SiteOut: user manual

DePace Lab

June 1, 2015

Contents

1 Introduction

This tool allows to remove specific nucleotide motifs, such as transcription factor binding sites, from a DNA sequence. It can be used to to design a new sequence from scratch, refine a predefined sequence, or to create motif-free spacers between functional sequences. The code will either look for explicit motifs or will use Patser to check for predicted motifs based on the chosen position weight matrices (PWMs). Once you click on 'Submit', your request will be sent to Orchestra, Harvard Medical School's high-performance computing environment, where it will run for a while depending on its characteristics. The output will be sent to you by email as soon as it is ready.

The tool is presented in REFERENCE, so we ask you to cite this reference if you have used it in your research.

2 Main options

To use the tool you have to follow three steps: specify the characteristics of the sequence you want to design, determine which motifs you want to avoid in your sequence, and provide an email where the results will be sent.

2.1 Step 1: sequence design

Random sequence

This option allows to design a random motif-free sequence of arbitrary length. The program will start with a random sequence and gradually remove motifs from it. You just have to specify the length of the sequence.

Refine a sequence

This option allows to remove motifs from a given sequence. The sequence can be entered directly into the text box or uploaded as a single-sequence fasta file Figure(1).

Spacer designer

This option allows to create motif-free spacers between functional sequences. A 'sequence design' file with the details of the sequence to be created has to be uploaded: it has to be a plain text file with functional sequences and motif-free spacer lengths intercalated forming the desired pattern. It can start and end with a spacer or with a functional sequence, and two consecutive sequences can be given; in that case no spacer will be created there (1).

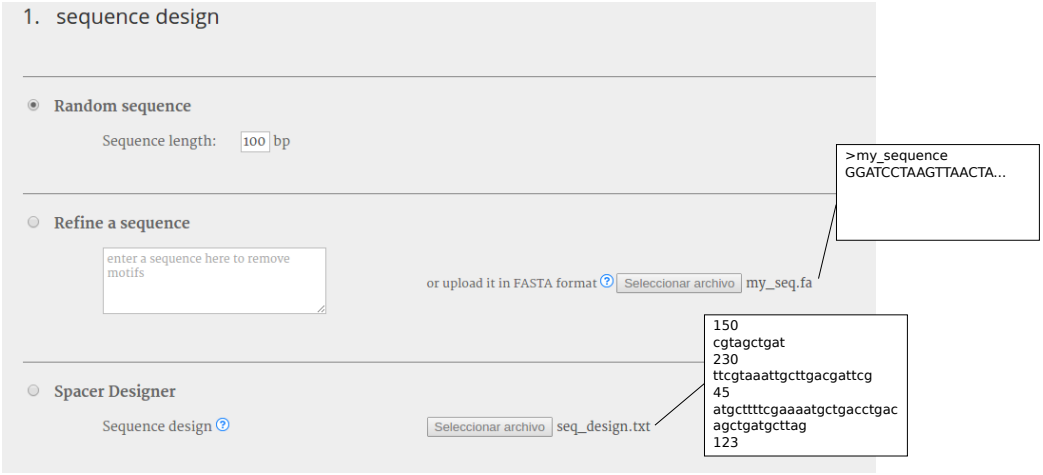

Figure 1: Screenshot of the 'sequence design' section, highlighting the content of the fasta format ('my_seq.fa') and sequence design ('seq_design.txt') files.

2.2 Step 2: motifs to avoid and GC content

Here you must provide information about the motifs to be removed from sequences or spacers chosen in the previous section. You have to upload a zipfile with the position weight matrices (PWMs) for the motifs of interest and/or a list of explicit motifs Figure(2). Patser will be used to find motifs based on PWMs with a cutoff threshold given by the P value.

where to find PWMs

The GC content is the percentage of Gs and Cs in the sequence, so must be a number between 1 and 100. Most natural occurring DNA sequences have GC content values of between 30% and 50%. The tool uses it to create initial random sequences or spacers, and to decide the probability of choosing G or C when mutations are introduced to remove motifs.

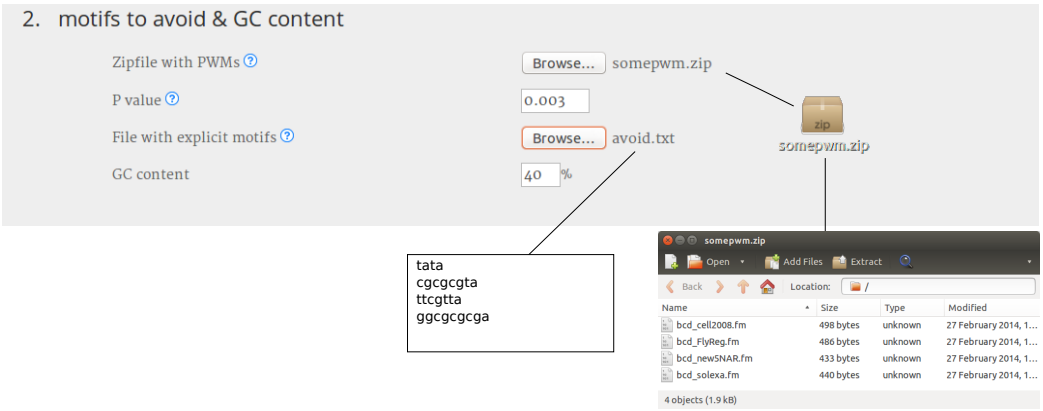

Figure 2: Screenshot of the 'motifs to avoid & GC content' section highlighting the content of the zipfile with PWMs and the format of the file with motifs to avoid.

2.3 Step 3: submission

The results will be sent to the email provided here. First, once the job starts running in the cluster you will receive a confirmation email with all the information regarding your request, as well as a confirmation number that you can use if you need to contact us. Once the job is done, you will receive a second email with the results: it will include the designed sequence and a .csv file that has information about the motifs in the final sequence (position and strength). The .csv file can be visualized using InSite (http://www.cs.utah.edu/~miriah/insite/).

3 Examples

The three options described above confer the tool with enough flexibility to help create most sequences where careful motif removal or manipulation is required. Here we describe four examples that show how the different sequence-design options can be combined to create complicated sequences.

3.1 Generating large sequences by merging smaller ones

When designing very long motif-free sequences, it is convenient to parallelize the job to save time and avoid hitting the 12 hour wall time. For example, to design a 10 kb sequence, it is much faster to submit five independent jobs using the *Random sequence* option, each designing a different 2 kb piece. These five sequences can be connected together and the resulting sequence refined using the *Refine a sequence* option, which will remove any motifs that formed at the junctions between them Fig(3).

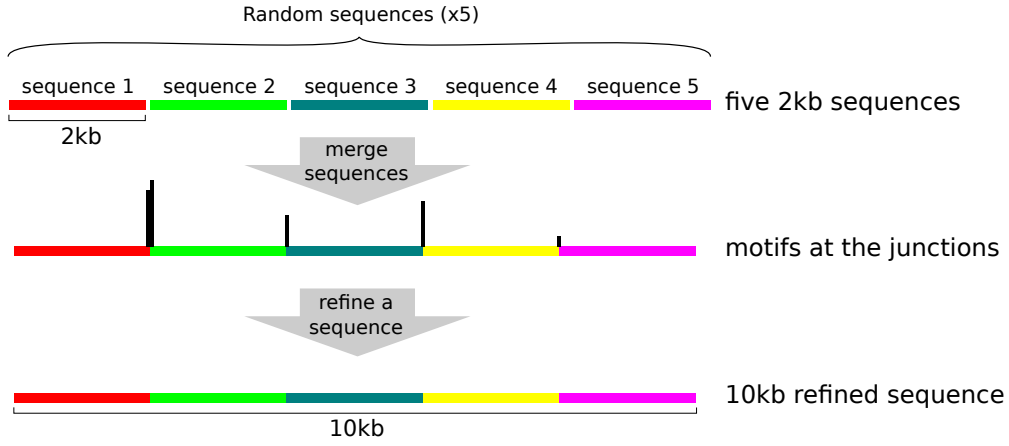

Figure 3: To generate a 10 kb sequence we ask SiteOut to create five 2 kb random motif-free sequences and we then merge them. Sticking sequences together creates motifs in the junctions, so we later ask SiteOut to refine the overall sequence.

3.2 Removing motifs in a hierarchical order

While SiteOut does efficiently remove motifs, there may be situations where the number of motifs to be removed is so high that no solution exists or, if it does, is extremely difficult to find. SiteOut will return the best sequence it has been able to find, but this will not have zero motifs. In these situations one may want to prioritize the removal of some motifs over the rest. This can be easily done in different steps Fig(4).:

1. Create a random sequence where only the least important motif is removed.
2. Refine the initial sequence step by step, removing one type of motif at a time, in order of increasing importance.
3. In the last step remove the motifs that you want by all means to be avoided in your sequence.

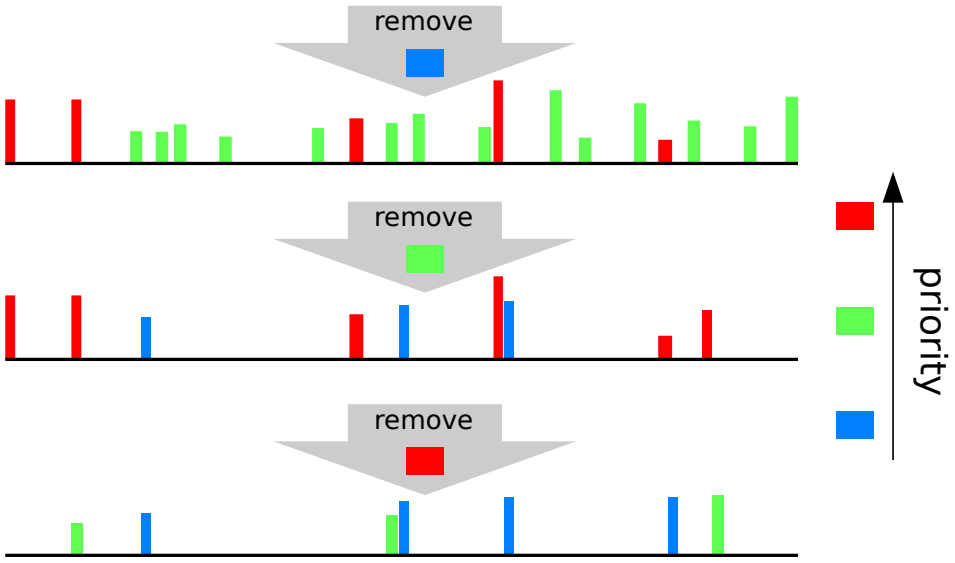

Figure 4: Removing motifs in a hierarchical fashion. We want to get rid of all red motifs and the majority of the green ones, while we do not care that much about the blue ones. We take three steps, removing first the blues (top), then the greens (middle) and finally the reds (bottom). In each step, the removal process may create new motifs of different type, but the most relevant ones are deleted in the subsequent steps

3.3 Separating functional sequences with motif-free spacers

The *Spacer designer* option enables to control the motif content of random spacers created in between functional sequences, while leaving the functional sequences themselves untouched. This function is particularly useful for designing synthetic enhancers in which it is desirable not only to have the spacers between clusters of transcription factor binding sites be 'neutral' but also to avoid any creation of binding sites at the junctions Fig(5).

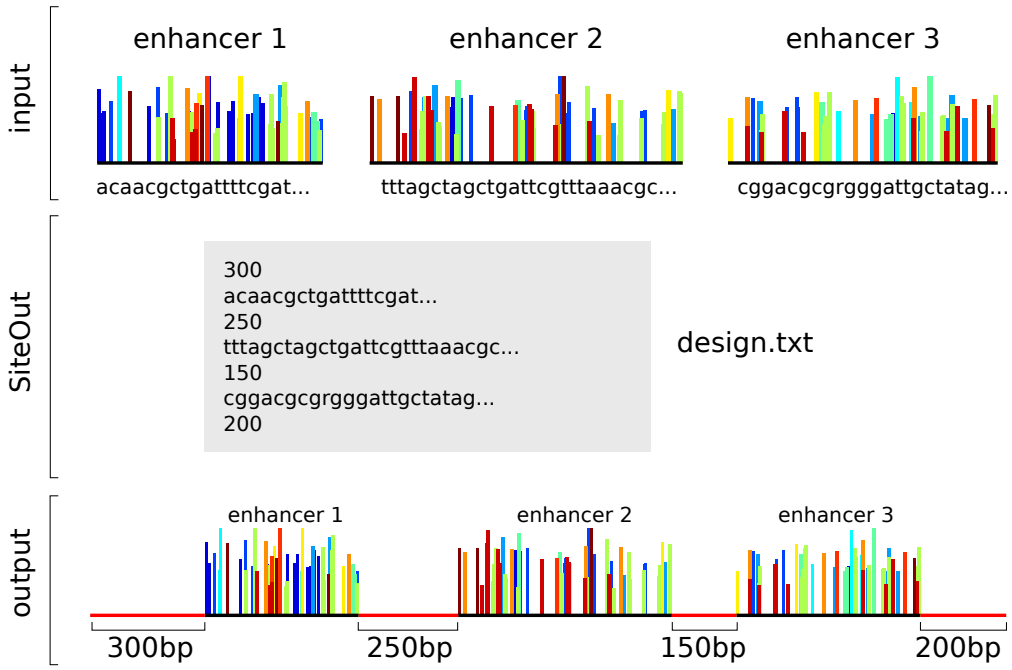

Figure 5: We want to create a construct where three enhancers are separated by spacers of different lengths. Colored bars state for binding sites along the sequences. We give SiteOut a 'design.txt' file with the desired arrangement of sequences. The tool sends us a sequence where enhancers are separated by spacers (red) with no motifs in them.

3.4 Adding transcription factor binding sites

Adding transcription factor binding motifs to a backbone of motif-free sequence can be done using, again, the *Spacer designer* tool. In this case, the transcription factor binding motifs to be inserted are the functional sequences that will be untouched, and the motif-free spacers between them will ensure that no additional binding sites are created in the junctions Fig(6).

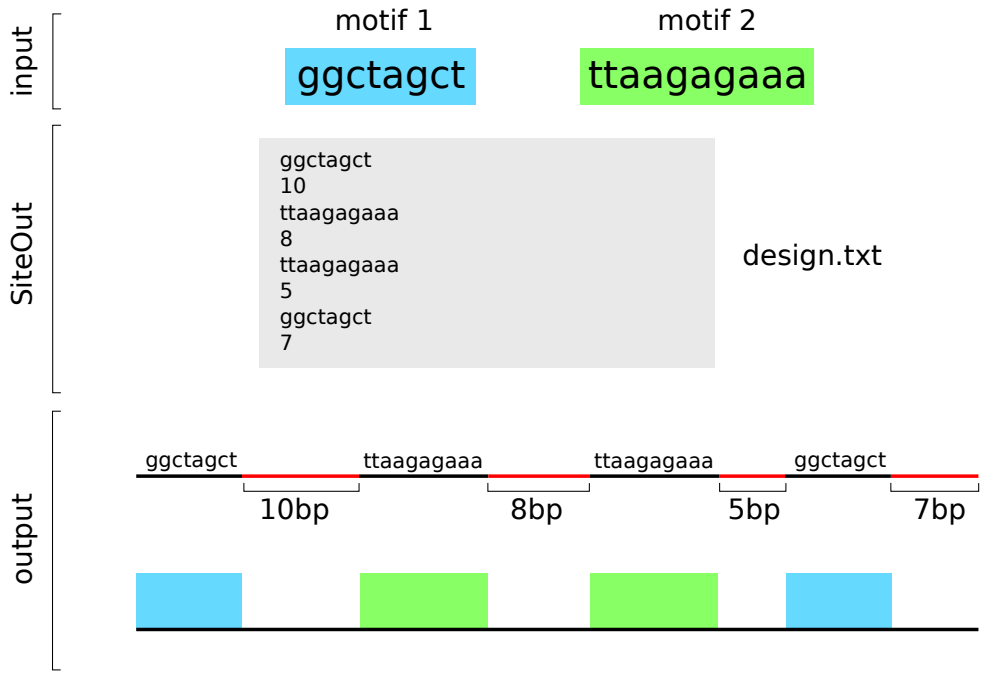

Figure 6: We want to create a construct where two transcription factor binding motifs (blue and green) are embedded into a motif-free backbone. We give SiteOut a 'design.txt' file with the desired distribution of binding sites within the motif-free backbone. The tool sends back a sequence with the desired binding motif pattern (blue, green, green, blue) inserted into a motif-free backbone (red).

3.5 Reusing an already synthesized motif-free sequence

Synthesizing sequences is not cheap and so users may want to reuse a motif-free spacer that has already been made. This can be achieved without compromising on motif generation at the junctions between functional and spacer sequences by setting the synthesized DNA as a "functional sequence" in the *Spacer designer* option so that it won't be touched by the algorithm. Short 10 bp spacers can then be designed to link them without creating new TFBMs; these can easily be incorporated into primers for cloning Fig(7).

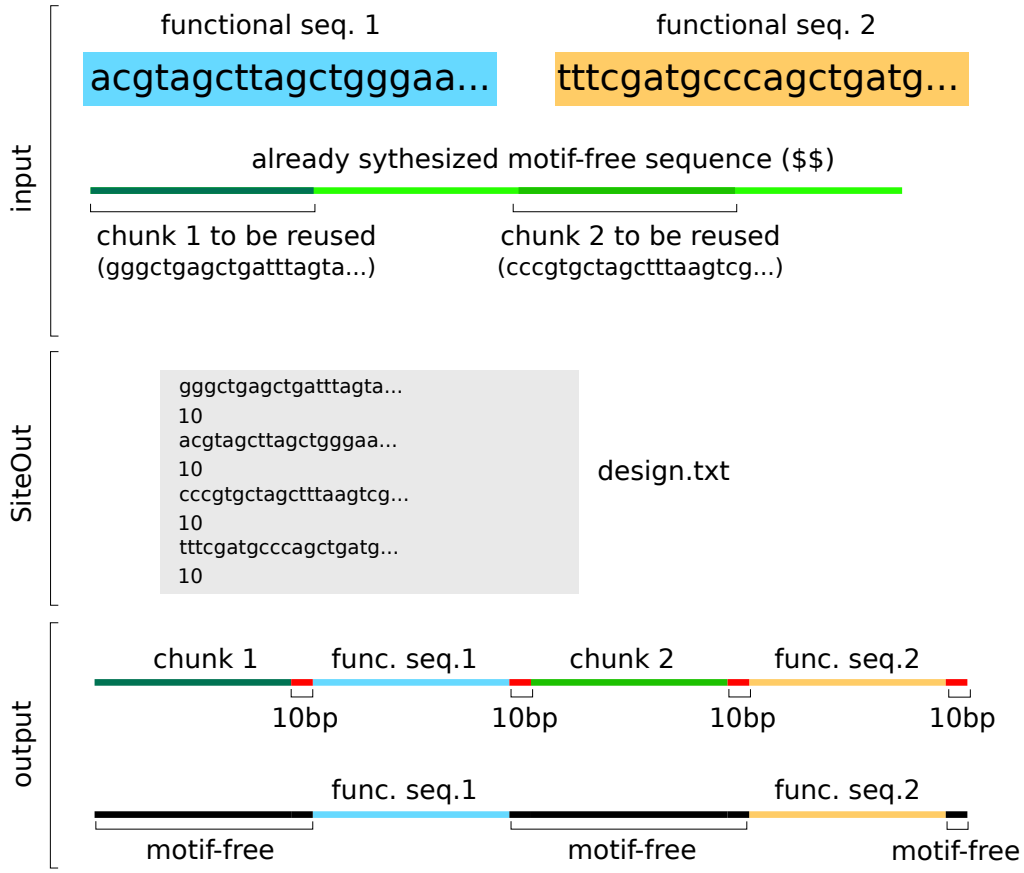

Figure 7: We want to create motif-free spacers between two functional sequences (blue and orange) reusing an expensive motif-free sequence we had already synthesized (green). We first select the chunks of the synthesized sequence we want to use (darker greens). We then give SiteOut a 'design.txt' file with the functional sequences to be untouched: the first chunk, the first functional sequence, the second chunk, and the second functional sequence, intercalated by small 10 bp linkers (red) that can be (red) easily cloned using primers. SiteOut returns a sequence where the two functional sequences are embedded into a motif-free backbone that we do not need to synthesize from scratch.

4 Technical information

SiteOut runs in Harvard Medical School's cluster, and has a wall time of 12 hours, which should be enough for the majority of requests. Figure(8) shows runtimes as a function of the number of PWMs for the design of 300 bp sequences with a P value of 0.003. The trend is exponential, being Patser the slowest step in each iteration. Not all the nodes in the cluster are equally fast, so when a job is submitted its runtime depends on where it ends up running.

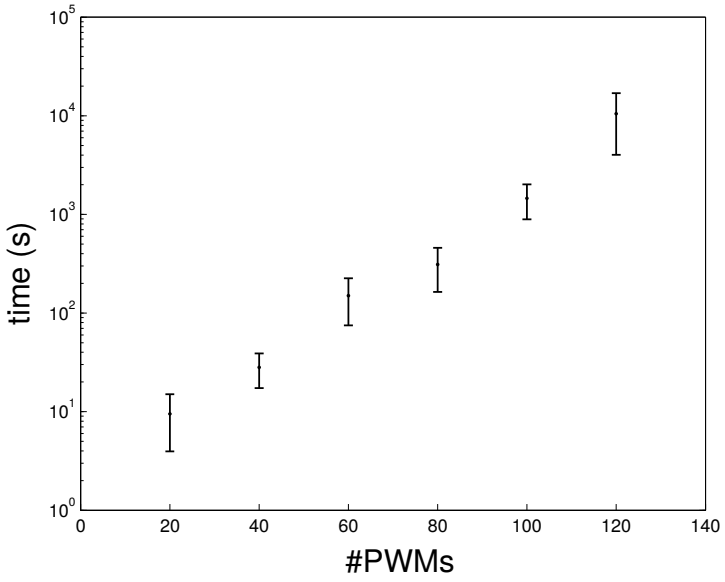

Figure 8: Benchmark of SiteOut running in Harvard Medical School's cluster. Design of 300 bp random sequences, P value of 0.003. For 140 PWMs the 12 hour wall time is always reached.

If you have a more complicated task and the 12 hour wall time is not enough (not even splitting the problem in multiple jobs -see first example above-), or if you wan to implement new functionalities for more specific problems, there is a standalone version of SiteOut you can download from the website. It has been tested in Linux and OSX, but should be easily implemented in Windows. The Monte Carlo algorithm is written in python, and requires Patser to be compiled, and the path to Patser to be correctly set. Patser can be downloaded from Gary Stormo's lab website `http://stormo.wustl.edu/resources.html`.