Inference of interaction networks using generalized Lotka Volterra equations on time-series data with package gLVInterNetworks

Lukas Hirsch

2016-12-21

Generation of in silico data

To download the package from my github account use

```
devtools::install_github("lkshrsch/gLVInterNetworks", build_vignettes = TRUE)
```

To load the package into the R environment

```
library(gLVInterNetworks)
```

To generate in silico data sets we use the following command with the following mandatory arguments:

- species: The amount of variables in the system
- number_of_interactions: The amount of interactions between different variables
- timepoints: Starting from 0, the number of measurements the output will have
- noise: The standard deviation of the stochasticity added to the data (variance of the gaussian noise)
- testData: The percentage of ending datapoints left out for validation purposes.

```
data <- gLVgenerateData(species = 2,
                        number_of_interactions = 2,
                        timepoints = 40,
                        noise = 0.01,
                        testData = 20)
```

Note that depending on the size of the system to simulate (number of species, etc) this command can output messages which can be ignored regarding the difficulty of the numerical integration of the solution.

```
data("exampleData2")
```

The variable data is a list containing the following attributes

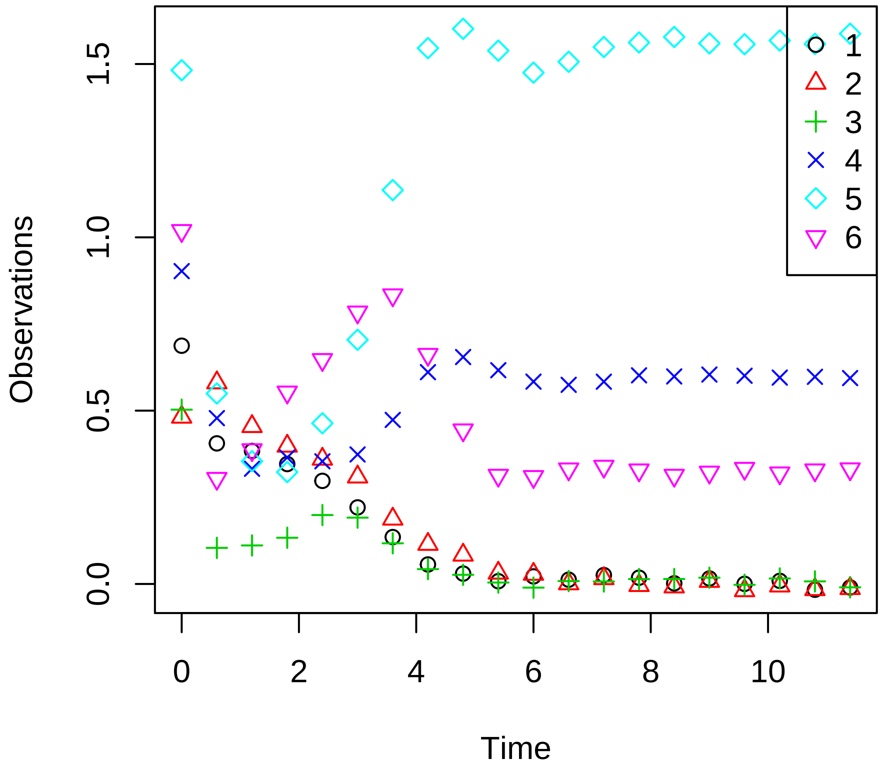
```
names(exampleData2)
```

```
## [1] "species"    "timepoints" "Parms"      "noise"      "sparsity"
## [6] "obs"        "testData"
```

From which *species*, *timepoints*, *noise*, *sparsity*, and *testData* correspond to the input arguments.

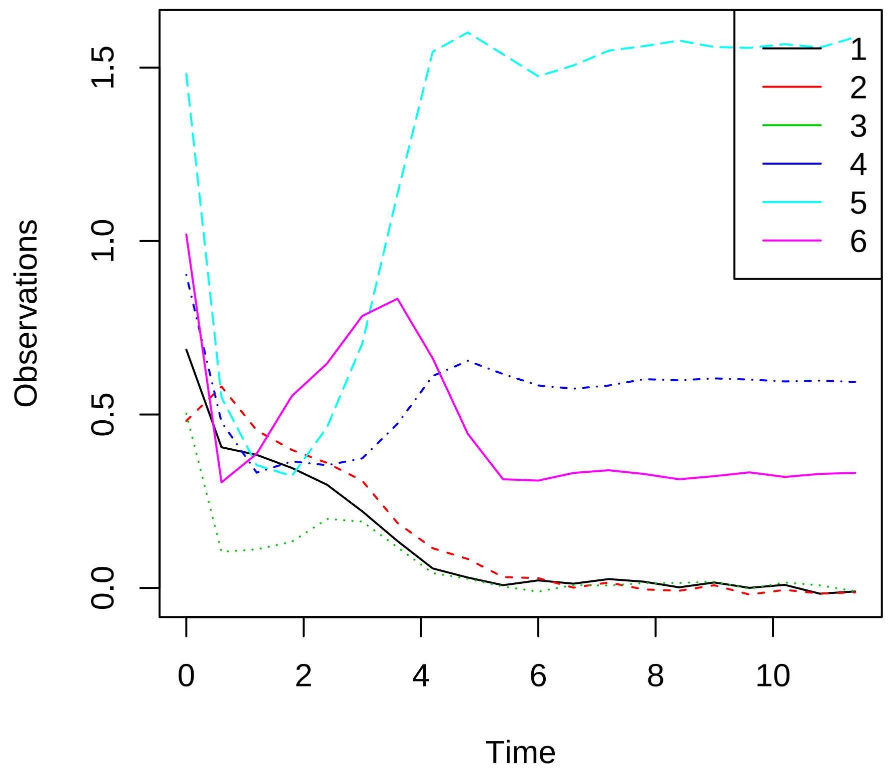
- Parms : Is the randomly generated parameter matrix under the given input constraints. It contains one row per variables / species in the system and the parameters correspond to :
- obs : Corresponds to the solution using the estimated model parameters for the time interval given. This are the values plotted when using

```
plot(exampleData2, pch = 1:exampleData2$species)
legend("topright", legend = colnames(exampleData2$obs[,-1]), pch = 1:exampleData2$species, col=1:exampl
```



Or alternatively with the legend flag set to TRUE for an easy plot generation

```
plot(exampleData2, legend = T)
```



Loading data sets

For this example I stored a table of observations with a time column with the dates when the observations were taken. The .csv file is a text file with entries sepparated by semicolons ";".

```r
setwd("home/user/example/directory/")
df <- read.csv2("realData.csv", header = TRUE)
```

DF IS AN EXAMPLE TEXT DATA IN PACKAGE LOAD(DF)

```r
head(df)
```

```r
matplot(df[,-1])
```

We can convert the data.frame to a numerical matrix using

```r
exampleData <- data.matrix(df)
```

With which we have transformed the dates into numerical values. The original measurement time rate (per day) will be the changing rate for the estimated parameters after fitting the model to the data.

```r
head(exampleData)
```

Fitting the generalized Lotka Volterra model to the data

The generalized Lotka Volterra model consists in a set of differential equations, nonlinear in the variables x_i. It describes the rate of change in time of the abundance of a variable (species, or taxonomical/functional unit)

$$\frac{dx_i}{dt} = \alpha_i x_i + \sum \beta_{ij} x_i x_j$$

Or written simpler

$$\dot{x_i} = x_i(\alpha_i + \sum \beta_{ij} x_j)$$

Written in matrix form for a two species system:

$$\begin{pmatrix} \dot{x_1} \\ \dot{x_2} \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \times \left[\begin{pmatrix} \alpha_1 & \beta_{11} & \beta_{12} \\ \alpha_2 & \beta_{21} & \beta_{22} \end{pmatrix} \cdot \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix} \right]$$

This matrix form for the parameter matrix with the growth term in the first column and interaction parameters in the right-side quadratic matrix corresponds to the output of the parameter in the R code:

HERE USE 2 SPECIES EXAMPLE DATA!

Nonlinear Regression to estimate values for the model parameters

To estimate the parameter matrix from the equations above, use

```r
nlr <- gLVnonlinearRegression(data = exampleData2)
```

The resulting list containts 12 objects, named Parms, SSR, residual_SD, SE, residuals_t.test, message, obs, Fit, df, quantitative, qualitative1, qualitative2

The algorithm achieved a fit with a total sum of squared residuals of

```r
print(nlrData2$SSR)
```

```
## [1] 0.008053986
```

The estimated parameter matrix is

```r
print(nlrData2$Parms)
```

```
##             [,1]          [,2]        [,3]        [,4]       [,5]
## [1,]  1.79223499 -1.6710904970 -0.04198621  3.53269933 -1.4976110
## [2,] -1.14900477  4.4813918003 -3.63452284  2.71661123  2.3632050
## [3,]  5.68942535 -0.0005482157 -8.21869778  1.05303570  1.7927713
## [4,]  0.08948867  5.5097018443 -5.47304844 -4.05040893 -0.7140135
## [5,]  1.81274708 -3.4934328640 -3.30201388 -0.08410942  1.6976769
## [6,]  3.57599179  2.9824639899 -1.69752061  0.48877777 -9.7911085
##             [,6]       [,7]
## [1,] -0.3371221 -2.4281236
## [2,] -0.7749937 -0.1214798
## [3,] -3.9802883 -2.2094892
## [4,] -0.2387796  2.2040078
## [5,] -2.2471820  2.2061703
## [6,]  1.7306970 -1.4043691
```

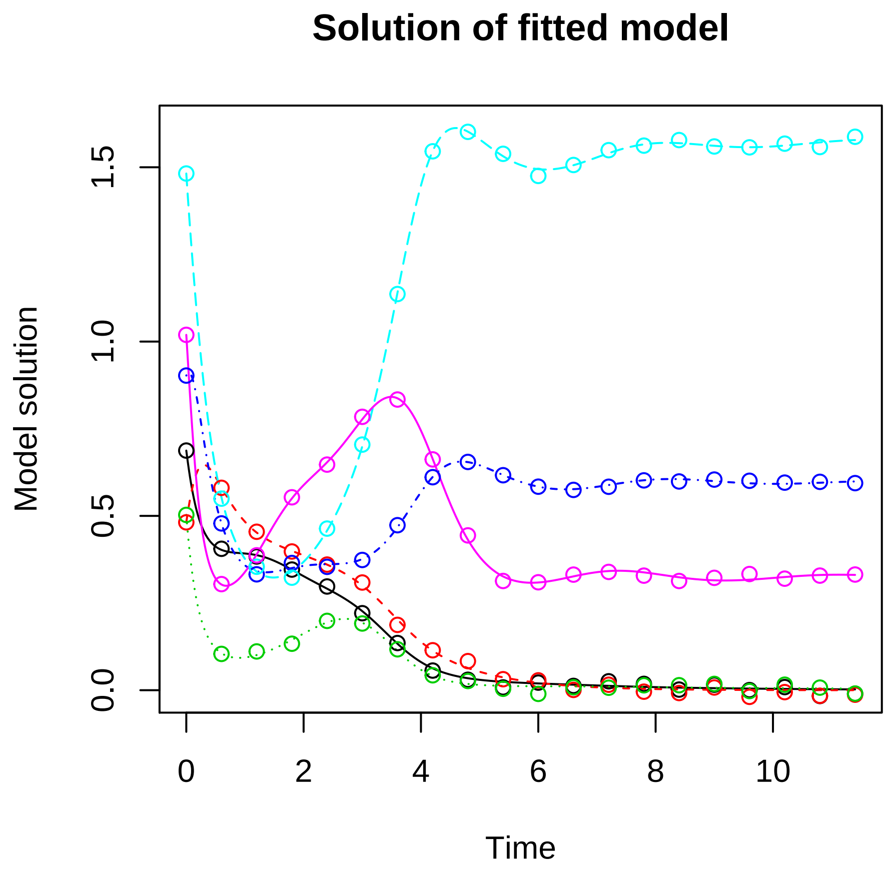
With standard deviations

```r
summary(nlrData2)
```

```
##           Estimate      StdErr t.value   p.value
##  [1,]  1.79223499  0.46961585  3.8164 0.0002698 ***
##  [2,] -1.14900477  0.54870800 -2.0940 0.0395064 *
##  [3,]  5.68942535  0.85978069  6.6173 4.173e-09 ***
##  [4,]  0.08948867  0.41054537  0.2180 0.8280178
##  [5,]  1.81274708  0.54137626  3.3484 0.0012536 **
##  [6,]  3.57599179  0.89334751  4.0029 0.0001416 ***
##  [7,] -1.67109050  3.64758521 -0.4581 0.6481285
##  [8,]  4.48139180  3.91428001  1.1449 0.2557586
##  [9,] -0.00054822  8.85457181 -0.0001 0.9999508
## [10,]  5.50970184  2.88580155  1.9092 0.0599081 .
## [11,] -3.49343286  3.28617960 -1.0631 0.2910312
## [12,]  2.98246399  4.98891756  0.5978 0.5516933
## [13,] -0.04198621  1.82769429 -0.0230 0.9817311
## [14,] -3.63452284  1.95242783 -1.8615 0.0664351 .
## [15,] -8.21869778  4.91626804 -1.6717 0.0985839 .
## [16,] -5.47304844  1.68613405 -3.2459 0.0017261 **
## [17,] -3.30201388  1.93846243 -1.7034 0.0924712 .
## [18,] -1.69752061  2.74310962 -0.6188 0.5378310
## [19,]  3.53269933  2.21432123  1.5954 0.1146698
## [20,]  2.71661123  2.24557865  1.2098 0.2300247
## [21,]  1.05303570  3.56257752  0.2956 0.7683344
## [22,] -4.05040893  1.55263263 -2.6087 0.0108885 *
## [23,] -0.08410942  1.24756004 -0.0674 0.9464206
## [24,]  0.48877777  1.94766130  0.2510 0.8025079
## [25,] -1.49761104  3.08391217 -0.4856 0.6285976
## [26,]  2.36320500  2.89647244  0.8159 0.4170473
## [27,]  1.79277129  5.86571855  0.3056 0.7606965
## [28,] -0.71401352  1.13928425 -0.6267 0.5326721
## [29,]  1.69767687  1.20417987  1.4098 0.1625679
## [30,] -9.79110852  2.13858372 -4.5783 1.745e-05 ***
## [31,] -0.33712206  1.30350650 -0.2586 0.7966043
## [32,] -0.77499375  1.28646795 -0.6024 0.5486421
## [33,] -3.98028829  2.54563956 -1.5636 0.1219667
## [34,] -0.23877962  0.68687232 -0.3476 0.7290520
## [35,] -2.24718204  0.68033025 -3.3031 0.0014452 **
## [36,]  1.73069704  1.13444722  1.5256 0.1311577
## [37,] -2.42812362  0.74038135 -3.2796 0.0015552 **
## [38,] -0.12147983  0.73655803 -0.1649 0.8694265
## [39,] -2.20948918  1.09072892 -2.0257 0.0462156 *
## [40,]  2.20400779  0.32405784  6.8013 1.877e-09 ***
## [41,]  2.20617034  0.27332327  8.0717 6.836e-12 ***
## [42,] -1.40436911  0.39934207 -3.5167 0.0007317 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can plot the solution of the fitted model with estimated parameter

```
plot(nlrData2, type="l", main="Solution of fitted model")
points(exampleData2)
```



Linear Regression

Interpretation of results and model identifiability

Standard error of model residuals vs Parameter collinearity, Summary of parameter estimates

GOTTA GET SENSITIVITY ANALYSIS RESULTS

```r
sd(nlrData2$Fit$residuals)
```

```
## [1] 0.008185737
```

Network structures