

Summary of changes by version

Details

Version 1.9.6 (1 February 2010)

- Writing the [popan.derived](#) function has led me down all sorts of paths. I had to make one small change to this function to handle externally saved models. However, various changes listed below were brought on by using this function with a relatively large POPAN model.
- Most importantly I discovered an error in computations of real parameters with an mlogit link in which some of the real parameters involved in the mlogit link were fixed. This is NOT a problem if you simply used the real parameter values extracted from MARK; however, if you were using either [compute.real](#) (model.average uses this function) or [covariate.predictions](#) to compute those real parameters (with an mlogit link) then they may be incorrect. This would have been apparent because their value would have changed relative to the original values extracted from the MARK output. Correcting this error involved changes in [compute.real](#), [convert.link.to.real](#) and [covariate.predictions](#) to correct the real parameter estimates and their standard errors. I've not found it in the MARK documentation but deduced that if you use the mlogit link and fix real parameters it uses those fixed real parameter values in the calculation. A simple example will make it clear. Consider pent for 5 occasions where the first is computed by subtraction and you then have 4 real parameters. Let's assume that the 3rd and fourth parameters were fixed to 0. Then the real parameters are calculated as follows: $\text{pent2} = \exp(\beta_2) / (1 + \exp(\beta_2) + \exp(\beta_3) + \exp(0) + \exp(0))$, $\text{pent3} = \exp(\beta_3) / (1 + \exp(\beta_2) + \exp(\beta_3) + \exp(0) + \exp(0))$, $\text{pent4} = 0$, $\text{pent5} = 0$ and $\text{pent1} = 1 - \text{pent2} - \text{pent3} - 0 - 0$ (in this case). Obviously you would not want to fix any real parameters to be >1 , <0 or to have the sum to be <0 or >1 . This structure also had implications on how the standard error was calculated.
- In addition the coding was made more efficient in [covariate.predictions](#) for the case where `data(index=somevector)` is used without any data entries for covariate values in the design matrix. While the task performed with that use of the function could be done with [compute.real](#), it is useful to have the capability in [covariate.predictions](#) as well because it will then model average over the listed set of parameters. The previous approach to coding was inefficient and led to very large matrices that were unnecessary and could cause failure with insufficient memory for large analyses.
- Calculation of `NGross` was added to [popan.derived](#) and logical arguments `N` and `NGross` were added to control what was computed in the call. In addition, argument `drop` was added which is passed to [covariate.predictions](#) to control whether models are dropped when variance of betas are not all positive.
- [var.components](#) was modified to use qr matrix inversion. The returned value for beta is now a dataframe that includes the std errors which are extracted from the vcv matrix. Also, if the design matrix only uses a portion of the vcv matrix, the appropriate rows and columns are now extracted. Prior to this change, the standard errors would have been unreliable if the design matrix didn't use the entire set of thetas and vcv matrix.

Version 1.9.5 (4 December 2009)

- A bug in [covariate.predictions](#) was fixed that would assign fixed values incorrectly if the parameter indices were specified in anything but ascending order. The error would have been

obvious to anyone that may have encountered it because estimated parameters would likely have been assigned a fixed value. In most cases indices would be passed in order if they were selected from the design data unless indices were chosen from more than one parameter type. I discovered it using the [popan.derived](#) function I added in v1.9.4 because it requests indices for multiple parameters in a single function call.

Version 1.9.4 (6 November 2009)

- Note that this version was built with R 2.10 which no longer supports compiled help files (chtml). If you were using compiled help files to get help with RMark, you'll need to switch to regular html files by using `options(help_type="html")` in R. You can put this command in your RProfile.site file so it is set up that way each time you start R. The functionality is the same but it is not as pretty. You can find the index (what used to be in a window on the left) as a link at the bottom of each help page.
- Changed [export.MARK](#) so it will not allow selection of a project name that would over-write an existing .inp file.
- Added the function [popan.derived](#) which for POPAN models computes derived abundance estimates by group and occasion and sum of group abundances for each occasion. For some reason RMark is unable to extract all of the derived parameters from the MARK binary file for POPAN models. This function provides the derived abundance estimates and their var-cov matrix and adds the abundance estimate sum across groups for each occasion which is not provided by MARK. Note that by default confidence intervals are based on a normal distribution to match the output of MARK, but if you want log-normal intervals use `normal=FALSE`.
- The [model.average.list](#) and [model.average.marklist](#) functions were modified to use revised estimator for the unconditional standard error (eq 6.12 of Burnham and Anderson (2002)) which is now the default in MARK. To use eq 4.9 (the prior formula) set the argument `revised=FALSE`.
- Fixed bug in [model.average.list](#) which in some cases failed when the list of var-cov matrices were specified.
- Code in [model.average.marklist](#) was changed to set standard error to 0 if the variance is negative. The same is done in the var-cov matrix for the variance and any corresponding covariances. The results from RMark will now match the model average results from MARK for this case. It is not entirely clear that this is the best approach when ill-fitted models are included.
- Changed code in [cleanup](#) to handle case in which a model did not run.
- Changed use of `grep` and `regexpr` in [convert.inp](#) and [extract.mark.output](#) to accommodate change in R.2.10. The code should work in earlier versions of R but if not update to R2.10.
- Created a function [adjust.value](#) and kept special case of `adjust.chat`. For any field other than `chat` it will adjust the value in `model$results`. As an example, to adjust the effective sample size (ESS) use `model.list=adjust.value("n",value,model.list)` where `value` is replaced with the ESS you want to use. As part of the change [model.table](#) was changed to recompute AICc.

Version 1.9.3 (24 September 2009)

- Default link function for Gamma in the Pradel seniority model had been incorrectly set to log and has now been changed to logit to restrict it to be a probability.
- A bug was fixed in [compute.real](#) and [covariate.predictions](#) in which confidence intervals were being incorrectly scaled by `c` (chat adjustment) instead of `sqrt(c)`. The reported standard errors were correctly using `sqrt(c)` and only the confidence intervals for the real predictions were too large. Simply re-running the prediction computations for a model will provide the correct results. There is no reason to re-run the models. Also this in no way affects model selection.

- A related bug was fixed in [compute.real](#) and [covariate.predictions](#) which created invalid confidence intervals for real parameters if a probability link other than logit was used and a single type of link was used for all real parameters (e.g., sin).

Version 1.9.2 (10 August 2009)

- Added the function [export.MARK](#) which creates a .Rinp, .inp and optionally renames one or more output files for import to MARK. The July 2009 version of MARK now contains a File/RMARK Import menu item which will automatically create the MARK project using the information in these files. This prevents problems that have been encountered in creating MARK projects with RMark output because the data/group structures are setup exactly in MARK as they were in RMark. See [export.MARK](#) for an example and instructions.
- Fixed a problem in [make.design.data](#) which prevented use of `remove.unused` with unequal time intervals and more than one group.
- At least one person has encountered a problem with a very large number of parameters in which RMark created the input file with PIMs written in exponential notation for the larger indices. MARK will not accept that format and it will fail. The solution to this is to set the R option `scipen` to a positive number. Start with `options(scipen=1)` and increase if necessary.

Version 1.9.1 (2 June 2009)

- Fixed a problem [make.design.data](#) which was not using `begin.time` to label the session values
- Made a change in [export.chdata](#) like the change in [make.mark.model](#) to accomodate change with release of version R2.9.0.
- Made a change in [process.data](#) so that `strata.labels` can be specified for Multistrata designs like with ORDMS so an unobserved strata can be included.
- A warning was added to the help file for [export.chdata](#) and [export.model](#) so it is clear that the MARK database must be created correctly and with the .inp file created by [export.chdata](#) from the processed data that was used to create the models that are being exported. This is to ensure that the group structure is setup such that the assumed model structure for groups matches the model structure setup in the .inp file.

Version 1.9.0 (30 April 2009)

- Fixed a bug in [summary.mark](#) which occasionally produced erroneous results with `showall=FALSE`.
- Made a change in [make.mark.model](#) to accomodate change with release of version R2.9.0.
- RMark now requires R version 2.8.1 or higher.

Version 1.8.9 (9 March 2009)

- Changed [model.average.marklist](#) and [covariate.predictions](#) to set NaN or Inf results in v-c matrix to 0 to cope with poorly determined models. Also, for each function the dropping of models is now restricted to cases in which there are negative variances for the betas being used in the averaged parameter estimates. Unused betas are ignored. For example, if [model.average](#) is called with `parameter="Phi"`, then the model will only be dropped if there is a negative variance for one of the betas associated with "Phi".
- In [var.components](#) the tolerance value (`tol`) in the call to `uniroot` was reduced to $1e-15$ which should provide better estimates of the process variance when it is small. Previously a process variance less than $1e-5$ would be treated as 0.

- Made changes to [cleanup](#), [coef.mark](#), and [make.mark.model](#) to accomodate externally saved model objects (`external=TRUE`).

Version 1.8.8 (5 December 2008)

- An error was fixed in [make.time.factor](#) which created incorrect assignments when only some of the time dependent variables contained a "." for occasions with no data.
- Patched [compute.design.data](#) which was not creating the design data in the same order as the PIM construction for the newly added ORDMS model.
- Generalized section of code in [make.mark.model](#) to handle `mlogit` structure for ORDMS model.
- Fixed a bug in [process.data](#) in which the initial ages were not correctly assigned in some situations with multiple grouping variables. Note that it is always a good idea to examine the design data after it is created to make sure it is structured properly because it relates the data and model structure via the grouping variables and the pre-defined variables (ie age, time etc), While I've done a lot of testing, I have certainly not tried every possible example and there is always the potential for an error to occur in a circumstance that I've not encountered.

Version 1.8.7 (13 November 2008)

- An argument `common.zero` was added to function [make.design.data](#) and [compute.design.data](#). It can be set to `TRUE` to make the `time` variable have a common time origin of `begin.time` which is useful for shared parameters like `p` and `c` in closed capture and similar models.
- The function [read.mark.binary](#) was patched to work with the newer versions of MARK.EXE since 1 Oct 2008.
- The model type `ORDMS` for open robust design multi-state models was added. An example data set will be added at a later date after further testing has been completed.
- Some patches were made to fix some aspects of profile intervals and to fix adjustment by chat in [summary.mark](#) when `showall=F`. The notation for profile intervals is now included in the field `model$results$real$note` where `model` is the name of a mark model. Previously an incomplete notation was kept in `model$results$real$fixed` but that field is now used exclusively to denote fixed parameters. It is important to realize that profile intervals computed by MARK are only found in `model$results$real$note` and are not changed by a `chat` adjustment unless the model is re-run. None of the intervals computed by `RMark` and displayed by [summary.mark](#) are profile intervals.

Version 1.8.6 (28 October 2008)

- A bug in an error message for `initial.ages` in [process.data](#) was fixed.
- A new function [var.components](#) was added to provide variance components capability as in the MARK interface except that shrinkage estimators are not computed currently.
- Fixed parameter values are now being reported correctly by [covariate.predictions](#). Also over-dispersion ($c > 1$) was not being included in the variances for parameters except those using the `mlogit` link.
- Some utility functions were added including [pop.est.nat.surv](#), and [extract.indices](#).
- The function [model.average](#) has been changed to a generic function. Currently it supports 2 classes: 1) `list`, and 2) `marklist`. The latter was the original `model.average` which has been renamed [model.average.marklist](#) and the first argument has been renamed `x` instead of `model.list` to match the standard generic function approach. The previous syntax `model.average(...)` will work as long as the usage does not name the first argument as in the

example `model.average(model.list=dipper.results,...)`. The list formulation ([model.average.list](#)) was created to enable a generic model averaging of estimates instead of just real parameter estimates from a `mark` model. It could be used with any set of estimates, model weights and estimates of precision.

- A change is needed to [read.mark.binary](#) to accomodate the change to `mark.exe` with the version dated 1 Oct 2008. Some data types (notably Nest survival) may not work with the new version of `mark.exe`. Working with Gary to make the patch. If you need an older version of `mark.exe` contact me.

Version 1.8.5 (8 October 2008)

- A bug in [process.data](#) was fixed that prevented use of a dataframe contained in a list while using the `groups` argument.
- Profile intervals on the real parameters can now be obtained from MARK using the arguments `profile.int` and optionally `chat` in [mark](#). The argument `profile.int` can be set to `TRUE` and a profile interval will be constructed for all real parameters, or a vector of parameter indices can be specified to restrict the profiling to certain parameters. The value specified by `chat` is passed to MARK for over-dispersion.
- Functions [js](#), [js.lnl](#) were added to provide code for Jolly-Seber models that will run independently of MARK. The [js](#) and [cjs](#) models have been more integrated into RMark via the function [crm](#) which is the wrapper function similar to [mark](#). They now use [process.data](#) and [make.design.data](#). All additional functionality and models will be added through `crm`. While I have done some testing on these functions there is much more to do and they should be treated as exploratory at present. The value from `crm` is an object with class of "crm" and sub-class of either "cjs" or "js". Thus the functions `print.cjs` and `coef.cjs`, were renamed to `print.crm` and `coef.crm` so they will be more general. Also, function [create.dmdf](#) and [cjs](#) were modified to allow for the integration. I do not expect that many people will use these in the near future so they may change structure as this part of the package is expanded.
- Yet another fix to [summary.ch](#) which gave incorrect results for the number recaptured at least once when `marray=F` and the data contained non-unity values for `freq`.

Version 1.8.4 (29 August 2008)

- A generic function [coef.mark](#) was added to extract the table of betas from the model with the expression `coef(model)` where `model` is a `mark` model that has been run and contains output. The table includes standard errors and confidence intervals.
- An argument `brief` was added to [summary.mark](#). If `brief=TRUE` the real parameters are not included in the summary.
- Several functions named [cjs](#), [cjs.lnl](#), [create.dmdf](#), [create.dm](#), [process.ch](#), and `print.cjs` and `coef.cjs` were added. These functions can be used to fit a Cormack-Jolly-Seber model independently from MARK. I still need to develop some more functions to manipulate the results and possibly integrate them better into the RMark structure. This effort is not meant to replace the CJS models in MARK. My intent is to expand this effort to develop a model that will incorporate tag loss estimation. In [create.dmdf](#) I contrast the structure adopted by MARK and the structure used in the package `mra` written by Trent McDonald. In constructing [cjs](#), I've adopted the approach Trent used in `mra` with improvements that have simplified the handling of covariates and modelling.
- A bug in [summary.ch](#) was fixed. It would produce erroneous results when the data contained a non-constant `freq` field. Results with the default of `freq=1` were fine.
- The function [adjust.chat](#) and its help file were changed such that it was clear that a `model.list`

argument was needed.

Version 1.8.3 (25 July 2008)

- For robust design models, an error trap was added to [process.data](#) to make sure that the capture history length matches the specification for the `time.intervals`. This error was already trapped for non-robust models.
- Fixed an error in [make.mark.model](#) that prevented interaction model of session/time-specific individual covariates in a robust design model.
- Fixed an error in [process.data](#) so that the field `freq` is optional for nest survival data sets.
- [print.mark](#) was modified to add an argument `input` which if set to `input=TRUE` will have the MARK input file be displayed rather than the output file. Also, `wait=FALSE` was set in the system command which means the viewer window will be opened and you can carry on with R. Before you had to close the viewer window before proceeding with R.
- An example [RDOccupancy](#) provided by Bret Collier was added for the Robust Occupancy model which shows the use of session and time-varying individual covariates in a robust design model.

Version 1.8.2 (26 June 2008)

- [summary.ch](#) was modified to allow missing cohorts (no captures/recaptures) for an occasion and to fix a bug in which `bygroup=FALSE` did not work when groups were defined.
- To avoid running out of memory, an argument `external` has been added to [collect.models](#), [mark](#), [rerun.mark](#), and [run.mark.model](#). As with all arguments of [mark](#), `external` can also be set in [mark.wrapper](#). Likewise, `external` can also be set in [run.models](#) and it is passed to [run.mark.model](#). The default is `external=FALSE` but if it is set to `TRUE` then the mark model object is saved in an external file with an extension `.rda` and the same base filename as its matching MARK output files. The mark object in the workspace is a character string which is the name of the file with the saved image (e.g., "mark001.rda"). If `external=TRUE` with [mark.wrapper](#) then the resulting marklist contains a list entry for each mark model which is only the filename and then the last entry is the `model.table`. All of the functions recognize the dual nature of the mark object (i.e., filename or mark object) in the workspace. So even if the mark object only contains the filename, functions like [print.mark](#) or [summary.mark](#) will work. However, if you have used `external=TRUE` and you want to look at part of a mark object without using one of the functions, then use the function `load.model`. Whereas, before you may have typed `mymark$results`, if you use `external=TRUE`, you would replace the above with `load.model(mymark)$results`.
- Functions [store](#) and [restore](#) were created to `store` externally and `restore` models from external storage into the R workspace. They work on a `marklist` and are only needed to `store` externally existing marklist models or ones originally created with `external=FALSE` or to `restore` if you change your mind and decide to keep them in the R workspace.
- Error in setup for robust design occupancy models with more than 2 primary sessions was fixed. The error resulted in `mark.exe` crashing.
- The concept of time-varying individual covariates has been expanded to include robust design models which have both primary (session) and secondary (time) occasion-specific data. For a robust design, a time-varying individual covariate can be either session-dependent or session-time dependent. As an example, if there are 3 primary sessions and each has 4 secondary occasions, then the individual covariates can be named `x1,x2,x3` to be primary session-dependent or named `x11,x12,x13,x14,x21,x22,x23,x24,x31,x32,x33,x34`. The value of `x` can be any name for the covariate. In the formula only the base name is used (e.g., `~x`) and RMark fills in the individual covariate names that it finds that match either the session or session-time individual covariates.

Version 1.8.1 (19 May 2008)

- Added function [summary.ch](#) to provide summaries of the capture history data (resighting matrices and m-arrays). It will not work with all types of models at present. It will work with CJS and Jolly-type models.
- Added argument `model.name` to [model.table](#) to be able to use alternate names in the model table. It can use either the model name with each mark object which uses a formula notation (the current approach) or it can use the name of the R object containing the mark model (`model.name=FALSE`). See [model.table](#) for an example. Also, the help file for `model.table` was updated to reflect the code changes implemented in version 1.7.3.
- Code in [mark.wrapper](#) was modified to output the number of columns and column names of the design matrix for each model if `run=FALSE`. This allows a check of each of the columns included in the model. By reviewing these you can assess whether the model was constructed as you intended. If there is any question you can either use [model.matrix](#) or [make.mark.model](#) to examine the design matrix more thoroughly.
- A bug in the new function [merge.design.covariates](#) was fixed in which `merge` was sorting the design data which does obvious bad things. Adding `sort=FALSE` does not appear to mean that the data frame is left in its original order. To prevent this, the dataframe is forced to remain in its original order by adding a sequence field for re-sorting after the merge.

Version 1.8.0 (8 May 2008)

- Fixed a bug in [model.average](#) that caused it to fail and issue an error when any of the models included a time dependent covariate in the parameter being averaged.
- Added [merge.design.covariates](#) which is meant to replace [merge.occasion.data](#). This new function allows covariates to be assigned by `time`, `time` and `group`, or just `group`. It also uses a simplified list of arguments and works with individual design dataframes rather than the entire `ddl`. It uses the R function [merge](#) which can be used on its own to merge design covariates into the design data. You can use [merge](#) directly as this function only checks for some common mistakes before it calls `merge` and it handles reassignment of row names in the case where design data have been deleted. An example, where you might want to use `merge` instead of this function would be situations where the design data are not just `group`, `time` or `group-time` specific. For example, if groups were specified by two different factor variables say initial age and region and the design covariates were only region-specific. It would be more efficient to use `merge` directly rather than this function which would require an entry for each group which would be each pairing of initial age and region. If you use [merge](#) and you deleted design data prior to merging, save the `row.names`, `merge` and then reassign the `row.names`.
- An argument `run` was added to [mark.wrapper](#). If set to `FALSE`, then it will run through each set of models in `model.list` and try to build each model but does not attempt to run it. This is useful to check for and fix any errors in the formula before setting off a large run. If you use `run=FALSE` do not include arguments that are meant to be passed to [run.mark.model](#) like `adjust`.

Version 1.7.9 (7 April 2008)

- [make.design.data](#) was fixed so that `remove.unused=T` will work properly when different `begin.time` values are specified for each group.

Version 1.7.8 (12 March 2008)

- Changed the default link for N to log in the [setup.parameters](#) for the `HetClosed` and `FullHet`.

It was incorrectly set to logit which created incorrect estimates to be computed in [model.average](#) because MARK forces the log link for N regardless of what is set in the input file.

Version 1.7.7 (6 March 2008)

- Supressed warning message that occurred with code to check the validity of the sin link in [make.mark.model](#).
- Fixed a couple of bugs in [covariate.predictions](#) that prevented it from working for some cases after including code for the sin link.
- Added function [release.gof](#) to construct the RELEASE goodness of fit test and extract the TEST2 and TEST3 final chi-square, df and P-values.

Version 1.7.6 (26 Feb 2008)

- [make.mark.model](#) was modified to change the capitalization of the link functions and to remove all spaces after "=" in the input file for mark.exe. These differences were preventing the MARK interface from fully importing the model. Although the model would be imported and could be run inside the MARK interface, median c-hat would not run and would give an error stating "Invalid Link" for any model imported from RMark. Now transferring a model from RMark to the MARK interface is fully functional (I hope). If you want to import an output file that was created with a prior version of RMark without re-running it, use a text editor on the output file and remove any spaces before and after an = sign. Then change the capitalization of the links to "Logit", "MLogit", "Log", "LogLog", "CLogLog", "Identity".
- The sin link is now supported if the formula for the parameter generates an identity matrix for the parameter. For example, if you use ~-1+time instead of ~time then the resulting design matrix will be an identity for time. Likewise, for interactions use ~-1+group:time instead of ~group*time. If you select the sin link and the resulting design matrix is not an identity for the parameter, an error will be given and the run will stop.
- To match the output from MARK, the confidence intervals for real parameters using any 0-1 link including loglog,cloglog,logit and sin are now computed using the logit transformation. For previous versions this will only affect any results that were using loglog and cloglog. Previously, it was using the chosen link to compute the se and the interval endpoints. The latter is still used for the log and identity links which are not bounded in 0-1.
- The model "Jolly" was added to the supported list of models. Parameters include Phi,p,Lambda,N. It is not a particularly numerically stable model and often will not converge. Use of options="SIMANNEAL" in call to [mark](#) is recommended for better convergence. It will take much longer to converge but is more reliable.

Version 1.7.5 (24 Jan 2008)

- [model.average](#) was modified to ignore any models that did not run and either had no attached output file or no results.
- [read.mark.binary](#) and [extract.mark.output](#) were modified to extract and store the real.vcv matrix (var-cov matrix of the simplified real parameters) in the mark object if realvcv=TRUE. The default is realvcv=FALSE. This argument has been added to functions [mark](#), [run.mark.model](#) and [rerun.mark](#).
- An argument delete has also been added to [mark](#) and [run.mark.model](#). The default value is FALSE but if set to TRUE it deletes all output files created by MARK after extracting the results. This is most useful for simulations that could easily create thousands of output files and after extracting the results the model objects are no longer needed. This is just a convenience to replace

the need to call [cleanup](#).

Version 1.7.4 (10 Jan 2008)

- A bug in [make.mark.model](#) was fixed. It was preventing creation of individual (site) covariate models for parameters with only a single parameter (single index) in certain circumstances like Psi1 in the MSOccupancy model.
- The fix to [merge.occasion.data](#) in version 1.7.1 did not work when design data had been deleted. That has been remedied.
- Various functions with some operating specific calls have been modified so they will work on either Windows or Linux. Thus, there is a single file for all source/help for both operating systems in RMarkSource.zip. It can be downloaded to either Windows or Linux to build the package. You need to build the package for Linux but not for Windows. For Windows, you only need RMark.zip which contains the pre-built package which only needs to be installed. Currently, with Linux the variable MarkPath is ignored and mark.exe is assumed to be in the path. Also, for Linux the default for MarkViewer is "pico" (an editor on some Linux machines). This can be modified in [print.mark](#) or by setting MarkViewer to a different value. The one Linux specific function is read.mark.binary.linux. The function [extract.mark.output](#) calls either read.mark.binary.linux or [read.mark.binary](#) depending on the operating system. A Linux version of mark.exe (32 or 64 bit) can be obtained from Evan.

Version 1.7.3 (4 Jan 2008)

- In working with the occupancy models, it became apparent that it would be useful to have a new function called [make.time.factor](#) which creates time-varying dummy variables from a time-varying factor variable. An example is given using observer with the occupancy dataset [weta](#) from the MacKenzie et al Occupancy modelling book.
- To match the results in the book, I added arguments `use.AIC` and `use.lnL` to function [model.table](#) to construct a results table with AIC rather than AICc and -2LnL values. The latter is more useful with a mix of models some using individual covariates and others not.
- A modification was made to [make.mark.model](#) with the MSOccupancy model to fix the name of the added data for parameter p1 when `share=TRUE` to be p2. For an example which uses p2 to construct an additive model, see [NicholsMSOccupancy](#).

Version 1.7.2 (20 Dec 2007)

- In changing code for the occupancy models, a brace was misplaced which prevented the nest survival models from working. This has been fixed. Also, the example code for [mallard](#) and [killdeer](#) was modified to exclude the calls to process the input file. This enables use of the function `example()` to run the example code (e.g. `example(mallard)`). From now on as I add examples they are being included in my test set to avoid this type of problem in the future.

Version 1.7.1 (14 Dec 2007)

- If you update with this version of RMark make sure to update MARK also, so you get the fixes for some of the occupancy models.
- A minor bug was fixed in function [merge.occasion.data](#) that created duplicate row names and prevented the design data from being used in a model.
- Thirteen different occupancy models were added. Models in the following list use the designation from MARK: `Occupancy`, `OccupHet`, `RDOccupEG`, `RDOccupPE`, `RDOccupPG`, `RDOccupHetEG`,

RDOccupHetPE, RDOccupHetPG, OccupRNpoisson, OccupRNNegBin, OccupRPoisson, OccupRNegBin, MSOccupancy. Het means it uses the Pledger mixture and those with RD are the robust design models. The 2 letter designations for the RD models are shorthand for the parameters that are estimated. For EG, Psi, Epsilon, and Gamma are estimated, for PE gamma is dropped and for PG, Epsilon is dropped. For the latter 2 models, Psi can be estimated for each primary occasion. The last 5 models include the Royle/Nichols count (RPoisson) and presence (RNpoisson) models and the multi-state occupancy model. See [salamander](#) for an example of Occupancy, OccupHet, [Donovan.7](#) for an example of OccupRNpoisson, OccupRNNegBin, [Donovan.8](#) for an example of OccupRPoisson, OccupRNegBin, see [RDSalamander](#) for an example of the robust design models and [NicholsMSOccupancy](#) for an example of MSOccupancy. [salamander](#) data.

- The functions [create.model.list](#) and [mark.wrapper](#) were modified so that a list of parameters can be used to loop. This is useful in the situation with shared parameters such as p1 and p2 in the MSOccupancy model, closed models etc. See p1.p2.different.dot in [NicholsMSOccupancy](#) for an example. It can also be useful if the model definitions are linked conceptually (e.g., when one parameter is time dependent, the other should also be time dependent).
- The "." value in an encounter history is now acceptable to RMark and gets passed to MARK for interpretation as a missing value.
- [print.marklist](#) was fixed to show the model table properly after a c-hat adjustment was made. The change in the code in version 1.6.5 to add parameter specific values to the model table had the side-effect of dropping the model name if c-hat was adjusted.

Version 1.7.0 (7 Nov 2007)

- A function [deltamethod.special](#) for computation of delta method variances of some special functions was added. It uses the function `deltamethod` from the package `msm`. You need to install the package `msm` from CRAN to use it.
- A more complete example ([mallard](#)) created by Jay Rotella was added for the nest survival model. His script provides a nice tutorial for RMark and the utility of R to provide a wide-open capability to calculate/plot etc with the results. It also demonstrates the advantages of scripting in R to document your analysis and enable it to be repeated. Before you use his tutorial you need to install the package `plotrix` from CRAN. At a later date, Jay has said he will add some additional examples to demonstrate use of the `deltamethod` function to create variances for functions of the results from MARK.
- Various changes were made to help files. A more complete description of [cleanup](#) was given to tie into [mallard](#) example.

Version 1.6.9 (10 Oct 2007)

- Nest survival model was added to list of MARK models supported by RMark. See [killdeer](#) for an example. Note that the data structure for nest models is completely different from the standard capture history so the functions `import.chdata`, `export.chdata` and `convert.inp` do not work with nest data structure.
- Slight change was made to [run.mark.model](#) and [print.mark](#) to accommodate change in R 2.6.0.

Version 1.6.8 (2 Oct 2007)

- Changes were made to [merge.occasion.data](#) to enable group and time-specific design covariates to be added to the design data.
- Change was made to [setup.parameters](#) to use a log-link for N in the closed-capture models.

MARK forces that link for N but the change was needed for [model.average](#) which does the inverse-link computation. Note that the reported N in [model.average](#) is actually f_0 (number not seen). To get the correct values for N simply add M_{t+1} (unique number captured) to f_0 . That is the way MARK computes N. The std error and confidence interval is on f_0 such that the lower ci on N will never be less than M_{t+1} .

- An error was fixed in the output of [model.average](#). When you selected a specific parameter, it was giving a UCL which was a copy from one of the models and not the UCL from the model averaging. If you didn't specify `vcv=T` it only showed the errant UCL and if you did specify `vcv=T` then it showed the correct LCL and UCL but then added the errant UCL in a column. This occurred because it was adding covariate data for the specific parameter and was shifted a column because of a change in 1.6.1.

Version 1.6.7 (7 Aug 2007)

- Changes were made in [print.mark](#), [print.summary.mark](#) and [compute.design.data](#) to accommodate changes in V2.5.1 of R. When upgrading versions of R problems may occur if RMark was built with an earlier version of R. The version of R that was used to build RMark is listed on the screen each time it is loaded with `library(RMark)` This is RMark 1.6.7 Built: R 2.5.1; i386-pc-mingw32; 2007-08-07 09:00:33; windows
- The help file for [import.chdata](#) was expanded to clarify the differences between it and [convert.inp](#) and the use of the `freq` field.

Version 1.6.6 (14 May 2007)

- Function [make.mark.model](#) was fixed so that the real label indices were properly written when `simplify=FALSE` is used.
- Function [make.mark.model](#) was also changed to remove the parameter simplification for `mlogit` parameters that was added in v1.4.5. I mistakenly assumed that the `mlogit` parameters were setup such that the normalization to sum to 1 was done will all the real parameters in the set (i.e., all PSI for a single stratum). In fact, the `mlogit` values are only specified for the unique real parameters so if there is any simplification and the sum of the probabilities is close to 1 (excluding subtraction value) the values will not be properly constrained. For example, with the [mstrata](#) data if the problem was constrained such that PSI from AtoB was equal to AtoC, it is still necessary to have these as separate real parameters and constrain them with the design matrix. As it turns out, with the [mstrata](#) example it does not matter because the problem is such that the sum of Psi for AtoB and AtoC is not close to 1 (same for other strata) and any link will work. This change will only be noticeable in situations in which the constraint matters (i.e., the probability for the subtraction parameter is near 0). The change back to non-simplification for `mlogit` parameters may increase execution times because the design matrix size has been increased. Previous users of the Multistrata design will see very little difference in there results if they only used models containing `stratum:tostratum` because that will create an all-different PIM within each `mlogit` set. When I ran the [mstrata](#) examples with this version and compared them to v1.6.5 the results were different but they were differences in the 5th or smaller decimal point due to differences in numerical optimization.

Version 1.6.5 (3 May 2007)

- Function [model.table](#) was modified to include parameter formula fields in the `model.table` dataframe of a `marklist`. Previously only the `model.name` was included which is a concatenation of the individual parameter formulas. The additional fields allows extracting the model table

results based on one of the parameter formulas or to create a matrix of model AICc or other values with rows as one parameter and columns as the other. See [model.table](#) for an example.

- Function [process.data](#) was modified such that factor variables used for grouping retain the ordering of the factor levels in the data file. Previously they would revert back to default ordering and the re-leveling would also have to be repeated on the design data also.
- An argument `brief` was added to [mark](#) to control amount of summary output.
- Fixed a bug in [get.real](#) that prevented computation for models without the stored covariate values.
- Added code to [make.mark.model](#) that prevents constructing models with empty rows in the design matrix unless the parameter is fixed. For example, if you were to try `~-1+Time` for the dipper data, it will fail now because there is no value for the intercept (`Time=0`).
- Function [mark.wrapper](#) outputs the model name to the screen before running the model which helps associate any error messages to the model if `output=F`.

Version 1.6.4 (7 March 2007)

- A new function [export.model](#) was created to copy the output files into the naming convention needed to append them into a MARK .dbf/.fpt database so they can be used with the MARK interface features. This is useful to be able to use some of the features not contained in RMark such as median c-hat and variance component estimation. To create a MARK database, first use [export.chdata](#) to create a .inp file to pass the data into MARK. Start MARK and use File/New to create a new database. Select the appropriate Data Type (model in RMark) and fill in the appropriate values for encounter occasions etc. For the Encounter Histories File Name, select the file you created with [export.chdata](#). Once you have created the database in the Program MARK interface, click on the Browse menu item and then Output/Append and select the output file(s) (i.e. those with a Y.tmp) that you exported with [export.model](#). Note that this will not work with output files run with versions of RMark prior to this one because the MARK interface will give a parse error for the design matrix. To get around that you can edit the output file and remove the spaces in the line with the design matrix header. For example, it should look as follows `design matrix constraints=7 covariates=7` without spaces around the = sign.
- The minor change described above was made in the input file with spacing on the design matrix line to enable proper appending of the output into a MARK .dbf/.fpt database.
- The function [cleanup](#) was modified to delete all `mark*.tmp` files. Do not use `cleanup` until you have appended any exported models.
- An argument `use.comments` was added to [import.chdata](#) to enable comment fields to be used as row names in the data frame. A comment is indicated as in MARK with `/* comment */`. They can be anywhere in the record but they must be unique and they can not have a column header (field name).
- Function [create.model.list](#) was modified such that it only includes lists with a `formula` element. This prevents collecting other objects that are named similarly but are not model definitions.

Version 1.6.3 (5 March 2007)

- A minor change in [make.mark.model](#) and [find.covariates](#) was made to accommodate use of the same covariate in different formulas (e.g. Phi and p). Previous code worked except any call to [get.real](#) would fail. Previously a duplicate of the covariate was entered in the data file to MARK. Now only a single copy is passed.
- An argument `default` has been added to the model definition (parameters in [make.mark.model](#) and `model.parameters` in [mark](#)). The argument sets the default value for parameters represented

by design data that have been deleted.

- Checks were added in [make.mark.model](#) to fail if any of the individual covariates used are either factor variables or contain NAs. Both could fail in the MARK.EXE run but the error message would be less obvious. Factor variables can work as an individual covariate, if the levels are numeric. But it was easier to exclude all factor variables from being individual covariates. They can easily be converted to a continuous version (e.g. `Blackduck$BirdAge=as.numeric(Blackduck$BirdAge)-1`). The code for the [Blackduck](#) was changed to make `BirdAge` a continuous rather than factor variable. Factor variables can still be used to define groups and then used in the formula. They just can't be used as individual covariates. This change was made because a factor variable was in the data but not defined in `groups` and when it was used in the formula it would create a float error in MARK.EXE and that would be confusing and hard to track down.

Version 1.6.2 (28 Feb 2007)

- The fix in 1.6.1 to avoid the incorrect design matrix was not sufficiently general and created a parse error in R if you attempted to use any design data covariates that were created with a cut function to create factor variables by binning a variable. This has been corrected in this version.
- The code in [read.mark.binary](#) has been changed to skip over the v-c matrix for the derived parameters if it is not found in the file. This was causing an error with the PRADREC model type.

Version 1.6.1 (17 Jan 2007)

- An important bug was fixed in [make.mark.model](#) in which an incorrect design matrix would be created if you used two individual covariates in the same formula whereby one of the covariate names was contained within the other. For example, if you used `~mass+mass2` where `mass2=mass^2`, it would actually create a design matrix with columns `mass` and `product(mass,mass2)` which would be the model `mass+mass^3`. This happened due to the way the code identified columns where it needed to replace dummy values with individual covariate names. Since `mass` was contained in `mass2` it added `mass` to the column as a product. The code now does exact matching so the error can no longer occur.
- An argument `indices` was added to the function [model.average](#) which enables restricting the model averaging to a specific set of parameters as identified by the all-different parameter indices. This is most useful in large models with many different indices such that memory limitations are encountered in constructing the variance-covariance matrix of the real parameters. For example, with a CJS analysis of data with 18 groups and 26 years of data, the number of parameter indices exceeds 22,000. Even by restricting the parameters to either `Phi` or `p` with the `parameter` argument there are still 11,000 which would attempt to create a matrix containing 11,000 x 11,000 elements which can exceed the memory limit. In most cases, there are far fewer unique parameters and this argument allows you to select which parameters to average.
- Time-varying covariates are no longer needed for all times if the formula is correctly written to exclude them in the resulting design matrix. [make.mark.model](#) still reports missing time-varying covariates but will continue to try and fit the model but if the missing variables are used in the design matrix the model will fail. As an example consider a time varying covariate `x` for recapture times 1990 to 1995. The code expects to find variables `x1990`, `x1991`, `x1992`, `x1993`, `x1994`, `x1995`. However, lets say that the values are only known for 1993-1995. If you define a variable I'll call `recap` in the design data which has a value 1 for 1993-1995 and a value 0 for 1990-1992 then if you use the formula `~recap:x` the resulting design matrix will only use the known variables for 1993-1995 but you will still be warned that the other values (`x1990 - x1992`) are missing.
- A bug was fixed in [extract.mark.output](#) which prevented it from obtaining more than the last

mean covariate value from the MARK output.

- [fill.covariates](#) was modified such that only a partial list of covariate values need to be specified with `data` and the remainder are filled in with default values depending on argument `usemean`.
- The output from [summary.mark](#) was modified for real parameters when `se=T` to include `all.diff.index` to provide the indices of each real parameter in the all-different PIM structure. They are useful to restrict [covariate.predictions](#) and [model.average](#) to a specific set of real parameters.
- A new function [covariate.predictions](#) was created to compute real parameter values for multiple covariate values and their variance-covariance matrix. It will also model average those values if a `marklist` is passed to the function. Two examples from chapter 12 of Cooch and White are provided to give examples of models with individual covariates and the use of this function.
- The default value of `vcv` in [model.average](#) has been changed to `FALSE`.

Version 1.6.0 (27 Nov 2006)

- A bug was fixed in [PIMS](#) which prevented it from working with Multistrata models.
- Bugs were fixed in [make.design.data](#) which prevented use of argument `remove.unused=T` with Multistrata models and also for any type of model when there were no grouping variables.
- Bugs were fixed in [process.data](#) which gave incorrect ordering of initial ages if the factor variable for the age group was numeric and more than two digits. Also, the number of groups in the data was not correct if the number of loss on capture records exceeded the number without loss on capture within a group.
- Bugs were fixed in [setup.parameters](#) and [setup.model](#) that prevented use of the Barker model and that reported an erroneous list of model names when an incorrect type of model was selected.

Version 1.5.9 (26 June 2006)

- A bug was fixed in [convert.inp](#) which prevented the code from working with groups and two or more covariates. Note that there are limitations to this function which may require some minor editing of the file. The limitations have been added to the help file ([convert.inp](#)).

Version 1.5.8 (22 June 2006)

- Argument `options` was added to [mark](#) and [make.mark.model](#) with a default `NULL` value. It is simply a character string that is tacked onto the `Proc Estimate` statement for the MARK `.inp` file. It can be used to request options such as `NoStandDM` (to not standardize the design matrix) or `SIMANNEAL` (to request use of the simulated annealing optimization method) or any existing or new options that can be set on the estimate proc.
- A bug in [model.table](#) was fixed so it would accommodate the change from v1.3 to a `marklist` in which the `model.table` was switched to the last entry in the list.
- A bug in [summary.mark](#) was fixed so it would properly display `QAICc` when `chat > 1`.
- Function [adjust.chat](#) was modified such that it returns a `marklist` with each model having a new `chat` value and the `model.table` is adjusted for the new `chat` value.
- Function [adjust.parameter.count](#) was modified so it returns the `mark` model object rather than using `eval` to modify the object in place. The latter does not work with models in a `marklist` and calls made within functions.

Version 1.5.7 (8 June 2006)

- Argument `data` was added to function [model.average](#) to enable model averaging parameters at specific covariate values rather than the mean value of the observed data. An example is given in the help file.
- Argument `parameter` of function [model.average](#) now has a default of NULL and if it is not specified then all of the real parameters are model averaged rather than those for a particular type of parameter (eg `p` or `Psi`).
- A bug was fixed in function [compute.real](#) that caused the function to fail for computations of `Psi`.

Version 1.5.6 (6 June 2006)

- `print.summary.mark` was modified so fixed parameters are noted.
- Argument `show.fixed` was added to [summary.mark](#) to control whether fixed parameters are shown as NA (FALSE) or as the value at which they were fixed. If `se=T` the default is `show.fixed=T` otherwise `show.fixed=F`. The latter is most useful in displaying values in PIM format (without std errors), so fixed values are displayed as blanks instead of NA.
- Argument `links` was added to [convert.link.to.real](#) and the default value for argument `model` is now NULL. One or the other must be given. If the value for `links` is given then they are used in place of the links specified in the `model` object. This provides for additional flexibility in changing link values for computation (eg use of log with `mlogit`).
- Argument `drop` was added to [model.average](#). If `drop=TRUE` (the default), then any model with one or more non-positive (0 or negative) variances is not used in the model averaging.
- An error in computation of the v-c matrix of `mlogit` link values in `compute.links.from.reals` was fixed. This did not affect confidence intervals for real parameters (eg `Psi`) in `model.average` because it uses the logit transformation for confidence intervals on real parameters that use `mlogit` link (eg `Psi`).
- [get.real](#) was unable to extract a single parameter value (eg constant `Phi` model). This was fixed.
- The argument `parm.indices` was removed from the functions [compute.real](#) and [convert.link.to.real](#) because the subsetting can be done easily with the complete results returned by the functions. This changed the examples in [fill.covariates](#).
- [compute.real](#) and subsequently [get.real](#) return a field `fixed` when `se=T` that denotes whether a real parameter is a fixed parameter or an estimated parameter at a boundary which is identified by having a standard error=0.

Version 1.5.5 (1 June 2006)

- `model` has been deleted from the arguments in `TransitionMatrix`. It was only being used to ascertain whether the model was a Multistrata model. This is now determined more accurately by looking for the presence of `tostratum` in the argument `x` which is a dataframe created for `Psi` from the function [get.real](#). The function also works with the estimates dataframe generated from [model.average](#). See help for [TransitionMatrix](#) for an example.
- An argument `vcv` was added to function [model.average](#). If the argument is TRUE (the default value) then the var-cov matrix of the model averaged real parameters is computed and returned and the confidence intervals for the model averaged parameters are constructed. Models with non-positive variances for betas are reported and dropped from model averaging and the weights are renormalized for the remaining models.
- A new function [compute.links.from.reals](#) was added to the library to transform real parameters to its link space. It has 2 functions both related to model averaged estimates. Firstly, it is used to transform model averaged estimates so the normal confidence interval can be constructed on the link values and then back-transformed to real space. The second function is to

enable parametric bootstrapping in which the error distribution is assumed to be multivariate normal for the link values. From a single model, the link values are easily constructed from the betas and design matrix so this function is not needed. But for model averaging there is no equivalent because the real parameters are averaged over a variety of models with the same real parameter structure but differing design structures. This function allows for link values and their var-cov matrix to be created from the model averaged real estimates.

Version 1.5.4 (30 May 2006)

- In function [mark](#) an argument `retry` was added to enable the analysis to be re-run up to the number of times specified. An analysis is only re-run if there are "singular" beta parameters which means that they are either non-estimable (confounded) or they are at a boundary. Beginning with this version, [extract.mark.output](#) was modified such that the singular parameters identified by MARK are extracted from the output (if any) and the indices for the beta parameters are stored in the list element `model$results$singular`. The default value for `retry` is 0 which means it will not retry. When the model is re-run the initial values are set to the values at the completion of the last run except for the "singular" parameters which are set to 0. Using `retry` will not help if the parameters are non-estimable. However, if the parameters are at a boundary because the optimization "converged" to a sub-optimal set of parameters, then setting `retry` to 1 or a suitably small value will often help it find the MLEs by moving away from the boundary. If the parameters are estimable and setting `retry` does not work, then it may be better to set new initial parameters by either specifying their values or using a model with similar parameters that did converge.
- A new function [rerun.mark](#) was created to simplify the process of refitting models with new starting values when the models were initially created with [mark.wrapper](#) which runs a list of models by using all combinations of the formulas defined for the various parameters in the model. Thus, individual calls to `mark` are not constructed by the user and re-running an analysis from the resulting list would require constructing those calls. The argument `model.parameters` is now stored in the model object and it is used by this new function to avoid constructing calls to rerun the analysis. With this new function you only need to specify the model list element to be refitted, the processed dataframe, the design data and the model list element (or different model) to be used for initial values. See [rerun.mark](#) for an example.
- To make [rerun.mark](#) a viable approach for all circumstances, the functions [mark.wrapper](#) and [model.table](#) were modified such that models that fail to converge at the outset (i.e., does not provide estimates in the output file) are stored in the model list created by the former function and they are reported as models that did not run and are skipped in the `model.table` by the latter function. This enables a failed model to be reanalyzed with [rerun.mark](#) using another model that converged for starting values.

Version 1.5.3 (25 May 2006)

- In function [get.real](#) a fix was made to accommodate constant pims and a warning is given if the v-c matrix for the betas has non-positive variances.
- In function [make.mark.model](#), the argument `initial` can now be a single value which is then assigned as the initial value for all betas. I have found this useful for POPAN models. For some models I have run, the models fail to converge in MARK with the default initial values it uses (I believe it uses `initial=0`). I have had better luck using `initial=1`. By allowing the use of a single value you can use the same generic starting value for each model without figuring out the number of betas in each model. Also note that you can specify another model that has already been run to use as initial values for a new model and it will match parameter values.
- A bad bug was fixed in [cleanup](#) which was unfortunately deleting files containing "out", "inp",

"res" or "vcv" rather than those having these as extensions. This happened without your knowledge if you chose `ask=FALSE`. Good thing I had a backup. Anyhow, I have now restricted it to files that are named by RMark with `markxxxx.inp` etc where `xxxx` is a numeric value. Thus if you assign your own basefile name for output files you'll have to delete them manually. Better safe than sorry.

Version 1.5.2 (18 May 2006)

- Two new functions were added in this version. [convert.inp](#) converts a MARK encounter history input file to an RMark dataframe. This will be particularly useful for those folks who have already been using MARK. Instead of converting and importing their data with [import.chdata](#) they can use the [convert.inp](#) to import their .inp file directly. It can also be used to directly import any of the example .inp files that accompany MARK and the MARK electronic book (<http://www.phidot.org/software/mark/docs/book/>). The second new function is only useful for tutorials and for first time users trying to understand the way RMark works. The function [PIMS](#) displays the full PIM structure or the simplified PIM structure for a parameter in a model. The user does not directly manipulate PIMS in RMark and they are essentially transparent to the user but for those with MARK experience being able to look at the PIMS may help with the transition.

Version 1.5.1 (11 May 2006)

- Functions [compute.link](#) and [get.link](#) were added to compute link values rather than the parameter estimates.
- A function [convert.link.to.real](#) was added to convert link estimates to real parameter estimates. Previously a similar internal function was used within `compute.real` but to provide more flexibility it was put into a separate function.
- An argument `beta` was added to [get.real](#) to enable it to be changed in the computation of the real parameters rather than always using the values in `results$beta`.
- A function [TransitionMatrix](#) was added to create a transition matrix for the Psi values. It is provided for all strata including the `subtract.stratum`. Standard errors and confidence intervals can also be returned.
- [make.mark.model](#) was modified to include `time.intervals` as an element in the mark object.

Version 1.5.0 (9 May 2006)

- If output file already exists user is given option to create mark model from existing files. Only really useful if a bug occurs (which occurred to me from 1.4.9 changes) and once fixed any models already run can be brought into R by running the same model over and specifying the existing base filename. Base filename values are no longer prefixed with MRK to enable this change.
- On occasion MARK will complete the analysis but fail to create the v-c matrix and v-c file. The code has been modified to skip over the file if it is missing and output a warning.
- Two new functions have been added to ease handling of marklist objects. [merge.mark](#) merges an unspecified number of marklist and mark model objects into a new marklist with an optional `model.table`. [remove.mark](#) can be used to remove mark models from a marklist. See [dipper](#) for examples of each function.
- Various changes were made to functions that compute real parameter estimates, their standard errors, confidence intervals and variance-covariance matrix. The functions that were changed include [compute.real](#), [find.covariates](#), [get.real](#), [fill.covariates](#). For examples, see help for latter two functions.

Version 1.4.9 (3 May 2006)

- Argument `initial` of [make.mark.model](#) was not working after model simplification was added in v1.2. This was modified to select initial values from the model based on names of design matrix columns rather than column contents which have different numbers of rows depending on the simplification.
- [extract.mark.output](#) was fixed to extract the correct -2LnL from the output file in situations in which initial values were specified.

Version 1.4.8 (25 April 2006)

- Argument `silent` was added to [mark](#) and [mark.wrapper](#) with a default value of `FALSE`. This overcomes the problem described above in 1.4.7.
- Code was added to [collect.model.names](#) to prevent it from tripping up when files contain an asterisk which R uses for special names.
- Use of T and F was properly changed to `TRUE` and `FALSE` in various functions to prevent errors when T or F are R objects.
- Code for naming files was modified to avoid problems when more than 999 analyses were run in the same directory.
- Bug in setting fixed parameters with argument `fixed=list(index=,value=)` was corrected.
- Argument `remove.intercept` was added to parameter definition to force removal of intercept in designs with nested factor interactions with additional factor variables (e.g., `Psi=list(formula=~sex+stratum:tostratum,remove.intercept=TRUE)`).

Version 1.4.7 (10 April 2006)

- An error was fixed in the `Psi` simplification code. Note that with the fix in 1.4.2 to trap errors, a side effect is that non-trapped errors that occur in the R code will now fail without any error messages. If the error occurs in making the model, then the model will not be run, but you will not receive a message that the model failed. I may have to make the error trapping a user-settable option to provide better error tracking.

Version 1.4.6 (7 April 2006)

- Assurance code was added to test that the mlogits were properly assigned. An error message will be given if there has been any unforeseen problem created by the simplification. This eliminates any need for the user to check them as described under 1.4.5 above.

Version 1.4.5 (6 April 2006)

- For multistrata models, the code for creating the mlogit links for `Psi` was not working properly if there was more than one group. This was fixed in this version.
- Simplification of the PIMS has now been extended to include mlogit parameters. That was not a trivial exercise and while I feel confident it is correct, double check the assignment of mlogit links for complex models, as I have not checked many examples at present. Within a stratum, the corresponding elements for `Psi` for each of the `tostratum` (movement from stratum to each of the other strata excluding the `subtract.stratum`) should have the same `mlogit(xxx)` value such that it can properly compute the value for `subtract.stratum` by subtraction.

Version 1.4.4 (4 April 2006)

- By including the test on model failure, errors that would stop program were not being displayed. This has been fixed in this version.
- An error was fixed in using time-varying covariates when some of the design data had been deleted.

Version 1.4.3 (30 March 2006)

- Problem with pop up window has been fixed. It will no longer appear if the model does not converge but the model will show as having failed.
- An error was fixed in extracting output from the MARK output file when for some circumstances the label for beta parameters included spaces. This now works properly.

Version 1.4.2 (14 March 2006)

- Errors in the FORTRAN code were preventing completion of large batch jobs. Now these errors are caught and models that fail are reported and skipped over. Unfortunately, it does require user intervention to close the popup window. Make sure you select Yes to close the window especially if you use the default `invisible=FALSE` such that the window does not appear. If you select No, you will not be able to close the window and R will hang.
- A new list element was added to `parameters` in [make.design.data](#) for parameters such as `Psi` to set the value of `tostratum` that is computed by subtraction. The default is to compute the probability of remaining in the stratum. The following is an example with strata A to D and setting A to be computed by subtraction for each stratum:

```
ddl=make.design.data(data.processed,
parameters=list(Psi=list(pim.type="constant",subtract.stratum=c("A","A","A","A")),
p=list(pim.type="constant"),S=list(pim.type="constant")))
```

Version 1.4.1 (11 March 2006)

- A value "constant" was added for the argument `pim.type`. Note that `pim.type` is only used for triangular PIMS. See [make.design.data](#)
- Some code changes were made to [make.mark.model](#) which lessen time to create the MARK input file for large models.
- Function [add.design.data](#) was modified to accommodate robust design and deletion of design data; this was missed in v1.4 changes.
- `model.name` argument in [mark](#) and [make.mark.model](#) was not working. This was fixed.

Version 1.4 (9 March 2006)

- Robust design models added. See [robust](#) for an example.
- Function [cleanup](#) was modified so warning messages/errors do not occur if no models/files are found.
- Parameters in the design matrix are now ordered in the same consistent arrangement. In prior versions they were arranged based on their order in the argument call.
- Argument `right` was added to [make.design.data](#), [add.design.data](#) and in `design.parameters` of [make.mark.model](#) to control whether bins are inclusive on the right (default). The [robust](#) example uses this argument in a call to [mark](#).

Version 1.3 (22 Feb 2006)

- Time varying covariates can now be included in the model formula. See [make.mark.model](#) for details.
- New model types for Known (Known-fate) and Multistrata (CJS with different strata) were added. See [Blackduck](#) and [mstrata](#) for examples.
- Specific rows of the design data can now be removed for parameters that should not be estimated. Default fixed values can be assigned. The function [make.design.data](#) now accepts an argument `remove.unused` which can be used to automatically remove unused design data for nested models. It's behavior is also determined by the new argument `default.fixed` in [make.mark.model](#).
- [summary.mark](#) now produces a summary object and [print.summary.mark](#) prints the summary object. Changes were made to output when `se=T`.
- A new function [merge.occasion.data](#) was created to add occasion specific covariates to the design data.
- New functions [mark.wrapper](#) and [create.model.list](#) were created to automate running models from a set of model specifications for each model parameter.
- The argument `begin.time` in [process.data](#) can now be a vector to enable a different beginning time for each group.
- An argument `pim.type` was added to parameter specification to enable using pims with time structure for data sets with a single release cohort for CJS. See [make.design.data](#)
- Model lists created with [collect.models](#) are now given the class "marklist" which is used with [cleanup](#) and [print.marklist](#) (see [print.mark](#)).
- The function [collect.models](#) now places the `model.table` at the end of the returned list such that each model number in `model.table` is now the element number in the returned list. Previously it was 1+ that number.
- Input, output, v-c and residual results files from MARK are now stored in the directory containing the `.Rdata` workspace. They are numbered consecutively and the field `output` contains the base filename. The function [cleanup](#) was created to delete files that are no longer linked to `mark` or `marklist` objects.
- Model averaged estimates and standard errors of real parameters can be obtained with the function [model.average](#).

Version 1.2 (4 Oct 2005)

- By default the PIM structure is simplified to use the fewest number of unique parameters. This reduces the size of the design matrix and should reduce run times.
- The above change was made in some versions still numbered 1.1, but it contained an error that caused the links command for MARK to be constructed incorrectly.
- `adjust` argument has been added to [collect.models](#) to enable control of number of parameters and resulting AIC values.
- `model.list` in [model.table](#) and [adjust.chat](#) can now also be a list of models created by [collect.models](#) which allows operating on sets of models.

Author(s)

Jeff Laake

[Package *RMark* version 1.9.6 [Index](#)]