Pacybara and BarseqPro

Pacybara is a long-read barcode clustering method designed for multiplexed assays of variant effect (MAVEs).

BarseqPro is an accompanying BarSeq MAVE analysis pipeline.

Pre-requisites and installation:

1. Pacybara and barseqPro are designed for a Slurm or PBS HPC cluster.
   - A single-node version of pacybara is also provided, but not recommended due to very long runtimes.
2. Install `clusterutil`. This handles the abstraction between Slurm and PBS interfaces.
3. Download or clone the code repository:
   - Either via git: `git clone https://github.com/rothlab/pacybara.git`
   - Or as a simple download `wget https://github.com/rothlab/pacybara/archive/refs/heads/master.zip&&unzip master.zip`
4. Installation
   - Automatic installation (via conda and R):
     - Make sure anaconda (or miniconda) is installed.
     - Run `./install.sh`. This will install all dependencies in a new conda environment called `pacybara`.
   - Alternative manual installation:
     - Make sure python version 3.8 or higher is installed.
     - Make sure R version 4.1 or higher as well as the following R packages are installed: `yogitools`, `yogiseq`, `hgvsParseR`,`argparser`,`pbmcapply`,`hash`,`bitops`. yogitools, yogiutil and hgvsParser can be installed via `remotes::install_github("jweile/yogitools")`, etc
     - Make sure emboss, bwa, bowtie2, muscle and samtools are installed. (e.g. via conda: `conda install -c bioconda emboss bwa bowtie2 muscle samtools`)
     - Copy all scripts from the `src/` folder into a directory on your path, e.g. `cp src/* ~/bin/`

If you need to perform CCS processing/filtering from Pacbio CLR reads to prepare your long read inputs, we also recommend installing `pacbiotools`.

Running pacybara

1. Prepare a parameter sheet file. An example can be found further down in this document.
   - As part of the parameter sheet you will need to define the amplicon sequence and note the start and end positions of the ORF within. Also take note of the barcode degeneracy code(s) in the reference file. These will need to be recorded in the `ORFSTART`, `ORFEND` and `BARCODE` fields of the sheet.
   - Carefully choose the parameters for `MINBCQ` and `MINQUAL`, as they are dependent on the overall distribution of Q-scores in your input FASTA, which can drastically differ depending on the number of passes in the Pacbio run (and are usually grossly over-inflated). If your input fastq file was processed via DeepConsensus you will need lower these values, as DeepConsenus produces much more reasonable Q-values than CCS. For example, for CCS at > 5 passes, `MINQUAL=67` is a good starting point, whereas for DeepConsensus, `MINQUAL=27` would be better.
2. If you used the automatic installation script or manually installed the dependencies above via anaconda / miniconda, make sure to activate the corresponding environment. (The one created by the install.sh script is called 'pacybara'): `conda activate pacybara`.
3. Let's assume your parameter sheet is called `pacybara_parameters.txt`. You can then execute pacybara via:

`pacybara.sh pacybara_parameters.txt`.

However, it is recommended that you run pacybara as a HPC job. Assuming we'd like to request 12CPU cores, 24GB of RAM and 36 hours runtime for the job and pass on the `pacybara` conda environment: `submitjob.sh -n myPacybaraJob -c 12 -m 24G -t 36:00:00 -l pacybara.log -e pacybara.log --conda pacybara -- pacybara.sh pacybara_parameters.txt`

There are a number of optional parameters to deal with cluster idiosyncrasies:

```
pacybara.sh [-c|--cpus <CPUS>] [-q|--queue <QUEUE>] [-b|--blacklist {<NODE>,}] <PARAMETERS>

-c|--cpus      : Number of CPUs to use per node, defaults to 4
```

```
-q|--queue     : The HPC queue (or slurm partition) to use
-b|--blacklist : A comma-separated list of HPC nodes to avoid
<PARAMETERS>   : The parameter sheet file
```

If you don't have an HPC environment you can instead try out our experimental single-machine version. However, without the ability to run parallel jobs, this will take a very long time to run (i.e. days or even weeks!).

`pacybara_nomux.sh pacybara_parameters.txt`.

Pacybara Output

The output directory will contain multiple items:

- A bam file of all the reads aligned against the amplicon
- A tarball containing the log files of all the parallel alignments
- A directory ending in `_extract`. This contains fastq files of the extracted barcodes for each read and a file called `genotypes.csv.gz` which contains the extracted ORF genotypes for each read.
- A directory ending in `_clustering`. This contains a number of intermediate files, as well as the final `clusters_transl.csv.gz` and `clusters_transl_filtered.csv.gz`
  - `clusters_transl.csv.gz` contains an unfiltered table of all final clusters (i.e. clones) and barcodes and genotypes.
  - `clusters_transl_filtered.csv.gz` is the same, but filtered for clones supported by >1 CCS read and no barcode collisions.
  - `clusters_transl_softfilter.csv.gz` is filtered less strictly. It still requires >1 CCS read, but allows clones from barcode collisions as long as they dominate that barcode with at least a 2/3 majority.
  - Within the `*_clustering` directory you will also find a `qc/` sub-directory. This contains a number of QC plots.

Converting the Pacybara output for use with barseqPro.

The barseqPro software requires a library of barcode associations. This table can be made using any of the `clusters_transl*` tables, depending on the desired filtering level. I recommend using the `softfilter` version for best results. To convert it to the required format, you can use the `pacybara_preplib.R` script:

`pacybara_preplib.R clusters_transl_softfilter.csv.gz myBarcodeLibrary.csv --mode up2up`

Here is a the full help output for the script:

```
pacybara_preplib.R --help
usage: pacybara_preplib.R [--] [--help] [--opts OPTS] [--mode MODE]
       clusters outfile

Prep Pacybara libraries for use in barseq

positional arguments:
  clusters    translated clusters csv.gz file
  outfile     output file name

flags:
  -h, --help  show this help message and exit

optional arguments:
  -x, --opts  RDS file containing argument values
  -m, --mode  translation mode. Valid arguments: up2up, down2down,
              virt2up, virt2down. 'up2up' directly writes uptag-based
              clusters to an uptag library. 'down2down' directly writes
              downtag-based clusters to a downtag library. 'virt2up'
              converts virtual-barcode-based clusters into an uptag
              library. 'virt2down' converts virtual-barcode-based
              clusters into a downtag library. [default: up2up]
```

Manual conversion

Alternatively, you can also do this manually. To do so, extract the `clusters_transl_softfilter.csv.gz` file using gunzip (or similar software) and open it in a spreadsheet software such as `Calc` or `Excel`. Then, assuming you want to use uptags:

1. Delete the following columns: `virtualBarcode`, `reads`, `collisions`, `upTagCollisions`, and `geno`.
2. Rename the `upBarcode` column to `barcode`.
3. Move the `size` column to the end (so that is now column `I`)

You should now have the following 9 columns, in the following order: `barcode`, `hgvsc`, `hgvsp`, `codonChanges`, `codonHGVS`, `aaChanges`, `aaChangeHGVS`, `offTarget`, `size`. Save it as a CSV file, for example `myBarcodeLibrary.csv`.

Running barseqPro

1. Prepare a parameter sheet for your run. An example can be found below.
2. Prepare a folder containing the FASTQ files for your samples. Make sure the names of the FASTQ files match the sample names in the parameter sheet.
3. The results will be written to the current working directory, so I recommend creating a new directory for this purpose first; e.g. `mkdir barseq_MFG_2022-08-15 && cd $_`.
4. BarseqPro just takes two arguments, the path to the FASTQ folder and the parameter sheet, e.g. `barseq.sh fastqFolder/ parameters.txt`. Again, it is recommended to submit this as a HPC job: `submitjob.sh -n myBarseqJob -c 12 -m 24G -t 36:00:00 -l barseq.log -e barseq.log -- barseq.sh fastqFolder/ parameters.txt`.

Runtime is very dependent on the maxError parameter in the parameter sheet. Allowing for only 1 error is much faster than 2 or 3!

BarseqPro Output:

BarseqPro will create the following folders: `counts/`, `logs/`, and `scores/` If some of these are missing or empty, you can check the log files for errors. The most important results will be in the `scores/` folder:

- `joint_scores_*.csv` will contain the map scores for each amino acid change in a given assay in MaveDB format.
- `all_aa_scores_*.csv` is similar, but contains a more detailed breakdown of scores based on single mutants only, multi-mutant averaging, and inverse multiplicative model inference.
- `allScores.csv` contains fitness scores for each individual barcoded clone.

Running barseq QC

You can run `barseq_qc.R` on the output of barseqPro. It takes the following arguments: The path to the `allLRs.csv` file, the path to the `allCounts.csv` file, and the desired output directory, e.g.: `barseq_qc.R scores/allLRs.csv counts/allCounts.csv qc/`

It will generate a number of QC plots in the specified output folder.

Example parameter sheets

Pacybara parameter sheet

A valid Pacybara parameter sheet is a simple plain-text .TXT file with two sections: `ARGUMENTS` and `AMPLICON SEQUENCE`. Each of them are demarcated using `#BEGIN` and `#END` statements.

The `ARGUMENTS` section is a bash script that needs to define the following variables: `TITLE`, `INFASTQ`, `WORKSPACE`, `BARCODE`, `ORFSTART`, `ORFEND`, `MAXQDROPS`, `MINBCQ`, `MINJACCARD`, `MINMATCHES`, `MAXDIFF`, `MINQUAL` and `CLUSTERMODE`. See the example below for individual explanations. Importantly, the `ORFSTART` and `ORFEND` parameters should correspond to the start and end positions of the open reading frame in the provided amplicon sequence.

The `AMPLICON SEQUENCE` section should follow FASTA format and contain the sequence of the amplicon including all barcode loci and the coding sequence.

```
#########################
#Pacybara parameter file
#########################

#BEGIN ARGUMENTS
#Experiment title
TITLE="LDLR-R01-test"
# Long read sequencing input file (fastq.gz)
INFASTQ=$HOME/data/pacbio/m54204U_210514_191645_ccs.fastq.gz
# Workspace directory
WORKSPACE=$HOME/projects/pacybara_workspace/LDLR_R01_test/
# The barcode degeneracy sequence used in the amplicon below
BARCODE=SWSWSWSWSWSWSWSWSWSWS
# The start position of the ORF in the amplicon below (1-based)
ORFSTART=207
# The end position of the ORF in the amplicon below (1-based)
ORFEND=2789
# The maximum number of low-quality bases allowed in a given barcode.
MAXQDROPS=5
# The minimum average quality score allowed in a given barcode.
MINBCQ=52
#The minimum Jaccard coefficient (relative overlap of variants) for a cluster merge
MINJACCARD=0.2
#The minimum number of variants two cluster need to have in common to merge
MINMATCHES=1
#The maximum number of errors allowed between two barcode reads
MAXDIFF=2
#The minimum Q-score for a variant basecall to be considered real based on a single read alone
MINQUAL=67
#Cluster based on which barcodes? uptag, downtag, or virtual
CLUSTERMODE=uptag
#END ARGUMENTS

#BEGIN AMPLICON SEQUENCE
>PacBio_pDONR_LDLR_Barcoded  (3013 bp)
GTAAAACGACGGCCAGTCTTAAGCTCGGGCCCCAAATAATGATTTTATTTTGACTGATAGTGACCTGTTCGTTGCAACAA
ATTGATGAGCAATGCTTTTTTTATAATGCCAACTTTgtacaaaaaaGCAGGCTCCATACGAGCACATTACGGGSWSWSWSW
SWSWSWSWSWSWSWSCTAACTCGCATACCTCTGATAACTGCACCATGGGGCCCTGGGGCTGGAAATTGCGCTGGACCG
TCGCCTTGCTCCTCGCCGCGGCGGGGACTGCAGTGGGCGACAGATGTGAAAGAAACGAGTTCCAGTGCCAAGACGGGAAA
TGCATCTCCTACAAGTGGGTCTGCGATGGCAGCGCTGAGTGCCAGGATGGCTCTGATGAGTCCCAGGAGACGTGCTTGTC
TGTCACCTGCAAATCCGGGGACTTCAGCTGTGGGGGCCGTGTCAACCGCTGCATTCCTCAGTTCTGGAGGTGCGATGGCC
AAGTGGACTGCGACAACGGCTCAGACGAGCAAGGCTGTCCCCCCAAGACGTGCTCCCAGGACGAGTTTCGCTGCCACGAT
GGGAAGTGCATCTCTCGGCAGTTCGTCTGTGACTCAGACCGGGACTGCTTGGACGGCTCAGACGAGGCCTCCTGCCCGGT
GCTCACCTGTGGTCCCGCCAGCTTCCAGTGCAACAGCTCCACCTGCATCCCCCAGCTGTGGGCCTGCGACAACGACCCCG
ACTGCGAAGATGGCTCGGATGAGTGGCCGCAGCGCTGTAGGGGTCTTTACGTGTTCCAAGGGGACAGTAGCCCCTGCTCG
GCCTTCGAGTTCCACTGCCTAAGTGGCGAGTGCATCCACTCCAGCTGGCGCTGTGATGGTGGCCCCGACTGCAAGGACAA
ATCTGACGAGGAAAACTGCGCTGTGGCCACCTGTCGCCCTGACGAATTCCAGTGCTCTGATGGAAACTGCATCCATGGCA
GCCGGCAGTGTGACCGGGAATATGACTGCAAGGACATGAGCGATGAAGTTGGCTGCGTTAATGTGACACTCTGCGAGGGA
CCCAACAAGTTCAAGTGTCACAGCGGCGAATGCATCACCCTGGACAAAGTCTGCAACATGGCTAGAGACTGCCGGGACTG
GTCAGATGAACCCATCAAAGAGTGCGGGACCAACGAATGCTTGGACAACAACGGCGGCTGTTCCCACGTCTGCAATGACC
TTAAGATCGGCTACGAGTGCCTGTGCCCCGACGGCTTCCAGCTGGTGGCCCAGCGAAGATGCGAAGATATCGATGAGTGT
CAGGATCCCGACACCTGCAGCCAGCTCTGCGTGAACCTGGAGGGTGGCTACAAGTGCCAGTGTGAGGAAGGCTTCCAGCT
GGACCCCCACACGAAGGCCTGCAAGGCTGTGGGCTCCATCGCCTACCTCTTCTTCACCAACCGGCACGAGGTCAGGAAGA
TGACGCTGGACCGGAGCGAGTACACCAGCCTCATCCCCAACCTGAGGAACGTGGTCGCTCTGGACACGGAGGTGGCCAGC
AATAGAATCTACTGGTCTGACCTGTCCCAGAGAATGATCTGCAGCACCCAGCTTGACAGAGCCCACGGCGTCTCTTCCTA
TGACACCGTCATCAGCAGGGACATCCAGGCCCCCGACGGGCTGGCTGTGGACTGGATCCACAGCAACATCTACTGGACCG
```

```
ACTCTGTCCTGGGCACTGTCTCTGTTGCGGATACCAAGGGCGTGAAGAGGAAAACGTTATTCAGGGAGAACGGCTCCAAG
CCAAGGGCCATCGTGGTGGATCCTGTTCATGGCTTCATGTACTGGACTGACTGGGGAACTCCCGCCAAGATCAAGAAAGG
GGGCCTGAATGGTGTGGACATCTACTCGCTGGTGACTGAAAACATTCAGTGGCCCAATGGCATCACCCTAGATCTCCTCA
GTGGCCGCCTCTACTGGGTTGACTCCAAACTTCACTCCATCTCAAGCATCGATGTCAATGGGGGCAACCGGAAGACCATC
TTGGAGGATGAAAAGAGGCTGGCCCACCCCTTCTCCTTGGCCGTCTTTGAGGACAAAGTATTTTGGACAGATATCATCAA
CGAAGCCATTTTCAGTGCCAACCGCCTCACAGGTTCCGATGTCAACTTGTTGGCTGAAAACCTACTGTCCCCAGAGGATA
TGGTCCTCTTCCACAACCTCACCCAGCCAAGAGGAGTGAACTGGTGTGAGAGGACCACCCTGAGCAATGGCGGCTGCCAG
TATCTGTGCCTCCCTGCCCCGCAGATCAACCCCCACTCGCCCAAGTTTACCTGCGCCTGCCCGGACGGCATGCTGCTGGC
CAGGGACATGAGGAGCTGCCTCACAGAGGCTGAGGCTGCAGTGGCCACCCAGGAGACATCCACCGTCAGGCTAAAGGTCA
GCTCCACAGCCGTAAGGACACAGCACACAACCACCCGGCCTGTTCCCGACACCTCCCGGCTGCCTGGGGCCACCCCTGGG
CTCACCACGGTGGAGATAGTGACAATGTCTCACCAAGCTCTGGGCGACGTTGCTGGCAGAGGAAATGAGAAGAAGCCCAG
TAGCGTGAGGGCTCTGTCCATTGTCCTCCCCATCGTGCTCCTCGTCTTCCTTTGCCTGGGGGTCTTCCTTCTATGGAAGA
ACTGGCGGCTTAAGAACATCAACAGCATCAACTTTGACAACCCCGTCTATCAGAAGACCACAGAGGATGAGGTCCACATT
TGCCACAACCAGGACGGCTACAGCTACCCCTCGAGACAGATGGTCAGTCTGGAGGATGACGTGGCGTGACTTCGATAGGT
GCGTGTGAAGGSWSWSWSWSWSWSWSWSWSWSCCTCAGTCGCTCAGTCAAGCACCCAGCtttCTTGTACAAAGTTG
GCATTATAAGAAAGCATTGCTTATCAATTTGTTGCAACGAACAGGTCACTATCAGTCAAAATAAAATCATTATTTGCCAT
CCAGCTGATATCCCCTATAGTGAGTCGTATTACATGGTCATAGCTGTTTCCTG
#END AMPLICON SEQUENCE
```

BarseqPro

A valid barseqPro parameter sheet is a simple plain-text .TXT file with four sections: `ARGUMENTS`, `FLANKING SEQUENCES`, `CODING SEQUENCE`, and `SAMPLE TABLE`. Each of them are demarkated using `#BEGIN` and `#END` statements.

The `ARGUMENTS` section is a bash script that needs to define the following variables: `TITLE`, `LIBRARY`, `BCLEN`, `BCMAXERR`, `REVCOMP`, and `PAIREDEND`. See the example below for individual explanations.

The `FLANKING SEQUENCES` section should follow FASTA format and contain two entries: `upstream` and `downstream`. They denote the upstream and downstream sequences of the barcode.

The `CODING SEQUENCE` section should also follow FASTA format and contain the subject coding sequence / open reading frame.

Finally, the `SAMPLE TABLE` should contain a tab-separated table with the following columns: `sample`, `assay`, `condition`, and `replicate`. Here the `sample` should correspond to the label of the corresponding FASTQ file.

```
#########################
#BarseqPro parameter file
#########################

#FASTQ location: $HOME/projects/barseqPro/LDLR_R5_downtag_fastq

#BEGIN ARGUMENTS
#Experiment title
TITLE="LDLR-pacybara-downtag-R05"
#Barcode library CSV table (from Pacbio pipeline)
LIBRARY="$HOME/projects/barseqPro/libraries/LDLR_R05_downtag.csv"
#Barcode length:
BCLEN=25
#Maximum number of errors (mismatches) allowed in barcode
BCMAXERR=1
#Whether reads are in reverse complement relative to
# the barcodes defined in the library table. 1=yes, 0=no
REVCOMP=0
#Whether the sequencing run uses paired-end mode. 1=yes, 0=no
PAIREDEND=0
#Frequency filter cutoff. The minimum relative frequency required in the nonselect (i.e. "all") condition.
FREQFILTER=5e-7
#Bottleneck filter cutoff. The minimum read count required in any select condition. 0 = off
```

```
BNFILTER=0
#END ARGUMENTS

#BEGIN FLANKING SEQUENCES
>upstream
TGTGAAGG
>downstream
CCTCAGTC
#END FLANKING SEQUENCES

#BEGIN CODING SEQUENCE
>CDS
ATGGGGCCCTGGGGCTGGAAATTGCGCTGGACCGTCGCCTTGCTCCTCGCCGCGGCGGGGACTGCAGTG
GGCGACAGATGTGAAAGAAACGAGTTCCAGTGCCAAGACGGGAAATGCATCTCCTACAAGTGGGTCTGC
GATGGCAGCGCTGAGTGCCAGGATGGCTCTGATGAGTCCCAGGAGACGTGCTTGTCTGTCACCTGCAAA
TCCGGGGACTTCAGCTGTGGGGGCCGTGTCAACCGCTGCATTCCTCAGTTCTGGAGGTGCGATGGCCAA
GTGGACTGCGACAACGGCTCAGACGAGCAAGGCTGTCCCCCCAAGACGTGCTCCCAGGACGAGTTTCGC
TGCCACGATGGGAAGTGCATCTCTCGGCAGTTCGTCTGTGACTCAGACCGGGACTGCTTGGACGGCTCA
GACGAGGCCTCCTGCCCGGTGCTCACCTGTGGTCCCGCCAGCTTCCAGTGCAACAGCTCCACCTGCATC
CCCCAGCTGTGGGCCTGCGACAACGACCCCGACTGCGAAGATGGCTCGGATGAGTGGCCGCAGCGCTGT
AGGGGTCTTTACGTGTTCCAAGGGGACAGTAGCCCCTGCTCGGCCTTCGAGTTCCACTGCCTAAGTGGC
GAGTGCATCCACTCCAGCTGGCGCTGTGATGGTGGCCCCGACTGCAAGGACAAATCTGACGAGGAAAAC
TGCGCTGTGGCCACCTGTCGCCCTGACGAATTCCAGTGCTCTGATGGAAACTGCATCCATGGCAGCCGG
CAGTGTGACCGGGAATATGACTGCAAGGACATGAGCGATGAAGTTGGCTGCGTTAATGTGACACTCTGC
GAGGGACCCAACAAGTTCAAGTGTCACAGCGGCGAATGCATCACCCTGGACAAAGTCTGCAACATGGCT
AGAGACTGCCGGGACTGGTCAGATGAACCCATCAAAGAGTGCGGGACCAACGAATGCTTGGACAACAAC
GGCGGCTGTTCCCACGTCTGCAATGACCTTAAGATCGGCTACGAGTGCCTGTGCCCCGACGGCTTCCAG
CTGGTGGCCCAGCGAAGATGCGAAGATATCGATGAGTGTCAGGATCCCGACACCTGCAGCCAGCTCTGC
GTGAACCTGGAGGGTGGCTACAAGTGCCAGTGTGAGGAAGGCTTCCAGCTGGACCCCCACACGAAGGCC
TGCAAGGCTGTGGGCTCCATCGCCTACCTCTTCTTCACCAACCGGCACGAGGTCAGGAAGATGACGCTG
GACCGGAGCGAGTACACCAGCCTCATCCCCAACCTGAGGAACGTGGTCGCTCTGGACACGGAGGTGGCC
AGCAATAGAATCTACTGGTCTGACCTGTCCCAGAGAATGATCTGCAGCACCCAGCTTGACAGAGCCCAC
GGCGTCTCTTCCTATGACACCGTCATCAGCAGGGACATCCAGGCCCCCGACGGGCTGGCTGTGGACTGG
ATCCACAGCAACATCTACTGGACCGACTCTGTCCTGGGCACTGTCTCTGTTGCGGATACCAAGGGCGTG
AAGAGGAAAACGTTATTCAGGGAGAACGGCTCCAAGCCAAGGGCCATCGTGGTGGATCCTGTTCATGGC
TTCATGTACTGGACTGACTGGGGAACTCCCGCCAAGATCAAGAAAGGGGGCCTGAATGGTGTGGACATC
TACTCGCTGGTGACTGAAAACATTCAGTGGCCCAATGGCATCACCCTAGATCTCCTCAGTGGCCGCCTC
TACTGGGTTGACTCCAAACTTCACTCCATCTCAAGCATCGATGTCAACGGGGGCAACCGGAAGACCATC
TTGGAGGATGAAAAGAGGCTGGCCCACCCCTTCTCCTTGGCCGTCTTTGAGGACAAAGTATTTTGGACA
GATATCATCAACGAAGCCATTTTCAGTGCCAACCGCCTCACAGGTTCCGATGTCAACTTGTTGGCTGAA
AACCTACTGTCCCCAGAGGATATGGTCCTCTTCCACAACCTCACCCAGCCAAGAGGAGTGAACTGGTGT
GAGAGGACCACCCTGAGCAATGGCGGCTGCCAGTATCTGTGCCTCCCTGCCCCGCAGATCAACCCCCAC
TCGCCCAAGTTTACCTGCGCCTGCCCGGACGGCATGCTGCTGGCCAGGGACATGAGGAGCTGCCTCACA
GAGGCTGAGGCTGCAGTGGCCACCCAGGAGACATCCACCGTCAGGCTAAAGGTCAGCTCCACAGCCGTA
AGGACACAGCACACAACCACCCGGCCTGTTCCCGACACCTCCCGGCTGCCTGGGGCCACCCCTGGGCTC
ACCACGGTGGAGATAGTGACAATGTCTCACCAAGCTCTGGGCGACGTTGCTGGCAGAGGAAATGAGAAG
AAGCCCAGTAGCGTGAGGGCTCTGTCCATTGTCCTCCCCATCGTGCTCCTCGTCTTCCTTTGCCTGGGG
GTCTTCCTTCTATGGAAGAACTGGCGGCTTAAGAACATCAACAGCATCAACTTTGACAACCCCGTCTAT
CAGAAGACCACAGAGGATGAGGTCCACATTTGCCACAACCAGGACGGCTACAGCTACCCCTCGAGACAG
ATGGTCAGTCTGGAGGATGACGTGGCGTGA
#END CODING SEQUENCE

#BEGIN SAMPLE TABLE
sample  assay condition replicate
LDLR_Reg5_Rep1_All  Uptake  All 1
LDLR_Reg5_Rep2_All  Uptake  All 2
```

```
LDLR_Reg5_Rep3_All  Uptake  All 3
LDLR_Reg5_Rep4_All  Uptake  All 4
LDLR_Reg5_Rep1_F5 Uptake  F5  1
LDLR_Reg5_Rep2_F5 Uptake  F5  2
LDLR_Reg5_Rep3_F5 Uptake  F5  3
LDLR_Reg5_Rep4_F5 Uptake  F5  4
#END SAMPLE TABLE
```