DR. CARLO PACIONI (Orcid ID : 0000-0001-5115-4120)

Article type   : Application

Handling editor: Dr. Sarah Goslee

vortexR: an R package for post Vortex simulation analysis

Carlo Pacioni[1,*] and Florian Mayer[2]

[1] School of Veterinary and Life Sciences, Murdoch University, Murdoch, WA, 6150, Australia.

[2] Department of Parks and Wildlife, 17 Dick Perry Avenue, Kensington 6151 WA, Australia

* Correspondence author. Email: carlo.pacioni@gmail.com

Summary

1. Population viability analysis is an important tool for wildlife ecologists, geneticists and managers, which is used for the assessment of extinction risks, the evaluation of threatening processes, and the establishment of conservation targets .

2. Vortex is among the leading population modelling software and the latest release includes an automated sensitivity test module. However, an equivalent automation of the post-simulation data inspection and analysis is currently missing.

3. *vortexR* is an R package to automate the analysis and visualisation of outputs from the population viability modelling software Vortex. *vortexR* facilitates collating Vortex output files, data visualisation and basic analyses (e.g. pairwise comparisons of scenarios), as well as providing more advanced statistics, such as searching for the best regression model(s) from a list of predictors to investigate the main effect and the interaction effects of the variables of interest.

4. This package speeds up and greatly facilitates the reproducibility and portability of post-simulation analysis results.

Tweetable Abstract

vortexR is an R package to automate the statistical analysis and visualisation of outputs of the PVA software Vortex.

Introduction

Population viability analysis is an important tool for wildlife ecologists, geneticists and managers. The use of population modelling includes the assessment of extinction risks (Lindenmayer et al. 1995; Fordham et al. 2014), the evaluation of the importance of threatening processes (e.g. Carrete *et al.* 2009; Manlik *et al.* 2016), and the establishment of conservation targets (Traill et al. 2009; Himes Boor 2014). Indeed, modelling of wildlife populations is a commonly required task, which is nowadays facilitated by the improved computation capacity of modern computers and increased availability of software to develop population viability models. Among these, one of the most commonly used is Vortex (Lacy 2000; Lacy & Pollak 2013) . Vortex (http://www.vortex10.org/Vortex10.aspx) is an incredibly versatile tool that is used to explore a large number of ecological and genetic questions. First developed in the early 1990s (Lacy 1993), Vortex became very popular as a result of being a free, highly flexible, user-friendly software. One of the most important changes in version 10 is that the sensitivity test module is almost entirely automated. The user can now set ranges for the parameters that need to be included in the sensitivity test and Vortex automatically generates the relevant combinations. This allows for a quick and easy setup of scenarios to evaluate the effect of individual parameters, as well as the interaction of parameters on the model forecast. As a result, a large number of data-files are generated and it rapidly becomes unpractical to manually analyse them. Even the simple graphical inspection of the trajectories may become a time consuming task. Therefore, an equivalent automation of the post-simulation data inspection and analysis was also required.

We introduce here *vortexR*: an R package that executes several operations on various Vortex outputs, including collating Vortex output files, generating plots and conducting basic analysis (e.g. pairwise comparisons of scenarios) and more advanced statistics such as searching for the best regression model(s) given a list of predictors. We included tasks of common interest to Vortex's user community to facilitate the evaluation and analysis of Vortex simulations. A summary of *vortexR* functionalities in comparisons with Vortex is presented in Table S1. The use of this package will facilitate a more standardised analytical approach and make studies more comparable. At the same time, *vortexR* exports Vortex data into a simple, machine-readable format (.csv), which lowers the barrier towards running analyses outside *vortexR*'s built-in ones. While we acknowledge that some limitations exist in terms of reproducibility because Vortex runs only on Windows operating system, *vortexR* provides an automated, cross-platform analytical framework that will ensure reproducibility of results starting from simulated data, which is considered a minimal standard in research fields that involve computer science (Peng 2011).

Description

We discuss *vortexR's* functionalities by grouping them in three components (Figure 1): data handling, visualisation and analysis. We provide worked examples in the supplementary material; further documentation is accessible from the package using the R command `?vortexR`.

The **data handling** component of *vortexR* is a collection of functions that either collate the results stored in different Vortex outputs or standardise their format for subsequent analyses.

Typically, Vortex stores data from different scenarios (or Sensitivity Testing – ST– samples) in different text files (that is, each parameter configuration is stored in one file) and different data types are stored in different files. For example, mean parameter values are saved in files with extension .dat or .stdat.

Demographic data for each iteration are located in files with the extension .yr while final expected heterozygosity is stored in files with the extension .run and so forth. When the simulations include multiple populations, data for these are stored in separate blocks within the same file.

By design, Vortex splits up its output into lots of small files with a nested structure. This design benefits manual inspection of individual files; however, it complicates the task of combining many Vortex outputs into a single tabular structure.

To facilitate the cumbersome task of combining the fragmented Vortex output, *vortexR* provides a set of 'collate' functions (`collate_one_dat; collate_dat; collate_run; collate_yr`).

The function `collate_proc_data` collates data generated with different Vortex modules. For example, running one simulation using the 'normal' Vortex module and one with the new ST module, these cannot be viewed or plotted together within Vortex's graphical user interface. `collate_proc_data` combines these data for use in other *vortexR* functions, or for export into a simple, machine-readable format (.csv).

The **data visualisation** functions generate line plots of multiple variables across simulated years, dot plots of mean values with standard deviation bars or matrix of pairwise scatter plots. This family of functions facilitate the creation of plots and their preparation in a high resolution, publication-ready format. The user can restrict analysis to a subset of populations or simulated time period as appropriate. Some manipulations of the plot aesthetics are offered within *vortexR* functions. However, *vortexR*'s plotting functions return ggplot objects (Wickham 2009), which can then be extensively manipulated as required.

The **data analysis** functions either calculate specific parameters or carry out statistical analysis. One function (`Ne`) automates the calculations of the effective population size (N_e) based on the loss of genetic diversity (expected heterozygosity) using the temporal approach:

$$N_e = -\frac{1}{\left(\frac{H_i}{H_f}\right)^{1/\tau} - 1} \times \frac{1}{2}$$

Where H_i and H_f are, respectively, the initial and final expected heterozygosity and τ is the number of generations that occurred between the indicated time interval. A second function (`Nadults`) calculates the effective number of adults (using demographic data) allowing estimation of N_e : N ratio (more information and examples for these functions are available in *vortexR*'s vignette).

A third function (`pairwise`) conducts pairwise comparisons of the simulated scenarios against a baseline scenario using sensitivity coefficients (SC, Drechsler, Burgman & Menkhorst 1998) and strictly standardised mean difference (SSDM, Zhang 2007). This may be relevant if the user is interested in testing the effect of one parameter at the time. It is possible to specify more than one simulated point in time and compare several parameters at once.

The sensitivity coefficients are calculated as (Drechsler, Burgman & Menkhorst 1998):

$$SC_i = \frac{V_i - V_B}{V_B}$$

where V_i and V_B are the mean value of the variable of interest in the *i*-th and the baseline scenarios, respectively.

The logit is used for probabilities (e.g. probability of extinction) (Drechsler, Burgman & Menkhorst 1998):

$$SC_i = -(\operatorname{logit} V_i - \operatorname{logit} V_B)$$

The strictly standardised mean difference is calculated as (Zhang 2007):

$$SSMD_i = \frac{V_i - V_B}{\sqrt{(s_i^2 + s_B^2)}}$$

with s being the standard deviations of the parameter being estimated. For probabilities, the formula becomes:

$$SSMD_i = \frac{V_B - V_i}{\sqrt{(s_i^2 + s_B^2)}}$$

Note that the direction of the changes is inverted if either of the statistics is calculated for probabilities. This provides a more intuitive result because, assuming that Vortex users are mainly interested in the probability of extinction, a negative result reflects an increased probability of extinction on the basis of a given scenario relative to the baseline scenario.

It should be also noted that, when SC is calculated for probabilities and one of these is zero, it has little meaning. Also, SC and $SSMD$ cannot be computed if the baseline variable is zero or both s are zero, respectively.

We included the commonly used SC in `pairwise` to allow comparisons with previous work, but the $SSMD$ offers some advantages: most importantly, it allows for a statistical test and considers the variability of the results, but it is not influenced by the sample size. These points are better illustrated with an example. Considering a baseline scenario with mean population size N_b = 5,000, s_b = 500, and aiming to compare two simulations where $N_1 = N_2$ = 4,000, but s_1 = 100 and s_2 = 450. Using SC the two coefficients are the same:

$SC_1 = SC_2$ = (4,000 - 5,000)/5,000 = -0.2

Using $SSMD$ we obtain two different values reflecting the larger N fluctuations in the second scenario:

$SSMD_1$ = (4,000 - 5,000)/√(100^2+500^2) = 1.961

$SSMD_2$ = (4,000 - 5,000)/√(450^2+500^2) = 1.487

Moreover, we can test the detected difference and verify that there is a statistical difference between N_1 and the baseline scenario (p = 0.025) while there is no statistical difference at all (p = 0.069) with the second scenario. Note that if we were to use a t-test, assuming that the user generated the results using (the quite standard) 1,000 iterations, both t-tests would result in p<0.0001, providing very little insight into the potentially important biological differences between these scenarios with the baseline. This results from the statistical significance of the t-test being influenced not only by the mean difference, but also by the sample size, which is typically large in simulation studies.

Additionally, `pairwise` provides the scenarios' rank based on the absolute value of the statistics (*SC* and *SSMD*) for each population. Lastly, the Kendall's coefficient of concordance (which ranges from 0 to 1, representing no agreement or complete agreement, respectively) is calculated to test whether the order of ranked scenarios is statistically consistent across the chosen points in time and parameters. For example, if the user simulated 100 years and used `pairwise` with `yrs=c(50,100)` and selected two parameters, say `params=c("Nall", "Het")`, the consistency of ranking will be tested across the four raters (i.e. `Nall` at year 50, and 100, and `Het` at year 50 and 100).

Depending on the simulation settings, `pairwise` can also evaluate the overall 'mean' effect of simulation parameters on the outcome variables of interest (i.e. the user is interested in the parameters' ranks, rather than scenarios' ranks). In these cases, *vortexR* will first compare all the scenarios against the baseline scenario and then, following Conroy and Brook (2003), calculate the mean *SC* and *SSMD* for each group and provide the ranks accordingly (see the package's vignette for details).

Vortex reports the probability of extinction on a per-year basis in the .dat (or .stdat) files, which are processed by *vortexR*. However, it may be relevant to calculate the cumulative probability of extinction. That is, the probability that a population may go extinct in any moment during the simulations (which is reported in the .sum files by Vortex, but not read in by *vortexR*). The function `Pextinct` calculates the cumulative probability of extinction and carries out a pairwise comparison with a baseline scenario using *SSMD*.

Pacioni et al (2017) defined the recovery rate as the mean growth rate over a period of time and used this as indication of the capacity of the population to recover from low density. *vortexR* calculates this parameter with the function `rRec`. This function also conducts statistical pairwise comparisons with a baseline scenario using *SSMD*. Examples for these functions are available in *vortexR*'s vignette.

The last function in this group is `fit_regression`, a first attempt to automate a general work-flow to explore the interactions of factors, and not only their main, independent effect. From a list of potential predictors, this function will conduct a search of the best regression model(s) using the R package `glmulti` (Calcagno & de Mazancourt 2010) for one or more populations.

In the initial fit of the model, the main and interactions effects are included and adequacy of the error distribution or link function (see below) is checked. Subsequently, the best model is sought limiting the search to the main effects only or all pairwise effects based on the user's selection.

The regression model fitted depend on the dependent variable. When this is a count (e.g. N) the function will fit a Generalized Linear Model. The first fit is attempted with a Poisson error distribution and if the dispersion parameter ($c\hat{} = \frac{residualdeviance}{df}$) is larger than 1.5, the model will be refitted with a quasipoisson error distribution. If the dependent variable is a proportion (i.e. heterozygosity or inbreeding), then the function uses a Beta regression (using the R package `betareg`, Cribari-Neto & Zeileis 2010). In the latter case, v*ortexR* tests different link functions and selects the one with the lowest AIC value. Beta regression models offer the dual advantage of adequately modelling data that are expressed in proportions and handling skewness in the distribution of the dependent variables. The trade-off is that, at least currently, they cannot handle analysis of data when the dependent variable takes value of either exactly 0 or 1 (Cribari-Neto & Zeileis 2010).

The function returns a `glmulti` object from which the user can extract model average estimates and other relevant information, as we show in the following examples.

Examples

When *vortexR* is installed, several examples can be retrieved by using the common R functions `help()` or `example()`. Further examples can be found in Pacioni et al (2017) and Campbell et al (2015) and in vortexR's vignette. In the supporting information (Appendix S1), we demonstrate *vortexR* usage using example data from these publications that are available when the package is installed.

Conclusions and future development

vortexR is, to our knowledge, the first attempt to automate the post-simulation analysis of the data generated by the very commonly used software Vortex. It provides a flexible tool to carry out quality check and data analysis, facilitate reproducible research and it is a mean to standardise analysis across projects. Possible future developments include parallelising some of the functions (especially reading multiple Vortex output files) to further improve the speed, and the inclusion of other output files that are not currently used by *vortexR*. Also, we envisage that the addition of zero-inflated models and generalised linear mixed models to `fit_regression` could further improve the suitability of this package to a wider range of analyses. *vortexR* is available under GPL-3 license from CRAN. To install from source, follow the instructions at *https://github.com/carlopacioni/vortexR*. The authors welcome feature requests, bug reports, and contributions via pull requests.

Acknowledgements

We are deeply indebted to Dr Robert Lacy, who has helped to clarify details about Vortex' algorithm and calculations during the development of *vortexR*. Moreover, he has been always opened to exchange opinions and ideas, generating many interesting conversations.

Some of the functionality included in *vortexR* was initially developed by CP as part of independent projects funded by Whiteman Park, the Western Australia Department of Agriculture and Food, and the Western Australia Department of Parks and Wildlife. We are grateful to these agencies for agreeing with the re-adaptation of these scripts for their use in *vortexR*.

Data accessibility

Data for the examples in this paper and the package's help documents and vignette are subsets of simulated data from Pacioni et al (2017) and Campbell et al (2016) and are stored in the R data package *vortexRdata*(`https://CRAN.R-project.org/package=vortexRdata`), which is available from CRAN and it is installed along side with *vortexR*. The data available can be retrieved with standard R commands:

```
# A list of data available
data(package="vortexRdata")

# Load the data in current R session
data("dataname") # dataname is the name of the dataset of interest

# Retrieve the path where the raw data are stored
pac.dir <- system.file("extdata", "pacioni", package="vortexRdata")
cam.dir <- system.file("extdata", "campbell", package="vortexRdata")
```

References

Calcagno, V. & de Mazancourt, C. (2010) glmulti: an R package for easy automated model selection with (generalized) linear models. *Journal of Statistical Software,* **34,** 1-29.

Campbell, S., Roberts, E.J., Craemer, R., Pacioni, C., Rollins, L. & Woolnough, A.P. (2015) Assessing the economic benefits of starling detection and control to Western Australia. *Australasian Journal of Environmental Management,* **23, 81-99**.

Carrete, M., Sánchez-Zapata, J.A., Benítez, J.R., Lobón, M. & Donázar, J.A. (2009) Large scale risk-assessment of wind-farms on population viability of a globally endangered long-lived raptor. *Biological Conservation,* **142,** 2954-2961.

Conroy, S.D.S. & Brook, B.W. (2003) Demographic sensitivity and persistence of the threatened white- and orange-bellied frogs of Western Australia. *Population Ecology,* **45,** 105-114.

Cribari-Neto, F. & Zeileis, A. (2010) Beta regression in R. *Journal of Statistical Software,* **34**.

Drechsler, M., Burgman, M.A. & Menkhorst, P.W. (1998) Uncertainty in population dynamics and its consequences for the management of the orange-bellied parrot Neophema chrysogaster. *Biological Conservation,* **84,** 269-281.

Fordham, D.A., Shoemaker, K.T., Schumaker, N.H., Akçakaya, H.R., Clisby, N. & Brook, B.W. (2014) How interactions between animal movement and landscape processes modify local range dynamics and extinction risk. *Biology Letters,* **10,** 20140198.

Himes Boor, G.K. (2014) A Framework for Developing Objective and Measurable Recovery Criteria for Threatened and Endangered Species. *Conservation Biology,* **28,** 33-43.

Lacy, R.C. (1993) VORTEX: a computer simulation model for population viability analysis. *Wildlife Research,* **20,** 45-65.

Lacy, R.C. (2000) Structure of the VORTEX simulation model for population viability analysis. *ecological Bulletins***,** 191-203.

Lacy, R.C. & Pollak, J.P. (2013) VORTEX: a stochastic simulation of the extinction process. Version 10. *Chicago Zoological Society, Brookfield*.

Lindenmayer, D.B., Burgman, M.A., Akçakaya, H.R., Lacy, R.C. & Possingham, H.P. (1995) A review of the generic computer programs ALEX, RAMAS/space and VORTEX for modelling the viability of wildlife metapopulations. *Ecological Modelling,* **82,** 161-174.

Manlik, O., McDonald, J.A., Mann, J., Raudino, H.C., Bejder, L., Krützen, M., Connor, R.C., Heithaus, M.R., Lacy, R.C. & Sherwin, W.B. (2016) The relative importance of reproduction and survival for the conservation of two dolphin populations. *Ecology and evolution,* **6,** 3496-3512.

Pacioni, C., Williams, M., Lacy, R.C., Spencer, P.B.S. & Wayne, A.F. (2017) Predators and genetic fitness: key threatening factors for the conservation of bettong species. *Pacific conservation biology*. In press

Peng, R.D. (2011) Reproducible Research in Computational Science. *Science (New York, N.y.),* **334,** 1226-1227.

Stein, M. (1987) Large Sample Properties of Simulations Using Latin Hypercube Sampling. *Technometrics,* **29,** 143-151.

Traill, L.W., Brook, B.W., Frankham, R.R. & Bradshaw, C.J.A. (2009) Pragmatic population viability targets in a rapidly changing world. *Biological Conservation,* **143,** 28-34.

Wickham, H. (2009) *ggplot2: elegant graphics for data analysis*. Springer.

Zhang, X.D. (2007) A pair of new statistical parameters for quality control in RNA interference high-throughput screening assays. *Genomics,* **89,** 552-561.

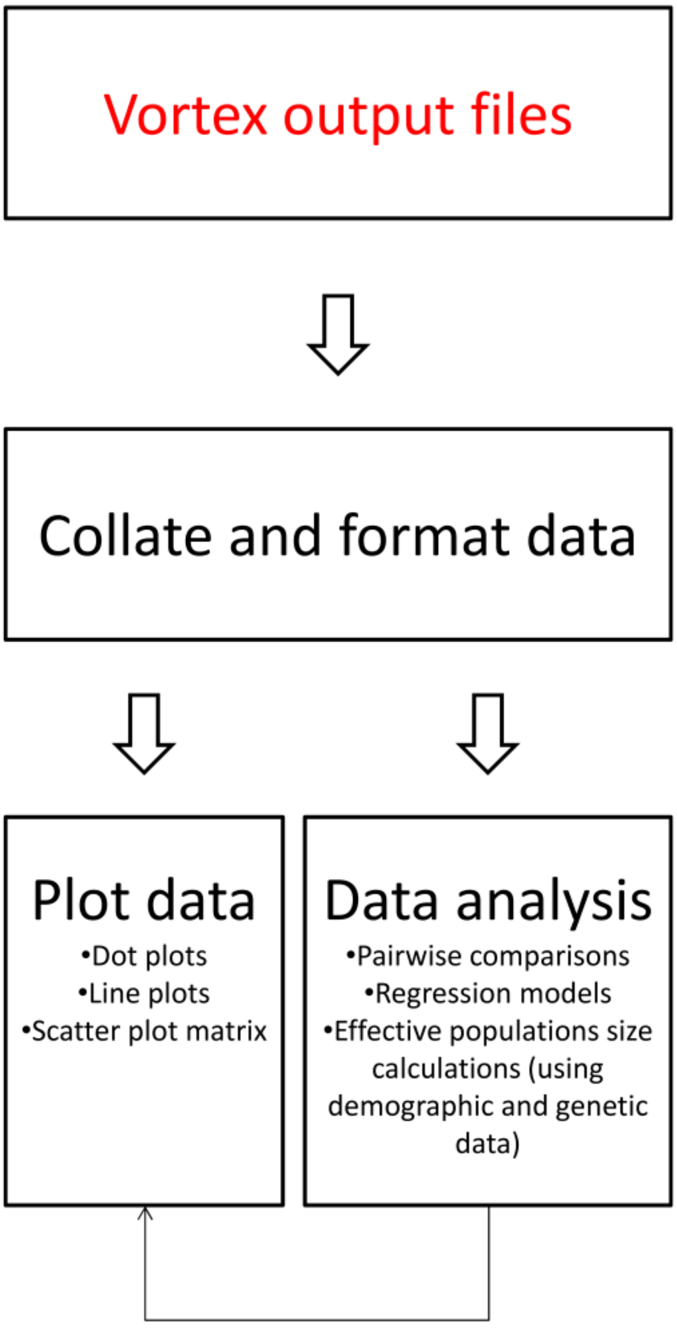
Supporting information

Table S1. Table summarising comparison of functionality between Vortex and *vortexR*.

Appendix S1. Worked examples demonstrating *vortexR* usage.

Figure legends

Figure 1. A possible workflow using *vortexR*. After Vortex has generated the output files, the data are collated into one R object. Plots are generated to visualise trends and inspect the behaviour and performance of Vortex's models. Lastly, the analysis of the data is carried out and relevant plots may be revisited to modify them for publications or reports. Some functions (e.g. `fit_regression`) will also generate separate plots to visualise results and diagnostics.



Author contribution statement

CP conceived the idea and wrote the manuscript with input from FM. Both authors wrote and tested the scripts. FM implemented the initial version as R package. Both authors gave final approval for publication.

vortexR: an R package for post Vortex simulation analysis

Appendix: vortexR examples

Pacioni C. and Mayer F.

Introduction

This document showcases `vortexR`'s **Data handling**, **Data visualisation**, and **Data analysis** of Vortex output using included sample data from two real-world use cases.

`vortexR` initially parses a number of Vortex output files into a single R object which can be saved as R data and text files (`save2disk=TRUE`) and then further visualised and analysed. Because 'vortexR' reads in the data from Vortex, all parameters will be named as for default by Vortex (e.g. Nall, Next, Het, etc.). It is assumed that the reader is familiar with Vortex definition of parameter, if not, please refer to Vortex manual.

Installation

`vortexR` can be installed using:

```
install.packages("vortexR")
```

`Alternatively, vortexR` can be installed from source using devtools:

```
library(devtools)
install_github("carlopacioni/vortexR")
```

The example data

When `vortexR` is installed, the package `vortexRdata` is also installed. The latter contains a few files that can be used to run the example presented here and in the package documentation.

To limit the amount of downloaded data when the package is installed, the 'pacioni' data are a subset of Pacioni et al (2017), containing only three runs for 120 time-steps, and only the scenarios 'ST_Classic' and 'ST_LHS', so the results do not match exactly the ones reported in the original publication. In this work, the authors developed a baseline PVA model for the woylie (*Bettongia penicillata*). They conducted a sensitivity analysis by varying the carrying capacity, the mortality rates for each age class, the standard deviation of these rates, the mate monopolization and the initial population size using the Single-Factor option of the ST module in Vortex. They evaluated the individual impact of each of these parameters (each parameter is modified one at the time) as well as at their interactions. For the latter, they used the Latin Hypercube Sampling to vary parameters concurrently.

Campbell et al (2015) investigated the cost-benefit ratios of different control levels (scenarios 'control_09', 'control_11', 'control_D09'), with or without the inclusion of new technology (scenarios ending with the suffix 'Tech') for the starling (*Sturnus vulgaris*) in Western Australia. They also evaluated whether the variation of the geographical extent of the application of control measures would alter the cost-benefit ratio of the control program (scenarios ending with the suffix 'Border' and 'SC'). Lastly, they evaluated whether the cyclical application of intense control ('Major Reduction'), would be more cost-effective than sustaining a constant level of investment. The latter component was modelled with a ST module in Vortex, while the previous were developed using the standard Vortex interface.

Data handling

This section describes `vortexR`'s data handling capabilities, leading from a number of Vortex output files to a single R object, and optionally a spreadsheet.

collate_dat

In the following example, a Sensitivity Test module (ST) was used in a Vortex project, which generates files with extension `.stdat`. `collate_dat` requires the ST project's and scenario's names (`"Pacioni_et_al"` and `"ST_Classic"`) as parameter. As with all other R functions, `?collate_dat` will show detailed documentation including required function parameters and example usage. The consolidated dataset is stored in the object `woylie.st.classic`.

```
library(vortexR)
pac.dir <- system.file("extdata", "pacioni", package="vortexRdata")
woylie.st.classic <- collate_dat("Pacioni_et_al", 3, scenario = "ST_Classic",
                                 dir_in = pac.dir, save2disk=FALSE, verbose=FALSE)
```

The first five rows and fist five columns of the dataset should look like the following:

```
woylie.st.classic[1:5, 1:5]

##        scen.name     pop.name Year PExtant SE.PExtant.
## 1 ST_Classic(1) Population 1    0       1           0
## 2 ST_Classic(1) Population 1    1       1           0
## 3 ST_Classic(1) Population 1    2       1           0
## 4 ST_Classic(1) Population 1    3       1           0
## 5 ST_Classic(1) Population 1    4       1           0
```

collate_yr

`collate_yr` collates all .yr files of the Vortex project and the specified ST scenario (here `ST_Classic`), and returns a list that we name here `yr.st.classic`.

```r
yr.st.classic <- collate_yr(project="Pacioni_et_al", scenario="ST_Classic",
                            dir_in = pac.dir, save2disk=FALSE, verbose=FALSE)
```

Again, we can inspect the first few rows and columns of the first element of the list with:

```
yr.st.classic[[1]][1:5, 1:8, with=FALSE]
```

```
##         Scenario Iteration Year Npop1 AMpop1 AFpop1 Subadultspop1 Juvpop1
## 1: ST_Classic(1)         1    0   300     97    117            24      62
## 2: ST_Classic(1)         2    0   300     97    117            24      62
## 3: ST_Classic(1)         3    0   300     97    117            24      62
## 4: ST_Classic(1)         1    1   361    116    137            28      80
## 5: ST_Classic(1)         2    1   357    115    136            29      77
```

The second element of the list includes the same data averaged across iterations. The averaged data show the mean number of animals that moved between populations (e.g. migration or translocations), or subclasses of individuals (e.g. by gender or age):

```
yr.st.classic[[2]][1:5, 1:7, with=FALSE]
```

```
##         Scenario Year    Npop1   AMpop1   AFpop1 Subadultspop1   Juvpop1
## 1: ST_Classic(1)    0 300.0000  97.0000 117.0000      24.00000  62.00000
## 2: ST_Classic(1)    1 362.0000 117.0000 138.0000      28.66667  78.33333
## 3: ST_Classic(1)    2 433.6667 141.0000 172.3333      34.00000  86.33333
## 4: ST_Classic(1)    3 500.6667 165.0000 200.0000      35.66667 100.00000
## 5: ST_Classic(1)    4 481.3333 166.6667 202.6667      40.66667  71.33333
```

lookup_table

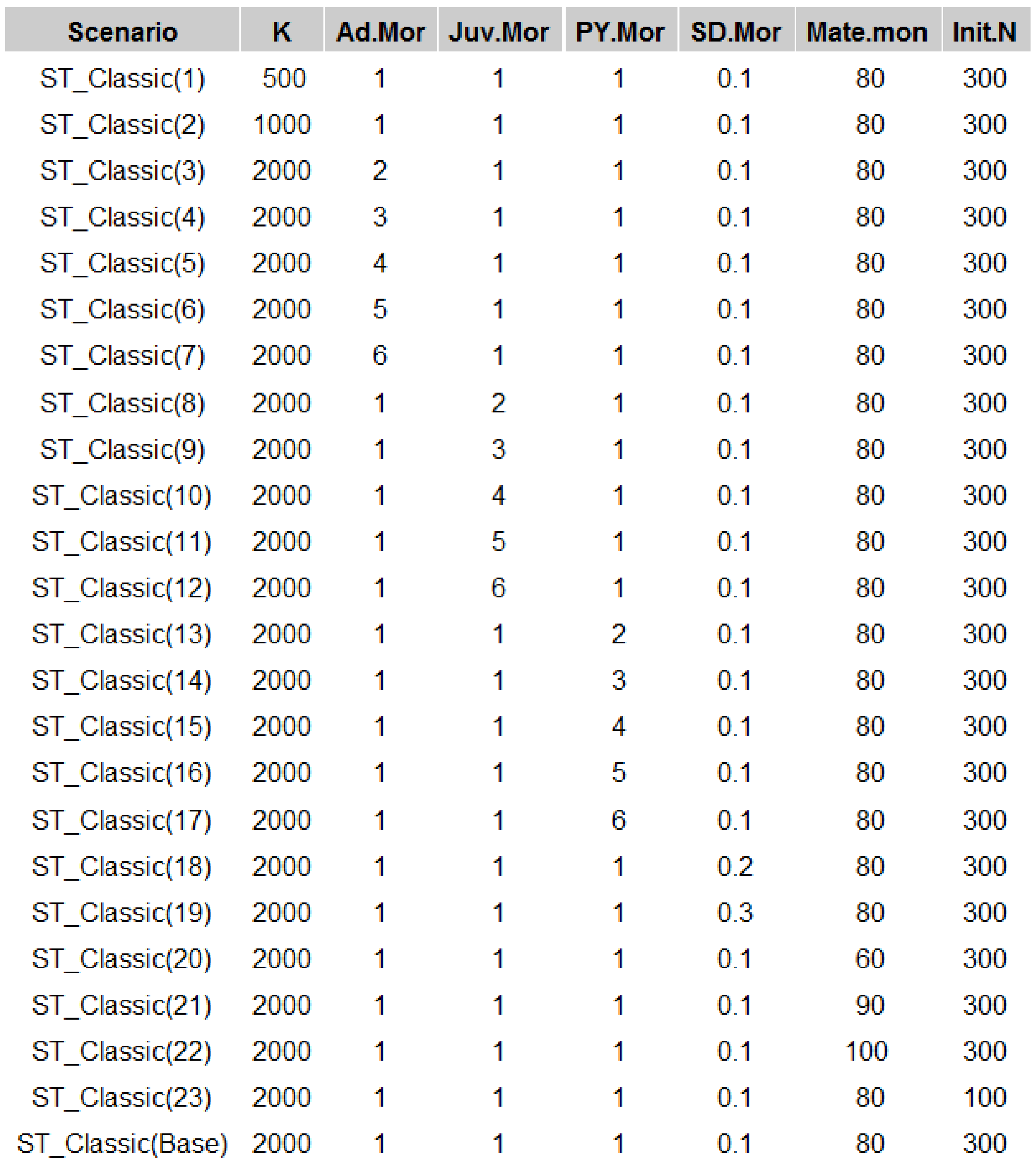
Lastly, `lookup_table` obtains a summary of the parameters' values used in the ST scenario that was collated before.

```
lkup.st.classic <- lookup_table(data=woylie.st.classic,
project="Pacioni_et_al",
                   scenario="ST_Classic", pop="Population 1",
                   SVs=c("SV1", "SV2", "SV3", "SV4", "SV5", "SV6", "SV7"),
                   save2disk=FALSE)
head(lkup.st.classic)
```

```
##         Scenario  SV1 SV2 SV3 SV4 SV5 SV6 SV7
## 1: ST_Classic(1)  500   1   1   1 0.1  80 300
## 2: ST_Classic(2) 1000   1   1   1 0.1  80 300
## 3: ST_Classic(3) 2000   2   1   1 0.1  80 300
## 4: ST_Classic(4) 2000   3   1   1 0.1  80 300
## 5: ST_Classic(5) 2000   4   1   1 0.1  80 300
## 6: ST_Classic(6) 2000   5   1   1 0.1  80 300
```

The variable names originate from Vortex outputs and may have little meaning outside the particular Vortex project. For readability, we will replace them with concise, but meaningful mnemonics.

```
library(data.table, quietly=TRUE)
library(gridExtra, quietly=TRUE)
setnames(lkup.st.classic, c("Scenario", "K", "Ad.Mor", "Juv.Mor",
                          "PY.Mor", "SD.Mor", "Mate.mon", "Init.N"))
grid.table(lkup.st.classic, rows=NULL,
           theme=ttheme_default(core=list(bg_params=list(fill="white"))))
```



Scenario	K	Ad.Mor	Juv.Mor	PY.Mor	SD.Mor	Mate.mon	Init.N
ST_Classic(1)	500	1	1	1	0.1	80	300
ST_Classic(2)	1000	1	1	1	0.1	80	300
ST_Classic(3)	2000	2	1	1	0.1	80	300
ST_Classic(4)	2000	3	1	1	0.1	80	300
ST_Classic(5)	2000	4	1	1	0.1	80	300
ST_Classic(6)	2000	5	1	1	0.1	80	300
ST_Classic(7)	2000	6	1	1	0.1	80	300
ST_Classic(8)	2000	1	2	1	0.1	80	300
ST_Classic(9)	2000	1	3	1	0.1	80	300
ST_Classic(10)	2000	1	4	1	0.1	80	300
ST_Classic(11)	2000	1	5	1	0.1	80	300
ST_Classic(12)	2000	1	6	1	0.1	80	300
ST_Classic(13)	2000	1	1	2	0.1	80	300
ST_Classic(14)	2000	1	1	3	0.1	80	300
ST_Classic(15)	2000	1	1	4	0.1	80	300
ST_Classic(16)	2000	1	1	5	0.1	80	300
ST_Classic(17)	2000	1	1	6	0.1	80	300
ST_Classic(18)	2000	1	1	1	0.2	80	300
ST_Classic(19)	2000	1	1	1	0.3	80	300
ST_Classic(20)	2000	1	1	1	0.1	60	300
ST_Classic(21)	2000	1	1	1	0.1	90	300
ST_Classic(22)	2000	1	1	1	0.1	100	300
ST_Classic(23)	2000	1	1	1	0.1	80	100
ST_Classic(Base)	2000	1	1	1	0.1	80	300

Where:

- K: carrying capacity,
- Ad.Mor: adult mortality,
- Juv.Mor: juvenile mortality,
- PY.Mor: pouch young mortality,
- SD.Mor: standard deviation of mortality rates,
- Mate.mon: mate monopolization, and
- Init.N: initial population size.

These data are now available for further data analysis either with `vortexR` or other software.

Data visualisation

This section demonstrates `vortexR`'s data visualisation capabilities. Recall that all parameters are identified as for default by Vortex (e.g. Nall, Next, Het, etc.). If uncertain, the reader is referred to Vortex manual.

dot_plot

The following will generate a dot plot of mean values with standard deviations for specific years (here: faceted for year 80 and 120). Parameter values are on the Y-axis and scenarios are on the X-axis. `dot_plot` uses data prepared by `collate_dat` and Vortex parameter names.

```
dot <- dot_plot(data=woylie.st.classic, project="Pacioni_et_al",
                scenario="ST_Classic", yrs=c(80, 120), params="Nall",
                save2disk=FALSE)
```

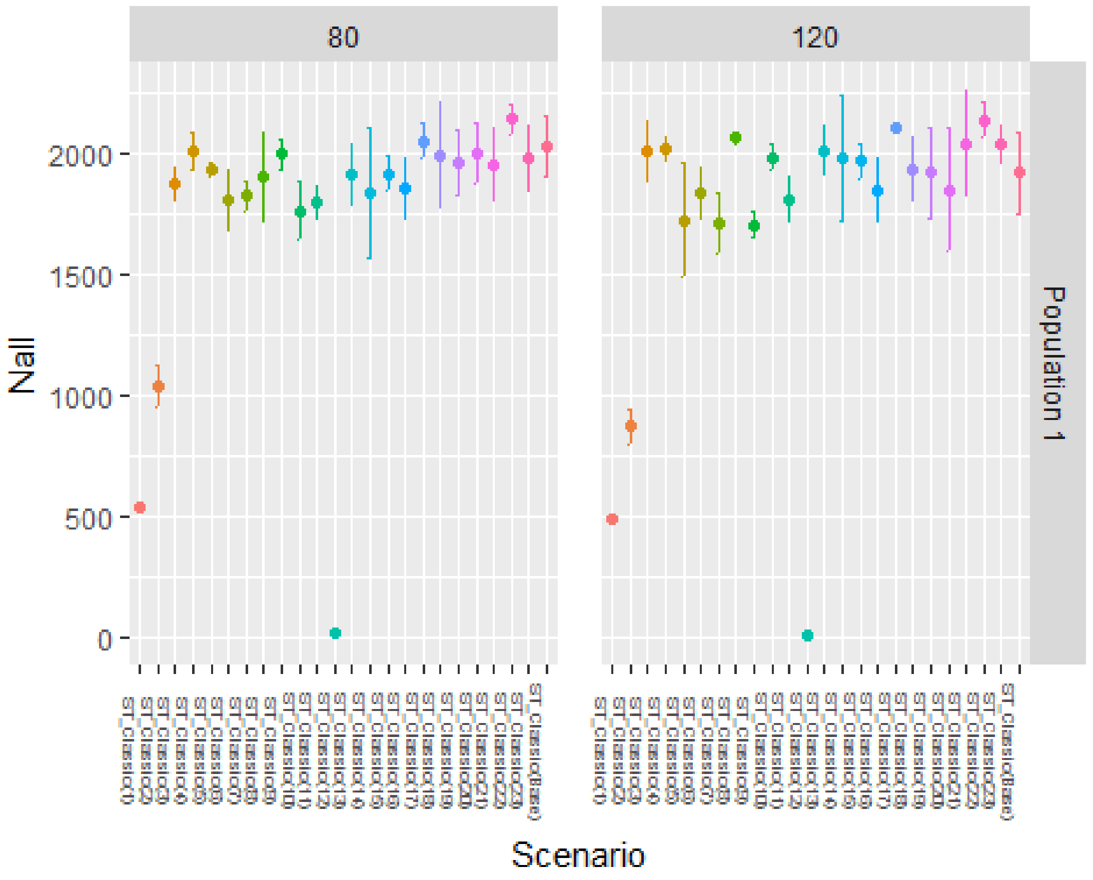


Figure S1. *Dot plot of mean values with standard deviations for year 80 and 120. Parameter values are on the Y-axis and scenarios are on the X-axis*

line_plot_year & line_plot_year_mid

The following two plots visualise changes of population size through time. The only difference between the two is that the time window in the second plot is limited between the beginning of the simulation and a `yrmid` year, in this case 50.

```
lineplot.st.classic <- line_plot_year(data=woylie.st.classic,
project="Pacioni_et_al",
                                      scenario="ST_Classic", params="Nall",
save2disk=FALSE)
```

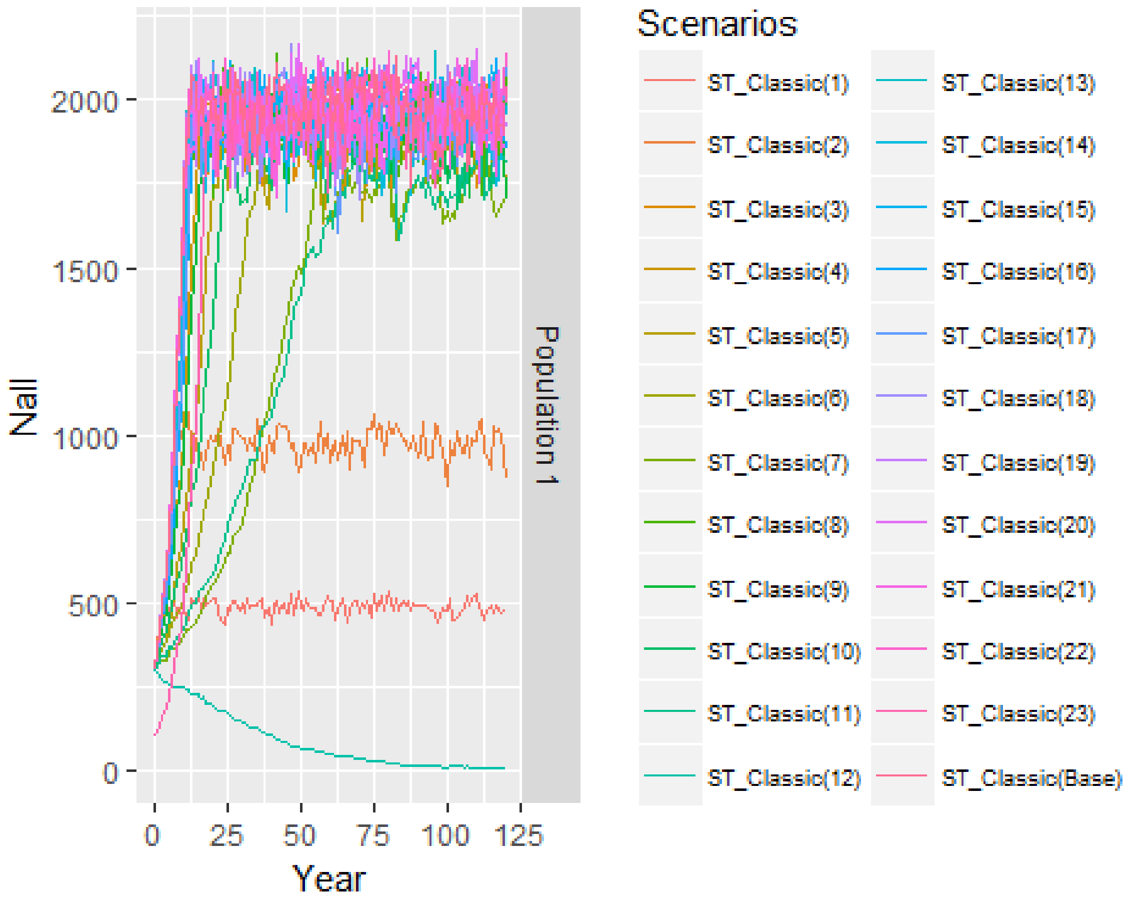


Figure S2. *Line plot of mean population size values (Y-axis) over simulated years. Each colour represent a different scenario.*

Although these plots are crowded by the multitude of scenarios, but they can be useful to give an overview. It is also very easy to subset the dataset based on parameter values or specific scenarios. To demonstrate this point, we include in the second plot only scenarios where the adult mortality is different from the baseline scenario.

```
lineMidPlot.st.classic <-
line_plot_year_mid(woylie.st.classic[woylie.st.classic$SV2 > 1, ],
                                        project="Pacioni_et_al",
scenario="ST_Classic",
                                                    yrmid=50, params="Nall",
save2disk=FALSE)

## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property
## instead
```

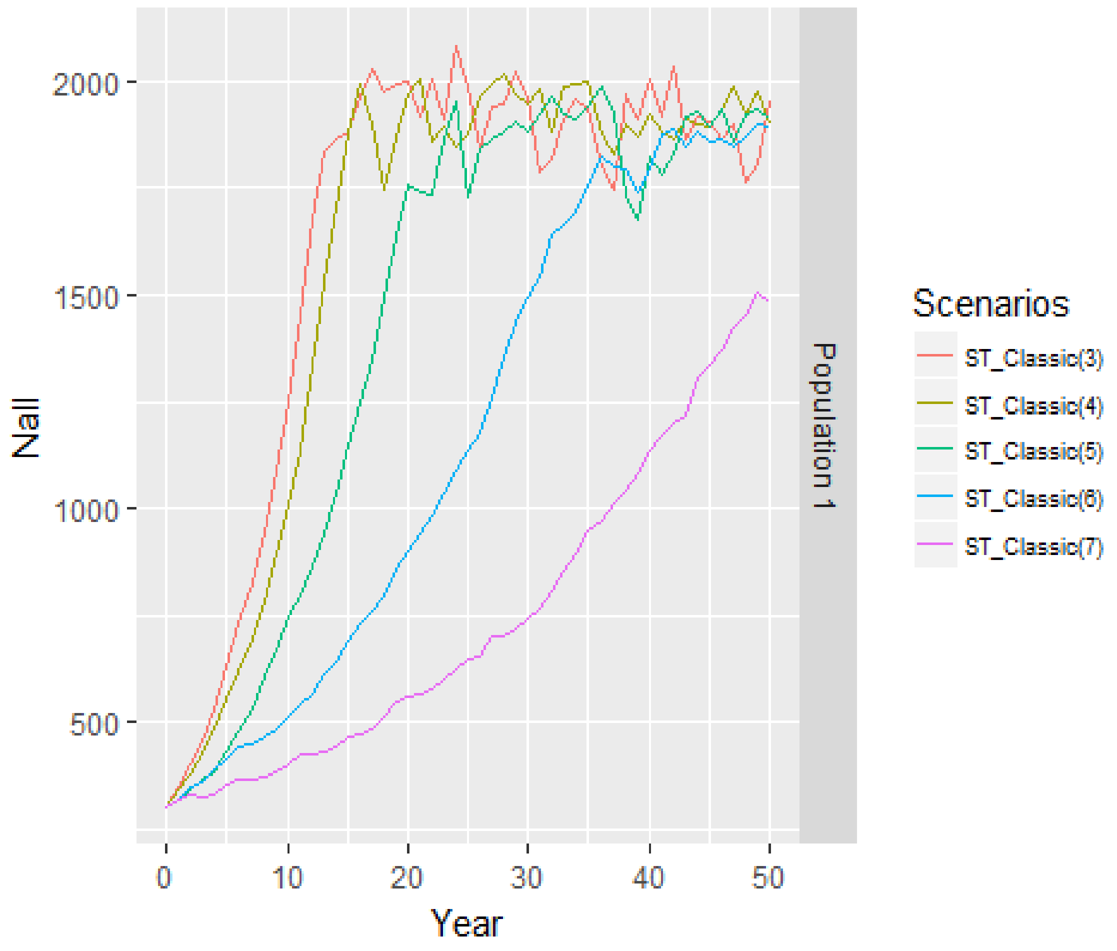


Figure S3. *Line plot of mean population size values (Y-axis) over simulated years. Note how the time window from the beginning of the population to the year 50 (as set by the yrmid argument). Each colour represent a different scenario.*

m_scatter

For brevity, the last type of plot (`m_scatter`) will be discussed together with the function `fit_regression`.

Data analysis

We focus our attention on `pairwise` and `fit_regression`, and refer the user to the package documentations for examples on the functions for the calculation of the effective population size (`Ne` and `Nadults`), the probability of extinction (`Pextinct`) and the recovery rate (`rRec`).

pairwise

We use `pairwise` to compare the scenarios of the sensitivity test 'ST_Classic' against its baseline scenario. This function returns a list with several elements that contained the results. We start exploring the first three, which include the sensitivity coefficients (SC), the strictly standardised mean difference (SSMD), and the p-values associated with SSMD.

```
pairw<-pairwise(data=woylie.st.classic, project="Pacioni_et_al",
scenario="ST_Classic",
                params=c("Nall", "Het"), yrs=120, ST=T, type="Single-Factor",
                SVs=c("SV1", "SV2", "SV3", "SV4", "SV5", "SV6", "SV7"),
                save2disk=FALSE)
pairw[[1]]
```

```
##          Scenario   Population    SC_Nall120     SC_Het120
## 1   ST_Classic(1) Population 1 -0.745396246 -0.0197640415
## 2   ST_Classic(2) Population 1 -0.545881126 -0.0075627710
## 3   ST_Classic(3) Population 1  0.047095933 -0.0012100434
## 4   ST_Classic(4) Population 1  0.054223149 -0.0028234345
## 5   ST_Classic(5) Population 1 -0.101840459 -0.0038318040
## 6   ST_Classic(6) Population 1 -0.043623566 -0.0063527276
## 7   ST_Classic(7) Population 1 -0.106882169 -0.0129071292
## 8   ST_Classic(8) Population 1  0.075599583  0.0000000000
## 9   ST_Classic(9) Population 1 -0.110703858 -0.0003025108
## 10 ST_Classic(10) Population 1  0.034760167 -0.0008066956
## 11 ST_Classic(11) Population 1 -0.055615224 -0.0028234345
## 12 ST_Classic(12) Population 1 -0.995135558 -1.0000000000
## 13 ST_Classic(13) Population 1  0.049880083 -0.0006050217
## 14 ST_Classic(14) Population 1  0.030933264 -0.0004033478
## 15 ST_Classic(15) Population 1  0.025547445  0.0001008369
## 16 ST_Classic(16) Population 1 -0.035974974  0.0003025108
## 17 ST_Classic(17) Population 1  0.096105318  0.0002016739
## 18 ST_Classic(18) Population 1  0.008342023 -0.0006050217
## 19 ST_Classic(19) Population 1  0.000000000 -0.0001008369
## 20 ST_Classic(20) Population 1 -0.035453597 -0.0004033478
## 21 ST_Classic(21) Population 1  0.063779979 -0.0003025108
## 22 ST_Classic(22) Population 1  0.114353493 -0.0003025108
## 23 ST_Classic(23) Population 1  0.062737226 -0.0063527276
```

```
pairw[[2]]
```

```
##          Scenario   Population SSMD_Nall120   SSMD_Het120
## 1   ST_Classic(1) Population 1  -8.26099855    -7.4508020
## 2   ST_Classic(2) Population 1  -5.59253479    -2.7477874
## 3   ST_Classic(3) Population 1   0.41904660    -2.6832816
## 4   ST_Classic(4) Population 1   0.58029707    -3.4729726
## 5   ST_Classic(5) Population 1  -0.66510998    -5.9346030
## 6   ST_Classic(6) Population 1  -0.41449553    -4.9805873
## 7   ST_Classic(7) Population 1  -0.95694624   -10.9357780
## 8   ST_Classic(8) Population 1   0.83068382     0.0000000
## 9   ST_Classic(9) Population 1  -1.18040589    -0.5303301
## 10 ST_Classic(10) Population 1   0.36870131    -1.4142136
## 11 ST_Classic(11) Population 1  -0.54150939    -2.0586009
## 12 ST_Classic(12) Population 1 -10.99812985 -2479.2500000
## 13 ST_Classic(13) Population 1   0.48193240    -0.6708204
## 14 ST_Classic(14) Population 1   0.18928233    -0.6246950
## 15 ST_Classic(15) Population 1   0.26307009     0.2000000
## 16 ST_Classic(16) Population 1  -0.31656651     0.3721042
## 17 ST_Classic(17) Population 1   1.06177944     0.3535534
## 18 ST_Classic(18) Population 1   0.07427443    -0.5570860
## 19 ST_Classic(19) Population 1   0.00000000    -0.1386750
## 20 ST_Classic(20) Population 1  -0.22067003    -0.3713907
## 21 ST_Classic(21) Population 1   0.44096584    -0.6708204
## 22 ST_Classic(22) Population 1   1.17067958    -0.3354102
## 23 ST_Classic(23) Population 1   0.63823134    -7.0436141
```

```
pairw[[3]]
```

```
##          Scenario   Population SSMD_Nall120  SSMD_Het120
## 1   ST_Classic(1) Population 1 7.222942e-17 4.638725e-14
## 2   ST_Classic(2) Population 1 1.118891e-08 2.999945e-03
## 3   ST_Classic(3) Population 1 3.375910e-01 3.645179e-03
## 4   ST_Classic(4) Population 1 2.808572e-01 2.573640e-04
## 5   ST_Classic(5) Population 1 2.529901e-01 1.472787e-09
## 6   ST_Classic(6) Population 1 3.392556e-01 3.169580e-07
## 7   ST_Classic(7) Population 1 1.692972e-01 3.886815e-28
## 8   ST_Classic(8) Population 1 2.030761e-01 5.000000e-01
## 9   ST_Classic(9) Population 1 1.189194e-01 2.979415e-01
## 10 ST_Classic(10) Population 1 3.561752e-01 7.864960e-02
## 11 ST_Classic(11) Population 1 2.940783e-01 1.976624e-02
## 12 ST_Classic(12) Population 1 1.950695e-28 0.000000e+00
## 13 ST_Classic(13) Population 1 3.149270e-01 2.511675e-01
## 14 ST_Classic(14) Population 1 4.249358e-01 2.660856e-01
## 15 ST_Classic(15) Population 1 3.962483e-01 4.207403e-01
## 16 ST_Classic(16) Population 1 3.757863e-01 3.549076e-01
## 17 ST_Classic(17) Population 1 1.441679e-01 3.618368e-01
## 18 ST_Classic(18) Population 1 4.703960e-01 2.887343e-01
## 19 ST_Classic(19) Population 1 5.000000e-01 4.448535e-01
## 20 ST_Classic(20) Population 1 4.126747e-01 3.551733e-01
## 21 ST_Classic(21) Population 1 3.296189e-01 2.511675e-01
## 22 ST_Classic(22) Population 1 1.208638e-01 3.686578e-01
## 23 ST_Classic(23) Population 1 2.616615e-01 9.365822e-13
```

Let us focus on the table with p-values of the scenario comparisons with SSMD (the element 3 of the list). To make it easier, we round the p-values to the fourth decimal digit.

```
pval<-pairw[[3]]
pval$SSMD_Nall120<-round(pval$SSMD_Nall120, 4)
pval$SSMD_Het120<-round(pval$SSMD_Het120, 4)
pval
```

```
##          Scenario   Population SSMD_Nall120 SSMD_Het120
## 1   ST_Classic(1) Population 1       0.0000      0.0000
## 2   ST_Classic(2) Population 1       0.0000      0.0030
## 3   ST_Classic(3) Population 1       0.3376      0.0036
## 4   ST_Classic(4) Population 1       0.2809      0.0003
## 5   ST_Classic(5) Population 1       0.2530      0.0000
## 6   ST_Classic(6) Population 1       0.3393      0.0000
## 7   ST_Classic(7) Population 1       0.1693      0.0000
## 8   ST_Classic(8) Population 1       0.2031      0.5000
## 9   ST_Classic(9) Population 1       0.1189      0.2979
## 10 ST_Classic(10) Population 1       0.3562      0.0786
## 11 ST_Classic(11) Population 1       0.2941      0.0198
## 12 ST_Classic(12) Population 1       0.0000      0.0000
## 13 ST_Classic(13) Population 1       0.3149      0.2512
## 14 ST_Classic(14) Population 1       0.4249      0.2661
## 15 ST_Classic(15) Population 1       0.3962      0.4207
## 16 ST_Classic(16) Population 1       0.3758      0.3549
## 17 ST_Classic(17) Population 1       0.1442      0.3618
## 18 ST_Classic(18) Population 1       0.4704      0.2887
## 19 ST_Classic(19) Population 1       0.5000      0.4449
## 20 ST_Classic(20) Population 1       0.4127      0.3552
## 21 ST_Classic(21) Population 1       0.3296      0.2512
## 22 ST_Classic(22) Population 1       0.1209      0.3687
## 23 ST_Classic(23) Population 1       0.2617      0.0000
```

It is evident that the changes in K have a significant effect because both scenarios 1 and 2 are statistically different for both parameters (Nall and Het) from the baseline (refer to the lookup table generated above to identify the variable that have been changed in each scenario). Changes in the adult mortality rate (scenarios 3 to 7) are responsible for a significantly reduced heterozygosity. When the mortality rate of juveniles is increased by a factor of five or six (scenario 11 and 12), the differences are significant in one or both parameters (it can be noted from the dot plot above that in scenario 12 the population goes extinct). Lastly, when initial population size is changed to 100 individuals, the final heterozygosity is also significantly reduced.

Next in the list `pairw` there are: the scenario ranks based on SC and SSMD, and the results of the Kendall's coefficient of concordance test. In this example, we used the population size (Nall) and heterozygosity (Het) at year 120 to compare the scenarios. The headings of the columns with the ranks synthesise the information.

```
pairw[[4]]

##       Population       Scenario SC_Nall120 SC_Het120
##  1: Population 1  ST_Classic(1)          2       2.0
##  2: Population 1  ST_Classic(2)          3       4.0
##  3: Population 1  ST_Classic(3)         15      10.0
##  4: Population 1  ST_Classic(4)         13       8.5
##  5: Population 1  ST_Classic(5)          7       7.0
##  6: Population 1  ST_Classic(6)         16       5.5
##  7: Population 1  ST_Classic(7)          6       3.0
##  8: Population 1  ST_Classic(8)         9      23.0
##  9: Population 1  ST_Classic(9)         5      17.0
## 10: Population 1 ST_Classic(10)        19      11.0
## 11: Population 1 ST_Classic(11)        12       8.5
## 12: Population 1 ST_Classic(12)         1       1.0
## 13: Population 1 ST_Classic(13)        14      12.5
## 14: Population 1 ST_Classic(14)        20      14.5
## 15: Population 1 ST_Classic(15)        21      21.5
## 16: Population 1 ST_Classic(16)        17      19.0
## 17: Population 1 ST_Classic(17)         8      20.0
## 18: Population 1 ST_Classic(18)        22      12.5
## 19: Population 1 ST_Classic(19)        23      21.5
## 20: Population 1 ST_Classic(20)        18      14.5
## 21: Population 1 ST_Classic(21)        10      17.0
## 22: Population 1 ST_Classic(22)         4      17.0
## 23: Population 1 ST_Classic(23)        11       5.5
##       Population       Scenario SC_Nall120 SC_Het120
```

```
pairw[[5]]
```

```
##       Population       Scenario SSMD_Nall120 SSMD_Het120
##  1: Population 1  ST_Classic(1)            2           3
##  2: Population 1  ST_Classic(2)            3           8
##  3: Population 1  ST_Classic(3)           15           9
##  4: Population 1  ST_Classic(4)           11           7
##  5: Population 1  ST_Classic(5)            9           5
##  6: Population 1  ST_Classic(6)           16           6
##  7: Population 1  ST_Classic(7)            7           2
##  8: Population 1  ST_Classic(8)            8          23
##  9: Population 1  ST_Classic(9)            4          16
## 10: Population 1 ST_Classic(10)           17          11
## 11: Population 1 ST_Classic(11)           12          10
## 12: Population 1 ST_Classic(12)            1           1
## 13: Population 1 ST_Classic(13)           13          13
## 14: Population 1 ST_Classic(14)           21          14
## 15: Population 1 ST_Classic(15)           19          21
## 16: Population 1 ST_Classic(16)           18          17
## 17: Population 1 ST_Classic(17)            6          19
## 18: Population 1 ST_Classic(18)           22          15
## 19: Population 1 ST_Classic(19)           23          22
## 20: Population 1 ST_Classic(20)           20          18
## 21: Population 1 ST_Classic(21)           14          12
## 22: Population 1 ST_Classic(22)            5          20
## 23: Population 1 ST_Classic(23)           10           4
##       Population       Scenario SSMD_Nall120 SSMD_Het120
```

```
pairw[[6]]
```

```
## $SC
##  [1] "Rank comparison of sensitivity coefficients "
##  [2] "$`Population 1`"
##  [3] "$method"
##  [4] "[1] \"Kendall's coefficient of concordance Wt\""
##  [5] ""
##  [6] "$subjects"
##  [7] "[1] 23"
##  [8] ""
##  [9] "$raters"
## [10] "[1] 2"
## [11] ""
## [12] "$irr.name"
## [13] "[1] \"Wt\""
## [14] ""
## [15] "$value"
## [16] "[1] 0.7238178"
## [17] ""
## [18] "$stat.name"
## [19] "[1] \"Chisq(22)\""
## [20] ""
## [21] "$statistic"
## [22] "[1] 31.84798"
## [23] ""
## [24] "$p.value"
## [25] "[1] 0.08002493"
## [26] ""
## [27] "attr(,\"class\")"
## [28] "[1] \"irrlist\""
## [29] ""
##
## $SSMD
##  [1] "Rank comparison of SSMD "
##  [2] "$`Population 1`"
##  [3] "$method"
##  [4] "[1] \"Kendall's coefficient of concordance Wt\""
##  [5] ""
##  [6] "$subjects"
##  [7] "[1] 23"
##  [8] ""
##  [9] "$raters"
## [10] "[1] 2"
## [11] ""
## [12] "$irr.name"
## [13] "[1] \"Wt\""
## [14] ""
## [15] "$value"
## [16] "[1] 0.7109684"
## [17] ""
## [18] "$stat.name"
## [19] "[1] \"Chisq(22)\""
## [20] ""
## [21] "$statistic"
## [22] "[1] 31.28261"
## [23] ""
## [24] "$p.value"
## [25] "[1] 0.09047599"
## [26] ""
## [27] "attr(,\"class\")"
## [28] "[1] \"irrlist\""
## [29] ""
```

Because multiple parameters are changed one at the time, `vortexR` also calculates automatically the mean effect of each of the modified parameter during the ST run. These results are returned as additional elements of the list and follow the same order as the above. We will only look at the ranks of the parameters with SSMD. To identify in what element these are contained, we can look at their names.

```
names(pairw)
```

```
##  [1] "coef.table"              "SSMD.table"
##  [3] "SSMD.table.pvalues"      "ranks.SC"
##  [5] "ranks.SSMD"              "Kendall"
##  [7] "mean.coef.table"         "mean.SSMD.table"
##  [9] "mean.SSMD.table.pvalues" "ranks.mean.SC"
## [11] "ranks.mean.SSMD"         "Kendall.means"
```

The first three additional elements (elements 7 to 9) refer to the mean value of SC and SSMD (for example, looking at the p-values of the mean SSDM, it is confirmed that the parameters 'Pouch young mortality', 'SD', and 'Mate monopolization' are not significantly affecting the final Nall and Het). The third last element of the list (ranks.mean.SSMD) reports the rank for these parameters based on the mean of the strictly standardised mean difference. We can explore these results as we did above.

```
pairw[[11]]
```

```
##      Population  SV SSMD_Nall120 SSMD_Het120
## 1: Population 1 SV1            1           4
## 2: Population 1 SV2            6           3
## 3: Population 1 SV3            2           1
## 4: Population 1 SV4            5           7
## 5: Population 1 SV5            7           6
## 6: Population 1 SV6            4           5
## 7: Population 1 SV7            3           2
```

Interestingly, the effects of these parameters, in terms of changes in the demography and heterozygosity of the population, are inconsistent (which is also confirmed by the non-significant Kendall's test). For example, note how the Adult Mortality (SV2) has a substantial effect on the final heterozygosity of the population, but not on the demography.

fit_regression

To evaluate the main and interaction effects of some of the parameters used in the simulations, we used the function `fit_regression`. First, we have to prepare the data obtained with the Latin hypercube sampling to be analysed with `fit_regression` with the following:

```r
# Collate all .run
run <- collate_run(project="Pacioni_et_al", scenario="ST_LHS", 1,
dir_in=pac.dir, save2disk=FALSE, verbose=FALSE)

# Remove base scenario from the output in long format
lrun.ST_LHS.no.base <- run[[2]][!run[[2]]$Scenario == "ST_LHS(Base)", ]
```

We also need to prepare a data.frame to match the scenarios with the parameter values used to set the simulations. We do that with `lookup_table` (we have to do this because we are using data collected in the .run Vortex outputs --- N --- and these do not contained the 'SV' variables used to run the simulations. To obtain the latter, we use the .stdat files). To save space, `vortexR` comes with the collated .stdat files in order to reduce the size of the package, so there is no need to run `collate_dat` for the LHS simulations, but we can directly load the data.

```r
# Load the already collated .stdat data
data(pac.lhs)
# Remove base scenario
stdat.ST_LHS.no.base <- pac.lhs[!pac.lhs$scen.name == "ST_LHS(Base)", ]

# Create the lookup table
lkup.ST_LHS <- lookup_table(data=stdat.ST_LHS.no.base,
project="Pacioni_et_al", scenario="ST_LHS", pop="Population 1",
SVs=c("SV1", "SV2", "SV3", "SV4", "SV5", "SV6", "SV7"), save2disk=FALSE)
```

We can graphically inspect the data with the function `m_scatter`.

```r
scatter.plot <- m_scatter(data=stdat.ST_LHS.no.base[1:33], data_type="dat",
                          lookup=lkup.ST_LHS, yr=120, popn=1, param="Nall",
                          vs=c("SV1", "SV2", "SV3"), save2disk=FALSE)
scatter.plot
```

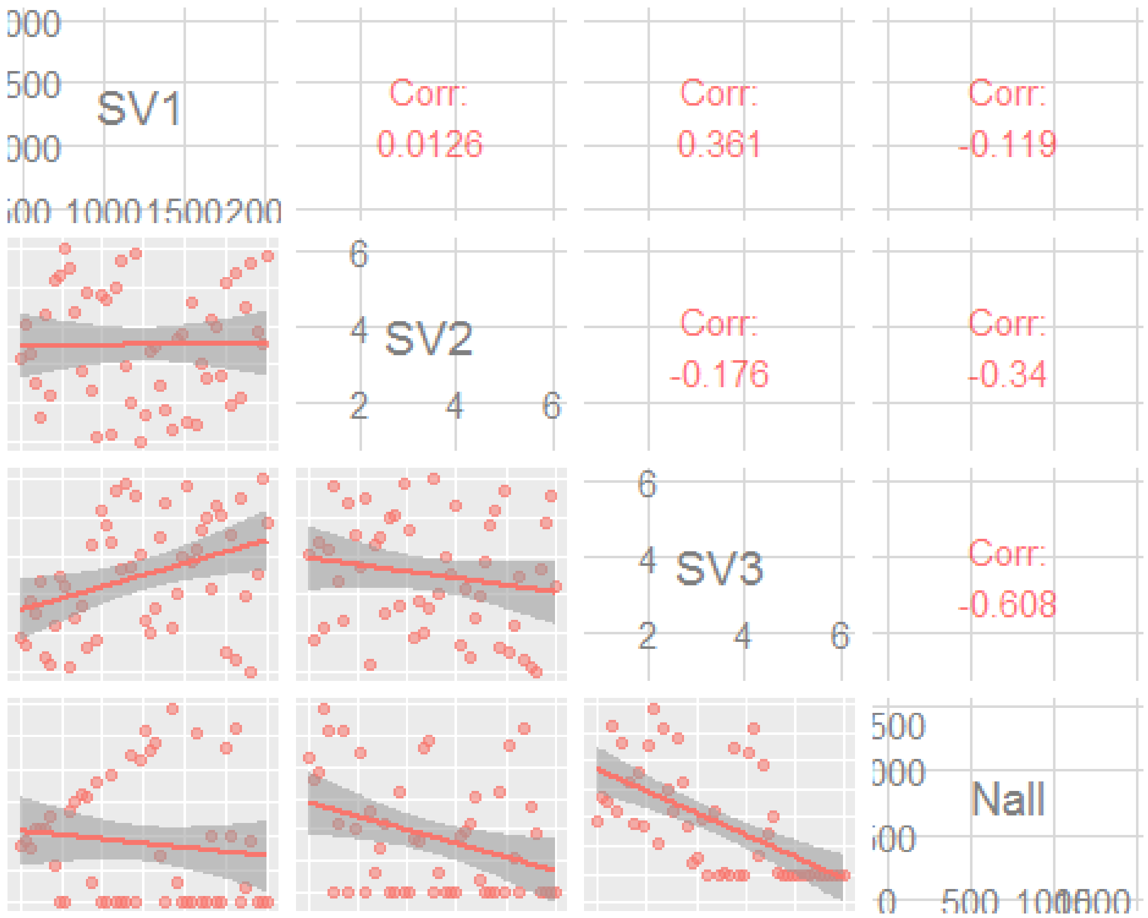


Figure S4. *Scatter plot matrix of pairs of selected variables, named as for Vortex output on the diagonal. Correlation coefficients are reported above the diagonal.*

If we are happy with how the simulations were carried out, we can carry out the analysis. In this example, `fit_regression` will fit a GLM. Just before the beginning of the search for the best model(s), a summary of the dependent variable is printed on the screen so that the user can check whether there is any problem or error. For the same reason, the distribution of the dependent variable is plotted. In this example, it is clear that there are lots of zeros. Indeed, in the original publication the authors used a zero-inflated model, but because here we are simply demonstrating the use of `fit_regression`, we will ignore this.

```
reg <- fit_regression(data=lrun.ST_LHS.no.base, lookup=lkup.ST_LHS, census=F,
                  project="Pacioni_et_al", scenario="ST_LHS", popn=1,
                  param="N", vs=c("SV1", "SV2", "SV3"), l=2,  ncand=30,
                  save2disk=FALSE)

## summary of N

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0     0.0   380.5   441.7   779.8  1544.0

## Fitting a GLM...
## Initialization...
## TASK: Diagnostic of candidate set.
## Sample size: 150
## 0 factor(s).
## 3 covariate(s).
## 0 f exclusion(s).
## 0 c exclusion(s).
## 0 f:f exclusion(s).
## 0 c:c exclusion(s).
## 0 f:c exclusion(s).
## Size constraints: min =  0 max = -1
## Complexity constraints: min =  0 max = -1
## Your candidate set contains 64 models.

## Overdispersion was detected in the data.

## Setting overdispersion parameter to: 160.960271847589

## NOTE: the Information Criterion for model search was changed to QAIC

## confsetsize set to 30

## Search method set to g

## Search for best candidate models using level = 2 started...

## TASK: Genetic algorithm in the candidate set.
## Initialization...
## Algorithm started...
## Improvements in best and average IC have bebingo en below the specified
goals.
## Algorithm is declared to have converged.
## Completed.

## Done! Elapsed time:
```

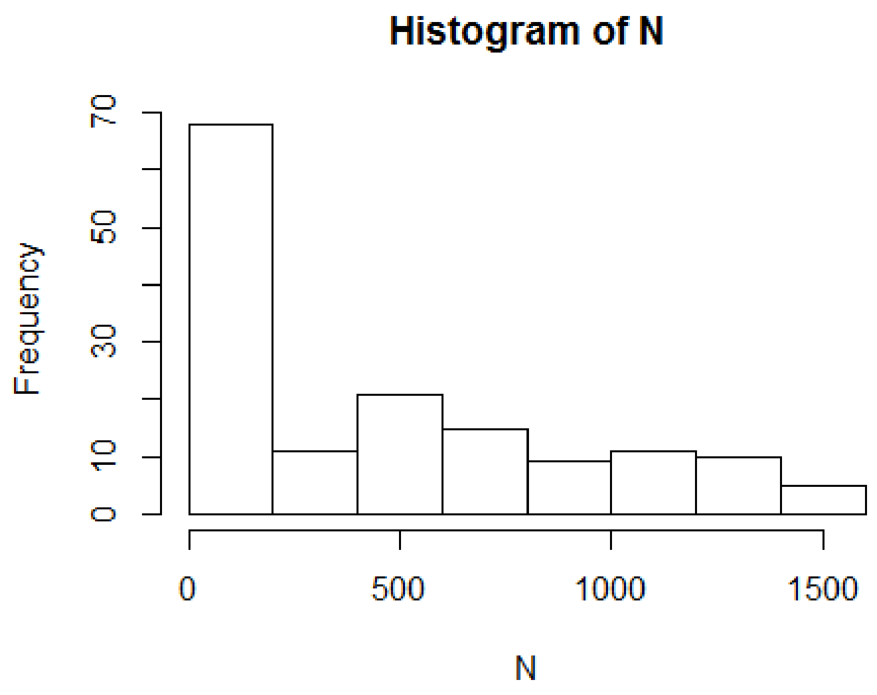


Figure S5. *Bar plot of the distribution of the dependent variable values.*

```
##    user  system elapsed 
##    0.67    0.01    0.58
```

Once the search is concluded, there are several information that can be retrieved. Detailed explanations and examples are in the documentations of the R package `glmulti` and we only report below a few examples.

It is possible to inspect the plot of the information criterion of the evaluated models. This is saved to disk if `save2disk=TRUE`, otherwise it can be called from the `glmulti` object:

```
plot(reg, type="p")
```

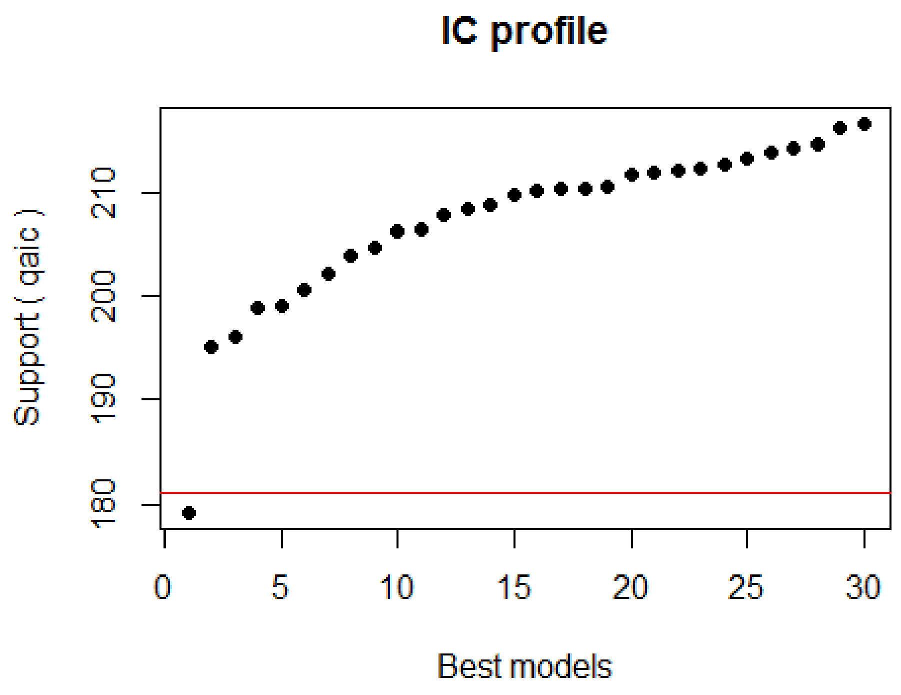


Figure S6. *Profile of the Information Criterion (IC) values (QAIC, y-axis) of the different models that were tested (x-axis).*

In our case, it is clear that one model (the full model) is markedly better that the others. The formula of the best model can be obtained with:

```
reg@formulas[1]

## [[1]]
## N ~ 1 + SV1 + SV2 + SV3 + SV2:SV1 + SV3:SV1 + SV3:SV2
## <environment: 0x00000000174e0b28>
```

Possible additional information the user may be interested into are, for example, the list of the best 5 models retained:

```
reg@formulas[1:5]

## [[1]]
## N ~ 1 + SV1 + SV2 + SV3 + SV2:SV1 + SV3:SV1 + SV3:SV2
## <environment: 0x00000000174e0b28>
##
## [[2]]
## N ~ 1 + SV1 + SV2 + SV2:SV1 + SV3:SV1 + SV3:SV2
## <environment: 0x00000000174e0b28>
##
## [[3]]
## N ~ 1 + SV1 + SV3 + SV2:SV1 + SV3:SV1 + SV3:SV2
## <environment: 0x00000000174e0b28>
##
## [[4]]
## N ~ 1 + SV1 + SV3 + SV3:SV1 + SV3:SV2
## <environment: 0x00000000174e0b28>
##
## [[5]]
## N ~ 1 + SV1 + SV2:SV1 + SV3:SV1 + SV3:SV2
## <environment: 0x00000000174e0b28>
```

The information criterion values:

```
reg@crits
```

```
##  [1] 179.1871 195.0635 196.1317 198.8619 199.0479 200.6671 202.1502
##  [8] 204.0119 204.7767 206.3085 206.5311 207.7941 208.4381 208.8112
## [15] 209.7318 210.1835 210.2785 210.4264 210.5629 211.7618 211.8596
## [22] 212.1689 212.2513 212.7004 213.2967 213.7967 214.2563 214.6940
## [29] 216.1124 216.6124
```

The regression coefficients of the best model:

```
coef(reg@objects[[1]])
```

```
##   (Intercept)           SV1           SV2           SV3       SV1:SV2
##  3.3744446279  0.0038676565  0.6612178819  0.8109509263 -0.0005148349
##       SV1:SV3       SV2:SV3
## -0.0007254374 -0.2078712635
```

The model averaged coefficients across the best 30 models:

```
library(glmulti, quietly=TRUE)
coef.glmulti(reg)
```

```
##                 Estimate Uncond. variance Nb models Importance
## SV3          0.8104692557     2.581781e-04        16  0.9995753
## SV2          0.6608555431     1.562960e-04        16  0.9996768
## SV1:SV2     -0.0005146110     7.445170e-11        16  0.9999082
## SV1:SV3     -0.0007251422     1.277369e-10        15  0.9999924
## SV1          0.0038662260     2.780616e-09        16  0.9999987
## (Intercept)  3.3764680272     4.311802e-03        30  1.0000000
## SV2:SV3     -0.2078325984     6.717802e-06        30  1.0000000
##             +/- (alpha=0.05)
## SV3             3.176133e-02
## SV2             2.471229e-02
## SV1:SV2         1.705597e-05
## SV1:SV3         2.234072e-05
## SV1             1.042340e-04
## (Intercept)     1.297981e-01
## SV2:SV3         5.123334e-03
```

The plot of model averaged importance of terms:

```
plot(reg, type="s")
```

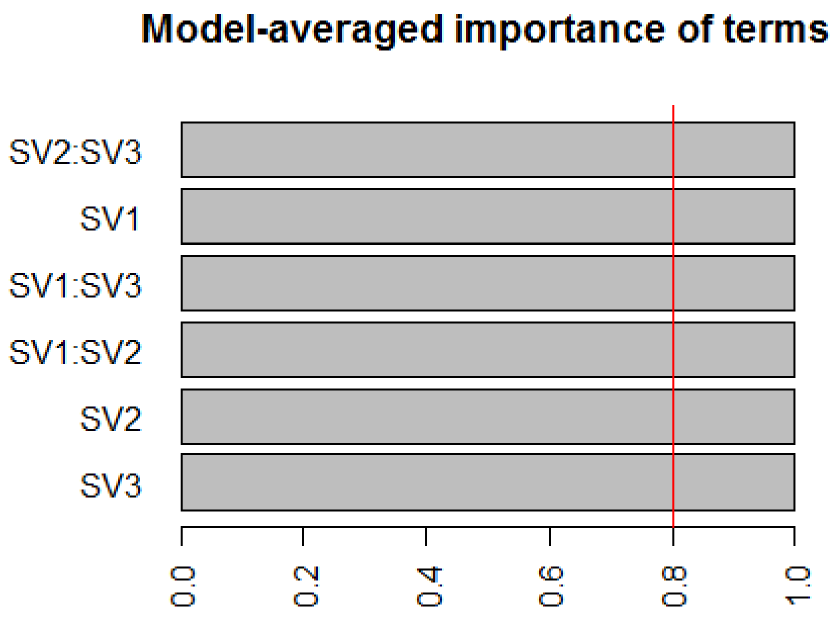


Figure S7. *Model-averaged importance of terms included in the models. Relative variable importance is obtained by summing the weights of the variables from the models where these appear on the sum of weights. The red line indicates the commonly accepted threshold at 0.8. For further information see documentation from the R package gmulti.*

In this case though, all terms have a model-averaged importance of 1.

Conclusions

`vortexR` is a flexible tool to facilitate and speed up data handling and analysis of Population Viability Analysis projects carried out with Vortex.

`vortexR` work-flows make the analysis of Vortex output reproducible. Combining `vortexR` analysis with explanatory prose into live notebooks such as Sweave (Leish 2002) or RMarkdown (Allaire et al 2016) allows creating reproducible research and quickly generating and updating reports and documents. For examples, this Appendix is written in Rmarkdown.

Here, we have briefly described the main functionalities of the package. More examples are available through: `help(package="vortexR")` or in the package's vignette. Report any issues or lodge feature requests at https://github.com/carlopacioni/vortexR/issues.

References

Allaire, J., Cheng, J., Xie, Y., McPherson, J., Chang, W., Allen, J., Wickham, H., Atkins, A. & Hyndman, R. (2016) rmarkdown: Dynamic documents for R.

Leisch, F. (2002) Sweave: Dynamic generation of statistical reports using literate data analysis. Compstat, pp. 575-580. Springer.

Table S1. Comparison between Vortex and vortexR functionality.

Function	Vortex	vortexR
Data handling		
collate_dat	-	√
collate_one_dat	-	√
collate_proc_data	-	√
collate_run	-	√
conv_l_yr	-	√
lookup_table	-†	√
Data visualisation		
dot_plot	-	√
line_plot_year	√	√
line_plot_year_mid	-	√
m_scatter	-	√
Data analysis		
Pairwise	-	√
fit_regression	-	√
Nadults	-	√
Ne	-*	√
Pexinct	√	√
rRec	-	√

† While Vortex does not generate a summary table like lookup, the same information can be extracted manually by other summary tables from Vortex

* Vortex calculates N_e across the whole simulated time, but it does not currently allow to limit the time window to a specific interval