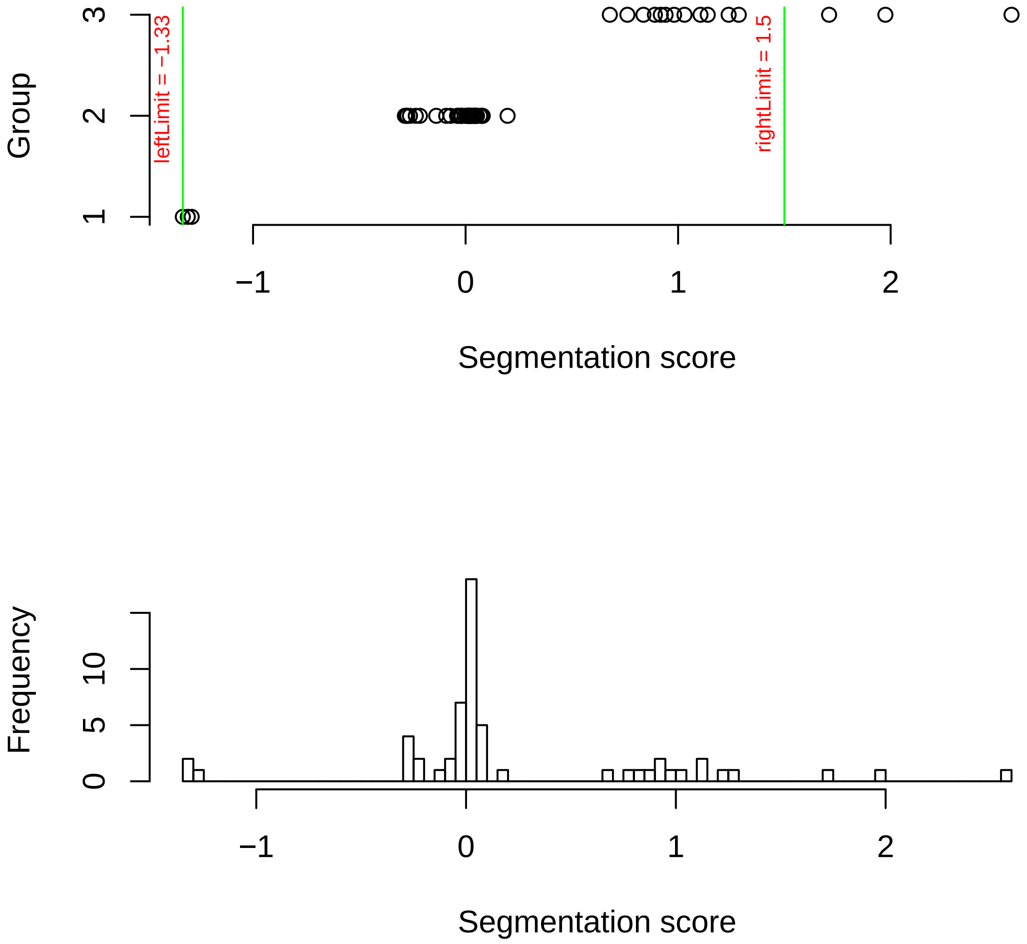


Figure 3: FCGR3B CN groups (rightLimit = 1.5).

2.1.5 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.

```
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)
```

2.2 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.

```
## 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,
```

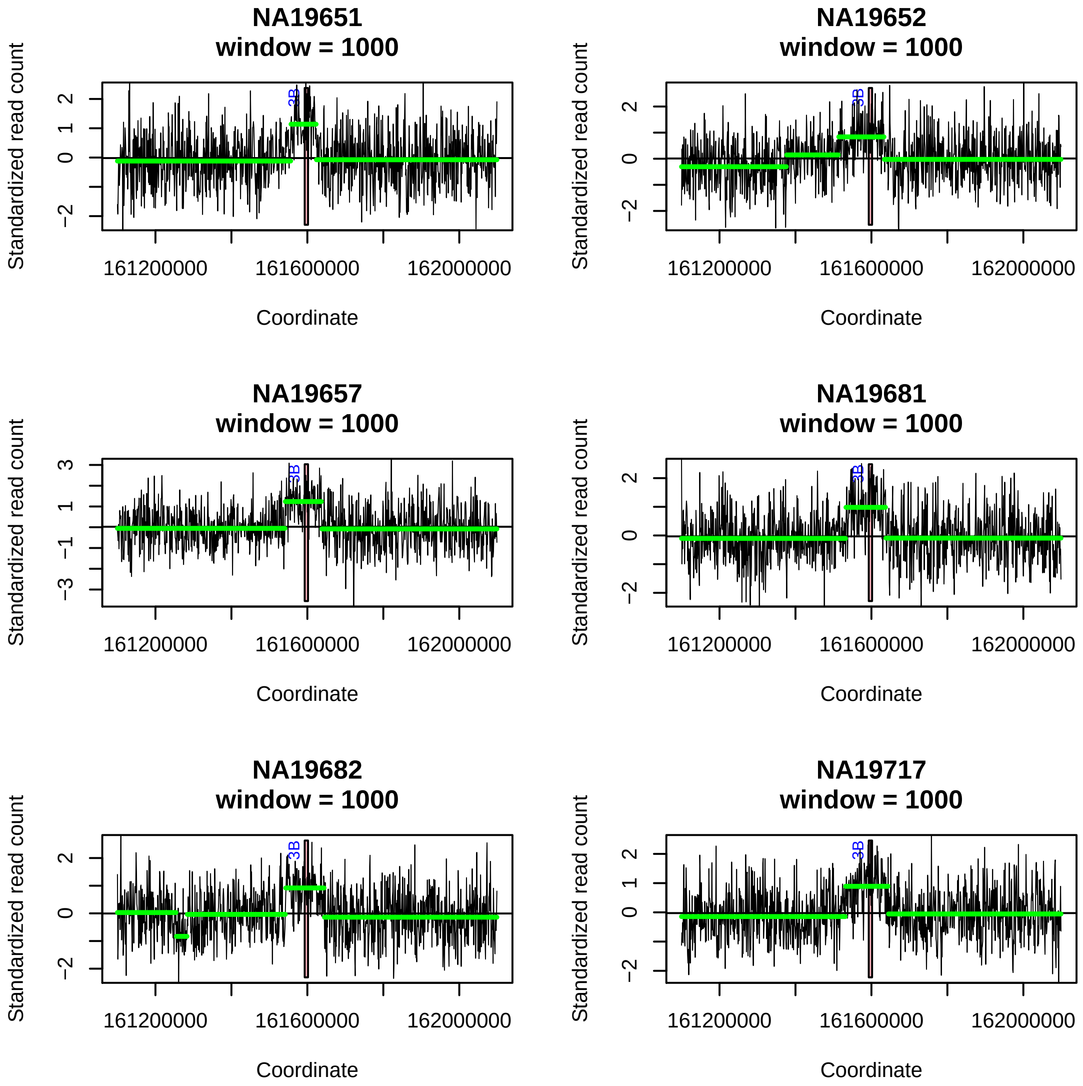


Figure 4: MXL duplicated samples.

```
    cnvColumn = 2, population = "MXL", popColumn = 3, nChunkForVcf = 5, chr = "1",
    st = 161600001, en = 161611000, codeSNP = "Three", codeCNV = "ThreeGroup")
```

```
head(tagSNPandINDELofMXL)

##             1CN_(n=3) 2CN_(n=36) 3CN_(n=15) p.values     r
## rs117435514      0.00       2.78      86.67  2.1e-09  0.81
## rs185696163      0.00       2.78      86.67  2.1e-09  0.81
## rs34015117       0.00       2.78      80.00  2.3e-08  0.77
## rs76736176       0.00       2.78      66.67  1.5e-06  0.68
## esv2661911      66.67       0.00       0.00  2.1e-03 -0.39
## rs72704050       0.00      50.00      86.67  4.6e-03  0.34
##             p.valuesAdjusted   r2 POP
## rs117435514          7.1e-08 0.66 MXL
## rs185696163          7.1e-08 0.66 MXL
## rs34015117           5.0e-07 0.59 MXL
## rs76736176           2.4e-05 0.46 MXL
## esv2661911           2.8e-02 0.15 MXL
## rs72704050           5.1e-02 0.11 MXL
```

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, $r^2 = 0.66$).

3 Working with complex loci

CNVrd2 can also be used to measure multiallelic copy number polymorphisms. For loci having high CN, users should use the function *segmentSamplesUsingPopInformation* to adjust the segmentation process across populations. An example of a gene exhibiting this type of complex CN polymorphism is *CCL3L1*. Below we measure *CCL3L1* CN and identify tag SNPs/INDELs for *CCL3L1* CNVs.

The data set used here includes 1,917 samples from five large populations: European, East Asian, West African, South Asian, and Americas, with a total of 26 small populations as shown in Table 1.

Population Group	Cohort	Sample size
Americas	ACB	74
Americas	ASW	50
Americas	CLM	65
Americas	MXL	59
Americas	PEL	60
Americas	PUR	74
East Asian	CDX	88
East Asian	CHB	83
East Asian	CHS	104
East Asian	JPT	82
East Asian	KHV	78
European	CEU	96
European	FIN	78
European	GBR	77
European	IBS	77
European	TSI	100
South Asian	BEB	50
South Asian	GIH	81
South Asian	ITU	39
South Asian	PJL	37
South Asian	STU	49
West African	ESN	64
West African	GWD	105
West African	LWK	90
West African	MSL	68
West African	YRI	89

Table 1: 1000 Genomes Project populations used in CCL3L1 copy number analysis.

3.1 Measuring CCL3L1 CN

The *ccl3l1data* data includes 1,917 samples downloaded from the 1000 Genomes Project in October 2012 and March 2013, their corresponding populations, segmentation scores and CNs [2]. The segmentation scores were obtained by using the function *segmentSamplesUsingPopInformation* for a 1Mb region (chr17:33670000-34670000) with 500bp-constant windows.

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

##      Name Pop      SS CN
## 1 HG00096 GBR -0.6933  1
## 2 HG00100 GBR -0.2308  2
## 3 HG00103 GBR -0.3511  2
## 4 HG00106 GBR -0.8012  1
## 5 HG00108 GBR -0.4120  2
## 6 HG00111 GBR -0.2880  2

hist(ccl3l1data$SS, 100, main = "", xlab = "Segmentation Score")
```

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 (cohorts CEU, TSI, IBS, GBR, FIN) to obtain prior information about the CN distribution.

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

As can be seen from Figure 6, the European-ancestry data exhibit relatively clear

[2] Manuscript in preparation

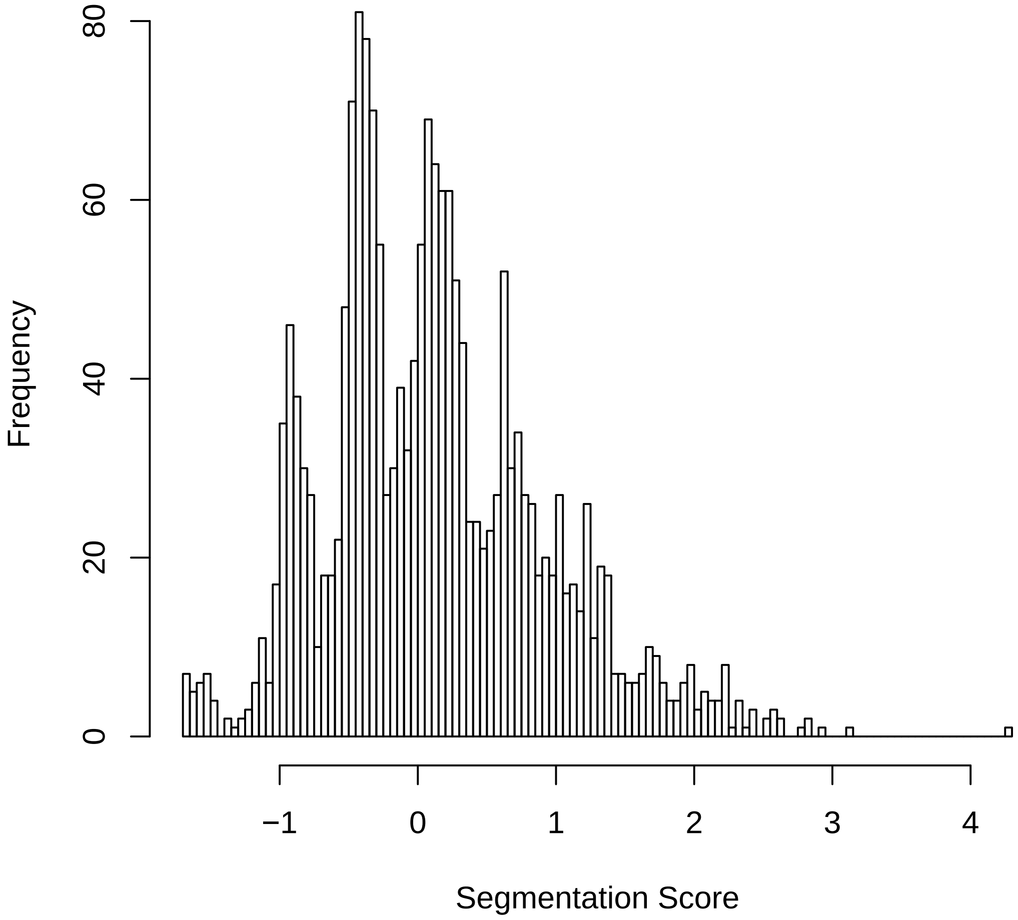


Figure 5: CCL3L1 segmentation score (all population groups).

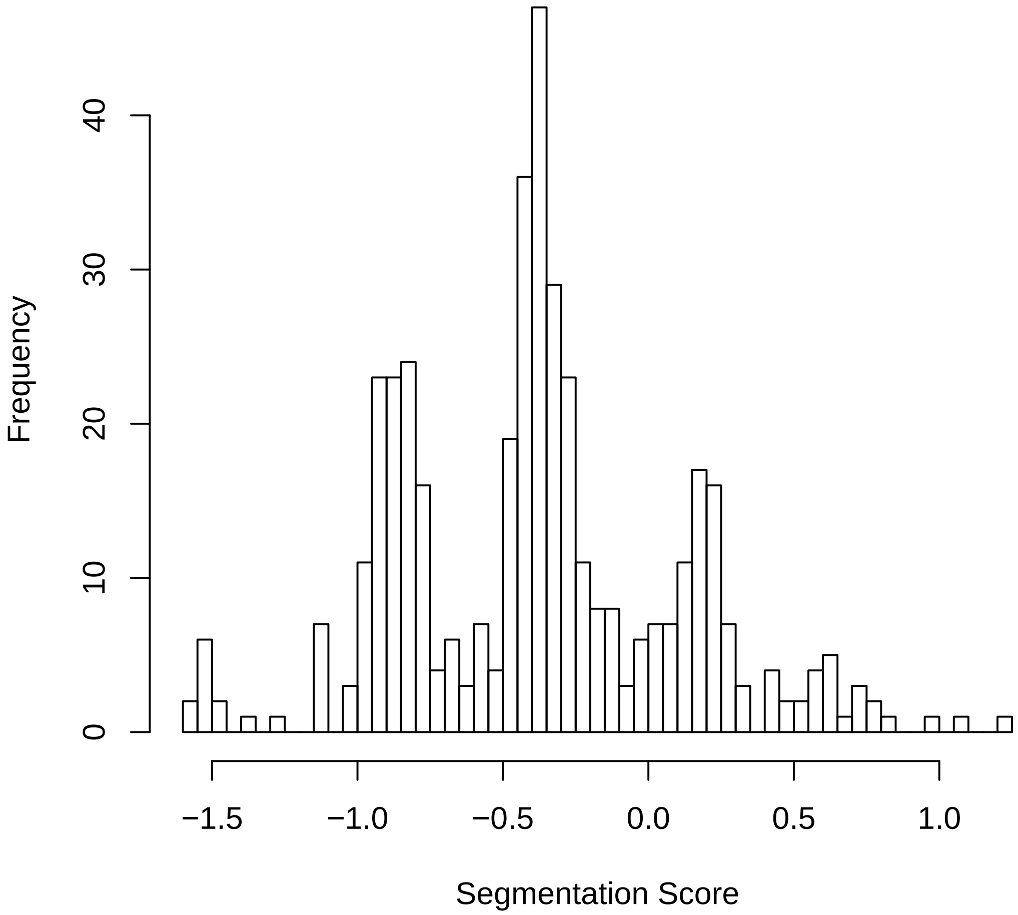


Figure 6: European-ancestry segmentation score.

clusters, allowing us to classify the samples into different CN groups.

Note: if we use the option *autoDetermineGroup* = *TRUE* in the function *groupCNVs* then the Bayesian information criterion (BIC) will be used to choose a suitable number of components (See Schwarz (1978)).

```
## Clustering European segmentation scores into group: 5 groups were
## chosen

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

europeanCCL3L1Groups <- groupCNVs(Object = objectClusterEuroCCL3L1)
```

Next, we use these results to infer *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.

```
# 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] -1.5140 -0.8719 -0.3596  0.1442  0.6111

sdEM

## [1] 0.04986 0.11320 0.10089 0.11547 0.23072

pEM

## [1] 0.02569 0.28138 0.44650 0.17878 0.06764

### Calculate the distances between groups
```

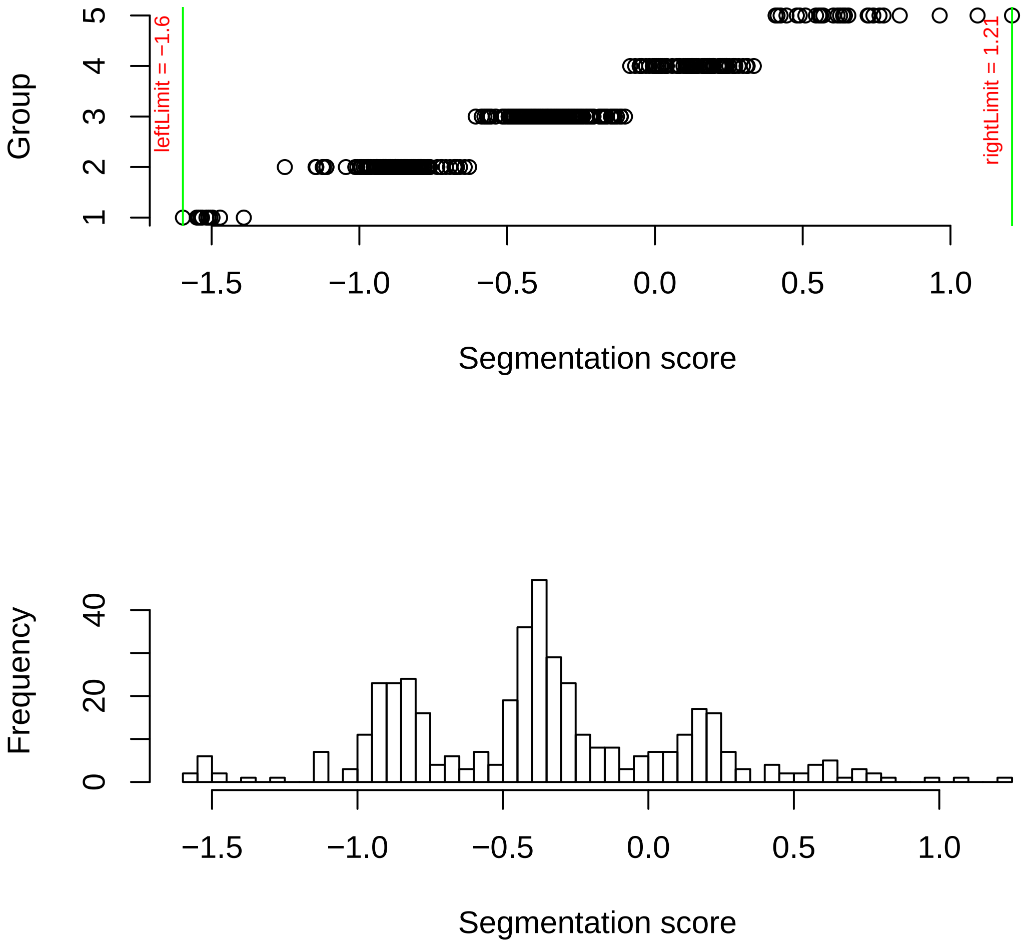


Figure 7: Clustering results of European-ancestry sample sets.

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

## [1] 0.6422
## [1] 0.5123
## [1] 0.5038
## [1] 0.4668


### All segmentation scores
ccl3l1X <- ccl3l1data$SS
names(ccl3l1X) <- as.character(ccl3l1data$Name)
range(ccl3l1X)

## [1] -1.675  4.258
```

The information above is then used by the function *groupBayesianCNVs* to cluster the segmentation scores for the combined set of all 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.

```
## 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, 5, rep(2, nGroups - length(pEM) - 1))
```

```
## Prior for the distances between groups
distanceBetweenGroups <- 0.485


sdEM <- sdEM
```

The final group 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
sdEM[5] <- sdEM[4]
```

Run the *groupBayesianCNVs* function to obtain CN groups.

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

With the random seed set to 123 (see above), the results obtained should be identical to those in the fourth column (CN) of the *ccl3l1data* data object.

3.2 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.

```
rownames(ccl3l1data) <- ccl3l1data[, 1]
```

Load VCF file into R and choose populations which we would like to find tagSNPs/INDELs.

```
## Obtain vcf-file information in CNVrd2
vcfFileCCL3L1 <- system.file(package = "CNVrd2", "extdata", "chr17.34800000.34830000.vcf.gz")
## Set populations we would like to identify tagSNPs
allPops <- c("TSI", "CEU", "GBR", "FIN", "IBS")
```

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

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

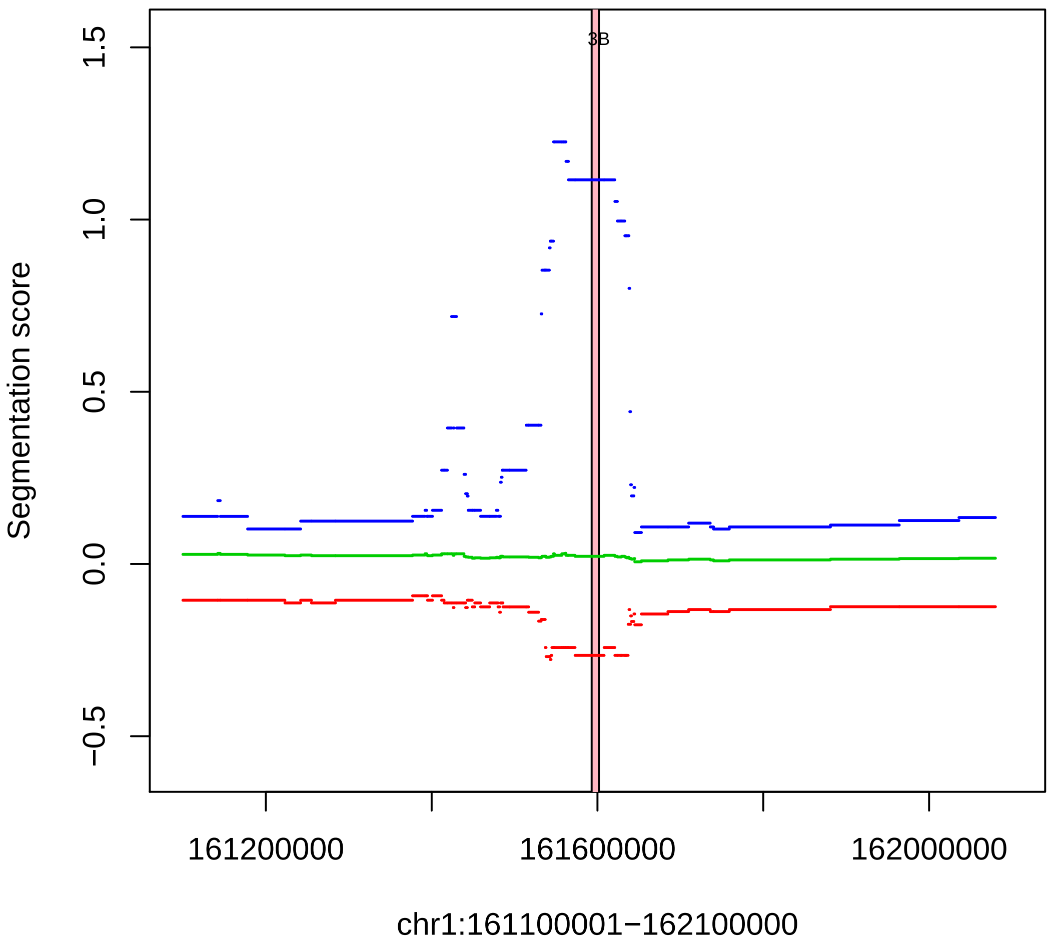
```
lapply(tagSNPandINDELofCCL3L1, head)

## $TSI
##             0CN_(n=5) 1CN_(n=32) 2CN_(n=41) 3CN_(n=16) 4CN_(n=4)
## rs8064426          80      87.50      12.20      18.75         0
## rs113877493        60      71.88       2.44       6.25         0
## rs11316723         80      84.38      12.20      18.75         0
## rs8072769          80      78.12      12.20      12.50         0
## rs9911791          80      87.50      17.07      18.75        25
## rs113435750        80      78.12      12.20      12.50        25
##             p.values     r p.valuesAdjusted   r2 POP
## rs8064426    3.2e-12 -0.64          2.1e-10 0.40 TSI
## rs113877493  6.7e-12 -0.63          2.3e-10 0.39 TSI
## rs11316723   3.1e-11 -0.62          7.1e-10 0.38 TSI
## rs8072769    5.3e-10 -0.61          7.2e-09 0.37 TSI
## rs9911791    2.5e-10 -0.58          4.2e-09 0.34 TSI
## rs113435750  1.4e-09 -0.57          1.4e-08 0.33 TSI
##
## $CEU
##             0CN_(n=2) 1CN_(n=20) 2CN_(n=39) 3CN_(n=14) 4CN_(n=3)
## rs8072769           0         75       2.56       0.00         0
## rs113435750         0         75       5.13       0.00         0
## rs138153523         0         75       5.13       0.00         0
## rs11316723          0         80       5.13       7.14         0
## rs8064426           0         80       5.13       7.14         0
## rs9911791           0         85      10.26       7.14         0
##             5CN_(n=1) p.values     r p.valuesAdjusted   r2 POP
## rs8072769           0  7.6e-10 -0.61          5.0e-08 0.37 CEU
## rs113435750         0  6.4e-09 -0.60          8.5e-08 0.35 CEU
## rs138153523         0  6.4e-09 -0.60          8.5e-08 0.35 CEU
## rs11316723          0  3.7e-09 -0.57          8.2e-08 0.33 CEU
## rs8064426           0  3.7e-09 -0.57          8.2e-08 0.33 CEU
## rs9911791           0  8.8e-09 -0.58          9.6e-08 0.33 CEU
##
## $GBR
##             1CN_(n=18) 2CN_(n=37) 3CN_(n=11) 4CN_(n=4) p.values     r
## rs11316723       72.22       2.70          0         0  1.2e-08 -0.65
## rs8064426        72.22       2.70          0         0  1.2e-08 -0.65
## rs9911791        72.22       8.11          0         0  5.6e-07 -0.62
## rs113877493      50.00       0.00          0         0  3.8e-06 -0.54
## rs8072769        50.00       2.70          0         0  5.1e-05 -0.52
## rs113435750      50.00       2.70          0         0  5.1e-05 -0.52
##             p.valuesAdjusted   r2 POP
## rs11316723           3.6e-07 0.43 GBR
## rs8064426            3.6e-07 0.43 GBR
## rs9911791            1.1e-05 0.38 GBR
## rs113877493          5.8e-05 0.29 GBR
## rs8072769            5.1e-04 0.27 GBR
## rs113435750          5.1e-04 0.27 GBR
##
## $FIN
##             0CN_(n=2) 1CN_(n=19) 2CN_(n=28) 3CN_(n=18) 4CN_(n=6)
## rs113877493       100      73.68       7.14       0.00         0
## rs6607368          100      84.21      21.43       5.56         0
## rs8067765          100      63.16       7.14       0.00         0
## rs60952743         100      63.16       7.14       0.00         0
## rs8070238          100      63.16       7.14       0.00         0
## rs8072238          100      63.16       7.14       0.00         0
##             5CN_(n=1) p.values     r p.valuesAdjusted   r2 POP
## rs113877493         0  2.7e-09 -0.67          1.5e-07 0.45 FIN
## rs6607368           0  4.7e-08 -0.66          8.8e-07 0.43 FIN
## rs8067765           0  2.3e-07 -0.61          1.3e-06 0.38 FIN
## rs60952743          0  2.3e-07 -0.61          1.3e-06 0.38 FIN
## rs8070238           0  2.3e-07 -0.61          1.3e-06 0.38 FIN
## rs8072238           0  2.3e-07 -0.61          1.3e-06 0.38 FIN
##
## $IBS
##             1CN_(n=2) 2CN_(n=1) 3CN_(n=1) 4CN_(n=2) p.values     r
## rs4796217         100       100       100         0  2.0e-01 -0.85
## rs28856610          0         0         0       100  2.0e-01  0.85
## rs11651338        100         0       100         0  2.0e-01 -0.70
## rs138347191         0         0         0        50  1.0e+00  0.54
## rs4796216          50       100       100       100  1.0e+00  0.54
## rs60520102          0         0         0        50  1.0e+00  0.54
##             p.valuesAdjusted   r2 POP
## rs4796217            9.7e-01 0.73 IBS
## rs28856610           9.7e-01 0.73 IBS
## rs11651338           9.7e-01 0.49 IBS
## rs138347191          1.0e+00 0.29 IBS
## rs4796216            1.0e+00 0.29 IBS
## rs60520102           1.0e+00 0.29 IBS
```

Th eoutput 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.

4 Indentifying poplymorphic regions

CNVrd2 can also be used to identity 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.

```
fcgr3PolymorphicRegion <- identifyPolymorphicRegion(Object = objectCNVrd2,
    segmentObject = resultSegment, thresholdForPolymorphicRegions = c(0.75,
        0.25), plotLegend = FALSE)
```



```
## Calculate segmentation scores for polymorphic regions
```

To plot a small region around the gene, we use the funtion *plotPolymorphicRegion*.

```
par(mfrow = c(2, 1))
plotPolymorphicRegion(Object = objectCNVrd2, polymorphicRegionObject = fcgr3PolymorphicRegion,
    xlim = c(161300000, 161800000), drawThresholds = TRUE, thresholdForPolymorphicRegions = c(0.75,
        0.25))

## IRanges of length 1
##         start       end  width
## [1] 161537001 161637000 100000
```

```
plotPolymorphicRegion(Object = objectCNVrd2, polymorphicRegionObject = fcgr3PolymorphicRegion,
    xlim = c(161300000, 161800000), drawThresholds = TRUE, thresholdForPolymorphicRegions = c(0.9,
        0.1))
```

```
## IRanges of length 3
##         start       end width
## [1] 161538001 161545000  7000
## [2] 161573001 161608000 35000
## [3] 161621001 161637000 16000
```

The boundaries of polymorphic regions rely on the two parameters *quantileValue* and *thresholdForPolymorphicRegions*. We can set high *thresholdForPolymorphicRegions* values to obtain only high-polymorphic regions (e.g., *CCL3L1*), but it can omit some medium-polymorphic regions (e.g., *FCGR3A/3B*). Figure 8 depicts two different thresholds resulting in different polymorphic regions.

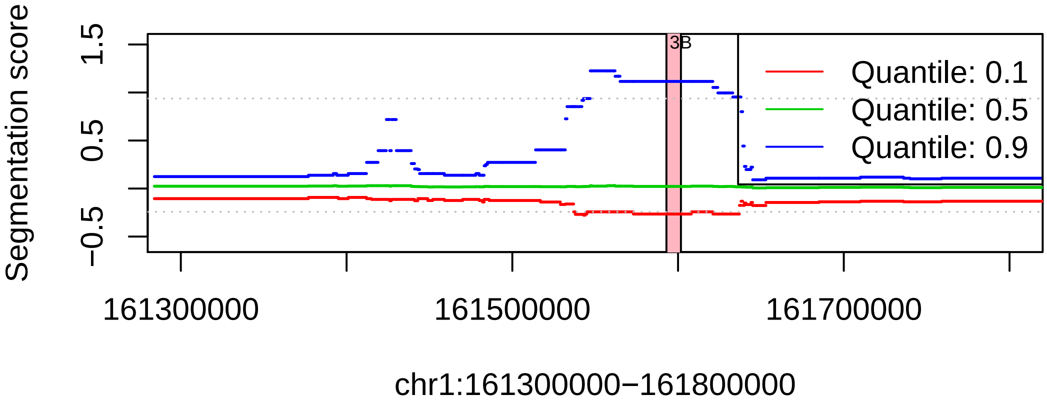
In the function *identifyPolymorphicRegion*, if we would like to obtain only polymorphic regions which differentiate between populations (e.g., to detect evidence of selection) then we can use the option *VstTest=TRUE*. This option will calculate the Vst statistics (Redon *et~al.*, 2006). Users have 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*.

5 Note

If we use the option *entireGene = FALSE* 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)).

6 Session information

```
sessionInfo()

## R version 3.0.1 (2013-05-16)
## Platform: x86_64-pc-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
```



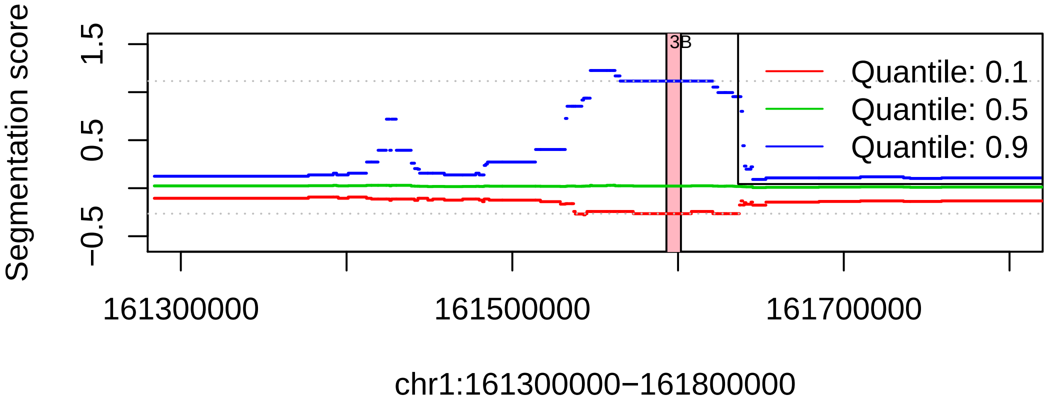


Figure 8: CN polymorphic region at FCGR3 locus, represented by quantiles of the distribution of segmentation scores across samples.

```
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=C                 LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets
## [7] methods   base
##
## other attached packages:
##  [1] CNVrd2_0.99.0           DNAcopy_1.34.0
##  [3] rjags_3-10              coda_0.16-1
##  [5] lattice_0.20-15         VariantAnnotation_1.6.6
##  [7] Rsamtools_1.12.3        Biostrings_2.28.0
##  [9] GenomicRanges_1.12.4    IRanges_1.18.2
## [11] BiocGenerics_0.6.0      codetools_0.2-8
## [13] knitr_1.2
##
## loaded via a namespace (and not attached):
##  [1] AnnotationDbi_1.22.6   Biobase_2.20.1
##  [3] biomaRt_2.16.0         bitops_1.0-5
##  [5] BSgenome_1.28.0        DBI_0.2-7
##  [7] digest_0.6.3           evaluate_0.4.4
##  [9] formatR_0.8            GenomicFeatures_1.12.3
## [11] grid_3.0.1             RCurl_1.95-4.1
## [13] RSQLite_0.11.4         rtracklayer_1.20.4
## [15] stats4_3.0.1           stringr_0.6.2
## [17] tools_3.0.1            XML_3.98-1.1
## [19] zlibbioc_1.6.0
```

References

Danecek P, Auton A, Abecasis G, Albers C, Banks E, DePristo M, Handsaker R, Lunter G, Marth G, Sherry S, *et˜al.* (2011). "The variant call format and VCFtools." *Bioinformatics*, **27**(15), 2156–2158.

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, *et˜al.* (2009). "The sequence alignment/map format and SAMtools." *Bioinformatics*, **25**(16), 2078–2079.

Nguyen HT, Merriman TR, Black MA (2013). "CNVrd, a Read-Depth Algorithm for Assigning Copy-Number at the FCGR Locus: Population-Specific Tagging of Copy Number Variation at FCGR3B." *PLOS ONE*, **8**(4), e63219.

Plummer M (2013). *rjags: Bayesian graphical models using MCMC*. R package version 3-10, URL `http://CRAN.R-project.org/package=rjags`.

Redon R, Ishikawa S, Fitch KR, Feuk L, Perry GH, Andrews TD, Fiegler H, Shapero MH, Carson AR, Chen W, *et˜al.* (2006). "Global variation in copy number in the human genome." *nature*, **444**(7118), 444–454.

Schwarz G (1978). "Estimating the dimension of a model." *The annals of statistics*, **6**(2), 461–464.

Venkatraman E, Olshen AB (2007). "A faster circular binary segmentation algorithm for the analysis of array CGH data." *Bioinformatics*, **23**(6), 657–663.