JNCC Report No. 657

A Population Viability Analysis Modelling Tool for Seabird Species – Tool Testing

Report for methodology and results for testing of tool

Butler, A., Searle, K., Mobbs, D.C. & Daunt, F.

September 2020

ISSN 0963 - 8091

For further information please contact:

Joint Nature Conservation Committee
Monkstone House
City Road
Peterborough PE1 1JY
https://jncc.gov.uk/

This document should be cited as:

Butler, A., Searle, K., Mobbs, D.C. & Daunt, F. 2020. A Population Viability Analysis Modelling Tool for Seabird Species – Tool Testing: Report for methodology and results for testing of tool, JNCC Report No. 657, JNCC, Peterborough, ISSN 0963-8091.

JNCC EQA Statement:

This document is compliant with JNCC's Evidence Quality Assurance Policy
https://jncc.gov.uk/about-jncc/corporate-information/evidence-quality-assurance/

Contents

1 Automated Testing of the R package

1.1 Overview

We conducted a range of automated tests to identify whether there are any situations in which errors or inconsistencies arise when running the R package which underpins the PVA Shiny tool. These tests included independent checking of the underlying R code, automated testing of the R package, comparison of R package and Shiny tool outputs, and biological plausibility of outputs. Note that we focus only on testing the components of the R package that are also used within the Shiny tool. Elements that were solely included in the R package to provide futureproofing against possible future developments (inclusion of skipped breeding, and inclusion of correlations between demographic rates) were not included within the testing. Global sensitivity analysis was only subject to very limited testing, since this is the most computationally intensive part of the code.

The focus of this component of the testing process is only on testing the R package (not the Shiny tool), and that it is solely designed to detect bugs (i.e. situations where the R package operates in a way that differs from the way it is intended, and documented, to operate) and potential areas for improvement in the way that code works, rather than to assess biological plausibility. If the process revealed any apparent bugs, the cause of these was investigated; where these were revealed to be genuine bugs, and could be fixed easily, the bugs were fixed and the testing process was re-run.

The R code that was used for automated testing of the R package has now been incorporated into the R package itself, as an additional set of files, to enable the testing to easily be repeated if the R package is updated in future.

We ran three separate batches of automated tests. The results of the automated testing that we report here are for Version 4.12 of the R package (the only differences between Version 4.12 and Version 4.13 relate to the presentation of outputs, and so are not relevant to the testing).

1.2 Batch 1 – Valid Inputs

1.2.1 Methods

The first batch involved running the R package using a large numbers of sets of inputs in which the code should run successfully - i.e. where the inputs have the correct format, dimension and naming, and contain logically valid values. In these situations, testing involves (a) checking that the code does run successfully, and (b) checking that it produces outputs that are of the correct dimension and format and contain logically valid values.

We achieved this by producing an R function that is able to simulate a random set of valid inputs for the tool, for each of the four main ways of running the tool (simulation, validation, local sensitivity analysis, global sensitivity analysis). A detailed description of how random valid inputs are generated is given in Appendix A. We generate a large number of sets of random inputs for each of these four variants (200 for the former two variants, 100 for the latter two variants which are more computationally intensive), to run the tool using each set of inputs, and to check the outputs produced.

1.2.2 Results

In Table 1 we summarize the results of this batch of testing. We regard there as being five “acceptable” outputs:

A. The run completes fully, without an error message.
B. The run only partially completes – e.g. reached a point where calculations are impossible – but produces partial output, and no error message.
C. The run aborts with error message E1:

```
Error in leslie.update(demobase.ests = demobase.ests[j; ; ]; nbyage.prev =
    nbyage.prev; : Population size explosion - will lead to numerical overflow
```

D. The run aborts with error message E2:

```
Error in inits.burned(nbyage.burned = nbyage.burned; inipop.totals =
    inipop.totals): Error! Zero values during burn-in...
```

E. The run completes with error message E3:

```
Error in leslie.update(demobase.ests = demobase.ests[j; ; ]; nbyage.prev =
    nbyage.prev; : Invalid survival/productivity probabilities simulated!
```

These three error messages are regarded as “acceptable” because they represent issues with the output, rather than bugs, and represent situations in which it would currently be impossible for the calculations to continue.

Table 1.

Status	**Number of simulations**			
	“simulation” mode	“validation”	“local sensitivity”	“global sensitivity”
Completed fully without error message	50	82	20	54
Completed partially without error message	37	30	28	
Completed with error message E1	18	9	24	16
Completed with error message E2	37	14	17	13
Completed with error message E3	58	65	11	17
Completed with other error messages	0	0	0	0

In all cases, the code completed either fully or partially, or exited with one of these three error messages. It may seem surprising that the code exits so frequently with error messages, or only completed partially, but it should be remembered that we are running the PVAs with inputs that are logically valid but otherwise random – there is no guarantee that these are biologically plausible, and in most cases they will not be. Leslie matrix models are fairly sensitive to slight variations in the demographic rates, so running PVAs with randomly generated inputs will lead to populations that explode (leading to error message E1) to become extremely large or else rapidly become extinct (leading to error message E2, or partial completion) in a very high proportion of cases.

1.3 Batch 2 – Invalid Inputs

1.3.1 Methods

The second batch of testing involves running a large number of scenarios in which the code should fail. These include situations in which inputs have incorrect format, incorrect dimension or incorrect naming, or inputs have values that are logically invalid (such as negative survival rates). The testing involves (a) checking that the code does indeed crash in these situations, and (b) checking that it produces clear and accurate error messages when it does so.

We conducted these tests by producing an R function that is able to introduce pre-specified errors into a set of valid inputs. We use the function from Batch 1 to generate a set of valid inputs, and then used this function to generate 20 sets of invalid inputs associated with this, by introducing 20 different possible errors. The set of possible errors that we considered is outlined in Appendix B. We repeated this for a number of different sets of valid inputs (e.g. 200 or 100 sets for each mode of running, as in Batch 1).

1.3.2 Results

Detailed results are given in Appendix B. The results suggest that the error messages that are produced do not always enable users to uniquely identify the issue that led the code to fail; in future versions of the R package we recommend improving the utility of the error messages.

Under scenarios 2-7, 11-17 and 19-20 the code always reported an error message. Under scenarios 1, 8, 9, 10 and 18 the code failed to report an error message in 26%, 64%, 8%, 39% and 32% of simulations, respectively. The explanation for this is that four of the underlying errors (A, H, I and J) did not always yield an error message. In the context of error A, this is because the value of “mbs” is not always used in the calculations (e.g. if “model.prodmax = FALSE”), so it is legitimate that the failure to provide this will not lead to an error in these situations. In the context of error H, negative values of “nburn” are interpreted by the code as corresponding to “no burn in” (i.e. nburn = 0); it future versions of the code it may be worth explicitly introducing an error message if users specify negative burn-in periods. In the context of error I, these appear to be situations where the inputs were actually valid (since the final value of “demobase.survadult” can be negative in models with density dependence), so it is correct that no error message is produced. In the context of error J, it appears that if the size of “npop” is larger than that required for the calculations this does not lead to an error message; whilst this is not a bug as such (the additional values are simply ignored), it may be desirable to introduce an error message in this situation, as the fact users have specified an input of incorrect size suggests they have misunderstood the format of the input, and so may have structured the inputs incorrectly.

1.4 Batch 3 – Consistency of Inputs

1.4.1 Methods

The final batch involves checking the internal consistency of outputs generated by the R package. This involves checking that the same results are obtained by specifying identical inputs to the tool, but in different ways, and is the most sophisticated and complex part of the automated testing – it is designed to detect general bugs in the code, and also (for models that include stochasticity) to check whether matching works correctly when inputs are specified in different format (i.e. that identical results are obtained for PVAs with the same format and specification, but generated using inputs in different formats).

We checked whether:

a. the same results are obtained by running a deterministic PVA or by running a stochastic PVA with zero stochasticity;
b. the same results are obtained by running a density independent model or by running a density dependent model with zero magnitude of density dependence;
c. the same results are obtained by specifying baseline demographic rates as being common to subpopulations or separate for different subpopulations, in situations where the rates used are actually the same for all subpopulations;
d. the same results are obtained by specifying baseline demographic rates as being common to immatures and adults, or separate for immatures and adults, in situations where the rates used are actually the same for immatures and adults;
e. the same results are obtained by specifying impacts as being common to subpopulations or separate for different subpopulations, in situations where the impacts are actually the same for all subpopulations;
f. the same results are obtained by specifying impacts as being common to immatures and adults, or separate for immatures and adults, in situations where impacts are actually the same for immatures and adults;
g. the same results are obtained by running a model without uncertainty in impacts, or by running a model with uncertainty, in a situation where the standard errors are actually equal to zero

We achieved this by repeatedly simulating a relevant set of valid inputs, specified in a specific way (Appendix C), and then creating $128 = 2^7$ sets of inputs that are all designed to represent the same model, but specify it in a different way for each of these 7 decisions.

We focused here solely upon the “simulation” mode for running the tool, as these aspects of model specification either act in the same way for other modes of running the tool or are not relevant for those models (e.g. “impacts” are not included when running in validation mode).

We compared the estimated final population sizes (from individual simulation runs) generated by using the 128 different ways of specifying the same set of inputs. We do this using a range of different numbers of simulations (sim.n) – if there are genuine discrepancies between the different specifications these will persist even as the number of simulations becomes large, but if the discrepancies arise solely from a failure to match random seeds when specifying the model in different ways (which is a known limitation of the tool) these discrepancies will reduce towards zero as the number of simulations becomes large.

We used a set of 20 random initial sets of inputs which will give a total of 20 * 128 = 2560 sets of inputs to consider – but, for computational reasons, we only chose to use a small number of simulations (10) within each input set.

1.4.2 Results

We found that *identical* results are (for all combinations, and all sets of simulations) always obtained:

- by running a density independent model or by running a density dependent model with zero magnitude of density dependence (b);
- by specifying baseline demographic rates as being common to subpopulations or separate for different subpopulations, in situations where the rates used are actually the same for all subpopulations (c);
- by specifying baseline demographic rates as being common to immatures and adults, or separate for immatures and adults, in situations where the rates used are actually the same for immatures and adults (d).

The results suggest, however, that different projections can be obtained in at least some situations when:

- by running a deterministic PVA or by running a stochastic PVA with zero stochasticity (a);
- by specifying impacts as being common to subpopulations or separate for different subpopulations, in situations where the impacts are actually the same for all subpopulations (e);
- by running a model without uncertainty in impacts, or by running a model with uncertainty, in a situation where the standard errors are actually equal to zero (f);
- by specifying impacts as being common to immatures and adults, or separate for immatures and adults, in situations where impacts are actually the same for immatures and adults (g).

These results are to be expected: we know that the matching procedure that we use will not always be effective in matching across different formats for specifying the impacts. The options where matching sometimes or always fails (a, e, f, g) are the options where the results of the option determines whether a stochastic simulation is generated or not, whereas the options where matching works (b, c, d) are those that solely relate to the way that non-stochastic (deterministic) calculations operate.

The numeric differences between runs can be large, but this may solely be due to lack of matching and to the use of a very small number of simulations, so it would be useful in future to check if the differences persist when using a much larger number of simulations.

1.5 Conclusions

We use a table to summarise the key findings of the automated testing, and to make recommendations that link to each of these.

Table 2.

Batch	Conclusion	Recommendations
Batch 1	When run with random but syntactically valid inputs, the package almost always either runs without error, or else crashes with one of three “acceptable” error messages (e.g. error messages that relate to situations in which the current version of the code is designed to fail). The three “acceptable” error messages correspond to situations in which (E1) zero abundance is obtained during the burn-in period, (E2) the simulated population size explodes to be so large that numerical instability is likely to become an issue (and so large that it is entirely implausible), and (E3) invalid survival or productivity values are simulated.	Suggested improvements: Error message E1: in this situation it should be possible to modify the code to produce partial output Error message E3: this error could be detected at the point that users specify inputs, because it appears to essentially arise from the specification of input values for which calculations are not actually possible.
Batch 2	The error messages that are produced when the code fails often do not easily allow a user to trace the underlying source of the error (in terms of the specification of invalid inputs)	Suggested improvement: The error messages should be improved, so that users can more

		easily identify situations in which errors have arisen due to specification of invalid inputs
Batch 3	Matching fails in situations where: the user changes the form of environmental stochasticity; the user changes whether impacts are split between subpopulations/ages or not; or the user changes whether standard errors are specified or not.	Suggested improvement: Amend the code so that matching continues to work in these situations – this is a fairly substantial piece of work, however, as it requires a restructuring of the existing code.

2 Review of R code

2.1 Summary

UKCEH (Kate Searle) spent approximately three days reviewing the R code for the R package, firstly with BioSS (Adam Butler), and then as an independent review. No major issues were discovered as a result of this review, and no additional bugs were identified, other than those identified during the automated testing described above.

3 Comparison of output between R package and Shiny tool

3.1 Summary

We ran a suite of tests to generate a range of PVA models and outputs within the R package, and similarly within the Shiny tool. We tested a range of PVA models within each of the three Shiny modes – Simulation, Validation and Sensitivity. These tests revealed a number of bugs, all to do with default settings within Shiny that needed to be updated to match those within the R package. After these bugs were updated, the Shiny tool produced almost identical output to the R package under all three modes of use. Very minor discrepancies between the two sets of output (Shiny versus R) were expected due to differences in seed matching arising from the automated generation of inputs within the methods used to generate the R outputs in the testing.

4 Biological plausibility of Shiny outputs

4.1 Overview

To check the biological plausibility of the tool output, we ran four case studies using data from the Forth Tay (Table 3). In all four cases, we ran a 'historical' PVA where we projected each population forwards to the current day using known population sizes from the 1980s. We then generated a future PVA with three impact scenarios of 0.01, 0.025 and 0.05 additional adult mortality.

Table 3. Baseline demographic rates and initial population sizes used in running the Population Viability Analysis (PVA) for each population (combination of species and SPA). Source: age at first breeding: Porter and Coulson (1987), Cam *et al*. (2002), Harris and Wanless (2011), Lahoz-Monfort *et al*. (2013), Harris *et al*. (2016); max brood size: CEH unpublished data; adult and immature survival: Coulson and White (1959), Breton *et al.* (2006), Harris *et al.* (2007), Lavers *et al*. (2008), Harris and Wanless (2011); Jitlal *et al*. (2017); breeding success: Newell *et al.* (2016); initial population size: http://archive.jncc.gov.uk/smp/.

Species	Age at first breeding	Max brood size	Adult survival (mean,SD)	SPA	Breeding success (mean,SD)	Immature survival (mean, SE)	Initial population size (year)
Kittiwake	4	2	0.857 (0.067)	Forth Islands	0.55 (0.35)	0.697 (0.054)	4206 (2019)
Guillemot	6	1	0.926 (0.044)	Forth Islands	0.725 (0.108)	0.796 (0.012)	21812 (2019)
Razorbill	5	1	0.909 (0.057)	Forth Islands	0.63 (0.078)	0.838 (0.021)	4855 (2019))
Puffin	5	1	0.906 (0.059)	Forth Islands	0.691 (0.133)	0.921 (0.019)	49210 (2017)

4.2 Results and Conclusions

The results for black-legged kittiwakes, common guillemots, razorbills and Atlantic puffins are presented in Figures 1-4 respectively. The results accord closely with recent population modelling of these species at this SPA (Freeman *et al*. 2014; Jitlal *et al.* 2017), both with respect to retrospective predictions of past population change and forecasting of future population change. Specifically, the results accord with past work showing that kittiwakes have shown a marked decline since the 1980s, which is predicted to continue in the future. Similarly, as shown in past work, guillemots and razorbills have shown a moderate increase since the 1980s and, therefore, are predicted to increase further in future. Finally, again as shown in past work, puffins have shown a marked exponential increase since the 1980s that is predicted to continue in future resulting in very large predicted population sizes at the end of the forecast period. Population counts of puffins are carried out approximately every five years, in contrast to the other species that are counted annually. As a result, there have only been nine counts since 1984, and this limited data together with the sharp past increase results in erratic future predictions that are markedly in excess of previous observations and of doubtful realism (see Freeman *et al.* 2014, for a full discussion on this).

The three impact scenarios imposed expected changes in future predictions, with the largest divergence from the baseline in the scenario of 0.05 additional adult mortality, smallest divergence in the 0.01 scenario and intermediate divergence for the 0.025 scenario.

In conclusion, we consider that the models and scenarios produced biologically plausible results for kittiwakes, guillemots and razorbills. For puffins, the lack of realism was a result of the limited availability of count data coupled with the exponential past increase in population size, a set of circumstances that has proved challenging for multiple population modelling exercises to date, and therefore does not reflect the methods employed or, in this case, the performance of the Shiny tool.

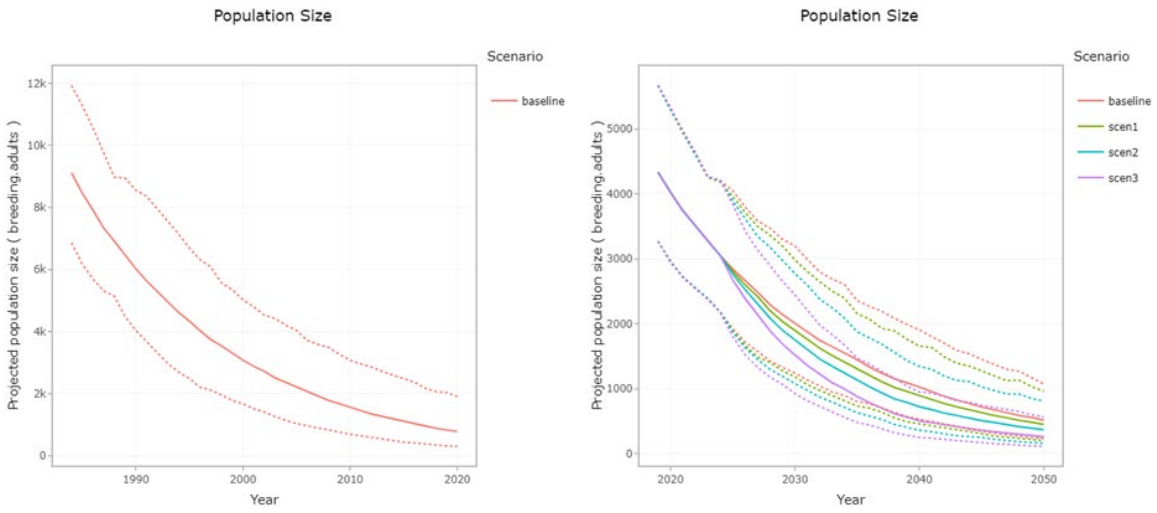

Figure 1. Historical population projection (left panel) and future PVA (right panel) for black-legged kittiwakes breeding at the Forth Islands SPA. Impact scenarios for future PVA are scen1: 0.01 reduction in adult survival; scen2: 0.025 reduction in adult survival; scen3: 0.05 reduction in adult survival.

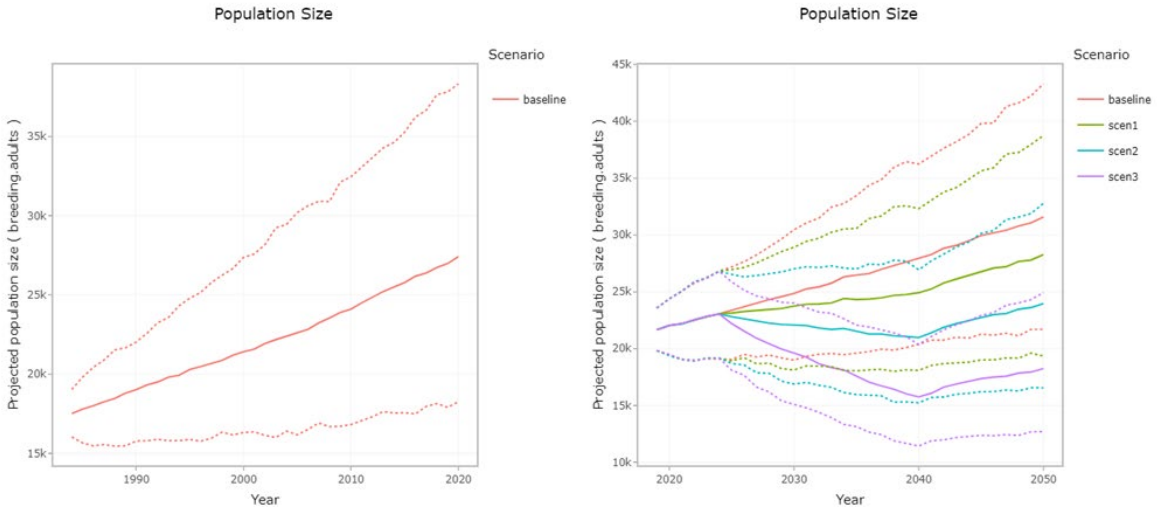

Figure 2. Historical population projection (left panel) and future PVA (right panel) for common guillemots breeding at the Forth Islands SPA. Impact scenarios for future PVA are scen1: 0.01 reduction in adult survival; scen2: 0.025 reduction in adult survival; scen3: 0.05 reduction in adult survival.

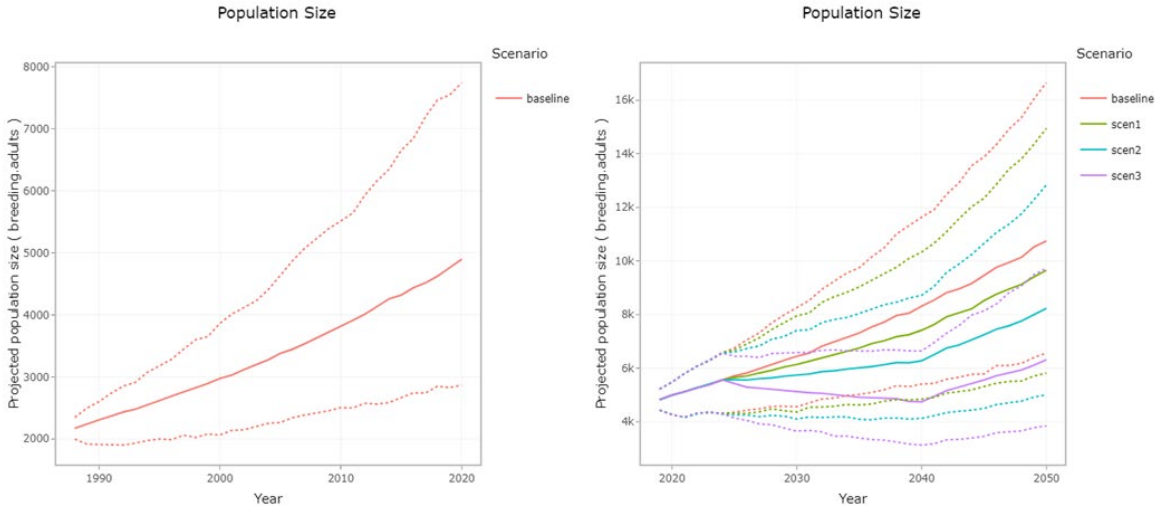

Figure 3. Historical population projection (left panel) and future PVA (right panel) for razorbills breeding at the Forth Islands SPA. Impact scenarios for future PVA are scen1: 0.01 reduction in adult survival; scen2: 0.025 reduction in adult survival; scen3: 0.05 reduction in adult survival.

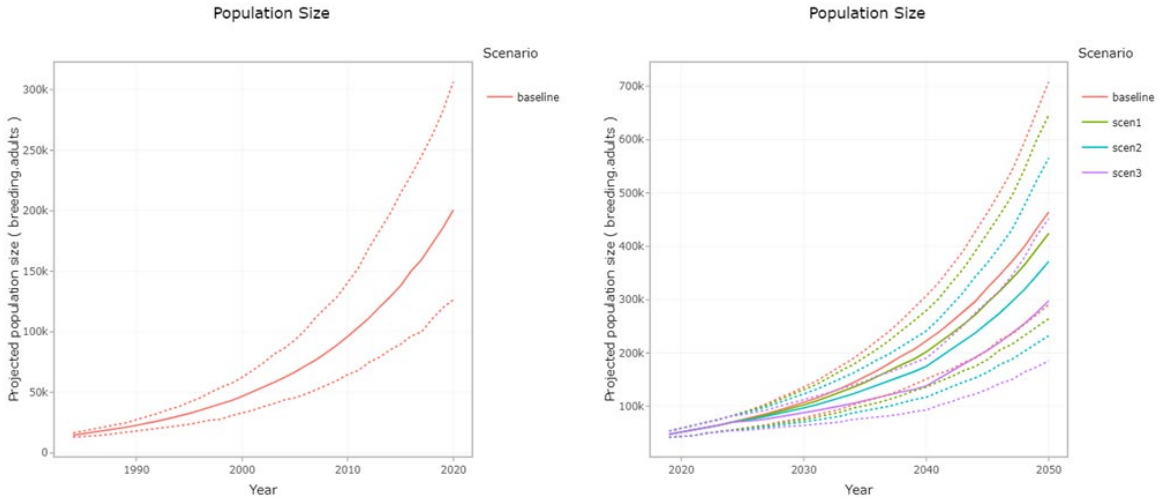

Figure 4. Historical population projection (left panel) and future PVA (right panel) for Atlantic puffins breeding at the Forth Islands SPA. Impact scenarios for future PVA are scen1: 0.01 reduction in adult survival; scen2: 0.025 reduction in adult survival; scen3: 0.05 reduction in adult survival.

5 References

Breton, A.R., Diamond, A.W. & Kress, S.W. (2006). Encounter, survival, and movement probabilities from an Atlantic puffin *Fratercula arctica* metapopulation. Ecological Monographs, 76, 133–149.

Cam, E., Cadiou, B., Hines, J.E. & Monnat, J.-Y. (2002) Influence of behavioural tactics on recruitment and reproductive trajectory in the kittiwake. Journal of Applied Statistics 29: 163–185.

Coulson, J.C. & White, E. 1959. The post-fledging mortality of the kittiwake. Bird Study, 6, 97–102.

Freeman, S., Searle, K., Bogdanova, M., Wanless, S. & Daunt, F. (2014) Population dynamics of Forth & Tay breeding seabirds: review of available models and modelling of key breeding populations (MSQ – 0006). Contract report to Marine Scotland Science http://www.gov.scot/Resource/0044/00449072.pdf

Harris, M.P. & Wanless, S. (2011) The Puffin. T & AD Poyser, London.

Harris, M.P., Albon, S.A. & Wanless, S. (2016) Age-related effects on breeding phenology and success of Common Guillemots *Uria aalge* at a North Sea colony. Bird Study 63:565-565.

Harris, M.P. Frederiksen, M. & Wanless, S. 2007. Within- and between-year variation in the juvenile survival of common guillemots *Uria aalge*. Ibis, 149, 472–481.

Jitlal, M., Burthe, S. Freeman, S. & Daunt, F. (2017) Testing and validating metrics of change produced by Population Viability Analysis (PVA) (Ref CR/2014/16). Scottish Marine and Freshwater Science Vol 8 No 23 https://data.marine.gov.scot/sites/default/files//SMFS%200823.pdf

Lahoz-Monfort, J.J., Morgan, B.J.T., Harris, M.P., Daunt, F., Wanless, S. & Freeman, S.N. (2013) Breeding together: modelling synchrony in productivity in a seabird community. Ecology, 94, 1, 3-10.

Lavers, J.L., Jones, I.L. & Diamond, A.W. 2007. Natal and Breeding Dispersal of razorbills *Alca torda* in Eastern North America. Waterbirds, 30, 588–594.

Newell, M., Harris, M., Wanless, S., Burthe, S., Bogdanova, M., Gunn, C. & Daunt, F. (2016). The Isle of May long-term study (IMLOTS) seabird annual breeding success 1982-2016. NERC Environmental Information Data Centre. https://doi.org/10.5285/02c98a4f-8e20-4c48-8167-1cd5044c4afe

Porter, J.M. & Coulson, J.C. (1987) Long-term changes in recruitment to the breeding group, and the quality of recruits at a kittiwake *Rissa tridactyla* colony. Journal of Animal Ecology 56: 675–689.

6 Acknowledgements

We thank Sue O’Brien and Mel Kershaw for their guidance and support throughout this project.

7 Appendix A: Simulating a random set of valid inputs

The following table outlines the approach taken to simulate a random set of valid inputs for the PVA tool. The "when" column indicates which run types this input is provided for: blank indicates all run types ("simplescenario", "validation", "sensitivity.local", "sensitivity.global"). "MV" indicates "simplescenario" and "validation" only, "M" indicates "simplescenario" only, "MS" indicates "simplescenario", "sensitivity.local" or "sensitivity.global", "V" indicates validation only, "S" indicates "sensitivity.local" or "sensitivity.global".

Table 4.

Input	When	How simulated
Inputs relating to model structure		
model.envstoch		Random select "betagamma" or "deterministic"
model.demostoch		Randomly select TRUE or FALSE
model.dd	MV	Randomly select "nodd" or "dduloglin"
model.prodmax		Randomly select TRUE or FALSE
mbs		Randomly select 1, 2, 3 or 4
afb		Randomly select 1, 2, 3, 4, 5 or 6
npop		Randomly select 1, 2, 3, 4 or 5
nscen		Randomly select 1, 2 or 3
nburn		Randomly select a whole number between 0 and 10
sim.n		Randomly select a whole number between 1 and 10
sim.seed		Randomly select a whole number between 1 and 10000
Inputs relating to baseline demography		
demobase.specify.as.params	MV	Fix to be FALSE
demobase.splitpops	M	Randomly select TRUE or FALSE
demobase.splitimmat	MV	Randomly select TRUE or FALSE
demobase.prod		To get mean values: select independent random numbers simulate uniformly between 0.2 and 0.9, and multiply by "mbs". To get SEs: multiply mean values by independent random numbers simulated uniformly between 0.01 and 0.2. For get density dependence magnitude (if model.dd = "dduloglin"): simulate independently from N(0,0.1) distributions
demobase.survadult		As for demobase.prod, but without multiplying by "mbs"
demobase.survimmat	MV	As for demobase.prod, but without multiplying by "mbs"
Initial population size(s)		
inipop.years		Sample a random whole number between 2000 and 2015 for each subpopulation
inipop.inputformat	MV	Randomly select "breeding adults", "breeding pairs" or "all individuals"
inipop.counts		For each subpopulation take 10 to the power of a random number simulated to lie between 1 and 4

Impacts		
impacts.relative	MS	Randomly select TRUE or FALSE
impacts.splitpops	M	Randomly select TRUE or FALSE
impacts.splitimmat	M	Randomly select TRUE or FALSE
impacts.provideses	M	Randomly select TRUE or FALSE
impacts.year.start	MS	Randomly select a whole number between max(inipop.years)+1 and max(inipop.years)+20
impacts.year.end	MS	Randomly select a whole number between impacts.year.start+1 and impacts.year.start+30
impacts.scennames	M	Fix to be “scen1”, “scen2”, *etc*.
imacts.matchscens	M	Randomly select TRUE or FALSE
impacts.prod.mean	MS	Randomly uniform numbers between 0.2 and 0.9
impacts.survadult.mean	MS	Randomly uniform numbers between 0.5 and 0.99
impacts.survimmat.mean	MS	Randomly uniform numbers between 0.5 and 0.99
impacts.prod.se	M	“impacts.prod.mean” multiplied by a random number between 0.1 and 0.5
impacts.survadult.se	M	“impacts.survadult.mean” multiplied by a random number between 0.1 and 0.5
impacts.survimmat.se	M	“impacts.survimmat.mean” multiplied by a random number between 0.1 and 0.5
Outputs		
output.agetype	MV	Randomly select “breeding.adults”, “breeding.pairs” or “ages.separately”
output.year.end		Randomly select a whole number between max(inipop.years)+30 and max(inipop.years)+60
output.year.start	MV	Randomly select a whole number between output.year.end – 30 and output.year.end – 20
output.popsize.target		Take 10 to the power of a random number simulated to lie between 1 and 4
output.popsize.qe		Take 10 to the power of a random number simulated to lie between 0 and 1.5
output.validation.years	V	See text
output.validation.counts	V	See text
Sensitivity analysis		
sens.pcr	S	Simulate three uniform random numbers between 0 and 50
sens.npvlocal or sens.npvglobal	S	Simulate a random whole number between 1 and 10

8 Appendix B: Simulating a random set of invalid inputs

We simulated a set of invalid inputs by simulating a set of valid inputs. We then introduced one or more errors into these inputs. Twenty possible "error scenarios" were considered, which involved introducing one or more of fifteen possible errors (A, B, C, D, E, F, G, H, I, J, K, L, M, N, O).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	X														
2		X													
3			X												
4				X											
5					X										
6						X									
7							X								
8								X							
9									X						
10										X					
11											X				
12												X			
13													X		
14														X	
15															X
16				X								X			
17									X				X		
18	X							X							
19	X					X		X					X		
20	X	X		X			X	X	X	X	X			X	

It can be seen that the first fifteen error scenarios involve introducing each of these fifteen errors in turn. The final five error scenarios involve introducing 2, 2, 2, 4 or 9 errors.

The individual errors are as follows:

Table 5.

Error	Description
A	Name of "mbs" is wrongly written as "mns"
B	"demobase.survadult" is given an incorrect dimension
C	"model.envstoch" is given as "bob"
D	"model.demostoch" is given as "fred"
E	"model.envstoch" is given as FALSE
F	"model.envstoch" is given as 3
G	"model.envstoch" is given as 7.4467
H	"nburn" is given as -1
I	"demobase.survadult" contains negative values
J	"npop" is inconsistent with the dimension of demographic rates & inputs
K	"output.year.end" is equal to min(inipop.years) – 1
L	The first entry in inipop.vals is missing (NA)

M	The first entry is inipop.vals in “NaN” (“Not-a-number”)
N	The first entry in inipop.vals is infinite
O	The length of “inipop.years” does not match that of “inipop.vals”

The choice of specific errors and construction of the scenarios is somewhat arbitrary but is designed to capture a broad range of potential errors, and combination of errors, that could conceivably occur. Note that these errors and error scenarios are designed to produce *invalid* inputs, not *implausible* inputs – i.e. the aim is to generate a broad suite of possible input sets under which the tool should definitely crash and produce an error message.

8.1 Detailed results

The following table lists all of the error messages that were encountered in this stage of testing, aside from the three “allowable” errors, and records the scenario(s) in which they occur. It also lists the underlying errors that could have caused the error message (as they are common to all scenarios).

Table 6.

Error message	Scenarios	Possible errors
"Error in !model.demostoch: invalid argument type",	4, 16	D
"Error in demomods$es[[zi]]: attempt to select less than one element in get1index"	3, 5, 6, 7,19,20	CEFG
"Error in inputs$demobase.prod[i;]: subscript out of bounds"	10	J
"Error in demobase.vals[i; nd;] <- as.numeric(c(inputs$demobase.survadult)): number of items to replace is not a multiple of replacement length"	2	B
"Error in demobase.vals[i; j + 1;] <- as.numeric(c(inputs$demobase.survadult)): number of items to replace is not a multiple of replacement length"	2	B
"Error in ru.base[j; tt; ; ; drop = FALSE]: subscript out of bounds",	11	K
"Error in ntot[; ic; ; j; drop = FALSE]: subscript out of bounds"	11	K
"Error in apply(ntots; 1; getsumy): dim(X) must have a positive length"	11	K
"Error in '[<-'('*tmp*'; i; j; inipop.relyears[j];"	11	K
"Error in nbyage[i; j; inipop.relyears[j]; ; a] <- inipop.counts[a; ; j]: NAs are not allowed in subscripted assignments"	10, 15	JO
"Error in eigen(mmat): infinite or missing values in 'x'"	10,12,13,14,16,17	JLMN
"Error in if (max(frot) > 1e+08) {: missing value where TRUE/FALSE needed"	12, 13, 14, 17	LMN
"Error in if (any(nprev < 0)) {: missing value where TRUE/FALSE needed"	12, 13, 14, 16, 17	
"Error in year.first:output.year.end: result would be too long a vector"	15	O

"Error in if (tt > inipop.relyears[j]) {: missing value where TRUE/FALSE needed"	10,15	O
"Error in out[1] <- mbs * fn.pred.con(pars = as.numeric(c(ests[1;])); : replacement has length zero"	1, 18	A
"Error in out[ids; idp; 1] <- prod: number of items to replace is not a multiple of replacement length"	10	J
"Error in ests[1; 1:2] <- fn.mom.con(c(prod.mn/mbs; prod.sd/mbs)): replacement has length zero"	1, 18	A
"….replacement has 11 rows; data has 2"	2, 20	B
"Error in Ops.data.frame(standard.vals; (1 + (pcchange/100))): '*' only defined for equally-sized data frames"	2, 20	B

9 Appendix C: Generating inputs to use for testing internal consistency

In order to test internal consistency, a random set of valid inputs, with a particular format, were simulated. 127 alternative ways of representing the same inputs were then constructed.

The random initial set of valid inputs was constructed in the same way as in Appendix A, except that:

1. The values of “npop” and “nscen” are simulated to be whole numbers between 2 and 5 and between 2 and 3 respectively (whereas in Appendix A they were simulated to be between 1 and 5 and between 1 and 3 respectively);

2. “model.envstoch” is fixed to be “betagamma”, “model.dd”is fixed to be “ddluoglin”, and “demobase.splitpops”, “demobase.splitimmat”, “impacts.splitpops”, “impacts.splitimmat” and “impacts.provideses” are all specified to be TRUE;

3. the values of “demobase.prod”, “demobase.survadult”, “demobase.survimmat”, “impacts.prod.mean”, “impact.survadult.mean”, “impacts.survimmat.mean”, “impacts.prod.se”, “impact.survadult.se”, “impacts.survimmat.se” are selected in such a way that the seven arguments just mentioned (“model.envstoch”, *etc*.) could equally have been specified in the opposite way. So, for example, the standard deviations for environmental stochasticity are fixed to be zero, so that it should be equivalent to either use “model.envstoch = betagamma” or “model.envstoch = deterministic”.

The remaining 127 alternative specifications of these inputs were formed by swapping some or the seven options being considered (“model.envstoch”, *etc*.) to have the opposite value, and then amending the dimensions of the “demobase.” and “impacts.” inputs accordingly.