BLAT Suite Program Specifications and User Guide

General:

Blat produces two major classes of alignments: at the DNA level between two sequences that are of 95% or greater identity, but which may include large inserts, and at the protein or translated DNA level between sequences that are of 80% or greater identity and may also include large inserts. The output of BLAT is flexible. By default it is a simple tab-delimited file which describes the alignment, but which does not include the sequence of the alignment itself. Optionally it can produce BLAST and WU-BLAST compatable output as well as a number of other formats.

The main programs in the blat suite are:

- gfServer – a server that maintains an index of the genome in memory and uses the index to quickly find regions with high levels of sequence similarity to a query sequence.
- gfClient – a program that queries gfServer over the network, and then does a detailed alignment of the query sequence with regions found by gfServer.
- blat –combines client and server into a single program, first building the index, then using the index, and then exiting.
- webBlat – a web based version of gfClient that presents the alignments in an interactive fashion.

Building an index of the genome typically takes 10 or 15 minutes. Typically for interactive applications one uses gfServer to build a whole genome index. At that point gfClient or webBlat can align a single query within few seconds. If one is aligning a lot of sequences in a batch mode then blat can be more efficient, particularly if run on a cluster of computers. Each blat run is typically done against a single chromosome, but with a large number of query sequences.

Other programs in the blat suite are:

- pslSort – combines and sorts the output of multiple blat runs. (The blat default output format is .psl).
- pslReps – selects the best alignments for a particular query sequence, using a 'near best in genome' approach.
- pslPretty – converts alignments from the psl format, which is tab-delimited format and does not include the bases themselves, to a more readable alignment format.
- faToTwoBit – convert Fasta format sequence files to a dense randomly-accessable .2bit format that gfClient can use.
- twoBitToFa – convert from the .2bit format back to fasta
- faToNib – convert from Fasta to a somewhat less dense randomly accessible format that predates .2bit. Note each .nib file can only contain a single sequence.
- nibFrag – convert portions of a nib file back to fasta.

In addition you may be interested in the following programs which are not part of the BLAT suite:

- In Silico PCR – given two primers quickly find the sequence between them. Available from Kent Informatics. This includes webPCR, an interface similar to webBlat.
- The Genome Browser – display annotations as a series of tracks on top of the genome. Available from the University of California Santa Cruz. See http://genome.ucsc.edu/license/.

Running the Programs:

The command line options of each of the programs is described below. Similar summaries of usage are printed when a command is run with no arguments. See the next section for info on installing webBlat.

blat

```
blat - Standalone BLAT sequence search command line tool
usage:
   blat database query [-ooc=11.ooc] output.psl
where:
   database is either a .fa file, a .nib file, or a list of .fa or .nib
   files, query is similarly a .fa, .nib, or list of .fa or .nib files
   -ooc=11.ooc tells the program to load over-occurring 11-mers from
               and external file.  This will increase the speed
               by a factor of 40 in many cases, but is not required
   output.psl is where to put the output.
```

```
options:
   -t=type     Database type.  Type is one of:
                 dna - DNA sequence
                 prot - protein sequence
                 dnax - DNA sequence translated in six frames to protein
               The default is dna
   -q=type     Query type.  Type is one of:
                 dna - DNA sequence
                 rna - RNA sequence
                 prot - protein sequence
                 dnax - DNA sequence translated in six frames to protein
                 rnax - DNA sequence translated in three frames to protein
               The default is dna
   -prot       Synonymous with -d=prot -q=prot
   -ooc=N.ooc  Use overused tile file N.ooc.  N should correspond to
               the tileSize
   -tileSize=N sets the size of match that triggers an alignment.
               Usually between 8 and 12
               Default is 11 for DNA and 5 for protein.
   -oneOff=N   If set to 1 this allows one mismatch in tile and still
               triggers an alignments.  Default is 0.
```

```
-minMatch=N sets the number of tile matches.  Usually set from 2 to 4
            Default is 2 for nucleotide, 1 for protein.
-minScore=N sets minimum score.  This is twice the matches minus the
            mismatches minus some sort of gap penalty.  Default is 30
-minIdentity=N Sets minimum sequence identity (in percent).  Default is
            90 for nucleotide searches, 25 for protein or translated
            protein searches.
-maxGap=N   sets the size of maximum gap between tiles in a clump.  Usually
            set from 0 to 3.  Default is 2. Only relevent for minMatch > 1.
-noHead     suppress .psl header (so it's just a tab-separated file)
-makeOoc=N.ooc Make overused tile file
-repMatch=N sets the number of repetitions of a tile allowed before
            it is marked as overused.  Typically this is 256 for tileSize
            12, 1024 for tile size 11, 4096 for tile size 10.
            Default is 1024.  Typically only comes into play with makeOoc
-mask=type  Mask out repeats.  Alignments won't be started in masked region
            but may extend through it in nucleotide searches.  Masked areas
            are ignored entirely in protein or translated searches. Types are
              lower - mask out lower cased sequence
              upper - mask out upper cased sequence
              out   - mask according to database.out RepeatMasker .out file
              file.out - mask database according to RepeatMasker file.out
-qMask=type Mask out repeats in query sequence.  Similar to -mask above but
            for query rather than target sequence.
-minRepDivergence=NN - minimum percent divergence of repeats to allow
            them to be unmasked.  Default is 15.  Only relevant for
            masking using RepeatMasker .out files.
-dots=N     Output dot every N sequences to show program's progress
-trimT      Trim leading poly-T
-noTrimA    Don't trim trailing poly-A
-trimHardA  Remove poly-A tail from qSize as well as alignments in psl output
-out=type   Controls output file format.  Type is one of:
                psl - Default.  Tab separated format without actual sequence
                pslx - Tab separated format with sequence
                axt - blastz-associated axt format
                maf - multiz-associated maf format
                wublast - similar to wublast format
                blast - similar to NCBI blast format
-fine       For high quality mRNAs look harder for small initial and
            terminal exons.  Not recommended for ESTs
```

Here are some blat settings for common usage scenarios:

1) Mapping ESTs to the genome within the same species
```
-ooc=11.ooc
```
2) Mapping full length mRNAs to the genome in the same species
```
-ooc=11.ooc -fine -q=rna
```
3) Mapping ESTs to the genome across species
```
-q=dnax -t=dnax
```
4) Mapping mRNA to the genome across species
```
-q=rnax -t=dnax
```
5) Mapping proteins to the genome
```
-q=prot -t=dnax
```
6) Mapping DNA to DNA in the same species
```
-ooc=11.ooc -fastMap
```
7) Mapping DNA from one species to another species
```
-q=dnax -t=dnax
When mapping DNA from one species to another the
query side of the alignment should be cut up into chunks
of 25kb or less for best performance.
```

gfServer

```
gfServer - Make a server to quickly find where DNA occurs in genome.
To set up a server:
   gfServer start host port file(s).nib
To remove a server:
   gfServer stop host port
To query a server with DNA sequence:
   gfServer query host port probe.fa
To query a server with protein sequence:
   gfServer protQuery host port probe.fa
To query a server with translated dna sequence:
   gfServer transQuery host port probe.fa
To process one probe fa file against a .nib format genome (not starting server):
   gfServer direct probe.fa file(s).nib
To figure out usage level
   gfServer status host port
To get input file list
   gfServer files host port
Options:
   -tileSize=N size of n-mers to index.  Default is 11 for nucleotides, 4 for
               proteins (or translated nucleotides).
   -minMatch=N Number of n-mer matches that trigger detailed alignment
               Default is 2 for nucleotides, 3 for protiens.
   -maxGap=N   Number of insertions or deletions allowed between n-mers.
               Default is 2 for nucleotides, 0 for protiens.
   -trans      Translate database to protein in 6 frames.  Note: it is best
               to run this on RepeatMasked data in this case.
   -log=logFile keep a log file that records server requests.
   -seqLog    Include sequences in log file
```

gfClient

```
gfClient - A client for the genomic finding program
usage:
   gfClient host port nibDir in.fa out.psl
where
   host is the name of the machine running the gfServer
   port is the same as you started the gfServer with
   nibDir is the path of the nib files relative to the current dir
       (note these are needed by the client as well as the server)
   in.fa a fasta format file.  May contain multiple records
   out.psl where to put the output
options:
   -t=type     Database type.  Type is one of:
                 dna - DNA sequence
                 prot - protein sequence
                 dnax - DNA sequence translated in six frames to protein
               The default is dna
   -q=type     Query type.  Type is one of:
                 dna - DNA sequence
                 rna - RNA sequence
                 prot - protein sequence
                 dnax - DNA sequence translated in six frames to protein
                 rnax - DNA sequence translated in three frames to protein
   -dots=N   Output a dot every N query sequences
   -nohead   Suppresses psl five line header
   -minScore=N sets minimum score.  This is twice the matches minus the
               mismatches minus some sort of gap penalty.  Default is 30
   -minIdentity=N Sets minimum sequence identity (in percent).  Default is
               90 for nucleotide searches, 25 for protein or translated
               protein searches.

   -out=type   Controls output file format.  Type is one of:
                   psl - Default.  Tab separated format without actual sequence
```

```
                  pslx - Tab separated format with sequence
                  axt - blastz-associated axt format
                  maf - multiz-associated maf format
                  wublast - similar to wublast format
                  blast - similar to NCBI blast format
   -maxIntron=N  Sets maximum intron size. Default is 750000
```

webBlat

```
webBlat generally is not run from the command line.  See 'Setting Up webBlat instructions below`
for information on this program.
```

faToTwoBit

```
faToTwoBit - Convert DNA from fasta to 2bit format
usage:
   faToTwoBit in.fa [in2.fa in3.fa ...] out.2bit
options:
   -noMask - Ignore lower-case masking in fa file.
```

twoBitToFa

```
twoBitToFa - Convert all or part of .2bit file to fasta
usage:
   twoBitToFa input.2bit output.fa
options:
   -seq=name - restrict this to just one sequence
   -start=X  - start at given position in sequence (zero-based)
   -end=X - end at given position in sequence (non-inclusive)
```

faToNib

```
faToNib - Convert from .fa to .nib format
usage:
   faToNib in.fa out.nib
```

nibFrag

```
nibFrag - Extract part of a nib file as .fa
usage:
   nibFrag file.nib start end strand out.fa
```

pslPretty

```
pslPretty - Convert PSL to human readable output
usage:
   pslPretty in.psl target.lst query.lst pretty.out
options:
   -axt - save in Scott Schwartz's axt format
   -dot=N Put out a dot every N records
   -long - Don't abbreviate long inserts
```

It's a really good idea if the psl file is sorted by target if it contains multiple targets. Otherwise this will be very very slow. The target and query lists can either be fasta files, nib files, or a list of fasta and/or nib files one per line. Currently this onl handles nucleotide based psl files.

pslSort

```
pslSort - merge and sort psCluster .psl output files
usage:
  pslSort dirs[1|2] outFile tempDir inDir(s)
This will sort all of the .psl files in the directories
inDirs in two stages - first into temporary files in tempDir
and second into outFile.  The device on tempDir needs to have
enough space (typically 15-20 gigabytes if processing whole genome)
  pslSort g2g[1|2] outFile tempDir inDir(s)
```

```
This will sort a genome to genome alignment, reflecting the
alignments across the diagonal.

Adding 1 or 2 after the dirs or g2g will limit the program to
only the first or second pass repectively of the sort

Options:
   -verbose=N Set verbosity level, higher for more output. Default 1

Note for huge files pslSort will run out of memory.  The unix sort command
  sort -k 10 *.psl > sorted.psl
may be preferable in these situations, though the psl header lines should
be removed or avoided with the -noHead option to blat.
```

pslReps

```
pslReps - analyse repeats and generate genome wide best
alignments from a sorted set of local alignments
usage:
    pslReps in.psl out.psl out.psr
where in.psl is an alignment file generated by psLayout and
sorted by pslSort, out.psl is the best alignment output
and out.psr contains repeat info
options:
    -nohead don't add PSL header
    -ignoreSize Will not weigh in favor of larger alignments so much
    -noIntrons Will not penalize for not having introns when calculating
              size factor
    -singleHit  Takes single best hit, not splitting into parts
    -minCover=0.N minimum coverage to output.  Default is 0.
    -ignoreNs Ignore 'N's when calculating minCover.
    -minAli=0.N minimum alignment ratio
               default is 0.93
    -nearTop=0.N how much can deviate from top and be taken
               default is 0.01
    -minNearTopSize=N  Minimum size of alignment that is near top
               for aligmnent to be kept.  Default 30.
    -coverQSizes=file Tab-separate file with effective query sizes.
                    When used with -minCover, this allows polyAs
                    to be excluded from the coverage calculation
```

Setting Up webBlat

INSTALLING WEBBLAT

Installing A Web-Based Blat Server involves four major steps:
1) Creating sequence databases.
2) Running the gfServer program to create in-memory indexes of the databases.
3) Editing the webBlat.cfg file to tell it what machine and port the gfServer(s) are running on, and optionally customizing the webBlat appearance to users.
4) Copying the webBlat executable and webBlat.cfg to a directory where the web server can execute webBlat as a CGI.

CREATING SEQUENCE DATABASES

You create databases with the program faToTwoBit. Typically you'll create a separate database for each genome you are indexing. Each database can contain up to four billion bases of sequence in an unlimited number of records. The databases for webPcr and webBlat are identical.

The input to faToTwoBit is one or more fasta format files each of which can contain multiple records. If the sequence contains repeat sequences, as is the case with vertebrates and many plants, the repeat sequences can

be represented in lower case and the other sequence in upper case. The gfServer program can be configured to ignore the repeat sequences. The output of faToTwoBit is a file which is designed for fast random access and efficient storage. The output files store four bases per byte. They use a small amount of additional space to store the case of the DNA and to keep track of runs of N's in the input. Non-N ambiguity codes such as Y and U in the input sequence will be converted to N.

Here's how a typical installation might create a mouse and a human genome database:

```
cd /data/genomes
mkdir twoBit
faToTwoBit human/hg16/*.fa twoBit/hg16.2bit
faToTwoBit mouse/mm4/*.fa twoBit/mm4.2bit
```

There's no need to put all of the databases in the same directory, but it can simplify bookkeeping.

The databases can also be in the .nib format which was used with blat and gfClient/gfServer until recently. The .nib format only packed 2 bases per byte, and could only handle one record per nib file. Recent versions of blat and related programs can use .2bit files as well.

CREATING IN-MEMORY INDICES WITH GFSERVER

The gfServer program creates an in-memory index of a nucleotide sequence database. The index can either be for translated or untranslated searches. Translated indexes enable protein-based blat queries and use approximately two bytes per unmasked base in the database. Untranslated indexes are used nucleotide-based blat queries as well as for In-silico PCR. An index for normal blat uses approximately 1/4 byte per base. For blat on smaller (primer-sized) queries or for In-silico PCR a more thorough index that requires 1/2 byte per base is recommended. The gfServer is memory intensive but typically doesn not require a lot of CPU power. Memory permitting multiple gfServers can be run on the same machine.

A typical installation might go:

```
ssh bigRamMachine
cd /data/genomes/twoBit
gfServer start bigRamMachine 17779 hg16.2bit &
gfServer -trans -mask start bigRamMachine 17778 hg16.2bit &
```

the -trans flag makes a translated index. It will take approximately 15 minutes to build an untranslated index, and 45 minutes to build a translate index. To build an untranslated index to be shared with In-silico PCR do

```
gfServer -stepSize=5 bigRamMachine 17779 hg16.2bit &
```

This index will be slightly more sensitive, noticeably so for small query sequences, with blat.

EDITING THE WEBBLAT.CFG FILE

The webBlat.cfg file tells the webBlat program where to look for gfServers and for sequence. The basic format of the .cfg file is line oriented with the first word of the line being a command. Blank lines and lines starting with # are ignored. The webBlat.cfg and webPcr.cfg files are similar. The webBlat.cfg commands are:

gfServer - defines host and port a (untranslated) gfServer is running on, the associated sequence directory, and the name of the database to display in the webPcr web page.
gfServerTrans - defines location of a translated server.
background - defines the background image if any to display on web page
company - defines company name to display on web page
tempDir - where to put temporary files. This path is relative to where the web

server executes CGI scripts. It is good to remove files that haven't been accessed for 24 hours from this directory periodically, via a cron job or similar mechanism.

The background and company commands are optional. The webBlat.cfg file must have at least one valid gfServer or gfServerTrans line, and a tempDir line. . Here is a webBlat.cfg file that you might find at a typical installation:

```
company Awesome Research Amalgamated
background /images/dnaPaper.jpg
gfServer bigRamMachine 17778 /data/genomes/2bit/hg16.2bit Human Genome
gfServer bigRamMachine 17779 /data/genomes/2bit/hg16.2bit Human Genome
gfServer mouseServer 17780 /data/genomes/2bit/mm4.2bit Mouse Genome
gfServer mouseServer 17781 /data/genomes/2bit/mm4.2bit Mouse Genome
tempDir ../trash
```

PUTTING WEBBLAT WHERE THE WEB SERVER CAN EXECUTE IT

The details of this step vary highly from web server to web server. On a typical Apache installation it might be:

```
ssh webServer
cd kent/webBlat
cp webBlat webBlat.cfg /usr/local/apache/cgi-bin
mkdir /usr/local/apache/trash
chmod 777 /usr/local/apache/trash
```

assuming that you've put the executable and config file in kent/webBlat. The program will create some files in the trash directory. It is good to periodically clean out old files from this directory. On Mac OS-X instead you might do:

```
cp webBlat webBlat.cfg /Library/WebServer/CGI-Executables
mkdir /Library/WebServer/trash
chmod 777 /Library/WebServer/trash
```

Unless you are administering your own computer you will likely need to ask your local system administrators for help with this part of the webBlat installation.

File Formats

.psl files

A .psl file describes a series of alignments in a dense easily parsed text format. It begins with a five line header which describes each field. Following this is one line for each alignment with a tab between each field. The fields are describe below in a format suitable for many relational databases.

```
    matches int unsigned ,            # Number of bases that match that aren't repeats
    misMatches int unsigned ,         # Number of bases that don't match
    repMatches int unsigned ,         # Number of bases that match but are part of repeats
    nCount int unsigned ,             # Number of 'N' bases
    qNumInsert int unsigned ,         # Number of inserts in query
    qBaseInsert int unsigned ,        # Number of bases inserted in query
    tNumInsert int unsigned ,         # Number of inserts in target
    tBaseInsert int unsigned ,        # Number of bases inserted in target
    strand char(2) ,                  # + or - for query strand, optionally followed by + or - for
                                        target strand
    qName varchar(255) ,              # Query sequence name
    qSize int unsigned ,              # Query sequence size
    qStart int unsigned ,             # Alignment start position in query
    qEnd int unsigned ,               # Alignment end position in query
    tName varchar(255) ,              # Target sequence name
    tSize int unsigned ,              # Target sequence size
    tStart int unsigned ,             # Alignment start position in target
    tEnd int unsigned ,               # Alignment end position in target
```

```
    blockCount int unsigned ,        # Number of blocks in alignment
    blockSizes longblob ,            # Size of each block in a comma separated list
    qStarts longblob ,               # Start of each block in query in a comma separated list
    tStarts longblob ,               # Start of each block in target in a comma separated list
```

Currently the program does not distinguish between matches and repMatches. repMatches is always zero.
There is a little gotcha in the .psl format. It has to do with how coordinates are handled on the negative strand. In the qStart/qEnd fields the coordinates are where it matches from the point of view of the forward strand (even when the match is on the reverse strand). However on the qStarts[] list, the coordinates are reversed.

Here's an example of a 30-mer that has 2 blocks that align on the minus strand and 2 blocks on the plus strand (this sort of stuff happens in real life in response to assembly errors sometimes).

```
0         1         2         3 tens position in query
012345678901234567890123456789 ones position in query
            ++++          +++++ plus strand alignment on query
    --------    ----------      minus strand alignment on query
```

Plus strand:

```
qStart 12 qEnd 31 blockSizes 4,5 qStarts 12,26
```

Minus strand:

```
qStart 4 qEnd 26 blockSizes 10,8 qStarts 5,19
```

Essentially the minus strand blockSizes and qStarts are what you would get if you reverse complemented the query.However the qStart and qEnd are non-reversed. To get from one to the other:

```
qStart = qSize - revQEnd
qEnd = qSize - revQStart
```

.2bit files

A .2bit file can store multiple DNA sequence (up to 4 gig total) in a compact randomly accessible format. The two bit files contain masking information as well as the DNA itself. The file begins with a 16 byte header containing the following fields:

1) signature – the number 0x1A412743 in the architecture of the machine that created the file.
2) version – zero for now. Readers should abort if they see a version number higher than 0.
3) sequenceCount – the number of sequences in the file
4) reserved – always zero for now.

All fields are 32 bits unless noted. If the signature value is not as given, the reader program should byte swap the signature and see if the swapped version matches. If so all multiple-byte entities in the file will need to be byte-swapped. This enables these binary files to be used unchanged on different architectures.

The header is followed by a file index. There is one entry in the index for each sequence. Each index entry contains three fields:

1) nameSize – a byte containing the length of the name field
2) name – this contains the sequence name itself, and is variable length depending on nameSize.
3) offset – 32 bit offset of the sequence data relative to the start of the file

The index is followed by the sequence records. These contain 9 fields:

1) dnaSize – number of bases of DNA in the sequence.
2) nBlockCount – the number of blocks of N's in the file (representing unknown sequence).
3) nBlockStarts – a starting position for each block of N's
4) nBlockSizes – the size of each block of N's
5) maskBlockCount – the number of masked (lower case) blocks
6) maskBlockStarts – starting position for each masked block
7) maskBlockSizes – the size of each masked block
8) packedDna – the dna packed to two bits per base as so: 00 – T, 01 – C, 10 – A, 11 – G. The first base is in the most significant 2 bits byte, and the last base in the least significant 2 bits, so that the sequence TCAG would be represented as 00011011. The packedDna field will be padded with 0 bits as necessary so that it takes an even multiple of 32 bit in the file, as this improves i/o performance on some machines.

.nib files

A .nib file describes a DNA sequence packing two bases into each byte. Each nib file contains only a single sequence. A nib file begins with a 32 bit signature which is 0x6BE93D3A in the architecture of the machine that created the file, and

possibly a byte-swapped version of the same number on another machine. This is followed by a 32 bit number in the same format which describes the number of bases in the file. This is followed by the bases themselves packed two bases to the byte. The first base is packed in the high order 4 bits (nibble), the second base in the low order four bits. In C code:

```
byte = (base1<<4) + base2
```

The numerical values for the bases are:

```
0 - T,  1 - C,  2 - A,  3 - G,  4 - N (unknown)
```

The most significant bit in a nibble is set if the base is masked.

Limits

The gfServer program requires approximately 1 byte for every 3 bases in the genome it is indexing in DNA mode, and 1.5 bytes for each unmasked base in translated mode. The blat program requires approximately two bytes for each base in the genome in DNA mode, and three bytes for each base in translated mode. The other programs use relatively little memory.