PharmaSUG 2014 - Paper HA07

Using the SAS System as a bioinformatics tool: A macro that translates BLASTn results to populate a DNA sequence database table

Kevin Viel; inVentiv Health Clinical and Histonis, Incorporated; Atlanta, GA

ABSTRACT

Using the Basic Local Alignment Search Tool (BLAST) to align DNA sequences to a known reference sequence is a common task in genomics research. The results of the BLASTn (nucleotide) alignment can be translated into a database table of a relational database of a sequencing project. Appropriately annotating the reference sequences increases the utility of the resulting table. The National Center for Biotechnology Information (NCBI) of National Institutes of Health (NIH) provides the standalone blast+ software package, which enables a SAS programmer to easily submit batches of query sequences. The **goals** of this paper are to describe a SAS macro that uses the results of BLASTn alignments and Phred data to populate a table and to discuss some issues encountered in a sequencing project that generated data on over 60 million nucleotides.

INTRODUCTION

DNA and RNA sequencing is undergoing another revolution in medicine and drug discovery. The observations of patient-specific responses to therapies, devices, and, pharmaceuticals are, instead of being novel, the expectation due to the uniqueness of the patient's genome, its expression, the "burden" of his or her microbiome, and influence of his or her environment (sufficiently or insufficiently encompassed by the term gene-environment interaction and, perhaps, by the term epigenetics). Often the patient-specific response to a drug is referred to pharmacogenomics, but, in practice, it may more accurately be termed patient-*group*-specific response because of the role of statistics and the incomplete characterization of the genomes of the patients dictating the need to group "similar" patients, for instance those that have the same genotype at one or more defined locus. Their genotypes at other (functional) loci are unknown and may not be the same, thus introducing noise, error, or bias. Resolution of the whole genomes of the patients will refine this grouping and improve the elucidation of the mechanisms of pharmacogenomics. Investigators and care providers might expect that the FDA "Black Box" warnings for drugs like Coumadin® (warfarin sodium) will become a required element of the package inserts.

The human genome is three billion base-pairs of DNA and the genome of the microbiome (viruses, bacteria, mold, parasites, fungi, et cetera) that either shares (commensal) the volume of the body or infects it is considerably larger, even if some of the organisms, like viruses, have miniscule genomes (20,000 base-pairs). In addition, the epigenetic patterns and expression add what is currently regarded as dizzying vastness to a database that attempts to catalogue the human health experience.

A genomics component to medical record as a standard is likely not far away, but will require a capable system to populate it and to allow important data to be abstracted from it in a timely fashion, for instance, when a paramedic arrives at the scene of an accident. The process to generate the "final" sequence may involve multiple attempts to cover a region, either by design or as a consequence of the sequencing technology. For instance, our recent Sanger sequencing project covered each amplicon a minimum of four times and some amplicons necessarily overlapped, meaning some bases were covered a minimum of eight times. The results of each attempt were used to generate a final consensus sequence. Details from each attempt (sequence file), including the number of failures or non-covered bases, were important to help identify sequencing (laboratory) issues or patients who might have gross DNA deletions or variants in the oligonucleotide to which the primer was intended to bind. The author answered these needs by creating and populating a table in a relational database that could be queried to provide reports, a potential forerunner to a genomics component (table or tables) in medical records.

At PharmaSUG 2011[1-2] and 2012[3], the author presented three SAS macros that create annotated (gene) reference sequences, hereafter abbreviated as **A(G)RS**; that create reference amplicons; and that aligns a test sequence (query) to a known A(G)RS (subject) using the BLASTn (Basic Local Alignment Search Tool, nucleotide)[4] program available in the stand-alone blast+ software package available from the National Center for Biotechnology Information (NCBI) of National Institutes of Health (NIH). **Table 1** presents a portion of the annotated gene reference sequence for *F8*, the structural gene that codes for coagulation Factor VIII.

Table 1. A subset of the *F8* gene reference sequence.

Obs	NN	Base	Location	P	Codonic_ nucleotide	CDS_ nucleotide	Codon	Exonic_ nucleotide	Intronic_ nucleotide
50170	170	T	5' UTR	154250829	.	.	.	170	.
50171	171	C	5' UTR	154250828	.	.	.	171	.
50172	172	**A**	Exon 01	154250827	1	1	**1**	172	.
50173	173	**T**	Exon 01	154250826	2	2	**1**	173	.
50174	174	**G**	Exon 01	154250825	3	3	**1**	174	.
50175	175	C	Exon 01	154250824	1	4	2	175	.
50176	176	A	Exon 01	154250823	2	5	2	176	.
50177	177	A	Exon 01	154250822	3	6	2	177	.

The macro presented in this paper populates a table in a relational genomics database based on the calculated *NN* (nucleotide number), and thus *P* (chromosomal position), of the base to which the query sequence base aligned. The amplicon sequence is known *a priori* and the position (*P*) of a base it contains can be inferred from the name of the amplicon and its relative position in the amplicon. For example, consider the amplicon F8_A092720L0550. F8 refers to the name of a reference sequence file (in this case the name of the gene, *F8*). It is 550 (L0550) nucleotides long and its first base is NN = 92,720 of the *F8* reference sequence (A092720, the capital A, for amplicon, indicates that the number is positive, that is 3' to the transcription start site). The macro determines the NN from the relative position of the alignment to the known reference amplicon, a subset sequence of the reference sequence, warranting a brief description of the reference sequence file (Table 1).

BASE and *P* (position or "point" analogous to the POINT = option to the SET statement, referring to sequential position relative to the first nucleotide of the chromosomal fasta file) are the "raw" variables from the chromosomal file; regardless of the annotation, these variables will be "constant": specifying the value of *P* determines the value of *BASE* in any reference sequence. The values of other variables are derived (annotation) and, for a given value of *P*, may change from (gene) reference file to (gene) reference file, even if the reference sequences are for two or more transcript variants of the same gene. The values of the variables CODONIC_NUCLEOTIDE, CDS_NUCLEOTIDE, CODON, EXONIC_NUCLEOTIDE, and INTRONIC_NUCLEOTIDE may differ between (distinct) A(G)RS's for a set of overlapping values of P. The value of *NN* depends on the start of transcription. The only variables of concern to the direct function of the macro in this paper are *NN*, *BASE*, and *P*.

Note for the F8 reference sequence in Table 1 that as the nucleotide number (*NN*) increases the *P* decreases: *F8* is transcribed telomeric to centromeric. Incidentally, but inconsequentially for this paper, *F8* is also on the "-" strand of the chromosome X reference sequence (an issue explained in the 2011 paper). The original macro[2] presented was not robust in that translation in some genes does not begin in Exon 01. The author has updated the macro to account for this issue and will happily provide a copy upon request.

The macro presented in the 2012 paper runs the stand-alone BLASTn program to create output like that in **Figure 1**. That macro[3] then reads this BLASTn output (the alignment and its details), which was directed to a text file for archival and quality control purposes, to create a SAS dataset. This current paper assumes that the user has access to this dataset, but the macro could be pared down to read the results from random-access memory without saving them to a file, although the user would still need to concatenate the BLASTn results. Specifically, the SAS lines 34-153 of the BLASTn macro in Figure 2 of the 2012 paper[3]. The dataset created is not normalized and waste disk space. This was acceptable for the small scale of the project, but likely infeasible for moderate or large projects.

This paper is geared towards Sanger Sequencing, which uses amplicons, but the concept is extendable to larger DNA regions such as exons (exome-sequencing), the entire gene, or, with some unstated nuances, whole chromosome sequencing (whole-genome sequencing). The quality of the sequencing and the presence of DNA variants affect the alignment. **Table 2** presents the types of DNA sequence variation detectable in typical sequencing projects.

Typically, one might not expect a full alignment to the reference sequence (amplicon) and the primers and amplicons are designed to account for this so that the bases of interest are far enough from the primers to have expected high quality coverage. Nonetheless, this macro creates an observation (record in database table) for each nucleotide in the reference sequence, even if the aligned sequence excludes certain nucleotides. For instance, for each attempt to (re-)sequence the amplicon F8_A092720L0550 results in at least 550 observations (records) in the table (SAS dataset). This is true even for PCR or sequencing reaction failures or (gross) DNA deletions. The values for most variables will be missing when alignment fails. In the event that a full alignment occurs and the query sequence has insertions, then the number of observations will *exceed* the length of the subject reference amplicon. In the case of deletions without insertions, an observation for each deleted nucleotide is still retained.

```
BLASTN 2.2.26+

Query= 0001_F_1_0092_0121_A05_a000148L0569_A21.fas

Length=899

Subject= F8 FORWARD   F8_a000148L0569

Length=569

 Score =  924 bits (500),  Expect = 0.0
 Identities = 518/526 (98%), Gaps = 4/526 (1%)
 Strand=Plus/Plus

Query  53   TCAGAGGTGAATGGGTTAAGTTTAGCAGCCTCCCTTTTGCTACTTCAGTTCTTCCTGTGG  112
            ||||| ||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  47   TCAGAAGTGAATGGGTTAAGTTTAGCAGCCTCCCTTTTGCTACTTCAGTTCTTCCTGTGG  106

Query  113  CTGCTTCCCACTGATAAAAAGGAAGCAATCCTATCGGTTACTGCTTAGTGCTGAGCACAT  172
            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  107  CTGCTTCCCACTGATAAAAAGGAAGCAATCCTATCGGTTACTGCTTAGTGCTGAGCACAT  166

Query  173  CCAGTGGGTAAAGTTCCTTAAAATGCTCTGCAAAGAAATTGGGACTTTTCATTAAATCAG  232
            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  167  CCAGTGGGTAAAGTTCCTTAAAATGCTCTGCAAAGAAATTGGGACTTTTCATTAAATCAG  226

Query  233  AAATTTTACTTTTTTCCCCTCCTGGGAGCTAAAGATATTTTAGAGAAGAATTAACCTTTT  292
            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  227  AAATTTTACTTTTTTCCCCTCCTGGGAGCTAAAGATATTTTAGAGAAGAATTAACCTTTT  286

Query  293  GCTTCTCCAGTTGAACATTTGTAGCAATAAGTCATGCAAATAGAGCTCTCCACCTGCTTC  352
            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  287  GCTTCTCCAGTTGAACATTTGTAGCAATAAGTCATGCAAATAGAGCTCTCCACCTGCTTC  346

Query  353  TTTCTGTGCCTTTTGCGATTCTGCTTTAGTGCCACCAGGGTGCAGTGC-ACCTGGGTGCA  411
            |||||||||||||||||||||||||||||||||||| |   | | | | ||||||||||||
Sbjct  347  TTTCTGTGCCTTTTGCGATTCTGCTTTAGTGCCACCA-GAAG-A-TACTACCTGGGTGCA  403

Query  412  GTGGAACTGTCATGGGACTATATGCAAAGTGATCTCGGTGAGCTGCCTGTGGACGCAAGG  471
            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  404  GTGGAACTGTCATGGGACTATATGCAAAGTGATCTCGGTGAGCTGCCTGTGGACGCAAGG  463

Query  472  TAAAGGCATGTCCTGTAGGGTCTGATCGGGGCCAGGATTGTGGGGATGTAAGTCTGCTTG  531
            ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  464  TAAAGGCATGTCCTGTAGGGTCTGATCGGGGCCAGGATTGTGGGGATGTAAGTCTGCTTG  523

Query  532  GAGGAAGGTGCAGACATCGGGTTAGGATGGTTGTGATGCTACCTGG  577
            |||||||||||||||||||||||||||||||||||||||||||||
Sbjct  524  GAGGAAGGTGCAGACATCGGGTTAGGATGGTTGTGATGCTACCTGG  569


Lambda     K      H
    1.33    0.621     1.12

Gapped
Lambda     K      H
    1.28    0.460    0.850

Effective search space used: 494059


Matrix: blastn matrix 1 -2
Gap Penalties: Existence: 0, Extension: 2.5
```

FIGURE 1. A BLASTn alignment

Table 2. Types of variants found in DNA.

Type	DNA Sequence
Wild-type (WT)	ATCTGACCGTGATGGCGGCCATTGGCTTGGGCTTCCTCACC |||||||||||||||||||||||||||||||||||| ATCTGACCGTGATGGCGGCCATTGGCTTGGGCTTCCTCACC
SNS	ATCTGACCGTGATGGCGGCC**A**TTGGCTTGGGCTTCCTCACC ||||||||||||||||||| ||||||||||||||||||| ATCTGACCGTGATGGCGGCC**C**TTGGCTTGGGCTTCCTCACC
Insertion	ATCTGACCGTGATGGCGGCCA**--**TTGGCTTGGGCTTCCTCACC |||||||||||||||||||| |||||||||||||||||||| ATCTGACCGTGATGGCGGCCA**CG**TTGGCTTGGGCTTCCTCACC
Deletion	ATCTGACCGTGATGGCGG**CCA**TTGGCTTGGGCTTCCTCACC ||||||||||||||||| |||||||||||||||||||| ATCTGACCGTGATGGCGG**---**TTGGCTTGGGCTTCCTCACC
Inversion	ATCTGACCGTGATGGCGGCCATTGGCTTGGGCTTCCTCACC |||||||||||||||||||||||||||||||||||| ATCTGACCGTGATGGCG**TACCG**TGGCTTGGGCTTCCTCACC

The ***goal*** of this paper is describe the macro and its technical use. The author assumes that the reader has a rudimentary understanding of DNA or RNA sequencing, or alignment and BLAST, and of a sequencing project that might benefit from the suite of programs. Even if this paper introduces the reader to DNA sequencing projects, however, the reader should be able to comprehend the concepts, the SAS code, and its purpose. The paper will also present some interesting cases and describe the database the created in his last project.

OVERVIEW OF THE MACRO

In Sanger sequencing, an amplicon is defined by two primers separated by typically not more than 1,200 base-pairs. The investigator then creates reference amplicons in fasta format (**Figure 2A**) and aligns the (QUERY) sequence (Figure 2B) to it using BLASTn, for instance using the BLASTn macro presented in 2012 or the web-based BLASTn. From the output in Figure 1, the macro uses the name of the amplicon and its orientation (forward or reverse) and the alignment, plus the .phd (**Figure 3A**) and .poly (Figure 3B) output from phred (www.phrap.org) to perform its task. Instead of using the phred data, the user could use the sequence and quality score data directly from the .ab1 file. Phred provides other data, such as the area under the peaks and the position of the peaks, which may or may not be available from the .ab1 file.

The six important items from the output in Figure 1 that macro uses are 1) the name of the amplicon to which the query was aligned (Subject=, highlighted green), 2) the starting position of the subject alignment from the start of the sequence (Sbjct, highlighted red), 3) the starting position of the query alignment from the start of the sequence (Query, highlighted violet), 4) the sequence of the query (highlighted turquoise), 5) the sequence of matching indicators (highlighted grey), and 6) the sequence of the subject (highlighted pink).

The amplicon is this example is F8_a000148L0569. It begins at NN = -148 and extends to NN = 421 (-148 + 569 – 1 = 420, but NN = 0 does not exist: -148 to -1 and 1 to 421 equals 569 base-pairs). The reverse orientation begins at NN = 421 and extends to NN = -148. The alignment begins at base 47 (Figure 2A) of the reference amplicon, which corresponds to the base 53 (Figure 2B) of the query sequence. Whereas the first 46 bases of the reference amplicon are real and distinct, the calling software (phred) attempted to define the first 52 “bases” of the query sequence, but viewing the chromatogram (Figure 2C) confirms that they are noise, artifact, or otherwise not of use. Phred and software used by Applied Biosystems (ABI) may yield different sequences and quality values, explaining the apparent difference in the start location of the alignment. The observant reader might also notice that the phred sequence (Figure 2B) consists of only A’s, C’s, G’s, T’s, and N’s whereas the ABI sequence (Figure 2C) also contains all of the IUPAC symbols, for instance R for A or G (pu**R**ine) and W for A or T (**W**eak).

The calculated value of NN for the first base of the alignment is thus -102 (-148 + 47 – 1 = -102), which maps to P = 154251100 in the reference sequence files. The values NN, P, and Base (nucleotide, in this case T), abstracted from the Query sequence, are recorded. The identity of the nucleotide from the reference amplicon is not recorded, nor is any indicator of a match. Since this is the first base of the alignment, however, it matches the reference amplicon since the BLASTn alignment starts and ends with a match. The .phd and .poly files have an

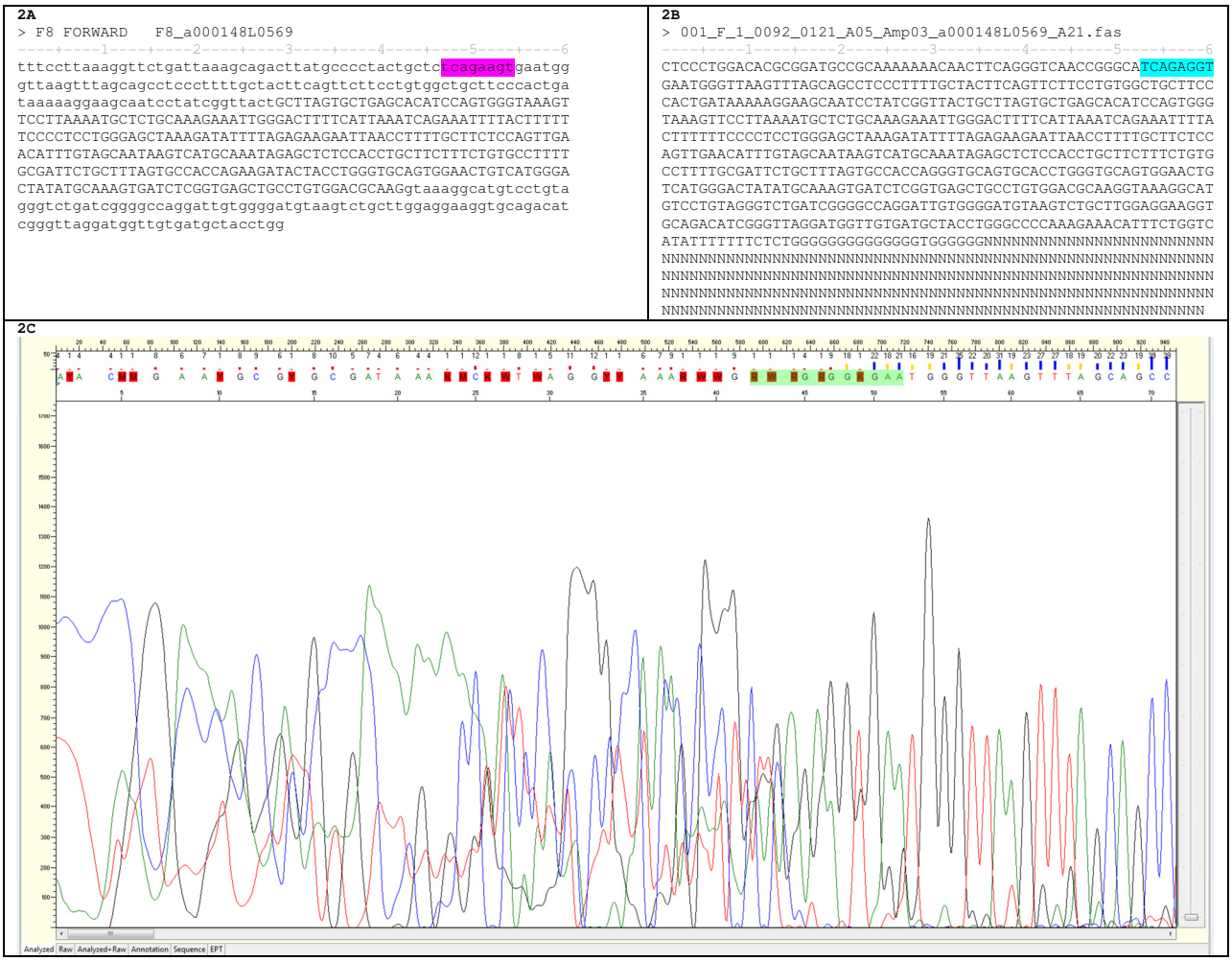

Figure 2. A) The query sequence, B) its reference sequence, and C) its chromatogram. Note that the ruler in the second line of the fasta files in (A) and (B) are not part of the fasta file, but are rather intended as a reference.

observation (record) for every base in the Query sequence since it was the source (in this case 899 “bases”). The macro obtains the respective sequencing data by indexing on the relative position of the base in the Query sequence.

After the data for the first aligned base are populated, the observation is output and the macro moves to the second base of the alignment. The value of NN may be incremented or not, depending on whether it is aligned to a base or a gap. The basis for the incrementation is explained with the code. The values of the variables for the first 52, except for NN and P, are set to missing. If the alignment does not include the last base of reference amplicon, then the values for the uncovered bases are set to missing in the same manner as the bases at the start of the amplicon. This is not the case in this example since the last base aligned in the reference amplicon was 569, the length of the amplicon. Extra bases at the end of an alignment could be the “tailings” of the sequencing reactions, i.e. noise or unimportant products, but high quality sequence could be important and will be addressed in the Discussion.

As a matter of housekeeping, this macro use three text files, but they have been translated to SAS datasets. The corresponding files are identified by their names, with changes to the extension of their filenames indicating the nature of the file (.fas, .phd.1, .poly, .blastn, all of which orginated from the .ab1 file). Phred appends .phd.1 and .poly to the name of the .ab1. The SAS program that wrote the .fas file to be used in the alignment also translated the .ab1 to .fas. The %BLASTn macro names the BLASTn output by translating (the TRANWRD() function) the .fas extension of the query sequence file name to .BLASTn. Nonetheless, for a project, an internal File Sequence Number (FSN) is used to uniquely identify the .ab1 file and the files derived from it. Using the FSN also keeps the user blinded to the patient identity and the region covered when viewing most tables.

3A

```
BEGIN_SEQUENCE 001_F_1_0092_0121_A05_Amp03_a000148L0569_A21.ab1

BEGIN_COMMENT

CHROMAT_FILE: 001_F_1_0092_0121_A05_Amp03_a000148L0569_A21.ab1
ABI_THUMBPRINT: 0
PHRED_VERSION: 0.020425.c
CALL_METHOD: phred
QUALITY_LEVELS: 99
TIME: Mon Aug 22 11:02:55 2011
TRACE_ARRAY_MIN_INDEX: 0
TRACE_ARRAY_MAX_INDEX: 11917
TRIM: 57 596 0.0500
TRACE_PEAK_AREA_RATIO: 0.0064
CHEM: term
DYE: big

END_COMMENT

BEGIN_DNA
c 7 14
t 7 21
c 7 38
c 9 46
c 9 60
t 9 68
g 6 78
```

3B

```
001_F_1_0092_0121_A05_Amp03_a000148L0569_A21.ab1 28839.183673 28966.893424 28839.183673 34673.426423 30725.343228
C  14  317191.836735  1.000000  T  11  176425.390400  1764253.904003  2028.567830  39810.612245  0.000000  21441.734960
T  21  204707.884713  0.645376  C  20  38321.632653  1.208149  2145.600589  38047.346939  0.000000  18352.359769
C  38  159791.020408  0.612344  T  32  17874.242179  0.342484  2301.644268  39771.428571  0.000000  4192.723474
C  46  378553.469388  1.665947  T  52  60684.155547  0.673928  11001.079383  42240.000000  195.543147  5112.180376
C  60  605152.653061  2.283069  A  54  250918.236146  2.786574  18881.285183  41064.489796  12254.037243  9047.455918
T  68  133615.477032  0.401152  A  67  92806.978201  0.616409  15448.324240  28447.346939  21314.203077  15005.536644
```

Figure 3. Portions of (1) the .phd file and (2) the.poly file produced for the .ab1 file by Phred.

EXPLANATION OF SAS CODE

The entirety of the program is the SAS macro BC_BLASTn (**B**ase **C**all from **BLASTn** alignment). The macro statement (lines 1-8) names the macro and eight required macro parameters, three of which have default values assigned. Currently, the macro is designed for "short" sequences that might be produced by Sanger Sequencing. The reason is that additional sequence data other than the position and the identity of the base are abstracted from a second dataset and temporarily stored in an array. The use of arrays, the size of which must be specified at compilation, places a practical limit on the size of the sequence that might be processed, but not unreasonably so given the current state of sequencing technology.

Lines 1-8. The macro parameter **LN** assigns the LIBNAME where the data are both read and written. The macro parameter **Phred** is the name of the dataset holding the Phred output data (*.phd.1 and *.poly). The macro parameter **Obs** is the maximum number of observations in the Phred data. Even though Sanger Sequencing might not produce sequences greater than 1,500 base-pairs, the default value is safely above that maximum. The macro parameter **Seq** is the A(G)RS of the project; from this sequence the reference amplicons should have been created. The macro parameter **BC_Base** is the name of the SAS dataset that contains the results, which may not exist prior to running this macro. The macro parameter **BLASTn** is the name of the dataset holding the results of the BLASTn alignments. This dataset may have been created using the BLASTn macro detailed in the 2012 PharmaSUG paper, lines 34-153[3]. The macro parameter **QC_Seq** indicates whether base from the reference sequencing file identified in **Seq** should be included in the output for quality control. Typically, if the user is satisfied with a few developmental runs, say a forward and reverse, then this flag can remain set to its default value of N (No).

Lines 11-26. The SQL procedure creates a temporary SAS dataset of the BLASTn data that has not already been processed. The name of the SAS dataset is BLASTn (line 12). All of variables in the input datasets are selected by virtue of the asterisk (line 12). The input dataset is given in line 14: &LN..&BLASTn. This consists of two macro parameters, but the explanation pertains to macro variables, too. A macro variable or parameter begins with the ampersand token. A period ends the name of the macro variable. "&LN." pertains to the value of the macro parameter provided in line 1 (LN =). "&BLASTn." pertains to the name of the dataset provided by the macro parameter in line 7 (BLASTN =). Since referencing a SAS datasets requires a two-level name (libref.SAS-data-set, note when a programmer uses a one-level name to reference a temporary dataset, the form WORK.SAS-data-set is assumed by default) a period separates the libref &LN. and the dataset name &BLASTn., thus the double period "..", the first period ends the macro parameter name and the second period separates the libref and SAS-data-set. The author assumes that SAS datasets have a password (PW) and that the passwords are the same for every dataset in the project. The value is provide by the global macro variable, **PW**, whose value is provided as "&PW.". This is NOT intended as a method of even mild security, but adds a layer of protection against loss of data by, for example,

deletion (the KILL options to the DATASETS procedure). Line 15 is a macro statement (%IF-%THEN/%ELSE Statement) that conditionally processes a portion of a macro. In this case, if the (permanent) SAS dataset &BC_Base. already exists, then FSN's in &BLASTn. that are already in &BC_BLASTn. are excluded from those selected from &BLASTn. The SAS function EXIST() checks for the existence of a SAS dataset and the %SYSFUNC() executes the function, making the value available to the macro logic. If the dataset exists, i.e. EXIST() returns 1, then the macro writes the SAS code, otherwise, lines 18-21 are omitted from the SAS code (will not be compiled, executed, or appear in the log).

Lines 29-34. This SQL procedure creates a dataset of distinct SUBJECT values that appear in the BLASTn dataset. In this case, SUBJECT refers to the identity of the reference sequence (highlighted in green in Figure 1) to which the query (test) sequence was aligned. The sequence of bases and their corresponding values of NN, P, and AMPLICON_POSITION for each SUBJECT will be enumerated in the next data step, hence the desire to have only one occurrence (distinct) for each SUBJECT, although it is very likely the many replicates exist in BLASTn.

Lines 36-107. This data step determines the start position, length, and orientation from the name of the SUBJECT (reference amplicon name combine with orientation) and enumerates the sequence of bases in the proper orientation by referencing the A(G)RS (line 62). At this point, the author assumes that each SUBJECT sequence in BLASTn is from the same A(G)RS; it is probably easier from the perspectives of both programming and housekeeping to call the macro separately from each A(G)RS than to have a combined dataset, which would require that a variable holding the name of A(G)RS be included in the INDEX.

Once the start position, length, and orientation are abstracted from SUBJECT (lines 43-48), the data step retrieves the value of BASE and P corresponding to the NN by using it as the KEY value when reading the SEQ dataset (lines 63 and 92). To repeat, a value of SUBJECT might be "F8 FORWARD F8_a000148L0569". For the forward orientation, the first base of this amplicon corresponds to the NN that has the value of SS_NN (-148, a000148), but for the reverse orientation SS_NN corresponds to the last base of the amplicon, which will have a value of AMPLICON_POSITION equal to SUBJECT_LENGTH (see **FIGURE 4**).

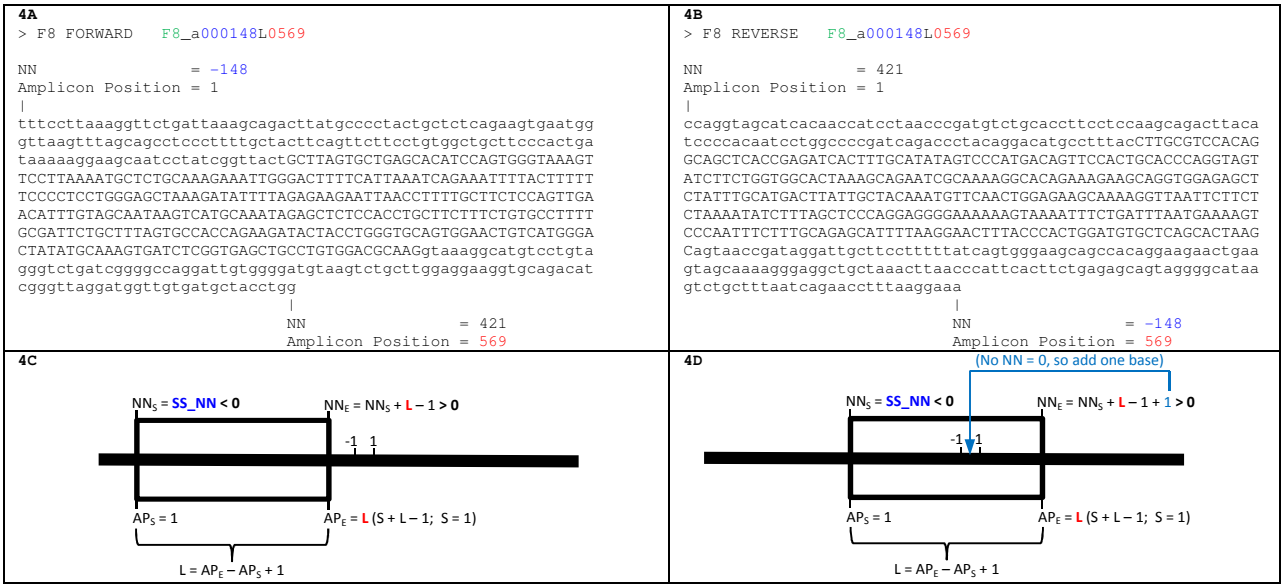

Figure 4. **Amplicon numbering and orientations relative to +1 transcription.**

AMPLICON_POSITION corresponds to the reference amplicon, not the reference sequence from which reference amplicon was derived, and it is orientation-specific; the NN for AMPLICON_POSITION = X for the forward orientation does not equal the NN for AMPLICON_POSITION = X for the reverse orientation (see FIGURE 4A and 4B). For the reverse orientation, the start is actually the last base of the forward sequence, so NN must be calculated from the starting position (the a000148 portion of the amplicon name) captured in SUBJECT is in the forward orientation. Two conditions may exist: SS_NN < 0 or SS_NN > 0. If SS_NN < 0, then the reference amplicon might be entirely in the "promoter" region (FIGURE 4C) or it might extend to or beyond +1 (FIGURE 4D). If the later condition is true, then the calculation of the NN for the last amplicon position must be adjusted for the absence of NN = 0 (line 78 and Figure 4D). In this case, the need to add one base for the starting position, which is included in the sequence is negated by the absence of NN = 0. To clarify, a sequence starting at 1 and extending to 3 (1 2 3) has length 3 = 3 – 1 + 1, not 2 = 3 - 1, but a sequence starting at -1 and extending to 2 (-1 1 2) has length 3 = 2 – (-1) + 1 - 1, not 4 = 2 – (-1) + 1.

Since each loop increments the value of NN by +1 or -1, for forward and reverse, respectively, and increments the value of AMPLICON_POSITION by +1, the initial values are offset to account for this (lines 53-55 for forward and lines 82-84 for reverse) so that at the first loop they are incremented to their original, starting values. The loop iterates until the value of AMPLICON_POSITION equals the value of SUBJECT_LENGTH (lines 57 and 86). In each iteration, NN and AMPLICON_POSITION are incremented as the loop progresses base-by-base (position-by-position) through the reference amplicon. If NN = 0 then NN is set to +1 (forward, line 60) or -1 (reverse, line 89). If the orientation is reverse and the value of the macro parameter QC_PARAMETER equals "Y", then the base must be translated to its compliment (lines 96-102). The forward and reverse sequences are reverse compliments (see FIGURE 3), but since it is a single base the compliment suffices; the "reverse" term is already covered by the value of the increment of NN: +1 and -1 for forward and reverse orientation, respectively.

Lines 109-616. This data step processes the alignment and data in Figure 1 that was stored in as single observation and available in the BLASTn dataset. While processing the alignment position-by-position (note not necessary base-by-base since gaps may exist) the data step will abstract data from the BLASTn_AC_NN_AP and SEQ datasets for the appropriate base, as "indexed" by NN. In the interest of space, only parts of the code will be explained, but the reader could always contact the author for more detailed explanations and examples.

For development runs, the user may wish to test a few sequences and then manually compare the calculated NN and the abstracted data. Setting the macro parameter QC_Seq = Y outputs a second temporary dataset that has the necessary variables (lines 113-115). The novel variables at this point are __QS, __MS, and __SS, which are one character variables holding the current position of the query string, match string, and subject string.

Given the size of these data, the author chose to use hash data step component objects, also known as associative arrays, to abstract the data from the reference amplicon (BLASTn_AC_NN_AP). Distinct from SAS arrays, the key (index) of a hash can be a character and can be compound, and therefore a mix of character and numeric variables. Further, the size of the hash does not need to be defined at compilation. To define the type and attributes of the variables, the program using the SET statement, but it is never executed by virtue of the condition (Line 151).

Two hash objects are created (lines 158-166 and lines 170-178) in a Do block that is only executed when _n_ = 1, i.e. the first iteration of the implicit loop of the data step. The keys and data (values) for both hashes are derived from BLASTn_AC_NN_AP. The hash AP_NN will provide the NN for a given AP and SUBJECT. The hash NN_AP will provide the AP for a given NN and SUBJECT. Note that subject specifies the orientation and name of the reference amplicon, as well as the A(G)RS. Originally, the keys and values were provided using the argument_tag_value DataSet: "BLASTn_AC_NN_AP" (commented lines 157 and 169, which remain only for teaching purposes). As the case report below will show, this resulted in the dataset being read twice for no reason. The Do-Until block (lines 180-184) obviated the multiple reads of the dataset and provides another opportunity to illustrate a method (ADD) for the SAS Component Object, Hash. The variables SUBJECT, AMPLICON_POSITION, and NN are read from BLASTn_AC_NN_AP and "entered" as the respective keys and values to the hashes. Later by supplying the keys, the FIND method will return their values. Hashes are an exquisite addition to the tools of a SAS programmer.

The alignment data are read in line 216. The BY variables in lines 217-218 are the file sequence number (unique index) and HIT. Due to the repetitive nature of DNA, mutation, or error (bleed-through, for instance), multiple alignments of a given sequence within the same amplicon might be possible. The various possible alignments for a given test sequence are indicated by HIT.

Alignment failures. In the event of a sequencing failure or an attempt to align to an incorrect SUBJECT, which may result, for instance, from mishandling the sample or misnaming it, BLASTn reports "No hits found". In that case, the values of all sequencing variables, plus I (INsertion) and D (DELetion) variables, are set to missing (lines 225-243). In addition, if the alignment was in the opposite orientation to what was expect, in which case the value of STRAND will be "Plus/Minus" (line 222), not "Plus/Plus", then the aforementioned values are also set to missing. The decision to "blank" the values instead of abstract their (misaligned) values is a matter of process quality stringency; something that should be investigated. In the case of an alignment with the value of strand being "Plus/Minus", the value of HIT will be greater than zero, which can be used to query the resulting table (observations with "No hits founds" will have HIT = 0). For such alignment failures, the value NN cannot be calculated from the alignment since it does not exist. Instead, the DO-Loop in lines 248-251 increments AMPLICON_POSITION from 1 to SUBJECT_LENGTH and uses its value as a key to the AP_NN hash to obtain the corresponding value NN and P (line 249), which is output in each iteration thus creating an observation for each base of the reference amplicon. For readers that are unfamiliar with hashes, the absence of any arguments or variable in the parentheses of the FIND method in line 249 might be puzzling. The keys are determined by the DEFINEKEY method; one could explicitly provide them: AP_NN.Find(key : SUBJECT , key : AMPLICON_POSITION), but little clarity is gained once the reader understands hashes. The values of the key variables in the program data vector are used and it is incumbent upon the programmer to be sure that the data step supplies them correctly (missing values are viable, but perhaps

trivial). The First.FSN condition (line 245) eliminates multiple sets of the missing data if multiple hits occur in the wrong orientation.

In the event of any alignment in the expected orientation, the phred sequence data must be "loaded" into the array. In order to avoid this loading during each iteration of the data step it is only performed when the FSN changes value, i.e. it does so only if the current value of FSN does not equal the value of FSN from the previous (implicit) data step loop (lines 256-309). If the value of FSN is new, then values of variables in the arrays are set to missing by the CALL MISSING() call routine (lines 259-271). Note, by virtue of using _TEMPORARY_ in the array statements, the data element values are automatically retained. The data are read from the Phred dataset (line 278), which is indexed by FSN. The nonsequential access to observations in the SAS data set will cause END to be equal 0 until no further observations with (index) key equal to FSN are available, which triggers an _ERROR_ = 1 event. At this point, all of the Phred data for that FSN have been read (the Do-Until Loop exits) and the data step resets the values of __OK, _ERROR_, and END (lines 302-304). The value of STOP is assigned the value of POSITION, i.e. the length of the sequence (according to Phred). At this point, the macro is ready to process the alignment data and sets the relevant sequence variables to missing (lines 311-329).

Not uncommonly, alignments do not begin at either the start of SUBJECT or QUERY. Nonetheless, the entire amplicon reference sequence will be represented in the table for at least the first hit (HIT = 1). When the alignment begins "downstream" of the start of the SUBJECT, say at AMPLICON_POSITION = SUBJECT_START, then values of all variables but AMPLICON_POSITION, NN, P, and FSN, for the first SUBJECT_START – 1 bases (observations) must be set to missing (lines 340-343).

At this point, it might is worth emphasizing that AMPLICON_POSITION is the relative position within the reference amplicon, whereas POSITION is relative position within the test sequence, which is, in the absence of any failures, longer due to the elution nature of the Sanger sequencing process. POSTION and AMPLICON_POSITION are initialized to the starting value of their respective alignments (lines 348 and 358, respectively: in FIGURE 1, these are the first values Query 53 and Sbjct 47, respectively). BLASTn *only* begins (and ends) with a match, thus the values or I (INsertion) and D (Deletion) are set to zero (lines 350-351). The data step will proceed position-by-position through the alignment, which may not be the same as base-by-base since gaps (INDELs, whether real or artifactual, for instance due to noise, i.e. low quality sequence) may exists, however a base will be present for either the subject sequence, the query sequence, or both (for a respective base, both the subject and query sequence will not be "-"). The first bases of the alignments (__QS and _SS, for query and subject sequence, respectively) and their match indicator (__MS) are abstracted (lines 353-355). Again using the AMPLICON_POSITION as the key to the hash AP_NN, the values NN and P are obtained (line 359). The value of NN is stored in the temporary variable __NN (line 360**)**. The corresponding Phred sequence data are obtained using POSITION as an index the arrays (lines 365-375). Two metrics of the quality of the current test sequence base are then calculated: one for the upstream flank (FPF, for 5' flank) and one for the downstream flank (TPF, for 3' flank). For clarification, by convention we refer to DNA in the 5' (upstream) to 3' (downstream) orientation. For the flanks, the 10 bases upstream and downstream of the current base, we calculate the number that have a QV (quality value) above 24 (line 387) and the number with noise (the relative area of the uncalled base greater than 0.15, line 388). Lines 384 and 396 assure that the attempt to use the flanks does not extend beyond the start and stop of the sequence, which would cause an error since those indices do not exist.

After this abstraction is completed for the first base of the alignment, which is a guaranteed matched, the process continues for the second to the last base of the alignment. Whereas the last base of the alignment is also guaranteed to be a match for BLASTn, gaps in the QUERY or SUBJECT (but NOT both) can exist and will be indicated by both a hypen ("-") in the respective sequence and a space in the MATCH indicator (a "|" indicates a match, whereas a space indicates a mismatch). The position-by-position increment in AMPLICON_POSITION or NN is predicated on the absences of the hyphen.

In each iteration through the alignment, i.e. from the second position to the size of the query sequence (line 407), the value of the query sequence (__QS), match sequence (__MS) and, subject sequence (__SS) are abstracted (lines 409-411). Note, the lengths of all three of these sequences are the same; they are aligned. If an insertion is encountered, that is, bases in the query sequence that are not in the subject sequence, resulting in hypens ("-") in the subject sequence, then at the first base of the insertion, the data step calculates its size (lines 417-421). As indicated in lines413-414, the first base of the insertion has __SS = "-" and I = 0. The size of the insertion is calculated to determine the logarithm base (note negative integers) of the numbering of the NN and AMPLICON_POSITION, which have decimal values (POSITION will only be positive integers). At the first match following either an insertion or deletion, i.e. the end of the gap, then I or D, respectively is set to 0 (line 425-426). Note that in typical projects, that is, in projects that do not technically have single molecular sequencing or mechanical separation of the chromosomes, heterozygous INDELs will result in "noise" due to the overlay of two distinct sequences after the INDEL (if the INDEL occurs in a single base repeat, the overlay may not be apparent until the end of the repeat). Hemizygous INDELs, such as those that may occur in *F8* of a XY male Hemophilia A patients or X0 (Turner's Syndrome) females, have only one haplotype, thus "clean" sequence (no overlay).

In each iteration (position-by-position parsing of the alignment), the orientation determines whether the value of NN is incremented by 1 (forward) or by -1 (reverse). The incrementation only occurs if the match sequence (__MS) is "|" or the query sequence has a gap (__QS = "-"), lines 432 and 436, respectively for the forward orientation and lines 447 and 457, respectively, for the reverse orientation. In the case that __QS = "-" D is also incremented by +1 (lines 437 and 452). If the gap is in the subject sequence (__SS = "-"), then I but not __NN is incremented (lines 439-440 and 454-455). If the incrementation results in __NN = 0, then if must be offset to 1 or -1, lines 441 and 456, respectively. The progression through both Do blocks (lines 429-442 and lines 444-457) tests first for a match, then a gap in the query sequence (a deletion), then a gap in the subject sequence (an insertion) and increments __NN, D, and I as appropriate.

Once __NN, D, and I are calculated, NN is assigned the value of __NN (always an integer) and used as the key to the NN_AP hash (note, not the AP_NN hash used above), thus providing the values of AMPLICON_POSITION and P for the NN (lines 460-461). The value of AMPLICON_POSITION could be calculated based on the value of SUBJECT_START and incrementation when __SS did not equal "-", but the hash seemed more straightforward and accurate. Lines 463-473 is just an optional integrity check that the value of the base in the subject sequence for the given NN matches the value of the base in the A(G)RS from which it was derived. A mismatch would indicate that NN was miscalculated.

If the base (__QS) from the query sequence is not a gap, then the (Phred) sequence data should be abstracted based on the value of POSITION, the index for the arrays (lines 482-492). Again the quality metrics of the flanks are also calculated (lines 497-518). If the query sequence has a gap at this base, then these values are set to missing (lines 521-536).

If the query sequence has an insertion (I > 0), then the numbering of NN and AMPLICON_POSITION will have a decimal portion instead of strictly being an integer (note that NN is the final output value, but __NN is a integer valued variable that tracks the incrementation from the SUBJECT_START based on the progression through the aligned subject sequence). The number of decimal places needs to be enough to cover the number of bases in the insertion. For instance, an insertion of 10 bases (or more) could not be represented with one decimal place: NN.1 to NN.9 is only 9 bases, thus the program would use NN.01 to NN.10. Note that NN.1 is indistinguishable from NN.10, but if one is presented with NN.11, then one deduces that the insertion has at least 11 and up to 99 bases (NN.01 to NN.99), but not 100 or more bases.

The base to accommodate the decimal portion is calculated on line 534. It will be of the form 10^x, where x = Ceil(Log10(Insertion_Size + **1**)). For example, if the insertion is a single base, then log10(2) = 0.3010299957 ($10^{0.3010299957}$=2.0000000002). Thus ceil(0.3010299957) = 1. The decimal portion will be tenths. If the insertion is 10 bases, then log10(11) = 1.0413926852 and ceil(1.0413926852) = 2. The decimal portion would be hundreds. In addition to NN, both AMPLICON_POSITION and P will have decimals to indicate the insertion (relative to the reference sequence). Although both start at 1, P may increase or decrease with NN, whereas AMPLICON_POSITION strictly increases with increasing NN and decreases with decreasing NN.

Genes may be oriented (transcribed) in the same or opposite orientation as the sequence of reference chromosome. Complicating this, they may be on the reverse compliment of the reference strand (strand = "-") in RefSeq parlance. The terms centromeric-to-telomeric or telomeric-to-centromeric may also apply. For instance, F8 is on the (-) strand of the X chromosome (more specifically at Xq28.1) reference sequence provided by NCBI and it is transcribed telomeric-to-centromeric. As NN increases for the A(G)RS of *F8*, P decreases (See **Figure 5**). Lines 546-550 (forward) and 555-559 (reverse) calculate the appropriate values (see **Figure 6**).

Fortunately, the numbering for deletions is simple. A deletion is the absence of a base. It also does not exist in the query sequence so the value of POSITION should be missing. Consider **A**C**GT** in the reference sequence and **AGT** in the test sequence. The record for C would have NN = 2 and the values of AMPLICON_POSITION and P associated with NN = 2 to from reference amplicon. The value of the CALLED_BASE and UNCALLED_BASE will be missing, as will the values for the other associated Phred data. The second position of the test sequence corresponds to G, thus the value of POSITION for C should be missing. POSITION = 1 for A; POSITION = 2 for G; and POSITION = 3 for T. Since POSITION may be incremented (i.e. it must be retained), its value is temporarily stored (line 566), then set to missing (line 567) for the output (line 570) and then reset to its temporarily stored value after the output (line 572). Note that the reset is conditioned upon the presence of a deletion.

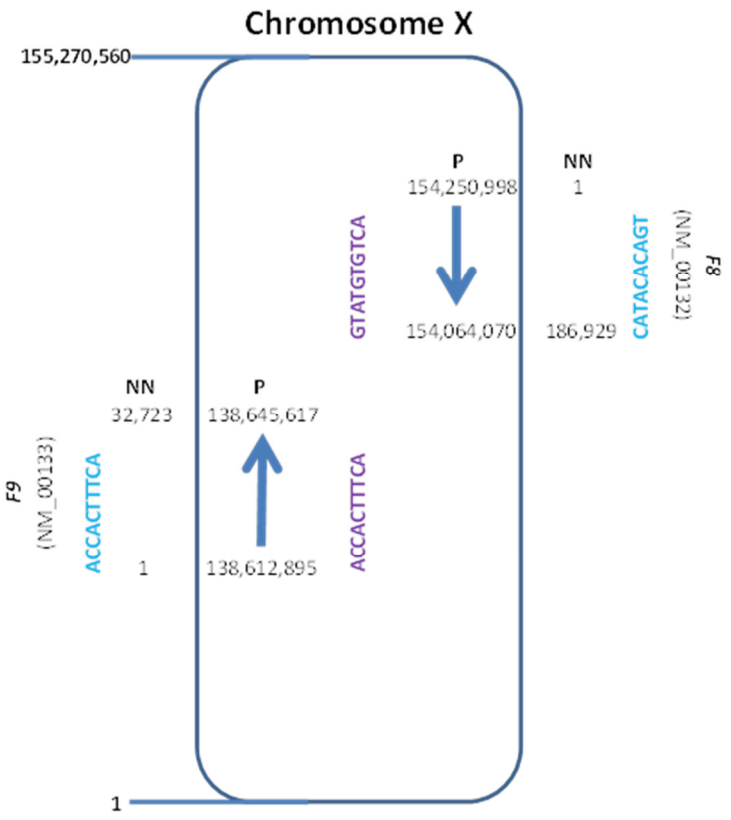

Figure 6. A schematic representation of 155,270,560 base-pair Chromsome X. The numbers correspond to the hg19 release from the National Center for Biotechnology Information (NCBI, http://www.ncbi.nlm.nih.gov/). **P** is relative position of the base in the fasta file of the chromosome sequence. **NN** is nucleotide number, the position of base relative to the start of transcription of the annotated gene reference sequence (AGRS). Two genes are represented: *F8* (NM_00132) and *F9* (NM_00133). *F8* is transcribed from the end of the fasta sequence to its start. The base that corresponds to **NN** = 1 has the largest value of P (154,250,998) of the genic bases of the *F8* AGRS. The top purple sequence in the middle of the chromosome corresponds to P = 154,064,070 to 154,064,079 of the Chromosome X fasta file. By indicating that *F8* is on the (-) strand, sequence of *F8* is taken to be the compliment of this sequence (light blue on the top right). The identity of the base corresponding to NN = 186,929 is C, the compliment of G. To determine the identity of the base while creating an AGRS, one has to specify both P and on which strand the sequence lies. *F9* is more straightforward. It is transcribed in the opposite orientation than *F8* and it is on the (+) strand. The NN of the *F9* AGRS increases with increasing P and the sequence of the *F9* AGRS matches that of the reference X chromosome sequence for a given value of P.

As the start of an alignment of a query sequence might not be at the beginning of the query sequence, the end of the alignment might not include the last base of the query sequence (or the subject sequence). If so, then these uncovered bases need to be included in the data set, but with Phred sequence variables set to missing; only FSN, HIT, NN, AMPLICON_POSITION and P will be populated. However, these observations are only generated for HIT = 1 (lines 607-610).

If the permanent dataset already exists, then the current data are appended to it (lines 618-625). If not, then a new permanent dataset is created (lines 627-631). The accessory temporary data sets created by the macro are deleted (lines 634-643) and the macro ends (line 646).

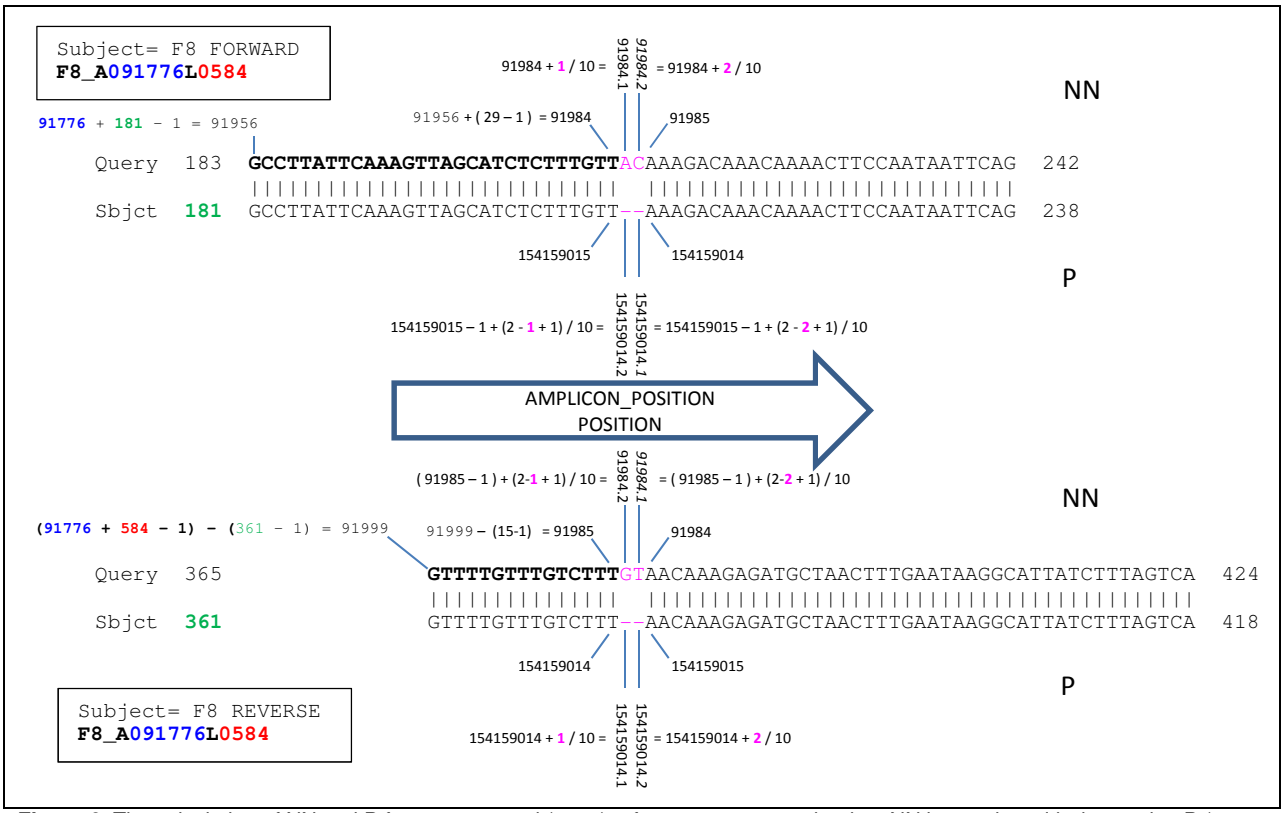

Figure 6. The calculation of NN and P for an annotated (gene) reference sequence that has NN increasing with decreasing P (see Figure 5).

Case Report: F8 re-sequencing project

We attempted to cover the conventionally "functional" regions of *F8* in patients with Hemophilia A (patients, usually male, who bleed due to a deficiency or absents in coagulation Factor VIII activity) with 38 amplicons that covered the 26 exons, their flanking introns, approximately 1,200 base-pairs of the promoter, and approximately 300 base-pairs of the 3' genomic DNA. The amplicons covered 19,892 base-pairs and were sequenced in the forward and reverse orientation. Our minimal depth of coverage was 4X, meaning we attempted to sequence each base of the amplicon at least 4 times: twice in each orientation. Including negative controls, we had 130,875 sequence files for 740 patients. The entire program that called the BC_BLASTn macro required 0:20:11 to run. In comparison, a program to genotype by flanks on a smaller set of data required 2:37:16. Neither of these times is exceedingly long, but considering that the target is the genome and metagenomes of the patient and his or her microbiome, not to mention, their expression, these times need to be markedly improved and the processes sent to cluster of computers. For instance, to guide evidence-based medical decisions for a patient who arrives in an emergency room and is instantly triaged for care.

Some notes from the log following with comments interleaved:

```
NOTE: Table WORK.BLASTN created, with 151805 rows and 26 columns.
(Comment: Some alignments contained multiple hits.)
NOTE: Table WORK.BLASTN_AC created, with 76 rows and 1 columns.
(Comment: 38 amplicons in forward and reverse orientation.)
NOTE: The data set WORK.BLASTN_AC_NN_AP has 39784 observations and 5 variables.
(Comment: 19,892 X 2 = 39,784)
NOTE: There were 39784 observations read from the data set WORK.BLASTN_AC_NN_AP.
NOTE: There were 39784 observations read from the data set WORK.BLASTN_AC_NN_AP.
(Comment: The two reads by the Dataset argument_tag to the hashes.)
NOTE: There were 151805 observations read from the data set WORK.BLASTN.
NOTE: The data set WORK.F8_BC_BASE_BLASTN has 69283345 observations and 23 variables.
NOTE: The data set WORK.QA_BASE has 0 observations and 9 variables.
NOTE: DATA statement used (Total process time):
      real time           13:04.54
```

```
      cpu time            4:46.33
(Comment: Properties according to Window 7 Profession©: 11.4 GB (12,338,480,128 bytes).)
NOTE: There were 69283345 observations read from the data set WORK.F8_BC_BASE_BLASTN.
NOTE: The data set PATH.F8_BC_BASE_BLASTN has 69283345 observations and 23 variables.
NOTE: DATA statement used (Total process time):
      real time           3:08.07
      cpu time            35.52 seconds
(Comment: Moved to permanent location, the next run will APPEND to these data.)
```

CONCLUSION

The tools of bioinformatics allow investigators to utilize genomics in biomedical research and clinicians to access genomic data when making evidence-based medical decisions. Both the quality of the DNA (and RNA) sequencing and a complete record of the sequence attempts, not just the observed variants, are essential to record. A proper investigation should not only confirm variants, but should also confirm that a locus was adequately covered and matches the reference sequence. INDELs and inversion are also prime information resulting from genomics data. The SAS System is an exception tool for bioinformatics. The BC_BLASTn macro presented in this paper populates a table in a relation genomics database that is used currently for research, but could extend to the clinical realm.

ACKNOWLEGEMENTS

This work was supported in part by the following grants: NIH-5RC2HL101851-02, NIH-5R01HL072533-04, and NIH-5K08HL071130-05. In addition, Greg and Rebecca Kaneb graciously supported this effort. Finally, the author is grateful for the faith, support, and mentorship of inVentiv Health Clinical.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:
Name: Kevin Viel
Enterprise: inVentiv Health Clinical
Histonis, Incorporated
E-mail: kevin.viel@inventivhealth.com, kviel@histonis.org
Web: http://www.inventivhealthclinical.com/

REFERENCES

[1] Viel, K. Creating reference amplicons and genotyping using the SAS System. *PharmaSUG 2011 Proceedings*. http://www.pharmasug.org/proceedings/2011/AD/PharmaSUG-2011-AD16.pdf

[2] Grabich S and Viel K. Processing the RefSeq and CCDS Annotation Datasets Using the SAS System: Creation of Gene Reference. *PharmaSUG 2011 Proceedings*. http://www.pharmasug.org/proceedings/2011/PO/PharmaSUG-2011-PO12.pdf

[3] Viel, K. Using the SAS System as a bioinformatics tool: a macro that calls the standalone Basic Local Alignment Search Tool (BLAST) setup. *PharmaSUG 2012 Proceedings*. http://www.pharmasug.org/proceedings/2012/HO/PharmaSUG-2012-HO07.pdf

[4] Camacho, C., et al., *BLAST+: architecture and applications.* BMC Bioinformatics, 2009. **10**: p. 421.

CODE 1

```
%Macro BC_BLASTn ( LN           =
                 , Phred        =
                 , Obs          = 5000
                 , Seq          =
                 , Fwd_NN_inc_P = Y     /* In the FORWARD orientation, does NN increase with P */
                 , BC_Base      =
                 , BLASTn       =
                 , QC_Seq       = N
                 ) ;

  Proc SQL ;
    Create Table BLASTn As
    Select *
    From &LN..&BLASTn. ( PW = &PW. )
    %If %SysFunc( Exist( &LN..&BC_Base. ))
    %Then
       %Do ;
          /* Do not select FSN's that have already been processed */
          Where     FSN ~ In ( Select FSN
                               From &LN..&BC_Base. ( PW = &PW. )
                             )
       %End ;
    Order By FSN
          , Hit
    ;
  Quit ;

  /* Amplicon Coverage */
  Proc SQL ;
    Create Table BLASTn_AC As
    Select Distinct Subject
    From BLASTn
    ;
  Quit ;

  Data BLASTn_AC_NN_AP ( Keep = Subject NN P Amplicon_Position
                                %If &QC_Seq. = Y %Then Base ;
                       ) ;

    Set BLASTn_AC ;

    /* SS_NN = Subject Start Nucleotide Number */
    SS_NN = Input( Scan( Scan( Upcase( Subject ) , -1 , " " ) , 2 , "AL" ) , 8. ) ;
    If Scan( Scan( Subject , -1 , " " ) , 2 , "_" ) =: "a" Then SS_NN = SS_NN * -1 ;

    Subject_Length = Input( Scan( Scan( Subject , -1 , " " ) , 2 , "L" ) , 8. ) ;

    Direction = Scan( Subject , 2 , " " ) ;

    If Direction = "FORWARD"
    Then
      Do ;
         NN = SS_NN - 1 ;
         If NN = 0 Then NN = -1 ;
         Amplicon_Position = 0 ;

         Do Until ( Amplicon_Position = Subject_Length ) ;
           Amplicon_Position + 1 ;
           NN + 1 ;
           If NN = 0 then NN = 1 ;
           /* P */
           Set &Seq. ( PW   = &PW.
                       Keep = NN P %If &QC_Seq. = Y %Then Base ;
                     )
               Key = NN / Unique
               ;
           Output ;
         End ;
      End ; /* Direction = "FORWARD" */
     Else If Direction = "REVERSE"
     Then
       Do ;
            /* Account for the lack of a zero */
          If SS_NN < 0
          Then
            Do ;
               If SS_NN + Subject_Length <= 0 Then NN = SS_NN + Subject_Length - 1 ;
                Else If SS_NN + Subject_Length > 0 Then NN = SS_NN + Subject_Length ;
            End ;
           Else If SS_NN > 0 Then NN = SS_NN + Subject_Length - 1 ;

          NN = NN + 1 ;
          If NN = 0 Then NN = 1 ;
          Amplicon_Position = 0 ;

          Do Until ( Amplicon_Position = Subject_Length ) ;
            Amplicon_Position + 1 ;
            NN + (-1) ;
            If NN = 0 then NN = -1 ;
            /* P */
            Set &Seq. ( PW   = &PW.
                        Keep = NN P %If &QC_Seq. = Y %Then Base ;
                      )
```

```
                Key = NN / Unique
                ;
            %If &QC_Seq. = Y %Then Base = Upcase( Translate( Upcase( Base )
                                                           , "a" , "T"
                                                           , "c" , "G"
                                                           , "g" , "C"
                                                           , "t" , "A"
                                                           )
                                                 ) %Str(;) ;
            Output ;
          End ;
       End ; /* Direction = "REVERSE" */

   Run ;

   Data &BC_Base. ( Keep = FSN Hit NN P I D Amplicon_Position Position
                           C_B U_B RA_C RA_U C_BP U_BP QV
                           QV_FPF QV_TPF Noise_FPF Noise_TPF
                           A_Amplitude C_Amplitude G_Amplitude T_Amplitude
                  )
       %If &QC_Seq. = Y %Then QA_Base ( Keep = FSN Hit NN __QS __MS __SS Base
                                              Amplicon_Position Position
                                      ) ;
       ;

     Length FSN
            Hit
            NN
            I
            D
            Amplicon_Position
            Position
            P                   8
            C_B
            U_B                 $ 1
            RA_C
            RA_U
            C_BP
            U_BP
            QV
            QV_FPF
            QV_TPF
            Noise_FPF
            Noise_TPF
            A_Amplitude
            C_Amplitude
            G_Amplitude
            T_Amplitude         8
            __QS
            __MS
            __SS
            Direction           $ 7
            %If &QC_Seq. = Y %Then Base $ 1 ;
            ;

     Retain Stop ;

     If 0 Then Set BLASTn_AC_NN_AP ;

     If _n_ = 1
     Then
       Do ;
          /* AP_NN */
          *Declare Hash AP_NN ( DataSet: "BLASTn_AC_NN_AP" ) ;
          Declare Hash AP_NN ( ) ;
          __RC = AP_NN.DefineKey ( "Subject"
                                 , "Amplicon_Position"
                                 ) ;
          __RC = AP_NN.DefineData( "NN"
                                 , "P"
                                 %If &QC_Seq. = Y %Then %Str(,) "Base" ;
                                 ) ;
          __RC = AP_NN.DefineDone( ) ;

          /* AP_NN */
          * Declare Hash NN_AP ( DataSet: "BLASTn_AC_NN_AP" ) ;
          Declare Hash NN_AP ( ) ;
          __RC = NN_AP.DefineKey ( "Subject"
                                 , "NN"
                                 ) ;
          __RC = NN_AP.DefineData( "Amplicon_Position"
                                 , "P"
                                 %If &QC_Seq. = Y %Then %Str(,) "Base" ;
                                 ) ;
          __RC = NN_AP.DefineDone( ) ;

          Do Until ( End_AC ) ;
            Set BLASTn_AC_NN_AP End = End_AC ;
            __RC_AP = AP_NN.Add() ;
            __RC_AP = NN_AP.Add() ;
          End ;


       End ;

```

```
189     /*********************/
190     /* Poly (Phred Data) */
191     /*********************/
192     /* Called base */
193     Array CB( &Obs. )   $  1 _Temporary_ ;
194     /* Uncalled base */
195     Array UB( &Obs. )   $  1 _Temporary_ ;
196     /* Called Base Relative Area */
197     Array CBRA( &Obs. ) _Temporary_ ;
198     /* Uncalled Base Relative Area */
199     Array UBRA( &Obs. ) _Temporary_ ;
200     /* Called Base Pos */
201     Array CBP( &Obs. )  _Temporary_ ;
202     /* Uncalled Base Relative Area */
203     Array UBP( &Obs. )  _Temporary_ ;
204
205     /* Amplitude */
206     Array A_Amp ( &Obs. ) _Temporary_ ;
207     Array C_Amp ( &Obs. ) _Temporary_ ;
208     Array G_Amp ( &Obs. ) _Temporary_ ;
209     Array T_Amp ( &Obs. ) _Temporary_ ;
210
211     /*********************/
212     /* PHD (Phred Data) */
213     /*********************/
214     Array Qual( &Obs. ) _Temporary_ ;
215
216     Set BLASTn ;
217     By FSN
218        Hit
219        ;
220
221     If      Query_Sequence =  "***** No hits found *****"
222        Or   Strand         ne "Plus/Plus"
223     Then
224       Do ;
225          Call Missing( I
226                      , D
227                      , Position
228                      , C_B
229                      , U_B
230                      , RA_C
231                      , RA_U
232                      , C_BP
233                      , U_BP
234                      , QV
235                      , A_Amplitude
236                      , C_Amplitude
237                      , G_Amplitude
238                      , T_Amplitude
239                      , QV_FPF
240                      , QV_TPF
241                      , Noise_FPF
242                      , Noise_TPF
243                      ) ;
244
245          If First.FSN
246          Then
247            Do ;
248               Do Amplicon_Position = 1 To Subject_Length ;
249                 __RC = AP_NN.Find() ;
250                 Output &BC_Base. ;
251               End ;
252             End ;
253       End ;
254      Else
255        Do ;
256           If FSN ne Lag( FSN )
257           Then
258             Do ;
259                Call Missing( of CB1    - CB&Obs.
260                            , of UB1    - UB&Obs.
261                            , of CBRA1  - CBRA&Obs.
262                            , of UBRA1  - UBRA&Obs.
263                            , of CBP1   - CBP&Obs.
264                            , of UBP1   - UBP&Obs.
265                            , of A_Amp1 - A_Amp&Obs.
266                            , of C_Amp1 - C_Amp&Obs.
267                            , of G_Amp1 - G_Amp&Obs.
268                            , of T_Amp1 - T_Amp&Obs.
269                            , of Qual1  - Qual&Obs.
270                            , Stop
271                            ) ;
272
273                /************************************/
274                /* Obtain the Sequence and Base data */
275                /************************************/
276                Do Until ( End ) ;
277
278                  Set &LN..&Phred. ( PW = &PW. )
279                      Key = FSN
280                      End = End
281                      ;
282
283                  If __OK ne "1" And _IORC_ = %SysRC(_SOK) Then __OK = "1" ;
```

```
                CB   ( Position ) = Called_base                ;
                UB   ( Position ) = Uncalled_base              ;
                CBRA ( Position ) = Called_base_rel_peak_area  ;
                UBRA ( Position ) = Uncalled_base_rel_peak_area ;
                CBP  ( Position ) = Called_base_pos            ;
                UBP  ( Position ) = Uncalled_base_pos          ;
                Qual ( Position ) = QV                         ;
                A_Amp( Position ) = A_Amplitude                ;
                C_Amp( Position ) = C_Amplitude                ;
                G_Amp( Position ) = G_Amplitude                ;
                T_Amp( Position ) = T_Amplitude                ;

              End ;

              If End And __OK = "1"
              Then
                Do ;
                   __OK   = "" ;
                  _Error_ = 0  ;
                  End     = 0  ;
                End ;

              Stop = Position ;

            End ; /* FSN ne Lag( FSN ) */

          Call Missing( I
                      , D
                      , Position
                      , C_B
                      , U_B
                      , RA_C
                      , RA_U
                      , C_BP
                      , U_BP
                      , QV
                      , A_Amplitude
                      , C_Amplitude
                      , G_Amplitude
                      , T_Amplitude
                      , QV_FPF
                      , QV_TPF
                      , Noise_FPF
                      , Noise_TPF
                      ) ;

          /* Beginning of the parsing of the BLASTn results */
          Direction = Scan( Subject , 2 , " " ) ;

          /* 5' Non-hits */
          /* Find the NN for the given AP */
          If     Subject_Start ne 1
             And Hit = 1
          Then
            Do ;
              Do Amplicon_Position = 1 To Subject_Start - 1 ;
                __RC = AP_NN.Find() ;
                Output &BC_Base. ;
              End ;
            End ;

          /* Start of sequence processing */
          /* Phred Index */
          Position = Query_Start ;

          I = 0 ;
          D = 0 ;

          __QS = Upcase( SubStr( Query_Sequence   , 1 , 1 )) ;
          __MS =         SubStr( Match_Sequence   , 1 , 1 )  ;
          __SS = Upcase( SubStr( Subject_Sequence , 1 , 1 )) ;

          /* Determine NN for AP = Subject_Start */
          Amplicon_Position = Subject_Start ;
          __RC = AP_NN.Find() ;
          __NN = NN ;

          /* QC */
          %If &QC_Seq. = Y %Then If Upcase( __SS ) ne Upcase(Base) Then Output QA_Base %Str(;) ;

          C_B         = CB   ( Position ) ;
          U_B         = UB   ( Position ) ;
          RA_C        = CBRA ( Position ) ;
          RA_U        = UBRA ( Position ) ;
          C_BP        = CBP  ( Position ) ;
          U_BP        = UBP  ( Position ) ;
          QV          = Qual ( Position ) ;
          A_Amplitude = A_Amp( Position ) ;
          C_Amplitude = C_Amp( Position ) ;
          G_Amplitude = G_Amp( Position ) ;
          T_Amplitude = T_Amp( Position ) ;


          /*****************/
```

```
379             /* FPF QV Noise */
380             /*****************/
381             QV_FPF    = 0 ;
382             Noise_FPF = 0 ;
383             Do _n_ = Position - 10 To Position - 1 ;
384               If 0 < _n_ <= Stop
385               Then
386                 Do ;
387                    QV_FPF    + ( Qual( _n_ ) > 24   ) ;
388                    Noise_FPF + ( UBRA( _n_ ) > 0.15 ) ;
389                 End ;
390             End ;
391
392             /* TPF QV Noise */
393             QV_TPF    = 0 ;
394             Noise_TPF = 0 ;
395             Do _n_ = Position + 1 To Position + 10 ;
396               If 0 < _n_ <= Stop
397               Then
398                 Do ;
399                    QV_TPF    + ( Qual( _n_ ) > 24 ) ;
400                    Noise_TPF + ( UBRA( _n_ ) > 0.15 ) ;
401                    End ;
402             End ;
403
404             Output &BC_Base. ;
405
406             /* Cycle through the sequence */
407             Do _n_ = 2 To Length( Query_Sequence ) ;
408
409               __QS = Upcase( SubStr( Query_Sequence   , _n_ , 1 )) ;
410               __MS =         SubStr( Match_Sequence   , _n_ , 1 ) ;
411               __SS = Upcase( SubStr( Subject_Sequence , _n_ , 1 )) ;
412
413               If     __SS = "-"
414                  And I    = 0
415               Then
416                 Do ;
417                    Insertion_Size = 0 ;
418                    Do Insertion_Size_n = _n_ By 1 Until ( __IS ne "-" ) ;
419                       __IS = Upcase( SubStr( Subject_Sequence , Insertion_Size_n , 1 )) ;
420                       If __IS = "-" Then Insertion_Size + 1 ;
421                    End ;
422                 End ;
423
424               /* INDEL */
425               If I > 0 and __SS ne "-" Then I = 0 ;
426               If D > 0 and __QS ne "-" Then D = 0 ;
427
428               /* NN */
429               If Direction = "FORWARD"
430               Then
431                 Do ;
432                    If __MS = "|" Then __NN + 1 ;
433                     Else If __QS = "-"
434                     Then
435                       Do ;
436                          __NN + 1 ;
437                          D + 1 ;
438                       End ;
439                     Else If __SS = "-" Then I + 1 ;
440                      Else __NN + 1 ;
441                    If __NN = 0 Then __NN = 1 ;
442                 End ; /* Direction = "FORWARD" */
443
444                Else If Direction = "REVERSE"
445                Then
446                  Do ;
447                     If __MS = "|" Then __NN + ( -1 ) ;
448                      Else If __QS = "-"
449                      Then
450                        Do ;
451                           __NN + ( -1 ) ;
452                           D + 1 ;
453                        End ;
454                      Else If __SS = "-" Then I + 1 ;
455                       Else __NN + ( -1 ) ;
456                    If __NN = 0 Then __NN = -1 ;
457                 End ; /* Direction = "REVERSE" */
458
459               /* Find the AP for the given NN */
460               NN   = __NN ;
461               __RC = NN_AP.Find() ;
462
463               %If &QC_Seq. = Y
464               %Then
465                  %Do ;
466                      /* Seq check */
467                      If     __MS =  "|"
468                         Or  __SS ne "-"
469                      Then
470                        Do ;
471                           If Upcase( __SS ) ne Upcase( Base ) Then Output QA_Base ;
472                        End ;
473                  %End ;
```

```
474
475             /* Phred Index */
476             If __QS ne "-"
477             Then
478               Do ;
479
480                  Position + 1 ;
481
482                  C_B         = CB   ( Position ) ;
483                  U_B         = UB   ( Position ) ;
484                  RA_C        = CBRA ( Position ) ;
485                  RA_U        = UBRA ( Position ) ;
486                  C_BP        = CBP  ( Position ) ;
487                  U_BP        = UBP  ( Position ) ;
488                  QV          = Qual ( Position ) ;
489                  A_Amplitude = A_Amp( Position ) ;
490                  C_Amplitude = C_Amp( Position ) ;
491                  G_Amplitude = G_Amp( Position ) ;
492                  T_Amplitude = T_Amp( Position ) ;
493
494                  /*****************/
495                  /* FPF QV Noise */
496                  /*****************/
497                  QV_FPF    = 0 ;
498                  Noise_FPF = 0 ;
499                  Do __np = Position - 10 To Position - 1 ;
500                    If 0 < __np <= Stop
501                    Then
502                      Do ;
503                         QV_FPF    + ( Qual( __np ) > 24   ) ;
504                         Noise_FPF + ( UBRA( __np ) > 0.15 ) ;
505                      End ;
506                  End ;
507
508                  /* TPF QV Noise */
509                  QV_TPF    = 0 ;
510                  Noise_TPF = 0 ;
511                  Do __np = Position + 1 To Position + 10 ;
512                    If 0 < __np <= Stop
513                    Then
514                      Do ;
515                         QV_TPF    + ( Qual( __np ) > 24 ) ;
516                             Noise_TPF + ( UBRA( __np ) > 0.15 ) ;
517                      End ;
518                  End ;
519               End ; /* __QS ne "-" */
520
521              Else Call Missing( C_B
522                               , U_B
523                               , RA_C
524                               , RA_U
525                               , C_BP
526                               , U_BP
527                               , QV
528                               , A_Amplitude
529                               , C_Amplitude
530                               , G_Amplitude
531                               , T_Amplitude
532                               , QV_FPF
533                               , QV_TPF
534                               , Noise_FPF
535                               , Noise_TPF
536                               ) ;
537
538             If I > 0
539             Then
540               Do ;
541                  /* For instance, 10 base insertion following NN = 1: 1.01 to 1.10 */
542                  I_Base = 10 ** Ceil( Log10( Insertion_Size + 1 )) ;
543                  If Direction = "FORWARD"
544                  Then
545                    Do ;
546                       If NN > 0 Then NN = NN + I / I_Base ;
547                        Else NN = ( NN + 1 ) - ( Insertion_Size - I + 1 ) / I_Base ;
548                      Amplicon_Position = Amplicon_Position + I / I_Base ;
549                      %If &Fwd_NN_inc_P. = Y %Then P = P + I / I_Base %Str(;) ;
550                       %Else P = P - 1 + ( Insertion_Size - I + 1 ) / I_Base %Str(;) ;
551                    End ;
552                   Else If Direction = "REVERSE"
553                   Then
554                     Do ;
555                       If NN > 1 Then NN = ( NN - 1 ) + ( Insertion_Size - I + 1 ) / I_Base ;
556                        Else NN = NN  - I / I_Base ;
557                       Amplicon_Position = Amplicon_Position + I / I_Base ;
558                      %If &Fwd_NN_inc_P. = Y %Then P = P - 1 + ( Insertion_Size - I + 1 ) / I_Base %Str(;) ;
559                        %Else P = P + I / I_Base %Str(;) ;
560                     End ;
561               End ; /* I > 0 */
562
563             If D > 0
564             Then
565               Do ;
566                  __Position = Position ;
567                  Position   = . ;
568               End ;
```

```
569
570             Output &BC_Base. ;
571
572             If D > 0 Then Position = __Position ;
573
574           End ; /* Cycled through the length of the sequence */
575
576           /* 3' Non-hits */
577           /* Find the AP for the given NN */
578           /* For BLASTn, the last position will not be an INDEL: NN will be known */
579           __RC = NN_AP.Find() ;
580
581           If Amplicon_Position ne Subject_Length
582           Then
583             Do ;
584                Call Missing( I
585                            , D
586                            , Position
587                            , C_B
588                            , U_B
589                            , RA_C
590                            , RA_U
591                            , C_BP
592                            , U_BP
593                            , QV
594                            , A_Amplitude
595                            , C_Amplitude
596                            , G_Amplitude
597                            , T_Amplitude
598                            , QV_FPF
599                            , QV_TPF
600                            , Noise_FPF
601                            , Noise_TPF
602                            ) ;
603
604                If Hit = 1
605                Then
606                  Do ;
607                     Do Amplicon_Position = Amplicon_Position + 1 To Subject_Length ;
608                       __RC = AP_NN.Find() ;
609                       Output &BC_Base. ;
610                     End ;
611                  End ;
612             End ; /* Amplicon_Position ne Subject_Length */
613
614       End ; /* ELSE: Query_Sequence ne "***** No hits found *****" */
615
616   Run ;
617
618   %If %SysFunc( Exist( &LN..&BC_Base. ))
619   %Then
620      %Do ;
621          Proc Append Base = &LN..&BC_Base. ( PW = &PW. )
622                      Data = &BC_Base.
623                      ;
624          Run ;
625      %End ;
626    %Else
627       %Do ;
628           Data &LN..&BC_Base. ( PW = &PW. ) ;
629             Set &BC_Base. ;
630           Run ;
631       %End ;
632
633   /********/
634   Proc Datasets Library = Work
635                 NoList
636                 MemType = Data
637                 ;
638     Delete &BC_Base.
639            BLASTn
640            BLASTn_AC
641            BLASTn_AC_NN_AP
642            ;
643   Quit ;
644
645
646 %MEnd BC_BLASTn ;
```