Technische Dokumentation der verwendeten Algorithmen zu den indirekten Schätzungen

im Rahmen der Studie zu
Armut und sozialer Eingliederung
in den Bundesländern

Zur Berechnung der Ergebnisse dient untenstehender Code, wobei hier die Anzahl der Bootstrap-Replikationen auf 4 beschränkt wurde.

```
rm(list=ls())
gc()
set.seed(1)
require(survey)
require(nnet)
require(snow)
smhv <- read.table("smhv_johnny_bis_2011.csv",sep=";",na.st=".",header=TRUE)
smhv$AROSE_ant <- smhv$AROSE
smhv$AROSE <- as.numeric(smhv$AROSE_ant>0)
smhv$ga <- ((smhv$X_hsize-smhv$anz60plus)/smhv$X_hsize)


REP<- 2 ## Anzahl WH-pro Cluster
CLREP <- 2 ##Anzahl Cluster
bla <- c("strat",
                    "silc",
                    #"ARPT60i",
                    "year",
                    #"wr",
                    "X_htyp5a", "X_migration",
                    "X_erwintkat", "X_hv_branche", "X_hv_funktion", "X_hv_lgru3", "X_xurb",
                    "X_hv_kartab4", "X_hv_age10R", "X_hv_bfst", "X_wrecht4", "X_xwauskat","X_xhtypt")
for(i in bla){
  smhv[,i]<- as.factor(smhv[,i]) ##Umwandeln in Faktor, vielleicht manche auch in Ordered?!

}

smhv$v10X <- smhv$v10-smhv$v10_lag## Unterschied zw. Jahren v10
smhv$id <- 1:nrow(smhv)
smhv$arm <- smhv$ARPT60i*100+smhv$X_minstand*10+smhv$X_manifest## Kombinationsvariable
table(smhv$arm)
idsremove <- smhv$id[!is.na(smhv$arm)][smhv$arm[!is.na(smhv$arm)]=="110"] ## Ausprägung 110 sollte nicht vorkommen
smhv <- smhv[order(smhv$source,smhv$hid,smhv$year),]
smhv <- smhv[!smhv$id%in%idsremove,]
# Datensatz trennen
mz <- smhv[smhv$silc==0,]
silc <- smhv[smhv$silc==1,]


silc$AROSE_antg <- silc$AROSE#*silc$ga#1
```

```
d1 <- svydesign(~ id, weights=~wr, strata=~strat, data=silc)
s1 <- svyby(~AROSE_ant,~year+strat+X_xhtypt,d1,svymean)[1:4]
s2 <- svyby(~AROSE_antg,~year+strat+X_xhtypt,d1,svymean)[4]
s1[,4] <- s1[,4]/s2
colnames(s1)[4] <- "fakAROSE"
silc <- merge(silc,s1)
mz <- merge(mz,s1)
silc$ga1 <- (silc$fakAROSE)#*silc$ga
silc$AROSE_antg <- silc$AROSE*silc$ga1
mz$ga1 <-mz$fakAROSE
d1 <- svydesign(~ id, weights=~wr, strata=~strat, data=silc)
svyby(~AROSE,~year,d1,svymean)
svyby(~AROSE_ant,~year,d1,svymean)
svyby(~AROSE_antg,~year,d1,svymean)

silc$gew <- silc$wr/max(silc$wr) # Für GLM, umskalierte Gewichte ->stabiler


#Signifikante Variablen für multivariates Modell
covar0 <- c("year",
   "X_htyp5a", "X_migration",
   "X_erwintkat", "X_hv_branche", "X_hv_funktion", "X_hv_lgru3", "X_xurb",
   "X_hv_kartab4", "X_hv_age10R", "X_hv_bfst", "X_wrecht4", "X_xwauskat")
covar2 <- c(covar0,
   "v6",
   "v7",
   "v8_lag",
   "v10X"
)
form2_m <- as.formula(paste("arm~",paste(covar2,collapse="+",sep=""),sep=""))

#Signifikante Variablen für AROSE-Modell
covar2c <- c(covar0,
   "v5",
   "v6",
   "v7",
   "v8_lag",
   "v10X"
)
form2c <- as.formula(paste("AROSE~",paste(covar2c,collapse="+",sep=""),sep=""))

# Hilfsfunktion fürs Diskretisieren der Ergebnisse von multinom für Variable arm
predictit <- function(p){
  ppp <- apply(p,1,function(x)sample(c(0,10,100,111),size=1,prob=x))
  ppp
}
```

```
#dim1=Jahr
#dim2=Schätzer
#dim3=Ö+BL
#dim4=ALLE+HHtyp
#dim5=Zielvariable
#dim6=Bootstrap-Replikationen
erg <- array(NA,dim=c(7,3,10,4,4,REP))
dimnames(erg)[[1]] <- 2005:2011
dimnames(erg)[[2]] <- c("SILC","mod","mod_SILC")#Direkter Schätzer, Modellschätzer auf MZ,
Modellschätzer auf SILC
dimnames(erg)[[3]] <- c("OE",sort(unique(as.character(smhv$strat))))
dimnames(erg)[[4]] <- c("ALLE","H1","H2","H3")
dimnames(erg)[[5]] <- c("ARPT60i","minstand","manifest","AROSE")
dimnames(erg)[[6]] <- paste("R",1:REP,sep="")

### Die Parameter aus multinom werden ebenfalls abgespeichert um später auch Bootstrapfehler
davon berechnen zu können
ecoef <- array(NA,dim=c(61,3,REP))
dimnames(ecoef)[[1]]<-c("Intercept.0", "year2006", "year2007", "year2008", "year2009",
   "year2010", "year2011", "X_htyp5a5", "X_htyp5a6", "X_htyp5a7",
   "X_htyp5a8", "X_htyp5a9", "X_htyp5a10", "X_migration2", "X_migration3",
   "X_erwintkat0", "X_erwintkat1", "X_erwintkat2", "X_erwintkat3",
   "X_erwintkat4", "X_hv_branche1", "X_hv_branche2", "X_hv_branche3",
   "X_hv_branche4", "X_hv_branche5", "X_hv_branche6", "X_hv_branche7",
   "X_hv_branche8", "X_hv_funktion1", "X_hv_funktion2", "X_hv_funktion11",
   "X_hv_funktion12", "X_hv_funktion13", "X_hv_funktion21", "X_hv_funktion22",
   "X_hv_lgru32", "X_hv_lgru33", "X_xurb2", "X_xurb3", "X_hv_kartab42",
   "X_hv_kartab43", "X_hv_kartab44", "X_hv_age10R20", "X_hv_age10R30",
   "X_hv_age10R40", "X_hv_age10R50", "X_hv_age10R60", "X_hv_age10R70",
   "X_hv_bfst2", "X_hv_bfst3", "X_hv_bfst4", "X_wrecht42", "X_wrecht43",
   "X_wrecht44", "X_xwauskat2", "X_xwauskat3", "X_xwauskat4", "v6",
   "v7", "v8_lag", "v10X")
dimnames(ecoef)[[2]]<- c("10","100","111")
dimnames(ecoef)[[3]] <- paste("R",1:REP,sep="")

#Erstellung eines Clusters mit #CLREP# Sockets und exportieren der Daten auf die Knoten
cl <- makeCluster(rep("localhost",CLREP), type = "SOCK")
clusterExport(cl, list=c("predictit","REP", "silc", "mz", "form2c","form2_m", "erg","ecoef"))

##Definition der Funktion, welche auf jedem Knoten ausgeführt wird.
doParallel <- function(...){
  require(survey)
  require(nnet)
  ## Berechnung der Bootstrap-Gewichte ##SILC##
  bw <- data.frame(matrix(NA,ncol=REP,nrow=nrow(silc)))#leer erzeugen
  hid <- floor(silc$hid/100)#Bei SILC letzten 2 Stellen für SplitHH reserviert
  TFdup <- !duplicated(hid)#TRUE=1.vorkommen,FALSE=wiederholt sich
  #Bundesland X JAHR als Schicht
```

```
  bw[TFdup,] <-
data.frame(subbootweights(as.numeric(silc$strat[TFdup])+as.numeric(silc$year[TFdup])*1000,
silc$id[TFdup], replicates = REP,compress=FALSE)[[1]])
  tmp1 <- data.frame(hid=hid[!TFdup],n=1:sum(!TFdup))
  tmp2 <- data.frame(hid=hid[TFdup],bw[TFdup,])
  tmp3 <- merge(tmp1,tmp2,by="hid",all.x=TRUE,all.y=FALSE)
  bw[!TFdup,] <- tmp3[order(tmp3$n),-c(1,2)]
  names(bw) <- paste("bw",1:REP,sep="")

  ## Berechnung der Bootstrap-Gewichte ##MZ## siehe SILC
  bwmz <- data.frame(matrix(NA,ncol=REP,nrow=nrow(mz)))
  hid <- mz$hid
  TFdup <- !duplicated(hid)
  bwmz[TFdup,] <-
data.frame(subbootweights(as.numeric(mz$strat[TFdup])+as.numeric(mz$year[TFdup])*1000,
mz$id[TFdup], replicates = REP,compress=FALSE)[[1]])
  tmp1 <- data.frame(hid=hid[!TFdup],n=1:sum(!TFdup))
  tmp2 <- data.frame(hid=hid[TFdup],bwmz[TFdup,])
  tmp3 <- merge(tmp1,tmp2,by="hid",all.x=TRUE,all.y=FALSE)
  bwmz[!TFdup,] <- tmp3[order(tmp3$n),-c(1,2)]
  names(bwmz) <- paste("bw",1:REP,sep="")

  ##Testen ob Merkmalskombinationen nicht vorhanden sind (kommt so gut wie nie vorher)
  ##Wenn es fehlende Merkmalskombinationen gibt, wird das Set an Bootstrap-Gewichten nicht
verwendet
  xx=unlist(by(bw,list(silc$year,silc$strat,silc$X_xhtypt),function(x)which(apply(x,2,sum)==0)))


  ##Durchlaufen der Replikationen
  for(i in 1:REP){
    if(!i %in%xx){
      ## Berechnung der aktuellen Gewichte für diesen Bootstrap-Durchgang
      silc$gewX <- silc$gew*bw[,i] #für glm
      silc$wrX <- silc$wr*bw[,i]
      mz$wrX <- mz$wr*bwmz[,i]
      ####Direkte Schaetzer (2.Dim=1)
      variables <- c("ARPT60i","X_minstand","X_manifest","AROSE_ant")
      for(varind in 1:length(variables)){
        variable <- variables[varind]
        erg[,1,1,1,varind,i] <-
unlist(by(silc[,c(variable,"gewX")],silc$year,function(x)weighted.mean(x[,1],x[,2])))
        erg[,1,1,-1,varind,i] <-
unlist(by(silc[,c(variable,"gewX")],list(silc$year,silc$X_xhtypt),function(x)weighted.mean(x[,1],x[,2])))
        erg[,1,-1,1,varind,i] <-
unlist(by(silc[,c(variable,"gewX")],list(silc$year,silc$strat),function(x)weighted.mean(x[,1],x[,2])))
        erg[,1,-1,-1,varind,i] <-
unlist(by(silc[,c(variable,"gewX")],list(silc$year,silc$strat,silc$X_xhtypt),function(x)weighted.mean(x[,1],x
[,2])))
      }
```

```
    #### Ende Direkte Schätzer

    #Multivariates Modell fuer ARPT60i, X_minstand, X_manifest
    mm2 <- multinom(form2_m, data=silc, weights=gewX)
    ecoef[,,i] <- coef(mm2)

    mz$pp2 <- predictit(predict(mm2,newdata=mz,type="probs"))
    mz$arm2 <- 0
    mz$arm2[mz$pp2%in%c(100,111)] <- 1
    mz$mins2 <- 0
    mz$mins2[mz$pp2%in%c(010,111)] <- 1
    mz$mani2 <- 0
    mz$mani2[mz$pp2%in%c(111)] <- 1
    silc$pp2 <- predictit(mm2$fitted.values)
    silc$arm2 <- 0
    silc$arm2[silc$pp2%in%c(100,111)] <- 1
    silc$mins2 <- 0
    silc$mins2[silc$pp2%in%c(010,111)] <- 1
    silc$mani2 <- 0
    silc$mani2[silc$pp2%in%c(111)] <- 1

    ##Univariates Modell AROSE
    m2c <- glm(form2c,family=quasibinomial,data=silc,weights=gewX)
    mz$AROSE_2 <-
mz$ga1*unlist(lapply(predict(m2c,type="response",newdata=mz),function(x)sample(c(0,1),size=1,prob=
c(1-x,x))))
    silc$AROSE_2 <-
silc$ga1*unlist(lapply(predict(m2c,type="response",newdata=silc),function(x)sample(c(0,1),size=1,prob
=c(1-x,x))))

    #### ENDE Modelle schätzen und Predicts berechnen



    #OE-Schaetzer univariat - AROSE
    #MZ
    erg[,2,1,1,4,i] <- unlist(by(mz[,c("AROSE_2","wrX")],mz$year,function(x)weighted.mean(x[,1],x[,2])))
    erg[,2,1,-1,4,i] <-
unlist(by(mz[,c("AROSE_2","wrX")],list(mz$year,mz$X_xhtypt),function(x)weighted.mean(x[,1],x[,2])))
    #Modellschaetzer auf SILC ausgewertet
    erg[,3,1,1,4,i] <- unlist(by(silc[,c("AROSE_2","wrX")],silc$year,function(x)weighted.mean(x[,1],x[,2])))
    erg[,3,1,-1,4,i] <-
unlist(by(silc[,c("AROSE_2","wrX")],list(silc$year,silc$X_xhtypt),function(x)weighted.mean(x[,1],x[,2])))

    #OE-Schaetzer multivariat
    variables <- c("arm","mins","mani")
    for(varind in 1:length(variables)){
      variable <- variables[varind]
      variable2 <- paste(variable,"2",sep="")
```

```
      #MZ
      erg[,2,1,1,varind,i] <-
unlist(by(mz[,c(variable2,"wrX")],mz$year,function(x)weighted.mean(x[,1],x[,2])))
      erg[,2,1,-1,varind,i] <-
unlist(by(mz[,c(variable2,"wrX")],list(mz$year,mz$X_xhtypt),function(x)weighted.mean(x[,1],x[,2])))
      #Modellschaetzer auf SILC ausgewertet
      erg[,3,1,1,varind,i] <-
unlist(by(silc[,c(variable2,"wrX")],silc$year,function(x)weighted.mean(x[,1],x[,2])))
      erg[,3,1,-1,varind,i] <-
unlist(by(silc[,c(variable2,"wrX")],list(silc$year,silc$X_xhtypt),function(x)weighted.mean(x[,1],x[,2])))
    }

    #BL-SCHAETZER
    for(b in 2:length(dimnames(erg)[[3]])){
      bb <- dimnames(erg)[[3]][b] #Bundesland Namen
      ##Einschränken der Datensätze
      mzX <- mz[mz$strat==bb,]
      silcX <- silc[silc$strat==bb,]

      ##Univariates Modell pro Bundesland auswerten - AROSE
      #MZ
      erg[,2,b,1,4,i] <-
unlist(by(mzX[,c("AROSE_2","wrX")],mzX$year,function(x)weighted.mean(x[,1],x[,2])))
      erg[,2,b,-1,4,i] <-
unlist(by(mzX[,c("AROSE_2","wrX")],list(mzX$year,mzX$X_xhtypt),function(x)weighted.mean(x[,1],x[,2]
)))
      #Modellschaetzer auf SILC ausgewertet
      erg[,3,b,1,4,i] <-
unlist(by(silcX[,c("AROSE_2","wrX")],silcX$year,function(x)weighted.mean(x[,1],x[,2])))
      erg[,3,b,-1,4,i] <-
unlist(by(silcX[,c("AROSE_2","wrX")],list(silcX$year,silcX$X_xhtypt),function(x)weighted.mean(x[,1],x[,2
])))


      ##Multivariates Modell pro Bundesland auswerten
      variables <- c("arm","mins","mani")
      for(varind in 1:length(variables)){
        variable <- variables[varind]
        variable2 <- paste(variable,"2",sep="")
        #MZ
        erg[,2,b,1,varind,i] <-
unlist(by(mzX[,c(variable2,"wrX")],mzX$year,function(x)weighted.mean(x[,1],x[,2])))
        erg[,2,b,-1,varind,i] <-
unlist(by(mzX[,c(variable2,"wrX")],list(mzX$year,mzX$X_xhtypt),function(x)weighted.mean(x[,1],x[,2])))
        #Modellschaetzer auf SILC ausgewertet
        erg[,3,b,1,varind,i] <-
unlist(by(silcX[,c(variable2,"wrX")],silcX$year,function(x)weighted.mean(x[,1],x[,2])))
        erg[,3,b,-1,varind,i] <-
unlist(by(silcX[,c(variable2,"wrX")],list(silcX$year,silcX$X_xhtypt),function(x)weighted.mean(x[,1],x[,2])))
```

```
      }
     }
    }
   }
  list(erg,ecoef)
}
### Funktion doParallel auf Cluster schicken
ergP <- clusterApply(cl,1:CLREP, doParallel)
CLREP <- length(ergP)
REP <- tail(dim(ergP[[1]][[1]]),1)
###Ergebnis-Array erzeugen, das die Ergebnisse aller Knoten sammelt
erg <- array(NA,dim=c(7,3,10,4,4,REP*CLREP))
dimnames(erg)[1:5] <- dimnames(ergP[[1]][[1]])[1:5]
dimnames(erg)[[6]] <- paste("R",1:(REP*CLREP),sep="")

ecoef <- array(NA,dim=c(61,3,REP*CLREP))
dimnames(ecoef)[1:2] <- dimnames(ergP[[1]][[2]])[1:2]
dimnames(ecoef)[[3]] <- paste("R",1:(REP*CLREP),sep="")

### Listenelement von ergP an die richtigen Stellen im Ergebnis-Array erg spielen
start <- seq(1,REP*CLREP-REP+1,by=REP)
end <- seq(REP,REP*CLREP,by=REP)
for(i in 1:CLREP){
  erg[,,,,,start[i]:end[i]] <- ergP[[i]][[1]]
  ecoef[,,start[i]:end[i]] <- ergP[[i]][[2]]
}


###Bias wird berechnet...
# "mod_SILC"
# minus
# SILC (direkt)
erg[,3,,,,]<-erg[,3,,,,]-erg[,1,,,,]

#Mittelwert über die Bootstrap-Dimension, 6
ergm <- apply(erg,c(1,2,3,4,5),mean,na.rm=TRUE)

#Standard-Deviation über die Bootstrap-Dimension, 6
ergsd <- apply(erg,c(1,2,3,4,5),sd,na.rm=TRUE)


##############################################################################################
##############################################################################################
################ Ergebnisse direkt aus dem Samples
##############################################################################################
##############################################################################################
### svydesign-Objekt mit "echten" Gewichten
des <- svydesign(~id,strata=~strat,data=silc,weights=~wr)
```

```
##Modell und Predicts werden berechnet

###multivariat
###Variante mit x sind die summierten Wahrscheinlichkeiten
###Variante ohne x sind diskretisiert
mm2 <- multinom(form2_m, data=silc, weights=gew,maxit=500)
xx=mm2$fitted.values
silc$arm2x <- xx[,3]+xx[,4]
silc$mins2x <- xx[,2]+xx[,4]
silc$mani2x <- xx[,4]
silc$pp2 <- predictit(xx)
silc$arm2 <- 0
silc$arm2[silc$pp2%in%c(100,111)] <- 1
silc$mins2 <- 0
silc$mins2[silc$pp2%in%c(010,111)] <- 1
silc$mani2 <- 0
silc$mani2[silc$pp2%in%c(111)] <- 1
xx <- predict(mm2,newdata=mz,type="probs")
mz$arm2x <- xx[,3]+xx[,4]
mz$mins2x <- xx[,2]+xx[,4]
mz$mani2x <- xx[,4]
mz$pp2 <- predictit(xx)
mz$arm2 <- 0
mz$arm2[mz$pp2%in%c(100,111)] <- 1
mz$mins2 <- 0
mz$mins2[mz$pp2%in%c(010,111)] <- 1
mz$mani2 <- 0
mz$mani2[mz$pp2%in%c(111)] <- 1
###Ende multivariat

#univariat
#p am Ende der neuen Variable bedeutet "Wahrscheinlichkeit", kein p -> Diskretisiert
#p wird verwendet --> damit der Zufall hier keine Rolle spielt
m2c <- glm(form2c,family=quasibinomial,data=silc,weights=gew)
xxp <- predict(m2c,type="response",newdata=silc)
silc$AROSE_2p <- silc$ga1*xxp
silc$AROSE_2 <- silc$ga1*unlist(lapply(xxp,function(x)sample(c(0,1),size=1,prob=c(1-x,x))))
xxp <- predict(m2c,type="response",newdata=mz)
mz$AROSE_2p <- mz$ga1*xxp
mz$AROSE_2 <- mz$ga1*unlist(lapply(xxp,function(x)sample(c(0,1),size=1,prob=c(1-x,x))))
#Ende univariat

## Neue svydesign Objekte, damit die Predicts drinnen sind
des <- svydesign(~id,strata=~strat,data=silc,weights=~wr)
desmz <- svydesign(~id,strata=~strat,data=mz,weights=~wr)


##Ergebnis-Array initialisieren
sch <- array(NA,dim=c(7,2,10,4,4))
```

```
dimnames(sch)[[1]] <- 2005:2011
dimnames(sch)[[2]] <- c("SILC", "mod")
dimnames(sch)[[3]] <- c("OE",sort(unique(as.character(smhv$strat))))
dimnames(sch)[[4]] <- c("ALLE","H1","H2","H3")
dimnames(sch)[[5]] <- c("ARPT60i","minstand","manifest","AROSE")
#ohne SILC:
sd_res <- sch_mz_bc <- sch_mz <- biasEst <- sch[,-1,,,]


##Univariat für AROSE die möglichen Kombinationen berechnen
#SILC
sch[,2,1,1,4] <- svyby(~AROSE_2p,~year,design=des,svymean)[,2]
sch[,2,1,-1,4] <- svyby(~AROSE_2p,~year+X_xhtypt,design=des,svymean)[,3]
sch[,2,-1,1,4] <- svyby(~AROSE_2p,~year+strat,design=des,svymean)[,3]
sch[,2,-1,-1,4] <- svyby(~AROSE_2p,~year+strat+X_xhtypt,design=des,svymean)[,4]
#MZ
sch_mz[,1,1,4] <- svyby(~AROSE_2p,~year,design=desmz,svymean)[,2]
sch_mz[,1,-1,4] <- svyby(~AROSE_2p,~year+X_xhtypt,design=desmz,svymean)[,3]
sch_mz[,-1,1,4] <- svyby(~AROSE_2p,~year+strat,design=desmz,svymean)[,3]
sch_mz[,-1,-1,4] <- svyby(~AROSE_2p,~year+strat+X_xhtypt,design=desmz,svymean)[,4]


##SILC DIREKT
variables <- c("ARPT60i","X_minstand","X_manifest","AROSE_ant")
for(varind in 1:length(variables)){
  variable <- as.formula(paste("~",variables[varind],sep=""))
  sch[,1,1,1,varind] <- svyby(variable,~year,design=des,svymean)[,2]
  sch[,1,1,-1,varind] <- svyby(variable,~year+X_xhtypt,design=des,svymean)[,3]
  sch[,1,-1,1,varind] <- svyby(variable,~year+strat,design=des,svymean)[,3]
  sch[,1,-1,-1,varind] <- svyby(variable,~year+strat+X_xhtypt,design=des,svymean)[,4]
}


##multivariat für alle (3) Variablen berechnen
variables <- c("arm","mins","mani")
for(varind in 1:length(variables)){
  variable <- variables[varind]
  variable2 <- as.formula(paste("~",variable,"2x",sep=""))
  #SILC
  sch[,2,1,1,varind] <- svyby(variable2,~year,design=des,svymean)[,2]
  sch[,2,1,-1,varind] <- svyby(variable2,~year+X_xhtypt,design=des,svymean)[,3]
  sch[,2,-1,1,varind] <- svyby(variable2,~year+strat,design=des,svymean)[,3]
  sch[,2,-1,-1,varind] <- svyby(variable2,~year+strat+X_xhtypt,design=des,svymean)[,4]
  #MZ
  sch_mz[,1,1,varind] <- svyby(variable2,~year,design=desmz,svymean)[,2]
  sch_mz[,1,-1,varind] <- svyby(variable2,~year+X_xhtypt,design=desmz,svymean)[,3]
  sch_mz[,-1,1,varind] <- svyby(variable2,~year+strat,design=desmz,svymean)[,3]
  sch_mz[,-1,-1,varind] <- svyby(variable2,~year+strat+X_xhtypt,design=desmz,svymean)[,4]
}
```

```
###Der geschätzte Bias wird berechnet durch: Ergebnis des jeweiligen Modells auf SILC minus Ergebnis SILC-direkt

biasEst[,,,] <- sch[,2,,,]-sch[,1,,,]

dimnames(sch)
## Endergebnis Modell
sch_model <- sch_mz[,,,]

###Richtige Varianzschätzungen aus den Bootstrap-Ergebnissen holen
sd_model <- ergsd[,2,,,]

#Bias ...
bias <- biasEst

#Direkter Schätzer und Fehler
sch_dir <- sch[,1,,,]
sd_dir <- ergsd[,1,,,]


##Bilden der 3-jaehrigen Durchschnitte
##Welche Jahre sollen kombiniert werden. 1,2 und 3, 2,3 und 4...
ll <- list(1:3,
   2:4,
   3:5,
   4:6,
   5:7)
#Array leer initialisieren
sch_3j <- sch_dir*NA

#Für 2006 bis 2010 die Durchschnitte berechnen
for(i in 1:length(ll))
  sch_3j[i+1,,,] <- apply(sch_dir[ll[[i]],,,],c(2,3,4),mean)

##Direkten SILC-Schaetzer aus Bootstrap-Ergebnissen rausziehen
tmp <- erg[,1,,,,]
dir_boot <- erg[-c(1,7),1,,,,] #ohne 2005 und 2011

#Für 2006 bis 2010 die Durchschnitte berechnen
for(i in 1:length(ll))
  dir_boot[i,,,,] <- apply(tmp[ll[[i]],,,,],c(2,3,4,5),mean)
sd_3jt <- apply(dir_boot,c(1,2,3,4),sd)

#Vollen Array mit Jahren 2005 und 2011 = NA erzeugen
sd_3j <- sch_3j*NA
sd_3j[-c(1,7),,,] <- sd_3jt
```

```
## Verwendete Modelle
multimod <- mm2 #multivariat für ARPT60i, minstand und manifest
arosemod <- m2c #univariat für AROSE

###Bias- Berechnung
# für Bundesländer, HH-Typ und Indikator (neu)

##B)
bias_b <- bias
sd_dir_b <- sd_dir
sd_model_b <- sd_model
m <- apply(bias,c(2,3,4),mean)
ms <- apply(sd_dir,c(2,3,4),mean)
ms2 <- apply(sd_model,c(2,3,4),mean)
for(i in 1:7){
  bias_b[i,,,] <- m
  sd_dir_b[i,,,] <- ms
  sd_model_b[i,,,] <- ms2
}


## Zugehörige Lambdas berechnen...
lambda_b <- sd_dir_b^2/(sd_dir_b^2 + sd_model_b^2 + bias_b^2)


##Kombinierte Schätzer ausrechnen
komb_b <- lambda_b*sch_model + (1-lambda_b) * sch_dir

##Fehler der kombinierten Schaetzer unter angenommener Unabhaengigkeit berechnen
sd_komb_b <- sqrt(lambda_b^2*sd_dir^2+(1-lambda_b)^2*sd_model^2)


### Root MSE berechnen...
rmse_model <- sqrt(sd_model^2+bias_b^2)
rmse_komb_b <- sqrt(sd_komb_b^2+(lambda_b*bias_b)^2)
```