



Alternative Splicing Array Design
www.AlexaPlatform.org

Pipeline Manual (v.11)

Malachi Griffith

13 August 2007



Table of Contents

Table of Contents	2
Authors.....	4
Acknowledgements.....	4
Affiliations.....	4
Introduction.....	4
Virtual Machine	4
Before Starting.....	4
Warnings.....	5
Installation of ALEXA.....	5
Script and Module Locations.....	5
Getting started – Creating a custom array from a pre-computed design	6
1.) Download the MYSQL tables from our FTP server.....	6
2.) Log into MySQL and create the target database	6
3.) Create a working directory and unpack the MYSQL tables.....	6
4.) Import the tables to the empty MYSQL database created above	6
5.) Create the design files.....	6
Creating a new design - Design Steps.....	7
Database Population and RepeatMasking.....	7
1.) Create a working directory for the new design.....	7
2.) Create database from the schema definition document:.....	7
3.) Database population	7
4.) Gene statistics summaries.....	8
5.) RepeatMasking.....	8
6.) Parsing RepeatMasked genes	9
Probe Sequence Extraction	9
7.) Extract exon junction probes	10
8.) Extract exon boundary (exon-intron junction) probes.....	11
9.) Extract exon internal probes.....	11
10.) Extract intron probes	11
11.) Generate random probe sequences to use as negative controls.....	12
Probe Quality Testing.....	12
12.) Test probes for folding potential	12
A.) Internal probe folding and self-self folding.....	13
13.) Check all probes for simple repeats/low complexity regions.....	13
14.) Testing specificity of probes.....	13
A.) Probe specificity versus a database of ESTs or mRNAs	14
B.) Probe specificity versus all Ensembl transcripts.....	16
C.) Probe specificity within probes.....	17
D.) Probe specificity within the target genomic sequence of each gene (exons + introns) ..	17
E.) Probe specificity within the entire genome sequence (chromosome contigs)	18
15.) Exon skipping determination.....	18
16.) Summarize gene probe coverage.....	18
17.) Create summary of all probe statistics for all raw probes	19
18.) Get chromosome coordinates for all probes	19
Probe Filtering	19
20.) Filter probes based on scores.....	19
21.) Create a final summary of all filtered probes	20
Importing Probes to the Database.....	20
22.) Populate database with probe info.....	20

23.) Define filtered probes as a probe 'set' in the database	21
24.) Backing-up/Restoring an ALEXA database.....	21
Creating and Visualizing a Custom Array	21
25.) Gene selection.....	21
26.) Select probes and generate Design Submission file	21
27.) Create UCSC tracks to visualize probe design.....	22
28.) Populate database with microarray design info	22
References.....	23

Authors

Written by Malachi Griffith (malachig@bcgsc.ca)

Acknowledgements

We thank J. Connors, N. Farnoud, J. Khattra, A. Petrescu and T. Pugh for input throughout this project. We are grateful for funding provided by the University of British Columbia Faculty of Graduate Studies and Faculty of Medicine, the Michael Smith Foundation for Health Research, the Natural Sciences and Engineering Research Council, Genome British Columbia, the Terry Fox Foundation, the National Cancer Institute of Canada and the BC Cancer Foundation.

Affiliations

Malachi Griffith is supervised by Marco A. Marra.
Canada's Michael Smith Genome Sciences Centre
British Columbia Cancer Agency
University of British Columbia - Faculty of Medicine – Department of Medical Genetics

Introduction

The following sections describe the complete process required to create a custom microarray design for the study of alternative splicing. The resulting 'splicing microarray' design consists of probes selected to interrogate the expression of specific exons as well as known and novel splicing events such as exon-skipping and alternate 5' or 3' splice site usage. The design process starts with the initial population of an Alternative Expression Analysis database (ALEXA) with gene annotation data. This database is a simple relational database implemented in MySQL. Genes and exon annotations are taken from Ensembl, used to extract probes of a variety of types (exon, exon-junction, exon-boundary, and intron) and filtered and scored by a number of algorithms. Finally, the choice of target genes and submission of design specifications to an array manufacturer is described. You may want to use this software to generate a new design if there is no pre-generated ALEXA database for your species of interest or if you wish to generate a design for an update of Ensembl for a particular species. Check the ALEXA website for more details on the designs already available for download before proceeding (www.AlexaPlatform.org).

Virtual Machine

The next few sections describe how to install the ALEXA platform. If you decide to download and use the VMware Virtual Machine appliance you can skip directly to the 'Design Steps' section. The virtual machine can be 'played' on Mac, Windows or Linux and consists of all necessary tools installed into a Linux Fedora7 environment. Before downloading the ALEXA Virtual Machine, you should download and install the free VMware player from: <http://www.vmware.com/download/player/>

To use the ALEXA virtual machine you will need the following username and password information. The login username is 'alexa' and password is 'alexa777'. If you need root access, the password is 'alexa_admin'. Finally, for access to the MySQL databases you can use the user 'alexa' (password: 'alexa_mysql_admin') or 'viewer' (password: 'viewer'). Please refer to the 'Platform' section of the ALEXA website for more details (www.AlexaPlatform.org). Once you log in simply go to the /home/alexa directory to get started.

Before Starting

The scripts making up the pipeline described in this directory can be run from where they are unpacked but is preferable to create a separate working directory for all output files and log files. It is also a good idea to create a separate directory for every species or Ensembl version you intend to use. It should

also be noted that this code is meant to run in a unix/linux environment. Before starting you should ensure that you have the following installed on your system (not necessary if you are using the virtual machine):

- Perl 5.8
- Perl DBI/DBD libraries.
- mySQL. You will need access to a mySQL database server with an account that allows you to create databases. The platform has been tested with mySQL 4.1 and 5.0.
- A working installation of 'R' (<http://cran.r-project.org/>). Any version should be fine.

Throughout this manual I will also provide instructions for the installation of other necessary tools and data sources. Specifically you will need the following (not necessary if you are using the virtual machine):

- Ensembl Perl API (version will depend on what version of databases you want to access)
 - o Specifically 'ensembl', 'ensembl-external', and 'ensembl-compara'
- BioPerl 1.2 or greater
- RepeatMasker and Repeat libraries for your species of interest
- Blast
- Mdust
- RNAssoft
- Various publicly available sequence databases

Warnings

You will require at least a basic familiarity with Perl and Unix/Linux to create your own array designs with this pipeline. Furthermore, certain aspects of this pipeline are extremely computationally intensive, particularly folding probe sequences and testing their specificity. I have included scripts to assist in the creation of parallel jobs to be run on a cluster of computers. If you do not have access to such computer resources it may not be practical to create designs for species with large a large number of genes and exons such as human. This may not be an issue for some species or if you are willing to focus on a subset of genes from the outset of the design process. To give you an idea, I created the *C. elegans* design in about 1 week using the Virtual Machine running on Windows XP host with 2 Gb of RAM and dual 2.2 GHz processors.

Installation of ALEXA

- Before proceeding, download and unpack the ALEXA code base as follows (not necessary if you are using the virtual machine).
 - o `mkdir /home/user/alexa`
 - o `cp alexa_v1.0.tar /home/user/alexa`
 - o `cd /home/user/alexa`
 - o `tar -xvf alexa_v1.0.tar`

Script and Module Locations

- The root/reference directory for all scripts is:
~/Array_design (where ~ is wherever the code was unpacked)
- Many of the scripts described below make use of methods that I have written and stored in utility modules. The details of these functions are beyond the scope of this document. Nevertheless, these perl modules follow POD (plain old documentation) format and are stored in:
~/Array_design/utilities
- Each script is written in such a way that you should be able to run from the installed location or by specifying the full path to the script. This will only work as long as the /utilities folder stays in the same directory as the scripts.

- The names of scripts and binaries described below are highlighted as blue text
- Each script can be run without arguments to get a list of instructions. Some scripts also have a corresponding ‘Perldoc’ in .html format
- The output of each script should be stored somewhere other than the location of the scripts themselves. For example you can create a target directory for each species/ensembl version combination (i.e. /home/user/alexa/ALEXA_version).

Getting started – Creating a custom array from a pre-computed design

If you would like to create a completely novel array design, skip ahead to ‘Creating a new design’. However, if you simply want to create a custom array for one of the species which has a pre-computed design, you can use the following instructions to get started. This is the simplest use of the ALEXA platform. Once you have installed the virtual machine or source code, download the MYSQL tables corresponding to the design of interest from our FTP server (<ftp2.bcgsc.ca/ALEXA>), install them, and create a custom array as follows (using ALEXA_cf_42_2 as an example).

1.) Download the MYSQL tables from our FTP server

- ftp
- open <ftp2.bcgsc.ca> (user: anonymous)
- get ALEXA_cf_42_2.tables.tar.gz
- exit

2.) Log into MySQL and create the target database

- mysql
- CREATE DATABASE ALEXA_cf_42_2
- exit

3.) Create a working directory and unpack the MYSQL tables

- cp -r /home/alexa/ALEXA_template /home/alexa/ALEXA_cf_42_2
- mv ALEXA_cf_42_2.tables.tar.gz ALEXA_cf_42_2/database_backup
- cd ALEXA_cf_42_2/database_backup
- gunzip ALEXA_cf_42_2.tables.tar.gz
- tar -xvf ALEXA_cf_42_2.tables.tar

4.) Import the tables to the empty MYSQL database created above

- ~/sql/[restoreAlexaDb.pl](#)

5.) Create the design files

- Create a design submission file and accompanying annotation and probe record files with the following script:
 - ~/Array_design/[createDesignSubmissionFile.pl](#)

Creating a new design - Design Steps

Database Population and RepeatMasking

1.) Create a working directory for the new design

- First create a working directory where all design related files will be stored.
- A template of this directory structure comes with the ALEXA code.
- Unpack ALEXA_template.tar
 - `'cp ALEXA_template.tar /home/user/alexa'`
 - `'cd /home/user/alexa'`
 - `'tar -xvf ALEXA_template.tar'`
- Rename this directory to match the design you are about to create
 - e.g. `'mv ALEXA_template ALEXA_hs_41_36c'`

2.) Create database from the schema definition document:

- `~/sql/ALEXA_schema.sql`
- This document contains all the necessary 'create table' statements as well as descriptions of many of the tables and fields
 - a) First log into your mySQL server and execute the following commands at the mySQL prompt
 - e.g. `/usr/bin/mysql -h server_name -u username`
 - b) Create an empty database with a suitable name:
 - `CREATE DATABASE ALEXA_hs_41_36c`
 - c) Now make this database active:
 - `USE ALEXA_hs_41_36c`
 - d) Now create the empty tables from the schema:
 - `SOURCE ~/sql/ALEXA_schema.sql`
 - e) You should now have a complete but empty database with several tables
 - f) Try 'SHOW TABLES' to make sure it worked

3.) Database population

- All basic gene data comes from Ensembl (Hubbard *et al.* 2005)
- Download updated Ensembl API
 - a) Create a new directory for the new API.
e.g. `/home/user/perl/ensembl_41_perl_API`
- then move to that new directory
 - b) Download the desired branch from ensembl cvs repository
 - `cvs -d :pserver:cvsuser@cvsro.sanger.ac.uk:/cvsroot/CVSmaster login`
 - `password: CVSUSER`

 - `cvs -d :pserver:cvsuser@cvsro.sanger.ac.uk:/cvsroot/CVSmaster checkout -r branch-ensembl-41 ensembl`

 - `cvs -d :pserver:cvsuser@cvsro.sanger.ac.uk:/cvsroot/CVSmaster checkout -r branch-ensembl-41 ensembl-external`

 - `cvs -d :pserver:cvsuser@cvsro.sanger.ac.uk:/cvsroot/CVSmaster checkout -r branch-ensembl-41 ensembl-compara`

- `cvs -d :pserver:cvsuser@cvsro.sanger.ac.uk:/cvsroot/CVSmaster
logout`

- Use the script [getEnsemblGeneData.pl](#) to populate the database, ALEXA.
- If the EnSEMBL API has been updated, do not forget to change update the script. You will also have to specify the location of your bioPerl installation within this script.
- This script prints out various statements to log the data retrieval and population process. This information is saved as a log and stored. For example:
`/home/user/alexa/ALEXA_version/logs/database_population/ALEXA_population_2005Jun22.log`
- You should check the mySQL database at a mySQL prompt to ensure that the database population worked correctly. Try some simple SQL queries to see if basic gene, exon and transcript data has been imported.
- As an example, this pipeline was used to generate probes based on EnSEMBL human version **41_36c**. This build contains: 31185 genes, 56364 transcripts, 284579 exons and 507997 transcript-exon relationships.
- Note that many tables in the database will still be empty at this stage.

4.) Gene statistics summaries

- Summarize basic information about the protein coding genes for which the probes will be designed.
- First use:
`displayGeneStats.pl --summarize`
- This will generate files containing the number of transcripts per gene, the sizes of each transcript, the number of exons per transcript and the sizes of each exon.
- These files can be stored in a user specified directory such as:
`/home/user/alexa/ALEXA_version/stats/genes`
- The script can also be used to summarize a single gene by using the options:
`--alexa_id` or `--ensembl_id` or all genes using the option: `--all_genes`
- Also use [summarizeEnsemblExonSkipping.pl](#) to identify all exon skipping events and how many exons were skipped in each event. Once again place this file in an output stats directory
- Once these five output files have been generated you can then use 'R' to generate simple statistics and figures. Example 'R' code for this purpose can be found in:
`~/R_bin/gene_stats.R`
- Go to the output directory containing these files and try the following command:
`/usr/bin/R --no-save < ~/R_bin/gene_stats.R > gene_stats_LOG.txt`

5.) RepeatMasking

- Reference: Smit, AFA, Hubley, R & Green, P. *RepeatMasker Open-3.0*. 1996-2004
- To ensure the most up-to-date repeatMasking is accomplished I install the newest version of RepeatMasker (Version 3.1.6 at the time of writing) before proceeding. You can download this from: www.repeatmasker.org
 - At this time you may wish to update your wuBlast or cross_match installation as RepeatMasker uses blast for pattern matching. You can get the newest version of wuBLAST for your system type from here: (<http://blast.wustl.edu/>). I use cross_match.
- To get repeat libraries themselves you will need to register with www.girinst.org and get a working account to do this (free for academics).
- At the time of writing these libraries were: RepBase Update 11.11, RM database version 20061006. To get the newest version log into:
`http://www.girinst.org/server/RepBase/repeatmaskerlibraries/`
User: username

pass: password

- Follow the online instructions for installing RepeatMasker and associated repeat libraries.
- The script [maskEnsemblGenes.pl](#) is used to break the ensembl genes into a series of fasta files which are used to specify batch jobs on a cluster. This will take a long time if you do not have access to significant computer resources...
- This creates a series of masked output files which are then parsed and the masked sequences are stored in the ALEXA database for convenience.
- The input fasta files containing unmasked gene sequences will be stored in:
/home/user/alexa/ALEXA_version/repeat_masking/ensembl_genes_fasta
- The resulting fasta files containing masked gene sequences as well as summary files will be temporarily stored in:
/home/user/alexa/ALEXA_version/repeat_masking/ensembl_genes_masked
- Sanity check: Use a 'grep' command to make sure a fasta record was created for every gene (i.e. `grep ">" * | wc -l`)
- The repeatMasking script also creates the necessary shell commands to be submitted to the cluster. To actually run this job, first check that this file is accurate. Then login to a cluster head node and submit the job. On our system this looks like this:
[mqsub.py](#) --file \$batch_file.sh --name \$job_name --mkdir
- This process will differ depending on your cluster but the batch file is very simple and can be run directly from a single computer if necessary (but this will take a long time to complete for large genomes).
- After repeatmasking is complete, check for the correct number of masked files. If no repeats are found for a file of input sequences, an error is thrown and no masked file is created. In that case you can simply copy over the unmasked file into the masked directory...
- Note: If you are unsure what species name to use with RepeatMasker, check the documentation for examples or use the full genus species name (e.g. Homo sapiens), or use the following RepeatMasker tool stored in the RepeatMasker installation directory: "util/queryRepeatDatabase.pl -species 'mouse' -stat" to test a species name.

6.) Parsing RepeatMasked genes

- The script [parseMaskedGenes.pl](#) is used to parse through the masked files produced in the previous step and dump this information to ALEXA
- Run this script without options for detailed instructions. Run the script once with the following option to test the masked files:
--update_database=no
- If this test is successful, run the script again and import masked sequences to the database.
- Masked gene sequences are stored separately in the 'MaskedGene' table for performance reasons
- The script will report the % of bases that are masked. Make note of this number and ensure that it is reasonable (e.g. 42% of human sequence content was masked for ALEXA_hs_41_36c).

Probe Sequence Extraction

- The following steps will extract probes corresponding to exon-exon junctions, exon-intron boundaries, within exons, and within introns.
- Each step will generate a probe file containing basic info about each probe and a log file containing a record of the design process.
- These probe files should be stored in ~/ALEXA_version/unfiltered_probes
- Important notes:
 - Only 'Known Genes' are considered for the following steps. If you wish to design probes for Ensembl predicted genes use the '--allow_predicted_genes=yes' option.

- Similarly, probes are not extracted for pseudogenes.
- A 'probeset' is a group of probes which interrogates a particular target sequence. For example, an 'exon-junction probeset' consists of probes which target a single junction of two exons. In this case, multiple probes will be essentially the same sequence but have slightly different lengths. By contrast, an 'exon probeset' will consist of probes that target a single exon (or part of an exon). Since exons can be quite large, these probes will often have completely different sequences. Every probe within a design will have a unique `probe_id` and a `probeset_id` corresponding to 1 or more probes targeting a particular sequence. A 'probeset' thus represents a group of probes whose expression values can be averaged or otherwise combined during the analysis.

7.) Extract exon junction probes

- The first step before generating probes of the various types described below is to determine the target probe T_m for the design. To determine the target T_m , first generate an anisothermal (fixed length) set of exon junction probes with the desired length (35-40 bp is recommended) using the following command.


```
generate_ExonJunctionProbes.pl --database=database_name --server=server_name
--user=username --password=pwd --target_tm=67.0 --target_length=36
--max_length_variance=0 --probes_per_junction=1 --all_genes=yes
--probe_dir=/home/user/alexa/ALEXA_version/unfiltered_probes
--logfile=/home/user/alexa/ALEXA_version/logs/generate_ExonJunctionProbes_LOG.txt
--outfile=/home/user/alexa/ALEXA_version/unfiltered_probes/exonJunctionProbes.txt
```
- Note: A Perl implementation of the 'Nearest Neighbour' method (Breslauer, *et al.*, 1986; Sugimoto *et al.* 1996) is used to calculate the T_m of each probe from its dinucleotide content.
- Since the `max_length_variance` was set to '0', only probes of exactly the target length will be generated. The value specified as the `target_tm` is thus irrelevant as the length will not be varied to achieve the target T_m .
- The file generated by this script contains the T_m of each probe. Using this column of values calculate the median T_m of all of these probes and use this median T_m as the target T_m for the entire design. Use [determineProbeTmStats.pl](#) to calculate the median T_m for all junction probes, then if you wish to generate an isothermal probe design, regenerate the junction probes with this median T_m as a target.
- It should be noted that the target t_m you chose should be tailored to the hybridization conditions you ultimately plan to use. In our pilot experiments we used NimbleGen Chip-chip hybridization conditions (designed for ~50 mers originally) with a probe design with a median T_m of 67°C. If you plan to follow the same experimental design chose a length of probe that gives T_m 's close to this.
- Once you have decided on a target T_m , rerun this script to generate an isothermal design. For example, I typically allow the length of each probe to vary +/- 10 bp to achieve the target length. I also generate two probes for each junction (the two probe lengths that result in a T_m closest to the target T_m will be selected).


```
generate_ExonJunctionProbes.pl --database=database_name --server=server_name
--user=username --password=pwd --target_tm=67.0 --target_length=36
--max_length_variance=10 --probes_per_junction=3 --all_genes=yes
--outfile=exonJunctionProbes.txt --logfile=./logs/generate_exonJunctionProbes_LOG.txt
```
- Note: Probes that are derived from genomic sequence containing N's or contain too many repeatMasked bases (more than 1/4 of the probe sequence) are eliminated. Also each exon used in a combination must be long enough so that the probe does not exceed its boundaries.
- The following steps describe how to generate exon boundary probes, exon probes, and intron probes.

In each case the style of command is similar as that described above for exon junction probes.

8.) Extract exon boundary (exon-intron junction) probes

- The script will parse the user specified probe directory to determine the current max probe and probeset IDs and continue from this point. All probe generation scripts do this.

[generate_ExonBoundaryProbes.pl](#) (use options similar to those for exon-junction probes)

9.) Extract exon internal probes

- The script to extract probes from within exons will attempt to extract probes from each exon region and will attempt to use the most informative regions for overlapping exons.
- Because of the incredible number of possibilities for probe sequences from within all exons of a gene, some initial filters or quality checks are applied immediately. Specifically, the user must specify the target Tm and an acceptable range or probe lengths to achieve this Tm.
- The user must also specify the tiling distance between probes to use when extracting the sequences e.g. '--overlap=5' will extract a probe sequence every 5 bp.
- As in the extraction of junction probes, this script will also do basic checks on the length of exons, presence of unknown genomic bases, and RepeatMasked bases.
- Example command:

```
generate_ExonProbes.pl --database=database_name --server=server_name --user=username
--password=pwd --overlap=5 --target_tm=67.0 --tm_range=3.0 --target_length=36
--max_length_variance=8 --all_genes=yes --verbose=no
--probe_dir=/home/user/alexa/ALEXA_version/unfiltered_probes/
--outfile=/home/user/alexa/ALEXA_version/unfiltered_probes/exonProbes.txt
--logfile=/home/user/alexa/ALEXA_version/logs/generate_exonProbes_LOG.txt
```

- This script will calculate the Tm of each probe as they are extracted. The output file is in the same format as that of the junction probes produced above.
- Note that at this time, many probes are selected for each exon region. Since we have choices as to where we place the probes we can select more probes than we need and then chose the 'best' probes after all quality checks are complete.

10.) Extract intron probes

- If you wish to design probes for introns to act as negative controls or to interrogate intron retention events you can use the following script. I typically only design intron probes for a set of ~100 housekeeping genes to allow a test of sensitivity and specificity for genes that are expected to be expressed and processed by pre-mRNA splicing.
- Note that for this probe type you can generate probes for all genes of provide a list of genes in a file (such as a list of housekeeping genes).
- If you wish to use intron probes for only a small number of housekeeping genes, you can use the list provided for each species (e.g. human_housekeeping_genes.txt). This list is based on the housekeeping genes used as controls on the Affymetrix Exon Array design. For species other than human, orthologs to these genes are used. If your species does not have a pre-generated list, you can use the human list to generate a new set of orthologs for your species using the script:
[getEnsemblOrthologs.pl](#)
- For some species, selecting the --all_genes option will generate an extremely large number of probes! The exact number will depend on the selected --overlap and other settings.
- If you specify a gene file, it must be tab-delimited with a header line and EnSEMBL gene IDs in the first column.

[generate_IntronProbes.pl](#) (use options similar to those for exon probes)

11.) Generate random probe sequences to use as negative controls

- Determine the T_m range of all experimental probes generated thus far or select a cutoff that will be used to filter/fail probes. For example we might choose to allow only probes that are within the target_tm +/- 4 °C.
- The purpose of this script is to randomly generate probes of a range of lengths within the specified T_m range such that a specified number are selected within every 0.1 °C block within this T_m range for each possible length. The result is a set of probes that uniformly represent the range of T_m and length for all experimental probes but since the sequences are random they will be unlikely to hybridize to a target. They will be tested against known sequences to further ensure that this is true.
- The purpose of these probes is thus to act as an estimate of the true background signal intensity of probes for which no hybridization to any sequence is expected. This should provide an estimate of the behaviour of probes in the absence of specific hybridization.

```
generate_NegativeControlProbes.pl --probe_length=36 --target_tm=67.0 --tm_range=3.0
--bin_size=1000 --max_length_variance=10
--output_file=/home/user/alexa/ALEXA_version/unfiltered_probes/negativeControlProbes.txt
--logfile=/home/user/alexa/ALEXA_version/logs/generate_negativeControlProbes_LOG.txt
```

- You should specify a range of T_m's that you plan to use for filtering. Decide what you will consider a failing T_m. You can then generate negative control probes to match this or exceed it slightly.
- Note: if you specify a wide T_m range, this will take a long time to complete.
- The output of this script is a probe file in the usual format. This file can be used to generate probe scores for probe folding, complexity, and specificity as described above.
- These probes will be blasted against ESTs, mRNAs, Ensembl transcripts, genomic sequence, and experimental probe sequences to make sure they have no similarity to any known sequences.

Probe Quality Testing

Note: Some of the following steps may be specific to junction probes or exon-internal probes. For this reason they are processed separately at most steps. Also, the order of these tests does not matter.

12.) Test probes for folding potential

- This test should be conducted for all probe types. You will have to go through this process for all probe files in the unfiltered probe directory.
- Folding potential is tested using the programs PairFold and Simfold, which are part of the RNASoft package (Andronescu, *et al.* 2003). You will need to download and install this tool from:
<http://www.rnasoft.ca/>
- I use version: MultiRNAFold-1.1
- Both of the following steps use the same wrapper for creating a cluster job:

[createPairFoldBatch.pl](#)

- The cluster job will run instances of two other scripts depending on the type of job specified as described below. In both cases a master probe file is divided into smaller pieces, processed and the

results are written to new files containing the probe_id in the first column followed by the simfold/pairfold scores.

- To create a master probe file of all of these smaller files and join it to the original probe file use the script:

[joinProbeFiles.pl](#)

- This script performs basic sanity checks on probe records, merges them (ordered by probe count ID) and creates an appended probe file.
- Once you have checked this appended file, rename it and delete the original one.

A.) Internal probe folding and self-self folding

The purpose of this step is to test each probe's tendency to fold on itself (internal folding) or to form a secondary structure between two copies of itself (self-self folding). These two types of folding will be tested with simFold and pairFold respectively.

- Create a batch job using the script [createPairFoldBatch.pl](#) and use the '--folding_bin' option to specify the full path to the script used to parse the results: [get_SimFold-PairFoldScores.pl](#)

Note: I have had some problems running these scripts on the cluster. I have been forced to copy the pairfold and simfold binaries and the parameters files to /tmp/ on each node. This 'staging' is done to reduce the load on the filer and network caused by so many system calls. Make sure you check the output files and error files for the cluster job very carefully after this step. If some jobs fail you can create a new batch file for only those probe files that did not result in a results file using the following option with [createPairFoldBatch.pl](#): '--repair=yes'.

13.) Check all probes for simple repeats/low complexity regions

- This test will be conducted for all probe types.
- Use a perl wrapper for 'mdust' to do this (www.tigr.org; Hancock and Armstrong, 1994). This script creates an appended probes file by adding the number of masked low complexity bases as the last column. The -v flag is used to specify the mdust threshold (smaller numbers mask more).

```
test_ProbeComplexity.pl --mdust_bin=/home/user/BioSw/dust/mdust/mdust
--temp_dir=/home/user/alexa/ALEXA_version/mdust
--probe_dir=/home/user/alexa/ALEXA_version/unfiltered_probes
--logfile=/home/user/alexa/ALEXA_version/logs/test_probeComplexity_LOG.txt
```

- Check that the files generated seem okay and if so delete the original probe file without mdust scores appended
- Mdust can be downloaded from <http://compbio.dfci.harvard.edu/tgi/software>

14.) Testing specificity of probes

- The following steps describe how to test the specificity of probe sequences against various databases of known sequence. Five possible tests are considered for each probe type: all known ESTs, all known mRNAs, all Ensembl transcripts (including predicted transcripts), gene sequences with introns present, the entire genome, and the complete set of probes in your design. Some of these tests do not apply or are not practical apply to particular probe types. A table describing which tests to apply to which probe types is provided.

- Note: In the following scripts, you will be asked to specify the location of a BLAST binary. You can use blastall or megablast. In either case, you must specify a suitable 'wordsize' and this may depend on the length of probes you have designed. If you use megablast then the '-w' or 'wordsize' option should be a multiple of 4. This guarantees that you will find matches of '-w'+3 in length or longer (sometimes matches as short as '-w' will be found but it is not guaranteed). For example, '-w = 12' will find all matches of 15 or more. I typically use 'blastall' with a 'wordsize' or 11.

Probe Type	BLAST Database	*Treat as negative control?
Exon Junction (Exon-Exon)	mRNA Ensembl transcripts Probes (all experimental probes) Within gene	No No No (hit to self allowed) N/A
Exon boundary (Exon-Intron)	mRNA Ensembl transcripts Probes (all experimental probes) Within gene	No No No (hit to self allowed) N/A
Exon (within exon)	mRNA Ensembl transcripts Probes (all experimental probes) Within gene	No No No (hit to self allowed) N/A
Intron (within intron)	mRNA EST Ensembl transcripts Probes (all experimental probes) Within gene	Yes Yes Yes No N/A
Negative Control (random)	mRNA EST Ensembl transcripts Probes (all experimental probes) Whole genome	Yes Yes Yes Yes Yes

*If 'yes', use --probe_parse_bin=/home/user/alexa/Array_design/testProbeSpecificity_Negative.pl

*If 'no', use --probe_parse_bin = one of the following:

- testProbeSpecificity_EST_mRNA.pl
- testProbeSpecificity_Ensembl.pl
- testProbeSpecificity_Probes.pl

A.) Probe specificity versus a database of ESTs or mRNAs

- Get ESTs and mRNAs for a particular species (human in this example) from UCSC. Make sure the genome version you get matches that used in your original population of ALEXA! For example, Ensembl databases that use human genome version 35 correspond to UCSC human genome version hg17 (similarly NCBI Build 36 = hg18)
 - A table of which UCSC releases correspond to which external genome builds can be found here: <http://genome.ucsc.edu/FAQ/FAQreleases>
- Connect via FTP or HTTP to hgdownload.cse.ucsc.edu (FTP recommended for large files)
 - <http://hgdownload.cse.ucsc.edu/goldenPath/hg17/bigZips/mrna.fa.gz>
 - <http://hgdownload.cse.ucsc.edu/goldenPath/hg17/bigZips/est.fa.gz>

- Login to the FTP server, go to the correct directory and then use ‘mget mrna.fa.gz est.fa.gz’
- Store these files in a suitable location:
e.g. /home/user/alexa/ALEXA_version/data_sources/ucsc
- Create a blastable database for each of these files. Make sure you use the formatdb binary from the same installation of blast that you will use for the subsequent blast analysis

ex. /usr/local/blast/blast2.2.15/bin/formatdb -i mrna.fa -t human_mrna -p F -o T -n human_mrna

- Get UCSC tables from the FTP site containing information about where each EST/mRNA aligns to the genome:

http://hgdownload.cse.ucsc.edu/goldenPath/hg17/database/all_mrna.txt.gz

http://hgdownload.cse.ucsc.edu/goldenPath/hg17/database/all_est.txt.gz

- Login to the FTP server, navigate to the correct directory, then use the command:
 - ‘mget all_mrna.txt.gz all_est.txt.gz’
- Unzip these files and store them in a suitable location:
e.g. /home/user/alexa/ALEXA_version/data_sources/ucsc/sql

- Dump these into ALEXA. The tables should have been created already when you created your ALEXA database. Now we are simply going to populate them using the following command at a mysql prompt:

```
LOAD DATA LOCAL INFILE '/home/user/alexa/ALEXA_version/data_sources/ucsc/all_est.txt'
INTO TABLE all_est
```

- NOTE_1: If you have problems with the import of this table, it is possible that UCSC changed the table format. You can try to import the new table structure from UCSC as well and recreate it, ..., but this may break other functionality in the ALEXA pipeline.
- NOTE_2: At this point you should see if the chromosome names in the UCSC tables are compatible with those in the ALEXA/Ensembl gene tables. Unfortunately some species do not use standard annotation for chromosome names. If necessary update the getGeneInfo() function of [ALEXA_DB.pm](#) to deal with special cases using the examples present.
- Create batch files to do a blast of all probes in a master file on the cluster and then another cluster job to parse the results and create a new probes file containing summarized results. The following script will do this creating temp files in /home/user/alexa/ALEXA_version/blast.

[blastProbesVersus_Database.pl](#) (use --db_type=EST or --db_type=mRNA)

- This script will create two batch files: megablast.sh and specificity.sh which can be placed in the following directory:
/home/user/alexa/ALEXA_version/batch_scripts
- First submit the blast job to the cluster and then submit the actual specificity testing job. This second job simply calls the script:

[testProbeSpecificity_EST-mRNA.pl](#)

- This script contains a complete description of how it works but very briefly it looks for significant hits of a probe to a sequence which maps to a part of the genome other than the region of the

targeted gene. If this happens the probe is considered to be non-specific.

- The result of the two cluster jobs described will be a series of probe files stored in:
/home/user/alexa/ALEXA_version/specificity/job_name/parsed_results
- These files contain a summary of the specificity of each probe. Specifically this includes: the number of hits to sequences that map outside the targeted genomic region (ideally 0), the number of hits to sequences that map within the targeted genomic regions (ideally more than 0), the length of the longest hit to a sequence that maps outside the targeted region (ideally as small as possible), and finally the length of the longest hit to a sequence that maps within the targeted region (ideally the length of the probe).
- To create a master probe file of all of these smaller files use the script:

[joinProbeFiles.pl](#)

- This script performs basic sanity checks on probe records and then merges them (ordered by probe count ID). It is a good idea to check if the number of probes in this file matches those in the original input file for which the probe specificity was to be determined.
- Note: To do this specificity test for negative probes in which you wish to ensure there are **no hits** to any sequence at all use an alternate --probe_parse_bin option with the script (use [testProbeSpecificity_Negative.pl](#)) to modify the behaviour of 'blastProbesVersus_Database.pl'. This will parse the blast results and determine the number of hits by a probe to any sequence and record them as non-target hits. This alternate parsing script should be used for intron probes as well as random negative control probes.

B.) Probe specificity versus all EnsEMBL transcripts

- Use the following script to extract all EnsEMBL transcript sequences from ALEXA and create a fasta file from them.

[createEnsemblTranscriptFasta.pl](#)

- This fasta file is stored at:
/home/user/alexa/ALEXA_version/data_sources/ensembl_transcripts/alexa_transcripts.fa
- Use formatdb to create a blastable database from this file:
/usr/local/blast2.2.15/bin/formatdb -t alexa_transcripts -i alexa_transcripts.fa -p F
-o T -n alexa_transcripts

- Now create a cluster job to actually conduct blasts and the ensembl specificity test using:

[blastProbeVersus_Database.pl](#) (use --db_type=enst)

- This script functions in the same way as 'blastProbesVersus_EST-mRNA-Genome.pl' by dividing probe files in to smaller batches, creating fasta files and creating batch jobs for the cluster to perform megablast and the actual specificity test with the script:

[testProbeSpecificity_Ensembl.pl](#)

- Once again, to create a master probe file of all of these smaller files use joinProbeFiles.pl
- Note: To do this specificity test for negative probes in which you wish to ensure there are **no hits** to

any sequence at all use an alternate `--probe_parse_bin` option with the script (use [testProbeSpecificity_Negative.pl](#)) to modify the behaviour of `'blastProbesVersus_Database.pl'`. This will parse the blast results and determine the number of hits by a probe to any sequence and record them as non-target hits. This alternate parsing script should be used for intron probes as well as random negative control probes.

C.) Probe specificity within probes

- Make sure no two probes are identical or nearly identical to each other by using a blast of all probes against all probes. The only highly significant hits that result from this should be self-self.
- This should help to eliminate bad probes that are missed by the above specificity tests. The most specific example of this is the case where probes are identical or nearly identical because of duplicate elements within a gene. The previous specificity tests would not have identified these.
- Use the following script to generate a fasta file of all probe sequences (negative control probes will be ignored) using:

[createProbeFasta.pl](#)

- Store this file in:
/home/user/alexa/ALEXA_version/data_sources/probes_blast_database
- Create a blastable database using
/usr/local/blast2.2.15/bin/formatdb -t all_probes -i all_probes.fa -p F -o T -n all_probes
- Now create a cluster job to actually conduct blasts and the probe specificity test using:

[blastProbesVersus_Database.pl](#) (use `--db_type=probe`)

- This script functions in the same way as `'blastProbesVersus_EST-mRNA-Genome.pl'` by dividing probe files into smaller batches, creating fasta files and creating batch jobs for the cluster to perform blasts and the actual specificity test with the script:

[testProbeSpecificity_Probes.pl](#)

- For probe-probe matches, we are only concerned with matches over most of the length of the probe. There will be a large number of matches between exon junction probes because many of them legitimately share at least 50% of their sequence with each other.
- I recommend using `'blastall'` with a wordsize of 11 to ensure that no significant hits between short probes are missed. This may result in the creation of very large output files, so make sure you have lots of space to work with or process them in batches.
- Note: To do this specificity test for negative probes in which you wish to ensure there are **no hits** to any sequence at all use an alternate `--probe_parse_bin` option with the script (use [testProbeSpecificity_Negative.pl](#)) to modify the behaviour of `'blastProbesVersus_Database.pl'`. This will parse the blast results and determine the number of hits by a probe to any sequence and record them as non-target hits. This alternate parsing script should be used for random negative control probes but not intron probes as these will be included in the probes database and at least one hit is therefore expected.

D.) Probe specificity within the target genomic sequence of each gene (exons + introns)

- Test specificity of each probe within each gene. The purpose of this step is to identify probes that correspond to duplicated sequences within the gene being targeted by a probe. These events are rare

but do occur.

- The following script will test all of the probes for a target gene against the complete genomic sequence of that gene. This process will be repeated for all genes targeted by probes in the input file.

[blastProbesVersus_TargetGene.pl](#)

E.) Probe specificity within the entire genome sequence (chromosome contigs)

- Testing specificity against the human genome (one contig for each chromosome) is only necessary for negative control probes.
- Human genomic fasta sequence was downloaded from:
<ftp://hgdownload.cse.ucsc.edu/goldenpath/hg18/bigZips/chromFa.zip>

- A blastable database was created from this file as described above
- Create batch files to do a blast of all probes in a master file on the cluster and then another cluster job to parse the results and create a new probes file containing them. The following script will do this creating temp files in /home/user/alexa/ALEXA_version/blast.

[blastProbesVersus_Database.pl](#) (use --db_type=genomic)

- I recommend using ‘blastall’ with a wordsize of at least 18 for this test. Otherwise the effective database for a large genome is too large and will take too long to process.
- Note: To do this specificity test for negative probes in which you wish to ensure there are **no hits** to any sequence at all use an alternate --probe_parse_bin option with the script (use [testProbeSpecificity_Negative.pl](#)) to modify the behaviour of ‘[blastProbesVersus_Database.pl](#)’. This will parse the blast results and determine the number of hits by a probe to any sequence and record them as non-target hits. The ‘genomic’ option should be used in conjunction with ‘testProbeSpecificity_Negative.pl’ for negative control probes but NOT intron probes, as these are expected to hit at least one genomic region.

15.) **Exon skipping determination**

- It may not be practical to include probes for every possible exon-exon junction on a single or even a series of custom arrays. For this reason you may decide to choose a reasonable threshold such that probes designed to detect exon skipping events of greater than some number will be excluded from the array. The script [summarizeEnsemblExonSkipping.pl](#) described above considers all the exon skipping events in Ensembl and summarizes them to assist in the selection of this threshold.
- Use the following script to examine each exon junction probe extracted above and determine the number of exons skipped in the theoretical splicing event it is designed to detect. For each canonical splicing event, this value will be 0, however, for genes with many exons this number can become quite high as probes are designed to detect every possible valid exon-exon connection:

[getProbeExonSkipping.pl](#)

- This script will create a new probes file with the exon-skipping values appended as a new column. Note that this determination is only valid for exon-exon junction probes and will be ‘na’ for all other probes.

16.) **Summarize gene probe coverage**

- The following script can be used to summarize the coverage of genes by the probes designed. This can be used to get sense for how informative a probe set for a particular gene will be with regards to the alternative splice variants that are possible. The coverage of each gene can also be compared

before and after filtering to get a sense for how well the filtered probes will cover the possible AS forms for a gene.

[summarizeGeneProbeCoverage.pl](#) --skipped_exon_threshold=6

- Note: This script requires that the script '[getProbeExonSkipping.pl](#)' is run first. This allows it to consider the exon-skipping parameter when determining probe coverage. For example, if you do not wish to consider probes that interrogate splicing events involving the skipping of more than 6 exons, you can specify this:

[summarizeGeneProbeCoverage.pl](#)

- If you decide that you are not interested in exon junction probes that involve more than 'n' exons skipped this means that probes which exceed this limit will be ignored and the theoretical/ideal number of probes for each gene will be adjusted to compensate for this.
- When this script is run, the user must also specify the desired size of exon probesets and junction probesets
- The script for filtering the probes described below also allows the user to specify criteria for eliminating probes that exceed this limit.

17.) Create summary of all probe statistics for all raw probes

- This step summarizes the probe characteristics after only the initial filtering has been done.
- This will help to determine the pass/fail thresholds to use for the following steps (for example, what free energy of folding is considered acceptable).
- These summaries will be conducted in 'R' but using a perl wrapper:
[getProbeStats.pl](#)
- Simply specify a directory containing probe files. Figures and statistics will be generated automatically.

18.) Get chromosome coordinates for all probes

- The coordinates stored in probes files to this point are relative to the complete gene sequence (1st base of 1st exon to last base of last exon including intronic sequences).
- To add chromosome coordinates to these files use the script: [getProbeChromosomePositions.pl](#)

Probe Filtering

20.) Filter probes based on scores

- Use [create_filterProbesBatch.pl](#) to divide probe files into blocks of probesets and reduce the memory required to filter large number of probes. All probes in an input file can be processed at once but this can be extremely memory intensive for large genomes.
[create_filterProbesBatch.pl](#)
- This will create a batch file with all the necessary commands. Simply run the following command at a unix/linux prompt: 'bash filter_probes.sh'
- Essentially each probe will be considered and given a pass or fail for each criterion as follows:
 - T_m within a specified range (e.g. 67°C +/- 3.0°C)
 - Good specificity – no significant hits to mRNAs, ESTs, Ensembl transcripts, and genomic

sequences that map outside the target region. The probe should be specific to a single site within the target gene. Random negative control probes should not have similarity to any known sequence.

- No significant self-self or internal folding potential (pairfold and simfold scores)
 - No low complexity regions (e.g. any thing with 6 or more mdust masked bases fails)
 - If multiple exon or intron probes pass for the same exon or intron region, select the best probes according to target T_m and target length until the desired probeset size is achieved. These probes will also be selected to minimize their overlap within the exon or intron sequence.
 - Exon junction probes will be failed if they correspond to an exon-skipping event of more than the specified limit
 - Probesets that do not have the required size (number of probes) will be noted but partial probesets are allowed
- Run the script above on each input probe file (exon junction, exon boundary, exon and intron) separately. These files should contain all the necessary scores to evaluate each probe.
 - The script above creates jobs to run the following script on blocks of probes.

[filterProbes.pl](#)

- This script will create a file of filtered probes that pass all tests. One of these files is created for each block of probes. Finally, these blocks are automatically merged into a single filtered probes file.
 - Follow the instructions displayed by this script when run without options.
 - While examining each probe it will summarize the number of probes that fail each step, the number of probes that fail any step and output the records for the probes that successfully passed all tests.
 - Carefully examine the results of these tests and ensure that a reasonable number of probes are passing. If not you may need to relax the stringency of some tests
 - Remember to store batch files and log files from filtering steps as a record of the filtering process.
 - These probes will be defined as a target set in ALEXA and the filtering criteria will be stored for future reference.
- Note that negative control probes are scored in a similar fashion to experimental probes as described above except their specificity is tested differently. Since these probes have no target gene they are selected such that they have 0 hits to any mRNA, EST, EnsEMBL transcript, experimental probe or the genome.
 - A small number of these probes will be selected to fill remaining space on your array after selecting all of the experimental probes for the desired target genes.

21.) Create a final summary of all filtered probes

Stats can be conducted as described in steps 16 and 17.

Importing Probes to the Database

22.) Populate database with probe info

- Dump all experimental probes and all of their scores to the probe table in ALEXA using the script: [importProbeRecords.pl](#)
- Note that many of these probes may be poor in design. An additional script ([filterProbes.pl](#)) will

consider all probes and try to pick the best probes and fail the worst probes. However, by keeping all of the probes except those that fail very basic tests and were skipped immediately you will have more flexibility in choosing probes to fit a particular experiment later. In other words there may be some cases where it is desirable to be more or less stringent when picking the probes to actually include on the array.

- This script will process probe files in order according to probe id ranges.

23.) Define filtered probes as a probe ‘set’ in the database

- Use the following script to define the subset of probes that pass the filtering stage as a ‘Probe_set’ in the ALEXA database (note that negative control probes are not considered here).

[defineFilteredProbeSet.pl](#)

24.) Backing-up/Restoring an ALEXA database

- Create a back-up of the ALEXA database using the following script:
[~/sql/backupAlexaDb.pl](#).
- To maximize compatibility, you may want to run this from your database server.
- To restore a database from a back-up, use the following script:
[~/sql/restoreAlexaDb.pl](#)

Creating and Visualizing a Custom Array

25.) Gene selection

- It is very difficult to describe the gene selection process because it will be highly dependent on the goals of your particular study. For species with a small number of known genes it may be possible to select probes such that all genes are covered by single design. This is not currently possible for mammalian species such as human and mouse.
- Note: A number of custom array manufacturer’s are planning to make custom high density arrays available with sufficient capacity to overcome this limitation (2-3 million probes on a single array).
- You will most likely have to come up with some way of creating a list of target genes on your own.

26.) Select probes and generate Design Submission file

- Select the probes to be used based on an input gene list and generate an output file for submission to NimbleGen. These probes can be selected in a number of ways. The following script accepts an input list of selected gene targets (single Ensembl ID per line), retrieves the probe IDs and sequences for these genes and generates a file for submission to an array manufacturer such as NimbleGen.

[createDesignSubmissionFile.pl](#)

- You can also use this script to generate genome-wide designs, limit the design to only those exon junctions and boundaries with sequence support (e.g. known splicing events), as well as many other options. Run the script without options for a complete list of input parameters.
- In addition to the design file to be submitted to an array manufacturer, this script generates a number of additional output files:
 - Design success file. Tells you which of the genes in your input file were successfully targeted and if not then why.
 - Design annotation file. For those genes successfully targeted provides basic information on

the gene.

- Probe record file. Detailed record of each probe. This file can be used to generate custom UCSC tracks (see step 27) to help visualize and validate the array design before submitting it for synthesis.
- Check this output file to see if it makes sense. It should contain the expected number of negative control probes, etc.
- Note: I recommend that every array contain a small number of negative control probes (say 1% of array space). Negative control probes will be selected to uniformly represent the range of lengths and T_m of all experimental probes.
- I recommend that you manually check some of the probesets to see if they look correct. For example you can take some of each type for a single gene and use blat to align them to your target genome and see if they make sense (<http://genome.ucsc.edu/cgi-bin/hgBlat?command=start>).

27.) Create UCSC tracks to visualize probe design

- Once you have filtered probes for quality and selected probes for a list of target genes you can generate custom UCSC tracks to visualize their position in the genome
- Detailed instructions on creating custom tracks can be found at:
<http://genome.ucsc.edu/goldenPath/help/customTrack.html>
- The following script should assist in the creation of custom tracks for your custom design:
[createProbeUcscTracks.pl](#)
- This script will also create an annotation file summarizing information at the gene level.
 - Gene_id, EnsEMBL_ID, Entrez_ID, # exons, # transcripts, # probes, coordinates, etc.

28.) Populate database with microarray design info

- Once you have created a design and submitted it to your array manufacturer, they should provide you with a 'Design File'. This file will describe the physical layout of your probes on the microarray. You should define this array layout in the ALEXA database used to create the design. The following script can be used to import design file:

[importArrayDesign.pl](#)

- Note: This script assumes the format of a NimbleGen design file which comes on the data DVD they ship to their customers. If you use a different designer, this script may need to be modified.

References

Andronescu, M., R. Aguirre-Hernandez, *et al.* (2003). "RNAsoft: A suite of RNA secondary structure prediction and design software tools." Nucleic Acids Res **31**(13): 3416-22.

Breslauer, K. J., R. Frank, *et al.* (1986). "Predicting DNA duplex stability from the base sequence." Proc Natl Acad Sci U S A **83**(11): 3746-50.

Hancock, J. M. and J. S. Armstrong (1994). "SIMPLE34: an improved and enhanced implementation for VAX and Sun computers of the SIMPLE algorithm for analysis of clustered repetitive motifs in nucleotide sequences." Comput Appl Biosci **10**(1): 67-70.

Hubbard, T., D. Andrews, *et al.* (2005). "Ensembl 2005." Nucleic Acids Res **33**(Database issue): D447-53.

Sugimoto, N., S. Nakano, *et al.* (1996). "Improved thermodynamic parameters and helix initiation factor to predict stability of DNA duplexes." Nucleic Acids Res **24**(22): 4501-5