St. Pierre bank assessment model

Jonathan Babyn

March 17, 2019

Contents

1 Introduction

2 Model Design

SPAM uses the standard population dynamics model for the abundance at age a in year y as,

$$N_{a,y} = N_{a-1,y-1} \exp(-Z_{a-1,y-1}) \tag{1}$$

and extends it for the plus group A, comprising all fish at and above age A with

$$N_{a,y} = N_{a-1,y-1} \exp(-Z_{a-1,y-1}) + N_{A,y-1} \exp(-Z_{A,y-1}). \tag{2}$$

Total morality $Z_{a,y}$ was assumed to be of the form

$$Z_{a,y} = F_{a,y} + M_{a,y}, \tag{3}$$

with $F_{a,y}$ being fishing mortality and $M_{a,y}$ natural mortality which was fixed to be a constant 0.2.

3 Smoothing Maturities and Weights

Two important pieces for calculating SSB are the weights used and maturity ogives. The maturity ogives provided are the output of a logistic regression run and updated each year from fish that have had their otoliths aged, lengths measured and maturity status determined. This results in very noisy maturity data set for each age group. For both the commercial and stock weights, output from a cohort model going from 1983 to 2016 were provided, along with commercial and stock weights from 1959 to 2014 and 1959 to 2017 respectively provided by DFO. The weights provided by DFO are noisy, and values prior to 1977 are identical and likely copy and pasted from some mean weights derived at some point. Neither of these situations are ideal. The weights either need to be extended thirty back into the past or improved. While the DFO provided weights unfortunately do not have real observations prior to 1977, the values provided at least seem consistent when compared with the output of the cohort model weights. I chose to try and deal with the noise in the DFO provided weights rather than extend the other weights back as the DFO weights are based on actual commercial sampling efforts rather than being based off research vessel survey data and weight length relationships.

Both the maturities and weights were smoothed using Functional Principal Component Analysis (FPCA). FPCA is a non-parametric technique that has been used to perform noise reduction on curve like data[4]. FPCA has at least been at least applied to fisheries data in the context of evaluating spatio-temporal patterns in the data[2] but it has not to the author's knowledge been applied to fisheries weight or growth data. The noise reduction is done by breaking the data down into it's mean function $\mu(t)$, eigenfunctions $\phi_k(t)$ and Functional Principal Components (FPC), ξ_{ki} and keeping only the number of FPCs that explain most of the variation in the data. Further details on FPCA and this process can be seen in Appendix B.

For the maturities, the logit was taken and the FPCA performed in logit space to ensure that maturity proportions would continue to lie between 0 and 1 after smoothing. The FPCA was performed with each age group being considered an observation measured at each year. Smoothing was allowed on the mean function and a user selected bandwidth of 14 was chosen which removed many of the peaks and valleys. The diagnostic plots of the FPCA performed on the maturities showcasing the mean function, first three eigenfunctions, fraction of variance explained by each FPC are seen in Figure 1. Over 90% of the variance is explained by the first FPC, because of this

only the first FPC was used in creating the smoothed data. Plots comparing the smoothed and raw versions of the maturities by age and year are in Figure **??**.

For the weights the years were treated as observations and the age groups as measurement points. Here the mean function was automatically smoothed using Generalized Cross Validation (GCV). The diagnostic plots for the FPCA on the commercial weights are in Figure 3, the diagnostic plots for the stock weights look very similar and are not included here. On the weights data, again over 90% of the variation is explained by a single FPC. Looking at the data represented with a single FPC revealed a distinct gap between the bulk of the data. This gap almost perfectly separates the weights into pre-1994 and post-1994 groups, with a further more obvious recent downshift in weights that occurred after 2011. This grouping along with plots of the raw and smoothed weights are shown in Figure 4. A k-means with $k = 3$ clustering algorithm was also applied to the FPCs of the weights to try and find groupings that way. With the repeated data from 1977 prior removed, almost the same groups as seen in Figure 4 were found on the commercial data and groups that mostly split the stock weight data into pre-1994 and post-1994 clusters..

The DFO provided weights were only provided for ages three through fourteen and needed to be extended for ages 1,2, 15 and 16. This was done by extrapolating the mean function and eigenfunctions of the FPCA using splines at those points. While extrapolating beyond the boundaries is not ideal, the mean function and the first eigenfunction seem well behaved at the boundaries and the imputed values seem reasonable.

Appendices

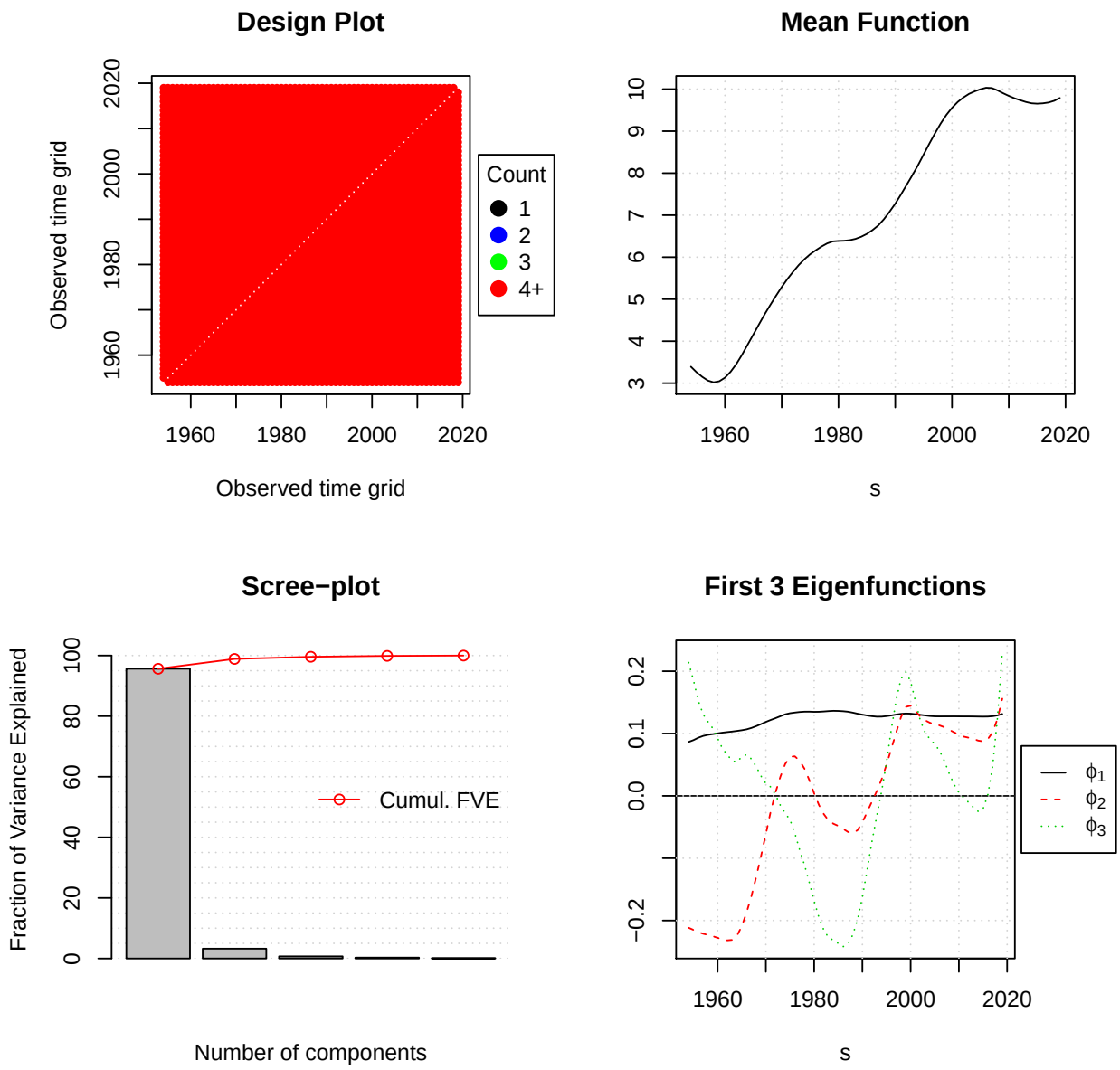

Figure 1: Diagnostic plots the FPCA performed on maturity proportions. Top right shows the smoothed mean function, bottom right shows the first 3 eigenfunctions, bottom left the fraction of variance explained by each FPC, top left is the number of measurements at each grid point

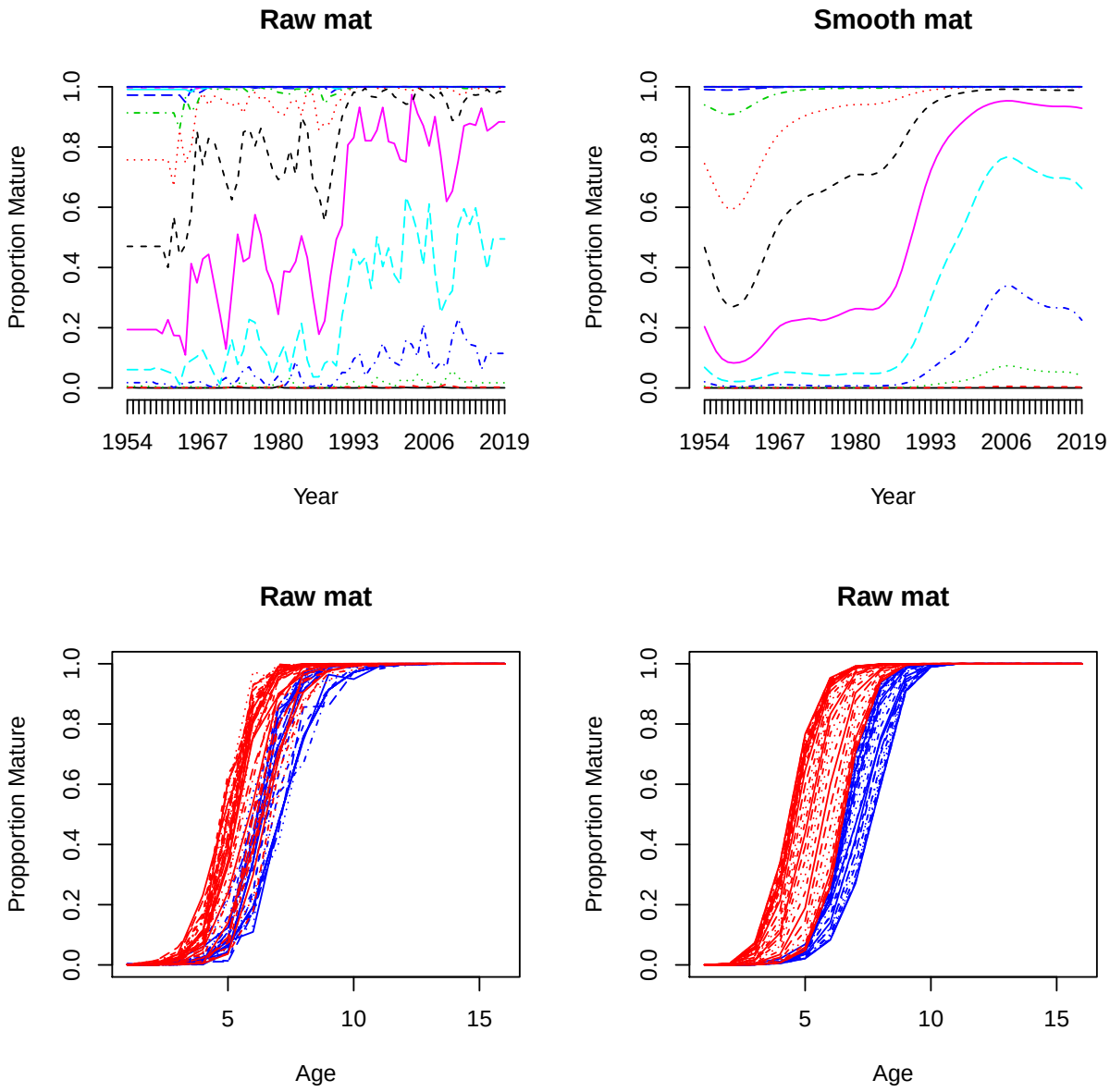

Figure 2: Raw and smoothed maturity proportion curves both by age and by year. For the proportions by age, red curves are those post-1980, blue are pre-1980 suggesting later cohorts mature faster.

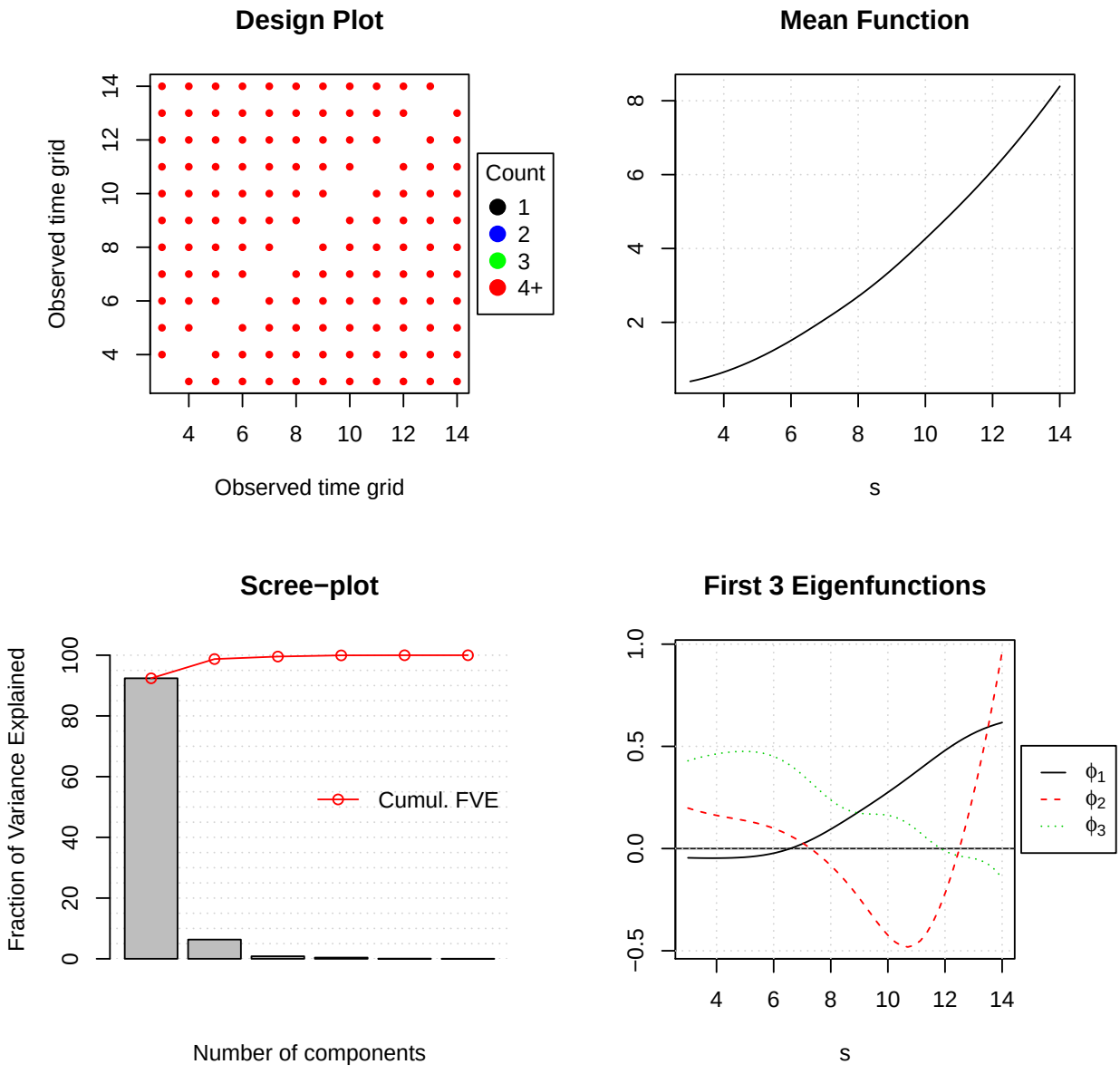

Figure 3: Diagnostic plots of the FPCA performed on the commercial weights.

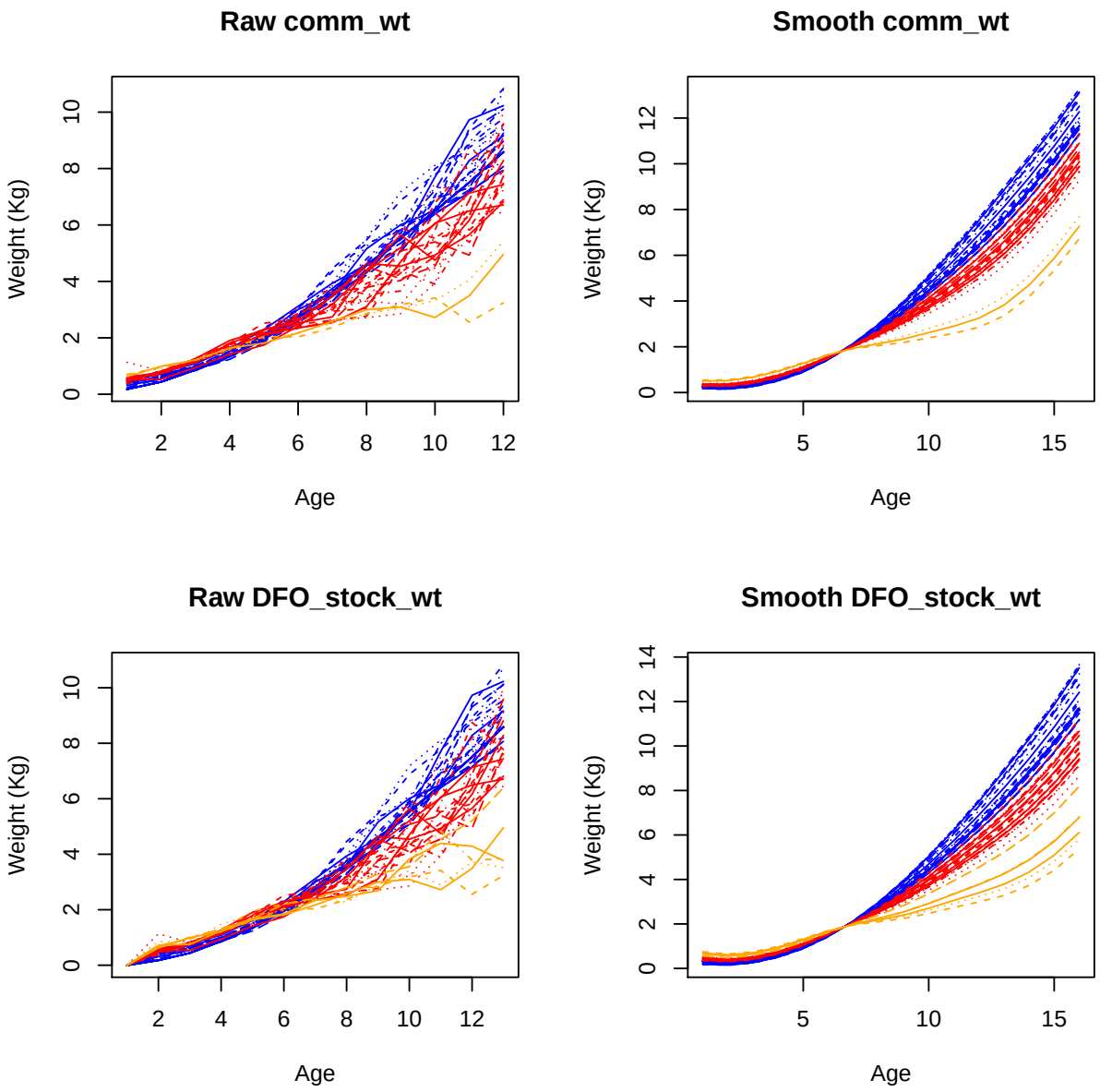

Figure 4: Raw DFO provided commercial weights vs. those smoothed by FPCA. Blue curves are those pre-1994, Red are post-1994 and orange are post-2011. Ages 2,3,15 and 16 weights were created by extrapolating the mean function and eigenfunctions to those points using splines and then imputing based on those extrapolations.

A Numerically stable logged normal cumulative distribution function derivatives in TMB

`TMB` uses Automatic Differentation (AD) to generate the first and second derivatives which are used to help with both performing MLE for parameters and integrating random effects out of the likelihood with the Laplace Approximation (LA). `TMB` accomplishes this with a mixture of forward and reverse AD accumulation modes. Further details on AD can be seen in Fournier et al[3]. `TMB` automatically tries to determine the derivatives of the user's template. However, it can not automatically determine the most numerically stable form of the derivative which can often be necessary when working with extremely small values that can commonly occur when working with probabilities. `TMB` also includes the ability to manually define the forward and reverse modes for a given function to overcome this limitation using atomic functions. By default `TMB` includes a built in atomic version of the normal CDF, `pnorm` based on the version in `R`'s `C` language math library. This built-in version does not support the ability to return a logged value of the probability and shares the same limitation of returning 0 or 1 for negative or positive values of standard normal Z scores when $|Z| >\sim 38$ due to floating point limitations. This clearly means that for any $Z < -38$, `TMB` will return a `NaN` in the gradient when trying to perform `log(pnorm(Z))` due to log(0) being negative infinity. In practice however this is actually much worse since the derivative of `log(pnorm(Z))` is

$$\frac{e^{-\frac{z^2}{2}}}{\sqrt{2\pi}\left(\frac{\mathrm{erf}\left(\frac{z}{\sqrt{2}}\right)}{2}+\frac{1}{2}\right)} \tag{4}$$

and easily results in the division of two extremely small numbers. With floating point limitations this quickly becomes undefined if not handled correctly.

I created an atomic function to specifically deal with the problem of numerical stability when trying to do `log(pnorm(Z))` and similar calculations in the objective function of SPAM such as those done when using a censored likelihood for the detection limit for fitting the observation equations for the survey indices. This requires specifying more numerically stable versions for both the forward and reverse AD modes. For the forward mode this is just the `double C++` function giving the log of the normal CDF and all that requires is calling the version of `pnorm` in `R`'s `C` math library with the `give.log` flag turned on. For the reverse mode just using Equation 4 is not numerically stable due to the division. It can easily be seen that Equation 4 is equivalent to

$$\exp\left(\log\left(\frac{e^{-\frac{z^2}{2}}}{\sqrt{2\pi}}\right)-\log\left(\frac{\mathrm{erf}\left(\frac{z}{\sqrt{2}}\right)}{2}+\frac{1}{2}\right)\right). \tag{5}$$

Working in log space is much less likely to result in under or overflow of floating point values when performing division of small numbers. Using numerically stable versions of the logged normal PDF & CDFs and the form in Equation 5 results in a numerically stable reverse mode derivative for `log(pnorm(Z))`. The atomic function was wrapped in the function `pnorm4` and made available for others to use in the `C++` header file `pnorm4.hpp`. Using `pnorm4(Z)` in place of `log(pnorm(Z))` allows for running one-sided censor likelihood bounds without the need for using two runs of optimization.

It's accuracy was checked by comparing the results of the gradient and hessian generated by `TMB` for `pnorm4` against the numerical first and second derivatives of `pnorm4` done by the `R` software package `numDeriv` along with the brute force first and second derivatives of the normal log(CDF) calculated by the open-source computer algebra system `Maxima` with 5000 digits of floating point precision.

A.1 Extending stability to censored bounds

Censored likelihood bounds pose a similar problem to the above. When using the built-in functions to add normally distributed censored log-likelihood bounds this can be done like `log(pnorm(ZU))-log(pnorm(ZU))` where `ZU` & `ZL` are Z scores of the upper and lower bounds respectively. Just replacing the calls to `pnorm` with the more stable `pnorm4` and using the brute force approach or `logspace_sub` to perform the subtraction is not stable either. This is because for $Z > 38$ `pnorm` will return 1, or 0 for the logged version which will then result in trying to take the log of 0, again leading to `NaN` in the gradient.

The normal CDF of the upper and lower bounds can be thought of as $\Phi(ZU) = 1 - \epsilon_u$ and $\Phi(ZL) = 1 - \epsilon_l$ where the $0 < \epsilon < 1$ and the censored log-likelihood bound is

$$\log(\Phi(ZU) - \Phi(ZL)) = \log(1 - \epsilon_u - (1 - \epsilon_l)) = \log(\epsilon_l - \epsilon_u). \tag{6}$$

It can also be seen that

$$\log(\Phi(-ZL) - \Phi(-ZU)) = \log(\epsilon_l - \epsilon_u) \quad \text{since} \quad \Phi(-Z) = 1 - \Phi(Z) = 1 - (1 - \epsilon) = \epsilon. \tag{7}$$

Since the problem of `NaN`s in the gradient is occurs because the relatively large value of 1 overwhelms the incredibly small values of ϵ for large positive values of Z again due to floating point limitations. This was solved by using equation 7 for positive values of Z instead but with probabilities returned in logged form and the subtraction done with `logspace_sub`. This prevents the small ϵs from disappearing in the subtraction.

Then a stable reverse mode similar to the one for `pnorm4` was made, and this was wrapped in the function `censored_bounds`, checked for accuracy the same way and again made available for everyone to use in `pnorm4.hpp`.

B Functional Principal Component Analysis

Functional Data Analysis (FDA) is a branch of statistics that deals with data that takes values in an infinite dimensional or functional space. Growth and maturity curves are one-dimensional examples of functional data. Several classical statistical techniques have been extended to work with functional data, Principal Component Analysis (PCA) being among them. In Principal Component Analysis (PCA) the goal is to perform dimension reduction while maximizing the variation explained by each dimension, Functional Principal Component Analysis (FPCA) extends this to functional data. Let X be a random function defined on the function grid $[0, T]$ with unknown smooth mean function

$$E[X(t)] = \mu(t) \quad t \in [0, T], \tag{8}$$

and covariance function

$$\text{Cov}(X(s), X(t)) = G(s, t) \quad s, t \in [0, T]. \tag{9}$$

$G(s, t)$ can be written in it's orthogonal expansion as

$$G(s, t) = \sum_{k=1}^{\infty} \lambda_k \phi_k(s) \phi_k(t) \tag{10}$$

where λ_k are the eigenvalues, and ϕ_k are eigenfunctions that form an othonormal basis with a unit norm. Using the orthogonal expansion allows re-writing each functional observation $X_i(t)$ in the

Karhunen-Loève representation

$$X_i(t) = \mu(t) + \sum_{k=1}^{\infty} \xi_{ik}\phi_k(t) \quad \text{where} \quad \xi_{ik} = \int (X_i(t) - \mu(t))\phi_k(t)dt. \tag{11}$$

The ξ_{ik} are the Functional Principal Components (FPC)[1]. Like PCA Principal Components, FPCs can also be clustered using clustering algorithms to find groups in the data.

The Karhunen-Loève representation enables a few different things. When using estimated forms of the mean function, FPCs, and eigenfunctions which can be found a number of ways such as numerical integration, using a limited number of FPCs that explain most of the variation in the data can be used to generate noise reduced versions of the data while keeping the parts of the process that explain the most variability[4]. If L is the number of FPC that explain for example 90% of the variation in the data, then noise reduced observations can be made by

$$\hat{X}_i(t) = \hat{\mu}(t) + \sum_{k=1}^{L} \hat{\xi}_{ik}\hat{\phi}_k(t). \tag{12}$$

This same method can be used to impute missing parts of the function as well[1].

References

[1] Jeng-Min Chiou et al. "A functional data approach to missing value imputation and outlier detection for traffic flow data". *Transportmetrica B: Transport Dynamics* 2.2 (Mar. 2014), pp. 106–129. ISSN: 2168-0582. DOI: 10.1080/21680566.2014.892847. URL: http://dx.doi.org/10.1080/21680566.2014.892847.

[2] Clare B. Embling et al. "Investigating fine-scale spatio-temporal predator-prey patterns in dynamic marine ecosystems: a functional data analysis approach". *Journal of Applied Ecology* 49 (2012), pp. 481–492.

[3] David A. Fournier et al. "AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models". *Optimization Methods and Software* 27.2 (Apr. 2012), pp. 233–249. ISSN: 1029-4937. DOI: 10.1080/10556788.2011.597854. URL: http://dx.doi.org/10.1080/10556788.2011.597854.

[4] James O Ramsay and Bernard W Silverman. *Applied functional data analysis: methods and case studies*. Springer, 2007.