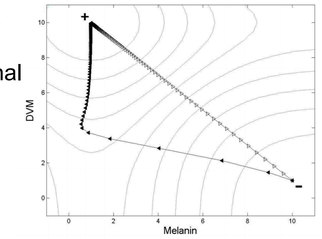
Integrated Information Sciences : Assessing the Evolutionary Consequences of Genetic Nonlinear Developmental Interactions

Rainier Harvey, Jesse Sliter, Elizabeth Brooks, Drs. Filip Jagodzinski and Ali Scoville (CWU Biology)

Abstract

Quantitative genetics is concerned with developing computational models to simulate and predict the evolution of traits in response to selection. Nonlinear interactions between developmental factors underlying the production of traits can drastically affect how two (or more) traits co-vary.



A fitness surface displays levels of high (+) and low (-) reproductive success as a function of two or more traits.

In this work we are developing a Graphical User Interface (GUI) and the accompanying back-end infrastructure to permit biologists to interface with **ModEvo**, our own Java code base for hypothesis testing about the effects of non-linear interactions on evolutionary dynamics.

We use *Google Go* as the backend server and Angular as the model-view controller. Users specify input parameters for the quantitative genetics models and invoke the back-end software with a single button click. We process the data generated by the models and present plots that permit easy analysis about the co-variance of traits. Our front-end, back-end solution is the first of its kind.

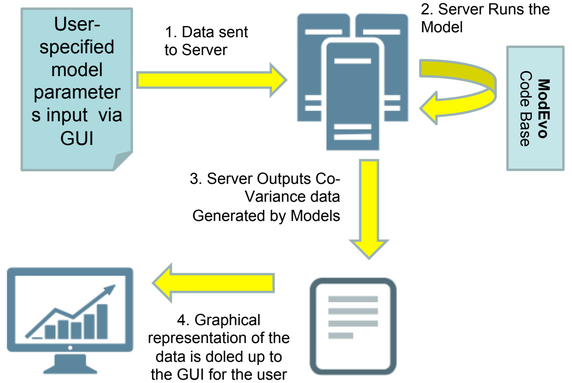
Motivation

The evolutionary response of a population depends upon patterns of inheritance as well as the strength and direction of selection. The additive genetic variance-covariance matrix (G-matrix), which summarizes the distribution of genetic variance among multiple traits, is a central concept in evolutionary theory [4]. However the predictive power of G-matrix models depends upon the stability of the matrix, which is poorly understood [1,2,3].

In our Java **ModEvo** code base, we have implemented Rice's mathematical framework [5], allowing a user to test hypotheses regarding the evolutionary consequences of nonlinear developmental interactions.

Implementation

We use Google Go (*Golang*) as the backend web server. It is comprised of an API and template server, which generates traditional web pages for a browser to display. The API allows a user to access the server in a stateless way. A user can create accounts, and the front end implements context independent states.



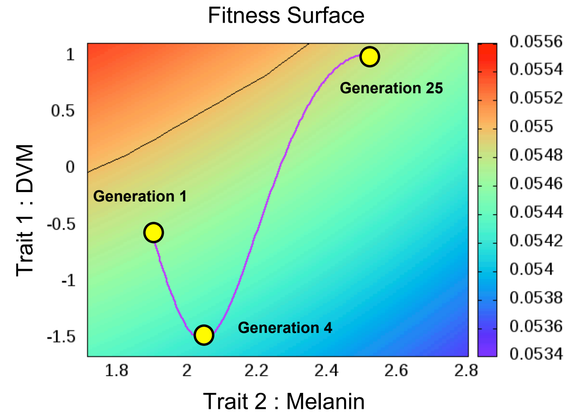
Graphical Plots Output For the User To Inspect

Trait Values Output by Model		
Generation	**Trait 1, DVM**	**Trait 2, Melanin**
1	-0.6	1.92
4	-1.4	2.09
...	...	...
25	0.9	2.54

The front end showcases a "Result" page for each invocation of a model, which outputs trait 1 and trait 2 data for each generation simulated.

For Daphnia, a small water "flea", the fitness of an individual is a function of two traits *x* (Melanin) and *y* (Daphnia Vertical Migration (DVM))

$$fitness(x,y) = \frac{1}{|\sqrt{100 \times 2 \times \Pi}|} \times e^{\frac{-(x-0.4)^2}{200}} + \frac{1}{|\sqrt{500 \times 2 \times \Pi}|} \times \frac{-(y-12)^2}{500}$$



The server plots the fitness surface, over which is overlaid the fitness of a species through the generations that are simulated by the model.

Sample Go Code

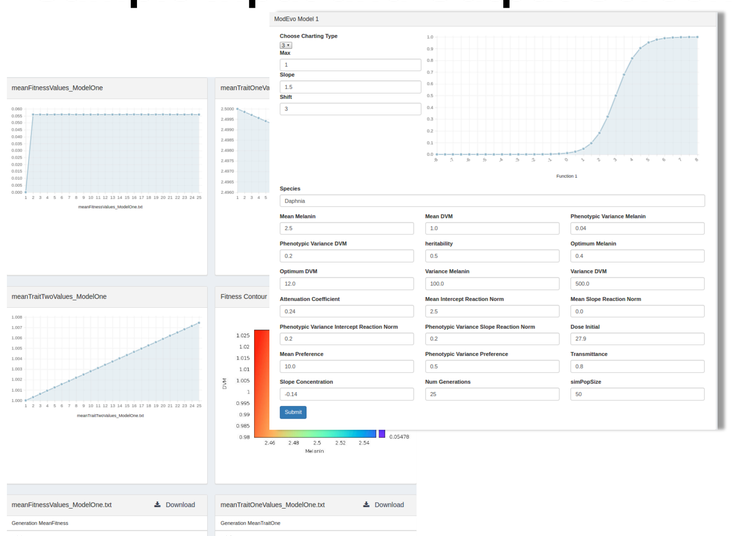
```
func ClearFiles() {
    for {
        select {
        case <-ticker.C:
            AddSoftware()
            var stored []Stored
            var a []interface{}
            err := DBRead(QueryStored, a, &stored)
            if err != nil {
                fmt.Println(err.Error())
                DBLogErrorLocal(err.Error(), "DBQuery")
                return
            }
            t := time.Now().Unix()
            for _, s := range stored {
                if t-s.Time > 1209600 {
                    err = DeleteFolder(path.Join(UserDir, s.UserName, s.Folder))
                    if err != nil {
                        fmt.Println(err.Error())
                        DBLogErrorLocal(err.Error(), s.Folder)
                        return
                    }
                    err = DeleteFolder(path.Join("public/img/gnu", s.Folder))
                    if err != nil {
                        fmt.Println(err.Error())
                        DBLogErrorLocal(err.Error(), s.Folder)
                        return
                    }
                    var args []interface{}
                    args = append(args, s.Folder)
                    err = DBWrite(DeleteStored, args)
                    if err != nil {
                        fmt.Println(err.Error())
                        DBLogErrorLocal(err.Error(), s.Folder)
                        return
                    }
                }
            }
            var users []User
            err = DBRead(QueryUsers, a, &users)
            if err != nil {
                fmt.Println(err.Error())
                DBLogErrorLocal(err.Error(), "User Query")
                return
            }
            for _, u := range users {
                if t-u.Time > 1209600 {
                    err = DeleteFolder(path.Join(UserDir, u.Name))
                    if err != nil {
                        fmt.Println(err.Error())
                        DBLogErrorLocal(err.Error(), u.Folder)
                        return
                    }
                    var args []interface{}
                    args = append(args, u.Name)
                    err = DBWrite(DeleteUser, args)
                }
            }
        }
    }
}
```

Example of automated file management. Cleans out old files which have not been interacted with

Database encapsulation. Interactions with external resources can be handled through simple interfaces

For loop example. Go uses unique iteration techniques, with all loops being for loops.

Sample Input and Output Screens



Future Work

Concurrent execution of multiple models in the case of thousands of requests and custom graph functions are two of the next features to be added. Also to be developed is a dynamic plotting feature, in which back-end models will be run as a user interacts with a front-end plots of trait variables.

Bibliography

1. M. Bjorklund, A. Husby, and L. Gustafsson. Rapid and unpredictable changes of the G-matrix in a natural bird population over 25 years. Journal of Evolutionary Biology, 26(1):1–13, Jan. 2013.
2. F. Eroukhmanoff. Just How Much is the G-matrix Actually Constraining Adaptation? Evolutionary Biology, 36(3):323–326, May 2009.
3. T. F. Hansen. The Evolution of Genetic Architecture. Annual Review of Ecology, Evolution, and Systematics, 37:123–157, Jan. 2006.
4. R. Lande and S. J. Arnold. The Measurement of Selection on Correlated Characters. Evolution, 37(6):1210–1226, Nov. 1983.
5. S. Rice. A general population genetic theory for the evolution of developmental interactions. Proceedings of the National Academy of Sciences, 99:15518–15523, 2002.