**Analyzing Visual Field Data**

In Part I of this series on statistic in glaucoma, we detailed the use of visual fields for understanding functional vision loss in glaucoma patients. Before discussing a new method for modeling visual field data that accounts for the anatomy of the eye, we discussed how visual field data is typically analyzed by introducing a common diagnostic metric, point-wise linear regression (PLR). PLR is a trend-based diagnostic that uses slope p-values from the location specific linear regressions to discriminate progression status. The motivation for PLR is straightforward, assuming that large negative slopes at numerous visual field locations is indicative of progression. This is characteristic of a large class of methods for analyzing visual field data that attempt to discriminate progression based on changes in the DLS across time. This technique is simple, intuitive, and effective; however, it is often limited due to the naivete of modeling assumptions, including the independence of visual field locations.

**Ocular Anatomy in the Neighborhood Structure of the Visual Field**

To properly account for the spatial dependencies on the visual field, Berchuck et al. 2018 introduce a neighborhood model that incorporates anatomical information through a dissimilarity metric. Details of the method can be found in Berchuck et al. 2018, but we provide a quick introduction. The key development is the specification of the neighborhood structure through a new definition of adjacency weights. Typically in areal data, the adjacency for two locations \(i\) and \(j\) is defined as \(w\_{ij} = 1(i \sim j)\), where \(i \sim j\) is the event that locations \(i\) and \(j\) are neighbors. As discussed in Part I, this assumption is not sufficient due to the complex anatomy of the eye. To account for this additional structure, a more general adjacency is introduced that is a function of a dissimilarity metric, \(w\_{ij}(\alpha\_t) = 1(i \sim j)\exp\{-z\_{ij}\alpha\_t\}\). Here, \(z\_{ij}\) is a dissimilarity metric that represents the absolute difference between the Garway-Heath angles of locations \(i\) and \(j\).

The parameter \(\alpha\_t\) dictates the importance of the dissimilarity metric at each visual field exam \(t\). When \(\alpha\_t\) becomes large, the model reduces to an independent process, and as \(\alpha\_t\) goes to zero, the process becomes the standard spatial model for areal data. Based on the specification of the adjacency weights, \(\alpha\_t\) has a useful interpretation with respect to deterioration of visual ability. In particular, \(\alpha\_t\) changing over exams indicates that the neighborhood structure on the visual field is changing, which in turn implies damage to the underlying retinal ganglion cell structure. This observation motivates a diagnostic of progression that quantifies variability in \(\alpha\_t\) across time. We choose the coefficient of variation (CV) and demonstrate that is a highly significant predictor of progression, and furthermore, independent of trend-based methods such as PLR.

**Navigating the womblR Package**

To make the method available to clinicians, the R package womblR was developed. The package provides a suite of functions that walk a user through the full process of analyzing a series of visual fields from beginning to end. The user interface was modeled after other impactful R packages for Bayesian spatial analysis, including spBayes and CARBayes. The package name combines Hadley’s naming convention for R packages (i.e., ending a package with the letter R) with the name of the author of the seminal paper on boundary detection, originally referred to areal wombling (Womble 1951).

We will now walk through the process of analyzing visual field data, estimating the \(\alpha\_t\) parameters, and assessing progression status. The main function in womblR is the Spatiotemporal Boundary Detection with Dissimilarity Metric model function (STBDwDM). Inference for the method is obtained through Markov chain Monte Carlo (MCMC), which is a computationally intensive method that iterates between updating individual model parameters until enough posterior samples have been collected post-convergence for making accurate posterior inference. Because of the iterative nature of MCMC, the majority of computation is performed within a for loop, so the package is built on C++ through the packages Rcpp and RcppArmadillo. Because of the increased complexity of writing in C++, the pre- and post-processing of the model are done in R with the for loop implemented in C++. The MCMC method employed in womblR is a Metropolis-Hastings within Gibbs algorithm.

Just as a quick aside, with the more recent advent of probabilistic programming, this model could have been implemented using the Hamiltonian Monte Carlo methods used in software like Stan or PyMC3. These programs do not require the derivation of full conditionals, and push the MCMC algorithm to the background. There is undoubtedly a huge market for this type of software, and it is clearly playing a significant role in the popularization of Bayesian modeling. At the same time, implementing MCMC samplers using Rcpp with traditional MCMC algorithms can be instructive, and for those with experience, nearly as quick of a coding experience.

We now begin by formatting the visual field data for analysis. According to the manual, the observed data Y must first be ordered spatially and then temporally. Furthermore, we will remove all locations that correspond to the natural blind spot (which, in the Humphrey Field Analyzer-II, correspond to locations 26 and 35).

###Load package

library(womblR)

###Format data

blind\_spot <- c(26, 35) # define blind spot

VFSeries <- VFSeries[order(VFSeries$Location), ] # sort by location

VFSeries <- VFSeries[order(VFSeries$Visit), ] # sort by visit

VFSeries <- VFSeries[!VFSeries$Location %in% blind\_spot, ] # remove blind spot locations

Y <- VFSeries$DLS # define observed outcome data

Now that we have assigned the observed outcomes to Y, we move onto the temporal variable Time. For visual field data, we define this to be the time from the baseline visit. We obtain the unique days from the baseline visit and scale them to be on the year scale.

Time <- unique(VFSeries$Time) / 365 # years since baseline visit

print(Time)

## [1] 0.0000000 0.3452055 0.6520548 1.1123288 1.3808219 1.6109589 2.0712329

## [8] 2.3780822 2.5698630

Next, we assign the adjacency matrix and dissimilarity metric (both discussed in Part I).

W <- HFAII\_Queen[-blind\_spot, -blind\_spot] # visual field adjacency matrix

DM <- GarwayHeath[-blind\_spot] # Garway-Heath angles

Now that we have specified the data objects Y, DM, W, and Time, we will customize the objects that characterize Bayesian MCMC methods, in particular, hyperparameters, starting values, Metropolis tuning values, and MCMC inputs. These objects have been detailed previously in the womblR package vignette, so we will not spend time going over their definitions. We will only note that they are each list objects similar to the spBayes package. We begin by specifying the hyperparameters.

###Bounds for temporal tuning parameter phi

TimeDist <- abs(outer(Time, Time, "-"))

TimeDistVec <- TimeDist[lower.tri(TimeDist)]

minDiff <- min(TimeDistVec)

maxDiff <- max(TimeDistVec)

PhiUpper <- -log(0.01) / minDiff # shortest diff goes down to 1%

PhiLower <- -log(0.95) / maxDiff # longest diff goes up to 95%

###Hyperparameter object

Hypers <- list(Delta = list(MuDelta = c(3, 0, 0), OmegaDelta = diag(c(1000, 1000, 1))),

T = list(Xi = 4, Psi = diag(3)),

Phi = list(APhi = PhiLower, BPhi = PhiUpper))

Then we specify the starting values for the parameters, Metropolis tuning variances, and MCMC details.

###Starting values

Starting <- list(Delta = c(3, 0, 0), T = diag(3), Phi = 0.5)

###Metropolis tuning variances

Nu <- length(Time) # calculate number of visits

Tuning <- list(Theta2 = rep(1, Nu), Theta3 = rep(1, Nu), Phi = 1)

###MCMC inputs

MCMC <- list(NBurn = 10000, NSims = 250000, NThin = 25, NPilot = 20)

We specify that our model will run for a burn-in period of 10,000 scans, followed by 250,000 scans post burn-in. In the burn-in period there will be 20 iterations of pilot adaptation evenly spaced out over the period. The final number of samples to be used for inference will be thinned down to 10,000 based on the thinning number of 25. We can now run the MCMC sampler. Details of the various options available in the sampler can be found in the documentation, help(STBDwDM).

reg.STBDwDM <- STBDwDM(Y = Y, DM = DM, W = W, Time = Time,

Starting = Starting, Hypers = Hypers, Tuning = Tuning, MCMC = MCMC,

Family = "tobit",

TemporalStructure = "exponential",

Distance = "circumference",

Weights = "continuous",

Rho = 0.99,

ScaleY = 10,

ScaleDM = 100,

Seed = 54)

## Burn-in progress: |\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|

## Sampler progress: 0%.. 10%.. 20%.. 30%.. 40%.. 50%.. 60%.. 70%.. 80%.. 90%.. 100%..

We quickly assess convergence by checking the traceplots of \(\alpha\_t\) (note that further MCMC convergence diagnostics should be used in practice).

###Load coda package

library(coda)

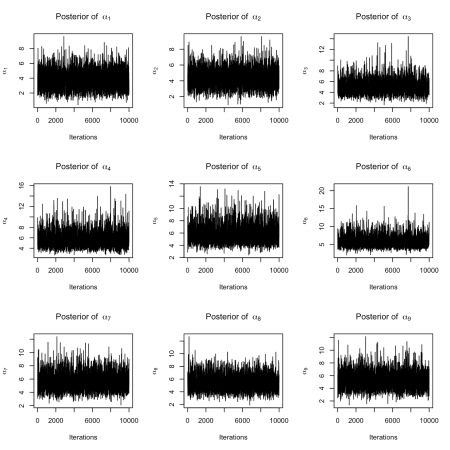
###Convert alpha to an MCMC object

Alpha <- as.mcmc(reg.STBDwDM$alpha)

###Create traceplot

par(mfrow = c(3, 3))

for (t in 1:Nu) traceplot(Alpha[, t], ylab = bquote(alpha[.(t)]), main = bquote(paste("Posterior of " ~ alpha[.(t)])))



**Converting MCMC Samples into Clinical Statements**

Now we calculate the posterior distribution of the CV of \(\alpha\_t\) and print its moments.

CVAlpha <- apply(Alpha, 1, function(x) sd(x) / mean(x))

plot(density(CVAlpha, adjust = 2), main = expression("Posterior of CV"~(alpha[t])), xlab = expression("CV"~(alpha[t])))



STCV <- c(mean(CVAlpha), sd(CVAlpha), quantile(CVAlpha, probs = c(0.025, 0.975)))

names(STCV)[1:2] <- c("Mean", "SD")

print(STCV)

## Mean SD 2.5% 97.5%

## 0.19121622 0.10205826 0.04636219 0.42744656

For this information to be useful clinically, we convert it into a probability of progression based on a model trained on a large cohort of glaucoma patients (Berchuck et al. 2019). Because the information from \(\alpha\_t\) is independent of trend-based methods, we show that the optimal use of \(\alpha\_t\) is combining it with a basic global metric that includes the slope and p-value (and their interaction) of the overall mean at each visual field exam. The trained model coefficients are publicly available and are used below. Furthermore, both the mean, standard deviation, and their interaction of the CV of \(\alpha\_t\) are included. The probability of progression can be calculated as follows.

###Calculate the global metric slope and p-value

MeanSens <- apply(t(matrix(VFSeries$DLS, ncol = Nu)) / 10, 1, mean) # scaled mean DLS

reg.global <- lm(MeanSens ~ Time) # global regression

GlobalS <- summary(reg.global)$coef[2, 1] # global slope

GlobalP <- summary(reg.global)$coef[2, 4] # global p-value

###Obtain probabiltiy of progression using estimated parameters from Berchuck et al. 2019

input <- c(1, GlobalP, GlobalS, STCV[1], STCV[2], GlobalS \* GlobalP, STCV[1] \* STCV[2])

coef <- c(-1.7471655, -0.2502131, -13.7317622, 7.4746348, -8.9152523, 18.6964153, -13.3706058)

fit <- input %\*% coef

exp(fit) / (1 + exp(fit))

## [,1]

## [1,] 0.4355997

The probability of progression is calculated to be 0.44, which can be compared to the threshold cutoff for the trained model of 0.325. This cutoff for the probability of progression was determined using operating characteristics, so that the specificity was forced to be in the clinically meaningful range of 85%. Based on this derived threshold, the probability of progression is high enough to indicate that this patient’s disease shows evidence of visual field progression (which is reassuring, because we know this patient has progression as determined by clinicians).

Looking ahead: The third installment will wrap up the discussion on the womblR package and ponder future directions for the role of statistics in glaucoma research. Furthermore, the role of open-source software in medicine will be discussed.