XSAM Documentation

Sondre Aanes
31/8-2016
Norwegian Computing Center

Documentation

This document documents the codes used for fitting the assessment model XSAM (Aanes 2016a and 2016b) to data.

The model is a statistical catch at age model that currently takes one set of commercial catch at age data, multiple survey indices. The data sets may cover different age and time spans and NA values are allowed.

For all model specific components in this document it is referred to Aanes (2016) for documentation.

The models are fitted using the R-library TMB (Kristensen, 2014) along with the XSAM C++ user template in XSAM.cpp. To be able to utilize the generality of the model framework a series of R functions are written to aid in fitting the model for different types of data and model setup.

Please notice the copyright issues in XSAM.cpp for terms of use.

The documentation is organized as follows:

1) A definition of the variables that is used by the XSAM C++-template
2) A summary of what is returned from the C++template
3) An outline of how to organize the data and choose model settings
4) A definition of the formats needed to use the R functions to aid in using the XSAM C++-template
5) A summary of available functions for extracting estimates, various types of diagnostics and do short term forecasts.

The documentation is currently not complete, but the attached examples (XSAMexample.R) should aid in getting XSAM to run on other data and model specifications.

Definition of variables in XSAM.cpp

The variables entering the C-template is a combination of setup of the model, parameters to be estimated and data used for optimizing the likelihood. Setup of the model and data has the leading term “DATA” while parameters “PARAMETER”. The variables entering the C-code must be a R-list with elements with the exact same names and definitions as the following:

DATA_INTEGER(noYears); vector with time (proportion of the year) for sampling of abudance indices.
DATA_INTEGER(minAge); minimum age in the model.
DATA_INTEGER(A); Index of maximum age; e.g. if the age range is 3-12; minAge=3 and A=10.

DATA_INTEGER(AT); AT≥A the age above data to make predictions. Usually AT=A.
DATA_INTEGER(maxAgePlusGroup); key determining if maximum age is a plus-group (integer 1) and to be modelled as a dynamic pool or not (integer 0).
DATA_INTEGER(RecruitmentProcess); key, 1 if to model as a process according to codes below, or 0 to treat recruits as fixed parameters to be estimated.
DATA_INTEGER(stockRecruitmentModelCode); integer codes 0 (one value for each year), 1 (RW), 2 (Ricker), 3 (BH), 4 (AR 1), and 5 (process with common mean). Note that in the present code choices 2 and 3 only works for minAge=1.
DATA_INTEGER(am); index age for which selectivity is constant.
DATA_INTEGER(Am); index age for which F is constant.
DATA_INTEGER(uam); index for dimension of multivariate selectivity process: Should be equal to am-1 due to sum-to-zero constraint.
DATA_INTEGER(LatentEffort); key, 1 if effort is a latent variable with an underlying AR process, 0 if effort directly is an AR process. Usually this will be set to 0 as the value 1 is relevant for transient changes in effort.
DATA_INTEGER(TSsel); key, 1 if time varying selectivity, 0 if constant selectivity.
DATA_INTEGER(EstimateARInterceptEffort); key, 1 if to estimate intercept in AR and 0 if to set intercept =slope*(1-mean y). Usually set to 1.
DATA_INTEGER(EstimateUInterceptEffort); key, 1 if to estimate intercept in AR (one for each age), 0 if to set intercept =slope*(1-mean y_a). Usually set to 1.
DATA_INTEGER(ymeantype); Relevant if EstimateUInterceptEffort=0, then if this key is 1 it uses values 1:(n-1) to calculate the mean otherwise it uses the values 2:n
DATA_VECTOR(keyLogFconst); vector of indices of age assigning for which age are fishing mortality is set Equal. Usually only F for the two oldest ages included in the analysis is set equal.
DATA_INTEGER(corFlagU); key determining if changes in selectivity is correlated across ages (integer 0 for no correlation, integer 1 for equal correlation, integer 2 for correlation decreasing by distance in age by r^dist). However, a correlation is also introduced due to the separability component of the model, and such that correlation in the selectivity process is generally not identifiable.
DATA_INTEGER(corFlaglogF); key determining if F is correlated (integer 0 for no correlation, integer 1 for equal correlation, integer 2 for correlation decreasing by distance in age by r^dist). But see comment for corFlagU above.
DATA_VECTOR(keyVarF); vector of indices of age assigning for which ages variances in fishing mortality are equal.
DATA_INTEGER(nvf); number of different variance components for logF, i.e. equals the number of unique indices in keyVarF.
DATA_INTEGER(Modelq); key; determining if catchability is modelled accoding to logq=a+b*age+c*age^2 (if Modelq=1). If Modelq=0 one catchability is estimated for each age index specified in keyLogqpar below.
DATA_ARRAY(keyLogqpar); matrix (number of indices by number of ages) of indices of age assigning for which age are q set equal (one row for each index).
DATA_INTEGER(nqs); integer total number of qs to calculate (if Modelq==1, then nqs=3*number of indices, due to 3 parameters in the function).
DATA_VECTOR(fleetIndex); vector with indices of fleets. Must correspond to indices in “obs” below
DATA_VECTOR(sampleTimes); vector with time (proportion of the year) for sampling of abudance indices
DATA_ARRAY(keyVarObs); matrix (number of fleets by number of ages) of indices of age assigning for which ages and fleets observation variances are set equal.
DATA_INTEGER(nObsVar); number of different observations variances for the observations, i.e. equals

the number of unique indices in keyVarObs.

DATA_INTEGER(CatchConstraint); integer key; if set to 1 the total catch component (calculated by sum of weight at age by numbers at age over ages for each year compared to total catch weight) is added to the likelihood. Should not be done since it means double use of catch at age in numbers, and hence should use CatchConstraint=0.

DATA_INTEGER(UseCatchPred); key, if set to 1 it uses prediction along with prediction error of total catch in weight in the assessment year (when catch at age not is available) specified by "CatchPrediction" below. If set to 0, the catch prediction will not contribute to the estimates.

DATA_VECTOR(agerangeC); vector of length 2 where 1. position gives lowest age in catch at age data, and 2. position gives highest.

DATA_ARRAY(agerangeI); matrix with for each index (row) where 1. position gives lowest age in the respective index, and 2. position gives highest. Number of rows equals "nIndices" below.

DATA_INTEGER(nIndices); Number of indices (fishery independent surveys) to be used for fitting the Model.

DATA_VECTOR(SurveyIndex); vector with indices of fleets except the catch data. This means that it equals "fleetIndex" without its first entry. Must correspond to indices in "obs" below

DATA_INTEGER(dimC); integer value of number of ages included in catch data

DATA_VECTOR(dimI); vector of integer values for each "SurveyIndex" of number of ages included in data

DATA_VECTOR(caton); vector holding total catch in weight by year.

DATA_VECTOR(CatchPrediction); vector of length 2 with first entry holding the predicted value of total catch in assessment year and second entry the corresponding prediction error. Values are only relevant if UseCatchPred=1.

DATA_INTEGER(Fbarminage); minimum age for calculating average F.

DATA_INTEGER(Fbarmaxage); maximum age for calculating average F.

DATA_SCALAR(NAval); Value in data to be interpreted as NA.

DATA_INTEGER(ReturnLL); Key to determine whether likelihood components should be returned or not

DATA_ARRAY(obs); matrix holding the data with 4 columns: the first column gives the year, the second the fleet index (corresponding to fleetIndex and SurveyIndex above), the third the age and the fourth the value of the observation. Hence the number of rows equals the total number of observations to use.

DATA_INTEGER(nobs); the total number of observations (number of rows in obs).

DATA_ARRAY(propMat); 2-dimensional array with proportion mature at age (columns) and year (row).

DATA_ARRAY(stockMeanWeight); 2-dimensional array with mean weight at age (columns) and year (row) in stock.

DATA_ARRAY(catchMeanWeight); 2-dimensional array with mean weight at age (columns) and year (row) in catch.

DATA_ARRAY(natMor); 2-dimensional array with natural mortality at age (columns) and year (row)

DATA_ARRAY(propF); 2-dimensional array with proportion of Fishing mortality before spawning at age (columns) and year (row).

DATA_ARRAY(propM); 2-dimensional array with proportion of Natural mortality before spawning at age (columns) and year (row).

DATA_ARRAY(sd_C); matrix holding sd of catch at log scale by year (n,y). n equals dimC.

DATA_ARRAY(R_C); 3 dim array holding correlation matrices of catch at log scale by year (n,n,y).

DATA_ARRAY(sd_I); 3 dim array holding sd of index at log scale by year and fleet (n,y,f). n equals the largest value in dimI. Only the first values up position dimI for the respective indices will be used.

DATA_ARRAY(R_I); 4 dim array holding correlations of index at log scale by year and fleet (n,n,y,f)

PARAMETER_VECTOR(FixedPars); Vector holding the values of the fixed parameters. The dimension of the vector will depend on age and time span in addition to how the model is set up and is most easily seen by looking at the code lines ~170-260. The available R code “CreateParameterList” will keep track of this.

PARAMETER_ARRAY(LP); matrix holding the values of the latent parameters. Since each of the latent parameters is by time step, the number of columns equals noYears while each row refers to each time dependent parameter. The numbers will depend on the age range and setup of the model and is most easily seen by looking at the code lines ~268-322. The available R code “CreateParameterList” will keep track of this.

Output from XSAM

The C-code for XSAM returns the value of the marginal likelihood function which is used by the R optimizer (e.g. nlminb) along with the gradients for optimizing the likelihood. In addition it returns the estimated parameters and it reports back other objects which are functions of the parameters through the ADREPORT macro from the user template. These parameters are SSB Fbar, FbarW, logN, logF, N, F, and optionally likelihood components (if ReturnLL=1).

Running XSAM

For detailed use of TMB it is referred to its documentation (Kristensen, 2014), and the standard TMB functions “MakeADFun” for making the objective function based on the XSAM template, “sdreport” to create standard deviations and correlations of all model parameters. The objective function is optimized using any optimizer implemented in R supporting the use of gradients in which the functions here uses “nlminb”. However, the setup of the data object and parameters correctly may be rather tedious for a none- TMB and/or C-expert user due to the generality and complexity of the model. If these are not entirely correct in terms of e.g. dimensionality of the various parameters, the R-session will terminate with rather mysterious error messages when calling MakeADFun. Therefore, in all following examples it is used R wrapper functions adapted to XSAM to create the dataobject and parameter objects to reduce the likelihood of misspecification of the setup. These functions are found in the file “XSAMutils.R” and are partly documented.

The best way of testing the code is to follow the example attached, but the key issues and objects are described in the following:

One of the main steps in running XSAM is to create a dataobject which is coherent with the settings in the model and the parameters needed to create the objective function. This object holds the objects defined above including the various data sets. In the wrapper functions the datasets are contained in a “datalist” object which in combination with model settings creates the dataobject. The code for creating the “datalist” object is adapted to historical data formats used for Norwegian Spring Spawning Herring and is therefore not likely to work on any other formats. However, the format of this object is documented and should aid in trying the model on other data than attached here for others than TMB, R and C-experts. The model settings are placed in a txt-file to more easily explore different models on the same data set.

Definition dataobject

This is a list with elements equivalent with Definition of variables in XSAM.cpp except for the PARAMETER elements.

If you create the dataobject using the "CreateDataObject" function it takes two objects as input: "Settings" and "datalist". This function does not take information about covariances of observational data as input, but instead it returns the objects sd_C, R_C, sd_I and R_I according to all Standard errors being 1 and all correlations are set to 0. To change this, these objects must be provided using external data. See below and example for more details.

Settings

A list holding the setup options for running the model. The names and definitions of the variables are kept the same as in the C-code as far as possible. The list must contain the following objects:

agerange: vector of length 2 providing minimum and maximum age for the model
yearrange: vector of length 2 providing first and last year
maxAgePlusGroup: key determining if maximum age is a plus-group (integer 1) and to be modelled as a dynamic pool or not (integer 0).
stockRecruitmentModelCode: key, 1 if to model as a process according to codes below, or 0 to treat recruits as fixed parameters to be estimated.
CatchConstraint: integer key; if set to 1 the total catch component (calculated by sum of weight at age by numbers at age over ages for each year compared to total catch weight) is added to the likelihood. Should not be done since it means double use of catch at age in numbers, and hence should use CatchConstraint=0.
corFlaglogF: key determining if changes in selectivity is correlated across ages (integer 0 for no correlation, integer 1 for equal correlation, integer 2 for correlation decreasing by distance in age by r^dist). However, a correlation is also introduced due to the separability component of the model, and such that correlation in the selectivity process is generally not identifiable.
LatentEffort: key, 1 if effort is a latent variable with an underlying AR process, 0 if effort directly is an AR process. Usually this will be set to 0 as the value 1 is relevant for transient changes in effort.
TSsel: key, 1 if time varying selectivity, 0 if constant selectivity.
corFlagU: key determining if F is correlated (integer 0 for no correlation, integer 1 for equal correlation, integer 2 for correlation decreasing by distance in age by r^dist). But see comment for corFlagU above.
EstimateARInterceptEffort: key, 1 if to estimate intercept in AR and 0 if to set intercept =slope*(1-mean y). Usually set to 1.
EstimateUInterceptEffort: key, 1 if to estimate intercept in AR (one for each age), 0 if to set intercept =slope*(1-mean y_a). Usually set to 1.
ymeantype: Relevant if EstimateUInterceptEffort=0, then if this key is 1 it uses values 1:(n-1) to calculate the mean otherwise it uses the values 2:n.
Am: age for which F is constant (note that Am is here the actual age compared to the C-code where it is the index age).
am: age for which selectivity is constant (note that am is here the actual age compared to the C-code where it is the index age).
AT: AT≥A the age above data to make predictions. Usually AT=A.
RecruitmentProcess: key, 1 if to model as a process according to codes below, or 0 to treat recruits as

fixed parameters to be estimated.

keyLogFconst: vector of indices of age assigning for which age are fishing mortality is set equal. Usually only F for the two oldest ages included in the analysis is set equal.

keyVarF: vector of indices of age assigning for which ages variances in fishing mortality are equal.

UseCatchPred: key, if set to 1 it uses prediction along with prediction error of total catch in weight in the assessment year (when catch at age not is available) specified by "CatchPrediction" below. If set to 0, the catch prediction will not contribute to the estimates.

fleetIndex: vector with indices of fleets. Must correspond to indices in "obs" below.

Modelq: key; determining if catchability is modelled accoding to logq=a+b*age+c*age^2 (if Modelq=1). If Modelq=0 one catchability is estimated for each age index specified in keyLogqpar below.

agerangeI: matrix with for each index (row) where 1. position gives lowest age in the respective index, and 2. position gives highest. Number of rows equals "nIndices" below.

keyLogqpar: matrix (number of indices by number of ages) of indices of age assigning for which age are q set equal (one row for each index).

keyVarObs: matrix (number of fleets by number of ages) of indices of age assigning for which ages and fleets observation variances are set equal.

Fbarrange: vector of length 2 providing minimum and maximum age for calculating average F

agerangeC: vector of length 2 providing minimum and maximum age in catch at age data.

NOTE: all indexing information (e.g. keyLogFconst) allows R-style indexing (i.e. refer to first position as 1) or C-style indexing (i.e. refer to first position as 0). Indexing type will be recognized and converted to C-style indexing in the function "CreateDataObject" later on. Index values for data values that are NA (e.g. value in keyLogqpar for an index outside the agerange in data) should be given the value NA.

Settings file:

Instead of creating the settings object it can be placed in a textfile which is read by the R functions "ReadXSAMsettings". This file should contain the exact same elements as the settings object described above. Any blank lines or lines staring with # are not read by the function allowing commenting on the file. See the attached example. Some tests to avoid some crucial errors are implemented in this function and warnings will be printed indicating what might be the problem.

Datalist:

List containing all data to be used for fitting the model. The list must include the following objects:

caa: 2 dimensional numeric object with catch at age (column) by year (row). The column names must be the actual age and row names the actual year. In addition it must contain the numerical attributes ages, years and a scaling factor k. The scaling factor k is multiplied to the catch at age numbers before the data enters the C++ template when this object is passed to the function "CreateDataObject".

SurveyIndices: List where each element holds the data for one survey. Each object in the list is a 2 dimensional numeric object with survey indices at age (column) by year (row). The column names must be the actual age and row names the actual year. In addition it must contain the numerical attributes ages, years and a scaling factor k. The scaling factor k is multiplied to the respective numbers before the data enters the C++ template when this object is passed to the function "CreateDataObject".

matprop: 2 dimensional numeric object with proportion mature at age (column) by year (row). The column names must be the actual age and row names the actual year. In addition it must contain the numerical attributes ages and years.

west: 2 dimensional numeric object with weight at age (column) by year (row) in stock. The column names must be the actual age and row names the actual year. In addition it must contain the numerical attributes ages and years.

weca: 2 dimensional numeric object with weight at age (column) by year (row) in catch. The column names must be the actual age and row names the actual year. In addition it must contain the numerical attributes ages and years.

caton: 2 dimensional numeric object with total catch (1 column) by year (row). The row names must be the actual year. In addition it must contain the numerical attributes years and a scaling factor k. The scaling factor k is multiplied to the respective numbers before the data enters the C++ template when this object is passed to the function “CreateDataObject”.

natMor: 2 dimensional numeric object with natural mortality at age (column) by year (row) in catch. The column names must be the actual age and row names the actual year. In addition it must contain the numerical attributes ages and years.

propF: 2 dimensional numeric object with proportion of F before spawning at age (column) by year (row) in catch. The column names must be the actual age and row names the actual year. In addition it must contain the numerical attributes ages and years.

propM: 2 dimensional numeric object with proportion of M before spawning at age (column) by year (row) in catch. The column names must be the actual age and row names the actual year. In addition it must contain the numerical attributes ages and years.

CatchPrediction: 2 dimensional numeric object with prediction of total catch in the assessment year with name “Prediction” with corresponding standard error at log scale with name “se” by year (rows). The column names must be prediction and row names the actual year. In addition it must contain the numerical attribute k. The scaling factor k is multiplied to the respective numbers before the data enters the C++ template when this object is passed to the function “CreateDataObject”.

NOTE 1: the scaling factors of caton and CatchPredicition must bring the data on the same scale. Moreover they must correspond with the scaling factor of caa such that the sum of catch at age by weight at age over ages brings the calculated total catch at the corresponding scale of caton and catchprediction!

NOTE 2: There are no checks that the settings are according to the data. If the settings represent a subset on terms of age and time span, the data will be reduced to the spans defined in the settings. If the settings defines spans outside the data range, the following estimation is likely to fail!

Parameter object:

A list with elements:

ParameterList: A list with elements:

FixedPars: A vector of starting values of the fixed parameters

LP: A matrix containing the starting values of random (or Latent) Parameters (LP).

MapVector: A vector corresponding to the fixed parameters containing NA values for parameters to be fixed according to the values in ParameterList.

ParameterNames: Names of the fixed parameters corresponding to the names in the C++ template. Parameters that are vectors (e.g. catchabilities for surveys) have the name with a tailing index. This is used later on to create understandable reports of parameter estimates.

To understand details in the naming and dimensions of parameters see the R code in the function "CreateParameterList" and the code lines ~230-360 in the XSAM C++ template.

This object can be created more easily applying the R function "CreateParameterList". This function uses the dataobject which contains the necessary information to create the parameterlist. This function also allows the user to fix certain parameters. By providing the name of the parameter to be fixed (according to its starting value), the function will create a vector of indices passed on to MakeADFun where parameters are fixed by setting the argument "map".

Specification of data error structures

This model template allows utilizing known error structures in data as documented in Aanes (2016a and 2016b). The error distribution is currently set to be log normal. The template takes the covariance structures (defined by decomposition via standard errors and correlation matrices) which is defined in the "dataobject" as fixed and estimates scaling constants of the covariance structures. The scaling constants are controlled by keyVarObs in the settings object. For example if the same scaling constant should be applied within or across different datasets. If the scaling constant is equal for all data sets, the model will estimate one value for the error structure. If all covariance structures are known and specified, this means that there is only one scaling factor which should be fixed at the value 1 (or 0 at the log scale), and no estimation of parameters in the observation error is done. This is achieved by setting all entries in keyVarObs to 1 and run "CreateParameterList" with the argument 'mapnames="loghvec"'. See the example file for more details. NOTE: the variances and correlations must correspond to errors on the log scale.

Using the function FitXSAMFunction

This function takes the dataobject and parameterlist and 1) Fits the model to data using "nlminb", then if convergence is obtained it continues to 2) create summary reports of parameter estimates using TMB::sdreport along with "CreateReportTable", 3) extracts summary statistics of N, F, SSB and Fbar using "extractStatsG" and, 4) extracts residuals using "ExtractResiduals". This function can take also control constraining of parameters (argument "constraints") as well as control parameters (argument "control") used by "nlminb".

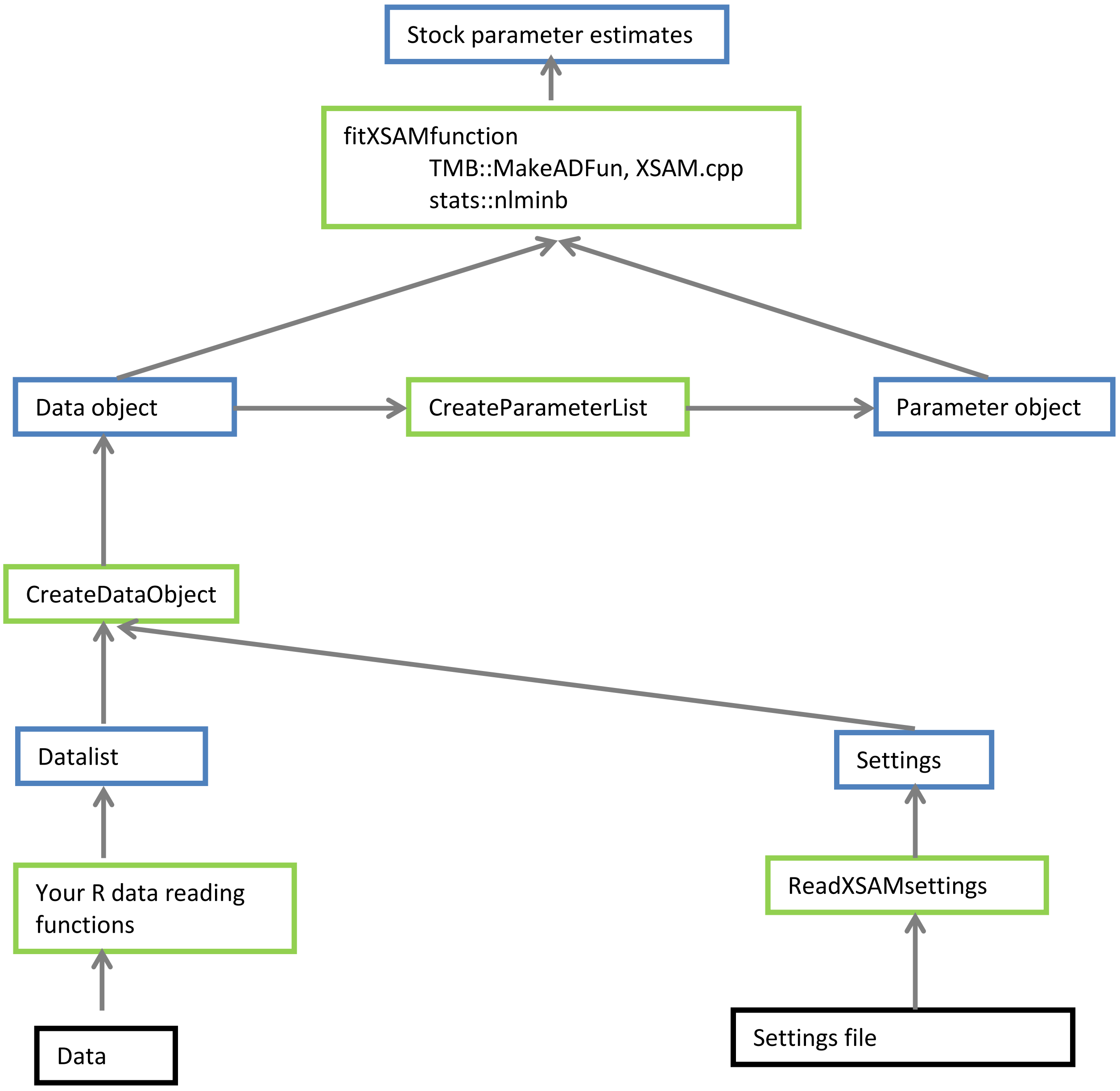
Constraining parameters

It may be necessary to constrain some parameters to achieve convergence. Generally this is the case for very small positive values which are log transformed (eg variance components). Here, this is sometimes the case for the variance components in fishing mortality model particularly if the model includes time varying selectivity. Since "nlminb" allows for box-constrained optimization bounds on parameter ranges may be specified using the "lower" and "upper" arguments in "nlminb". In XSAM all parameters are transformed to the real numbers ranging from –Inf to Inf, and by default there are no constraints on the parameters. Using the function "FitXSAMFunction" constraining parameters is achieved by setting the argument "constraints" appropriately. See the example file for more details.

Known problems

- This code is only properly tested using StockRecruitmentCode=5 and codes 2 and 3 (BH and Ricker) is only correctly implemented if minAge=1!!!

Simplified flow chart of estimation process in XSAM



Black boxes represent data from external resources
Green boxes are R-functions
Blue Boxes are R-objects

References

Aanes, S. 2016a. A statistical model for estimating fish stock parameters accounting for errors in data: Applications to data for Norwegian Spring-spawning herring. WD4 in ICES. 2016. Report of the Benchmark Workshop on Pelagic stocks (WKPELA), 29 February–4 March 2016, ICES Headquarters, Copenhagen, Denmark. ICES CM 2016/ACOM:34. 106pp.

Aanes, S. 2016b. Diagnostics of models fits by XSAM to herring data. WD12 in ICES. 2016. Report of the Benchmark Workshop on Pelagic stocks (WKPELA), 29 February–4 March 2016, ICES Headquarters, Copenhagen, Denmark. ICES CM 2016/ACOM:34. 106pp.

ICES. 2016. Report of the Benchmark Workshop on Pelagic stocks (WKPELA), 29 February–4 March 2016, ICES Headquarters, Copenhagen, Denmark. ICES CM 2016/ACOM:34. 106pp.

Kristensen, K. 2014. TMB: General random effect model builder tool inspired by ADMB. R package version 1.1.