ISC/20/ALBWG-01/03

The test run of future projection for North Pacific albacore stock using the SSfuture C++ and the multivariate normal distribution.

Hirotaka Ijima*

*National Research Institute of Far Seas Fisheries, Fisheries Research and Education Agency 5-7-1, Orido, Shimizu, Shizuoka, 424-8633, Japan

Email: ijima@affrc.go.jp

This working paper was submitted to the ISC Albacore Working Group Stock Assessment Workshop, 16-23 March 2019, held at the NOAA/SWFSC, La Jolla, CA, USA.

Abstract
This working paper shows the preliminary results of a stochastic future projection using SSfuture C++. For the uncertainty of the initial condition, the multivariate normal distribution generated the 1,000 number at age. The additional SS3 outputs ("ss.cor") file are available for the multivariate normal distribution. Lognormal distribution makes stochastic recruitment in this projection. In terms of the management options, we set the constant catch and the constant F scenario that the ISC albacore working group will use in the 2020 stock assessment. In comparison with the 2017 stock assessment result, the initial values of the future projection were improved. I also attached the original Rcpp cord of the SSfuture C++ and that simple example in the Appendix.

Introduction
The ISC Albacore Working Group (ALBWG) agreed to use the Markov chain Monte Carlo methods (MCMC) in the 2020 stock assessment to estimate uncertainties of the Stock Synthesis 3 (SS3) (Methot 2013). However, the results of MCMC was different from the result of maximum likelihood estimates. Thus, ALBWG needs to reconsider how to make initial population numbers for future projection.

This working paper shows the alternative methodology that multivariate normal distribution (MVN) generates the initial population number in the future projection process. Furthermore, using the sampling value of the MVN, the future projection program SSfuture C ++ was executed, and that operation was confirmed. To maintain transparency in the stock assessment, the Rcpp code of SSfuture C++ released, and as an example, the R script that compared with the future projection of SS3 also was described in this document.

Materials and methods
Make initial population number
SS3 has the additional option that estimates the standard deviation (SD) of expected values. The "ss.cor" file reports the SD values and correlation between the predicted values that include the number at age. Thus, we can make the variance-covariance matrix and the MNV. When the statistical distribution generates multiple variables at the same time, the MVN generates realistic stochastic value than the simple normal distribution because the MVN includes the correlation between the variables. Thus, this study made the initial population number that is number at age (NAA) using the MVN.

The test run of the future projection
For the test run, I used on progress SS3 file of North Pacific albacore that version is V3.30.14.08. The MVN generated the 1,000 initial population number, and the other parameters set the constant value that is from the MLE. The latest version of the SSfuture C++ was used for this test run (Ijima 2019). For the stochastic projection, simple log-normal distribution with sigma R was used. The management options were Constant F and Constant Catch Scenario, and the selectivity and F multiplier (Fmulti) was set mean value from 2015 to 2017.

Ensuring transparency of the SSfuture C++

To keep transparency in the stock assessment, the Rcpp code of the SSfuture C++ published in Appendix 1. Appendix 2 is the simple example of the deterministic future projection in R. This R script is comparing with the future projection of SS3 and the results of the SSfuture C++.

Result and discussion

The CV of the number at age was summarised (Figure 1). Juvenile and old populations that have little information shows high uncertainty (Figure 1). Thus, in the generating initial value process, most of the minus values will replace to zero in juvenile and old population bin. It was thought that the trend of uncertainty would keep in different models because the uncertainty of outputs depends on the data.

Evaluating the range of uncertainty, the higher the uncertainty replaces the minus values to the zero (Table 1). Consequently, the average spawning biomass will increase (Table 1). When the CV rises to 0.25, the SSB will be overestimated by 2.84% (Table 1). The lower SD is better because the juvenile population becoming zero in 2018 is unrealistic.

The 95% confidence interval of the estimated future prediction was smoothly connected to the confidence interval of SS3. Also, the average value of the future projection by SSfuture C++ was consistent with the deterministic future forecast of SS3. Hence, the accuracy of the future projection was improved compared to the previous stock assessment of the North Pacific albacore tuna.

References

Methot Jr, Richard D., and Chantell R. Wetzel. "Stock synthesis: a biological and statistical framework for fish stock assessment and fishery management." Fisheries Research 142 (2013): 86-99.

Ijima Hirotaka., "Update of future projection program SSfuture C++." ISC/19/ALBWG-02/07 pp 10.

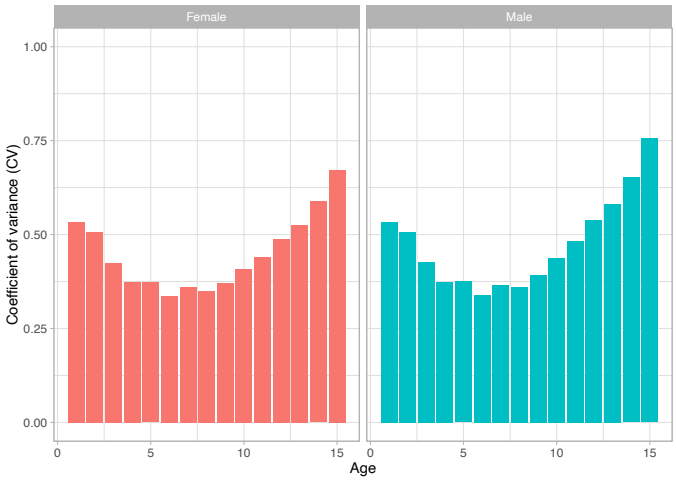

Figure1. Coefficient of variance (CV) of the number at age. The number at age and CV was the result of North Pacific albacore model that is on going.

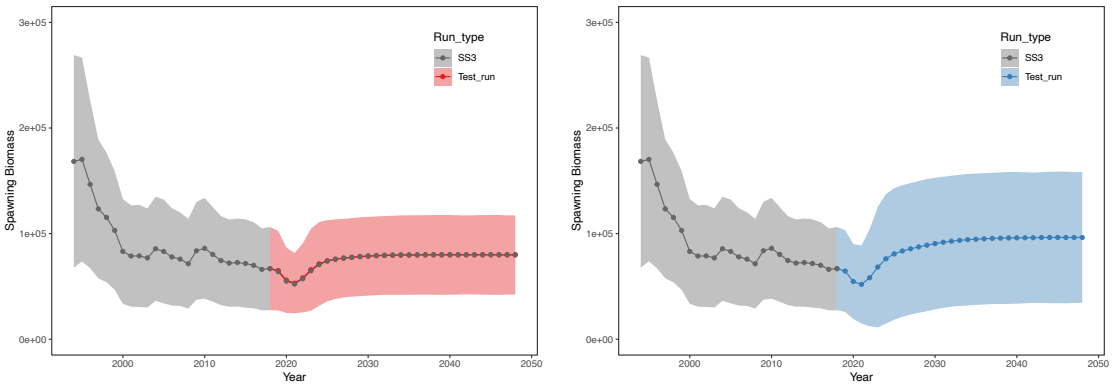

Figure 2. Results of a test run using the SSfuture C++ (Left: constant F scenario, Right: constant catch scenario). The multivariate normal distribution generated the initial condition of future projection that are the number at age.

Table 1. Simulated spawning stock biomass in 2018 using the Multivariate normal distribution. Multivariate normal distribution generates 1,000 samples of the number at age. To evaluate the effect of standard deviation range, CV was added for the result of SS3.

Scenario	SSB 2018 (Mean)	% changes in SSB	% of <0 values
SS3 output	74625.62	—	—
CV+0.00	75729.94	1.48	2.07
CV+0.05	75933.41	1.75	2.98
CV+0.10	76201.5	2.11	4.01
CV+0.15	76526.42	2.55	5.11
CV+0.20	76929.67	3.09	6.43
CV+0.25	76744.21	2.84	7.61

Appendix 1: Rcpp code of SSfuture C++ (ssfcpp20191125.cpp)

```
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]

List ssfcpp(List x)

{

//Parameter and variable declaration----------------------------------------------------------------------------------
    ----------
  double imax = x["imax"];                    //Iteration times
  double tmax = x["tmax"];                    //Projection period
  double recfun = x["recfun"];                //Recruitment function 1:normal, 2:Autocorrelation
  double SB0 = x["SB0"];                      //SB0
  double R0 = x["R0"];                        //R0
  double h = x["h"];                          //Steepness
  double sigmar = x["sigmar"];                //Sigma R
  double age = x["age"];                      //Maximum age (start age 0)
  double gender = x["gender"];                //Number of gender
  double rho = x["rho"];                      //Autocorrelation parameter
  double manage = x["manage"];                //Management scenario 1:Constant F, 2:Constant
    catch
  double catb = x["catb"];                    //Average catch
  double Fmult = x["Fmult"];                  //F multiplier
  double upto = 0.0001;                       //Difference between estimated catch and target catch
  double thresh = 0;                          //Threshold of the target catch
  double limit = x["limit"];                  //Biological limit reference point
  double target = x["target"];                //Biological target reference point
  int assess = x["assess"];                   //Assessment period
  double alpha = x["alpha"];                  //Coefficient of environmental index

  NumericVector spawn_seas = x["spawn_seas"]; //Spawning season (0:no spawning, 1:spawning)
  NumericVector int_n = x["int_n"];           //Initial population number
  NumericVector maa = x["maa"];               //Natural Mortality at Age
  NumericVector vec_sel = x["sel"];           //F at age by quarterly
  NumericVector vec_waa2 = x["waa_mid"];      //Weight at age middle in quarter
  NumericVector fec = x["fec"];               //Maturity X Fecundity
  NumericVector env = x["env"];               //Environmental index

  NumericMatrix fmulti (imax+1, tmax+1);
  vec_sel.attr("dim") = Dimension(gender*(age+1), 4);
  NumericMatrix sel = as<NumericMatrix>(vec_sel);
  NumericMatrix faa ((age+1)*gender, 4);
```

```
  NumericMatrix faa_tmp ((age+1)*gender, 4);
  NumericMatrix zaa ((age+1)*gender, 4);
  NumericMatrix zaa_tmp ((age+1)*gender, 4);
  double sb_tmp = 0;
  double R_tmp = 0;

  vec_waa2.attr("dim") =    Dimension(gender*(age+1), 4);
  NumericMatrix waa2 = as<NumericMatrix>(vec_waa2);

  NumericVector tmp_N ((age+1)*gender);          //Temporal population number matrix
  NumericMatrix sb (imax+1, tmax);               //Spawning biomass
  NumericMatrix R (imax+1, tmax);                //Recruitment
  NumericMatrix Sigma_R (imax+1, tmax);          //Uncertainty of recruitment
  NumericMatrix n(imax+1, tmax+1);     //Uncertainty of recruitment with environmental effect

  NumericVector c (4);                           //Catch weight (mt)
  NumericVector c_est (4);
  NumericMatrix C (imax+1, tmax);                //Catch weight (mt)
  NumericMatrix C_est (imax+1, tmax);            //Estimated catch weigt under constant catch
      scenario

  NumericMatrix N_qt1 ((age+1)*gender, tmax);//Generate quarterly population number to check
      the program
  NumericMatrix N_qt2 ((age+1)*gender, tmax);
  NumericMatrix N_qt3 ((age+1)*gender, tmax);
  NumericMatrix N_qt4 ((age+1)*gender, tmax);

  NumericMatrix N ((age+1)*gender, 5);
  NumericMatrix N_est ((age+1)*gender, 4);

//Set the recruitment deviation-----------------------------------------------------------------------------------
      ------------------

  for (int i = 0; i < imax; i++) {
    Sigma_R(i, _) = rnorm(tmax+1, 0, sigmar);
  }

//Start future projection-----------------------------------------------------------------------------------------
      ----------------

  for (int i = 0; i < imax+1; i++) {

    N(_,0) = clone(int_n); //Set initial population number.

    for (int t = 0; t < tmax; t++) {
```

```
//Calculate total mortality which depends on management scenario.
        if (manage == 1) {

           for (int q = 0; q < 4; q++) {
              faa(_,q) = Fmult*sel(_,q);
              zaa(_,q) = faa(_,q)+maa/4.0;
           }

        } else if (manage == 2) {

           for (int x = 0; x < 10000; x++) {

              fmulti(i,t) = x*0.001;
              N_est = clone(N);

              for (int q = 0; q < 4; q++) {
                 faa(_,q) = fmulti(i,t)*Fmult*sel(_,q);
                 zaa(_,q) = faa(_,q)+maa/4.0;
                 c_est[q] = sum(waa2(_,q)*(faa(_,q)/zaa(_,q))*(1-exp(-zaa(_,q)))*N_est(_,q));
                 N_est(_,q+1) = N_est(_,q)*exp(-zaa(_,q));
              }

              C_est(i,t) = sum(c_est);
              thresh = catb - C_est(i,t);

              if (thresh < upto) break;

           }

        } else if (manage == 3) {

           fmulti(i,0) = 1.0;
           N_est = clone(N);

           for (int q = 0; q < 4; q++) {

              faa_tmp(_,q) = fmulti(i,t)*Fmult*sel(_,q);
              zaa_tmp(_,q) = faa_tmp(_,q) + maa/4.0;
              N_est(_,q+1) = N_est(_,q)*exp(-zaa_tmp(_,q));

              if (spawn_seas[q] == 1) {
                 sb_tmp = sum(fec*N_est(_,q));
              }
           }

           if (t % assess == 0 && t>0) {
```

```
        if(sb_tmp >= target) {
          fmulti(i,t+1) = 1.0;
        } else if (sb_tmp    >= limit && sb_tmp    < target){
          fmulti(i,t+1) = sb_tmp/(target-limit)-limit/(target-limit);
        } else {
          fmulti(i,t+1) = 0.0;
        }

      } else {
        fmulti(i,t+1) = fmulti(i,t);
      }

      for (int q = 0; q < 4; q++) {
        faa(_,q) = fmulti(i,t)*Fmult*sel(_,q);
        zaa(_,q) = faa(_,q) + maa/4.0;
      }

    } else {

      fmulti(i,0) = 1.0;
      N_est = clone(N);

      for (int q = 0; q < 4; q++) {

        faa_tmp(_,q) = fmulti(i,t)*Fmult*sel(_,q);
        zaa_tmp(_,q) = faa_tmp(_,q) + maa/4.0;
        N_est(_,q+1) = N_est(_,q)*exp(-zaa_tmp(_,q));

        if (spawn_seas[q] == 1) {

          sb_tmp = sum(fec*N_est(_,q));
          R_tmp = ((4*h*R0*sb(i,t))/(SB0*(1-h)+sb(i,t)*(5*h-1)))*exp(Sigma_R(i,t)-
(pow(sigmar,2))/2);

        }
      }

      if (t % assess == 0 && t>0) {

        if(R_tmp >= target) {
          fmulti(i,t+1) = 1.0;
        } else if (R_tmp    >= limit && R_tmp    < target){

          fmulti(i,t+1) = R_tmp/(target-limit)-limit/(target-limit);
```

```
            } else {
               fmulti(i,t+1) = 0.0;
            }

         } else {
            fmulti(i,t+1) = fmulti(i,t);
         }

         for (int q = 0; q < 4; q++) {

            faa(_,q) = fmulti(i,t)*Fmult*sel(_,q);
            zaa(_,q) = faa(_,q) + maa/4.0;

         }

      }


//First of all, recruitment need to calculate
   for (int q = 0; q < 4; q++) {

      if (spawn_seas[q] == 1) {

         sb(i,t) = sum(fec*N(_,q));                              //Calculate spawning biomass

         if (i < imax) {

            if (recfun == 1) {

               R(i,t) = ((4*h*R0*sb(i,t))/(SB0*(1-h)+sb(i,t)*(5*h-1)))*exp(Sigma_R(i,t)-
      (pow(sigmar,2))/2);              //Recruitment in time t+1

            } else if (recfun == 2) {

               n(i,0) = Sigma_R(i,0);      //Coefficient representing first-order autocorrelation in time 0
               n(i,t+1) = rho*n(i,t) + pow((1-pow(rho,2)),0.5)*Sigma_R(i,t+1); //Coefficient
      representing first-order autocorrelation in time t+1
               R(i,t) = ((4*h*R0*sb(i,t))/(SB0*(1-h)+sb(i,t)*(5*h-1)))*exp(n(i,t)-(pow(sigmar,2))/2);
      //Recruitment with enviroment effect in time t+1

            } else {

               R(i,t) = ((4*h*R0*sb(i,t))/(SB0*(1-h)+sb(i,t)*(5*h-
      1)))*exp(alpha*env(t))*exp(Sigma_R(i,t)-(pow(sigmar,2))/2);
            }
```

```
        } else {

            R(i,t) = ((4*h*R0*sb(i,t))/(SB0*(1-h)+sb(i,t)*(5*h-1)));
        }

        if (gender == 1) {
            N(0,q) = N(0,q)+R(i,t);                          //Number of recruitment
        } else {

            N(0,q) = N(0,q)+0.5*R(i,t)                       //Number of Female recruitment
            N(age+1,q) = N(age+1,q)+0.5*R(i,t);              //Number of Male recruitment

        }
    }

//Next, population number at next quarter and quarterly catch will calculate
    N(_,q+1) = N(_,q)*exp(-zaa(_,q));
    c[q] = sum(waa2(_,q)*(faa(_,q)/zaa(_,q))*(1-exp(-zaa(_,q)))*N(_,q));     //Catch weight at quarter

    }

    N_qt1(_,t) = N(_,0);                                     //Save quarterly population number in qt1
    N_qt2(_,t) = N(_,1);                                     //Save quarterly population number in qt2
    N_qt3(_,t) = N(_,2);                                     //Save quarterly population number in qt3
    N_qt4(_,t) = N(_,3);                                     //Save quarterly population number in qt4
    C(i,t) = sum(c);                                         //Save catch weight

//Calculate plus group
    if (gender == 1) {

        for (int j = 0; j < age-1; j++) {

            tmp_N[j+1] = N(j,4);

        }

        tmp_N[age] = N(age-1,4)+N(age,4);

    } else {
        for (int j = 0; j < age-1; j++) {
            tmp_N[j+1] = N(j,4);
        }

        for (int j = age+1; j < age*2; j++) {
            tmp_N[j+1] = N(j,4);
        }
```

```
        tmp_N[age] = N(age-1,4)+N(age,4);
        tmp_N[age*2+1] = N(age*2,4)+N(age*2+1,4);

    }

    N(_,0) = tmp_N;

    }
    }

return List::create(_["SB"]=sb, _["C"]=C, _["N_qt1"]=N_qt1, _["N_qt2"]=N_qt2, _["N_qt3"]=N_qt3,
          _["N_qt4"]=N_qt4,
_["R"]=R, _["fmulti"]=fmulti);

}
```

Appendix 2: Simple example

```
#Simple test run---------------------------------

library(dplyr)
library(tidyr)
library(ggplot2)
library(Rcpp)
sourceCpp("ssfcpp20191125.cpp")

#Make input data for SSfuture C++
d = list(
  imax = 100, # Iteration number
  tmax = 30, # Projectoin year
  manage = 1, # Management scenario 1:Constant F, 2:Constant catch, 3:HCR1, 4:HCR2
  limit = NA, # Biological limtit reference point HCR1=ssb, HCR2=R (Target F = Fmult)
  target = NA, # Biological target reference point HCR1=ssb, HCR2=R (Target F = Fmult)
  assess = NA, # Assessment period (management scenario=3,4)
  recfun = 1, # Recruitment function 1:Simple sigma R, 2:Autocorrelation, 3:Environmental effects
  alpha = NA, # Coefficient of environmental index (Needs to Environmental effects recruitment
    function)
  env = NA, # Environmental index (Needs to Environmental effects recruitment function)
  SB0 = 169098, # SSB0
  R0 =    222592.9, # R0
  h = 0.9, # Steepness
  sigmar = 0.5, # Sigma r
  rho = NA, # Autocorrelation parameter that is an option of recruitment when recfun=2.
  age = 15, # Max Age
  gender = 2, # Gender type 1:one gender, 2:two gender
  spawn_seas = c(0, 1, 0, 0), # Spawning season
  int_n = c(0.0000, 38870.2000, 23257.8000, 12566.0000, 4243.2500, 2930.3700, 1858.8400,
    1105.0700, 680.3670, 241.7570, 137.3310, 84.1311, 68.1036, 24.0629, 11.0205, 14.6217,
          0.0000, 38870.2000, 23034.6000, 12387.1000, 4569.9400, 3447.4200, 2412.6100,
    1557.9700, 1026.2300, 386.7980, 229.0170, 146.3000, 124.4960, 46.7694, 23.1059,
    40.8595), # Initial population number (quarter 1 in end year+1)
  maa = c(1.36, 0.56, 0.45, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48, 0.48,
        1.36, 0.56, 0.45, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39, 0.39),
    # Natural mortality at age
  catb = NA, # TAC for constant catch scenarilo
  fec = c(0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 6.40114, 14.95880, 16.70330, 18.08020,
    19.14880, 19.96820, 20.59110, 21.06160, 21.41550, 21.68080, 22.02520,
        0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000,
    0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000), # Maturity X Fecundity: SS3
    configulation
  sel = c(1.470216e-07, 4.528517e-08, 1.896056e-05, 1.946590e-03, 4.476757e-03, 4.398918e-03,
    5.221016e-03, 6.918268e-03, 8.867189e-03, 1.064834e-02, 1.209239e-02, 1.318963e-02,
```

1.399605e-02, 1.457994e-02, 1.500076e-02, 1.552248e-02, 1.470216e-07, 9.775542e-08, 3.927829e-05, 2.027623e-03, 4.440410e-03, 4.459138e-03, 5.788301e-03, 8.645600e-03, 1.202517e-02, 1.489779e-02, 1.675534e-02, 1.760302e-02, 1.769850e-02, 1.733872e-02, 1.675945e-02, 1.524105e-02, 4.023443e-08, 2.454340e-04, 3.660107e-03, 4.890974e-03, 3.196218e-02, 1.299066e-02, 2.030225e-03, 2.867717e-03, 3.904580e-03, 4.820574e-03, 5.524231e-03, 6.029878e-03, 6.382501e-03, 6.626037e-03, 6.794496e-03, 7.000891e-03, 3.758701e-08, 1.417415e-03, 4.344438e-03, 4.859483e-03, 3.195814e-02, 1.176606e-02, 2.393512e-03, 3.983999e-03, 5.707856e-03, 6.893210e-03, 7.411663e-03, 7.440768e-03, 7.206340e-03, 6.870493e-03, 6.523122e-03, 5.834812e-03, 1.605913e-08, 7.697597e-04, 8.638682e-03, 1.238619e-02, 1.112450e-02, 7.876593e-03, 1.880904e-03, 2.987110e-03, 4.031758e-03, 4.886036e-03, 5.530165e-03, 5.995371e-03, 6.324812e-03, 6.556764e-03, 6.720397e-03, 6.924284e-03, 1.465619e-08, 1.870041e-03, 8.840929e-03, 1.236622e-02, 1.101778e-02, 7.163678e-03, 2.510719e-03, 4.310902e-03, 5.946609e-03, 7.033751e-03, 7.523421e-03, 7.563585e-03, 7.338714e-03, 6.992901e-03, 6.616486e-03, 5.832759e-03, 1.929425e-08, 9.080991e-05, 5.913703e-04, 1.628444e-03, 2.951154e-03, 4.416367e-03, 6.056093e-03, 7.850800e-03, 9.376105e-03, 1.055897e-02, 1.142791e-02, 1.204873e-02, 1.248747e-02, 1.279720e-02, 1.301676e-02, 1.329190e-02, 2.009504e-08, 1.543815e-04, 6.129697e-04, 1.629955e-03, 3.034362e-03, 4.863857e-03, 7.331951e-03, 1.009330e-02, 1.232590e-02, 1.371750e-02, 1.432593e-02, 1.437512e-02, 1.409278e-02, 1.364970e-02, 1.315548e-02, 1.209687e-02), # Selectivity

Fmult = 5.030639, # F multiplier (Option: fmsy, fspr, and bench) Startyr and endyr means the time period of selectivity

waa_mid = c(0.0469702, 1.5868700, 4.4681700, 7.5224800, 10.4635000, 13.0519000, 15.2100000, 16.9480000, 18.3154000, 19.3740000, 20.1843000, 20.7994000, 21.2637000, 21.6126000, 21.8740000, 22.2142000,

0.0469702, 1.9989400, 4.7839100, 7.5766400, 10.4625000, 13.2492000, 15.8231000, 18.1283000, 20.1474000, 21.8869000, 23.3671000, 24.6144000, 25.6578000, 26.5256000, 27.2439000, 28.5022000,

0.0736294, 2.0517700, 4.7185200, 7.7073100, 10.5736000, 13.0951000, 15.1988000, 16.8943000, 18.2294000, 19.2637000, 20.0558000, 20.6574000, 21.1116000, 21.4530000, 21.7089000, 22.0409000,

0.0813481, 2.5421600, 4.9486100, 7.7246400, 10.5998000, 13.3855000, 15.9672000, 18.2864000, 20.3229000, 22.0812000, 23.5800000, 24.8450000, 25.9044000, 26.7863000, 27.5170000, 28.7933000,

0.2364650, 2.5748300, 5.4977900, 8.7081900, 11.7638000, 14.4433000, 16.6758000, 18.4742000, 19.8899000, 20.9866000, 21.8264000, 22.4643000, 22.9459000, 23.3080000, 23.5792000, 23.9312000,

0.2874040, 3.0229300, 5.6707200, 8.7038000, 11.8405000, 14.8812000, 17.7025000, 20.2400000, 22.4711000, 24.3996000, 26.0450000, 27.4348000, 28.5996000, 29.5698000, 30.3739000, 31.7790000,

0.6367130, 3.2701200, 6.2459000, 9.3344600, 12.1806000, 14.6263000, 16.6368000, 18.2416000, 19.4967000, 20.4646000, 21.2033000, 21.7630000, 22.1849000, 22.5016000, 22.7387000, 23.0461000,

0.8013890, 3.6489500, 6.3525300, 9.3236300, 12.3157000, 15.1637000, 17.7718000, 20.0950000, 22.1228000, 23.8657000, 25.3463000, 26.5926000, 27.6343000, 28.5000000, 29.2164000, 30.4657000) # Weight at age in the middle of season

```
)

t = proc.time()
set.seed(2020)
res_list = ssfcpp(d)
proc.time()-t

#names(res_list)
#save(res_list, file="res_ssf_f1214.Rdata")

#Compare with deterministic projection result (i max +1 is deterministic projection result)
ssf_ssb = data.frame(res_list$SB) %>% mutate(itre=rep(1:101)) %>%
   gather(t, ssb, -itre) %>%
   arrange(itre) %>%
   mutate(year=rep(2016:2045,101)) %>%
   filter(itre==101) %>%
   select(year,ssb) %>%
   mutate(Run_type="Test_run")

ss_ssb = data.frame(
   year = rep(1994:2045),
   ssb = c(117969, 134084, 131662, 121784, 113464, 98997.2, 80441, 78222.6, 74888.5,  66782.9,
           75377.5, 80526.8, 79887.1, 77721.1, 75587.2, 87139.3, 87850.4, 78733.2, 70459.4,
           72416, 78189.2, 78014.8, 76229.4, 67309.4, 66062, 71964.7, 75955.1, 77765.6, 78578,
           78858.1, 79067, 79301, 79503.9, 79653.4, 79753.2, 79816.1, 79856.7, 79884.3, 79903.7,
           79917.4, 79927, 79933.6, 79938.1, 79941.2, 79943.3, 79944.7, 79945.7, 79946.4,
           79946.9, 79947.2, 79947.4, 79947.5),
   Run_type = rep("SS3",52)
)

d_plot = bind_rows(ssf_ssb,ss_ssb)

g = ggplot()
g = g + geom_line(data=d_plot, aes(x=year,y=ssb,color=Run_type))
g = g + geom_point(data=d_plot, aes(x=year,y=ssb,color=Run_type))
g = g + xlab("Year") + ylab("Spawning Biomass (mt)")
g = g + ylim(c(0,150000))
g = g + theme(legend.title=element_blank())
g
```