Appendix 2: Technical Appendix for "Effect Size, Statistical Power and Sample Size Requirements for the Bootstrap Likelihood Ratio Test in Latent Class Analysis"

John J. Dziak, Stephanie M. Lanza, and Xianming Tan

Contents

In this appendix we provide additional simulation results and supporting SAS and R code for the manuscript "Effect Size, Statistical Power and Sample Size Requirements for the Bootstrap Likelihood Ratio Test in Latent Class Analysis."

1 Effect Size Measures

Figure A1 shows the relationship of several possible measures of effect size, to simulated power. Clearly, w and KL were good predictors of power, while I_D, SEP, and entropy were not.

2 Effect Sizes Macro

In order to use the effect size measures w and KL, it is important to have a convenient way to calculate them. The method of calculation is described in Appendix 1. The code can be downloaded as a text file at `http://methodology.psu.edu/LcaEffectSizes/`.

2.1 Example of Use

Sample code for using this macro is as follows.

```
%INCLUDE "C:\My Documents\FindEffectSizes.sas";

DATA gammas;
    INPUT gammas;
    DATALINES;
    .40
    .30
    .30
; /* There will be three classes. */
 RUN;
DATA rhos;
    INPUT rho1 rho2 rho3;
  /* There are three variables because there are three classes.  If
         this dataset is not read in correctly, then the macro will not
         run correctly.  */
    DATALINES;
   .20 .30 .40
   .40 .50 .60
   .20 .20 .90
   .20 .90 .20
   .90 .20 .20
;  /* There should be as many columns here as there were
    rows in the list of gammas. */
 RUN;
%FindEffectSizes(gammas=gammas,rhos=rhos,nstarts=200);
```

The dataset provided for "gammas" contains the proposed class sizes under H_1. The dataset provided for "rhos" contains the proposed item-specific response probabilities under H_0, with rows

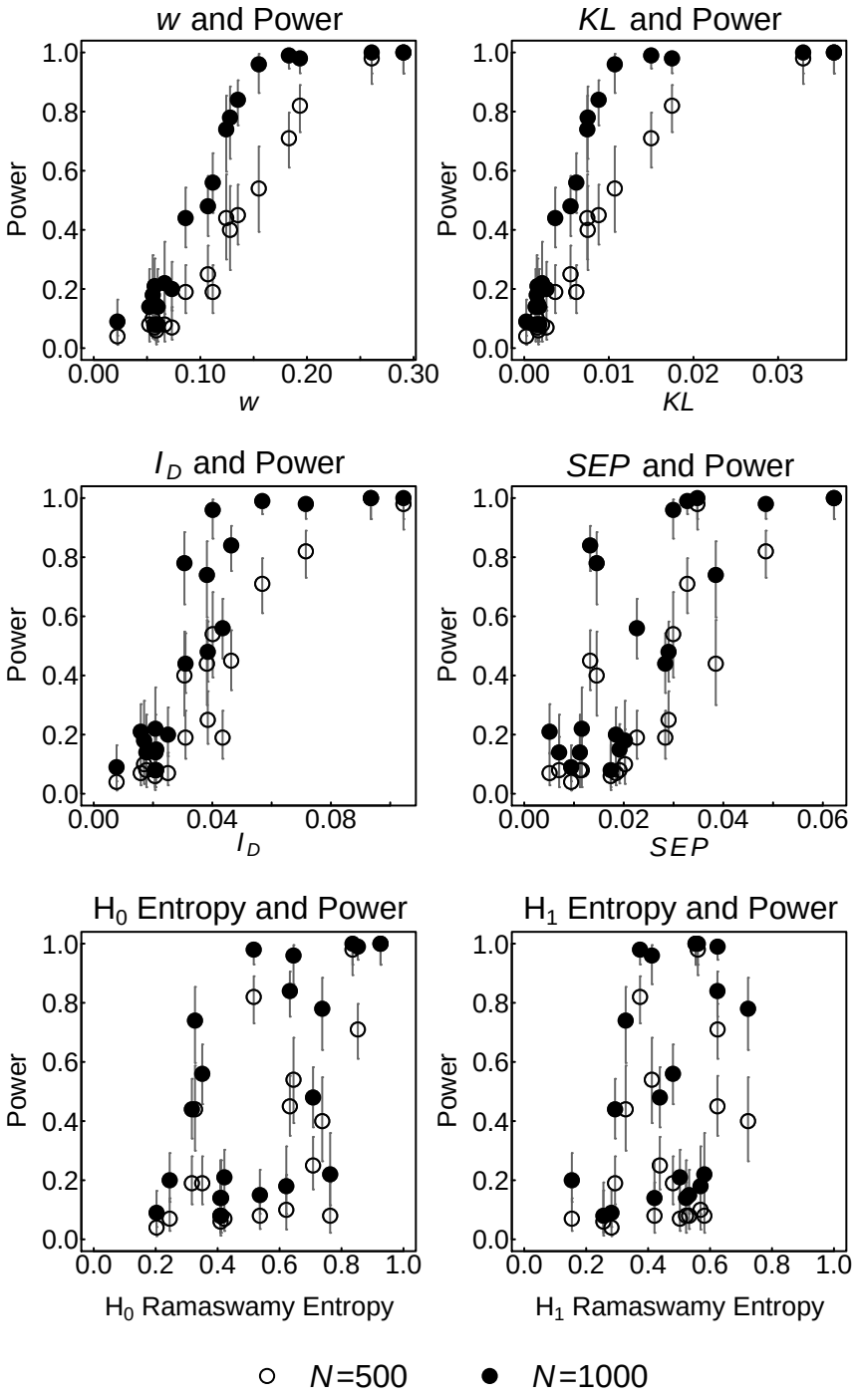

Figure 1: Figure A1: Simulated power as a function of various candidate effect size measures in the case of 5 items and 3 classes. Each point is an average of simulated datasets for a given random parameter set and (empty dots for 500, solid dots for 1000). The short vertical line segments associated with the points are confidence intervals for power for each simulation, calculated using the binom.test function in R. "Ramaswamy Entropy" refers to the population version of the entropy statistic of Ramaswamy, DeSarbo, Reibstein & Robinson (1993) and Muthén (2004).

representing items and columns representing classes. The items are assumed all to be dichotomous, and the rhos provided are the hypothesized probability of a "yes" for each item.

The resulting SAS output looks like this:

Obs	J	K	W	KL	H0Rescaled Entropy	H1Rescaled Entropy
1	5	3	0.32777	0.054069	0.82100	0.57977

The "Obs" column is just added as a default by SAS. J and K tell the assumed number of items and classes. W and KL represent the w and KL effect sizes. The population version of the entropy statistic (Ramaswamy, DeSarbo, Reibstein & Robinson, 1993; Muthén, 2004) is also given for both the H_0 and H_1 model.

3 Implementation of Simulation Experiment 1

A macro to simulate dichotomous data from a latent class analysis (LCA) model is given at `http://methodology.psu.edu/downloads/sassimulate`. A macro to perform the bootstrap likelihood ratio test (BLRT) for dichotomous LCA data is given at `http://methodology.psu.edu/downloads/sasbootstrap`. Both macros require SAS 9.1 or above for Windows and also the freely downloadable PROC LCA (Lanza, Lemmon, Dziak, Huang, Schafer, & Collins 2010), available at `http://methodology.psu.edu/downloads/proclcalta`. However, in our simulations we used essentially equivalent R code which we were able to run very quickly and in parallel on a Linux cluster, in order to complete the large number of conditions required.

The random parameters for the scenarios (i.e., the "points" in the figures) were generated using the following code.

```
n.scenarios.per.size <- 20;
J.values <- c(5,9,13);
K.values <- c(3,4,5);
for (J.index in 1:length(J.values)) {
  for (K.index in 1:length(K.values)) {
    for (scenario.index in 1:n.scenarios.per.size) {
       n.items <- J.values[J.index];
       n.classes.true <- K.values[K.index];
       ok.value <- FALSE;
       while (!ok.value) {
           temp <- rexp(n.classes.true);
           true.gamma <- temp/sum(temp);
           if (min(true.gamma)>.05) {ok.value <- TRUE;}
        }
       true.rho.yes <- matrix(.05+.90*runif(n.items*n.classes.true),
                              n.items,n.classes.true);
       # The items are assumed to be dichotomous, so we only need the
       # rho (i.e. response probability) for a yes (since the rho for
       # a no is 1 minus the other.)
       write.table(true.gamma,
              file=paste(paste(path,"true-gamma",sep=""),
```

```
              n.classes.true,
              paste(ifelse(n.items<10,"0",""),
              n.items,sep=""),
       paste(ifelse(scenario.index<10,"0",""),scenario.index,".txt",sep=""),
              sep="-"), row.names=FALSE, col.names=FALSE);
       write.table(true.rho.yes,
                   file=paste(paste(path,"true-rho",sep=""),
                          n.classes.true,
                          paste(ifelse(n.items<10,"0",""),n.items,sep=""),
     paste(ifelse(scenario.index<10,"0",""),scenario.index,".txt",sep=""),
                              sep="-"),
                   row.names=FALSE,
                   col.names=FALSE);
    }
  }
}
```

For each given scenario S, with sample size N (500 or 1000), number of classes K and number of items J, we loaded the parameters for the scenario and then ran the following R code. This was done using many separate R files submitted to Linux clusters, to run in parallel as much as possible.

```
lca_blrt = tan_lca_blrt_simul(   true_k=K,
                  nrc=rep(2,J),
                  H1_gamma = array(true.gamma,c(K,1)),
                  H1_rho=array(c(as.vector(true.rho.yes),
                                 1-as.vector(true.rho.yes)),c(J,K,2,1)),
                   n_size=N,
                  repl= 1:n.samples,
                   num_seeds0 = 50,
                  num_seeds1 = 50,
                  numboot = 100,
                  nstartboot = 50,
                   appd = paste(paste("bootdata",J,K,S,sep="-"),".dat",sep="")    )

tan_lca_blrt_simul = function(
    true_k=3, nrc=c(2,2,2,2,2),
    H1_gamma = my_gamma , H1_rho=my_rho,
    n_size=500, repl=(1:10),
    num_seeds0 = 50, num_seeds1 = 50,
    numboot = 200, nstartboot = 50,
    appd = "_examp_1_01.dat"  # added 2010-04-09
    )
{
  nc = true_k; nci = length(nrc); nstr = 1; nrcmax=max(nrc);
  true_gam = H1_gamma;   # true gam parameters, i.e., H1 parameter
  true_rho = H1_rho;  # true rho parameters, i.e., H1 parameter
  ## this part set up the structure for null parameters, the values have no meanings
```

```
null_gam =  array( true_gam[-1, ]/( sum(true_gam[-1, ]) ), c(nc-1, nstr) )
null_rho =  array(true_rho[ ,-1 , ,], c(nci, nc-1, nrcmax, nstr) );
allEstH0 = NULL;
allEstH1 = NULL;
allBootslrt = NULL;
allRepsum = NULL;
allIdentify = NULL; # H0 identify, H1 identify
## main loops
h1_lca_pat = tan_lca_sample_0(nrc_par=nrc, rho_par=true_rho, gam_par=true_gam);
true_rep = 0;
for (thisrep in repl )
{
  print(thisrep);
   true_rep=true_rep+1;
  #  step 1: generate sample under H1;
  lca_sample =  tan_lca_sample_1(n=n_size, seed_par=thisrep*10, pat_a_prop=h1_lca_pat ) ;
  #  step 2: fit H0 model given LCA sample,
  fit_H0 = proc_lca(nclass = nc-1,
                    items =  lca_sample[,1:length(nrc)],  # name of variables, y
                    categories = nrc,
                    gam_init = null_gam,    # initial gamma input
                    rho_init = null_rho,    # initial rho input
                    id = NULL,         # we don't need id here
                    groups = NULL,       # for now, we assume single group
                    groupnames = NULL,      # single group, groupnames not needed
                    measurement = NULL,      # single group, no need to use this option
                    covariates = NULL,          # for now, no covariate included
                    reference = 1,      # specify latent class as reference
                    binary  = NULL,       # by default,  bBinary = 0, binary = 1
                    stablize = c(0.0, 0.0, 0.0),    # priors for (beta, gamma, rho)
                    freq = lca_sample[,1+length(nrc)],
                    estimation = NULL,
                    SEED = 5*thisrep,
                    nstarts = num_seeds0,
                    maxiter = 5000,
                    criterion = 0.000001  );
h0_est_gam = array( fit_H0$new_gamma, c(nc-1, nstr) );
h0_est_rho = array( fit_H0$new_rho, c(nci, nc-1, nrcmax, nstr) );
#  step 3: fit H1 model given LCA sample,
fit_H1 = proc_lca( nclass = nc,
    items =  lca_sample[,1:length(nrc)],  # name of variables, y
    categories = nrc,
    gam_init = true_gam,    # inital gamma input
    rho_init = true_rho,    # inital gamma input
    id = NULL,         # we don't need id here
    groups = NULL,       # for now, we assume single group
    groupnames = NULL,      # single group, groupnames not needed
    measurement = NULL,      # single group, no need to use this option
```

```
    covariates = NULL,          # for now, no covariate included
    reference = 1,      # specify latent class as reference
    binary  = NULL,       # by default,  bBinary = 0, binary = 1
    stablize = c(0.0, 0.0, 0.0),    # priors for (beta, gamma, rho)
    freq = lca_sample[,1+length(nrc)],  #
    estimation = NULL,         # EM is the only available method
    SEED = 5*thisrep,
    nstarts = num_seeds1,
    maxiter = 5000,
    criterion = 0.000001
 );
 #  step 4: parametric bootstrap
   lca_boot = tan_lca_boot(
       size = n_size, nboot= numboot,  nH1start = nstartboot,
       boot_rho=h0_est_rho, boot_gam= h0_est_gam, boot_nrc=nrc,
       boot_nc=nc, boot_H1_rho = true_rho, boot_H1_gam = true_gam)
   #  step 5: summarize this replication
   allEstH0 = rbind(allEstH0, c(thisrep, fit_H0$aic, fit_H0$bic,
                    fit_H0$loglik, fit_H0$new_gamma, fit_H0$new_rho) );
   allEstH1 = rbind(allEstH1, c(thisrep, fit_H1$aic, fit_H1$bic,
                    fit_H1$loglik, fit_H1$new_gamma, fit_H1$new_rho) );
   allBootslrt = rbind(allBootslrt ,cbind(thisrep, lca_boot) );
   boot_lrt = lca_boot[ ,1]; # lca_boot;
   data_lrt = 2.0*(fit_H1$loglik - fit_H0$loglik);
   p_value = sum( boot_lrt  > data_lrt)/numboot;
   c_value = quantile(boot_lrt, 0.95);
             # critical value, i.e., cut-off point
   test_gt_crit = as.numeric(1.0*( data_lrt > c_value))
   allRepsum = rbind(allRepsum,
                     cbind(thisrep, data_lrt, p_value,
                           c_value, test_gt_crit ) );
   llks = fit_H1$output_seeds_loglik;
   gams = matrix(fit_H1$output_seeds_gamma, nrow=num_seeds1);
   H1_log_discrep = abs(llks[1] - llks[2]);
   H1_gam_discrep = max(abs(sort(gams[1,]) - sort(gams[2,])));
   H1_identy = c( 1.0*( (H1_log_discrep<0.1) & (H1_gam_discrep<0.1) ) ,
               1.0*(H1_log_discrep<0.1) , H1_log_discrep, H1_gam_discrep);
   llks = fit_H0$output_seeds_loglik;
   gams = matrix(fit_H0$output_seeds_gamma, nrow=num_seeds0);
   H0_log_discrep = abs(llks[1] - llks[2]);
   H0_gam_discrep = max(abs(sort(gams[1,]) - sort(gams[2,])));
   H0_identy = c( 1.0*( (H0_log_discrep<0.1) & (H0_gam_discrep<0.1) ) ,
       1.0*(H0_log_discrep<0.1) , H0_log_discrep, H0_gam_discrep);
   allIdentify = rbind(allIdentify, c(thisrep, H0_identy, H1_identy) )
 }  # end of R code for main loop
 list(allEstH0, allEstH1, allBootslrt, allRepsum) # output
}
...
```

```
##############################################################################
tan_lca_boot = function(size = n_size, nboot= numboot,  nH1start = nstartboot,
boot_rho=h0_est_rho, boot_gam= h0_est_gam, boot_nrc=nrc,
boot_nc=nc, boot_H1_rho = true_rho, boot_H1_gam = true_gam)
## size: sample size; nboot: number of bootstrap samples;
## nH1start: number of random starts for H0 fit or H1 fit
{
blrts = matrix(NA, nrow=nboot, ncol=3);
# main bootstrp loop
boot_lca_pat = tan_lca_sample_0(nrc_par=boot_nrc, rho_par=boot_rho, gam_par=boot_gam);
  for (thisboot in (1: nboot) )
  { # thisboot=1;
#  step 1: generate sample under H1;
boot_sample =  tan_lca_sample_1(n=size, seed_par=thisboot*10, pat_a_prop=boot_lca_pat ) ;
#  step 2: fit H0 model given LCA sample,
boot_fit_H0 = proc_lca(
    nclass = boot_nc-1,
    items =  boot_sample[,1:length(boot_nrc)],  # name of varaibles, y
    categories = boot_nrc,
    gam_init = boot_gam,# inital gamm input
    rho_init = boot_rho,# inital gamm input
    id = NULL,  # we don't need id here
    groups = NULL,  # for now, we assume single group
    groupnames = NULL,# single group, groupnames not needed
    measurement = NULL,# single group, no need to use this option
    covariates = NULL,       # for now, no covariate included
    reference = 1,# specify latent class as reference
    binary = NULL,  # by default,  bBinary = 0, binary = 1
    stablize = c(0.0, 0.0, 0.0),# priors for (beta, gamma, rho)
    freq = boot_sample[,1+length(boot_nrc)],
    estimation = NULL,
    SEED = 10*thisboot,
    nstarts = nH1start,
    maxiter = 5000,
    criterion = 0.000001
);
#  step 2: fit H0 model given LCA sample,
boot_fit_H1 = proc_lca(
    nclass = boot_nc,
    items =  boot_sample[,1:length(boot_nrc)],  # name of varaibles, y
    categories = boot_nrc,
    gam_init = boot_H1_gam,# initial gamma input
    rho_init = boot_H1_rho,# initial gamma input
    id = NULL,  # we don't need id here
    groups = NULL,  # for now, we assume single group
    groupnames = NULL,# single group, groupnames not needed
    measurement = NULL,# single group, no need to use this option
    covariates = NULL,       # for now, no covariate included
```

```
    reference = 1,# specify latent class as reference
    binary = NULL,  # by default,  bBinary = 0, binary = 1
    stablize = c(0.0, 0.0, 0.0),# priors for (beta, gamma, rho)
    freq = boot_sample[,1+length(boot_nrc)],
    estimation = NULL,
    SEED = 10*thisboot,
    nstarts = nH1start,
    maxiter = 5000,
    criterion = 0.000001
);
   blrts[thisboot,] = c(2.0*(boot_fit_H1$loglik - boot_fit_H0$loglik),
                        boot_fit_H0$loglik, boot_fit_H1$loglik)
  } # end of main boot loop
   blrts;  # output
}
```

The function "proc_lca" in the code above was a wrapper for a specially compiled Linux version of the PROC LCA computational engine. The resulting estimated power values were plotted against effect size measures for each scenario, in scatterplots shown in Figure A1.

4 Implementation of Simulation Experiment 2

The parameter values for the scenarios were generated in a similar way to Experiment 1. To implement the computational shortcut, we first ran 5 scenarios with each value of J and K, each scenario having only a very small number of bootstrap samples.

```
lca_blrt = tan_lca_blrt_simul( true_k=K,
                      nrc=rep(2,J),
                      H1_gamma = array(true.gamma,c(K,1)),
                      H1_rho=array(c(as.vector(true.rho.yes),
                                  1-as.vector(true.rho.yes)),c(J,K,2,1)),
                      n_size=N,
                      repl= 1:n.samples,
                      num_seeds0 = 50,
                      num_seeds1 = 50,
                      numboot = 10,
                      nstartboot = 50,
                      appd = paste(paste("bootdata",J,K,S,sep="-"),".dat",sep=""));
```

The mean cutoff value proposed from each of these sets of bootstraps was used to provide a cutoff value for a naïve LRT which would be expected to have about the same power in each case as the BLRT.

The naïve LRT's from a new set of simulations were then used to compute power.

```
power <- mean(lrteststats>critval);
```

Then $N \times KL$ (or the equivalent for w) was plotted against power for each value of J.

```
    xmax <- ifelse(J>9,100,50);
    these <- which((powerTable[,"J"]==J)&(powerTable[,"N"]<1000));
    x <- powerTable[these,"N"]*(powerTable[these,"KL"]);
    y <- powerTable[these,"Power"];
    classes <- powerTable[these,"K"];
    m1 <- gam(y~s(x),family=binomial    );
    newx <- seq(min(x),max(x),length=100000);
    newy.object <- predict(m1,type="response",newdata=data.frame(x=newx),se=TRUE);
    newy <- newy.object$fit;
    goal80 <- newx[min(which(newy>.80))];
    plot(x,
         y,
         type="n",
         main="",
         xlab="",
         ylab="",
         ylim=c(0,1),
         xlim=c(0,xmax) );
    title(xlab=expression(N%*%KL),line=1.05,cex=1);
    title(ylab=expression("Power"),line=1.45);
    lines(newx,newy,col=gray(.4),lwd=2);
    text(x,y,labels=classes,cex=0.95);
    text(.8*xmax,.15,paste("(J = ",J,")",sep=""),cex=1.6);
    goal80 <- newx[min(which(newy>.80))];
    goal90 <- newx[min(which(newy>.90))];
    lines(c(goal80,goal80),c(0,.8),lty="dotted");
    goal80table <- c(goal80table,goal80);
    goal90table <- c(goal90table,goal90);
```

The interpolation line comes from the gam function in the mgcv library (`http://cran.r-project.org/web/packages/mgcv/index.html`; Wood, 2003, 2004, 2006). "goal80" and "goal90" are the thresholds at which the fitted curves crossed the desired power of .80 or .90. These became the m values in Table 4.

5 Implementation of Simulation Experiment 3

The new simulations were done without the computational shortcut.

```
lca_blrt = tan_lca_blrt_simul(  true_k=K,
                                nrc=rep(2,J),
                                H1_gamma = array(true.gamma,c(K,1)),
                                H1_rho=array(c(as.vector(true.rho.yes),
                                       1-as.vector(true.rho.yes)),c(J,K,2,1)),
                                n_size=N,
                                repl= 1:n.samples,
                                num_seeds0 = 50,
                                num_seeds1 = 50,
                                numboot = 100,
```

```
nstartboot = 50,
basic_seed = seed,
appd = paste(paste("bootdata",which.model,
       which.seed,N,sep="-"),".dat",sep=""));
```

6 References

Dziak, J. J., Lanza, S. T., & Xu, S. (2011). LcaBootstrap SAS macro users' guide (version 1.1.0). University Park: The Methodology Center, Penn State. Retrieved from `http://methodology.psu.edu/downloads/sasbootstrap`.

Dziak, J. J., Lanza, S. T., & Xu, S. (2011). SimulateLcaDataset SAS macro users' guide (Version 1.1.0). Univeristy Park: The Methodology Center, Penn State. Retrieved from `http://methodology.psu.edu`.

Lanza, S. T., Lemmon, D. R., Dziak, J. J., Huang, L., Schafer, J. L., & Collins, L. M. (2010). PROC LCA & PROC LTA User's Guide Version 1.2.5. University Park: The Methodology Center, Penn State. The latest version is available at `http://methodology.psu.edu/downloads/proclcalta`.

LCABootstrap SAS Macro (Version 1.1.0) [Software]. (2011). University Park: The Methodology Center, Penn State. Retrieved from `http://methodology.psu.edu`

Muthén, B. O. (2004). *Mplus technical appendices.* Los Angeles, CA: Muthén & Muthén.

Ramaswamy, V., DeSarbo, W. S., Reibstein, D. J., & Robinson, W. T. (1993). An empirical pooling approach for estimating marketing mix elasticities with PIMS data. *Marketing Science, 12,* 103-124.

R Core Team (2012). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. Accessed at `http://www.R-project.org/`.

SimulateLcaDataset SAS Macro (Version 1.1.0) [Software]. (2011). University Park: The Methodology Center, Penn State. Retrieved from `http://methodology.psu.edu`.

Wood, S.N. (2003) Thin-plate regression splines. *Journal of the Royal Statistical Society (B), 65,* 95-114.

Wood, S.N. (2004) Stable and efficient multiple smoothing parameter estimation for generalized additive models. *Journal of the American Statistical Association, 99,* 673-686.

Wood, S.N. (2006) *Generalized Additive Models: An Introduction with R.* Chapman and Hall/CRC.