Chapter 6: Multivariate Fay-Herriot model with missing dependent variables (MMFH)

Anna-Lena Wölwer

## Description

The theoretical basis of the *Multivariate Fay-Herriot model with missing dependent variables (MMFH)* and the presented code is given in Chapter 6 of the dissertation *Model-Based Prediction and Estimation Using Incomplete Survey Data* by *Anna-Lena Wölwer*, which is available [here](https://doi.org/10.25353/ubtr-xxxx-25a6-5f2c).

This folder contains files MMFH\_gen\_dat\_m3.R and MMFH\_fitting.R, both of which contain executable functions which are illustrated in the following.

* MMFH\_gen\_dat\_m3.R contains functions for generating data according to an MMFH model
* MMFH\_fitting.R contains a function for fitting a MMFH model (parameter estimation, predictions, MSE estimates)

## Libraries

For the examples, we use the following libraries.

library(Matrix)  
library(mvtnorm)  
library(reshape2)  
library(ggplot2)  
library(sae)

Lade nötiges Paket: MASS

Lade nötiges Paket: lme4

## Overview of the R codes in this folder

### MMFH\_gen\_dat\_m3.R

source(paste0(getwd(), "/MMFH\_gen\_dat\_m3.R"))

The file comes with two functions f\_prep\_MMFH\_m3& f\_gen\_MMFH\_m3 for generating data according to a MMFH model with dependent variables (therefore the *\_m3* in the name). The code contains comments on the inputs of the functions.

Function f\_prep\_MMFH\_m3 is used to generate all quantities which are typically considered to be fixed in model-based small area simulation studies, e.g. the matrix of auxiliary information. For a model-based simulation study, we would execute this function only once. The function takes as input all parameters of a multivariate Fay-Herriot model like the fixed effects, the variance components, and the number of domains.

Function f\_gen\_MMFH\_m3 is used to generate all quantities which are typically considered random in model-based small area simulation studies like the random effects and sampling errors. For a simulation study, we would execute this function in each Monte Carlo iteration. The function takes as input all inputs and outputs of function f\_prep\_MMFH\_m3.

### MMFH\_fitting.R

source(paste0(getwd(), "/MMFH\_fitting.R"))

The file contains function f\_MMFH. The code contains comments on the inputs of the function.

Function f\_MMFH is used to fit a MMFH model to input data. This includes the estimation of the model fixed effects () and variance components () via Fisher-Scoring, either based on maximum likelihood (ML) or restricted maximum likelihood (REML). Furthermore, based on the parameter estimates, the model returns estimates of the random effects, the synthetic predictions () and the Empirical Best Predictions (EBPs). In addition, the MSE estimates are given.

Although in this example we only cover the case of dependent variables, function f\_MMFH works for an arbitrary number of dependent variables.

## Generate example data

Generate the fixed quantities of a MMFH model including randomly generated auxiliary information (documentation of required inputs is given in MMFH\_gen\_dat\_m3.R).

Set input quantities

m = 3 # total number of variables of interest, number of dependent variables  
D = 100 # total number of domains  
v\_ref = c(2,3,4) # variances of random effects of the 3 variables  
cor\_ref = c(.2,.3,.4) # correlations of random effects  
v\_rer = c(2.5,3.5,4.5) # variances of sampling errors of the 3 variables  
cor\_rer = c(.15,.25,.35) # correlations of sampling errors  
beta = list(c(1.5, 2.5), # list, for each variable: vector of fixed effects  
 c(2.3,3.3,4.3),  
 c(4.1,3.1,2.1, 2.2))  
range\_aux = c(10, 100) # range of the uniform distribution from which auxiliary information is sampled from  
perc\_mis = c(5, 5, 2) # percentage of missing domain information per variable

Generate data

d\_fix <- f\_prep\_MMFH\_m3(seed = 56)  
names(d\_fix)

[1] "m" "D" "v\_ref" "cor\_ref" "v\_rer" "cor\_rer"   
 [7] "beta" "range\_aux" "seed" "m" "V\_ed" "V\_ud"   
[13] "x"

Use f\_gen\_MMFH\_m3 to generate the model information which typically varies between Monte Carlo iterations. That is, the generation of the dependent variables. The function allows to set certain values as missing, input perc\_mis determines the number of domains for which the survey information of the three dependent variables is missing. Note that in this code the missing dependent variables are non-overlapping. That is, there is maximum one missing dependent variable per domain. This, however, can easily be changed in the code.

d\_var <- f\_gen\_MMFH\_m3( x = d\_fix$x,  
 beta = d\_fix$beta,  
 V\_ud = d\_fix$V\_ud,  
 V\_ed = d\_fix$V\_ed,  
 seed = 67,  
 verbose = TRUE )

Generated number of missing y values (non-overlapping over the 100 domains) for the three dependent variables is 5 5 0

str(d\_var)

List of 3  
 $ y\_true: num [1:100, 1:3] 220 239.4 67.8 201 53.1 ...  
 $ y\_obs : num [1:100, 1:3] 220.1 238.6 67 202.7 55.3 ...  
 $ y\_mis : num [1:100, 1:3] NA NA NA NA NA ...

y\_true are the (according to the model) true values of the dependent variables, y\_obs are the survey estimates of the dependent variables, y\_mis are the survey estimates, which we consider to be available, some of which are missing (determined by perc\_mis in f\_gen\_MMFH\_m3). The missing mechanism is *missing completely at random* (MCAR).

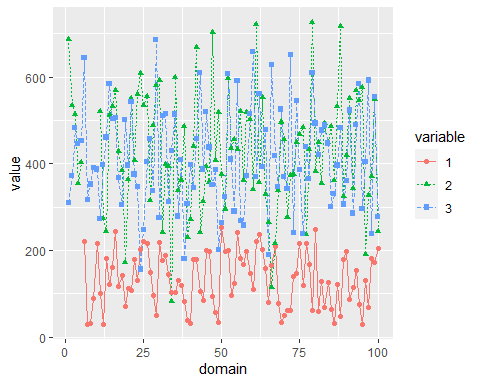
Have a look at the generated dependent variables.

d\_plot <- reshape2::melt(d\_var$y\_mis)  
d\_plot$Var2 <- factor(d\_plot$Var2)  
colnames(d\_plot) <- c("domain", "variable", "value")

ggplot2::ggplot(d\_plot,  
 aes(x = domain, y = value, group = variable,   
 color = variable, fill = variable,   
 shape = variable, lty = variable)) +   
 geom\_line() +  
 geom\_point()

Warning: Removed 5 row(s) containing missing values (geom\_path).

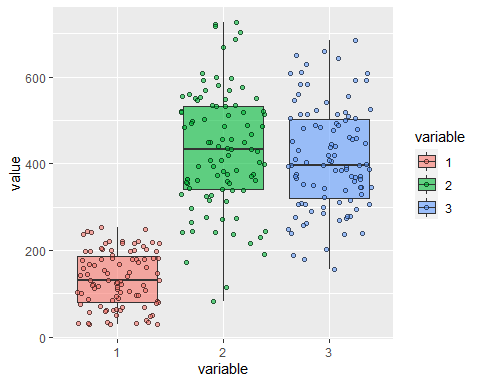
Warning: Removed 10 rows containing missing values (geom\_point).



ggplot2::ggplot(d\_plot,  
 aes(x = variable, y = value,  
 fill = variable)) +   
 geom\_boxplot(alpha = .6) +  
 geom\_jitter(pch = 21, alpha = .6)

Warning: Removed 10 rows containing non-finite values (stat\_boxplot).

Warning: Removed 10 rows containing missing values (geom\_point).



From the error message, you can see that there are (as we wanted) missing values in the dependent variables. Furthermore, you can play around with the parameters of the data generation and see how the outcomes of the sampling estimates change.

The number of domains with missing values of variables 1, 2, and 3 is

colSums(is.na(d\_var$y\_mis))

[1] 5 5 0

## Function f\_MMFH

Set some of the function inputs.

method <- "REML" # REML or ML in Fisher-Scoring  
# method <- "ML"  
verbose <- TRUE # print intermediate outputs  
eps <- 1e-8 # convergence tolerance  
maxiter <- 100 # maximum number of iterations of Fisher-Socring

Fit a MMFH model to survey estimates d\_var$y\_mis (documentation of required input in MMFH\_fitting.R).

res\_MMFH <- f\_MMFH(y = d\_var$y\_mis,  
 x = d\_fix$x,  
 V\_ed = d\_fix$V\_ed,  
 theta = NULL,   
 theta\_start = NULL,   
 method = method,   
 eps = eps,   
 maxiter = maxiter,   
 verbose = TRUE)

iter loglike lambda var\_1 var\_2 var\_3 rho\_12 rho\_13 rho\_23 v1.x1 v1.x2 v2.x1 v2.x2 v2.x3 v3.x1 v3.x2 v3.x3 v3.x4   
 0 -424.337 NA 1.977 1.855 4.218 0.016 0.020 0.003 1.267 2.502 2.346 3.297 4.299 5.046 3.084 2.107 2.191   
 1 -422.650 1 1.989 1.849 4.213 0.064 0.272 0.346 1.218 2.503 2.372 3.297 4.299 4.978 3.086 2.106 2.191   
 2 -422.641 1 2.007 1.833 4.202 0.086 0.286 0.330 1.208 2.503 2.362 3.297 4.300 4.972 3.086 2.106 2.191   
 3 -422.641 1 2.009 1.834 4.202 0.087 0.288 0.331 1.208 2.503 2.362 3.297 4.300 4.971 3.086 2.106 2.191   
 4 -422.641 1 2.009 1.834 4.202 0.087 0.288 0.331 1.208 2.503 2.362 3.297 4.300 4.971 3.086 2.106 2.191   
 5 -422.641 1 2.009 1.834 4.202 0.087 0.288 0.331 1.208 2.503 2.362 3.297 4.300 4.971 3.086 2.106 2.191   
 6 -422.641 1 2.009 1.834 4.202 0.087 0.288 0.331 1.208 2.503 2.362 3.297 4.300 4.971 3.086 2.106 2.191   
 7 -422.641 1 2.009 1.834 4.202 0.087 0.288 0.331 1.208 2.503 2.362 3.297 4.300 4.971 3.086 2.106 2.191   
 8 -422.641 1 2.009 1.834 4.202 0.087 0.288 0.331 1.208 2.503 2.362 3.297 4.300 4.971 3.086 2.106 2.191   
 9 -422.641 1 2.009 1.834 4.202 0.087 0.288 0.331 1.208 2.503 2.362 3.297 4.300 4.971 3.086 2.106 2.191   
 10 -422.641 1 2.009 1.834 4.202 0.087 0.288 0.331 1.208 2.503 2.362 3.297 4.300 4.971 3.086 2.106 2.191

With verbose = TRUE, the model returns the intermediate parameter estimates.

See the model output.

str(res\_MMFH)

List of 2  
 $ est:List of 4  
 ..$ ebp : num [1:100, 1:3] 218.5 239.9 68.2 203.5 52.1 ...  
 .. ..- attr(\*, "dimnames")=List of 2  
 .. .. ..$ : chr [1:100] "1" "2" "3" "4" ...  
 .. .. ..$ : chr [1:3] "v1" "v2" "v3"  
 ..$ ref : num [1:100, 1:3] -0.0946 -0.1153 0.6099 0.404 0.5723 ...  
 .. ..- attr(\*, "dimnames")=List of 2  
 .. .. ..$ : chr [1:100] "1" "2" "3" "4" ...  
 .. .. ..$ : chr [1:3] "v1" "v2" "v3"  
 ..$ Xbeta: num [1:100, 1:3] 218.6 240 67.6 203.1 51.5 ...  
 .. ..- attr(\*, "dimnames")=List of 2  
 .. .. ..$ : chr [1:100] "1" "2" "3" "4" ...  
 .. .. ..$ : chr [1:3] "v1" "v2" "v3"  
 ..$ fit :List of 6  
 .. ..$ method : chr "REML"  
 .. ..$ covergence: logi TRUE  
 .. ..$ iterations: num 10  
 .. ..$ estcoef :'data.frame': 9 obs. of 4 variables:  
 .. .. ..$ beta : num [1:9] 1.21 2.5 2.36 3.3 4.3 ...  
 .. .. ..$ std\_error: num [1:9] 0.4909 0.0083 0.74481 0.00879 0.00868 ...  
 .. .. ..$ tvalue : num [1:9] 2.46 301.71 3.17 375.2 495.52 ...  
 .. .. ..$ pvalue : num [1:9] 0.01388 0 0.00152 0 0 ...  
 .. ..$ refvar : Named num [1:6] 2.0092 1.8342 4.2023 0.0875 0.2878 ...  
 .. .. ..- attr(\*, "names")= chr [1:6] "var\_1" "var\_2" "var\_3" "rho\_12" ...  
 .. ..$ goodness : Named num -423  
 .. .. ..- attr(\*, "names")= chr "ll"  
 $ mse: num [1:100, 1:6] 2.08 2.14 2.51 2.22 2.5 ...  
 ..- attr(\*, "dimnames")=List of 2  
 .. ..$ : chr [1:100] "1" "2" "3" "4" ...  
 .. ..$ : chr [1:6] "v1" "cov12" "cov13" "v2" ...

The function returns the parameters estimates, EBPs, predictions of random effects, synthetic predictions, and MSE estimates.

## Compare MMFH fitting algorithm to sae::eblupFH (for univariate FH models)

We make an example to compare the MMFH output to the output of a (univariate) Fay-Herriot model using function sae::mseFH.

# Calculate (univariate) Fay-Herriot models with function mseFH from the sae package  
res\_FH <- list()  
for (k in 1:m) {  
 a\_tmp <- which(!is.na(d\_var$y\_mis[,k]))  
 res\_FH[[k]] <- sae::mseFH(d\_var$y\_mis[,k][a\_tmp] ~ -1 + d\_fix$x[[k]][a\_tmp,],  
 vardir = sapply(d\_fix$V\_ed[a\_tmp], function (d){ d[k,k] }),  
 method = method,  
 PRECISION = eps,  
 MAXITER = maxiter)  
}

For illustration, we choose variable 1.

# For variable 1: Get FH results  
k = 1  
a\_tmp <- which(!is.na(d\_var$y\_mis[,k]))  
eblup\_tmp <- rep(NA, D)  
mse\_tmp <- rep(NA, D)  
eblup\_tmp[which(!is.na(d\_var$y\_mis[,k]))] <- as.vector(res\_FH[[k]]$est$eblup)   
mse\_tmp[which(!is.na(d\_var$y\_mis[,k]))] <- as.vector(res\_FH[[k]]$mse)

Compare the EBPs:

dat\_comb <- cbind("true" = d\_var$y\_true[,k],  
 "FH\_EBLUP" = eblup\_tmp,  
 "FH\_SYN" = as.vector(d\_fix$x[[k]] %\*% res\_FH[[k]]$est$fit$estcoef[,1]),  
 "MMFH" = res\_MMFH$est$ebp[,k])  
dat\_comb[1:10,]

true FH\_EBLUP FH\_SYN MMFH  
1 220.00284 NA 218.39499 218.47039  
2 239.43961 NA 239.78456 239.87485  
3 67.76777 NA 67.70109 68.23079  
4 200.96017 NA 202.99708 203.54559  
5 53.12371 NA 51.63372 52.09918  
6 220.62982 221.45050 221.51361 221.52032  
7 29.86880 30.61783 30.96469 30.50256  
8 30.34092 29.82867 28.04025 29.77866  
9 90.08939 89.70961 89.84759 89.68889  
10 215.92293 215.14182 213.88382 215.22413

Exemplary for the first 10 domains, you can see the true values of the dependent variables in the first column. Furthermore, column 2 shows the FH EBLUPs (*FH\_EBLUP*). For domains 1 to 5, the survey estimates were considered missing. Therefore, the FH model cannot be used to calculated EBLUPs and only return synthetic predictions *FH\_SYN*. In addition, column 4 gives the EBPs of the MMFH model. With the MMFH model, we can calculate EBPs also for the domains with missing values of variable 1 as the model uses the correlations of the variable to variables 2 and 3 in a multivariate model.

Compare the MSE estimates:

cbind("FH" = mse\_tmp,  
 "MMFH" = res\_MMFH$mse[,"v1"])[1:10,]

FH MMFH  
1 NA 2.075295  
2 NA 2.139458  
3 NA 2.508396  
4 NA 2.224925  
5 NA 2.495239  
6 1.205542 1.176734  
7 1.216372 1.203678  
8 1.218619 1.403823  
9 1.184507 1.135853  
10 1.200842 1.257563

Also for the MSE, only the MMFH model can give estimates for domains 1 to 5, for which the survey direct estimates are considered missing.