# Making a High Confident BED file

**Variables used in this document**

$hcdir = /groups/umcg-gdio/tmp01/umcg-mbeukers/highconfident\_bedfile/

$noxyhcdir = /groups/umcg-gdio/tmp01/umcg-mbeukers/noxy\_hc\_bedfile/

$scriptdir = /groups/umcg-gdio/tmp01/umcg-mbeukers/umcu\_exomedepth/scripts/

$hcdirscripts = {$hcdir}scripts/

## Goal of creating a High Confident BED file

“The overall goal of creating the High Confident BED file is to identify variable probes within a WES enrichtment design and exclude them from future WES CNV calling analysis. To create a High Confident (HC) BED file, BAM files of two runs, or populations, are required. Each population should have a 50/50 male-female ratio with a minimum of 60 samples for each population. Multiple sequence runs can be combined within a single population if required“

## Preparation

Creating a High Confident BED file from our captured.merged.bed requires a set of steps. First we need a working directory. In this document, my working directory will be `/groups/umcg-gdio/tmp01/umcg-mbeukers/highconfident\_bedfile/` to which I will refer to as $hcdir. This will be the directory in which all work to create the HC BED file will be performed.

Second, several UMCU Python3 scripts need to be installed, which are required during some of the steps. The scripts are available on github () of which only the ExomeDepth folder is required. These scripts are currently available on gearshift in `/groups/umcg-gdio/tmp01/umcg-mbeukers/umcu\_exomedepth/scripts/` to which I will refer to as $scriptdir. First activate the python3 venv using:

| source /groups/umcg-gdio/tmp01/umcg-mbeukers/umcu\_exomedepth/scripts/exomedepth/bin/activate |
| --- |

## Preparing the BED file

First we need to retain only three columns (chr, start, stop) in the BED file:

| cut -f1,2,3 /apps/data/Agilent/Exoom\_v3/human\_g1k\_v37/captured.merged.bed > captured.merged.bed |
| --- |

Then, we slice the BED file into intervals of 300 bases in length:

| python /groups/umcg-gdio/tmp01/umcg-mbeukers/umcu\_exomedepth/scripts/slice\_bed\_file.py captured.merged.bed 300 > sliced\_captured.merged.bed |
| --- |

## Making the two populations

To create the High Confident BED file, we need two populations of equal size. Each population should consist of a male and female group of samples of equal size. Furthermore the two populations should have the same number of samples. So the two populations could each have 30 female and 30 male samples. In the case of our test data, each male and female population had only 8 samples.

Make two sets of directories:

| mkdir -p populatie1/male  mkdir -p populatie1/female  mkdir -p populatie2/male  mkdir -p populatie2/female |
| --- |

Then link the samples via `ln -s`.

If you are not entirely sure about the sex of each samples in the populations you can run:

| python {$scriptdir}check\_gender\_bam.py {$hcdir}populatie1/male  python {$scriptdir}check\_gender\_bam.py {$hcdir}populatie1/female  python {$scriptdir}check\_gender\_bam.py {$hcdir}populatie2/male  python {$scriptdir}check\_gender\_bam.py {$hcdir}populatie2/female |
| --- |

Finally, we need to calculate the coverage for all the samples. This will be the longest step and may take some time. I have a script available at `/groups/umcg-gdio/tmp01/umcg-mbeukers/highconfident\_bedfile/scripts/generate\_sambamba.py` with which sambamba sbatch jobs can be generated, one for each sample in a directory:

| python {$hcdirscripts}generate\_sambamba.py {$hcdir}populatie1/female path/to/outputdir  python {$hcdirscripts}generate\_sambamba.py {$hcdir}populatie1/male path/to/outputdir  python {$hcdirscripts}generate\_sambamba.py {$hcdir}populatie2/female path/to/outputdir  python {$hcdirscripts}generate\_sambamba.py {$hcdir}populatie2/male path/to/outputdir |
| --- |

Once the jobs have been generated, you can submit them to the slurm scheduler. The sambamba jobs have a path to the sambamba 0.6.5 executable in one of my home directories. This is because sambamba 0.7 has an error in which the -q option, used in the commands, doesn’t work properly causing an error.

## Using the populations to create the HC BED file

After the coverage for all population samples has been determined, we calculate the statistics for female and male for each population:

| python {$scriptdir}filter\_probe\_file.py {$hcdir}populatie1/female > {$hcdir} populatie1\_female\_output\_all  python {$scriptdir}filter\_probe\_file.py {$hcdir}populatie1/male > {$hcdir} populatie1\_male\_output\_all  python {$scriptdir}filter\_probe\_file.py {$hcdir}populatie2/female > {$hcdir} populatie2\_female\_output\_all  python {$scriptdir}filter\_probe\_file.py {$hcdir}populatie2/male > {$hcdir} populatie2\_male\_output\_all |
| --- |

Next, we calculate the total number of targets in the sliced BED file twice. First the number of targets for chromosomes 1-22 and X are determined. Second, the number of targets for chromosome Y are determined. This results in two numbers: {auto\_chrX} and {chrY}. To calculate the number of targets:

| cat sliced\_captured.merged.bed | awk ‘($1 != “Y”)’ | wc -l  cat sliced\_captured.merged.bed | awk ‘($1 == “Y”)’ | wc -l |
| --- |

Next we use {auto\_chrX} and {chrY} to calculate the number of targets that should remain, {auto\_chrX\_remain} and {chrY\_remain}, as follows:

| {auto\_chrX} \* 0.95  {auto\_chrY} \* 0.33 |
| --- |

Once we know the number of remaining targets, we filter both populations. Female samples are filtered using {auto\_chrX\_remain} whereas male samples are filtered using {chrY\_remain}:

| cat {$hcdir}populatie1\_female\_output\_all | sed 's/inf/99999/g' | awk '($1 != "Y")' | sort -nk6 | head -n {auto\_chrX\_remain} |sed 's/X/999999999/g' | sort -nk1 -nk2 |sed 's/999999999/X/g' | awk '{OFS="\t"; print $1,$2,$3,$4"\_"$5"\_"$6}' > populatie1\_female\_auto\_chrX  cat {$hcdir}populatie1\_male\_output\_all | sed 's/inf/99999/g' | awk '($1 == "Y")' | sort -nk6 | head -n {chrY\_remain} |sed 's/X/999999999/g' | sort -nk1 -nk2 |sed 's/999999999/X/g' | awk '{OFS="\t"; print $1,$2,$3,$4"\_"$5"\_"$6}' > populatie1\_male\_chrY  cat {$hcdir}populatie2\_female\_output\_all | sed 's/inf/99999/g' | awk '($1 != "Y")' | sort -nk6 | head -n {auto\_chrX\_remain} |sed 's/X/999999999/g' | sort -nk1 -nk2 |sed 's/999999999/X/g' | awk '{OFS="\t"; print $1,$2,$3,$4"\_"$5"\_"$6}' > populatie2\_female\_auto\_chrX  cat {$hcdir}populatie2\_male\_output\_all | sed 's/inf/99999/g' | awk '($1 == "Y")' | sort -nk6 | head -n {chrY\_remain} |sed 's/X/999999999/g' | sort -nk1 -nk2 |sed 's/999999999/X/g' | awk '{OFS="\t"; print $1,$2,$3,$4"\_"$5"\_"$6}' > populatie2\_male\_chrY |
| --- |

After filtering, we merge the female and male for each population and check to make sure both populations have the same number of entries:

| cat populatie1\_female\_auto\_chrX populatie1\_male\_chrY > HC\_populatie1  cat populatie2\_female\_auto\_chrX populatie2\_male\_chrY > HC\_populatie2  wc -l HC\_populatie1  wc -l HC\_populatie2 |
| --- |

If both populations have the same number of entries we can continue with the process by calculating the percent overlap, {population\_overlap}, as follows: :

| cat HC\_populatie1 HC\_populatie2 | cut -f1,2,3 | sort | uniq -c | awk '($1==2)' | wc -l |
| --- |

Ideally, the population\_overlap percentage should be 99% or higher. With our test data however, the population overlap is at best 98.5%. This may be due to the relatively small number of samples used to create the two populations. The {population\_overlap} percentage can be calculated via:

| {population\_overlap} / ({auto\_chrX\_remain} + {chrY\_remain}) \* 100 |
| --- |

Once satisfied with the population overlap, the HighConfident BED file can be made:

| cat HC\_populatie1 HC\_populatie2 | cut -f1,2,3 | sort | uniq -c | awk '($1==2)' | sed 's/ \+/\t/g' |cut -f 3,4,5 | sed 's/X/999999999/g'| sed 's/Y/9999999999/g' | sort -nk1 -nk2 |sed 's/9999999999/Y/g' | sed 's/999999999/X/g' > HC\_target.bed |
| --- |

As well as the exon tsv file:

| cat HC\_target.bed | awk '{OFS="\t"; print $1,$2,$3,$1":"$2"-"$3 }' > HC\_exon.tsv |
| --- |

## Adding gene names to the HighConfident BED file

The High Confident BED file only contains three columns, namely chromosome, start and end position. I made two scripts to add gene names to the intervals in the High Confident BED file that are both available on gearshift: `/groups/umcg-gdio/tmp01/umcg-mbeukers`.

The first step is to make an alternate sliced BED file with `alt\_slice\_bed\_file.py`. This script does the same as the UMCU script `slice\_bed\_file.py` with the small difference that it retains the gene names and adds them to each interval.

To make the alternate sliced BED file:

| python alt\_slice\_bed\_file.py {$hcdir}captured.merged.bed 300 > {$hcdir}alt\_sliced\_captured.merged.bed |
| --- |

Once the alt\_sliced\_captured.merged.bed file has been created, it can be used to add gene names back to the actual High Confident BED file using the `add\_genenames\_to\_hcbedfile.py` script:

| python add\_genenames\_to\_hcbedfile.py {$hcdir}alt\_sliced\_captured.merged.bed {$hcdir}HC\_target.bed > {$hcdir}alt\_HC\_target.bed |
| --- |

# Creating the noXY HighConfident BED file

We also created a so-called noXY High Confident BED file. This HC BED file is similar to a normal one, with the exception that the X and Y chromosomes are excluded, hence the name noXY.

First we only retain chromosomes 1-22. Then we perform the two steps described in ‘Preparing the BED file’ above (retain the first three columns and slice intervals into 300bp).

Next, we make two population folders, without the male and female subfolders and place the samples for each population in the two population folders. For population1 we could select the first 8 female reference samples and first 33 male reference samples and for population2 the last 8 female reference samples and last 33 male reference samples for our test data for example.

Since we have a different BED file, you might want to recalculate the coverage with sambamba for these samples again. Currently you would need to modify the script to do so (future work to be able to provide a bed file)

After the coverage calculation, we again calculate the statistics, but only need to do so twice:

| python {$scriptdir}filter\_probe\_file.py {$noxyhcdir}populatie1/female > {$noxyhcdir} populatie1\_output\_all  python {$scriptdir}filter\_probe\_file.py {$noxyhcdir}populatie1/male > {$noxyhcdir} populatie2\_output\_all |
| --- |

Calculating the number of targets and the number of targets to remain is much easier as we don’t have to filter on the Y chromosome. The number of targets, {auto\_chr} can therefore we calculated as:

| wc -l noxy\_sliced\_captured.merged.bed |
| --- |

And the remaining targets, {auto\_chr\_remain}, simply as: {auto\_chr} \* 0.95.

Knowing {auto\_chr\_remain}, we then filter both populations:

| cat {$noxycdir}populatie1\_output\_all | sed 's/inf/99999/g' | sort -nk6 | head -n {auto\_chr\_remain} |sed 's/X/999999999/g' | sort -nk1 -nk2 |sed 's/999999999/X/g' | awk '{OFS="\t"; print $1,$2,$3,$4"\_"$5"\_"$6}' > populatie1\_autochr  cat {$noxycdir}populatie1\_output\_all | sed 's/inf/99999/g' | sort -nk6 | head -n {auto\_chr\_remain} |sed 's/X/999999999/g' | sort -nk1 -nk2 |sed 's/999999999/X/g' | awk '{OFS="\t"; print $1,$2,$3,$4"\_"$5"\_"$6}' > populatie2\_autochr |
| --- |

We can check that the two populations have the same number of entries and then calculate the overlap {population\_overlap}.

| wc -l populatie1\_autochr  wc -l populatie2\_autochr  cat populatie1\_autochr populatie2\_autochr | cut -f1,2,3 | sort | uniq -c | awk '($1==2)' | wc -l |
| --- |

As with the normal High Confident BED file, ideally the population overlap should be >= 99%. We can calculate this in a similar way:

| {population\_overlap} / {auto\_chr\_remain} \* 100 |
| --- |

And finally, we make our noXY High Confident BED file and exon tsv file:

| cat populatie1\_autochr populatie2\_autochr | cut -f1,2,3 | sort | uniq -c | awk '($1==2)' | sed 's/ \+/\t/g' |cut -f 3,4,5 | sed 's/X/999999999/g'| sed 's/Y/9999999999/g' | sort -nk1 -nk2 |sed 's/9999999999/Y/g' | sed 's/999999999/X/g' > noXY\_HC\_target.bed  cat noXY\_HC\_target.bed | awk '{OFS="\t"; print $1,$2,$3,$1":"$2"-"$3 }' > noXY\_HC\_exon.tsv |
| --- |