User manual

Background selection maps

1. Main function:

The program is run from the command line and has the following usage:

```
calc_bkgd <config_file> <out_token>
```

The main function reads several parameters from a `config` file, including the chromosome name and length, the path to the genetic map file, the path to the annotation file, selection parameters and other technical run parameters and flags (see annotated example `config` file in this directory). Once the data inputs are loaded, the program carries out some initial pre-calculation steps to facilitate the main calculations carried out across the chromosome (detailed below). With these initial steps in place, the `calc_bkgd` program creates a map of the effects of background se- lection at each position in the genome by estimating these effects one position at the time. In practice, there is a precision parameter (called `BKGD_SCALE` in the `config` file) that determines the format of the output. Because the effects of background selection generally do not change substantially from base-to-base, the program records contiguous tracts of the chromosome within which b changes by less than the precision set by `BKGD_SCALE`. In the example output below, the first column gives the b value and the second column gives the length of the corresponding tract:

```
 85 812345
109 79024
205 21532
```

The first 812345 bases correspond to a b value of 85, etc.

2. Data processing and precalculations

The genetic map file (called `RECOMB_RATE_TABLE` in the `config` file) is stored in a struct that can convert physical coordinates to genetic map coordinates. If the input genetic map does not cover the whole chromosome (i.e., starts at a physical position > 1 or ends at a physical position < the chromosome length) then

recombination rates are extrapolated backwards to position 1 and forwards to the end of the chromosome using the recombination rates in the first and last genetic map bins, respectively.

The selected annotations file (called `CONS_TABLE` in the `config` file) is used to create a linked list of structs storing the physical and genetic map coordinates of each annotation segment. The program sums the effects of each of these segments to estimate the effects of background selection at each point in the genome. In practice, the program employs several shortcuts (described below) so that this sum is only calculated for a subset of sites.

For the deleterious mutation rate, selection coefficient and distribution of selection coefficients (respectively called `PARAM_U, PARAM_T` and `PARAM_T_DIST_TYPE` in the `config` file), two sets of b values and their 1st and second derivatives `b, b_drv1, b_drv2` are calculated and stored in a lookup table for the effects of selected regions on neutral sites across a logarithmic range of n genetic distances r : $\{r_0, ..., r_n\}$. The two tables are constructed in the following ways:

Point table: The first set of `b, b_drv1, b_drv2` values give the effect of a selected *point* x on a neutral point y for genetic distances r : $\{r_0, ..., r_n\}$ between x and y. The result is an $n \times 1$ table.

Segment table: The second set of `b, b_drv1, b_drv2` values give the effect of a selected *segment* x on a neutral point y for genetic distances r : $\{r_0,...,r_n\}$ between x and y but now for a given genetic distance r_i, calculate `b, b_drv1, b_drv2` for segments of a range of genetic lengths from c : $\{c_0, ..., c_n\}$. The result is an $n \times n$ table where each row represents the genetic distance to the segment and each column represents the genetic length of the segment.

NOTES: range c is identical to range r

3. Calculating genome-wide effects of background selection

Once the annotations, genetic map and pre-calculated tables of `b, b_drv1, b_drv2` have been loaded, the program begins calculating the effects of background selection at every position in the chromosome. To speed up the calculations, the program only calculates *exact* b values b_i, b_j at points p_i, p_j, where p_i, p_j are selected by a search algorithm requiring that $|b_i - b_j| < 2\varepsilon$ (the value of 2ε is set by the parameter `BKGD_CHANGE_THRESH` in the `config` file). For positions p_x where

$p_i <= p_x <= p_j$, b values are estimated using linear interpolation with slope $(b_j - b_i)/(r_j - r_i)$, where r_i, r_j are the genetic map coordinates of points p_i, p_j. The "exact" b values b_i, b_j are calculated by summing the contribution of selected points and segments at p_i and p_j using the b value lookup tables (described above) in the

following ways:

Point contribution: For a selected point x_{point} at genetic distance r_X from p_i, get the nearest table values r_1, r_2 above and below r_X where $r_1 = max(r < r_X)$ and $r_2 = min(r > r_X)$, respectively. Use the corresponding b values at these points $b(r_1)$, $b(r_2)$ to approximate the contribution to b at p_i due to x_{point} by the following linear interpolation:

$$b_x = b_1 + r_x \cdot \frac{b_2 - b_1}{r_2 - r_1}$$

Segment contribution: For a selected *segment* $x_{segment}$ with genetic length $c_X > 0$ at genetic distance r_X from p_i, the program gets the nearest table rows r_1, r_2 above and below r_X, respectively (as described above) but now for each row also find the nearest table columns c_1, c_2 above and below c_X, where $c_1 = max(c < c_X)$ and $c_2 = min(c > c_X)$, respectively. The b value $b_{1,1} = b(r_1, c_1)$ corresponds to the effect of a segment with genetic length c_1 which begins at genetic distance r_1 from selected point p_i. Estimate the contribution to b at p_i due to $x_{segment}$ with the following bilinear interpolation:

$$b_x = \frac{b_{1,1}(r_2 - r_x)(c_2 - c_x) + b_{2,1}(r_x - r_1)(c_2 - c_x) + b_{1,2}(r_2 - r_x)(c_x - c_1) + b_{2,2}(r_x - r_1)(c_x - c_1)}{(r_2 - r_1)(c_2 - c_1)}$$

NOTES:

- Each "exact" evaluation of b the program uses the same strategy calculate the 1st and 2nd derivatives using their respective lookup tables. Use of the 1st and 2nd derivatives is explained below.
- b_X is multiplied by the segment's recombination rate in M/bp to scale the contribution to b by number of selected sites in the segment.
- Contribution to b from a selected segment $x_{segment}$ with genetic length 0 ($c_X = 0$) at a genetic distance r_X from p_i is calculated identically to a selected **point** at genetic distance r_X except that the total contribution b_x is multiplied by the length of the segment in bp.

4. Setting a new endpoint for the interpolation window:

When the end of a given interpolation window p_i, p_j is reached, the next window is started with the current endpoint p_j used as the next start point. The next end point is found using a search algorithm with an optional set of constraints. Search

constraints increase background selection map precision under certain conditions but may also dramatically increase program run time depending on specific factors discussed below. The steps of the basic, unconstrained algorithm are as follows:

1. Reset window: Set next window start point to previous window end point in both physical and genetic maps units: $p_i, r_i = p_j, r_j$

2. Reset b values: Set b, b_drv1 and b_drv2 at the next window start point to values from previous window end point: $b_i, b^{'}, b^{''} = b_j, b^{''}, b^{''}$

3. Extrapolate distance to new end point: Use a second order Taylor approximation in terms of $b_i, b^{'}, b^{''}$ and r_i to extrapolate the distance δ_r to the nearest genetic map position r_x where $|b(r_x) - b_i| = \varepsilon$ (the value of ε is set by the parameter `BKGD_OFFSET` in `config` file). Solve for δ_r from the quadratic:

$$\pm\epsilon = b_i'(\delta_r) + \frac{1}{2}b_i''(\delta_r)^2$$

4. Keep minimum acceptable distance: As a precaution, impose a maximum step size in genetic map units, r_{max}, (r_{max} is set by the parameter `BKGD_INTERP_MAX_DIST` in `config` file). Only consider next end point values larger than the current start point (otherwise the interpolation window would have negative width). The next genetic map end point r_j for the new interpolation window is given by: $r_j = r_i + min\{r_{max}, min(\delta_r > 0)\}$

5. Find the physical end point: To find the physical map end point p_j corresponding to the genetic map end point r_j, start with the variable $p_x = p_i + 1$. Increment p_x one base at a time and check its current genetic map position for each increment using the function F_{gmap} (where $F_{gmap}(x)$ gives the genetic map position of physical chromosomal position x). When $F_{gmap}(p_x) > r_j$, move p_x back a single base. In the limiting case that this happens after the first increment, $p_i + 1$ is the final value of p_x and the interpolation window will only span a single base.

6. Keep or reject new end point: Use the lookup tables to estimate the effects of background selection at the new end point b_j (descibed in section 3 above). If $|b_i - b_j| \geq 2\varepsilon$, reduce the distance to the next end point by 1/2 and return to step 5:

$$r_j = r_i + \frac{1}{2} min\{r_{max}, min(\delta_r > 0)\}$$

If $|b_i - b_j| \leq 2\varepsilon$, use the new end point for the interpolation window and exit the search algorithm.

The constrained version of the algorithm imposes the following conditions:

- For a given interpolation window start point in genetic map units r_i, consider the genetic map coordinates of the nearest selected annotation region to the left and right, r_L, r_R, respectively.
- If the point has a selected region to its left *and* right, calculate the genetic map midpoint $r_{1/2}$ between the end of the left region and the start of the right region: $r_{1/2} = r_L + 1/2\ (r_R - r_L)$
- If the point is towards the start of the chromosome, there may only be selected regions to the right: calculate the genetic map midpoint between position 1 and the first selected region start point.
- If the point is towards the end of the chromosome, there may only be selected regions to the left: calculate the genetic map midpoint between the last selected region end point and the final position in the chromosome.
- Perform steps 1-3 of the search algorithm and modify step 4 for the following conditions:
    - If $r_i < r_{1/2} : r_j = min\{r_{1/2}, r_{max}, min(r_x > r_i)\}$
    - If $r_i \geq r_{1/2} : r_j = min\{r_{Ri-1}, r_{max}, min(r_x > r_i)\}$, where r_{Ri-1} is the genetic map position 1bp to the left of r_{Ri}.
    - Continue to step 5.
- If the previous endpoint result was $r_j = r_{Ri-1}$, the *next* endpoint is automatically set to $r_j = r_{Li+1}$, where r_{Li+1} is the genetic map position 1bp to the right of r_{Li}.

NOTES: When $|b_i - b_j| \geq 2\varepsilon$ and the new point is rejected, the rejected point is saved in a queue - After step 3, the queue is checked for the latest rejected point (if any) - If the last queue point was closer than the new extrapolated point, queue point is used instead and popped from the queue -- this minimizes the expensive calculation of b using lookup tables in step 6, since the calculation was already done on saved points

Classic Selective Sweep Maps

0. Outline of the `css_map` program:
TODO: brief theoretical intro w/ overview? Or just refer to theory section of paper?

1. Main function:

The program is run from the command line and has the following usage:

```
css_map <config_file>
```

The `config` file path read by is the same file read by the `calc_bkgd` program, with many overlapping fields like `RECOMB_RATE_TABLE` used for both programs as well as several new fields for specific inputs used in calculating the effects of classic sweeps.

2. The `CSCalculator` class:

The `css_map` program is controlled by a class called `CSCalculator` which stores an internal representation of relevant data for estimating genome wide effects of sweeps and a suite of member functions used in each step of these estimates. The main function used for the purpose of pre-calculation in the inference pipeline is the `fixrate_blocks` function, which yields a map of the effects of classic sweeps for a given set of a putative selected substitutions formatted just like the map of the effects of background selection given by the `calc_bkgd` program described above. One column of the sweep map gives the average increase in the coalescent rate due to sweeps (the effect of CS in other words), the other column gives the length of the chromosome along which this average effect occurs.

3. Components of the `fixrate_blocks` function:

The `fixrate_blocks` function calls a number of helper functions needed for each step to generate a map of the effects of CS described above. These steps are described here:

1. Generate a grid of genetic positions where effects of CS are expected to change by less than 2ε, (where ε is a precision parameter called `CSS_SCALE` in the `config` file, analogous to `BKGD_SCALE`).
2. Generate exact values of the effects of CSS at each position in the grid from step 1: given the genetic map, annotated putative adaptive substitutions,

selection coefficient and effective population size (specified by `RECOMB_RATE_TABLE` and `ANNO_PTS, PARAM_S, PARAM_NE` in the `config` file), calculate the effects of CSS in terms of increase to the coalescent rate due to selection (where neutral rate = 0).

3. Using this grid of exact values and corresponding genetic map positions, iterate over every position in a specified chromosome and estimate the effects of CSS at that point using linear interpolation in terms of the genetic map. For each step where CSS changes by less than ε, increment a variable for length by 1 and store the current CSS value to compare with the next position. When CSS changes by more than ε at a given step, record the cumulative length and previous CSS value as a segment-estimate pair and resent the length variable to 0. Continue this process until the entire chromosome has been segmented.

NOTE: The precision parameters for maps of BS and CSS should be set to the same value for consistency when running the joint inference.