Reconciling empirical interactions with species coexistence

*Daniel S. Maynard, J. Timothy Wootton, Carlos A. Serv'an, Stefano Allesina

First, source the R files containing the fitting functions:

Then, set the seed and the number of species

Generate a skew-symmetric payoff matrix

```
##              [,1]        [,2]       [,3]        [,4]        [,5]
##  [1,]  0.00000000  0.34488716  0.4378136 -0.15750504  0.19034789
##  [2,] -0.34488716  0.00000000  0.5018434 -0.50283718 -0.12914603
##  [3,] -0.43781356 -0.50184343  0.0000000 -0.18576565 -0.39141607
##  [4,]  0.15750504  0.50283718  0.1857656  0.00000000  0.30632696
##  [5,] -0.19034789  0.12914603  0.3914161 -0.30632696  0.00000000
##  [6,] -0.12889145 -0.50762312  0.4607805  0.27460501 -0.03037759
##  [7,]  0.24264237 -0.06256427  0.3693522  0.42566584 -0.81657654
##  [8,]  0.19673477 -0.27024912 -0.4017674  0.42891433  0.44681896
##  [9,] -0.24555313 -0.06560125  0.5479041  0.06189491 -0.10074753
## [10,] -0.05883441  0.12472507 -0.2360372 -0.01665365  0.41991932
##              [,6]        [,7]        [,8]        [,9]       [,10]
##  [1,]  0.12889145 -0.24264237 -0.19673477  0.24555313  0.05883441
##  [2,]  0.50762312  0.06256427  0.27024912  0.06560125 -0.12472507
##  [3,] -0.46078052 -0.36935219  0.40176745 -0.54790410  0.23603718
##  [4,] -0.27460501 -0.42566584 -0.42891433 -0.06189491  0.01665365
##  [5,]  0.03037759  0.81657654 -0.44681896  0.10074753 -0.41991932
##  [6,]  0.00000000  0.30164855  0.07842579 -0.38015914  0.32949273
##  [7,] -0.30164855  0.00000000 -0.07474495  0.71419100  0.02986950
##  [8,] -0.07842579  0.07474495  0.00000000  0.01697733  0.33264934
##  [9,]  0.38015914 -0.71419100 -0.01697733  0.00000000  0.57368769
## [10,] -0.32949273 -0.02986950 -0.33264934 -0.57368769  0.00000000
```

And generate a relative abundance vector:

```
##  [1] 0.12194154 0.34495456 0.03767715 0.04408449 0.14086959 0.21119463
##  [7] 0.00650181 0.04623493 0.02046318 0.02607813
```

First, let's implement quadratic programming to reconcile

$$P$$

with

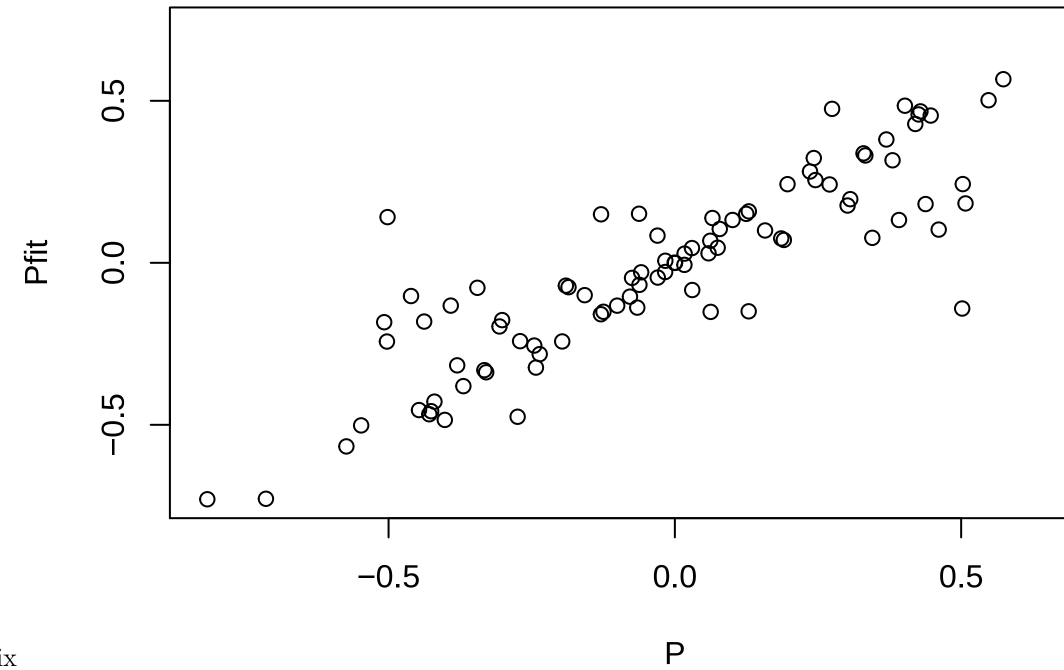
$$x_equil$$

, assuming each entry is weighted equally:

We can view the best-fitting payoff matrix

```
##              [,1]        [,2]        [,3]         [,4]        [,5]
##  [1,]  0.00000000  0.07708977  0.18128418 -0.099865846  0.07053001
##  [2,] -0.07708977  0.00000000 -0.14109695 -0.242969894 -0.15872756
##  [3,] -0.18128418  0.14109695  0.00000000 -0.075215561 -0.13208858
##  [4,]  0.09986585  0.24296989  0.07521556  0.000000000  0.19642412
##  [5,] -0.07053001  0.15872756  0.13208858 -0.196424122  0.00000000
##  [6,]  0.14972798 -0.18325726  0.10257572  0.475159188  0.08397319
```

```
##  [7,]  0.32333655  0.15142893  0.38060694  0.457911796 -0.72974540
##  [8,]  0.24268590 -0.24179732 -0.48483439  0.467380927  0.45447299
##  [9,] -0.25535962 -0.13828180  0.50182556  0.068022167 -0.13218301
## [10,] -0.02927272  0.15108020 -0.28176405  0.006360121  0.42844568
##              [,6]        [,7]        [,8]        [,9]        [,10]
##  [1,] -0.14972798 -0.32333655 -0.24268590  0.25535962  0.029272722
##  [2,]  0.18325726 -0.15142893  0.24179732  0.13828180 -0.151080202
##  [3,] -0.10257572 -0.38060694  0.48483439 -0.50182556  0.281764051
##  [4,] -0.47515919 -0.45791180 -0.46738093 -0.06802217 -0.006360121
##  [5,] -0.08397319  0.72974540 -0.45447299  0.13218301 -0.428445677
##  [6,]  0.00000000  0.17674732  0.10448188 -0.31641944  0.337878738
##  [7,] -0.17674732  0.00000000 -0.04659929  0.72825528  0.045550366
##  [8,] -0.10448188  0.04659929  0.00000000  0.02840664  0.331267834
##  [9,]  0.31641944 -0.72825528 -0.02840664  0.00000000  0.566629705
## [10,] -0.33787874 -0.04555037 -0.33126783 -0.56662971  0.000000000
```



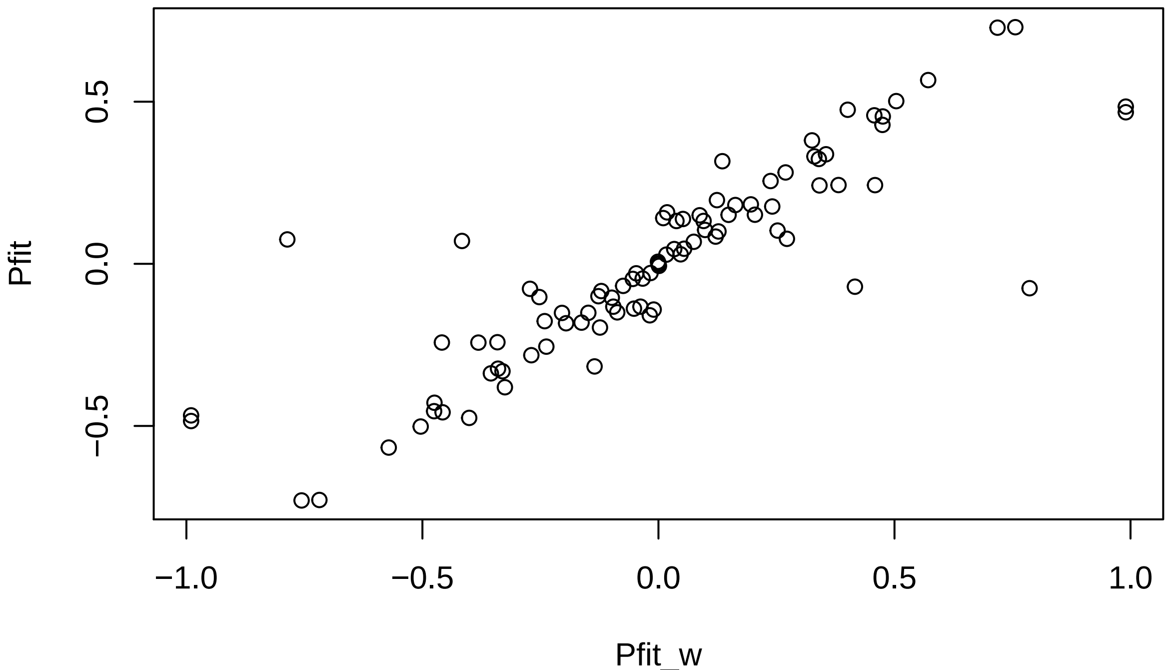
And compare to the original matrix

Last, we can double check that it is indeed an exact an exact solution by checking the growth rates at x_equil:

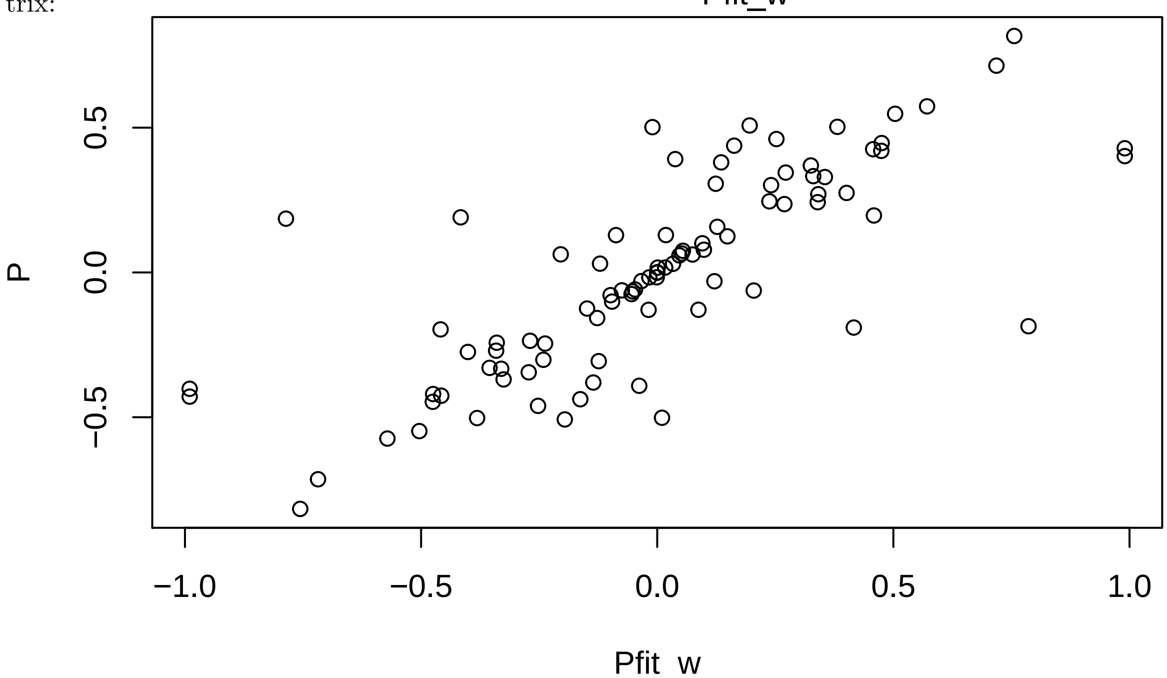
```
##       [,1]
##  [1,]    0
##  [2,]    0
##  [3,]    0
##  [4,]    0
##  [5,]    0
##  [6,]    0
##  [7,]    0
##  [8,]    0
##  [9,]    0
## [10,]    0
```

Now let's repeat, but this time generate a random matrix of weights, reflecting, for example, different sample sizes for each entry:

And we can compare this new weighted payoff matrix to the original best-fitting matrix, and the empirical ma-



trix:



And once again, it's an exact solution:

```
##       [,1]
##  [1,]    0
##  [2,]    0
##  [3,]    0
##  [4,]    0
##  [5,]    0
##  [6,]    0
##  [7,]    0
##  [8,]    0
##  [9,]    0
```

```
## [10,]    0
```

Lotka Volterra Example

generate some growth rates

r <- runif(nspp)

generate an interaction matrix

A <- -matrix(runif(nspp^2), nspp,nspp) diag(A) <- diag(A)*2

generate an abundance vector

x_obs <- runif(nspp)

get the best fitting result

result_LV <- fit_qp_LV(A=A,r=r,x_obs=x_obs,tol=1000)

Afit <- t(matrix(result_LV$X[1:nspp^2], nspp,nspp))

rfit <- result_LV$X[(nspp$^{\text{2+1):(nspp}}$2+nspp)]

plot(Afit$_{\text{A) plot(rfit}}$r)

check to make sure it's a solution

x_obs*(rfit+Afit%%*x_obs)

constrain the entries to be within 100% of the observed

result_LV_100 <- fit_qp_LV(A=A,r=r,x_obs=x_obs,tol=1)

Afit_100 <- t(matrix(result_LV_100$X[1:nspp^2], nspp,nspp))

rfit_100 <- result_LV_100$X[(nspp$^{\text{2+1):(nspp}}$2+nspp)]

compare to the original. Note that it sets a bunch of interactions to be zero

plot(Afit_100~A)

growth rate is just scales by a constant

plot(rfit_100~r)

compare the fits

plot(Afit_100~Afit)

check to make sure it's a solution

x_obs*(rfit_100+Afit_100%*%x_obs)