**Overview of the Consensus Protein Pileups (CPP) Tool**

**Objective:** to reconstruct the one most likely protein sequence for a gene or exon from Illumina short read DNA data. Given the facts that:

1. NGS data is noisy
2. Parasite infections are often poly-clonal mixed populations
3. Isolates may have more than one copy of the gene of interest
4. “Align to Reference” strategies like Bowtie2 are good starts, but not perfect
5. “De Novo Assembly” strategies like Velvet are not perfect, and do very poorly on highly similar gene families like parasite PfEMP1 exported proteins

We can not expect to see exactly one perfect protein sequence. The best we can hope to capture is the consensus, i.e. the amino acids calls most often observed at every position along the protein, with the assumption that the true protein will be what we see the most copies of.

**Motivation:** wanted to get a better sense of the true sequence variation in VAR2CSA in field collected parasite infections. Read pileups from genomic DNA and RNA-seq strongly suggested that local sequence deviation from reference 3D7 is a main reason why pileups were so spotty, as if no reads land in many areas of the gene. Working hypothesis was that the “missing reads” – those that were truly part of VAR2CSA but failed to align well enough to reference 3D7 – should end up on our “No Hits” bin of unaligned reads. These “no-Hit” reads could help reveal the true sequence of VAR2CSA for that parasite isolate.

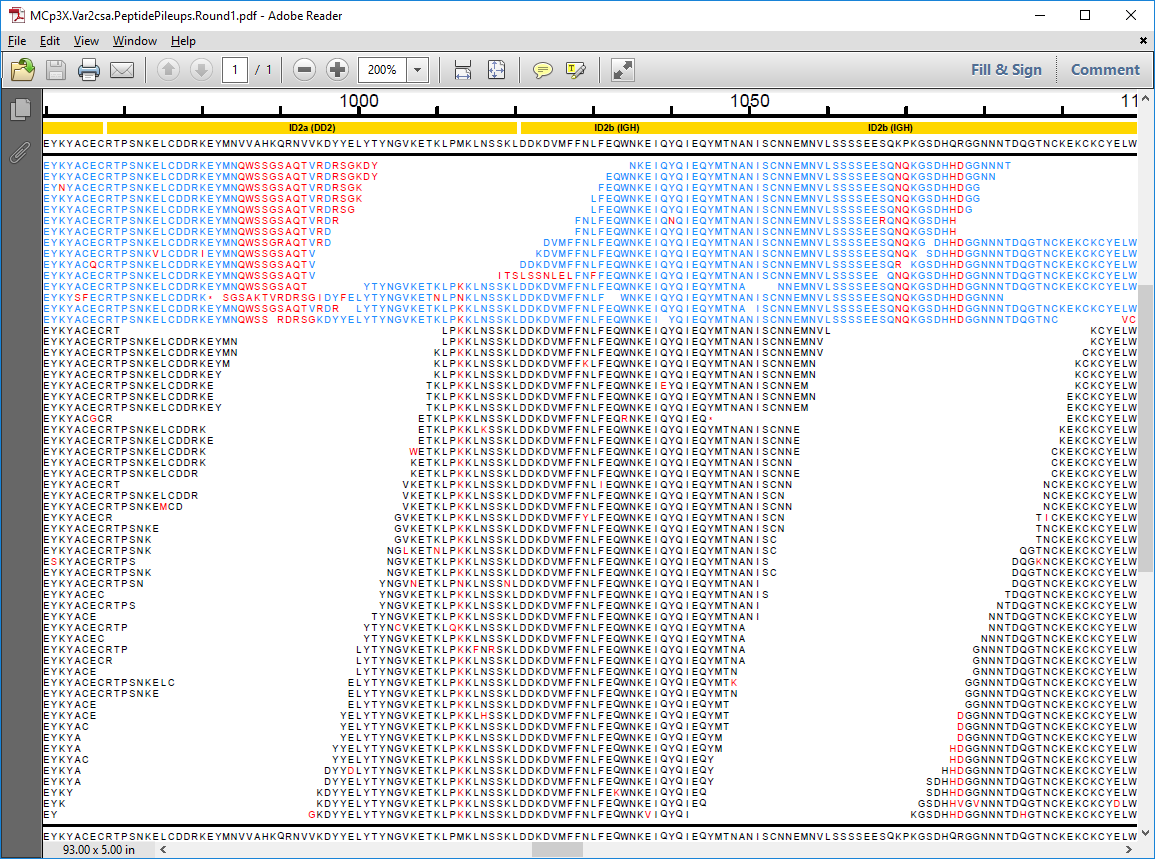
**Strategy:**  converting all relevant data from DNA reads to AA “peptide fragments” would make the data easier to manage and visualize, and should give better sequence alignment/comparison specificity. Thus step 1 is to translate all reads that do align to the gene of interest in all 6 reading frames, keeping only those without excessive internal stop codons. Next, do the same for all reads in the “No Hits” bin, translated in all 6 reading frames, to create a large pool of protein fragments that, by definition, seem to exist in this isolate but failed to align to reference 3D7. To keep this pool of a manageable size, we remove fragments having very low AA sequence complexity. All truly novel variant protein fragments should reside in this pool.

**Usage:** the Consensus Protein Pileups tool is part of the DuffyNGS R package of Next Gen Sequencing tools, consisting of a family of specific functions.

**The Consensus & Plot function** is invoked with command:

pipe.ConsensusProteinPileups( sampleID, geneID, geneName = geneID, optionsFile = "Options.txt", results.path = NULL, exon = NULL, maxNoHits.pileup = 1e+06, max.depth = 60, txt.cex = 0.25, forceSetup = FALSE, maxNoHits.setup = 1e+06, mode = c("normal", "realigned"), plotOnly = FALSE, max.drawnPerSite = 3, trim5.aligns = 0, trim3.aligns = 0, trim5.nohits = 0, trim3.nohits = 0, draw.box = FALSE, chunkSize.pileup = 50000, useCutadapt = FALSE)

Generates a family of text files and PDF images that represent the current state of the protein and DNA consensus for the gene. Next, successive rounds of visually inspecting the PDF pileups and modifying the construct’s current state are performed to iteratively converge the read data and the sequence until they agree as best as possible. An example image below shows a region of ID2a-ID2b in parasite Malayan Camp, with Bowtie2 aligned reads in black, unaligned No-Hit reads in blue, showing amino acids that contradict the current consensus highlighted in red.



The two regions that accumulated zero Bowtie2 aligned reads do in fact accumulate several No-Hit reads that reveal the most likely true amino acid sequence in those areas.

**The Inspection function**:

inspectConsensus ( sampleID, geneName = "Var2csa", context = NULL, extra.rows = 3, readingFrame = c("BestFrame", "Frame1", "Frame2", "Frame3"), optionsFile = "Options.txt", results.path = NULL)

gives the precise DNA locations where discrepancies are present, as a data frame of nucleotide pileup details. The row numbers of these data frames are needed to specify the “location” for any subsequent modification operations. The inspection & modification functions operate in both DNA and AA worlds simultaneously, and the inspection results reveal all details about SNPs, indels, and the current possible coding sequence in all 3 reading frames.

**The modification function** has several modes, to maximize the flexibility of making sequence modifications:

modifyConsensus( sampleID, geneName = "Var2csa", command = c("delete", "insert", "replace", "frameshift", "append"), location = NULL, seq, readingFrame = c("BestFrame", "Frame1", "Frame2", "Frame3"), optionsFile = "Options.txt", results.path = NULL)

The modification operations alter the consensus AA and DNA sequences, by performing that one “command” at the specified location or range of locations, using the given “seq” argument as the new AA sequence to insert/replace/append at that location. In a typical session, one can step along the PDF from 5’ to 3’, to repeatedly make the most obvious modifications needed. After getting to the 3’ end, we can then proceed to the “Realignment” step that will rerun Bowtie2 on the new updated sequence. Note that most modifications cannot be completely resolved in a single pass, as the current state of the pileup image can only suggest at most one half of a read length of alternate sequence at a time.

To run the **realignment step**, use function:

realignConsensus( sampleID, geneID = "PF3D7\_1200600", geneName = "Var2csa", readingFrame = c("BestFrame", "Frame1", "Frame2", "Frame3"), optionsFile = "Options.txt", results.path = NULL, extra.fastq.keyword = NULL, useCutadapt = FALSE)

Followed by the **Accept** function which validates and certifies the realignment as the new official consensus:

acceptRealignedConsensus( sampleID, geneID = "PF3D7\_1200600", geneName = "Var2csa", optionsFile = "Options.txt", results.path = NULL)

At this point we can repeat the Consensus & Plot function again, to get updated PDF files and FASTA sequences, thus completing one cycle around the CPP iterative flowchart process. If the raw data is of good quality, and the parasite is close to clonal, then after some number of CPP cycles the consensus sequence will agree completely with the raw read data and the process is complete. Parasites similar to the 3D7 reference need just a few cycles of CPP, while parasites very different from the reference may need several dozen iterations to converge.

Once done, there is one **Audit** function that gives a terse summary of the history of CPP modification steps that were run:

CPP.AuditSummary( sampleID, geneName, results.path = getOptionValue("Options.txt", "results.path", verbose = F)

**Results:** Example #1: running CPP on lab strain IT4/FCR3 shows how many iterations of CPP a typical parasite isolate needs. The audit summary shows it took 14 times around the CPP flowchart loop for full convergence. The vast majority of modifications was in the DBL6 region, consistent with that being the most variant and least conserved DBL domain in Var2csa.

> CPP.AuditSummary( "IT4\_FCR3", "Var2csa")

Audit Summary: IT4\_FCR3

Counts by CPP Command Type:

Modify Pileup Realign

49 15 14

Counts by CPP 'Modify' Sub-Command Type:

append delete replace

33 14 2

Locations of Modify Operations By Var2csa Domain:

DomainID N\_Modify\_Ops Pct\_Modify\_Ops

1 NTS 0 0.0

2 DBL1 2 4.1

3 ID1 0 0.0

4 DBL2 7 14.3

5 ID2a 0 0.0

6 ID2b 2 4.1

7 DBL3 3 6.1

8 DBL4 4 8.2

9 DBL5 2 4.1

10 DBL6 29 59.2

Example #2: running CPP on a very novel field isolate that contains a 7th DBL domain takes many additional iterations of CPP, mostly because the reference 3D7 has no similar domain, and so the starting consensus has no DBL7 at all, and it will only become apparent after many iterations of extending DBL6 and never encountering the transmembrane domain or ATS. Below is the audit summary for novel maternal Malian isolate M0200101:

> CPP.AuditSummary( "S\_200101\_W4\_CD36", "Var2csa")

Audit Summary: S\_200101\_W4\_CD36

Counts by CPP Command Type:

Modify Pileup Realign

109 62 61

Counts by CPP 'Modify' Sub-Command Type:

append delete replace

84 22 3

Locations of Modify Operations By Var2csa Domain:

DomainID N\_Modify\_Ops Pct\_Modify\_Ops

1 NTS 0 0.0

2 DBL1 5 4.6

3 ID1 4 3.7

4 DBL2 11 10.1

5 ID2a 3 2.8

6 ID2b 4 3.7

7 DBL3 2 1.8

8 DBL4 6 5.5

9 DBL5 6 5.5

10 DBL6 68 62.4

The audit summary shows it took 61 times around the CPP flowchart loop for full convergence. The vast majority of modifications were in the region the tool calls DBL6 (since the reference does not know about a DBL7, it calls all of the novel extra domain part of DBL6).

The maximum amount a sequence can get extended in any one operation is a function of the raw read length, and so the total number of cycles depends on raw read length. Typically we can see trustable evidence of the true protein sequence for about 40-50% of a protein fragment length, so a 100bp raw read would make a 33aa fragment, and thus an upper limit of growing the protein of about 10-15aa per CPP iteration per modification site.

**Limitations:**  like any software tool, the method has several limitations. While not an exhaustive list, the following are some of the main points to consider:

1. **Gene sequence uniqueness within its genome:** this limitation cannot be over stressed. Not all genes can be done by CPP. A gene must be unique enough for the Bowtie2 ‘align to reference’ strategy to yield enough aligned reads to create a starting consensus. Further, any gene having regions too similar to other genes in the genome runs a strong risk of having its variant raw reads wrongly aligned to other genes, and thus not present in the No-Hits bin. For these reasons, most PfEMP1 genes could never be evaluated by CPP as they are a family of ~60 highly similar and highly variant proteins. VAR2CSA happens to be different enough from the other PfEMP1 proteins so its raw reads are seldom mis-aligned, and its sequence is conserved enough such that Bowtie2 can give a good starting consensus to work from. Even so, the DBL6 domain is so poorly conserved and has high enough similarity to another 3-DBL epsilon gene (PFF1580c), that about 15% of all field isolates seem to lose many of their true VAR2CSA DBL6 reads because of better alignment during original Bowtie2 processing to the 3 DBL epsilon domains of reference 3D7 gene PFF1580c. Any time a raw read that should actually belong to our gene of interest gets aligned to some other genomic location instead, it becomes unavailable for the CPP tool. There is a potential work-around, described in more detail at the end of this document.
2. **Intron/Exon boundaries are difficult:** because we are working in both DNA and AA at the same time, and our final result is protein, intron boundaries always present noise that can complicate finding the true consensus. If the raw data is genomic DNA, then reads will never present the spliced form of the exon junction, and we see some red ‘disagreeing’ amino acids on both edges of the exon splice. If the raw reads are mRNA, then we tend to see both states – both the fully processed transcribed form with intron removed as well as the unprocessed mRNA with the intron still present – so we still see some amount of red ‘disagreeing’ amino acids in the plot images.
3. **Large repeating sequence regions can never be truly resolved:** any location in a protein containing a long repeating sequence that is longer that the raw read length can never have its true protein sequence known from short read Illumina type data. So any genes that we classically think of as having microsatellites of varying length are impossible to reconstruct by CPP. These include the internal repeat in CSP, MSP1, etc. The tool does the best it can, to fit the raw read to the reference sequence for the protein, so you do get an answer for these types of proteins. But the exact length of that repeat in that field isolate in unknowable.
4. **Raw data type, read depth, and read quality:** low quality raw reads make for low quality protein fragments. Any negative features like partial adaptor ends, “N” uncalled bases, etc., corrupt the protein fragments and render them unusable noise. Very long reads tend to contain small reading frame shifts, that the alignment tool can easily accommodate as gaps, but those same reading frame shifts ruin the protein translation as the one fragment ends up containing two incompatible reading frames. For these reasons, we typical trim the 3’ low quality ends of the raw DNA reads to keep the working read length no more that ~150bp. It does not eliminate the problem, but it does help to minimize it. Next, the choice of raw read data type introduces issues. Both gDNA and mRNA each have their limitations. Messenger RNA is only present for transcribed genes, and thus read depth is dependent on active expression of the gene of interest. If the gene is not highly enough expressed, then the alignment step creates no starting consensus and there will be no needed variant protein fragments for that gene in the No-Hits bin. Not enough read data means no protein answer. In practice, the tool seems to need roughly a minimum of 15-20 read depth all along the gene’s length to have a chance to reconstruct a good complete protein, and more depth is always better. For gDNA, the problem is reversed. Because you get raw reads for the entire genome regardless of active expression, two problems arise. First, large amounts of intergenic reads end up in the No-Hits bin of protein fragments, diluting its role of holding mostly novel true protein sequences. Secondly, the problem of highly similar gene families is magnified with gDNA. Whereas in practice only a single PfEMP1 is actively expressed at any one time, gDNA data gives equal read depth to all ~60 members, swamping the ‘align to reference’ processing step with many copies of highly variant sequence. This tends to make the least conserved regions of the gene of interest – like DBL6 in VAR2CSA – receive a mashup of reads from many different sites in the parasite’s genome, masking the one true sequence under a crush of competing highly similar foreign sequences.

**Overview of CPP Result Files:**

**ConsensusAA.fasta:** this is the primary result, the final protein sequence in FASTA format.

**ConsensusDNA.fasta:** this is the second main result, the final protein cDNA sequence in FASTA format.

**ConsensusBaseCalls.txt:** a giant table of details about every base location of the cDNA, exported as an R data frame. (It can be opened in Excel, but the column names need to be shifted right one cell because of the ‘R rownames’ in column one). It records all base depth details, final consensus DNA base, protein in 3 reading frames, etc. It is the underlying data that all the “Modify” operations act on, not intended as a user-friendly file.

**ConsensusProteinSummary.txt:** a large table that summarizes the amino acid calls at every position along the protein, that reveals the proportion of minor variants, domain types and boundaries, etc. Consider this as supplemental information. One detail this file can help reveal is it can show the percentage of minor alleles detected all along the protein, and thus can be used to estimate the clonality of the sample.

**GoodScorePeptides.txt:** a large table that summarizes all the peptides used to construct the final protein consensus answer. It contains details about peptide frequency, location in the protein, a scoring metric, and the original source the peptide came from (aligned or No-Hit). Consider this as supplemental information. (Note: due to the iterative nature of the CPP method, what was originally a ‘No-Hit’ peptide can eventually get called as being an ‘aligned.read’, as subsequent iterations successfully align it. This is not an error or a bug, just a limitation of how the algorithm categorizes the peptides over time).

**PeptidePileups.pdf:** a large plot image that visually captures many aspects of the current state of the CPP consensus protein. It is the main working file for all the inspection and modification operations. Given the length of the protein, you may need to zoom in quite a bit to see the details of interest.

**DomainDetails.csv:** optional files that describe the locations and closest matching identities of PfEMP1 var gene domains. There may be two versions of this file, one that assumes the protein can contain only ‘Var2csa’ domains, and one that assumes the protein can contain any PfEMP1 domain present in the ‘VSA’ Vardom set from Rask et.al. (PLos Comp.Bio. 2010). If a novel Var2csa protein exhibits extra domains after DBL6, only the VSA domain search is apt to reveal it’s presence and identity.

**Method for Recovering Genomic Mis-assigned Reads:**

If you encounter an unexpected truncation with no apparent reads, the most likely cause is that those reads were assigned to some other genomic region during the main pipeline processing phase, making them not available to the CPP tool. For the DBL6 region of Var2csa, experience shows that the 3 DBL-epsilon region of another PfEMP1 gene (PF3D7\_0632500, PFF1580c) is the most likely region. CPP provides a work-around to make those mis-assigned reads available as follows:

1. Once you identify the genomic region, extract those reads into a separate file of additional reads. Try to narrow the region as best you can. For the VAR2CSA DBL6 region, try:

pipe.GatherRegionAlignments( sampleID, seqids=”Pf3D7\_06\_v3”, starts=1356201, stops=1359194, asFASTQ=T, fastq.keyword=”PFF1580c”)

1. During the realignment step, give this extra set of reads as an optional argument, to increase the pool of reads that could align to the current CPP construct:

realignConsensus( sampleID, geneID = "PF3D7\_1200600", geneName = "Var2csa", extra.fastq.keyword=”PFF1580c”)