Copy number calling and SNV classification using targeted short read sequencing

Markus Riester [1]

[1]Novartis Institutes for BioMedical Research, Cambridge, MA

May 27, 2017

Abstract

PureCN [1] is a purity and ploidy aware copy number caller for cancer samples inspired by the *ABSOLUTE* algorithm [2]. It was designed for hybrid capture sequencing data, especially with medium-sized targeted gene panels without matching normal samples in mind (matched whole-exome data is of course supported).

It can be used to supplement existing normalization and segmentation algorithms, i.e. the software can start from BAM files, from target-level coverage data, from copy number log-ratios or from already segmented data. After the correct purity and ploidy solution was identified, *PureCN* will accurately classify variants as germline vs. somatic or clonal vs. sub-clonal.

PureCN was further designed to integrate well with industry standard pipelines [3], but it is straightforward to generate input data from other pipelines.

Package

PureCN 1.6.3

Contents

1 Introduction

This tutorial will demonstrate on a toy example how we recommend running *PureCN* on targeted sequencing data. To estimate tumor purity, we jointly utilize both target-level[1] coverage data and allelic fractions of single nucleotide variants (SNVs), inside - and optionally outside - the targeted regions. Knowledge of purity will in turn allow us to accurately (i) infer integer copy number and (ii) classify variants (somatic vs. germline, mono-clonal vs. sub-clonal, heterozygous vs. homozygous etc.).

[1]The captured genomic regions, e.g. exons.

This requires 3 basic input files:

1. A VCF file containing germline SNPs and somatic mutations. Somatic status is not required in case the variant caller was run without matching normal sample.
2. The tumor BAM file.
3. A BAM file from a normal control sample, either matched or process-matched.

In addition, we need to know a little bit more about the assay. This is the annoying step, since here the user needs to provide some information. Most importantly, we need to know the positions of all targets. Then we need to correct for GC-bias, for which we need GC-content for each target. Optionally, if gene-level calls are wanted, we also need for each target a gene symbol. To obtain best results, we can finally use a pool of normal samples to automatically learn more about the assay and its biases and common artifacts.

The next sections will show how to do all this with *PureCN* alone or with the help of *GATK* and/or existing copy number pipelines.

2 Basic input files

2.1 VCF

Germline SNPs and somatic mutations are expected in a single VCF file. At the bare minimum, this VCF should contain read depths of reference and alt alleles in an AD genotype field and a DB info flag for dbSNP membership. Without DB flag, variant ids starting with `rs` are assumed to be in dbSNP. If a matched normal is available, then somatic status information is currently expected in a SOMATIC info flag in the VCF. The *VariantAnnotation* package provides examples how to add info fields to a VCF in case the used variant caller does not add this flag. If the VCF contains a BQ genotype field containing base quality scores, *PureCN* can remove low quality calls.

VCF files generated by *MuTect* [4] should work well and in general require no post-processing. *PureCN* can handle *MuTect* VCF files generated in both tumor-only and matched normal mode. Experimental support for *MuTect 2* and *FreeBayes* VCFs generated in tumor-only mode is available.

2.2 Coverage data

For the default segmentation function provided by *PureCN*, the algorithm first needs to calculate log-ratios of tumor vs. normal control coverage. Coverage data needs to be provided in *GATK DepthOfCoverage* format[2]:

[2]*GATK* will generate additional columns which are not shown here and which *PureCN* will ignore when provided

```
Target  total_coverage  average_coverage
chr1:69091-70009    0   0   0
chr1:367659-368598  6358    9.25
chr1:621096-622035  6294    9.16
```

The intervals define the captured genomic regions (targets) and are provided by the manufacturer of your capture kit[3]. Default parameters assume that these intervals do NOT include a "padding" to include flanking regions of targets. *PureCN* will automatically include variants in the 50bp flanking regions if the variant caller was either run without interval file or with interval padding (See section 13.2). Please double check that the genome version of the interval file matches the reference.

[3]While *PureCN* can use a pool of normal samples to learn which intervals are reliable and which not, it is highly recommended to provide the correct intervals. Garbage in, garbage out.

It is recommended to GC-normalize coverage (how this done is described later in Section 3). The GC-content for each target is expected in *GATK GCContentByInterval* format:

```
Target    gc_bias    Gene
chr1:69091-70009    0.427638737758433    OR4F5
chr1:367659-368598    0.459574468085106    OR4F29
chr1:621096-622035    0.459574468085106    OR4F3
```

The `Gene` column is optional and only required for providing gene-level copy number and LOH calls. This information should be available in the technical documentation of your capture kit. Simplistic example code for annotating target intervals with gene symbols is given in the Appendix of this vignette for testing purposes.

2.3 Generating coverage data without *GATK*

The `calculateBamCoverageByInterval` function can be used to generate the required coverage data from BAM files:

```
bam.file <- system.file("extdata", "ex1.bam", package="PureCN",
    mustWork = TRUE)
interval.file <- system.file("extdata", "ex1_intervals.txt",
    package="PureCN", mustWork = TRUE)

calculateBamCoverageByInterval(bam.file=bam.file,
 interval.file=interval.file, output.file="ex1_coverage.txt")
```

To calculate GC-content, *PureCN* provides the `calculateGCContentByInterval` function:

```
interval.file <- system.file("extdata", "ex2_intervals.txt",
        package = "PureCN", mustWork = TRUE)
reference.file <- system.file("extdata", "ex2_reference.fa",
    package = "PureCN", mustWork = TRUE)
calculateGCContentByInterval(interval.file, reference.file,
    output.file = "ex2_gc_file.txt")
```

`calculateGCContentByInterval` can also be used to create an interval file from a BED file for example:

```
bed.file <- system.file("extdata", "ex2_intervals.bed",
        package = "PureCN", mustWork = TRUE)
```

```
intervals <- import(bed.file)

calculateGCContentByInterval(intervals, reference.file,
    output.file = "ex2_gc_file.txt")
```

2.4 Third-party segmentation tools

PureCN integrates well with existing copy number pipelines. Instead of coverage data, the user then needs to provide either already segmented data or a wrapper function. This is described in Section 10.1.

2.5 Example data

We now load a few example files that we will use throughout this tutorial:

```
library(PureCN)

normal.coverage.file <- system.file("extdata", "example_normal.txt",
    package="PureCN")
normal2.coverage.file <- system.file("extdata", "example_normal2.txt",
    package="PureCN")
normal.coverage.files <- c(normal.coverage.file, normal2.coverage.file)
tumor.coverage.file <- system.file("extdata", "example_tumor.txt",
    package="PureCN")
seg.file <- system.file("extdata", "example_seg.txt",
    package = "PureCN")
vcf.file <- system.file("extdata", "example_vcf.vcf", package="PureCN")
gc.gene.file <- system.file("extdata", "example_gc.gene.file.txt",
    package="PureCN")
```

3 GC-bias

The algorithm works best when the coverage files are GC-normalized. We can easily create GC-normalized coverage files (Figure 1).

```
correctCoverageBias(normal.coverage.file, gc.gene.file,
    output.file="example_normal_loess.txt", plot.gc.bias=TRUE)
```

All the following steps in this vignette assume that the coverage data are GC-normalized. The example coverage files are already GC-normalized. Section 11.2 presents a convenient command line script for generating GC-normalized coverage data from BAM files or from *GATK* coverage files.

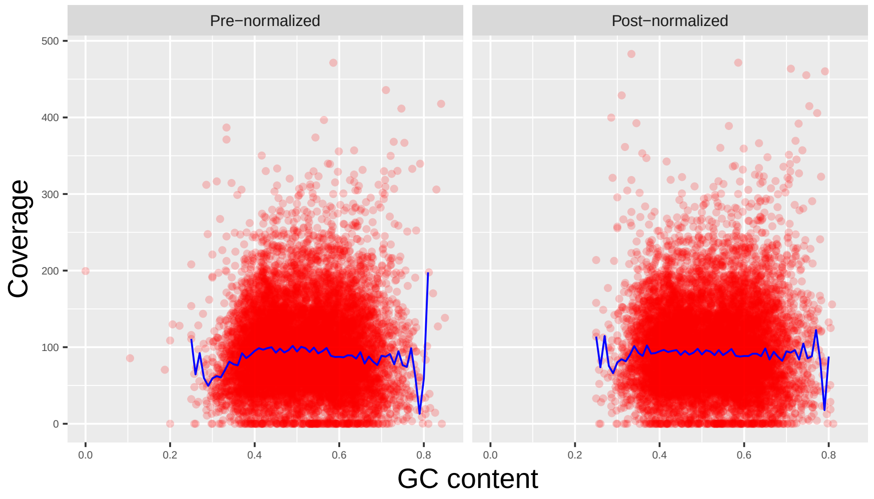

Figure 1: Coverage before and after GC-normalization
This plot shows coverage as a function of target GC-content before and after normalization. Each dot is a target interval. The example data is already GC-normalized; real data will show more dramatic differences.

4 Pool of normals

4.1 Selection of normals for log-ratio calculation

For calculating copy number log-ratios of tumor vs. normal, *PureCN* requires coverage from a process-matched normal sample. Using a normal that was sequenced using a similar, but not identical assay, rarely works, since differently covered genomic regions result in too many log-ratio outliers. This section describes how to identify a good process-matched normal in case no matched normal is available or in case the matched normal has low or uneven coverage.

The `createNormalDatabase` function builds a database of coverage files (a command line script proving this functionality is described in Section 11.3):

```
normalDB <- createNormalDatabase(normal.coverage.files)

## WARN [2017-05-27 20:29:38] Allosome coverage missing, cannot determine sex.
## WARN [2017-05-27 20:29:38] Allosome coverage missing, cannot determine sex.

# serialize, so that we need to do this only once for each assay
saveRDS(normalDB, file="normalDB.rds")
```

Again, please make sure that all coverage files were GC-normalized prior to building the database (Section 3). Internally, `createNormalDatabase` determines the sex of the samples and trains a PCA that is later used for clustering a tumor file with all normal samples in the database. This clustering is performed by the `findBestNormal` function:

```
normalDB <- readRDS("normalDB.rds")
# get the best normal
best.normal.coverage.file <- findBestNormal(tumor.coverage.file,
    normalDB)
```

This function can also return multiple normal files that can be averaged into a single pool:

```
# get the best 2 normals and average them
pool <- findBestNormal(tumor.coverage.file, normalDB,
    num.normals=2, pool=TRUE, remove.chrs=c("chrX", "chrY"))

## INFO [2017-05-27 20:29:39] Pooling example_normal.txt, example_normal2.txt.
```

This function will by default optimize averaging weights using the `voomWithQualityWeights` function of the *limma* package. The `num.normals` should be set to a value between 2 and 10. More than 10 usually results in long runtimes with no significant gain in accuracy.

Giving recommendations for optimal strategies is hard since they depend on the size, coverage and variance of the pool of normals. It is worth experimenting with different strategies and the `plotBestNormal` function might be helpful. Starting with the most simple one, i.e. using the single best normal, and then testing more complicated strategies, is a sound approach.

Note that this example removes coverage from sex chromosomes; if the normal database contains a sufficient number of samples with matching sex, `findBestNormal` will return only normal samples with matching sex.

4.2 Artifact filtering

It is important to remove as many artifacts as possible, since low ploidy solutions are typically punished more by artifacts than high ploidy solutions. High ploidy solutions are complex and usually find ways of explaining artifacts reasonably well. The following steps in this section are optional, but recommended since they will reduce the number of samples requiring manual curation, especially when matching normal samples are not available.

4.2.1 VCF

We recommend running *MuTect* with a pool of normal samples to filter common sequencing errors and alignment artifacts from the VCF. *MuTect* requires a single VCF containing all normal samples, for example generated by the *GATK CombineVariants* tool (see Section 13.2).

It is highly recommended to provide *PureCN* this combined VCF as well; it might help the software correcting non-reference read mapping biases. This is described in the `setMappingBiasVcf` documentation. To reduce memory usuage, the normal panel VCF can be reduced to contain only variants present in 5 or more samples (the VCF for *MuTect* should however contain variants present in 2-3 samples).

4.2.2 Coverage data

We next use coverage data of normal samples to estimate the expected variance in coverage per target:

```
target.weight.file <- "target_weights.txt"
createTargetWeights(tumor.coverage.file, normal.coverage.files,
    target.weight.file)

## INFO [2017-05-27 20:29:46] Loading coverage data...
## INFO [2017-05-27 20:29:46] Mean coverages: 112X (tumor) 99X (normal).
## INFO [2017-05-27 20:29:46] Mean coverages: 112X (tumor) 43X (normal).
```

This function calculates target-level copy number log-ratios using all normal samples provided in the `normal.coverage.files` argument. Assuming that all normal samples are in general diploid, a high variance in log-ratio is indicative of an target with either common germline alterations or frequent artifacts; high or low copy number log-ratios in these targets are unlikely measuring somatic copy number events. For the log-ratio calculation, we provide a coverage file that is used as tumor in the log-ratio calculation. The corresponding `tumor.coverage.file` argument can also be an array of a small number of coverage files, in which case the target coverage variance is averaged over all provided tumor files.

This `target.weight.file` is automatically generated by the NormalDB.R script described in Section 11.3.

4.3 Artifact filtering without a pool of normals

By default, *PureCN* will exclude targets with coverage below 15X from segmentation. For SNVs, the same 15X cutoff is applied. *MuTect* applies more sophisticated artifact tests and flags suspicious variants. If *MuTect* was run in matched normal mode, then both potential artifacts and germline variants are rejected, that means we cannot just filter by the PASS/REJECT *MuTect* flags. The `filterVcfMuTect` function optionally reads the *MuTect 1.1.7* stats file and will keep germline variants, while removing potential artifacts. Without the stats file, *PureCN* will use only the filters based on read depths as defined in `filterVcfBasic`. Both functions are automatically called by *PureCN*, but can be easily modified and replaced if necessary.

Instead of using a pool of normals to find SNPs with extremely biased allelic fractions, we can also use a BED file to blacklist regions of low mappability. For example the simple repeats track from the UCSC. This is recommended when neither matching normals nor a pool of normal VCF (Section 4.2) is available.

```r
# Instead of using a pool of normals to find low quality regions,
# we use suitable BED files, for example from the UCSC genome browser.

# We do not download these in this vignette to avoid build failures
# due to internet connectivity problems.
downloadFromUCSC <- FALSE
if (downloadFromUCSC) {
    library(rtracklayer)
    mySession <- browserSession("UCSC")
    genome(mySession) <- "hg19"
    simpleRepeats <- track( ucscTableQuery(mySession,
        track="Simple Repeats", table="simpleRepeat"))
    export(simpleRepeats, "hg19_simpleRepeats.bed")

    # when off-target reads are used, we can provide one of the
    # whole-genome blacklists tracks
    # hg19_DukeBlacklist <- track( ucscTableQuery(mySession,
    #     track="Mappability",
    #     table="wgEncodeDukeMapabilityRegionsExcludable"))
    # export(hg19_DukeBlacklist, "hg19_DukeBlacklist.bed")
}

snp.blacklist <- "hg19_simpleRepeats.bed"
```

5 Recommended run

Finally, we can run *PureCN* with all that information:

```
ret <-runAbsoluteCN(normal.coverage.file=pool,
# normal.coverage.file=normal.coverage.file,
    tumor.coverage.file=tumor.coverage.file, vcf.file=vcf.file,
    genome="hg19", sampleid="Sample1",
    gc.gene.file=gc.gene.file, normalDB=normalDB,
# args.setMappingBiasVcf=list(normal.panel.vcf.file=normal.panel.vcf.file),
# args.filterVcf=list(snp.blacklist=snp.blacklist,
# stats.file=mutect.stats.file),
    args.segmentation=list(target.weight.file=target.weight.file),
    post.optimize=FALSE, plot.cnv=FALSE, verbose=FALSE)

## WARN [2017-05-27 20:29:47] Allosome coverage missing, cannot determine sex.
## WARN [2017-05-27 20:29:47] Allosome coverage missing, cannot determine sex.
```

The `normal.coverage.file` argument points to a coverage file obtained from either a matched or a process-matched normal sample, but can be also a small pool of best normals (Section 4.1).

The `normalDB` argument (Section 4.1) provides a pool of normal samples and for example allows the segmentation function to skip targets with low coverage in the pool of normals. If available, a VCF containing all variants from the normal samples should be provided via `args.setMappingBiasVcf` to correct read mapping biases. The files specified in `args.filterVcf` help *PureCN* filtering SNVs more efficiently for artifacts as described in Sections 4.2 and 4.3. The `snp.blacklist` is only necessary if neither a matched normal nor a pool of normals is available.

The `post.optimize` flag will increase the runtime by about a factor of 2-5, but might return slightly more accurate purity estimates. For high quality whole-exome data, this is typically not necessary for copy number calling (but might be for variant classification, see Section 6.2.1). The `plot.cnv` argument allows the segmentation function to generate additional plots if set to `TRUE`. Finally, `verbose` outputs important and helpful information about all the steps performed and is therefore set to `TRUE` by default.

6 Output

6.1 Plots

We now create a few output files:

```
file.rds <- "Sample1_PureCN.rds"
saveRDS(ret, file=file.rds)
pdf("Sample1_PureCN.pdf", width=10, height=11)
plotAbs(ret, type="all")
dev.off()

## pdf
##   2
```

The RDS file now contains the serialized return object of the `runAbsoluteCN` call. The PDF contains helpful plots for all local minima, sorted by likelihood. The first plot in the generated PDF is displayed in Figure 2 and shows the purity and ploidy local optima, sorted by final likelihood score after fitting both copy number and allelic fractions.

```
plotAbs(ret, type="overview")
```

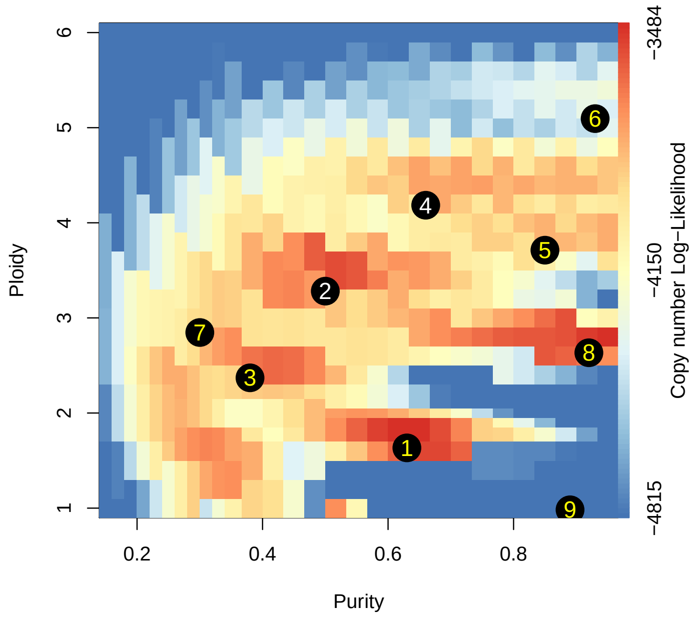

Figure 2: Overview
The colors visualize the copy number fitting score from low (blue) to high (red). The numbers indicate the ranks of the local optima. Yellow fonts indicate that the corresponding solutions were flagged, which does not necessarily mean the solutions are wrong. The correct solution (number 1) of this toy example was flagged due to large amount of LOH.

We now look at the main plots of the maximum likelihood solution in more detail.

```
plotAbs(ret, 1, type="hist")
```

Figure 3 displays a histogram of tumor vs. normal copy number log-ratios for the maximum likelihood solution (number 1 in Figure 2). The height of a bar in this plot is proportional to the fraction of the genome falling into the particular log-ratio copy number range. The vertical dotted lines and numbers visualize the, for the given purity/ploidy combination, expected log-ratios for all integer copy numbers from 0 to 7. It can be seen that most of the log-ratios of the maximum likelihood solution align well to expected values for copy numbers of 0, 1, 2 and 4.

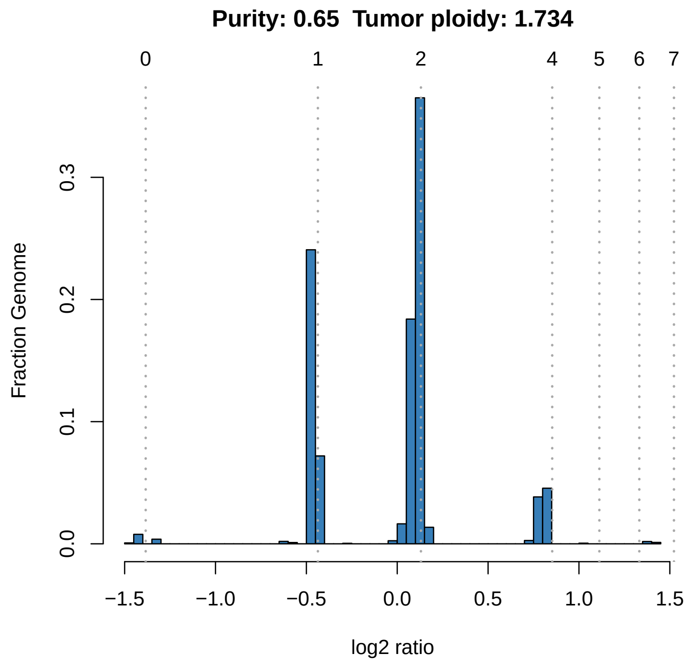

Figure 3: Log-ratio histogram

```r
plotAbs(ret, 1, type="BAF")
```

Germline variant data are informative for calculating integer copy number, because unbalanced maternal and paternal chromosome numbers in the tumor portion of the sample lead to unbalanced germline allelic fractions. Figure 4 shows the allelic fractions of predicted germline SNPs. The goodness of fit (GoF) is provided on an arbitrary scale in which 100% corresponds to a perfect fit and 0% to the worst possible fit. The latter is defined as a fit in which allelic fractions on average differ by 0.2 from their expected fractions. Note that this does not take purity into account and low purity samples are expected to have a better fit. In the middle panel, the corresponding copy number log-ratios are shown. The lower panel displays the calculated integer copy numbers, corrected for purity and ploidy. We can zoom into particular chromosomes (Figure 5).

```r
plotAbs(ret, 1, type="BAF", chr="chr19")
```

```r
plotAbs(ret, 1, type="AF")
```

Finally, Figure 6 provides more insight into how well the variants fit the expected values.

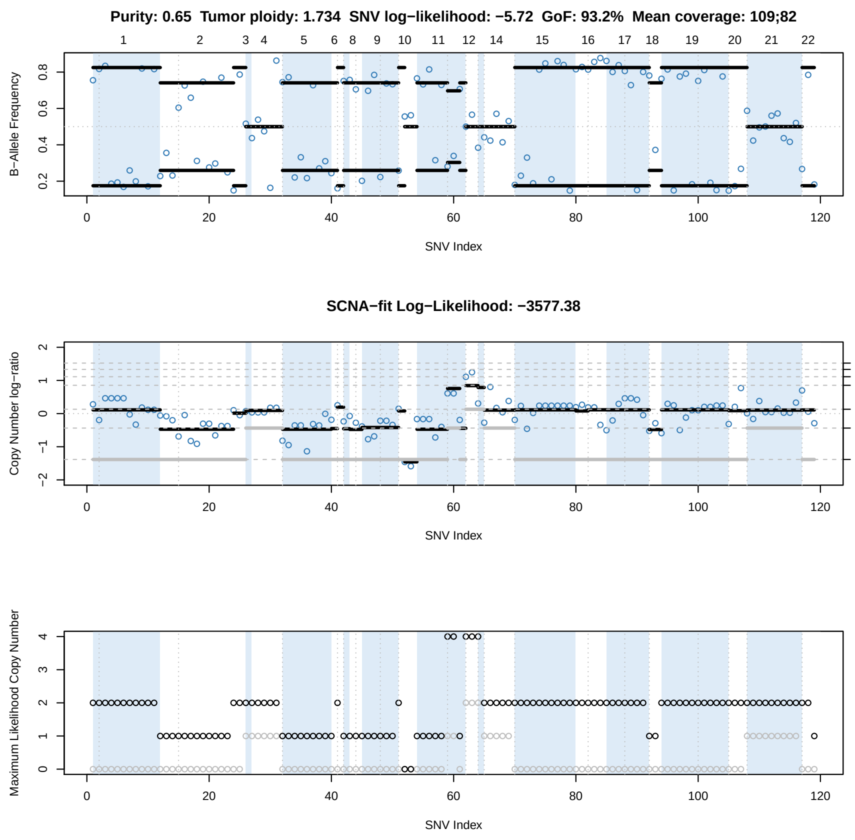

Figure 4: B-allele frequency plot

Each dot is a (predicted) germline SNP. The first panel shows the allelic fractions as provided in the VCF file. The alternating blue and white background colors visualize odd and even chromosome numbers, respectively. The black lines visualize the expected (not the average!) allelic fractions in the segment. These are calculated using the estimated purity and the total and minor segment copy numbers. These are visualized in black and grey, respectively, in the second and third panel. The second panel shows the copy number log-ratios, the third panel the integer copy numbers.

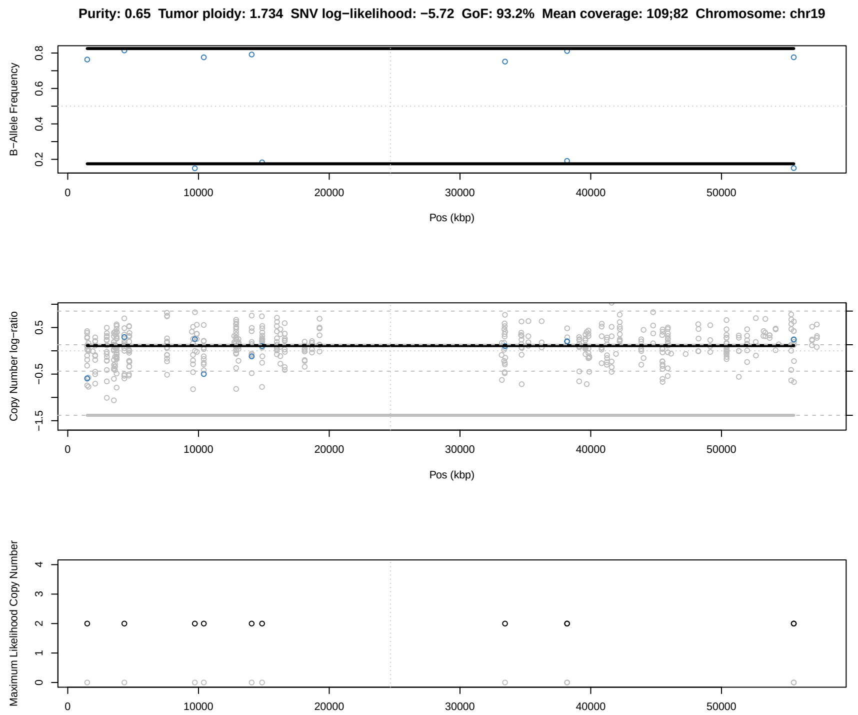

Figure 5: **Chromosome plot**
Similar to Figure 4, but zoomed into a particular chromosome. The grey dots in the middle panel visualize copy number log-ratios of targets without heterozygous SNPs, which are omitted from the previous genome-wide plot. The x-axis now indicates genomic coordinates in kbps.

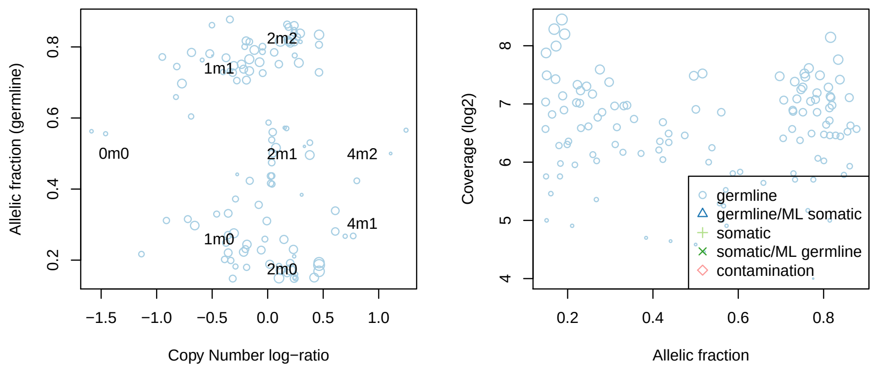

Figure 6: **Allele fraction plots**
Each dot is again a (predicted) germline SNP. The size of dots indicate quality, defined here as the product of mapping bias and coverage. The shapes visualize the different SNV groups based on prior and posterior probabilities. The labels show the expected values for all called states; 2m1 would be diploid, heterozygous, 2m2 diploid, homozygous. The relationship of allelic fraction and coverage is shown in the top right panel. This plot normally also shows somatic mutations in two additional panels, with the left panel showing the same plot as for germline SNPs and the bottom right panel a histogram of cellular fraction of predicted somatic mutations. This toy example contains only germline SNPs however.

6.2 Data structures

The R data file (`file.rds`) contains gene-level copy number calls, SNV status and LOH calls. The purity/ploidy combinations are sorted by likelihood and stored in `ret$results`.

```
names(ret)

## [1] "candidates" "results"    "input"
```

We provide convenient functions to extract information from this data structure and show their usage in the next sections. We recommend using these functions instead of accessing the data directly since data structures might change in future versions.

6.2.1 Prediction of somatic status and cellular fraction

To understand allelic fractions of particular SNVs, we must know the (i) somatic status, the (ii) tumor purity, the (iii) local copy number, as well as the (iv) number of chromosomes harboring the mutations or SNPs. One of *PureCN* main functions is to find the most likely combination of these four values. We further assign posterior probabilities to all possible combinations or states. Availability of matched normals reduces the search space by already providing somatic status.

The `predictSomatic` function provides access to these probabilities. For predicted somatic mutations, this function also provides cellular fraction estimates, i.e. the fraction of tumor cells with mutation. Fractions significantly below 1 indicate sub-clonality[4]:

[4]This number can be above 1 when the observed allelic fraction is higher than expected for a clonal mutation. This may be due to random sampling, wrong copy number, sub-clonal copy number events, or wrong purity/ploidy estimates.

```
head(predictSomatic(ret), 3)

##                    chr     start       end    SOMATIC.M0   SOMATIC.M1
## chr1114515871xxx chr1 114515871 114515871 1.134700e-101 1.601833e-38
## chr1150044293xxx chr1 150044293 150044293  1.993040e-91 8.148404e-39
## chr1158449835xxx chr1 158449835 158449835 4.785366e-147 5.033359e-62
##                    SOMATIC.M2    SOMATIC.M3 SOMATIC.M4 SOMATIC.M5 SOMATIC.M6
## chr1114515871xxx 2.740821e-07 3.445243e-258          0          0          0
## chr1150044293xxx 3.034580e-10 2.902343e-260          0          0          0
## chr1158449835xxx 7.514223e-15 3.608361e-262          0          0          0
##                  SOMATIC.M7   GERMLINE.M0  GERMLINE.M1 GERMLINE.M2
## chr1114515871xxx          0  3.852893e-69 1.075412e-15   0.9999997
## chr1150044293xxx          0  7.365854e-64 1.541661e-18   1.0000000
## chr1158449835xxx          0 3.661565e-105 9.013783e-30   1.0000000
##                    GERMLINE.M3 GERMLINE.M4 GERMLINE.M5 GERMLINE.M6
## chr1114515871xxx 3.561635e-256           0           0           0
## chr1150044293xxx 1.951352e-254           0           0           0
## chr1158449835xxx 5.707828e-254           0           0           0
##                  GERMLINE.M7 GERMLINE.CONTHIGH GERMLINE.CONTLOW
## chr1114515871xxx           0      2.458486e-42    5.657574e-288
## chr1150044293xxx           0      9.428729e-21    1.122262e-242
## chr1158449835xxx           0      3.555243e-26     0.000000e+00
##                  GERMLINE.HOMOZYGOUS ML.SOMATIC POSTERIOR.SOMATIC ML.M ML.C
## chr1114515871xxx                   0      FALSE      2.740821e-07    2    2
## chr1150044293xxx                   0      FALSE      3.034580e-10    2    2
## chr1158449835xxx                   0      FALSE      7.514223e-15    2    2
##                    ML.M.SEGMENT ML.AR       AR AR.ADJUSTED MAPPING.BIAS ML.LOH
## chr1114515871xxx            0 0.825 0.755183   0.7760674    0.9730894   TRUE
## chr1150044293xxx            0 0.825 0.817078   0.8396741    0.9730894   TRUE
## chr1158449835xxx            0 0.825 0.834266   0.8573374    0.9730894   TRUE
##                    CN.SUBCLONAL CELLFRACTION FLAGGED  log.ratio depth
## chr1114515871xxx          FALSE           NA   FALSE  0.2807083   184
## chr1150044293xxx          FALSE           NA   FALSE -0.1940654   138
## chr1158449835xxx          FALSE           NA   FALSE  0.4630149   217
##                    prior.somatic prior.contamination on.target seg.id
## chr1114515871xxx         9.9e-05                0.01         1      1
## chr1150044293xxx         9.9e-05                0.01         1      1
## chr1158449835xxx         9.9e-05                0.01         1      1
##                    gene.symbol
## chr1114515871xxx         HIPK1
## chr1150044293xxx         VPS45
## chr1158449835xxx        OR10R2
```

The output columns are explained in Table 1.

To annotate the input VCF file with these values:

```
vcf <- predictSomatic(ret, return.vcf=TRUE)
writeVcf(vcf, file="Sample1_PureCN.vcf")
```

For optimal classification results:

- Set `post.optimize=TRUE` since small inaccuracies in purity can decrease the classification performance significantly
- Provide `args.setMappingBias` a pool of normal VCF to obtain position-specific mapping bias information
- Exclude variants in regions of low mappability
- Use a somatic posterior probability cutoff of 0.8 and 0.2 for somatic and germline variants, respectively. This appears to be a good compromise of call rate and accuracy.

Note that the posterior probabilities assume that the purity and ploidy combination is correct. Before classifying variants, it is thus important to manually curate samples.

6.2.2 Amplifications and deletions

To call amplifications, we recommend using a cutoff of 6 for focal amplifications and a cutoff of 7 otherwise. For homozygous deletions, a cutoff of 0.5 is useful to allow some heterogeneity in copy number.

For samples that failed *PureCN* calling we recommended using common log-ratio cutoffs to call amplifications, for example 0.9.

This strategy is implemented in the `callAlterations` function:

```
gene.calls <- callAlterations(ret)
head(gene.calls)

##             chr     start       end C seg.mean seg.id number.targets
```

Table 1: `predictSomatic` output columns

Column name	Description
`chr, start, end`	Variant coordinates
`SOMATIC.M*`	Posterior probabilities for all somatic states. M0 to M7 are multiplicity values, i.e. the number of chromosomes harboring the mutation (e.g. 1 heterozygous, 2 homozygous if copy number C is 2). SOMATIC.M0 represents a sub-clonal state (somatic mutations by definition have a multiplicity larger than 0).
`GERMLINE.M*`	Posterior probabilities for all heterozygous germline states
`GERMLINE.CONTHIGH`	Posterior probability for contamination. This state corresponds to homozygous germline SNPs that were not filtered out because reference alleles from another individual were sequenced, resulting in allelic fractions smaller than 1.
`GERMLINE.CONTLOW`	Posterior probability for contamination. This state corresponds to non-reference alleles only present in the contamination.
`GERMLINE.HOMOZYGOUS`	Posterior probability that SNP is homozygous in normal. Requires the `model.homozygous` option in `runAbsoluteCN`. See Section 8.
`ML.SOMATIC`	`TRUE` if the maximum likelihood state is a somatic state
`POSTERIOR.SOMATIC`	The posterior probability that the variant is somatic (sum of all somatic state posterior probabilities)
`ML.M`	Maximum likelihood multiplicity
`ML.C`	Maximum likelihood integer copy number
`ML.M.SEGMENT`	Maximum likelihood minor segment copy number
`ML.AR`	Expected allelic fraction of the maximum likelihood state
`AR`	Observed allelic fraction (as provided in VCF)
`AR.ADJUSTED`	Observed allelic fraction adjusted for mapping bias
`ML.LOH`	`TRUE` if variant is most likely in LOH
`CN.SUBCLONAL`	`TRUE` if copy number segment is sub-clonal
`CELLFRACTION`	Fraction of tumor cells harboring the somatic mutation
`FLAGGED`	Flag indicating that call is unreliable (currently only due to high mapping bias and high pool of normal counts)
`log.ratio`	The copy number log-ratio (tumor vs. normal) for this variant
`depth`	The total sequencing depth at this position
`prior.somatic`	Prior probability of variant being somatic
`prior.contamination`	Prior probability that variant is contamination from another individual
`on.target`	1 for variants within intervals, 2 for variants in flanking regions, 0 for off-target SNVs
`seg.id`	Segment id
`pon.count`	Number of hits in the `normal.panel.vcf.file`
`gene.symbol`	Gene symbol if available

```
## EIF2A    chr3 150264590 150301699 6   1.3685      5            14
## AADAC    chr3 151531951 151545961 6   1.3685      5             5
## GPNMB    chr7  23286477  23313844 7   1.4388     19            11
## SH2D4B  chr10  82300653  82403838 0  -1.4555     26             7
## IFIT2   chr10  91065719  91067133 0  -1.3335     28             1
## SLC35G1 chr10  95653791  95661248 0  -1.3335     28             3
##          gene.mean   gene.min  gene.max focal breakpoints       pvalue
## EIF2A    1.5119032  0.6971850  2.140631  TRUE           0 0.0002120768
## AADAC    0.8104999  0.4703347  1.201471  TRUE           1 0.0300191267
## GPNMB    1.4410240  0.9017780  1.822610  TRUE           0 0.0004220955
## SH2D4B  -1.4562558 -1.5868775 -1.260739 FALSE          0 0.0107040380
## IFIT2   -1.1437052 -1.1437052 -1.143705 FALSE          0 0.0099694165
## SLC35G1 -1.6626012 -1.8138843 -1.447743 FALSE          0 0.0028376310
##                  type num.snps.segment  loh
## EIF2A   AMPLIFICATION                0   NA
## AADAC   AMPLIFICATION                0   NA
## GPNMB   AMPLIFICATION                0   NA
## SH2D4B       DELETION                2 TRUE
## IFIT2        DELETION                0   NA
## SLC35G1      DELETION                0   NA
```

It is also often useful to filter the list further by known biology, for example to exclude non-focal amplifications of tumor suppressor genes. The Sanger Cancer Gene Census [5] for example provides such a list.

The output columns of `callAlterations` are explained in Table 2.

Table 2: `callAlterations` output columns

Column name	Description
`chr, start, end`	Gene coordinates
`C`	Segment integer copy number
`seg.mean`	Segment mean of copy number log-ratios (not adjusted for purity/ploidy)
`seg.id`	Segment id
`number.targets`	Number of targets annotated with this symbol
`gene.*`	Gene copy number log-ratios (not adjusted for purity/ploidy)
`focal`	`TRUE` for focal amplification, as defined by the `fun.focal` argument in `runAbsoluteCN`
`breakpoints`	Number of breakpoints between `start` and `end`
`pvalue`	Gene p-value against pool of normals. Requires `normalDB`. Not adjusted for multiple testing.
`num.snps.segment`	Number of SNPs in this segment informative for LOH detection
`loh`	`TRUE` if segment is in LOH, `FALSE` if not and `NA` if number of SNPs is insufficient
`type`	Amplification or deletion

6.2.3 Find genomic regions in LOH

The `gene.calls` data.frame described above provides gene-level LOH information. To find the corresponding genomic regions in LOH, we can use the `callLOH` function:

```
loh <- callLOH(ret)
head(loh)

##    chr     start       end arm C M                        type
## 1 chr1 114515871 121535434   p 2 0 WHOLE ARM COPY-NEUTRAL LOH
## 2 chr1 124535434 247419499   q 2 0 WHOLE ARM COPY-NEUTRAL LOH
## 3 chr2  10262881  92326171   p 1 0             WHOLE ARM LOH
## 4 chr2  95326171 231775198   q 1 0             WHOLE ARM LOH
## 5 chr2 236403412 239039169   q 2 0          COPY-NEUTRAL LOH
## 6 chr3  11888043  90504854   p 2 1
```

The output columns are explained in Table 3.

Table 3: `callLOH` output columns

Column name	Description
`chr, start, end`	Segment coordinates
`arm`	Chromosome arm
`C`	Segment integer copy number
`M`	Minor integer copy number $(M + N = C, M \leq N)$
`type`	LOH type if $M = 0$

7 Curation

For prediction of SNV status (germline vs. somatic, sub-clonal vs. clonal, homozygous vs. heterozygous), it is important that both purity and ploidy are correct. We provide functionality for curating results:

```
createCurationFile(file.rds)
```

This will generate a CSV file in which the correct purity and ploidy values can be manually entered. It also contains a column "Curated", which should be set to `TRUE`, otherwise the file will be overwritten when re-run.

Then in R, the correct solution (closest to the combination in the CSV file) can be loaded with the `readCurationFile` function:

```
ret <- readCurationFile(file.rds)
```

This function has various handy features, but most importantly it will re-order the local optima so that the curated purity and ploidy combination is ranked first. This means `plotAbs(ret,1,type="hist")` would show the plot for the curated purity/ploidy combination, for example.

The default curation file will list the maximum likelihood solution:

```
read.csv("Sample1_PureCN.csv")

##   Sampleid Purity   Ploidy Sex Contamination Flagged Failed Curated
## 1  Sample1   0.65 1.734105   ?             0    TRUE  FALSE   FALSE
##                           Comment
## 1 EXCESSIVE LOSSES;EXCESSIVE LOH
```

PureCN currently only flags samples with warnings, it does not mark any samples as failed. The `Failed` column in the curation file can be used to manually flag samples for exclusion in downstream analyses.

8 Cell lines

Default parameters assume some normal contamination. In 100% pure samples without matching normal samples such as cell lines, we cannot distinguish homozygous SNPs from LOH by looking at single allelic fractions alone. It is thus necessary to keep homozygous variants and include this uncertainty in the likelihood model. This is done by setting the `runAbsoluteCN` argument `model.homozygous=TRUE`. If matched normals are available, then variants homozygous in normal are automatically removed since they are uninformative.

For technical reasons, the maximum purity *PureCN* currently models is 0.99. We recommend setting `test.purity=seq(0.9,0.99,by=0.01)` in `runAbsoluteCN` for cell lines.

9 Maximizing the number of heterozygous SNPs

It is possible to use SNPs in off-target reads in the SNV fitting step by running *MuTect* without interval file and then setting the `filterVcfBasic` argument `remove.off.target.snvs` to `FALSE`. We recommend a large pool of normals in this case and then generating SNP blacklists as described in Sections 4.2 and 4.3. Remember to also run all the normals in *MuTect* without interval file.

An often better alternative to including all off-target reads is only including SNVs in the flanking regions of targets (between 50-100bp). This will usually significantly increase the number of heterozygous SNPs (see Section 13.2). These SNPs are automatically added if the variant caller was run without interval file or with interval padding.

10 Advanced usage

10.1 Custom normalization and segmentation

Copy number normalization and segmentation are crucial for obtaining good purity and ploidy estimates. If you have a well-tested pipeline that produces clean results for your data, you might want to use *PureCN* as add-on to your pipeline. By default, we will use *DNAcopy* [6] to segment normalized target-level coverage log-ratios. It is straightforward to replace the default with other methods and the `segmentationCBS` function can serve as an example.

The next section describes how to replace the default segmentation. For the probably more uncommon case that only the coverage normalization is performed by third-party tools, see Section 10.1.2.

10.1.1 Custom segmentation

It is possible to provide already segmented data, which is especially recommended when matched SNP6 data are available or when third-party segmentation tools are not written in R. Otherwise it is usually however better to customize the default segmentation function, since the algorithm then has access to the raw log-ratio distribution[5]. The expected file format for already segmented copy number data is[6]:

[5]If the third-party tool provides target-level log-ratios, then these can be provided via the `log.ratio` argument in addition to `seg.file` though. See also Section 10.1.2.

[6]This segmentation file can contain multiple samples, in which case the provided `sampleid` must match a sample in the column `ID`

```
ID  chrom   loc.start   loc.end num.mark    seg.mean
Sample1   1   61723   5773942 2681    0.125406444072723
Sample1   1   5774674 5785170 10  -0.756511807441712
```

Since its likelihood model is exon-based, *PureCN* currently still requires an interval file to generate simulated target-level log-ratios from a segmentation file. For simplicity, this interval file is expected either in *GATK DepthOfCoverage* format and provided via the `tumor.coverage.file` argument or via the `gc.gene.file` argument (see Figure 7). Note that *PureCN* will re-segment the simulated log-ratios using the default `segmentationCBS` function, in particular to identify regions of copy-number neutral LOH and to cluster segments with similar allelic imbalance and log-ratio. The provided interval file should therefore cover all significant copy number alterations[7]. Please check that the log-ratios are similar to the ones obtained by the default *PureCN* segmentation and normalization.

[7]If this behaviour is not wanted, because maybe the custom function already identifies CNNLOH reliably, `segmentationCBS` can be replaced with a minimal version.

```
retSegmented <- runAbsoluteCN(seg.file=seg.file,
    gc.gene.file=gc.gene.file, vcf.file=vcf.file,
    max.candidate.solutions=1, genome="hg19",
    test.purity=seq(0.3,0.7,by=0.05), verbose=FALSE,
    plot.cnv=FALSE)

## WARN [2017-05-27 20:32:15] Allosome coverage missing, cannot determine sex.
## WARN [2017-05-27 20:32:15] Allosome coverage missing, cannot determine sex.
```

The `max.candidate.solutions` and `test.purity` arguments are set to non-default values to reduce the runtime of this vignette.

```
plotAbs(retSegmented, 1, type="BAF")
```

10.1.2 Custom normalization

If third-party tools such as *GATK4* are used to calculate target-level copy number log-ratios, and *PureCN* should be used for segmentation and purity/ploidy inference only, it is possible to provide these log-ratios:

```
# We still use the log-ratio exactly as normalized by PureCN for this
# example
log.ratio <- calculateLogRatio(readCoverageFile(normal.coverage.file),
    readCoverageFile(tumor.coverage.file))

retLogRatio <- runAbsoluteCN(log.ratio=log.ratio,
    gc.gene.file=gc.gene.file, vcf.file=vcf.file,
    max.candidate.solutions=1, genome="hg19",
    test.purity=seq(0.3,0.7,by=0.05), verbose=FALSE,
    normalDB=normalDB, plot.cnv=FALSE)

## WARN [2017-05-27 20:32:46] Allosome coverage missing, cannot determine sex.
## WARN [2017-05-27 20:32:46] Allosome coverage missing, cannot determine sex.
```

Again, the `max.candidate.solutions` and `test.purity` arguments are set to non-default values to reduce the runtime of this vignette. It is highly recommended to compare the log-ratios obtained by *PureCN* and the third-party tool, since some pipelines automatically adjust log-ratios for a default purity value. Note that this example uses a pool of normals to

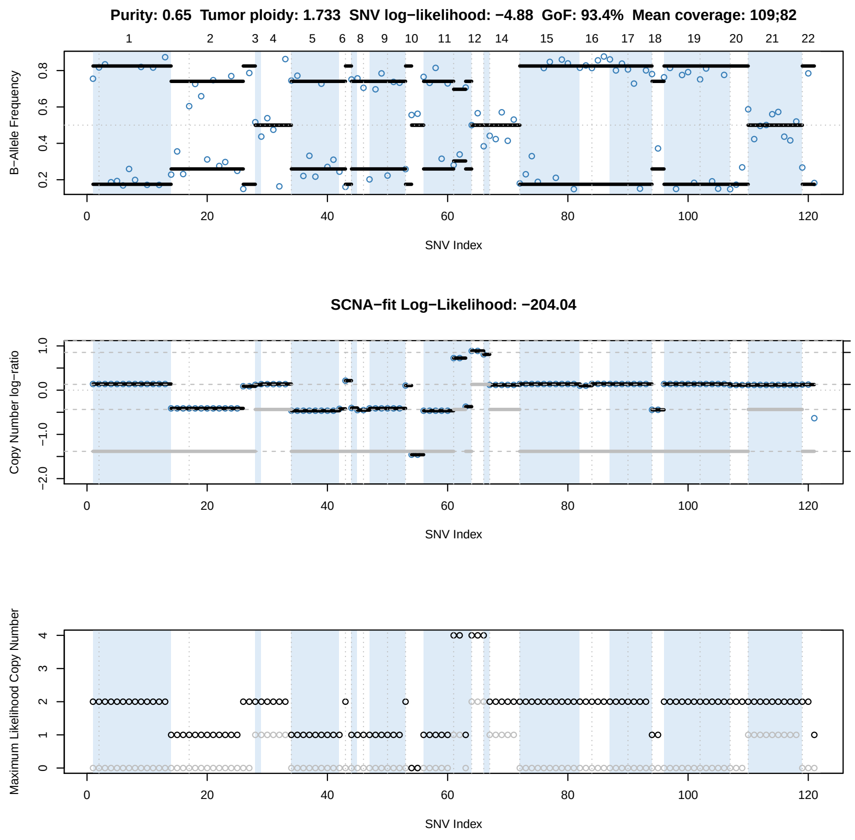

Figure 7: B-allele frequency plot for segmented data
This plot shows the maximum likelihood solution for an example where segmented data are provided instead of coverage data. Note that the middle panel shows no variance in log-ratios, since only segment-level log-ratios are available.

filter low quality targets. Interval coordinates are again expected in either a `gc.gene.file` or a `tumor.coverage.file`. If a tumor coverage file is provided, then all targets below the coverage minimum are further excluded.

10.2 COSMIC annotation

If a matched normal is not available, it is also helpful to provide `runAbsoluteCN` the COSMIC database [7] via `cosmic.vcf.file` (or via a `Cosmic.CNT` INFO field in the VCF). While this has limited effect on purity and ploidy estimation due the sparsity of hotspot mutations, it often helps in the manual curation to compare how well high confidence germline (dbSNP) vs. somatic (COSMIC) variants fit a particular purity/ploidy combination.

10.3 Power to detect somatic mutations

As final quality control step, we can test if coverage and tumor purity are sufficent to detect mono-clonal or even sub-clonal somatic mutations. We strictly follow the power calculation by Carter et al. [2].

The following Figure 8 shows the power to detect mono-clonal somatic mutations as a function of tumor purity and sequencing coverage (reproduced from [2]):

```
purity <- c(0.1,0.15,0.2,0.25,0.4,0.6,1)
coverage <- seq(5,35,1)
power <- lapply(purity, function(p) sapply(coverage, function(cv)
    calculatePowerDetectSomatic(coverage=cv, purity=p, ploidy=2,
    verbose=FALSE)$power))

# Figure S7b in Carter et al.
plot(coverage, power[[1]], col=1, xlab="Sequence coverage",
    ylab="Detection power", ylim=c(0,1), type="l")

for (i in 2:length(power)) lines(coverage, power[[i]], col=i)
abline(h=0.8, lty=2, col="grey")
legend("bottomright", legend=paste("Purity", purity),
    fill=seq_along(purity))
```

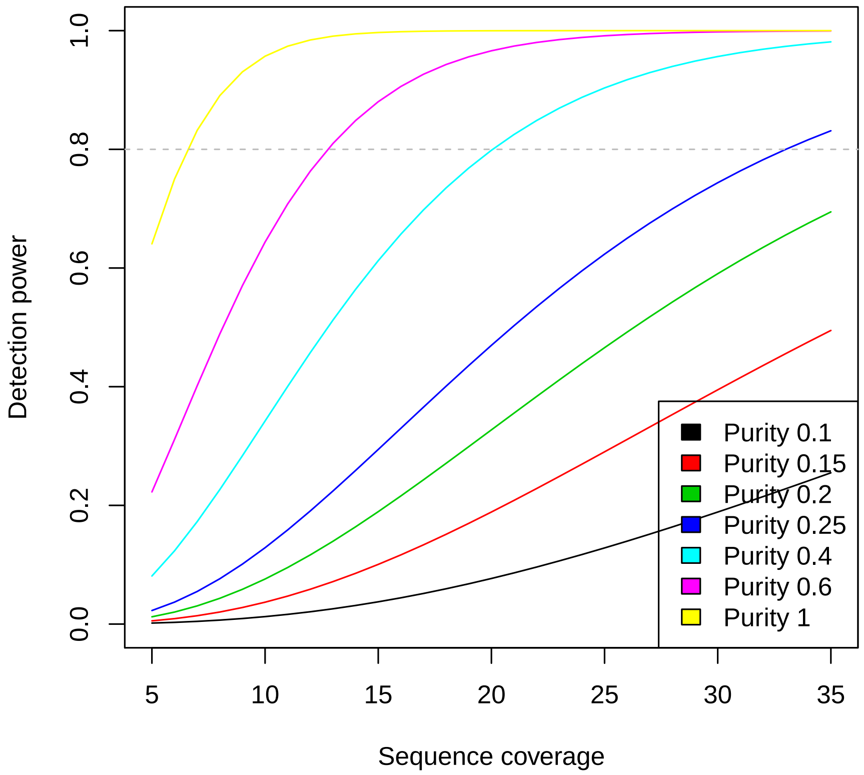

Figure 8: Power to detect mono-clonal somatic mutations
Reproduced from [2].

Figure 9 then shows the same plot for sub-clonal mutations present in 20% of all tumor cells:

```
coverage <- seq(5,350,1)
power <- lapply(purity, function(p) sapply(coverage, function(cv)
    calculatePowerDetectSomatic(coverage=cv, purity=p, ploidy=2,
    cell.fraction=0.2, verbose=FALSE)$power))
```

```
        plot(coverage, power[[1]], col=1, xlab="Sequence coverage",
            ylab="Detection power", ylim=c(0,1), type="l")

        for (i in 2:length(power)) lines(coverage, power[[i]], col=i)
        abline(h=0.8, lty=2, col="grey")
        legend("bottomright", legend=paste("Purity", purity),
            fill=seq_along(purity))
```

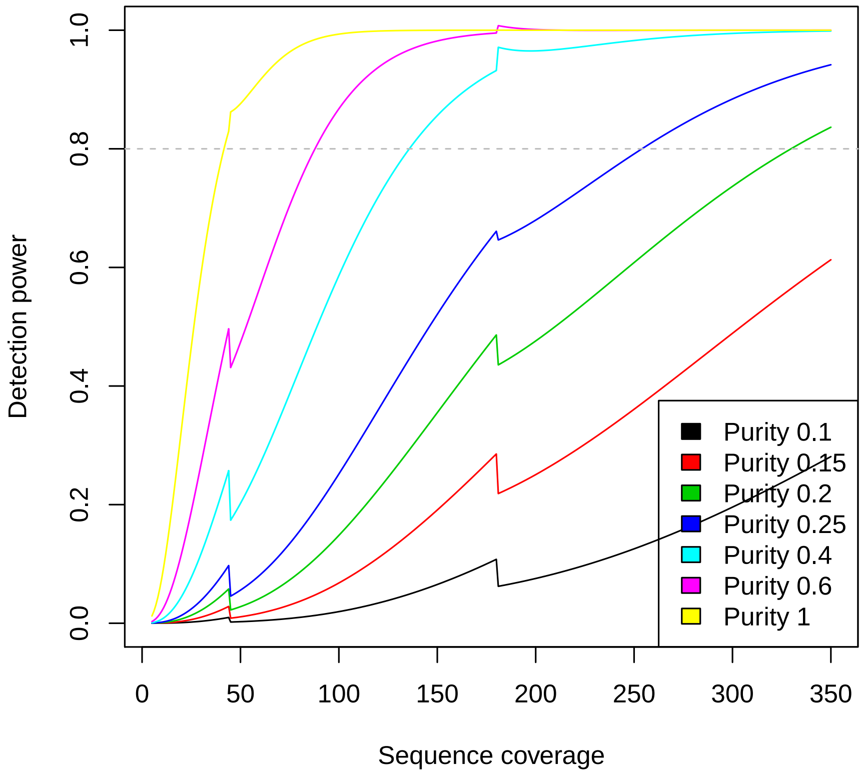

Figure 9: Power to detect sub-clonal somatic mutations present in 20% of all tumor cells
Reproduced from [2].

11 Command line scripts

The following section describe command line tools that provide a complete pipeline from BAM files.

11.1 IntervalFile

A simple helper script to convert a BED file containing target coordinates into a valid `gc.gene.file` (without gene symbols):

```
 Rscript IntervalFile.R --infile ex_intervals.bed --fasta ex_reference.fa \
 --outfile ex_gcgene.txt
```

In case third-party segmentation tools are used, you can now jump to Section 11.4.

Table 4: IntervalFile command line script

Argument name	Corresponding PureCN argument	PureCN function
–help -h		
–version -v		
–force -f		
–fasta -a	reference.file	`calculateGCContentByInterval`
–infile -i	interval.file	`calculateGCContentByInterval`
–outfile -o		

11.2 Coverage

We provide a basic template for GC-normalizing BAM or *GATK DepthOfCoverage* files from the command line. For example:

```
# From a BAM file (not working with provided toy example BAM file)
Rscript Coverage.R --outdir ~/tmp/ --bam ex1.bam \
     --gcgene ex1_gcgene.txt

# From a GATK DepthOfCoverage file
Rscript Coverage.R --outdir ~/tmp/ --gatkcoverage example_tumor.txt \
    --gcgene  example_gc.gene.file.txt
```

11.3 NormalDB

NormalDB.R is a convenient helper script that generates pool of normal files given a list of coverage files. It can automatically GC-normalize if necessary:

```
# From GATK DepthOfCoverage files, not GC-normalized
Rscript NormalDB.R --outdir ~/tmp --coveragefiles example_normal.list \
 --gcgene example_gc.gene.file.txt --genome hg19 --method LOESS

# From already GC-normalized files
```

Table 5: Coverage command line script

Argument name	Corresponding PureCN argument	PureCN function
–help -h		
–version -v		
–force -f		
–seed -S		
–outdir -o		
–bam -b	bam.file	`calculateBamCoverageByInterval`
–bai -a	index.file	`calculateBamCoverageByInterval`
–gatkcoverage -g	coverage.file	`correctCoverageBias`
–gcgene -c	gc.gene.file	`correctCoverageBias`
–method -m	method	`correctCoverageBias`

```
Rscript NormalDB.R --outdir ~/tmp --coveragefiles example_normal.list \
 --genome hg19
```

Table 6: NormalDB command line script

Argument name	Corresponding PureCN argument	PureCN function
–help -h		
–version -v		
–force -f		
–seed -S		
–outdir -o		
–coveragefiles -b	coverage.file	`correctCoverageBias`
–gcgene -c	gc.gene.file	`correctCoverageBias`
–method -m	method	`correctCoverageBias`
–assay -a	Optional assay name	Used in output file names.
–genome -g	Optional genome version	Used in output file names.

11.4 PureCN

PureCN.R is an example script to run PureCN with basic parameters:

```
# From GC-normalized coverage data
Rscript PureCN.R --outdir ~/tmp/ --tumor example_tumor.txt \
    --normal example_normal.txt --vcf example_vcf.vcf -i Sample1 \
    --genome hg19 --gcgene  example_gc.gene.file.txt

# Without a matched normal
Rscript PureCN.R --outdir ~/tmp/ --tumor example_tumor.txt \
    --normaldb normalDB.rds --pool 5 --vcf example_vcf.vcf -i Sample1 \
    --genome hg19 --gcgene example_gc.gene.file.txt

# From already segmented data
Rscript PureCN.R --outdir ~/tmp/ --segfile example_seg.txt \
    --vcf example_vcf.vcf -i Sample1 --genome hg19 \
    --funsegmentation none \
    --gcgene  example_gc.gene.file.txt

# Two-pass loess normalization from NOT YET GC-normalized tumor coverage
# data (normal controls need to be GC-normalized by the LOESS method)
Rscript PureCN.R --outdir ~/tmp/ --tumor example_tumor.txt \
    --normaldb normalDB.rds --pool 5 --vcf example_vcf.vcf -i Sample1 \
    --genome hg19 --gcgene example_gc.gene.file.txt \
    --twopass --postoptimize

# Recreate output after manual curation of Sample_purecn.csv
Rscript PureCN.R --rds Sample1_purecn.rds
```

Table 7: **PureCN command line script**

Argument name	Corresponding PureCN argument	PureCN function
–help -h		
–version -v		
–force -f		
–seed -S		
–outdir -o		
–normal -n	normal.coverage.file	`runAbsoluteCN`
–tumor -t	tumor.coverage.file	`runAbsoluteCN`
–vcf -b	vcf.file	`runAbsoluteCN`
–rds -r	file.rds	`readCurationFile`
–genome -g	genome	`runAbsoluteCN`
–gcgene -c	gc.gene.file	`runAbsoluteCN`
–segfile -l	seg.file	`runAbsoluteCN`
–snpblacklist -s	snp.blacklist	`filterVcfBasic`
–normal_panel -p	normal.panel.vcf.file	`setMappingBiasVcf`
–statsfile -a	stats.file	`filterVcfMuTect`
–targetweightfile -e	target.weight.file	`segmentationCBS`
–normaldb -d	normalDB (serialized with `saveRDS`)	`findBestNormal`, `filterTargets`
–pool -m	pool, num.normals	`findBestNormal`
–minaf -j	af.range	`filterVcfBasic`
–minpurity -k	test.purity	`runAbsoluteCN`
–maxpurity -x	test.purity	`runAbsoluteCN`
–minploidy -K	min.ploidy	`runAbsoluteCN`
–maxploidy -X	max.ploidy	`runAbsoluteCN`
–postoptimize -z	post.optimize	`runAbsoluteCN`
–modelhomozygous -y	model.homozygous	`runAbsoluteCN`
–model -q	model	`runAbsoluteCN`
–funsegmentation -w	fun.segmentation	`runAbsoluteCN`
–sampleid -i	sampleid	`runAbsoluteCN`
–twopass -T	purecn.output	`correctCoverageBias`
–outvcf -u	return.vcf	`predictSomatic`

These R scripts can be found in the extdata directory of the installed package.

12 Limitations

PureCN currently assumes a completely diploid normal genome. For human samples, it tries to detect sex by calculating the coverage ratio of chromosomes X and Y and will then remove sex chromosomes in male samples[8]. For non-human samples, the user needs to manually remove all non-diploid chromosomes from the coverage data and specify `sex="diploid"` in the *PureCN* call.

[8]Loss of Y chromosome (LOY) can result in wrong female calls, especially in high purity samples or if LOY is in both tumor and contaminating normal cells. The software throws a warning in this case when germline SNPs are provided.

While *PureCN* supports and models sub-clonal somatic copy number alterations, it currently assumes that the majority of alterations are mono-clonal. For most clinical samples, this is reasonable, but very heterogeneous samples are likely not possible to call without manual curation. Poly-genomic tumors are often called as high ploidy or low purity. The former usually happens when sub-clonal losses are called as 2 copies and mono-clonal losses correctly as 1 copy. The latter when sub-clonal losses are called mono-clonal, which only happens when there are far more sub-clonal than mono-clonal losses. Please note however that unless purities are very high, algorithms that model poly-genomic tumors do not necessarily have a higher call rate, since they tend to overfit noisy samples or similarly confuse true high-ploidy with poly-genomic tumors. Due to the lack of signal, manual curation is also recommended in low purity samples or very quiet genomes.

Finally, the software currently does not officially support VCF files containing indels. Support for VCFs generated by *MuTect 2* that include both single nucleotide variants (SNVs) and indels is planned for Bioconductor 3.5.

13 Support

If you encounter bugs or have problems running *PureCN*, please post them at https://support.bioconductor.org/p/new/post/?tag_val=PureCN.

If *PureCN* throws user errors, then there is likely a problem with the input files. If the error message is not self-explanatory, feel free to seek help at the support site. In your report, please add the outputs of the `runAbsoluteCN` call (with `verbose=TRUE`) and `sessionInfo()`. Please also check that your problem is not already covered in the following sections.

For general feedback such as suggestions for improvements, feature requests, complaints, etc. please do not hesitate to send us an email.

13.1 Checklist

- Used the correct interval files provided by the manufacturer of the capture kit and the genome version of the interval file matches the reference.
- BAM files were generated following established best practices and tools finished successfully.
- Checked standard QC metrics such as AT dropout.
- Tumor and normal data were obtained using the same capture kit and pipeline.
- Coverage data of tumor and normal were GC-normalized.
- The VCF file contains germline variants (i.e. not only somatic calls).

- Maximized the number of high coverage heterozygous SNPs, for example by running *MuTect* with a 50bp interval padding (Section 9).
- If a pool of normal samples is available, followed the steps in Section 4.2.
- Read the output of `runAbsoluteCN` with `verbose=TRUE`, fixed all warnings.
- If third-party segmentation tools are used, checked that normalized log-ratios are not biased, i.e. very similar compared to *PureCN* log-ratios (some pipelines already adjust for a default normal contamination).

13.2 FAQ

If the ploidy is frequently too high, please check:

- Does the log-ratio histogram (Figure 3) look noisy? If yes, then
  - Is the coverage sufficient? Tumor coverages below 80X can be difficult, especially in low purity samples. Normal coverages below 50X might result in high variance of log-ratios. See Section 4.1 for finding a good normal sample for log-ratio calculation.
  - Is the coverage data of both tumor and normal GC-normalized? If not, see `correctCoverageBias`.
  - Is the quality of both tumor and normal sufficient? A high AT or GC-dropout might result in high variance of log-ratios. Challenging FFPE samples also might need parameter tuning of the segmentation function. See `segmentationCBS`. A high expected tumor purity allows more aggressive segmentation parameters, such as `prune.hclust.h=0.2` or higher.
  - Was the correct target interval file used (genome version and capture kit, see Section 2.2)? If unsure, ask the help desk of your sequencing center.
  - Were the normal samples run with the same assay and pipeline?
  - Did you provide `runAbsoluteCN` all the recommended files as described in Section 5?
  - For whole-genome data, you will get better results using a specialized third-party segmentation method as described in section 10.1, since our default is optimized for targeted sequencing.
  - For smaller targeted panels, copy number events can interfere with the GC-normalization. Try either the `POLYNOMIAL` method in `correctCoverageBias` or the two-pass loess normalization (Section 11.4) if the data looks noisy. Third-party segmentation tools (Section 10.1) that utilize off-target reads might be also worth trying.
- Otherwise, if log-ratio peaks are clean as in Figure 3:
  - Was MuTect run without a matched normal? If yes, then make sure to provide either a pool of normal VCF or a SNP blacklist (if no pool of normal samples is available) as described in Sections 4.2 and 4.3.
  - A high fraction of sub-clonal copy-number alterations might also result in a low ranking of correct low ploidy solutions (see Section 12).

If the ploidy is frequently too low:

- *PureCN* with default parameters is conservative in calling genome duplications.
- This should only affect low purity samples ($< 35\%$), since in higher purity samples the duplication signal is usually strong enough to reliably detect it.
- In whole-exome data, it is usually safe to decrease the `max.homozygous.loss` default, since such large losses are rare.

Will PureCN work with my data?

- *PureCN* was designed for medium-sized (>2-3Mb) targeted panels. The more data, the better, best results are typically achieved in whole-exome data.
- The same is obviously true for coverage. Coverages below 80X are difficult unless purities are high and coverages are even.
- Samples with tumor purities below 20% usually cannot be analyzed with this algorithm and *PureCN* might return very wrong purity estimates.
- The number of heterozygous SNPs is also important (>1000 per sample). Copy number probes enriched in SNPs are therefore very helpful (see Section 9).
- While *PureCN* supports to some degree uneven tiling of targets, the more evenly distributed the better. Large gaps are problematic and might require third-party segmentation tools that support off-target reads (Section 10.1).
- Whole-genome data is not officially supported and specialized tools will likely provide better results. Third-party segmentation tools designed for this data type would be again required.
- Amplicon sequencing data is not supported.
- *PureCN* also needs process-matched normal samples, again, the more the better.

If you have trouble generating input data PureCN accepts, please check:

- For problems related to generating valid coverage data, either consult the *GATK* manual for the *DepthOfCoverage* tool or Section 2.3 for the equivalent function in *PureCN*.
- Currently only VCF files generated by *MuTect 1* are officially supported and well tested. A minimal example *MuTect* call would be:

```
$JAVA -Xmx6g -jar $MUTECT \
--analysis_type MuTect -R $REFERENCE \
--dbsnp $DBSNP_VCF \
--cosmic $COSMIC_VCF \
-I:normal $BAM_NORMAL \
-I:tumor $BAM_TUMOR  \
-o $OUT/${ID}_bwa_mutect_stats.txt \
-vcf $OUT/${ID}_bwa_mutect.vcf
```

The normal needs to be matched; without matched normal, only provide the tumor BAM file (do NOT provide a process-matched normal here). The default output file is the stats or call-stats file; this can be provided in addition to the required VCF file via `args.filterVcf` in `runAbsoluteCN`. If provided, it may help *PureCN* filter artifacts. This requires *MuTect* in version 1.1.7.

Note that this *MuTect* VCF will contain SNVs in off-target reads. By default, *PureCN* will only keep SNVs in the 50bp flanking regions of the provided targets. We highly recommend finding good values for each assay. A good cutoff will maximize the number of heterozygous SNPs and keep only an acceptable number of lower quality calls. This cutoff is set via `interval.padding` in `args.filterVcf`. See Section 9.

- For VCFs generated by other callers, the required dbSNP annotation can be added for example with *bcftools*:

```
bcftools annotate --annotation $DBSNP_VCF \
    --columns ID --output $OUT/${ID}_bwa_dbsnp.vcf.gz --output-type z \
        $OUT/${ID}_bwa.vcf.gz
```

- To generate the normal panel VCF (`normal.panel.vcf.file`) with *GATK*:
  - Run *MuTect* on the normal with `-I:tumor $BAM_NORMAL` and optionally with the `-artifact_detection_mode` flag.
  - Remove the empty `none` sample from the VCF:

```
$JAVA -Xmx6g -jar $GATK \
--analysis_type SelectVariants -R $REFERENCE \
--exclude_sample_expressions none \
-V $OUT/${ID}_bwa_mutect_artifact_detection_mode.vcf \
-o $OUT/${ID}_bwa_mutect_artifact_detection_mode_no_none.vcf
```

  - Merge the VCFs:

```
CMD="java  -Xmx12g -jar $GATK -T CombineVariants --minimumN 5 -R $REFERENCE"
for VCF in $OUT/*bwa_mutect_artifact_detection_mode_no_none.vcf;
do
    CMD="$CMD --variant $VCF"
done
CMD="$CMD -o $OUT/normals_merged_min5.vcf"
echo $CMD > $OUT/merge_normals_min5.sh
```

Questions related to pool of normals. As described in detail in Sections 4.1 and 4.2, a pool of normal samples is used in *PureCN* for coverage normalization (to adjust for target-specific capture biases) and for artifact filtering. A few recommendations:

- Matched normals are obviously perfect for identifying most alignment artifacts and mapping biases. While not critical, we still recommend generating a `normal.panel.vcf.file` for *MuTect* and `setMappingBiasVcf` using the available normals.
- Without matched normals, we need process-matched normals for coverage normalization. We recommend at least 2, ideally more than 5 from 2-3 different library preparation and sequencing batches.

- For artifact removal, the more normals available, the more rare artifacts are removed. We recommend at least 10, 50 or more are ideal.
- For position-specific mapping bias correction, the more normals are available, the more rare SNPs will have reliable mapping bias estimates. This requires at least about 25 normals to be useful, 100 or more are ideal.
- With smaller pool of normals, we additionally recommend filtering SNPs from low quality regions (Section 4.3). Additionally, it is worth trying the beta-binomial function instead of the default in the `model` argument of `runAbsoluteCN`. This will incorporate uncertainty of observed variant allelic fractions in the variant fitting step.

Questions related to manual curation. *PureCN*, like most other related tools, essentially finds the most simple explanation of the data. There are three major problems with this approach:

- First, hybrid capture data can be noisy and the algorithm must distinguish signal from noise; if the algorithm mistakes noise for signal, then this often results in wrong high ploidy calls (see Sections 4.2 and 4.3). If all steps in this vignette were followed, then *PureCN* should ignore common artifacts. Noisy samples thus often have outlier ploidy values and are often automatically flagged by *PureCN*. The correct solution is in most of these cases ranked second or third.
- The second problem is that signal can be sparse, i.e. when the tumor purity is very low or when there are only few somatic events. Manual curation is often easy in the latter case. For example when small losses are called as homozygous, but corresponding germline allele-frequencies are unbalanced (a complete loss would result in balanced germline allele frequencies, since only normal DNA is left). Future versions might improve calling in these cases by underweighting uninformative genomic regions.
- The third problem is that tumor evolution is fast and complex and very difficult to incorporate into general likelihood models. Sometimes multiple solutions explain the data equally well, but one solution is then often clearly more consistent with known biology, for example LOH in tumor suppressor genes such as *TP53*. A basic understanding of both the algorithm and the tumor biology of the particular cancer type are thus important for curation. Fortunately, in most cancer types, such ambiguity is rather rare. See also Section 12.

If all or most of the samples are flagged as:

Noisy segmentation: The default of 300 for `max.segments` is calibrated for high quality and high coverage whole-exome data. For whole-genome data or lower coverage data, this value needs to be re-calibrated. In case the copy number data looks indeed noisy, please see the first FAQ. Please be aware that *PureCN* will apply more aggressive segmentation parameters when the number of segments exceeds this cutoff. If the high segment count is real, this might confound downstream analyses.

High AT/GC dropout: If the data is GC-normalized, then there might be issues with either the target intervals or the provided GC content. Please double check that all files are correct and that all the coverage files are GC-normalized (Section 3).

References

[1] Markus Riester, Angad P. Singh, A. Rose Brannon, Kun Yu, Catarina D. Campbell, Derek Y. Chiang, and Michael P. Morrissey. PureCN: copy number calling and SNV classification using targeted short read sequencing. *Source code for biology and medicine*, 11:13, Dec 2016.

[2] Scott L. Carter, Kristian Cibulskis, Elena Helman, Aaron McKenna, Hui Shen, Travis Zack, Peter W. Laird, Robert C. Onofrio, Wendy Winckler, Barbara A. Weir, Rameen Beroukhim, David Pellman, Douglas A. Levine, Eric S. Lander, Matthew Meyerson, and Gad Getz. Absolute quantification of somatic DNA alterations in human cancer. *Nature biotechnology*, 30(5):413–421, May 2012.

[3] Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, and Mark A. DePristo. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome research*, 20(9):1297–1303, Sep 2010.

[4] Kristian Cibulskis, Michael S. Lawrence, Scott L. Carter, Andrey Sivachenko, David Jaffe, Carrie Sougnez, Stacey Gabriel, Matthew Meyerson, Eric S. Lander, and Gad Getz. Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples. *Nature biotechnology*, 31(3):213–219, Mar 2013.

[5] P. Andrew Futreal, Lachlan Coin, Mhairi Marshall, Thomas Down, Timothy Hubbard, Richard Wooster, Nazneen Rahman, and Michael R. Stratton. A census of human cancer genes. *Nature reviews. Cancer*, 4(3):177–183, Mar 2004.

[6] E. S. Venkatraman and Adam B. Olshen. A faster circular binary segmentation algorithm for the analysis of array CGH data. *Bioinformatics (Oxford, England)*, 23(6):657–663, Mar 2007.

[7] Simon A. Forbes, David Beare, Prasad Gunasekaran, Kenric Leung, Nidhi Bindal, Harry Boutselakis, Minjie Ding, Sally Bamford, Charlotte Cole, Sari Ward, Chai Yin Kok, Mingming Jia, Tisham De, Jon W. Teague, Michael R. Stratton, Ultan McDermott, and Peter J. Campbell. COSMIC: exploring the world's knowledge of somatic mutations in human cancer. *Nucleic acids research*, 43(Database issue):D805–D811, Jan 2015.

Annotating intervals with gene symbols

A simple function that annotates intervals with the **first** overlapping interval from a given *TxDb* database:

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
library(org.Hs.eg.db)

.annotateIntervals <- function(gc.gene.file, txdb, output.file = NULL) {
    gc <- read.delim(gc.gene.file, as.is=TRUE)
    # misuse this function to convert interval string into data.frame
    gc.data <- readCoverageFile(gc.gene.file)
    grGC <- GRanges(seqnames=gc.data$chr,
        IRanges(start=gc.data$probe_start, end=gc.data$probe_end))
    id <- transcriptsByOverlaps(txdb, ranges=grGC, columns = "GENEID")
    id$SYMBOL <-select(org.Hs.eg.db, as.character(id$GENEID), "SYMBOL")[,2]
    gc$Gene <- id$SYMBOL[findOverlaps(grGC, id, select="first")]
    if (!is.null(output.file)) {
        write.table(gc, file = output.file, row.names = FALSE,
            quote = FALSE, sep = "\t")
    }
    invisible(gc)
}

.annotateIntervals(gc.gene.file, TxDb.Hsapiens.UCSC.hg19.knownGene)
```

Session Info

- R version 3.4.0 (2017-04-21), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 16.04.2 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.5-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.5-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.38.0, Biobase 2.36.2, BiocGenerics 0.22.0, Biostrings 2.44.0, DNAcopy 1.50.1, DelayedArray 0.2.4, GenomeInfoDb 1.12.1, GenomicFeatures 1.28.0, GenomicRanges 1.28.3, IRanges 2.10.2, PureCN 1.6.3, Rsamtools 1.28.0, S4Vectors 0.14.2, SummarizedExperiment 1.6.2, TxDb.Hsapiens.UCSC.hg19.knownGene 3.2.2, VariantAnnotation 1.22.1, XVector 0.16.0, matrixStats 0.52.2, org.Hs.eg.db 3.4.1
- Loaded via a namespace (and not attached): BSgenome 1.44.0, BiocParallel 1.10.1, BiocStyle 2.4.0, DBI 0.6-1, GenomeInfoDbData 0.99.0, GenomicAlignments 1.12.1, Matrix 1.2-10, RColorBrewer 1.1-2, RCurl 1.95-4.8, RSQLite 1.1-2, Rcpp 0.12.11, VGAM 1.0-3, XML 3.98-1.7, backports 1.1.0, biomaRt 2.32.0, bitops 1.0-6, colorspace 1.3-2, compiler 3.4.0, data.table 1.10.4, digest 0.6.12, edgeR 3.18.1, evaluate 0.10, futile.logger 1.4.3, futile.options 1.0.0, ggplot2 2.2.1, grid 3.4.0, gtable 0.2.0, highr 0.6, htmltools 0.3.6, knitr 1.16, labeling 0.3, lambda.r 1.1.9, lattice 0.20-35, lazyeval 0.2.0, limma 3.32.2, locfit 1.5-9.1, magrittr 1.5, memoise 1.1.0, munsell 0.4.3, plyr 1.8.4, rlang 0.1.1, rmarkdown 1.5, rprojroot 1.2, rtracklayer 1.36.3, scales 0.4.1, splines 3.4.0, stringi 1.1.5, stringr 1.2.0, tibble 1.3.1, tools 3.4.0, yaml 2.1.14, zlibbioc 1.22.0