



User-defined models

In the previous 2 chapters, we have seen how we can use SURGE to test a variety of different models, using different choices from the model selection menus, and how to constrain these models using VAR and PAR files.

- In this chapter, we will introduce user-defined models. As we have frequently noted in earlier chapters, the menu choices presented by SURGE are put there merely as a convenience - a tool to save you a bit of time in fitting some of the more common models.
- However, all of the models that SURGE presents in the model selection menus, plus a VERY large number more, can be created using user-defined models. You may recall that choice “4” on the model selection menu allows you to apply a user-defined model.
- In fact, if you’ve been wondering why we have placed so much emphasis on the connection between the parameter structure of our models, and the way that SURGE represents them, you’ll now get your answer.
- In simplest terms, **a user-defined model is simply a model created to specify the parameter indexing you want SURGE to use.**
- For example, for a standard CJS model applied to a 5 occasion study, with time variation in both survival and recapture, the user defined models for survival and recapture (respectively) would be

4				
1	2	3	4	
	2	3	4	
		3	4	
			4	

4				
5	6	7	8	
	6	7	8	
		7	8	
			8	

- The bottom parts of these 2 matrices should look familiar. And, if you think back to VAR and PAR files, you might be able to guess the meaning of the number in the top row (4) - yes, it is the number of “new” parameters in the model (4 survival parameters, and 4 recapture parameters). We’ll discuss what we mean by “new” parameters later on.
- So, to create a user-defined model (MOD - .MOD is a useful extension for these files), all you need to do is take your editor, and create the necessary matrices! That’s it! You start by outlining your model structure, then convert it into SURGE parameters (i.e., using the indexing scheme SURGE uses), and then simply build the appropriate files with your editor.
- A few examples will make this clear.

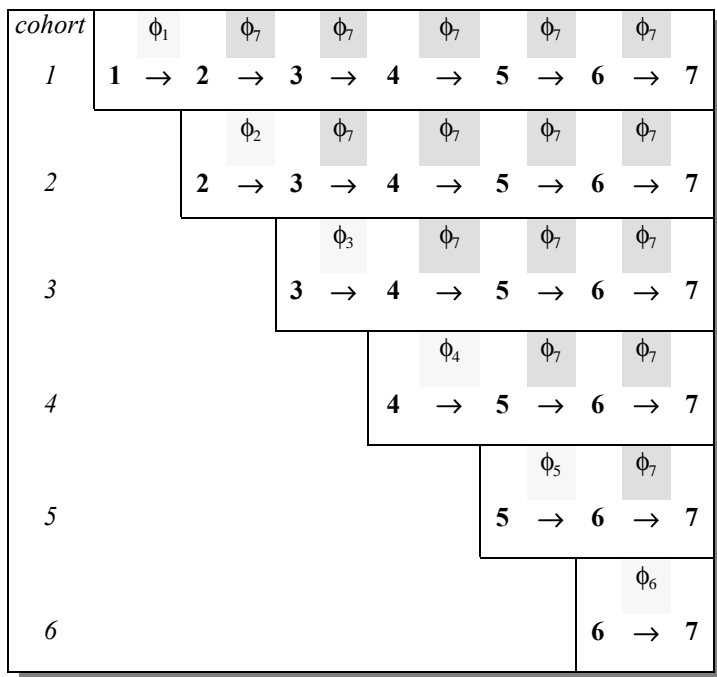
2 age class survival - 1 time-dependent, 1 constant

- Suppose we want to try the following model: two age classes for survival, with time-dependence in the first age class, but constant survival in the second age-class.
- As you’ll recall from the last chapter, this is not a model you can build from the model selection menus: choice “6” let’s you build a 2 age-class model, but with time-dependence in both age classes. Choice “1” also lets you build a 2 age-class model, but with constant survival in both age classes.
- In this case, we want an intermediate model: time-dependence in one age class, but not the other. Let’s assume that recapture rate is simply time-dependent.
- How do we construct the MOD file for this analysis? To demonstrate how, we’ll re-analyze the AGE.SUR data set we first looked at in Chapter 7.
- In that first analysis, we saw that the fit of a 2 age-class time-dependent model was significantly better than the fit of the CJS time-dependent model. Further, the 2 age-class model with time-dependence in both survival rates fit significantly better than the 2 age-class model with constant survival rate in both age classes.

- Now, we will test the intermediate model: time-dependence in the first age class, constant survival in the second (older) age class.

Step 1 - write out the parameter structure

- The first thing to do is write out the parameter structure for this model. Let's say we have 7 occasions. Let's also assume that the first age-class in the survival model spans one interval, while the second age-class spans the remaining intervals. You may recall that this is analogous to the structure specified by choice "6" on the model specification menu, except that we have constant survival in the second age class (choice 6 specifies full time-dependence in both age classes).
- Here is the structure for the survival model:



- SURGE would translate this into the following parameter matrix:

1	7	7	7	7	7
	2	7	7	7	7
		3	7	7	7
			4	7	7
				5	7
					6

- For recaptures, we'll simply assume time-dependence.

Step 2 - create the user-defined model matrix

- The user-defined model we want to fit is: 2 age-classes in survival, time-dependence in the first age class, constant survival in the second age class (as shown in the preceding diagrams). So, all we do is take the parameter matrix we just wrote out (above), and using an editor, create a file which contains this matrix, with an additional row on top showing the number of "new" parameters in this model.
- The only thing we need to remember is that we must add an additional row on top of the matrix which has the number of "new" parameters in the model. In this, the number is "7".
- Let's call the file which will contain the model AGE2.MOD. Here is what it will look like:

simply take this matrix, and add a row with the number of “new” parameters (we’ll talk about what we mean by “new” in a minute). In this case, we would add the number “7” to the top row.

7					
7	13	13	13	13	13
	8	13	13	13	13
		9	13	13	13
			10	13	13
				11	13
					12

- However, the “trick” involves put a “-” sign in front of the “7”. In other words, making the “7” negative. By putting “-7” on the top row, you are “telling” SURGE that you want the following model to start indexing *starting from the first available index value*.
- Now, why is this useful? Well, the negative value on the top row (“-7”) forces SURGE to start the indexing of the matrix elements from where the survival elements (which were previously specified) left off.
- In other words, you don’t have to keep track of the indexing per se - you simply construct the model you want, and put a “-” sign in front of the number of “new” parameters. This allows you to apply the model to either the survival or recaptures very quickly and easily, since SURGE will increment the index values automatically, depending upon whether or not the model is being applied to recaptures or survival.
- A simple example will show how this trick works. Let’s take our user-defined model (2 age-classes, time-dependence in the first age class, constant in the second age class), and apply it to BOTH survival and recaptures. Do we need to make 2 user-defined MOD files, one each for survival and recaptures? No! All you need to do is make one file, and put the “-” sign in front of the parameter count on the top

row.

-7					
1	7	7	7	7	7
	2	7	7	7	7
		3	7	7	7
			4	7	7
				5	7
					6

- Now, run through SURGE, and from the survival model specification menu, choose option “4”, and use our new AGE2.MOD file.
- Then, do the same thing for recaptures - selection option “4”, and use the same AGE2.MOD file we did for survival.
- When you look at the model (either in the output file, or using “-1” from the constraint menu), you will see that SURGE has correctly incremented the parameter indices for survival and recaptures, respectively:

1	7	7	7	7	7
	2	7	7	7	7
		3	7	7	7
			4	7	7
				5	7
<i>survival</i>					6

8	14	14	14	14	14
	9	14	14	14	14
		10	14	14	14
			11	14	14
				12	14
<i>recapture</i>					13

- Remember, as with any “trick”, there are cases when it may get you into trouble (as we’ll see below). Explicitly writing the correct index values into your MOD files will always work, even if it can take a bit

of time.

- Let's try another example, that will show another, slightly more complex application of user-defined models. We will also see how in **some** cases, we can use VAR and PAR files and user-defined models synonymously.

User-defined models and multiple groups

- Consider the following example. We have 2 groups of marked individuals: males and females. We believe that there is age-structure in survival for both groups - we want to test whether or not survival differs between groups. Any difference might occur in one or both age classes, or not at all.
- A previous study of a different population suggested that only juvenile survival varied with time, and that adult survival was pretty well constant. This is lucky for us, since we can use the same user-defined model we used in the last example (very convenient...). This preliminary analysis also showed that recapture rate in a given year was the same for both sexes, but different among years.
- The male and female data are contained in the files M_AGE.SUR and F_AGE.SUR respectively. Before we begin our analysis, let's lay out the parameter structure of the model we are going to fit. First, for survival:

1	7	7	7	7	7
	2	7	7	7	7
		3	7	7	7
			4	7	7
				5	7
<i>males</i>					6

8	14	14	14	14	14
	9	14	14	14	14
		10	14	14	14
			11	14	14
				12	14
<i>females</i>					13

- And then, for recaptures

15	16	17	18	19	20
	16	17	18	19	20
		17	18	19	20
			18	19	20
				19	20
<i>males</i>					20

15	16	17	18	19	20
	16	17	18	19	20
		17	18	19	20
			18	19	20
				19	20
<i>females</i>					20

- So, at this stage, we can use our “trick”, and build a single user-defined model. In fact, it is conveniently exactly the same as the AGE2.MOD file we modified on the previous page!

-7					
1	7	7	7	7	7
	2	7	7	7	7
		3	7	7	7
			4	7	7
				5	7
					6

- Now, you should recognize that what we want to do is run SURGE, and fit the user-defined model to the survival rate for both males and females. This is easy enough - you select option “4” from the survival model specification menu, and tell SURGE to use AGE2.MOD.
- Since there is more than one group, SURGE will ask you if you want to use the same parameter values across data sets - you answer “no”, since at this stage, we want to allow the groups to be different.
- SURGE will then ask you if you want the same parameter structure for

both groups. You answer “yes”, since you want the same model structure (defined in AGE2.MOD) to apply to both sexes.

- For recaptures, we want time-dependence, but the **same** parameter values for both sexes.
- You then proceed with the rest of the analysis. The deviance for this model is 649.685 (np = 20, AIC = 689.685).
- How would we test the hypothesis that (for example), survival rate in the adult age classes is the same for both sexes, but still potentially different in the first age class? If you think about this carefully, you should see how to handle this problem.
- For a hint, look back at parameter matrices for survival on the previous page. In this matrices, we have parameter 7 for the constant survival rate for adult males, and parameter 14 for constant adult survival for females. The fact that these number differ means we are allowing adult survival to differ between groups. If we want to test the hypothesis that these survival rates are, in fact, the same between the sexes, we simply need to reconstruct the user-defined models to reflect this - how? By using the same parameter index for adult survival for both sexes!
- What we want the matrices to look like now is:

survival

1	13	13	13	13	13
	2	13	13	13	13
		3	13	13	13
			4	13	13
				5	13
<i>males</i>					6

7	13	13	13	13	13
	8	13	13	13	13
		9	13	13	13
			10	13	13
				11	13
<i>females</i>					12

recapture

14	15	16	17	18	19
	15	16	17	18	19
		16	17	18	19
			17	18	19
				18	19
<i>males</i>					19

14	15	16	17	18	19
	15	16	17	18	19
		16	17	18	19
			17	18	19
				18	19
<i>females</i>					19

- The question is, how do we construct the user-defined MOD files to accomplish this?
- Well, here is one case where the “trick” won’t work (discussed below). We need to explicitly write two separate MOD files: one for males, and one for females (call them MALE.MOD and FEMALE.MOD, respectively). They will have the same structure as the matrices to the left - the only difference will be the top row values:

7					
1	13	13	13	13	13
	2	13	13	13	13
		3	13	13	13
			4	13	13
				5	13
					6

MALE.MOD

6					
7	13	13	13	13	13
	8	13	13	13	13
		9	13	13	13
			10	13	13
				11	13
					12

FEMALE.MOD

- Why is the first row value for MALE.MOD 7, and FEMALE.MOD 6? Well, recall that periodically we have referred to “new” parameters in

the MOD files? By “new” we mean those parameters which are “new” in this MOD file. In MALE.MOD, there are 7 “new” parameters: 1 → 6 and 13.

- In the FEMALE.MOD file, however, there are only 6 “new” parameters: 7 → 12. Parameter 13 is NOT a “new” parameter, since it exists in MALE.MOD as well.
- So, why isn’t the order “6” in MALE.MOD, and “7” in FEMALE.MOD, rather than the other way around? Simply - a “new” parameter is one which has not occurred in a preceding MOD file. Thus, the numbering we have used implies that we process MALE.SUR first, and then FEMALE.SUR.
- Now, why doesn’t the “trick” work? The “trick” of using the “-” sign doesn’t work when we have common parameters among MOD files - it only works if you have completely different parameter sets among the MOD files. As we noted, “tricks” can be useful, but if you’re not aware of what they do in every circumstance where you try to use them, they could get you into trouble. If “a bug is just an undocumented feature”, then an “undocumented feature is a bug waiting for the right opportunity”.
- Now that we have created MALE.MOD and FEMALE.MOD, we can run our next model.
- From the survival model specification screen, we again choose option “4”. Now, however, for MALE.SUR we want to apply MALE.MOD.
- Next, SURGE asks if we want to use the same parameter values across data sets. This question might lead to a bit of confusions, since clearly, we want a model where adult survival is the same across data sets (i.e., between the sexes), but we don’t want juvenile (first age-class) to necessarily be the same. In this case, we answer “no”, and will use the structure of the MOD files themselves to create the desired parameter equivalencies.
- Next, SURGE will ask if you want to use the same parameter structure across data set. The answer here? “no”!! We want to use a different parameter structure for the females - FEMALE.MOD!! So, we enter “no”, and hit the <enter> key. SURGE will then ask us to pick another model. We again choose option “4”, and enter FEMALE.MOD when prompted (Fig. 8.1).

```

M* MODEL CHOICE : SURVIVAL PROBABILITIES

M*          AGE DEPENDENCE           = 1
M*          TIME DEPENDENCE          = 2
M*          CONSTANCY OVER AGE AND TIME = 3
M*          USER DEFINED MODEL       = 4
M*          TIME DEPENDENCE, DEP. ON EXT.UAR. = 5
M*          2 AGE CLASSES 1=TIME DEP. 2=TIME DEP. = 6
M*                      1=TIME DEP. 2=EXT.UAR. = 7
M*                      1=EXT.UAR. 2=TIME DEP. = 8
M*                      1=EXT.UAR. 2=EXT.UAR. = 9

M* OPTION FOR DATA SET males.sur ? 4

M* USER-DEFINED STRUCTURE IN FILE <=<—|> : male.mod

M* SAME PARAM. VALUES ACROSS DATA SETS ? n
M* SAME PARAM. STRUCTURE ACROSS DATA SETS ? n

M* OPTION FOR DATA SET females.sur ? 4

M* USER-DEFINED STRUCTURE IN FILE <male.mod=<—|> : female.mod

```

Fig. 8.1

- The deviance for this model is 651.198 (np = 19, AIC = 689.198). Thus, this model would appear to be a better model than the starting model where adult survival varied between the sexes ($\chi^2=1.513, df=1, P=0.219$).
- Of course, we could now proceed to test for equivalence of survival in the first age class, additivity in survival between sexes, and forth - again, everything else we’ve done up until now can be applied to user-defined models. All you need to do is keep track of the parameter indexing of the new models.

Another way to constrain user-defined models

- In the preceding example, we built 2 separate user-defined model files (MALE.MOD and FEMALE.MOD) to test for equivalence of adult survival rate between the sexes.
- However, if you think back to Chapter 6, where we introduced the concept of constraints (i.e., VAR and PAR files), you might have

thought about using a different approach to testing this hypothesis other than creating 2 separate MOD files. Could we have used the same MOD file for both sexes, and then used a VAR and PAR file to constrain the adult survival rates to be the same (which, obviously, is what we were doing mechanically by creating the MALE.MOD and FEMALE.MOD files)?

- The answer is yes - all we need to do is set up the appropriate VAR and PAR file, and apply it to the original MOD file (AGE2.MOD). Recall that when we applied the original AGE2.MOD file, the survival model structure was:

1	7	7	7	7	7
	2	7	7	7	7
		3	7	7	7
			4	7	7
				5	7
<i>males</i>					6

8	14	14	14	14	14
	9	14	14	14	14
		10	14	14	14
			11	14	14
				12	14
<i>females</i>					13

- Therefore, what we want to do is constrain parameter 7 and parameter 14 to have the same estimate (i.e., force 7 = 14). How do we do this?
- Recall from Chapter 6 that in our VAR file, we had a column of dummy values (“0” or “1”) to distinguish between different groups. In other words, a different combination of “0” or “1” would characterize 1 group from another.
- In this case, we want to do the opposite - we want to “force” the two groups together - at least in terms of estimating the adult survival rate.
- The first thing we need to do is set up the PAR file - call it MOD.PAR. It is VERY brief (since it only deals with 2 parameters!).

2
7 14

- Now, the VAR file - call it MOD.VAR. It also is very brief - for the same reason:

2	1
\$	
\$	
1	
1	

- Again, 2 parameters. There are 2 groups, so we normally would code them as (say) “1” for males, “0” for females. But since we are constraining the adult survival parameters to be the same for the 2 groups, we use the same dummy variable (in this case, we chose “1”) for both groups.
- If we now apply these VAR and PAR files to AGE2.MOD, you will see we get precisely the same estimates as we got when using MALE.MOD and FEMALE.MOD.
- Now, what about counting parameters? (Yes, by now we all hope that the next version of SURGE will do this for us automatically). We know that the number of parameters should be 19 (since there were 20 parameters when adult survival was allowed to differ between the sexes - since adult survival is constant, removing the group effect lowers the number of parameters by 1 from 20 to 19).
- However, to estimate a 1 degree of freedom difference, a linear model still requires 2 parameters: a slope and an intercept. Thus, it will appear as though 20 parameters have been estimated: 1 slope and 1 intercept (used to reconstitute the common estimate of adult survival), and 18 free survival and recapture estimates.
- Have we made a mistake? Well, not exactly. But, as it turns out, we’ve done more work than we need to. Think back to what we have covered concerning linear models. All we’re really trying to do here is constrain parameters 7 and 14 to have the same value.
- In fact, all we need to do is fit a constraint with an intercept only - no slopes! How do we do this? Actually, it is quite easy.
- First, tell SURGE you want to fit 1 constraint to the model. When



prompted for the .PAR file, enter MOD.PAR (which we created on the previous page).

- Now, the trick - when SURGE then prompts you for “the number of variables in the constraint”, enter “0” <enter>. By telling SURGE you want to use zero variables in the constraint, this is, in effect, telling SURGE to fit “no group effect”, which (if you look at the .MOD.VAR file (previous page), is clearly exactly what we’re trying to do. Try it and confirm this for yourself.
- Clearly, you can do a lot of things with VAR and PAR files, and user-defined models. To be entirely redundant: *if you can conceive of an analysis as a linear model, you can implement it for CMR data with SURGE.*

A more complex example - marked as young and old

- In Chapter 7 we briefly mentioned the question of how to deal with the situation where we have animals marked as both young and adults. We noted that, in effect, the analysis becomes one of comparing survival over a given interval between these two “groups” - we condition our data based on the age at which the individual was marked: young or adult.
- For the birds marked as young, we clearly anticipate the possibility of age-specific differences in survival. Let’s assume time-dependence in survival in the first age-class (spanning one year), with time variation in the adult age classes - for this group (i.e., marked as young). Assume 5 occasions. Thus, our .MOD file would have as its basic structure:

1	5	6	7
	2	6	7
		3	7
			4

- But what about the birds marked as adults? Well, for adults, we

assume simple time-dependence. We don’t expect any age-specificity in adult survival. So, our .MOD file for adults would be:

8	9	10	11
	9	10	11
		10	11
			11

- What comes next? Well, basically what we want to determine is whether or not the “adult survival” differs between the 2 groups. In other words, do the estimates for parameters 5 → 7 (adult survival rates for birds marked as young) differ significantly from estimates for parameters 9 →11 (adult survival for individuals marked as adults).
- Why is parameter 8 not included in our comparison? If you think about it for a moment, you’ll realize why. Parameter 8 is the probability of a bird marked as an adult at occasion 1 surviving to occasion 2. It cannot be compared with the estimate for parameter 1, which is the survival of an individual marked as a juvenile at occasion 1 to occasion 2. So, we would be comparing “apples and oranges” - a juvenile survival rate with an adult survival rate.
- By now, you should know exactly how to do this comparison. In fact, you should realize (especially given the previous 1-2 pages) that there are several ways you could do this. We’ll illustrate the .VAR and .PAR file approach. To use this method, you simply create a .PAR file containing the parameters to be constrained:

6
5 6 7 9 10 11

- What about the .VAR file? Well, again, you should have a pretty good idea what to do - we want to set these parameters equal to each other (i.e., 5=9, 6=10, 7=11). To do this, create a .VAR file with two columns of dummy variables - one column for each group. Then,

simply use the same dummy pair for each parameter between the two groups. For example,

6	2
\$	
\$	
0	0
1	0
0	1
0	0
1	0
0	1

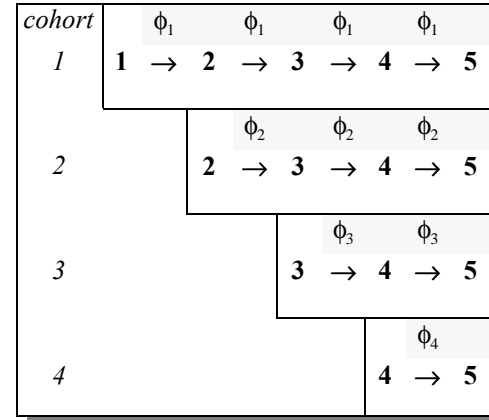
- That’s it! If you’re not sure of the logic, look closely at the structure of the .MOD files, and the .VAR and .PAR files, in this section.

A final example - cohort models

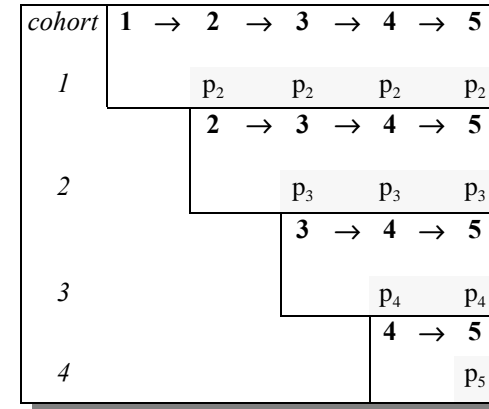
- As our final example of the application of user-defined models, we’ll go back to a topic we touched on briefly in Chapter 7 - cohort models.
- Recall that in a cohort model, we’re interested in testing whether or not survival or recapture differs as a function of the cohort in which an individual is marked. Clearly, this is a case where the standard time-dependent CJS model would not apply (since it assumes that the probability of survival or recapture is not a function of when marked).
- Although cohort models are potentially of real interest, the supported version of SURGE does not offer a simple “option” on the model specification menus to construct cohort models. However, we should now realize that this is not a barrier - only an inconvenience. The solution? User-defined models (of course!).
- Recall that in its simplest form (no time-dependence within cohort), a

cohort model can be represented as:

“survival”



“recapture”



- So, what we need to do to build a user-defined model to represent this “cohort structure” is to first create the parameter matrices corresponding to each of these model structures. Recall from our first 2 examples that a user-defined model is simply the matrix of parameter index values.
- Here are the matrices corresponding to this model (for both survival and recapture):

1	1	1	1
	2	2	2
		3	3
<i>survival</i>			4

5	5	5	5
	6	6	6
		7	7
<i>recapture</i>			8

- Do you understand why? If so, congratulations - you've really been paying attention! If not, go back and re-read p. 8-3 → 8-7. Hint - it involves the use of "-4".

- All that remains is to convert these into .MOD files. In fact, with this example, we could do it in 2 ways. We could create, for example COH_SURV.MOD and COH_REC.MOD (for survival and recapture, respectively) as:

4			
1	1	1	1
	2	2	2
		3	3
			4

4			
5	5	5	5
	6	6	6
		7	7
			8

- However, we've also seen how, in some cases, we can use a time-saving "trick". We could accomplish fitting these user-defined models using only one .MOD file. What would it look like?

-4			
1	1	1	1
	2	2	2
		3	3
			4

As we've seen in this Chapter, the combination of .MOD files (which allows us to specify any model we believe might be appropriate) with the .VAR and .PAR constraint files gives us considerable flexibility in analyzing CMR data. Of course, more complex models require a deeper understanding of linear models, but at the least you should have a good sense of the breadth of possibilities.

