# PureCN - Quick Start

This tutorial provides a quick overview of the command line tools shipping with *PureCN*. For the R package and more detailed information, see the main vignette.

## Update from previous stable versions

*PureCN* is backward compatible with input generated by the previous stable version. For optimal results, we recommend starting from scratch following the tutorial below. *PureCN* 1.8 introduced off-target read support and minor improvements that are not available with previously generated data.

## Prepare environment and files

- Start R and enter the following to get the path to the command line scripts:

```
system.file("extdata", package="PureCN")

## [1] "/tmp/RtmptLuyLI/Rinstb971c57315b/PureCN/extdata"
```

- Exit R and store this path in an environment variable, for example in BASH:

```
$ export PURECN="/path/to/PureCN/extdata"
$ Rscript $PURECN/PureCN.R --help
Usage: /path/to/PureCN/inst/extdata/PureCN.R [options] ...
```

- Generate an interval file from a BED file containing target coordinates:

```
$ Rscript $PURECN/IntervalFile.R --infile baits_hg19.bed \
    --fasta hg19.fa --outfile baits_hg19_gcgene.txt \
    --offtarget --genome hg19 \
    --export baits_optimized_hg19.bed \
    --mappability wgEncodeCrgMapabilityAlign100mer.bigWig
```

  Internally, this script uses *rtracklayer* to parse the `infile`. Make sure that the file format matches the file extension. See the *rtracklayer* documentation for problems loading the file.

  The `-offtarget` flag will include off-target reads. Including them is recommended except for Amplicon data.

  The `-genome` version is needed to annotate exons with gene symbols.

  The `-export` argument is optional. If provided, this script will store the modified intervals as BED file for example (again every *rtracklayer* format is supported). This is useful when the coverages are calculated with third-party tools like GATK.

  The `-mappability` argument should provide a *rtracklayer* parsable file with a mappability score in the first meta data column. If provided, off-target regions will be restricted to regions specified in this file. On-target regions with low mappability will be excluded. For a test run, you can skip this argument or simply download the file from the UCSC website (see the FAQ section of the main vignette for instruction how to generate such a file).

# Create VCF files

*PureCN* does not ship with a variant caller. Use a third-party tool to generate a VCF for each sample. A few recommendations:

- Use *MuTect* 1.1.7 if possible
- Support for *MuTect 2* and *FreeBayes* is available for tumor-only VCFs, but currently poorly tested and only very limited artifact filtering will be performed for these callers.
- Since germline SNPs are needed to infer allele-specific copy numbers, the provided VCF needs to contain both somatic and germline variants.
- Run the variant caller with a 50 base pair interval padding to increase the number of heterozygous SNPs

# Run PureCN with internal segmentation

The following describes *PureCN* runs with internal copy number normalization and segmentation.

## Coverage

For each sample, tumor and normal:

```
# Calculate and GC-normalize coverage from a BAM file
$ Rscript $PURECN/Coverage.R --outdir $OUT/$SAMPLEID \
    --bam ${SAMPLEID}.bam \
    --gcgene baits_hg19_gcgene.txt

# GC-normalize coverage from a GATK DepthOfCoverage file
Rscript $PURECN/Coverage.R --outdir $OUT/$SAMPLEID \
    --coverage ${SAMPLEID}.coverage.sample_interval_summary \
    --gcgene  baits_hg19_gcgene.txt
```

Similar to GATK, this script also takes a text file containing a list of BAM or coverage file names (one per line). The file extension must be .list:

```
# Calculate and GC-normalize coverage from a list of BAM files
$ Rscript $PURECN/Coverage.R --outdir $OUT/$SAMPLEID \
    --bam normals.list \
    --gcgene baits_hg19_gcgene.txt \
    --cpu 4
```

## NormalDB

To build a normal database, copy the paths to all GC-normalized normal coverage files in a single text file, line-by-line:

```
ls -a normal*loess.txt | cat > example_normal.list

# From already GC-normalized files
```

```
$ Rscript $PURECN/NormalDB.R --outdir $OUT \
    --coveragefiles example_normal.list \
    --genome hg19
```

- The resulting `normalDB_hg19.rds` file contains absolute paths to the coverage files. It is thus necessary to re-run this command when the coverage files are moved.
- Consider generating different databases when differences are significant, e.g. for samples with different read lengths or insert size distributions
- In particular, do not mix normal data obtained with different capture kits (e.g. Agilent SureSelect v4 and v6)

## PureCN

Now that the assay-specific files are created and all coverages calculated, we run PureCN.R to normalize, segment and determine purity and ploidy:

```
cd $OUT/$SAMPLEID

# Without a matched normal (minimal test run)
$ Rscript $PURECN/PureCN.R --out . --tumor ${SAMPLEID}_coverage_loess.txt \
    --normaldb ../normalDB_hg19.rds \
    --sampleid $SAMPLEID \
    --vcf ${SAMPLEID}_mutect.vcf \
    --pool 5 \
    --genome hg19 --gcgene baits_hg19_gcgene.txt

# Production pipeline run
$ Rscript $PURECN/PureCN.R --out . --tumor ${SAMPLEID}_coverage_loess.txt \
    --normaldb ../normalDB_hg19.rds \
    --normal_panel $NORMAL_PANEL \
    --sampleid $SAMPLEID \
    --vcf ${SAMPLEID}_mutect.vcf \
    --statsfile ${SAMPLEID}_mutect_stats.txt \
    --pool 5 \
    --genome hg19 --gcgene baits_hg19_gcgene.txt
    --snpblacklist hg19_simpleRepeats.bed \
    --targetweightfile ../target_weights_hg19.txt
    --force --postoptimize

# With a matched normal (test run)
$ Rscript $PURECN/PureCN.R --out . --tumor ${SAMPLEID}_coverage_loess.txt \
    --normal ${SAMPLEID_NORMAL}_coverage_loess.txt \
    --sampleid $SAMPLEID \
    --vcf ${SAMPLEID}_mutect.vcf \
    --genome hg19 \
    --normaldb ../normalDB_hg19.rds \
    --gcgene baits_hg19_gcgene.txt

# Recreate output after manual curation of ${SAMPLEID}.csv
$ Rscript $PURECN/PureCN.R --rds ${SAMPLEID}.rds
```

A few important recommendations:

- Even if matched normals are available, it is often safer to use the normal database for coverage normalization
- Providing the normal database in addition to a matched normal is optional, but helps ignoring low quality regions in the segmentation
- The `-pool` flag specifies how many normal samples should be used to calculate the tumor vs. normal coverage. In a large, homogeneous database, this should be set to 10. In a heterogeneous database (for example different insert sizes, coverages, library preparation protocols, etc.), this value is set to an appropriate group size between 2 and 10.
- The normal panel VCF file is useful for mapping bias correction and especially recommended without matched normals. See the FAQ of the main vignette how to generate this file. It is not essential for test runs.
- The *MuTect* 1.1.7 stats file (the main output file besides the VCF) should be provided for better artifact filtering. If the VCF was generated by a pipeline that performs good artifact filtering, this file is not needed.
- The `-postoptimize` flag defines that purity should be optimized using both variant allelic fractions and copy number instead of copy number only. This results in a significant runtime increase for whole-exome data.
- If `-out` is a directory, it will use the sample id as file prefix for all output files. Otherwise *PureCN* will use `-out` as prefix.

## Run PureCN with third-party segmentation

If you already have a segmentation from third-party tools (for example CNVkit, EXCAVATOR2). For a test run:

```
Rscript $PURECN/PureCN.R --out $OUT/$SAMPLEID  \
    --sampleid $SAMPLEID \
    --segfile $OUT/$SAMPLEID/${SAMPLEID}.cnvkit.seg \
    --vcf ${SAMPLEID}_mutect.vcf \
    --genome hg19 --gcgene baits_hg19_gcgene.txt
```

See the main vignette for more details and file formats.

For a production pipeline run we provide a bit more information about the assay and genome:

```
# Recommended CNVkit example
Rscript $PURECN/PureCN.R --out $OUT/$SAMPLEID  \
    --sampleid $SAMPLEID \
    --tumor $OUT/$SAMPLEID/${SAMPLEID}.cnr \
    --segfile $OUT/$SAMPLEID/${SAMPLEID}_cnvkit.seg \
    --normal_panel $NORMAL_PANEL \
    --vcf ${SAMPLEID}_mutect.vcf \
    --statsfile ${SAMPLEID}_mutect_stats.txt \
    --snpblacklist hg19_simpleRepeats.bed \
    --genome hg19 \
    --funsegmentation none \
    --force --postoptimize
```

- The `-funsegmentation` argument controls if the data should to be re-segmented using germline BAFs (default). Set this value to `none` if the provided segmentation should be used as is.

## Dx

Dx.R extracts copy number and mutation metrics from PureCN.R output.

```
# Provide a BED file with callable regions, for examples obtained by
# GATK CallableLoci. Useful to calculate mutations per megabase and
# to exclude low quality regions.
grep CALLABLE ${SAMPLEID}_callable_status.bed > \
    ${SAMPLEID}_callable_status_filtered.bed

Rscript $PureCN/Dx.R --out . --rds ${SAMPLEID}.rds \
    --callable ${SAMPLEID}_callable_status_filtered.bed
```

# Reference

**Table 1: IntervalFile**

| Argument name | Corresponding PureCN argument | PureCN function |
|---|---|---|
| –fasta | reference.file | `calculateGCContentByInterval` |
| –infile | interval.file | `calculateGCContentByInterval` |
| –offtarget | off.target | `calculateGCContentByInterval` |
| –targetwidth | average.target.width | `calculateGCContentByInterval` |
| –offtargetwidth | average.off.target.width | `calculateGCContentByInterval` |
| –offtargetseqlevels | off.target.seqlevels | `calculateGCContentByInterval` |
| –mappability | mappability | `calculateGCContentByInterval` |
| –genome | txdb, org | `annotateTargets` |
| –outfile | | |
| –export | | `rtracklayer::export` |
| –version -v | | |
| –force -f | | |
| –help -h | | |

**Table 2: Coverage**

| Argument name | Corresponding PureCN argument | PureCN function |
|---|---|---|
| –bam | bam.file | `calculateBamCoverageByInterval` |
| –bai | index.file | `calculateBamCoverageByInterval` |
| –coverage | coverage.file | `correctCoverageBias` |
| –gcgene | gc.gene.file | `correctCoverageBias` |
| –method | method | `correctCoverageBias` |
| –keepduplicates | keep.duplicates | `calculateBamCoverageByInterval` |
| –outdir | | |
| –cpu | | Number of CPUs to use |
| –seed | | |
| –version -v | | |
| –force -f | | |
| –help -h | | |

**Table 3: NormalDB**

| Argument name | Corresponding PureCN argument | PureCN function |
|---|---|---|
| –coveragefiles | normal.coverage.files | `createNormalDatabase` |
| –maxmeancoverage | max.mean.coverage | `createNormalDatabase` |
| –assay -a | Optional assay name | Used in output file names. |
| –genome -g | Optional genome version | Used in output file names. |
| –outdir -o | | |
| –version -v | | |
| –force -f | | |
| –help -h | | |

**Table 4: PureCN**

| Argument name | Corresponding PureCN argument | PureCN function |
|---|---|---|
| –sampleid -i | sampleid | `runAbsoluteCN` |
| –normal | normal.coverage.file | `runAbsoluteCN` |
| –tumor | tumor.coverage.file | `runAbsoluteCN` |
| –vcf | vcf.file | `runAbsoluteCN` |
| –rds | file.rds | `readCurationFile` |
| –normal_panel | normal.panel.vcf.file | `setMappingBiasVcf` |
| –normaldb | normalDB (serialized with `saveRDS`) | `findBestNormal`, `filterTargets` |
| –segfile | seg.file | `runAbsoluteCN` |
| –sex | sex | `runAbsoluteCN` |
| –pool | pool, num.normals | `findBestNormal` |
| –genome | genome | `runAbsoluteCN` |
| –gcgene | gc.gene.file | `runAbsoluteCN` |
| –statsfile | stats.file | `filterVcfMuTect` |
| –minaf | af.range | `filterVcfBasic` |
| –snpblacklist | snp.blacklist | `filterVcfBasic` |
| –error | error | `runAbsoluteCN` |
| –funsegmentation | fun.segmentation | `runAbsoluteCN` |
| –alpha | alpha | `segmentationCBS` |
| –maxsegments | max.segments | `runAbsoluteCN` |
| –targetweightfile | target.weight.file | `segmentationCBS` |
| –minpurity | test.purity | `runAbsoluteCN` |
| –maxpurity | test.purity | `runAbsoluteCN` |
| –minploidy | min.ploidy | `runAbsoluteCN` |
| –maxploidy | max.ploidy | `runAbsoluteCN` |
| –postoptimize | post.optimize | `runAbsoluteCN` |
| –modelhomozygous | model.homozygous | `runAbsoluteCN` |
| –model | model | `runAbsoluteCN` |
| –logratiocalibration | log.ratio.calibration | `runAbsoluteCN` |
| –outvcf | return.vcf | `predictSomatic` |
| –out -o | | |
| –seed | | |
| –version -v | | |
| –force -f | | |
| –help -h | | |

**Table 5:** Dx

| Argument name | Corresponding PureCN argument | PureCN function |
|---|---|---|
| –rds | file.rds | `readCurationFile` |
| –callable | callable | `callMutationBurden` |
| –exclude | exclude | `callMutationBurden` |
| –out | | |
| –version -v | | |
| –force -f | | |
| –help -h | | |

## Session Info

- R version 3.4.2 (2017-09-28), `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 16.04.3 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.6-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.6-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: Biobase 2.38.0, BiocGenerics 0.24.0, Biostrings 2.46.0, DNAcopy 1.52.0, DelayedArray 0.4.0, GenomeInfoDb 1.14.0, GenomicRanges 1.30.0, IRanges 2.12.0, PureCN 1.8.0, Rsamtools 1.30.0, S4Vectors 0.16.0, SummarizedExperiment 1.8.0, VariantAnnotation 1.24.0, XVector 0.18.0, matrixStats 0.52.2
- Loaded via a namespace (and not attached): AnnotationDbi 1.40.0, BSgenome 1.46.0, BiocParallel 1.12.0, BiocStyle 2.6.0, DBI 0.7, GenomeInfoDbData 0.99.1, GenomicAlignments 1.14.0, GenomicFeatures 1.30.0, Matrix 1.2-11, R6 2.2.2, RColorBrewer 1.1-2, RCurl 1.95-4.8, RMySQL 0.10.13, RSQLite 2.0, Rcpp 0.12.13, VGAM 1.0-4, XML 3.98-1.9, assertthat 0.2.0, backports 1.1.1, biomaRt 2.34.0, bit 1.1-12, bit64 0.9-7, bitops 1.0-6, blob 1.1.0, colorspace 1.3-2, compiler 3.4.2, data.table 1.10.4-3, digest 0.6.12, edgeR 3.20.0, evaluate 0.10.1, futile.logger 1.4.3, futile.options 1.0.0, ggplot2 2.2.1, grid 3.4.2, gtable 0.2.0, highr 0.6, htmltools 0.3.6, knitr 1.17, labeling 0.3, lambda.r 1.2, lattice 0.20-35, lazyeval 0.2.1, limma 3.34.0, locfit 1.5-9.1, magrittr 1.5, memoise 1.1.0, munsell 0.4.3, plyr 1.8.4, prettyunits 1.0.2, progress 1.1.2, rlang 0.1.2, rmarkdown 1.6, rprojroot 1.2, rtracklayer 1.38.0, scales 0.5.0, splines 3.4.2, stringi 1.1.5, stringr 1.2.0, tibble 1.3.4, tools 3.4.2, yaml 2.1.14, zlibbioc 1.24.0