The Mathematica® Journal

A Computational Strategy for Effective Gene Silencing through siRNAs

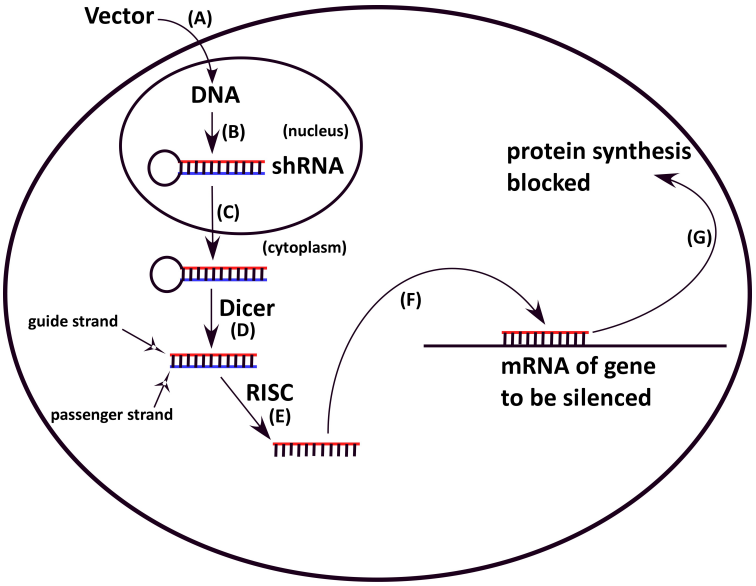
Designing siRNAs for Gene Silencing

Todd D. Allen

Biological systems possess traits that are a product of coordinated gene expression, heavily influenced by environmental pressures. This is true of both desirable and undesirable traits (i.e. disease). In cases of unwanted traits, it would be tremendously useful to have a means of systematically lowering the expression of genes implicated in producing the undesirable traits. This article describes the implementation of a computational biology algorithm designed to compute small interfering RNA sequences, capable of systematically silencing the expression of specific genes.

Introduction and Background

Small interfering RNAs (siRNAs) are short, single-stranded fragments of RNA, typically 20–30 nucleotides in length, that have created tremendous excitement in the biological sciences because of their ability to down-regulate the expression of genes in a targeted, systematic manner. siRNAs mediate their knockdown effects by activating a gene expression surveillance mechanism widely conserved in eukaryotic cells, known as RNA interference (RNAi). Once activated by siRNA, the RNAi machinery of the cell suppresses the expression of a specific gene by blocking translation (the synthesis of protein by ribosomes) or by promoting the degradation of mRNA needed to synthesize a protein. Figure 1 highlights the basic interplay between siRNA and RNAi. Further excellent reviews of siRNA and RNAi can be found in [1–3]. Having the ability to purposefully engineer siRNAs to "turn off" (or "turn down") the expression of genes promoting disease phenotypes like cancer, multiple sclerosis, diabetes, inflammation, etc. is of immediate and widespread interest to scientists, clinicians, and patients living with the burden of disease. Indeed, several clinical trials are currently evaluating the effectiveness of siRNA therapy for conditions such as transthretin amyloidosis [4], elevated intraocular pressure [5], and non-small cell lung cancer [6].



▲ **Figure 1.** (A) A genetically modified retroviral vector is used to insert DNA instructions into the nucleus of a eukaryotic cell. (B) The cell uses the inserted DNA instructions to manufacture short hairpin RNA (shRNA). (C) Following enzymatic trimming in the nucleus, the shRNA is exported from the nucleus. (D) An enzyme named Dicer removes the loop from the shRNA hairpin, releasing a double-stranded version of siRNA. (E) A complex of proteins named RISC (short for RNA-induced silencing complex) enzymatically removes the passenger strand of the siRNA duplex. (F) The mature single-stranded siRNA binds to its target—a specific mRNA from a gene being expressed by the cell. (G) The presence of siRNA attached to the mRNA interferes with translation (protein production), effectively short-circuiting the cell's intention to fully express the information contained in its DNA.

This article describes a Mathematica implementation (named siRNA Silencer) of the seminal algorithm created by Naito et al. [7] to design siRNA sequences for the express purpose of silencing a specific gene. Like Naito's algorithm, siRNA Silencer (SRS for short) uses "rules" of effective siRNA design that have been elucidated through meticulous experimentation [8]. To generate viable siRNA candidates, SRS progressively works through the four steps described in Table 1.

Step 1 (create 23-mers)

Generate a list of all possible 23-mer sequences from the transcript's nucleotide sequence.

These 23-mer sequences correspond to the complementary sequence of 21-nucleotide guide strand and 2-nucleotide 3' overhang of the passenger strand within the target sequence (Figures 1 and 2).

Step 2 (select functional siRNAs)

Choose entries from the 23-mer list that simultaneously satisfy the following rules:

Rule A: Choose 23-mers that have an A or a U at the 5' terminus of the guide strand.

Rule B: Choose 23-mers that have a G or C at the 5' terminus of the passenger strand.

Rule C: Choose 23-mers that have at least 4 A or U residues in the 5' terminal 7 base pairs of the guide strand.

Rule D: Remove any 23-mers that have G and/or C stretches longer than 9 base pairs in length.

Step 3 (reduce seed-dependent off-target effects)

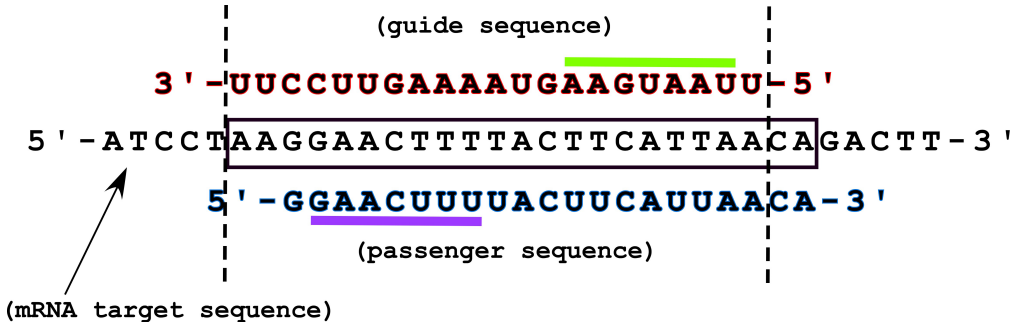
Calculate the melting temperature (Tm) of the seed-target duplex (i.e. nucleotide pairs 2–8 of the siRNA guide and passenger strands as bound to their target sequence) for the 23-mers that survive Step 2. Accept only those sequences with a Tm less than 21.5 °C.

Step 4 (eliminate siRNAs that may have off-targets)

Search for off-target gene sequences that proposed siRNAs may inappropriately silence. Present these off-target possibilities to the user for consideration in designing future experiments.

▲ **Table 1**. The rules used to computationally build siRNA sequences capable of silencing a specific gene, while simultaneously minimizing the possibility of inappropriately silencing similar genes.

As an example of how the steps in Table 1 are applied, Figure 2 presents a detailed illustration of some of the output generated by SRS when it is tasked with silencing the human gene IFNβ1. (Interferon beta 1 is a powerful antiviral protein, manufactured from instructions in the IFNβ1 gene [9].)



▲ **Figure 2.** One of 23 siRNA candidates generated by siRNA Silencer to silence IFNβ1 expression is presented in detail here. The middle strip of letters contains a portion of the mRNA sequence manufactured from instructions in the human IFNβ1 gene. The black box surrounds nucleotides 591 to 613 of the mRNA target, which is precisely 23 nucleotides in length, corresponding to Step 1 above. The sequence outlined in red is the guide strand of RNA that will eventually bind to the target mRNA and block protein synthesis (Figure 1). Notice how the guide sequence satisfies rules A, C, and D in Step 2 above. The passenger sequence (blue outline) satisfies rules B and D in Step 2. The predicted melting temperatures (details described below) of the guide-target seed sequence (green bar) and the passenger-target seed sequence (purple bar) are 8.98 °C and 13.3 °C, respectively, satisfying Step 3 above. Step 4 output will be discussed below.

As the end user interprets the output of siRNA Silencer and ultimately makes a decision about which presented siRNA candidate to use, it becomes a (relatively) simple cloning project to engineer the guide and passenger sequences into an expression vector (Figure 1) to silence the gene in a living cell.

The siRNA Silencer (SRS) Algorithm

SRS is template driven, meaning the algorithm expects several pieces of user-defined information to be provided in a notebook cell that is used as a template for entering information. The features of SRS will be illustrated using the pre-mRNA sequence of the human gene IFNβ1 (interferon beta 1) to design siRNA candidates capable of silencing IFNβ1.

```
refseqlocation =
  "C:\\Users\\Wookie\\Desktop\\Mathematica
    Projects\\Mathematica Journal Projects\\Data\\RefSeq
    transcript data\\";

refdatabaseversion = humantranscripts;

savelocationroot = "C:\\Users\\Wookie\\Desktop\\";

studyname = silence ifnb1;

query =
  "ACATTCTAACTGCAACCTTTCGAAGCCTTTGCTCTGGCACAACAGGTAGTAGGCGA:
     CACTGTTCGTGTTGTCAACATGACCAACAAGTGTCTCCTCCAAATTGCTCT:
     CCTGTTGTGCTTCTCCACTACAGCTCTTTCCATGAGCTACAACTTGCTTGG:
     ATTCCTACAAAGAAGCAGCAATTTTCAGTGTCAGAAGCTCCTGTGGCAATT:
     GAATGGGAGGCTTGAATACTGCCTCAAGGACAGGATGAACTTTGACATCCC:
     TGAGGAGATTAAGCAGCTGCAGCAGTTCCAGAAGGAGGACGCCGCATTGAC:
     CATCTATGAGATGCTCCAGAACATCTTTGCTATTTTCAGACAAGATTCATC:
     TAGCACTGGCTGGAATGAGACTATTGTTGAGAACCTCCTGGCTAATGTCTA:
     TCATCAGATAAACCATCTGAAGACAGTCCTGGAAGAAAAACTGGAGAAAGA:
     AGATTTCACCAGGGGAAAACTCATGAGCAGTCTGCACCTGAAAAGATATTA:
     TGGGAGGATTCTGCATTACCTGAAGGCCAAGGAGTACAGTCACTGTGCCTG:
     GACCATAGTCAGAGTGGAAATCCTAAGGAACTTTTACTTCATTAACAGACT:
     TACAGGTTACCTCCGAAACTGAAGATCTCCTAGCCTGTGCCTCTGGGACTG:
     GACAATTGCTTCAAGCATTCTTCAACCAGCAGATGCTGTTTAAGTGACTGA:
     TGGCTAATGTACTGCATATGAAAGGACACTAGAAGATTTTGAAATTTTTAT:
     TAAATTATGAGTTATTTTTATTTATTTAAATTTTATTTTGGAAAATAAATT:
     ATTTTTGGTGCAAAAGTCA";
```

The example above contains several variables that must be completed by the user to let SRS do its job. The reader may modify them to run the code.

The variables requiring user input are:

1. `refseqlocation`: This variable holds the directory location for finding pre-parsed RNA data files from specific organisms of interest. The pre-parsed RNA data files were generated by downloading transcript source files from NCBI's Map Viewer FTP (ftp://ftp.ncbi.nih.gov/genomes/MapView). The data contained in these files is part of NCBI's RefSeq database, which maintains a nonredundant catalog of all known biological molecules in a species-specific manner [10]. Original source files include:

Organism	Annotation Release Number	Date Modified	Number of Transcripts
Homo sapiens	106	02 – 14 – 2014	98 746
Mus musculus	104	12 – 18 – 2013	115 860
Rattus norvegicus	105	08 – 09 – 2014	79 019
Canis lupus familiaris	103	09 – 24 – 2013	48 370
Felis catus	101	02 – 25 – 2014	32 542

Once downloaded, the source files were parsed by custom Mathematica code (not shown here) to generate expressions containing transcript accession values, annotation information, and sequence information for the genes contained in the source files. These expressions were saved and serve as the primary data source for the SRS program.

2. `refdatabaseversion`: This variable contains the name of the specific organism from which a gene is to be silenced. Options available include: `"humantranscripts"`, `"mousetranscripts"`, `"rattranscripts"`, `"dogtranscripts"`, and `"cattranscripts"`. The option chosen must correspond to the name of a file present in < `refseqlocation` > defined above.

3. `savelocationroot`: This variable holds the location where the user would like the final results of the analysis to be saved.

4. `studyname`: This variable lets the user name the output generated by SRS. The output of SRS is saved using this name to the location provided in `"saverootlocation"` above.

5. `query`: This variable contains the transcript sequence, in string format, of the gene to be silenced.

SRS starts by loading the pre-parsed RNA data file from which the query gene sequence is to be silenced and then proceeds to create a preliminary list of siRNA candidates that are filtered through the four steps referenced in Table 1.

```
starttime = AbsoluteTime[];

SetDirectory[refseqlocation];

refdatabase = Get[ToString[refdatabaseversion]];
```

```
mers23 = Partition[Characters[ToUpperCase[query]], 23, 1];

cgaffsrulesdna = {"T" → "A", "A" → "T", "C" → "G", "G" → "C"};

cgaffsrulesrna = {"T" → "A", "A" → "U", "C" → "G", "G" → "C",
    "U" → "A"};

tTOu = {"T" → "U"};

uTOt = {"U" → "T"};

guidemers =
  Table[Reverse[Drop[mers23[[i]], -2]],
    {i, 1, Length[mers23]}] /. cgaffsrulesrna;

passengermers =
  Table[Drop[mers23[[i]], 2], {i, 1, Length[mers23]}] /.
   tTOu;

siRNAcandidates =
  MapThread[List, {guidemers, passengermers}];

filter1 = Cases[siRNAcandidates,
   x_ /; ((x[[1, 1]] == "A" ⋁ x[[1, 1]] == "U") ⋀
       (x[[2, 1]] == "G" ⋁ x[[2, 1]] == "C"))];

filter2a = Table[Tally[filter1[[i, 1, 1 ;; 7]]],
   {i, 1, Length[filter1]}];

filter2b1 =
  Table[
   Total[
    Select[filter2a[[i]], #[[1]] == "A" ⋁ #[[1]] == "U" &][[
     All, 2]]], {i, 1, Length[filter2a]}];

filter2b2 = Position[filter2b1, x_ /; x ≥ 4];

filter2c = filter1[[Flatten[filter2b2]]];

gctuples = Tuples[{"G", "C"}, {10}];

filter3a =
  Table[Position[filter2c,
    Flatten[{___, gctuples[[i]], ___}]],
   {i, 1, Length[gctuples]}];

filter3b = Union[Partition[Flatten[filter3a], 2][[All, 1]]];
```

```
filter3b = Table[{filter3b[[i]]}, {i, 1, Length[filter3b]}];

filter3c = Delete[filter2c, filter3b];
```

For the specific gene highlighted here (IFNβ1), successive pruning of the initial list of 818 23-nucleotide-long sequences is highlighted by a decline in the length of variables containing the pruning results. The variables highlighted below contain the results of completing Step 1, Step 2 (Rules A and B), and Step 2 (Rules C and D).

```
{Length[mers23], Length[filter1], Length[filter3c]}

{818, 186, 145}
```

A small portion of initial siRNA candidates is shown here, where each row contains a guide and passenger sequence (Figure 2) generated from IFNβ1 that may, if they pass additional requirements, prove useful in silencing the expression of IFNβ1.

```
filter3c[[1 ;; 5]]

{{{U, U, C, G, A, A, A, G, G, U, U, G, C, A, G, U, U, A, G, A, A},
  {C, U, A, A, C, U, G, C, A, A,
   C, C, U, U, U, C, G, A, A, G, C}},
 {{A, A, A, G, G, C, U, U, C, G, A, A, A, G, G, U, U, G, C, A, G},
  {G, C, A, A, C, C, U, U, U, C,
   G, A, A, G, C, C, U, U, U, G, C}},
 {{A, G, A, G, C, A, A, A, G, G, C, U, U, C, G, A, A, A, G, G, U},
  {C, U, U, U, C, G, A, A, G, C,
   C, U, U, U, G, C, U, C, U, G, G}},
 {{U, G, U, U, G, U, G, C, C, A, G, A, G, C, A, A, A, G, G, C, U},
  {C, C, U, U, U, G, C, U, C, U,
   G, G, C, A, C, A, A, C, A, G, G}},
 {{A, C, U, A, C, C, U, G, U, U, G, U, G, C, C, A, G, A, G, C, A},
  {C, U, C, U, G, G, C, A, C, A,
   A, C, A, G, G, U, A, G, U, A, G}}}
```

The algorithm moves next to complete Step 3, in which the predicted melting temperatures of the guide and passenger seed sequences (Figure 2) are calculated and screened to accept only those sequences with predicted melting temperatures below 21.5 °C. Melting temperature is a reflection of thermodynamic stability, and for the purposes of gene silencing, experiments have suggested melting temperatures below 21.5 °C are optimal [11]. If no candidate siRNAs can meet that demand, the requirement is waived, but the algorithm will print a warning message that the requirement could not be met.

```
thermdynpara = {{{"AU", "AU"}, {"UA", "UA"}, -6.6, -18.4},
    {{"AU", "UA"}, {"AU", "UA"}, -5.7, -15.5},
    {{"UA", "AU"}, {"UA", "AU"}, -8.1, -22.6},
    {{"CG", "AU"}, {"UA", "GC"}, -10.5, -27.8},
    {{"CG", "UA"}, {"AU", "GC"}, -7.6, -19.2},
    {{"GC", "AU"}, {"UA", "CG"}, -13.3, -35.5},
    {{"GC", "UA"}, {"AU", "CG"}, -10.2, -26.2},
    {{"CG", "GC"}, {"CG", "GC"}, -8.0, -19.4},
    {{"GC", "CG"}, {"GC", "CG"}, -14.2, -34.9},
    {{"GC", "GC"}, {"CG", "CG"}, -12.2, -29.7}};

tmA = -10.8 ;
tmR = 1.987;
tmCt = 0.0001;
tmNa = 0.1;

tmcalc[string_] :=
 Module[{string2to8, comp2to8, doublets, choosedata,
   deltaHsum, deltaSsum, tmfinal},
  string2to8 = string[[2 ;; 8]];
  comp2to8 = string2to8 /. cgaffsrulesrna;
  doublets =
   Partition[Map[StringJoin,
     Partition[Riffle[string2to8, comp2to8], 2]], 2, 1];
  choosedata =
   Flatten[
    Table[Select[thermdynpara,
      #[[1]] == doublets[[i]] ⋁ #[[2]] == doublets[[i]] &],
     {i, 1, Length[doublets]}], {1, 2}];
  deltaHsum = Total[choosedata[[All, 3]]];
  deltaSsum = Total[choosedata[[All, 4]]];
  tmfinal =
   ((1000.0 * deltaHsum) /
      (tmA + deltaSsum + (tmR * Log[tmCt / 4]))) - 273.15 +
    (16.6 * Log10[tmNa])]

guidetms = Map[tmcalc, filter3c[[All, 1]]];
passengertms = Map[tmcalc, filter3c[[All, 2]]];

datacombine =
  MapThread[List, {filter3c, guidetms, passengertms}];

step2finalists = Select[datacombine,
   (#[[2]] < 21.5 ⋀ #[[3]] < 21.5) &];
```

```
If[Length[step2finalists] == 0, (step2finalists = datacombine;
   Print[
    Style[
     "Warning: siRNA candidates generated from this
       query fail to meet the 21.5 deg Tm
       cutoff for their seed-target duplexes.
       The algorithm will proceed by ignoring
       this requirement. Careful inspection
       of output is required.", Red, Bold, 16]])];

step2cleanupa = Map[StringJoin, step2finalists[[All, 1, 1]]];
step2cleanupb = Map[StringJoin, step2finalists[[All, 1, 2]]];
step2clean = MapThread[List,
   {step2cleanupa, step2cleanupb, step2finalists[[All, 2]],
    step2finalists[[All, 3]]}];
```

Inspection of the first five candidates that survived Step 3 screening reveals that none of them match the first five candidates that survived Step 2 screening. This means that none of the first five candidates from Step 2 screening had melting temperatures below 21.5 ℃, causing SRS to remove them from further consideration.

```
step2finalists[[1 ;; 5]]
```

```
{{{{A, C, U, U, G, U, U, G, G, U, C, A, U, G, U,
    U, G, A, C, A, A}, {G, U, C, A, A, C, A, U, G, A, C,
    C, A, A, C, A, A, G, U, G, U}}, 16.7534, 20.5384},
 {{{A, G, U, U, G, U, A, G, C, U, C, A, U, G, G, A, A,
    A, G, A, G}, {C, U, U, U, C, C, A, U, G, A, G, C,
    U, A, C, A, A, C, U, U, G}}, 19.0935, 20.1472},
 {{{U, U, C, U, U, U, G, U, A, G, G, A, A, U, C, C, A,
    A, G, C, A}, {C, U, U, G, G, A, U, U, C, C, U, A,
    C, A, A, A, G, A, A, G, C}}, 19.2339, 20.1472},
 {{{U, C, U, C, A, U, A, G, A, U, G, G, U, C, A, A, U,
    G, C, G, G}, {G, C, A, U, U, G, A, C, C, A, U, C,
    U, A, U, G, A, G, A, U, G}}, 17.8428, 20.5384},
 {{{A, A, A, U, A, G, C, A, A, A, G, A, U, G, U, U, C,
    U, G, G, A}, {C, A, G, A, A, C, A, U, C, U, U, U,
    G, C, U, A, U, U, U, U, C}}, 19.7848, 19.2113}}
```

Indeed, there is considerable reduction in the size of the candidate list from Step 2 to Step 3.

```
{Length[filter3c], Length[step2finalists]}
```

```
{145, 23}
```

After the candidate list of potential siRNAs is pruned through Step 3, the algorithm presents a visual representation of the current list of siRNA candidates as they are mapped to their respective positions within the query sequence.

```
targetseqinfo[inputseq_] :=
 Module[{ uTOt, tsp1, tsp2, targetpos, targetseq, targetanti },

  uTOt = {"U" → "T"};
  tsp1 = inputseq /. uTOt;

  tsp2 = StringPosition[query, StringJoin[tsp1]];
  targetpos = {tsp2[[1, 1]] - 2, tsp2[[1, 2]]};

  targetseq = StringTake[query, targetpos];
  targetanti =
   StringJoin[Reverse[Characters[targetseq]] /.
     cgaffsrulesdna];

  {targetpos, targetseq, targetanti}]

siRNAprettyprint[query_] :=
 Module[{querygva, qgvdeletepos, querygvb, posrulera,
   posrulerpad, posrulerb, posrulerc, posrulerd,
   overlapcoord, upboundaries, lowboundaries, boundaries,
   firstintcheck, secondintcheck, combinedintcheck,
   spillover, nospillover, countnospill, nospillentries,
   spillentries, nospillgraphic, nospilldata,
   spillovergraphic, spilldata, overlapadj, allgraphic,
   allgraphic2, allgraphic3, query100mers,
   wspacetoallgraphica, wspacetoallgraphicb,
   rulerwithlinepos, buildoutputa, buildoutputb},

  querygva = Partition[Characters[query], 100, 100,
    {1, 1}, "F"];

  qgvdeletepos = Position[querygva[[-1]], "F"];

  querygva[[-1]] = Delete[querygva[[-1]], qgvdeletepos];

  querygvb = Map[StringJoin, querygva];

  posrulera =
   Partition[Delete[Table[i, {i, 0, StringLength[query], 10}],
     1], 10, 10, {1, 1}, 0];

  posrulera[[-1]] = DeleteCases[posrulera[[-1]], x_ /; x == 0];

  posrulerpad[x_] :=
   ConstantArray[" ", (10 - IntegerLength[x])];

posrulerb = Table[Map[posrulerpad, posrulera[[i]]],
```

```
    {i, 1, Length[posrulera]}];

  posrulerc =
   Table[Map[ToString,
     Flatten[Riffle[posrulerb[[i]], posrulera[[i]]]]],
    {i, 1, Length[posrulera]}];

  posrulerd = Map[StringJoin, posrulerc];

  overlapcoord = masterlist[[All, 1 ;; 2]];

  upboundaries = Drop[Range[0, (Length[querygva] * 100), 100],
    1];

lowboundaries = Range[1, (Length[querygva] * 100), 100];

  boundaries = Partition[Riffle[lowboundaries, upboundaries],
    2];

  firstintcheck =
   Table[{i, j, IntervalMemberQ[Interval[boundaries[[i]]],
      overlapcoord[[j, 1]]]}, {i, 1, Length[boundaries]},
    {j, 1, Length[overlapcoord]}];

secondintcheck =
   Table[{i, j, IntervalMemberQ[Interval[boundaries[[i]]],
      overlapcoord[[j, 2]]]}, {i, 1, Length[boundaries]},
    {j, 1, Length[overlapcoord]}];

combinedintcheck =
   Partition[Riffle[Flatten[firstintcheck, 1],
     Flatten[secondintcheck, 1]], 2];

spillover = Select[combinedintcheck,
    #[[1, 3]] == True && #[[2, 3]] == False &];

nospillover =
   Select[combinedintcheck,
     #[[1, 3]] == True && #[[2, 3]] == True &][[All, 1]];

countnospill = Tally[nospillover[[All, 1]]];

nospillentries = nospillover[[All, 2]];

spillentries = spillover[[All, 1, 2]];

nospillgraphic =
   Table[siRNAgraphic[overlapcoord[[nospillentries[[i]]]]],
    {i, 1, Length[nospillentries]}];

nospilldata = MapThread[List,
```

```
    {nospillover[[All, 1]], overlapcoord[[nospillentries]],
     nospillgraphic}];

  spillovergraphic =
   Table[siRNAgraphic[overlapcoord[[spillentries[[i]]]]],
    {i, 1, Length[spillentries]}];

spilldata = MapThread[List,
    {spillover[[All, 1, 1]], overlapcoord[[spillentries]],
     spillovergraphic}];

overlapadj =
   Flatten[Table[corrforoverlap[spilldata[[i]]],
     {i, 1, Length[spilldata]}], 1];

  allgraphic = Sort[Join[nospilldata, overlapadj],
    (#1[[1]] ≤ #2[[1]] &)];
allgraphic2 = GatherBy[allgraphic, #[[1]] &];
allgraphic3 = Flatten[Map[Sort, allgraphic2], 1];

query100mers =
   Partition[Riffle[Range[Length[querygvb]], querygvb], 2];

  wspacetoallgraphica = Table[{allgraphic3[[i, 1]],
     Join[ConstantArray[" ",
       If[allgraphic3[[i, 2, 1]] < 100,
        allgraphic3[[i, 2, 1]] - 1,
        If[Mod[allgraphic3[[i, 2, 1]], 100] == 0,
         (99),
         (allgraphic3[[i, 2, 1]] - 1 -
           Floor[allgraphic3[[i, 2, 1]], 100])]]],
      Characters[allgraphic3[[i, 3]]],
      ConstantArray[" ",
       (Ceiling[allgraphic3[[i, 2, 2]], 100] -
         allgraphic3[[i, 2, 2]])]]},
    {i, 1, Length[allgraphic3]}];

wspacetoallgraphicb =
   Table[{wspacetoallgraphica[[i, 1]],
     Style[StringJoin[wspacetoallgraphica[[i, 2]]], Red]},
    {i, 1, Length[wspacetoallgraphica]}];

rulerwithlinepos =
   Partition[Riffle[Range[Length[posrulerd]], posrulerd], 2];

  buildoutputa = Join[rulerwithlinepos, query100mers,
    wspacetoallgraphicb];

buildoutputb = Flatten[GatherBy[buildoutputa, #[[1]] &], 1];

  Column[buildoutputb[[All, 2]]]]
```

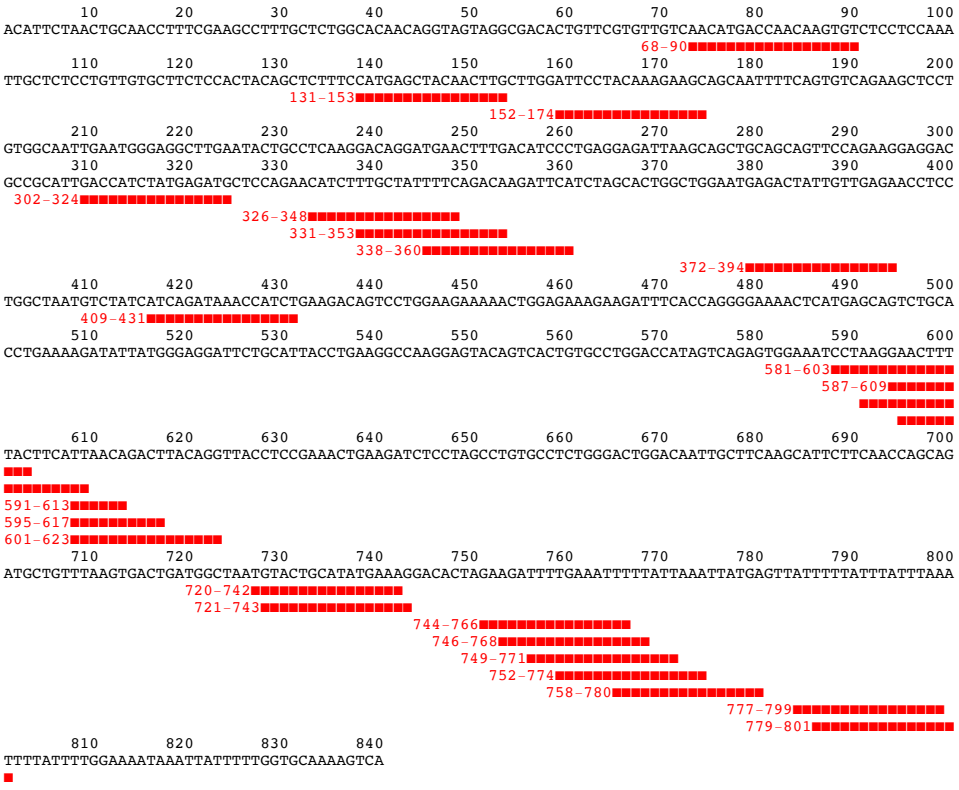
```
siRNAgraphic[entry_] := Module[{middlestring, beginning, end},

  beginningstring = StringJoin[ToString[entry[[1]]],
    "-", ToString[entry[[2]]]];
  endingstring =
   StringJoin[ConstantArray[ToString[■],
     (23 - StringLength[beginningstring])]];

  final = StringJoin[beginningstring, endingstring]]

corrforoverlap[entry_] :=
 Module[{deletionamount, updategraphic, updatecoord,
   updateoverlap, newlinepos, newlinecoord, newlinegraphic,
   newlinedat, newlinedata, origlinegraphic, origlinedata},

  deletionamount = entry[[2, 2]] - Floor[entry[[2, 2]], 100];
  newlinepos = entry[[1]] + 1;

  If[deletionamount ≤ 11,

   (updategraphic = StringDrop[entry[[3]], -deletionamount];
    updatecoord = {entry[[2, 1]],
      (entry[[2, 2]] - deletionamount)};
    updateoverlap = {entry[[1]], updatecoord, updategraphic};
    newlinecoord = {updatecoord[[2]] + 1,
      updatecoord[[2]] + deletionamount};
    newlinegraphic =
     StringJoin[ConstantArray[ToString[■], deletionamount]];
    newlinedata = {newlinepos, newlinecoord, newlinegraphic};
    {updateoverlap, newlinedata}),

   (updatecoord = {entry[[2, 1]], Floor[entry[[2, 2]], 100]};
    origlinegraphic =
     StringJoin[ConstantArray[ToString[■],
       (updatecoord[[2]] - updatecoord[[1]] + 1)]];
    origlinedata = {entry[[1]], updatecoord, origlinegraphic};
    newlinecoord = {updatecoord[[2]] + 1, entry[[2, 2]]};
    updategraphic = StringDrop[entry[[3]],
      -StringLength[origlinegraphic]];
    newlinedata = {newlinepos, newlinecoord, updategraphic};
    {origlinedata, newlinedata})]]

targsequences = Table[targetseqinfo[step2finalists[[i, 1, 2]]],
   {i, 1, Length[step2finalists]}];
masterlist =
  Partition[
   Flatten[Partition[Riffle[targsequences, step2clean], 2]],
   8];
```

```
Print[
  Style[
   "Positions of siRNA candidates\nwithin the query
     sequence ...", 14, Bold]];
Print[];
graphicoutput = siRNAprettyprint[query];
```

```
Positions of siRNA candidates
within the query sequence ...
```

For publication purposes, the next output is reduced in size to allow the data to fit within printable margins.

```
Magnify[graphicoutput, 0.55]
```



▲ **Figure 3.** A map of the siRNA candidates (shown in red) that have past the first three steps of Table 1, aligned to their positions within the target gene sequence (shown in black) to be silenced. The example here shows siRNA candidates suggested for silencing IFNβ1.

From here, SRS begins an exhaustive search for genes within < `refdatabase`: `version` > that contain subsequences that match the siRNA candidates graphically presented above. In our current example, SRS searches the siRNA candidates generated from IFNβ1 that may also bind to alternative human genes with similar subsequences. The results of this search are presented for inspection so the user can decide which of the potential siRNA candidates is least likely to create undesirable effects if used to silence the gene of interest. Making this decision is largely based on human experience, and is an area that highlights the "art of doing science."

Caution: Due to the sheer volume of computation that needs to be completed using the data described in this article, the next segment of code will likely take 20–40 minutes to complete (depending on the speed of your computer) and consume roughly 24 GB of RAM. Computations on machines with less RAM will finish, but will require significant use of the hard drive, slowing computation considerably.

```
mismatchsearch[mer19_, gene_] :=
 Module[{temp1, temp2, temp3, temp4, temp5, temp6, compfor,
   temp7, temp8, temp9, temp10, temp10a, temp11, temp11a,
   temp12, ok1, ok2, resultalignment},

temp1 = LongestCommonSubsequence[mer19, gene[[3]]];
temp2 = Flatten[StringPosition[gene[[3]], temp1]];

temp3 = Differences[temp2] + 1;
temp4 = 19 - temp3[[1]];

  temp5 =
   Which[
    (temp2[[1]] - temp4 > 0.0 &&
      temp2[[2]] + temp4 < StringLength[gene[[3]]]),
    StringTake[gene[[3]],
     {(temp2[[1]] - temp4), (temp2[[2]] + temp4)}],
    (StringLength[gene[[3]]] ≤ 40),
    StringTake[gene[[3]], {1, StringLength[gene[[3]]]}],
    (temp2[[1]] - temp4 ≤ 0.0),
    StringTake[gene[[3]],
     {1, (temp2[[2]] + temp4 + Abs[temp2[[1]] - temp4])}],
    (temp2[[2]] + temp4 ≥ StringLength[gene[[3]]]),
    StringTake[gene[[3]],
     {temp2[[1]] - temp4 -
       (temp2[[2]] + temp4 - StringLength[gene[[3]]]),
      StringLength[gene[[3]]]}]];

temp6 = Partition[Characters[temp5], 19, 1];

compfor = {"A" → 1.0, "T" → 2.0, "C" → 3.0, "G" → 4.0};
temp7 = temp6 /. compfor;
```

```
temp8 = ConstantArray[Characters[mer19] /. compfor,
    Length[temp6]];
temp9 = temp8 - temp7;

 temp10 = Map[Count[#, 0.] &, temp9];
temp10a = Map[Position[#, x_ /; x ≠ 0.] &, temp9];

 temp11 = Flatten[Position[temp10, Max[temp10]]];
temp11a = temp10a[[temp11[[1]]]];

 temp12 = StringJoin[temp6[[temp11[[1]]]]];

 ok1 = Characters[mer19];
 ok2 = Characters[temp12];

resultalignment =
   Grid[{ok1, ok2},
    Background → {Automatic, Automatic,
      Thread[Map[List[2, #] &, Flatten[temp11a]] → Green]}];

 {mer19, resultalignment, gene[[2]], Length[temp11a]}]

Print[];
Print[
  Style[
   "Searching RefSeq for potential\noff-target hits
     from siRNA candidates ...", 14, Bold]];
Print[];
```

```
Searching RefSeq for potential
off-target hits from siRNA candidates ...
```

```
alltarget =
  ParallelTable[
   mismatchsearch[StringTake[masterlist[[i, 3]], {2, 20}],
    refdatabase[[j]]], {i, 1, Length[masterlist]},
   {j, 1, Length[refdatabase]}, Method → "FinestGrained"];

target4orless = Table[Select[alltarget[[i]], (#[[4]] ≤ 4) &],
   {i, 1, Length[alltarget]}];
```

```
gathertarget4orless =
  Table[Flatten[Gather[target4orless[[i]],
     #1[[2]] == #2[[2]] &], 1], {i, 1, Length[target4orless]}];

gathertarget4sort =
  Table[Sort[gathertarget4orless[[i]], (#1[[4]] < #2[[4]]) &],
   {i, 1, Length[gathertarget4orless]}];

Clear[alltarget];

alltargetanti =
  ParallelTable[
   mismatchsearch[StringTake[masterlist[[i, 4]], {2, 20}],
    refdatabase[[j]]], {i, 1, Length[masterlist]},
   {j, 1, Length[refdatabase]}, Method → "FinestGrained"];

antitarget4orless =
  Table[Select[alltargetanti[[i]], (#[[4]] ≤ 4) &],
   {i, 1, Length[alltargetanti]}];

gatherantitarget4orless =
  Table[Flatten[Gather[antitarget4orless[[i]],
     #1[[2]] == #2[[2]] &], 1],
   {i, 1, Length[antitarget4orless]}];

gatherantitarget4sort =
  Table[Sort[gatherantitarget4orless[[i]],
    (#1[[4]] < #2[[4]]) &],
   {i, 1, Length[gatherantitarget4orless]}];

Clear[alltargetanti];

guidepopup =
  Table[Row[{Rest[gathertarget4sort[[i]][[All, 4]]][[1]],
     " ", PopupWindow[Checkbox[],
      Grid[gathertarget4sort[[i]][[All, 2 ;; 3]],
       Frame → All],
      WindowSize → {1500, 500},
      WindowElements → {"VerticalScrollBar",
        "MagnificationPopUp"},
      WindowTitle →
       "Potential off target hits for siRNA candidate
         (guide): ", "65-69",
      WindowFloating → False]}],
   {i, 1, Length[gathertarget4sort]}];
```

```
passengerpopup =
  Table[Row[{Rest[gatherantitarget4sort[[i]][[All, 4]]][[1]],
     " ", PopupWindow[Checkbox[],
      Grid[gatherantitarget4sort[[i]][[All, 2 ;; 3]],
       Frame → All],
      WindowSize → {1500, 500},
      WindowElements → {"VerticalScrollBar",
        "MagnificationPopUp"},
      WindowTitle →
       "Potential off target hits for siRNA candidate
         (passenger): ", "65-69",
      WindowFloating → False]}],
   {i, 1, Length[gatherantitarget4sort]}];

masterlistprettya =
  Table[StringJoin[ToString[masterlist[[i, 1]]], "-",
    ToString[masterlist[[i, 2]]]],
   {i, 1, Length[masterlist]}];
masterlistprettyb =
  Table[Column[{masterlist[[i, 5]], masterlist[[i, 6]]}],
   {i, 1, Length[masterlist]}];
masterlistprettyc =
  Map[NumberForm[#, 3] &, masterlist[[All, 7]]];
masterlistprettyd =
  Map[NumberForm[#, 3] &, masterlist[[All, 8]]];

masterlistprettyfinal =
  Grid[
   Prepend[
    Prepend[MapThread[List,
      {masterlistprettya, masterlist[[All, 3]],
       masterlistprettyb, masterlistprettyc,
       masterlistprettyd, guidepopup, passengerpopup}],
     Map[Style[#, Bold] &,
      {"", "", "", "guide", "passenger", "guide",
       "passenger"}]],
    Map[Style[#, Bold] &,
     {"target\nposition",
      "target sequence\n21nt target + 2nt overhang",
      "RNA oligo sequences\n21nt guide (5'->3')\n21nt
        passenger (5'->3')",
      "seed-duplex\nstability (Tm)", "",
      "specificity check\nminimum number of\nmismatches
        against\nany off-targets"}]], Frame -> All];
```

For publication purposes, the next output is reduced in size to allow the data to fit within printable margins.

```
Magnify[masterlistprettyfinal, 0.50]
```

target position	target sequence 21nt target + 2nt overhang	RNA oligo sequences 21nt guide (5'->3') 21nt passenger (5'->3')	seed-duplex stability (Tm)		specificity check minimum number of mismatches against any off-targets	
			guide	passenger	guide	passenger
68-90	TTGTCAACATGACCAACAAGTGT	ACUUGUUGGUCAUGUUGACAA GUCAACAUGACCAACAAGUGU	16.8	20.5	2 ☐	3 ☐
131-153	CTCTTTCCATGAGCTACAACTTG	AGUUGUAGCUCAUGGAAAGAG CUUUCCAUGAGCUACAACUUG	19.1	20.1	3 ☐	2 ☐
152-174	TGCTTGGATTCCTACAAAGAAGC	UUCUUUGUAGGAAUCCAAGCA CUUGGAUUCCUACAAAGAAGC	19.2	20.1	2 ☐	3 ☐
302-324	CCGCATTGACCATCTATGAGATG	UCUCAUAGAUGGUCAAUGCGG GCAUUGACCAUCUAUGAGAUG	17.8	20.5	3 ☐	3 ☐
326-348	TCCAGAACATCTTTGCTATTTTC	AAAUAGCAAAGAUGUUCUGGA CAGAACAUCUUUGCUAUUUUC	19.8	19.2	2 ☐	2 ☐
331-353	AACATCTTTGCTATTTTCAGACA	UCUGAAAAUAGCAAAGAUGUU CAUCUUUGCUAUUUUCAGACA	12.2	12.1	2 ☐	2 ☐
338-360	TTGCTATTTTCAGACAAGATTCA	AAUCUUGUCUGAAAAUAGCAA GCUAUUUUCAGACAAGAUUCA	19.2	-4.34	3 ☐	2 ☐
372-394	CTGGAATGAGACTATTGTTGAGA	UCAACAAUAGUCUCAUUCCAG GGAAUGAGACUAUUGUUGAGA	12.1	20.4	2 ☐	3 ☐
409-431	GTCTATCATCAGATAAACCATCT	AUGGUUUAUCUGAUGAUAGAC CUAUCAUCAGAUAAACCAUCU	20.	17.4	3 ☐	3 ☐
581-603	TGGAAATCCTAAGGAACTTTTAC	AAAAGUUCCUUAGGAUUUCCA GAAAUCCUAAGGAACUUUUAC	13.3	18.8	2 ☐	3 ☐
587-609	TCCTAAGGAACTTTTACTTCATT	UGAAGUAAAAGUUCCUUAGGA CUAAGGAACUUUUACUUCAUU	14.6	19.9	2 ☐	2 ☐
591-613	AAGGAACTTTTACTTCATTAACA	UUAAUGAAGUAAAAGUUCCUU GGAACUUUUACUUCAUUAACA	8.98	13.3	2 ☐	2 ☐
595-617	AACTTTTACTTCATTAACAGACT	UCUGUUAAUGAAGUAAAAGUU CUUUUACUUCAUUAACAGACU	11.9	4.98	2 ☐	3 ☐
601-623	TACTTCATTAACAGACTTACAGG	UGUAAGUCUGUUAAUGAAGUA CUUCAUUAACAGACUUACAGG	21.4	8.98	3 ☐	3 ☐
720-742	TGGCTAATGTACTGCATATGAAA	UCAUAUGCAGUACAUUAGCCA GCUAAUGUACUGCAUAUGAAA	21.1	11.7	3 ☐	3 ☐
721-743	GGCTAATGTACTGCATATGAAAG	UUCAUAUGCAGUACAUUAGCC CUAAUGUACUGCAUAUGAAAG	15.	8.51	3 ☐	2 ☐
744-766	GACACTAGAAGATTTTGAAATTT	AUUUCAAAAUCUUCUAGUGUC CACUAGAAGAUUUUGAAAUUU	7.72	19.	2 ☐	3 ☐
746-768	CACTAGAAGATTTTGAAATTTTT	AAAUUUCAAAAUCUUCUAGUG CUAGAAGAUUUUGAAAUUUUU	7.5	20.3	2 ☐	2 ☐
749-771	TAGAAGATTTTGAAATTTTTATT	UAAAAAUUUCAAAAUCUUCUA GAAGAUUUUGAAAUUUUUAUU	-12.	5.31	2 ☐	2 ☐
752-774	AAGATTTTGAAATTTTTATTAAA	UAAUAAAAAUUUCAAAAUCUU GAUUUUGAAAUUUUUAUUAAA	-9.72	7.5	1 ☐	1 ☐
758-780	TTGAAATTTTTATTAAATTATGA	AUAAUUUAAUAAAAAUUUCAA GAAAUUUUUAUUAAAUUAUGA	-7.53	-12.	2 ☐	2 ☐
777-799	ATGAGTTATTTTTATTTATTTAA	AAAUAAAUAAAAAUAACUCAU GAGUUAUUUUUAUUUAUUUAA	-10.3	4.69	1 ☐	2 ☐
779-801	GAGTTATTTTTATTTATTTAAAT	UUAAAUAAAUAAAAAUAACUC GUUAUUUUUAUUUAUUUAAAU	-7.53	-9.72	1 ☐	1 ☐

▲ **Table 2**. A list of 23 suggested siRNA candidates capable of silencing the gene IFNβ1 that have completed all four steps of Table 1. The last two columns of the table include checkboxes that the user may select to explore genes that share varying amounts of sequence similarity to the gene of interest. Similar genes represent possible off-targets that may inappropriately be silenced by the proposed siRNA candidate list and must be taken into account when designing future experiments.

SRS then finishes by saving its results to the location specified by the user in the template described above. SRS will output a JPEG version of Figure 3, as well as Mathematica and Microsoft Excel versions of the data in Table 2. The Mathematica version of Table 2 is complete, including all the potential off-targets that were found. The Excel version of Table 2 lacks the data of the potential off-targets because there is no reasonable way to organize such a large volume of information within Excel.

```
date = DateString[];

date = DateString[date,
    {"Month", "Day", "Year", "Hour", "Minute", "Second"}];

foldername = StringJoin[ToString[studyname], " - ", date];

SetDirectory[savelocationroot];

savelocationfinal = CreateDirectory[foldername];

SetDirectory[savelocationfinal];

Export[StringJoin[ToString[studyname],
    " - siRNA alignment.jpg"], Pane[graphicoutput, 801],
  ImageResolution → 300];

Export[StringJoin[ToString[studyname],
    " - siRNA options.xls"],
  MapThread[List, {masterlistprettya, masterlist[[All, 3]],
    masterlistprettyb[[All, 1, 1]],
    masterlistprettyb[[All, 1, 2]], masterlistprettyc,
    masterlistprettyd, guidepopup[[All, 1, 1]],
    passengerpopup[[All, 1, 1]]}]];

Put[masterlistprettyfinal,
  StringJoin[ToString[studyname], " - siRNA options"]];

Print[];
Print[Style["Saving data to: ", 14, Bold],
  Style[savelocationroot, 14, Bold]];
Print[];

endtime = AbsoluteTime[] - starttime;

Print[Style["Computational Time: ", 14, Bold],
  Style[endtime, 14, Bold], Style[" seconds", 14, Bold]];
```

```
Computational Time: 2456.034647 seconds
```

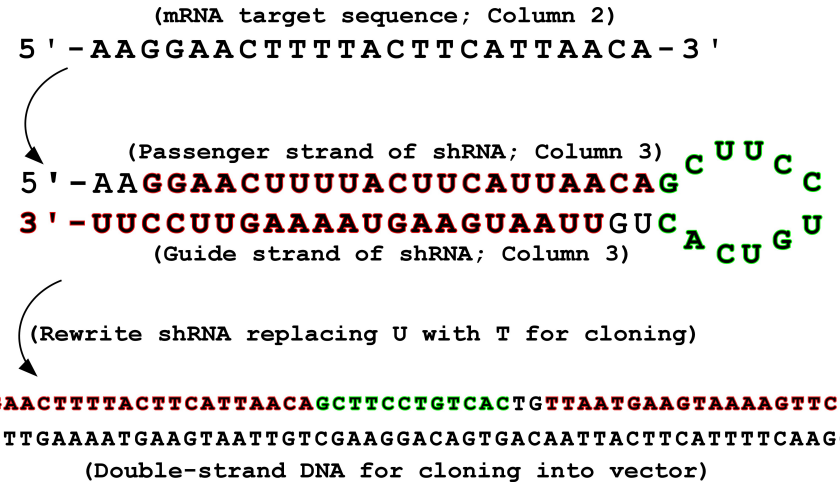
Interpreting SRS Output

Table 2 provides summary information about the siRNA candidates the algorithm is proposing as effector mediators of gene silencing. In the example of this article, the IFNβ1 gene generates 23 siRNA candidates that should be capable of effectively reducing the expression of the IFNβ1 gene; all 23 candidates have passed the four steps of Table 1. It is left to the end user to make the final decision about which siRNA sequence to use. While much is known about RNA interference and the use of siRNAs to down-regulate the expression of a gene, there are still aspects that benefit from human experience and intuition, which is why Table 2 presents information that may be contextually important to the researcher.

The first column of Table 2 conveys the positions of the siRNA candidates in relation to the target gene sequence. This also corresponds to the same graphical information presented in Figure 3. These 23-mer candidates represent subregions of the gene's transcript that are likely to be suitable targets for the guide RNA (in combination with RISC, Figure 1) to actually silence a gene. While the position of the siRNA candidate may be important to effect gene silencing, there are no universally accepted rules to develop algorithmic prediction based on position alone, which is one reason the final decision about which siRNA candidate to use is best left to the end user. Maximal suppression of gene expression by the siRNA candidate list will likely require empirical comparison of several siRNA candidates.

The second column of Table 2 provides 23-mer sequences (written 5' to 3') specific to the gene of interest to be silenced. The black box of Figure 2 highlights one example of a 23-mer from within IFNβ1. Each 23-mer represents a region of the mRNA transcript to which a guide sequence of RNA (i.e. the guide strand, Figure 1) can bind and mediate repression of protein synthesis.

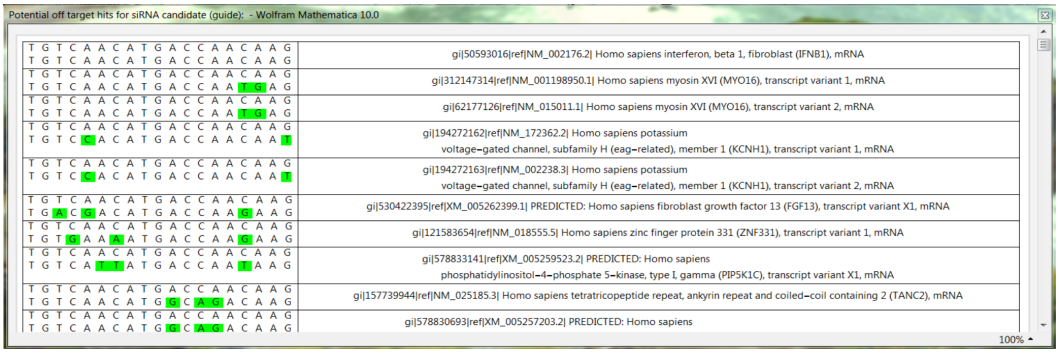
The third column of Table 3 reports the 21-nucleotide guide and passenger sequences (Figures 1 and 2) that are found originally in the small hairpin RNA (shRNA), itself manufactured from the vector referenced in Figure 1. DNA versions of these RNA sequences can be subsequently used to construct vectors for permanent or transient repression of a specific gene. As this sounds more complicated than it actually is, Figure 4 provides a graphical description of how to take the information from Table 2 and create a vector capable of using the information provided by SRS to silence a gene.



▲ **Figure 4.** Basic workflow involved in using the output of SRS to silence expression of IFNβ1. Here, the siRNA candidate at position 591–613 of the target gene (referenced in Figure 2 and Table 2) is used to compute the passenger and guide strand RNA sequences that a cell would need to manufacture from instructions contained in a suitable expression vector (Figure 1). The sequences outlined in red refer to the passenger and guide strand RNAs, respectively, while the sequence outlined in green represents one of many possible loop structures that could be designed to ensure the expression vector contains the necessary information for a cell to manufacture short hairpin RNA (shRNA, Figure 1). Where indicated, "Column 2" and "Column 3" refer to the columns of Table 2 that contain information displayed here.

The fourth and fifth columns of Table 2 report the melting temperatures (Tm) for the seed-target duplexes (Figure 2) of both the guide and passenger RNA sequences. The lower the Tm value, the more stable the seed-target duplex is. A good rule of thumb for selecting siRNA candidates, which is computationally enforced by SRS, is to use candidates that have seed-target melting temperatures below 21.5 °C. All things considered, the lower the Tm, the better.

The final two columns of Table 2 (columns 6 and 7) contain checkboxes that will, when selected, reveal the results of SRS's attempt to find similar genes that may be inadvertently silenced by the siRNA candidates under consideration. Checkboxes are used because the amount of information to be displayed can be considerable. When a checkbox is selected, a pop-up window reveals the data that was discovered. Figure 5 shows a typical result displayed when selecting a checkbox in Table 2. SRS will discard any similar gene sequences with five or more differences, as the possibility of those sequences being repressed by the siRNA sequence is virtually zero.



▲ **Figure 5.** Example output produced when the first checkbox contained in the third row of Table 2 is selected. The first column of output displays sequence alignments between the siRNA candidate chosen (in this case the 23-mer sequence positioned between nucleotides 68–90 of IFNβ1) and other gene sequences contained within the RefSeq database used by SRS. Mismatch positions are highlighted in green. The second column contains annotation information about the genes that are aligned to the siRNA candidate. As it is possible that siRNA candidates with a limited number of mismatches to other gene sequences could inappropriately silence these other genes, SRS provides critical information to allow the end user to choose the siRNA candidate that best fits their needs.

SRS Performance

To gauge the performance of SRS, two genes were chosen randomly from each of the five species the program can accommodate, and each of the genes was used as an input query to generate a list of siRNA candidates (Table 3). Timings came from running Mathematica 10 under Windows 7 (64 bit) using an Intel 2500K processor overclocked to 4.48 GHz. Total system memory is 32 GB. All reported timings use a fresh kernel.

Gene name	Gene accession	Number of siRNA candidates	Time (sec)	Memory (GB)
IFNβ1	NM_ 002 176.2	23	1098.6	25.7
POLG	NM_ 001 126 131.1	21	1046.9	24
Prss40	XM_ 006 495 859.1	18	1028.2	25.2
Kctd17	NM_ 001 081 367.1	4	317.2	9.9
March3	NM – 001 007 759.1	42	1672.7	30.8
Cxcr6	NM – 001 102 587.1	15	600	15.6
C1QL4	NM_ 845 969.3	34	759.4	19
SLITRK3	XM_ 005 639 884.1	46	1006	24.2
EML2	XM_ 006 941 101.1	11	188	6.6
ACO2	XM_ 003 989 332.2	28	419	11.7

▲ **Table 3**. Performance timings of SRS using 10 randomly chosen genes. In descending order, by groups of two, the species represented are: *Homo sapiens* (human), *Mus musculus* (mouse), *Rattus norvegicus* (rat), *Canis lupus familiaris* (dog), and *Felis catus* (cat).

Conclusion

As biologists continue to develop better methods to identify causal genes driving changes in phenotype, facile methods of altering the expression of those genes are necessary to interrupt undesirable phenotypes, most notably those of disease. SRS brings to the Mathematica community a contemporary algorithm enabling the end user to quickly design small interfering RNAs capable of suppressing a gene's output, while simultaneously minimizing off-targeting effects that have challenged earlier attempts at using siRNA as a gene suppression technology.

References

[1] E. M. Youngman and J. M. Claycomb, "From Early Lessons to New Frontiers: The Worm as a Treasure Trove of Small RNA Biology," *Frontiers in Genetics*, Nov 27, 2014. doi:10.3389/fgene.2014.00416.

[2] D. Barbosa Dogini, V. D'Avila Bittencourt Pascoal, S. H. Avansini, A. Schwambach Vieira, T. Campos Pereira, and I. Lopes-Cendes, "The New World of RNAs," *Genetics and Molecular Biology*, **37**(1 Suppl), 2014 pp. 285–293. www.ncbi.nlm.nih.gov/pmc/articles/PMC3983583.

[3] K. Gavrilov and W. M. Saltzman, "Therapeutic siRNA: Principles, Challenges, and Strategies," *Yale Journal of Biology and Medicine*, **85**(2), 2012 pp. 187–200. www.ncbi.nlm.nih.gov/pmc/articles/PMC3375670.

[4] T. Coelho, D. Adams, A. Silva, P. Lozeron, P. N. Hawkins, T. Mant, J. Perez, J. Chiesa, S. Warrington, E. Tranter, M. Munisamy, R. Falzone, J. Harrop, J. Cehelsky, B. R. Bettencourt, M. Geissler, J. S. Butler, A. Sehgal, R. E. Meyers, Q. Chen, T. Borland, R. M. Hutabarat, V. A. Clausen, R. Alvarez, K. Fitzgerald, C. Gamba-Vitalo, S. V. Nochur, A. K. Vaishnaw, D. W. Y. Sah, J. A. Gollob, and O. B. Suhr, "Safety and Efficacy of RNAi Therapy for Transthyretin Amyloidosis," *New England Journal of Medicine*, **369**, 2013 pp. 819–829. doi:10.1056/NEJMoa1208760.

[5] J. Moreno-Montañés, B. Sádaba, V. Ruz, A. Gómez-Guiu, J. Zarranz, M. V. González, C. Pañeda, and A. I. Jimenez, "Phase I Clinical Trial of SYL040012, a Small Interfering RNA Targeting β-Adrenergic Receptor 2, for Lowering Intraocular Pressure," *Molecular Therapy*, **22**(1), 2014 pp. 226–232. www.nature.com/mt/journal/v22/n1/full/mt2013217a.html.

[6] J. A. McCarroll, T. Dwarte, H. Baigude, J. Dang, L. Yang, R. B. Erlich, K. Kimpton, J. Teo, S. M. Sagnella, M. C. Akerfeldt, J. Liu, P. A. Phillips, T. M. Rana, and M. Kavallaris, "Therapeutic Targeting of Polo-Like Kinase 1 Using RNA-Interfering Nanoparticles (iNOPs) for the Treatment of Non-Small Cell Lung Cancer," *Oncotarget*, **6**(14), 2014 pp. 12020–12034. www.ncbi.nlm.nih.gov/pmc/articles/PMC4494920.

[7] Y. Naito, J. Yoshimura, S. Morishita, and K. Ui-Tei, "siDirect 2.0: Updated Software for Designing Functional siRNA with Reduced Seed-Dependent Off-Target Effect," *BMC Bioinformatics*, **10**(392), 2009. doi:10.1186/1471-2105-10-392.

[8] K. Ui-Tei, Y. Naito, F. Takahashi, T. Haraguchi, H. Ohki-Hamazaki, A. Juni, R. Ueda, and K. Saigo, "Guidelines for the Selection of Highly Effective siRNA Sequences for Mammalian and Chick RNA Interference," *Nucleic Acids Research*, **32**(3), 2004 pp. 936–948. doi:10.1093/nar/gkh247.

[9] J. Bekisz, H. Schmeisser, J. Hernandez, N. D Goldman, and K. C Zoon, "Human Interferons Alpha, Beta and Omega," *Growth Factors*, **22**(4), 2004 pp. 243–251. doi:10.1080/08977190400000833.

[10] K. D. Pruitt, T. Tatusova, G. R. Brown, and D. R. Maglott, "NCBI Reference Sequences (RefSeq): Current Status, New Features and Genome Annotation Policy," *Nucleic Acids Research*, **40**(D1), 2012 pp. D130–D135. doi:10.1093/nar/gkr1079.

[11] K. Ui-Tei, Y. Naito, K. Nishi, A. Juni, and K. Saigo, "Thermodynamic Stability and Watson–Crick Base Pairing in the Seed Duplex Are Major Determinants of the Efficiency of the siRNA-Based Off-Target Effect," *Nucleic Acids Research*, **36**(22), 2008 pp. 7100–7109. doi:10.1093/nar/gkn902.

T. D. Allen, "A Computational Strategy for Effective Gene Silencing through siRNAs," *The Mathematica Journal*, 2016. dx.doi.org/doi:10.3888/tmj.18-1.

About the Author

Todd Allen is an associate professor of biology at HACC, Lancaster. His interest in computational biology using Mathematica took shape during his postdoctoral research years at the University of Maryland, where he developed a custom cDNA microarray chip to study gene expression changes in the chestnut blight pathogen, *Cryphonectria parasitica*.

Todd D. Allen, Ph.D.
Harrisburg Area Community College (Lancaster Campus)
East 206R
1641 Old Philadelphia Pike
Lancaster, PA 17602
tdallen@hacc.edu