General overview of the algorithm

The algorithm is composed of the following parts:

- **main.r** : main script. It declares the variables, reads the data, calls the other scripts, iterates through generations and finally plots the data.
- **kalman_filter.r** : it contains the algorithm of the Kalman filter explained in Appendixes A and B.
- **find.rho.r** : this script makes predictions with a given rho inside the moving time window, and compares predictions with the observed changes inside the window. This script calls the kalman_filter.r
- **predict.r** : this script makes the actual prediction for the response to selection, using the rho obtained with find_rho.r. This script calls the kalman_filter.r

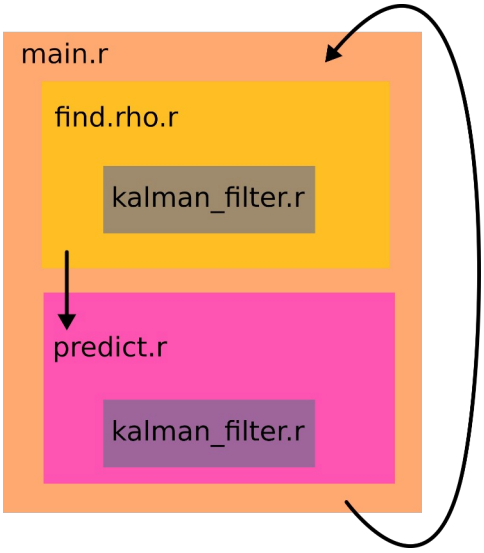

Figure 1. The algorithm is composed of the scripts main.r, find.rho.r, predict.r and kalman_filter.r

The moving time window

Here we will explain the use of the moving time window which is central to the method. As it is explained in the text, we use the moving time window to find the parameter rho that minimizes the prediction error. This is done in every generation *i*, and the moving window is composed of the generations from $i - L.dyn + 1$ to *i*, where *L.dyn* is the dynamical size of the time window (i.e. $L.dyn=L$ if $i>=L$ and $L.dyn=i$ if $i<L$, where *L* is the maximum size of the window).

The Kalman filter is used recursively inside the time window to obtain estimates for each generation inside the window. The estimate obtained in the last element of the time window (i.e. generation *i*) is the prediction for the mean of the traits in the next generation. When used in the find_rho.r subroutine, the predictions for all generations inside the window are compared with the observed changes inside the window. The rho that minimizes this difference is the one used in the predict.r subroutine.

To begin the recursion inside the time window, and because the Kalman filter is recursive, we need initial estimates of the states and initial estimates of error covariance matrices. For the first generation, the initial estimates of the state variables are the breeder's prediction and zero for the bias, and the initial estimate for the error covariance matrix is the zero matrix. For all other generations, the initial estimate used is taken from the first recursion of the last time window. This is done because the result of the first recursion of the window for *i* is the first element of the window for $i+1$ (see Figure 2). To store these values used to initialize the time window in the next generation, two matrices are used, namely *PP* and *XX* (see Figure 2). Matrix *PP* stores the error

covariance matrices for each trait (size *2nt x 2*, where *nt* is the number of traits). Matrix *XX* stores the estimate of the states for all traits (size *2nt x 1*).

The logic above is used every time the Kalman filter is used. That is, both inside the find_rho.r and predict.r subroutines. The only difference is that the in the find_rho.r subroutine, multiple values of rho are tested to find the one that minimizes the prediction error inside the window. In the predict.r subroutine, only this value of rho is used. The estimate of the states at generation i of the predict.r subroutine is the one used as prediction of our method since it is the one with the correct rho. Further, it is inside the subroutine predict.r where the values of *PP* and *XX* are updated, to be used in the next time window.

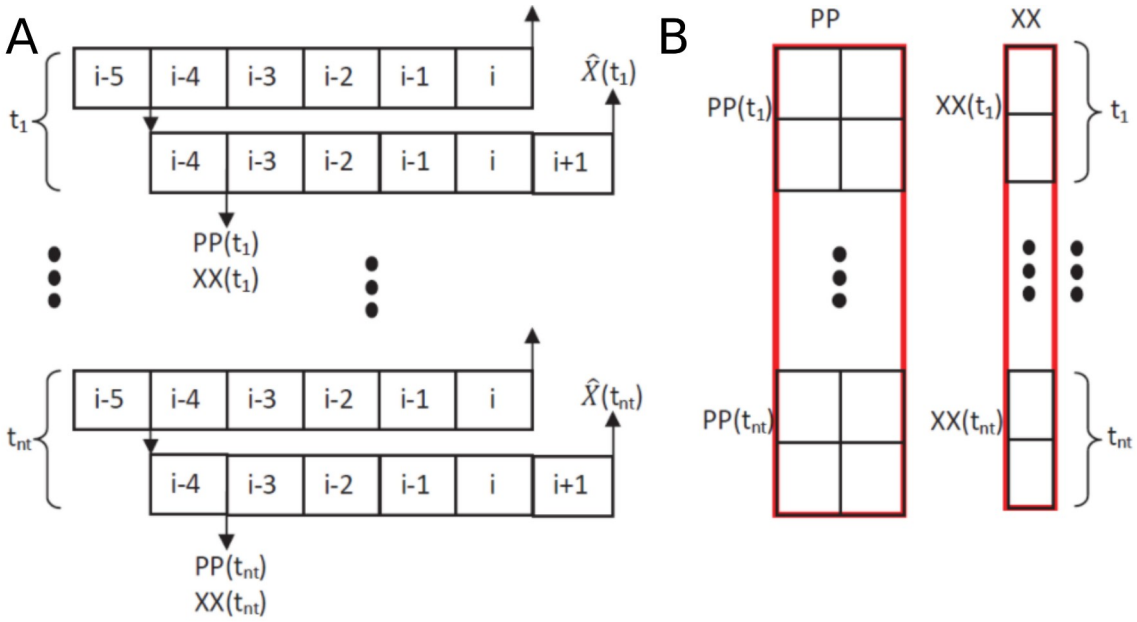

Figure 2. Summary of the moving time window. An example moving time window with *L.dyn=5* is shown. **A** shows the logic of the moving time window. The estimates of the state variables and of the error covariance matrix are stores in *XX* and *PP*, respectively, after the first recursion, to be used as the initial conditions in the time window of the next generation. This process is done separately for each trait (from t_1 to t_{nt}). **B** shows a scheme of the matrices *PP* and *XX*.