# FunGraph-vignette

```
library(FunGraph)
```

take a look at genotypic data used in example

```
View(geno)
```

| | Linkage | Genetic_Distances(cM) | 2 | 4 | 6 | 9 | 11 | 12 | 15 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|
| nn_np_9292 | lg1 | 0.00 | 0 | 1 | 9 | 0 | 0 | 1 | 1 | 0 |
| lm_ll_11315 | lg1 | 1.71 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| lm_ll_4392 | lg1 | 2.34 | 0 | 1 | 1 | 1 | 9 | 9 | 1 | 1 |
| nn_np_8171 | lg1 | 2.55 | 0 | 1 | 9 | 0 | 0 | 1 | 9 | 0 |
| lm_ll_12452 | lg1 | 2.89 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| lm_ll_7926 | lg1 | 3.63 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 9 |

take a look at phenotypic data used in example

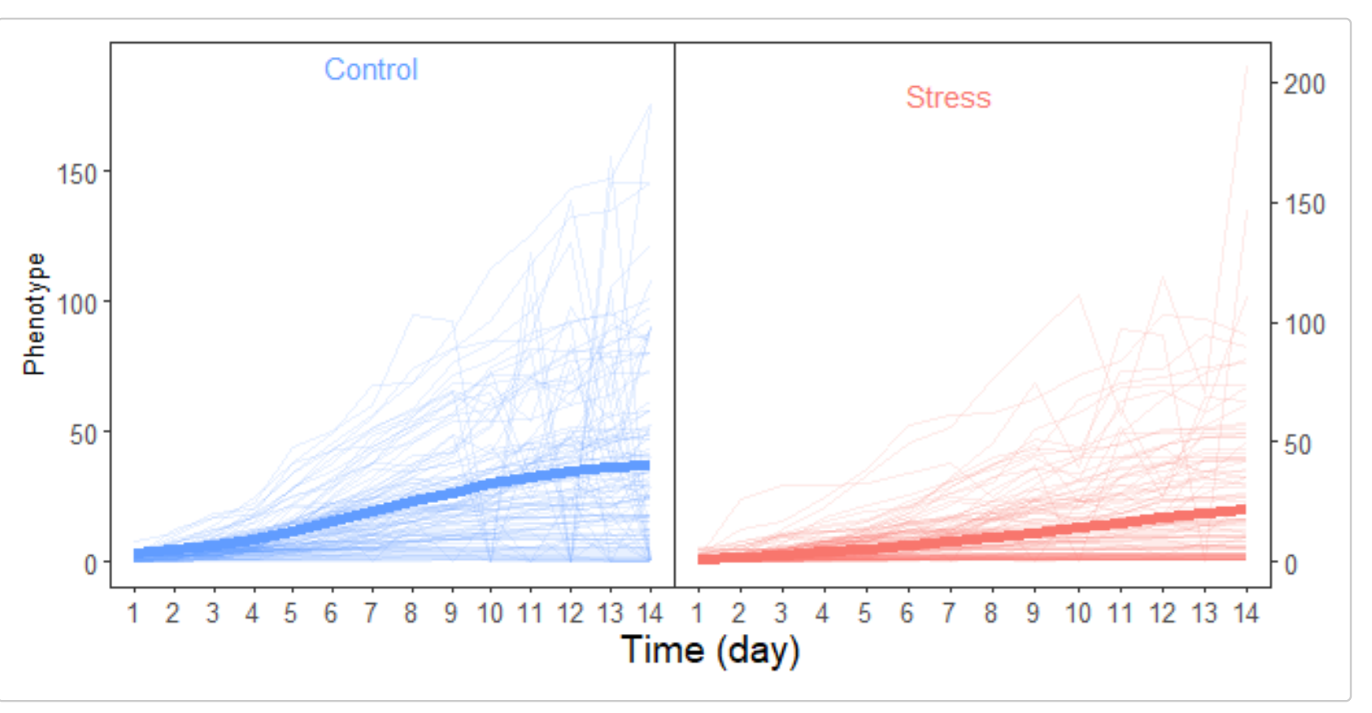
```
View(pheno)
```

| | N1.x | N2.x | N3.x | N4.x | N5.x | N6.x | N7.x | N8.x | N9.x | N10.x |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2.000000 | 3.000000 | 8.000000 | 12.000000 | 16.000000 | 18.000000 | 18.00000 | 21.00000 | 21.00000 | 23.00000 |
| 4 | 3.666667 | 4.333333 | 5.000000 | 5.666667 | 6.666667 | 8.333333 | 11.00000 | 13.66667 | 17.33333 | 24.33333 |
| 6 | 2.000000 | 2.666667 | 2.666667 | 2.666667 | 3.333333 | 6.666667 | 10.66667 | 17.00000 | 20.33333 | 24.33333 |
| 9 | 0.000000 | 1.000000 | 2.500000 | 5.000000 | 7.000000 | 13.000000 | 15.50000 | 23.50000 | 25.50000 | 26.00000 |
| 11 | 1.000000 | 1.000000 | 6.000000 | 19.000000 | 38.000000 | 43.000000 | 56.00000 | 74.00000 | 82.00000 | 85.00000 |
| 12 | 5.000000 | 6.000000 | 5.000000 | 6.000000 | 13.000000 | 18.000000 | 31.00000 | 41.00000 | 42.00000 | 52.00000 |

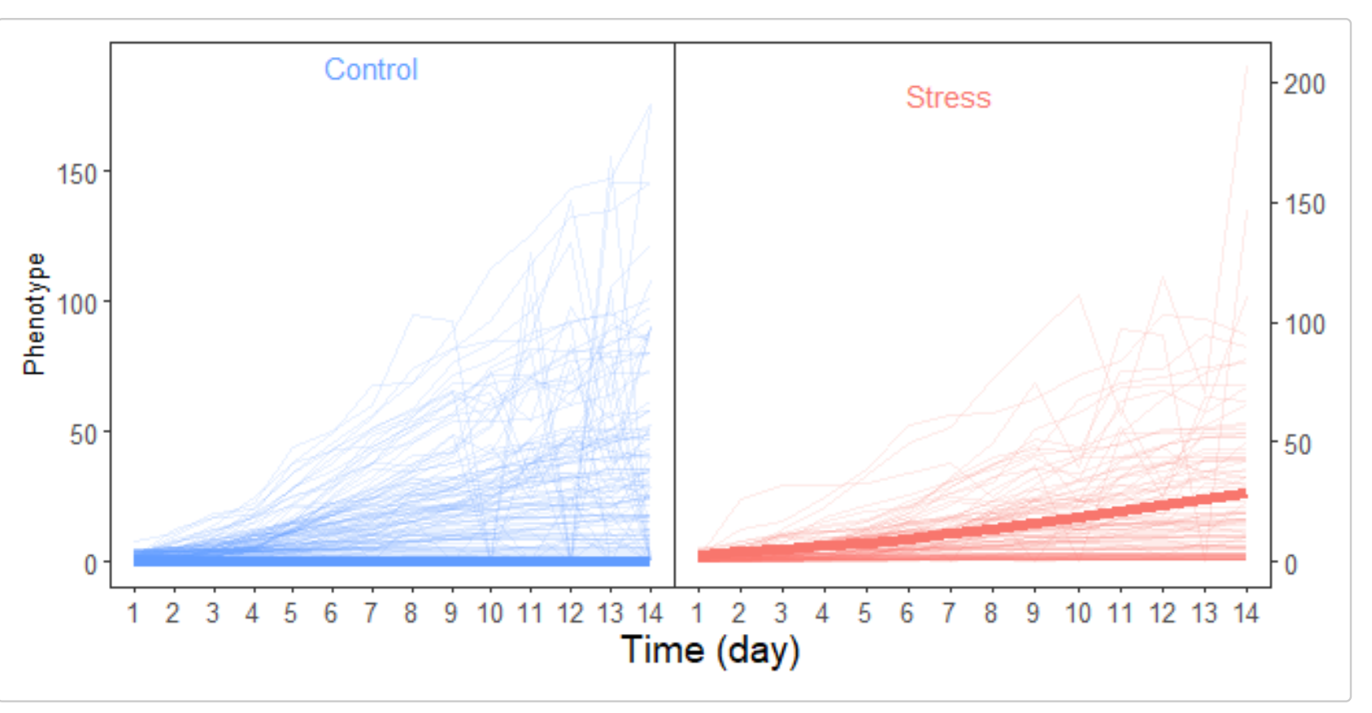
# 1.biFunMap exampleples

first plot mean curve to check initial parameters is OK

```
get_mean_curve_plot(pheno_df = pheno, times = 1:14)
```



```
#Notice wrong given inital parameters results in incorrect mean curve(blue line)
get_mean_curve_plot(pheno_df = pheno, times = 1:14, init_sd_par = c(2,0.1,2,0.1))
```



calculate LR value for a SNP

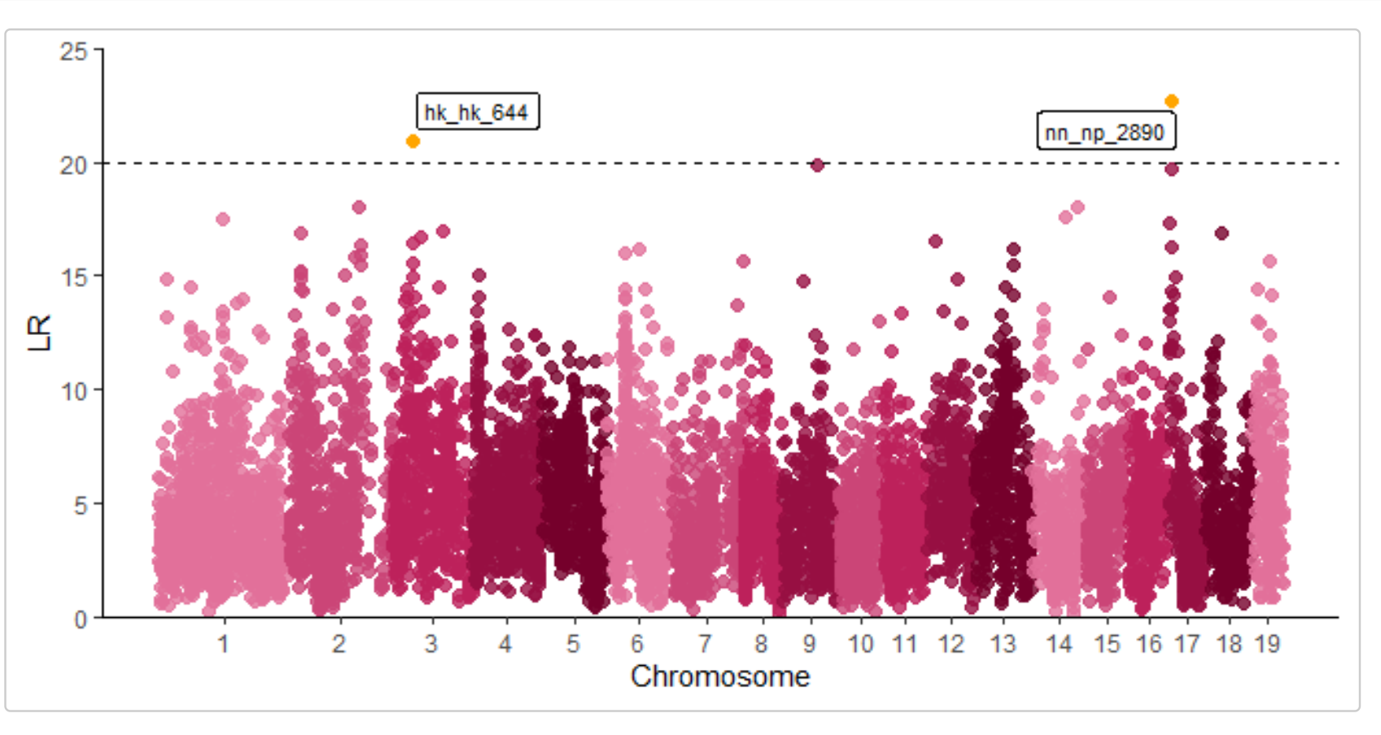
```
#geno[,-1:-2] remove redundant information for calculation
get_LR_effect(marker = 1,
              geno_df = geno[,-1:-2],
              pheno_df = pheno,
              times = 1:14)
#> $LR
#> [1] 2.967023
#>
#> $genetic_effect
#>  [1] 0.201284440 0.264772200 0.346411862 0.456347711 0.610781174 0.830762203
#>  [7] 1.133019240 1.516061350 1.952992338 2.399895761 2.813927481 3.167246698
#> [13] 3.450041283 3.665821955 0.250562724 0.176137078 0.123056444 0.084406580
#> [19] 0.055011630 0.031269674 0.011047813 0.006439401 0.020841086 0.031011650
#> [25] 0.035465411 0.032863884 0.022421353 0.004131180
#>
#> $NH0_pars
#>  [1] 38.7016809 17.3447288  0.4081965 -2.6843421 10.4652100 31.1062723
#>  [7]  8.9873842  0.2298382  3.0402288  0.1321409  0.8847419 13.1424409
#> [13]  1.0636326  6.6143222
#>
#> $NH1_pars
#>  [1] 42.2212696 18.9548114  0.4060741  1.0666182 11.6341456 35.8311259
#>  [7] 10.4193945  0.2290326  3.7232069  0.2276818 33.7701211 17.6666255
#> [13]  0.4254191 -7.0663826  8.8127371 36.7754624 10.0386477  0.2210702
#> [19]  3.2022028  0.1827884  0.8758515 13.0793325  1.0659165  6.5823555
```

calculate all biFunMap result

```
get_biFunMap_result(geno_df = geno[,-1:-2],
                    pheno_df = pheno,
                    times = 1:14)
```

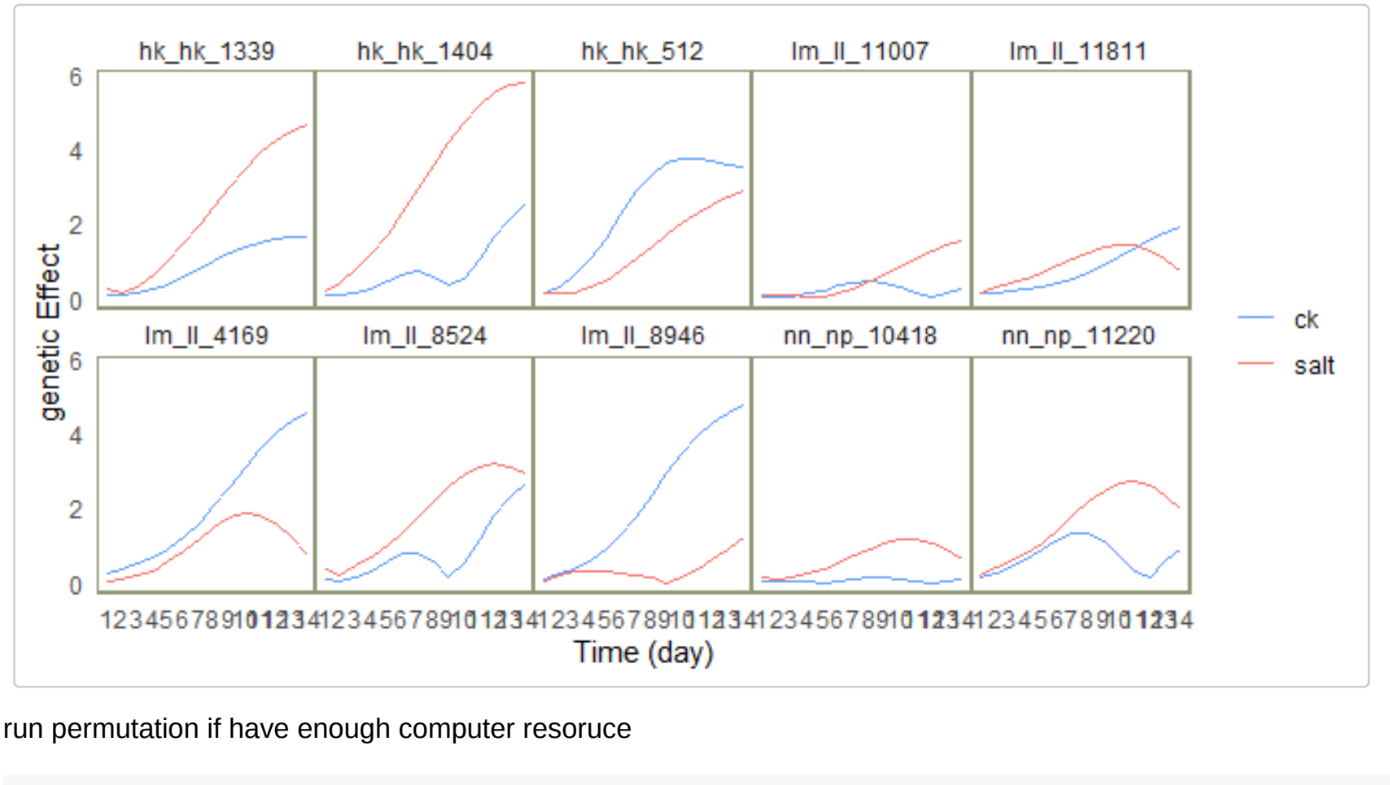
manhattan plot using calculate LR values

```
get_manh_plot(geno_df = geno, LR = LR)
```



choose 10 SNPs to show their genetic effect

```
get_genetic_effect_plot(genetic_effect = genetic_effect, number = 10)
#> Using rownames(df_ck) as id variables
#> Using rownames(df_salt) as id variables
```



run permutation if have enough computer resoruce

```
#example use permutation 5 times, each permutation use first 3 SNPs
get_permutation(n = 5,
                geno_df = geno[1:3,-1:-2],
                pheno_df = pheno,
                times = 1:14)
```

see more about how FunMap work on funmap2.

# 2.biFunClu exampleples

prepare initial parameters for biFunClu (may need to change init_sd_par manually) in this example only use first 100 rows to cluster, larger data set took more time

```
set.seed(2021) #use same initial value

input <- get_init_par(data = genetic_effect[1:100,],
                      k = 5,
                      legendre_order = 3,
                      times = 1:14)
#> Loading required package: polynom

c1 <- get_cluster(input = input, itermax = 10)
#> Start biFunClu Calculation
#> Cluster_number=5 Legendre_order=3
#> iter= 1 LL= -1079.353
#> iter= 2 LL= -1119.94
#> iter= 3 LL= -1128.265
#> iter= 4 LL= -1133.143
#> iter= 5 LL= -1138.135
#> iter= 6 LL= -1141.217
#> iter= 7 LL= -1143.213
#> iter= 8 LL= -1144.979
#> iter= 9 LL= -1146.217
#> iter= 10 LL= -1146.848
#> Finish biFunClu Calculationcluster result data written

#plot the result
get_cluster_base_plot(c1$clustered_data[[1]])
#> Warning: Use of `long_df$time` is discouraged. Use `time` instead.
#> Warning: Use of `long_df$effect` is discouraged. Use `effect` instead.
#> Warning: Use of `long_df$marker` is discouraged. Use `marker` instead.
#> Warning: Use of `long_df$cluster` is discouraged. Use `cluster` instead.
```

see more about how FunClu work on funclu.

# 3.LASSO-based variable selection examples

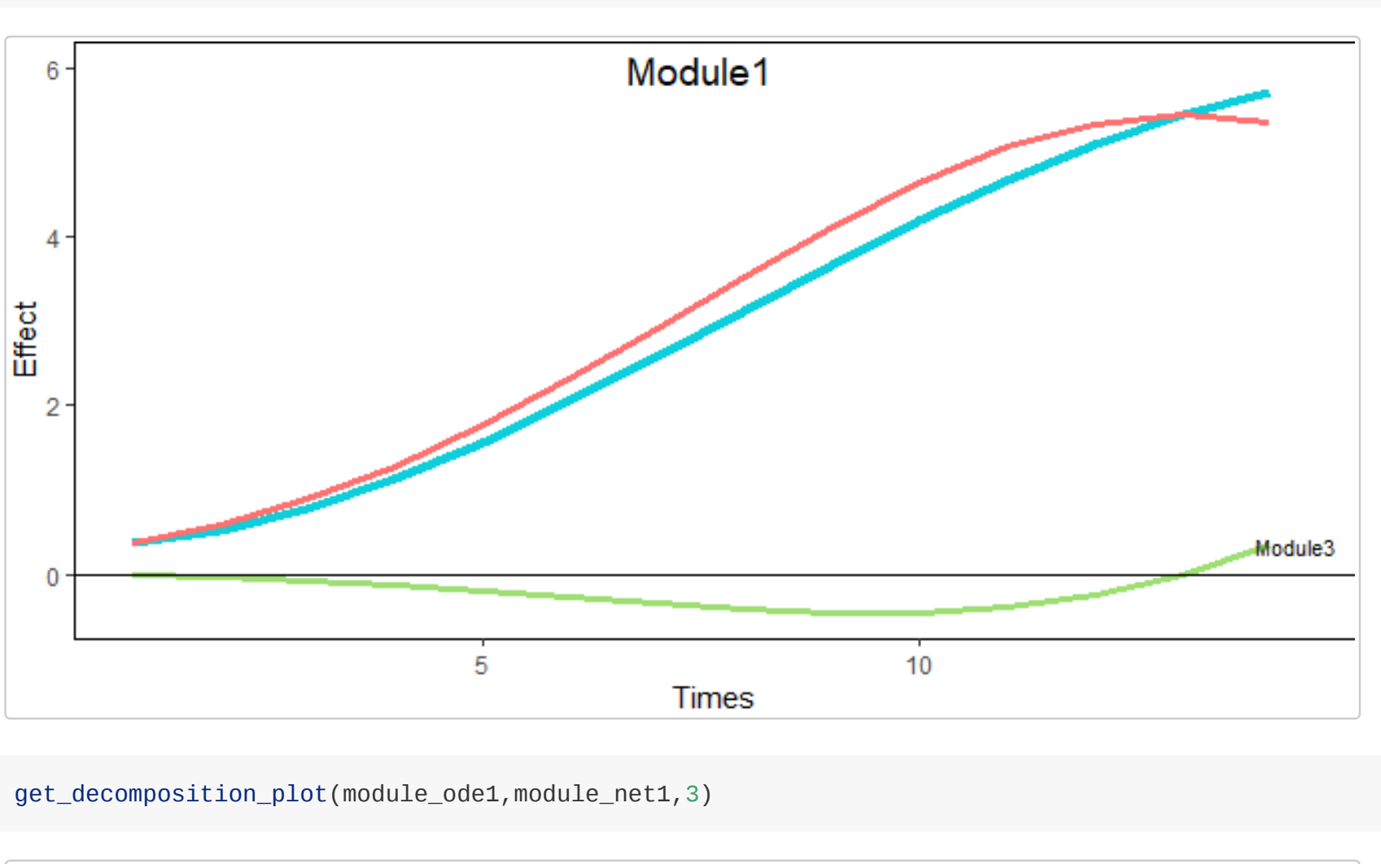
```
module_data <- get_module_data(data_par = c1$curve_par, times = 1:14)

#select ck data for variable selection
get_interaction(data = module_data[[1]], col = 1)
#> [[1]]
#> [1] "Module1"
#>
#> [[2]]
#> [1] "Module3"
#>
#> [[3]]
#> [1] 3.785921

#alternatively, use a cluster data(ck) for variable selection
get_interaction(data = c1$clustered_data[[1]][,-ncol(c1$clustered_data[[1]])],
                col = 1,
                reduction = TRUE)
#> [[1]]
#> [1] "nn_np_9292"
#>
#> [[2]]
#> [1] "lm_ll_2416"
#>
#> [[3]]
#> [1] 2.981482
```

# 4.ODE solving examples

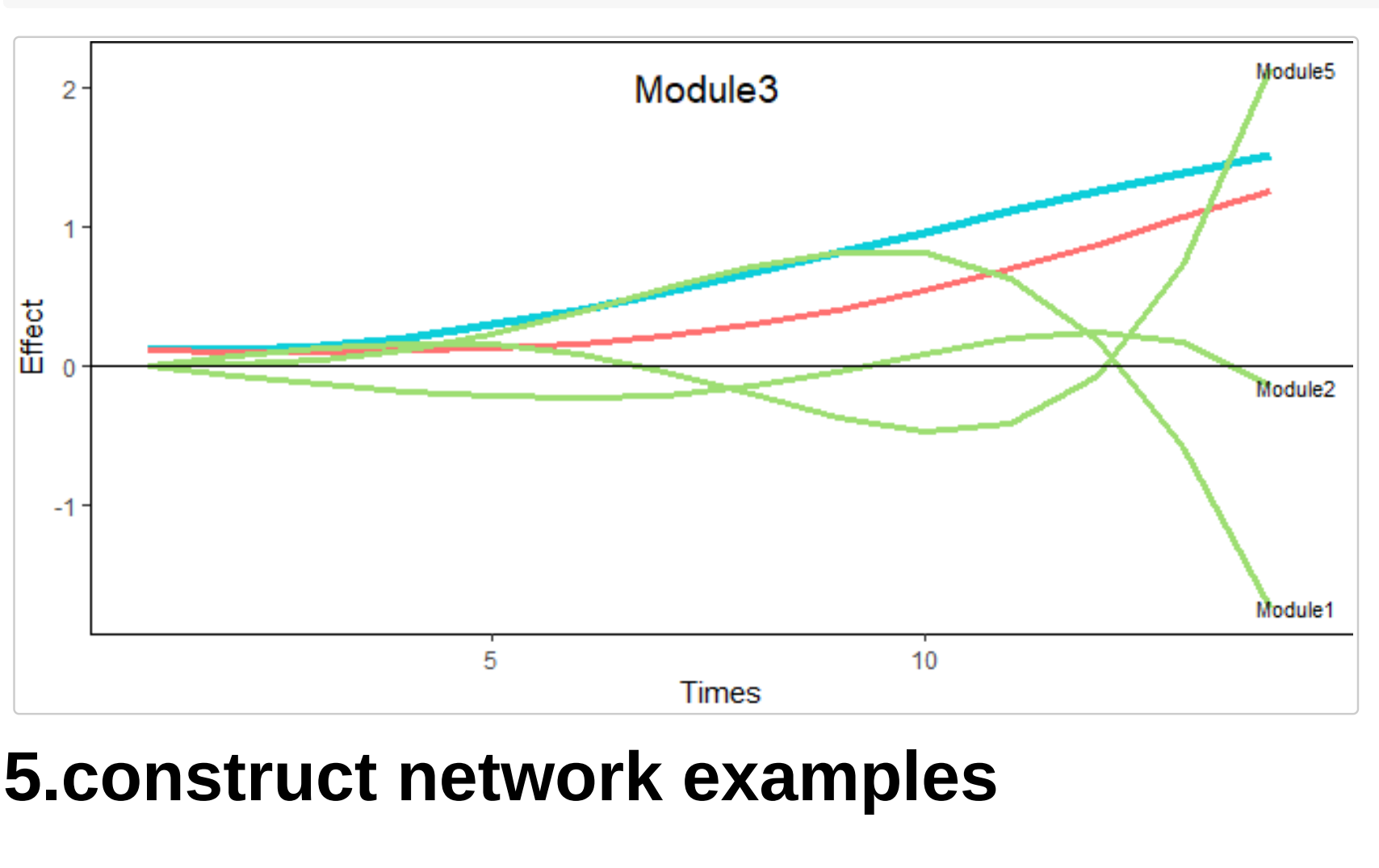
```
module_data <- get_module_data(data_par = c1$curve_par, times = 1:14)
#for module, just input module data
module_ode1 <- get_ode_par(data = module_data[[1]],
                           times = 1:14,
                           order = 3,
                           reduction = FALSE,
                           parallel = FALSE)
#> Start variable selection
#> Finish variable selection
#> Start ODE solving
#> initial  value 231.202046
#> iter  10 value 0.004505
#> iter  20 value 0.004431
#> iter  30 value 0.004232
#> iter  40 value 0.004145
#> iter  50 value 0.004096
#> iter  60 value 0.004079
#> iter  70 value 0.004072
#> iter  80 value 0.004063
#> final  value 0.004062
#> converged
#> initial  value 3.134045
#> iter  10 value 0.001245
#> iter  20 value 0.000372
#> iter  30 value 0.000372
#> iter  40 value 0.000372
#> iter  50 value 0.000372
#> final  value 0.000372
#> converged
#> initial  value 0.919899
#> iter  10 value 0.001476
#> iter  20 value 0.000684
#> iter  30 value 0.000683
#> iter  40 value 0.000683
#> iter  40 value 0.000683
#> iter  40 value 0.000683
#> final  value 0.000683
#> converged
#> initial  value 0.391978
#> iter  10 value 0.000187
#> iter  20 value 0.000119
#> iter  30 value 0.000118
#> iter  40 value 0.000117
#> iter  50 value 0.000117
#> iter  60 value 0.000117
#> iter  70 value 0.000117
#> final  value 0.000117
#> converged
#> initial  value 3.792728
#> iter  10 value 0.005885
#> iter  20 value 0.004033
#> iter  30 value 0.004023
#> iter  40 value 0.004022
#> iter  40 value 0.004022
#> final  value 0.004022
#> converged
#> Finish ODE solving
#the result should be further processed
module_net1 <- get_all_net(module_ode1)
```

take a look at whats going on with 1st and 3rd module(i=1, i=3) of CK data

```
get_decomposition_plot(module_ode1,module_net1,1)
```



```
get_decomposition_plot(module_ode1,module_net1,3)
```



# 5.construct network examples

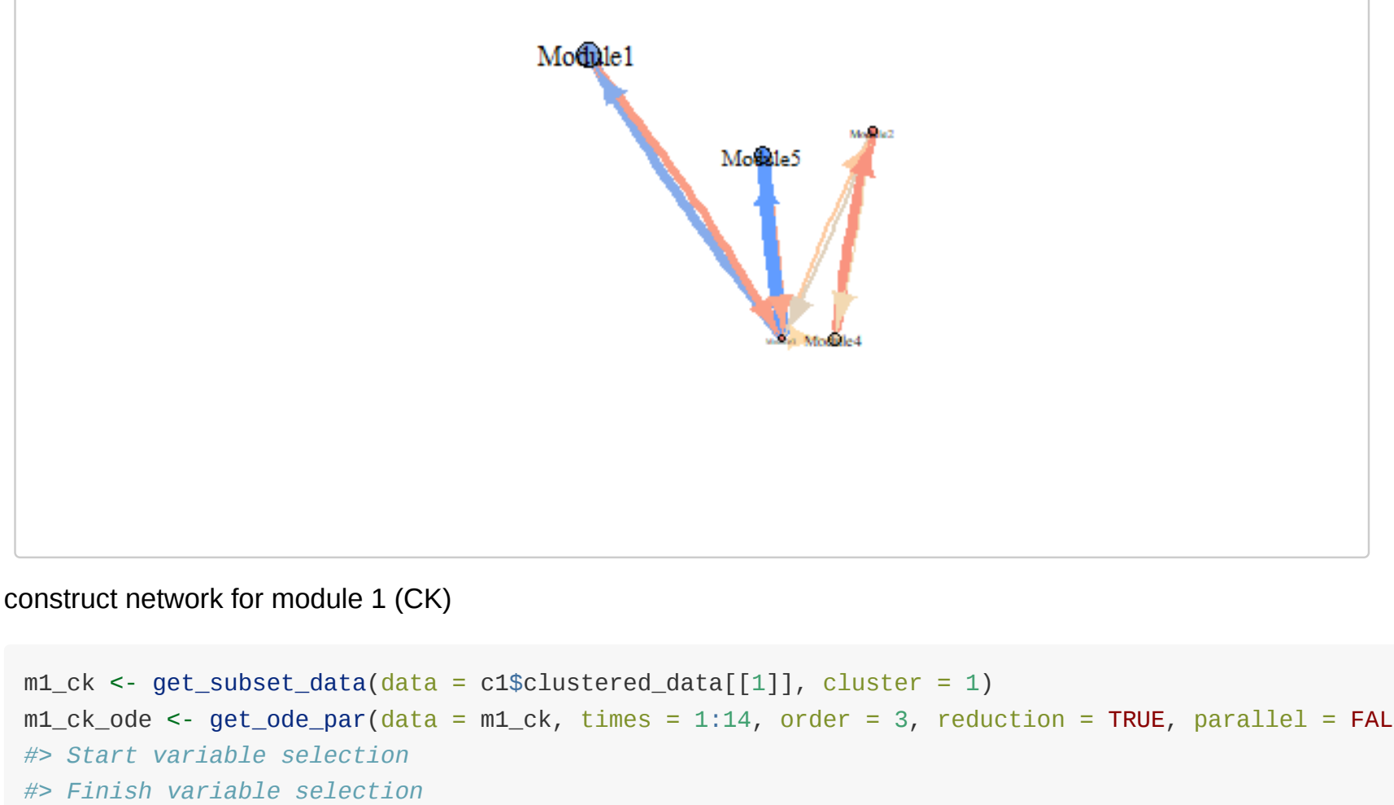
write files for Cytoscape

```
get_net_output(module_ode1,module_net1,write_data = FALSE)
#> $links
#>      from      to  dep_effect effect_type
#> 1 Module3 Module1 0.193903577           -
#> 2 Module3 Module2 0.047638701           +
#> 3 Module4 Module2 0.181438366           +
#> 4 Module1 Module3 0.157234797           +
#> 5 Module2 Module3 0.046536628           -
#> 6 Module5 Module3 0.135069370           +
#> 7 Module2 Module4 0.017824990           -
#> 8 Module3 Module4 0.005045651           -
#> 9 Module3 Module5 0.256124592           -
#>
#> $nodes
#>        id    name received_effect   influence
#> 1 Module1 Module1       3.1164808 -0.19390358
#> 2 Module2 Module2       0.9334943  0.22907707
#> 3 Module3 Module3       0.4326814  0.24576754
#> 4 Module4 Module4       1.4576002 -0.02287064
#> 5 Module5 Module5       2.1528598 -0.25612459
```

see more on how to use Cytoscape.

or use igraph package for visulation

```
#acquire max_effect to control color
max_effect <- cbind(get_max_effect(module_net1),get_max_effect(module_net1))
network_plot(k = module_net1,
             title = 'CK',
             max_effect = max_effect,
             save_plot = FALSE)
```



construct network for module 1 (CK)

```
m1_ck <- get_subset_data(data = c1$clustered_data[[1]], cluster = 1)
m1_ck_ode <- get_ode_par(data = m1_ck, times = 1:14, order = 3, reduction = TRUE, parallel = FALS
#> Start variable selection
#> Finish variable selection
#> Start ODE solving
#> initial  value 38.570698
#> iter  10 value 0.003212
#> iter  20 value 0.001157
#> iter  30 value 0.001157
#> iter  40 value 0.001156
#> iter  50 value 0.001156
#> final  value 0.001156
#> converged
#> initial  value 643.124332
#> iter  10 value 0.009925
#> iter  20 value 0.009661
#> iter  30 value 0.009649
#> iter  40 value 0.009646
#> iter  50 value 0.009646
#> final  value 0.009646
#> converged
#> initial  value 25.939055
#> iter  10 value 0.001087
#> iter  20 value 0.000994
#> iter  20 value 0.000994
#> final  value 0.000994
#> converged
#> initial  value 105.024243
#> iter  10 value 0.002151
#> iter  20 value 0.000863
#> iter  30 value 0.000862
#> iter  40 value 0.000862
#> iter  50 value 0.000862
#> final  value 0.000862
#> converged
#> initial  value 397.599609
#> iter  10 value 0.006147
#> iter  20 value 0.006098
#> iter  30 value 0.006081
#> iter  40 value 0.006079
#> final  value 0.006079
#> converged
#> initial  value 15.494284
#> iter  10 value 0.000233
#> iter  20 value 0.000137
#> final  value 0.000129
#> converged
#> initial  value 5.519125
#> iter  10 value 0.002997
#> iter  20 value 0.002780
#> iter  30 value 0.002405
#> iter  40 value 0.002401
#> iter  50 value 0.002400
#> iter  60 value 0.002399
#> iter  70 value 0.002399
#> iter  80 value 0.002399
#> iter  90 value 0.002399
#> final  value 0.002399
#> converged
#> Finish ODE solving
m1_ck_net <- get_all_net(m1_ck_ode)

max_effect <- cbind(get_max_effect(m1_ck_net),get_max_effect(m1_ck_net))

network_plot(k = m1_ck_net, title = 'Module1_CK', max_effect = max_effect, save_plot = FALSE)
```