A method for building a CRISPR library

FLASH is a biochemical method for enriching for certain sequences of nucleic acid present in a sample. Blocked DNA is treated with Cas9 and a targeted library of CRISPR guides. Any fragments of DNA that result from a pair of cuts (one on each end) may be amplified and then sequenced, so long as the fragments are of a certain length.

A suitable multiplexed library of guides must satisfy the following requirements.

1. The guides must not cut the wrong sequences. For a given set of experiments, there will be some known background (eg host DNA) that you do not wish to cut, both because that will cause amplification of the wrong thing and because it will deplete the Cas9.
2. The guides must cut the right sequences into fragments of the right length. For any sequence of DNA that you want to amplify, there must be guides in the library whose targets are somewhat evenly spaced (eg 200-500 bp apart) along the sequence.
3. The guides must not overlap with known SNPs or variable bases. (Otherwise, they may fail to cut.)
4. The library of guides should be as small as possible given constraints 1-3.

Because one CRISPR guide can cut multiple target sequences, condition (2) can be difficult to satisfy. By selecting guides that optimally tile one target sequence, you may inadvertently shred another target sequence into confetti.

To see this combinatorial phenomenon clearly, consider a graph whose nodes are target sequences. Draw an edge between each pair of target sequences for which there is a CRISPR guide that would cut both of them. The library must be constructed for each connected component of the graph in a coordinated way; for closely related families of target sequences (eg: Beta Lactamase genes) there may be hundreds of genes that must be considered simultaneously.

Input data

The input to the computation is

1. A set of target sequences, $\mathcal{T}$.
2. A set of off-target sequences, $\mathcal{O}$.
3. An quality scoring function for guides (for eg secondary structure and GC content) q.
4. Maximum $F_{\max}$ and minimum $F_{\min}$ acceptable lengths for fragments.

Let $\mathcal{G}$ be the set of all guides that have target sites in $\mathcal{T}$.

Integer programming formulation

We formulate the library design problem as an integer program.

For each $g \in \mathcal{G}$, let X_g be a binary variable representing the membership of g in our final library. We begin by setting $X_g = 0$ for each guide g with one of the following problems:

1. The guide g cuts an off-target sequence in $\mathcal{O}$.
2. The guide g has a quality score greater than some threshold T.
3. The guide g would cut some target t on a site known to include a SNP.

For each target sequence $t \in \mathcal{T}$, there is a list of CRISPR cut sites $\text{cuts}(t) = \{(p_i, g_i)\}$, where guide g_i cuts t in position p_i. We add the following constraints:

1. Cut at least twice:

$$\sum_i X_{g_i} \geq 2.$$

2. No confetti:

$$X_{g_i} + X_{g_j} \leq 1 \text{ if } |p_i - p_j| < F_{\min}.$$

We then maximize the efficiency of the library satisfying the above criteria:

$$\max 1.1 \cdot \sum_{g \in \mathcal{G}} \sum_{t \in \mathcal{T}} \delta(g, c) X_g - \sum_{g \in \mathcal{G}} X_g,$$

where $\delta(g, c) = 1$ if g cuts t and 0 otherwise. This term is equal to the 1.1 times the total number of cuts made by the library minus the size of the library.

This objective tries to tile the guides as closely as possible while staying above the minimum fragment size, rewarding the use of guides that cut in many places and punishing a large library size.

The integer program can then be solved using an off-the-shelf solver, such as GLPK or gurobi.

Extensions

The integer programming framework described above is very extensible; it's easy to modify constraints or the objective function to suit a particular design problem.

Capturing SNPs

If there are certain regions of variability one wants to capture, eg, clinically relevant SNPs, then we add a constraint that each SNP be contained in a fragment of a suitable length.

To formulate this as an integer constraint, consider a SNP at position k in target sequence t. Let $I_1 = [k - F_{max}, k]$, $I_2 = [k, k + F_{max}]$, $I_3 = [k - 150, k + 150]$. Let $(p_i, g_i) \in (cuts)(t)$ be the set of cut sites in t, as before.

These two constraints guarantee that the SNP will be in a fragment of an appropriate length:

$$\sum_{i:p_i \in I_1} X_{g_i} \geq 1$$

$$\sum_{i:p_i \in I_2} X_{g_i} \geq 1$$

This constraint guarantees that the SNP will be covered by a 150 bp Illumina read:

$$\sum_{i:p_i \in I_3} X_{g_i} \geq 1.$$

Quality scores for guides

In the formulation above, we use hard cutoffs for guides. Each guide is either forbidden (for off-targets or structure) or equally valuable.

It is straightforward to use a quantitative quality score $q(g)$. One merely adds a term:

$$\sum_{g \in \mathcal{G}} q(g) * X_g$$

to the objective function.

Library extension

Given a partial library of guides $\mathcal{L}$, one might wish to extend it to a new set of targets. That can be accomplished in a hard way, by forcing those guides to be included in the new library,

$$X_g = 1 \text{ if } g \in \mathcal{L}.$$

It can also be accomplished in a soft way, by adding an additional term

$$\lambda \sum_{g \in \mathcal{L}} X_g$$

to the objective function.

Dual formulation

The current framework is maximization: maximize the total number of cuts while minimizing the library size. It sets up a direct tradeoff: adding a guide to the library is always worthwhile, so long as it does not.

There is a dual formulation involving minimization, where we make the coverage constraints explicit (cuts every k basepairs along each target) and then try to minimize the library size.