# Parameter restrictions for the sake of identification: Example 4

Paul Gustafson

September 30, 2022

```
### calculation sub-routines
source("bwd.B.R")
source("bwd.D.R")
source("updt.R")
source("limBF.R")
source("limPM.R")
```

```
### simulations per batch
NREP <- 50

### which batches to simulate?
### (versus those batches already saved)
BATCHLIST.GEN <-  NULL

for (batch in BATCHLIST.GEN) {

rslt <- matrix(NA, 3*NREP,8)
set.seed(13*batch+55)

for (mnlp in 1:NREP) {

  ### generate from M0
  flg <- T
  while (flg) {
    tmp <- rgamma(8,1); tmp <- tmp/sum(tmp)
    tmp <- array(tmp, dim=c(2,2,2))
    q <- list(c = sum(tmp[,,2]),
         xstr..c = apply(tmp[,2,], 2, sum) / apply(tmp, 3, sum),
         y..xstr.c = tmp[2,,] / apply(tmp, c(2,3), sum))
    sn <- runif(1, .5,1)
    p <- bwd.D(q,sn)
    if (!is.null(p)) {
      flg <- F
      trg <- (1-p$c)*(p$y..x.c[2,1]-p$y..x.c[1,1]) +
                 p$c*(p$y..x.c[2,2]-p$y..x.c[1,2])
    }
  }
  rslt[mnlp,] <- c(1,trg,unlist(limBF(q)$pspr),unlist(limPM(q)))

  ### generate from M1
  tmp <- rgamma(8,1); tmp <- tmp/sum(tmp)
  tmp <- array(tmp, dim=c(2,2,2))
  q <- list(c = sum(tmp[,,2]),
```

```
            xstr..c = apply(tmp[,2,], 2, sum) / apply(tmp, 3, sum),
            y..xstr.c = tmp[2,,] / apply(tmp, c(2,3), sum))
  trg <- (1-q$c)*(q$y..xstr.c[2,1]-q$y..xstr.c[1,1]) +
           q$c*(q$y..xstr.c[2,2]-q$y..xstr.c[1,2])
  rslt[NREP+mnlp,] <- c(2,trg,unlist(limBF(q)$pspr),unlist(limPM(q)))

  ### generate from M2
  flg <- T
  while (flg) {
    tmp <- rgamma(8,1); tmp <- tmp/sum(tmp)
    tmp <- array(tmp, dim=c(2,2,2))
    q <- list(c = sum(tmp[,,2]),
              xstr..c = apply(tmp[,2,], 2, sum) / apply(tmp, 3, sum),
              y..xstr.c = tmp[2,,] / apply(tmp, c(2,3), sum))
    p <- bwd.B(q)
    if (!is.null(p)) {
      flg <- F
      trg <- p$del
    }
  }
  rslt[2*NREP+mnlp,] <- c(3,trg,unlist(limBF(q)$pspr),unlist(limPM(q)))
}
saveRDS(rslt, paste("zopt",as.character(batch),".rds",sep=""))
}
```

```
### output to analyze
BATCHLIST.NLZ <- 1:32
NREP <- 0
rslt <- NULL
for (batch in BATCHLIST.NLZ) {
  rslt <- rbind(rslt, readRDS(paste("zopt",as.character(batch),".rds",sep="")))
}
NREP <- (dim(rslt)[1])/3
```

```
lpm <- rep(NA, 3*NREP)
for (i in 1:(3*NREP)) {
  ndx <- (rslt[i,3:5]>0)
  lpm[i] <- sum(  ((rslt[i,3:5])[ndx])* ((rslt[i,6:8])[ndx]) )
}

amse <- se.mc <- matrix(NA,2,2)

### Nature M0
ndx <- (rslt[,1]==1)
tmp.d <- (rslt[ndx,6]-rslt[ndx,2])^2
amse[1,1] <- mean(tmp.d)


tmp.n <- (lpm[ndx]-rslt[ndx,2])^2
amse[1,2] <- mean(tmp.n)

### Nature mix
ndx <- 1:(3*NREP)
```

```r
tmp.d <- (rslt[ndx,6]-rslt[ndx,2])^2
amse[2,1] <- mean(tmp.d)

tmp.n <- (lpm[ndx]-rslt[ndx,2])^2
amse[2,2] <- mean(tmp.n)
```

```r
### ramse
print(round(sqrt(amse),4))
```

```
##        [,1]   [,2]
## [1,] 0.0452 0.0594
## [2,] 0.0778 0.0470
```

```r
### percentage changes
100*(sqrt(amse[1,2]/amse[1,1])-1)
```

```
## [1] 31.33749
```

```r
100*(1-sqrt(amse[2,2]/amse[2,1]))
```

```
## [1] 39.53361
```

```r
### bootstrap to check stability
set.seed(13)
bs.rep <- 200
r1.bs  <- r2.bs <- rep(NA,bs.rep)
for (i in 1:bs.rep) {
  smp.bs <- sample(1:(3*NREP), size=3*NREP, replace=T)
  rslt.bs <- rslt[smp.bs,]; lpm.bs <- lpm[smp.bs]

  ### Nature M0
  ndx <- (rslt.bs[,1]==1)
  tmp.d <- (rslt.bs[ndx,6]-rslt.bs[ndx,2])^2
  tmp.n <- (lpm.bs[ndx]-rslt.bs[ndx,2])^2
  r1.bs[i] <- sqrt(mean(tmp.n)/mean(tmp.d))

  ### Nature mix
  ndx <- 1:(3*NREP)
  tmp.d <- (rslt.bs[ndx,6]-rslt.bs[ndx,2])^2
  tmp.n <- (lpm.bs[ndx]-rslt.bs[ndx,2])^2
  r2.bs[i] <- sqrt(mean(tmp.n)/mean(tmp.d))
}
```

```r
summary(r1.bs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.215   1.289   1.316   1.316   1.344   1.428
```

```r
sqrt(var(r1.bs))
```

```
## [1] 0.03965041
```

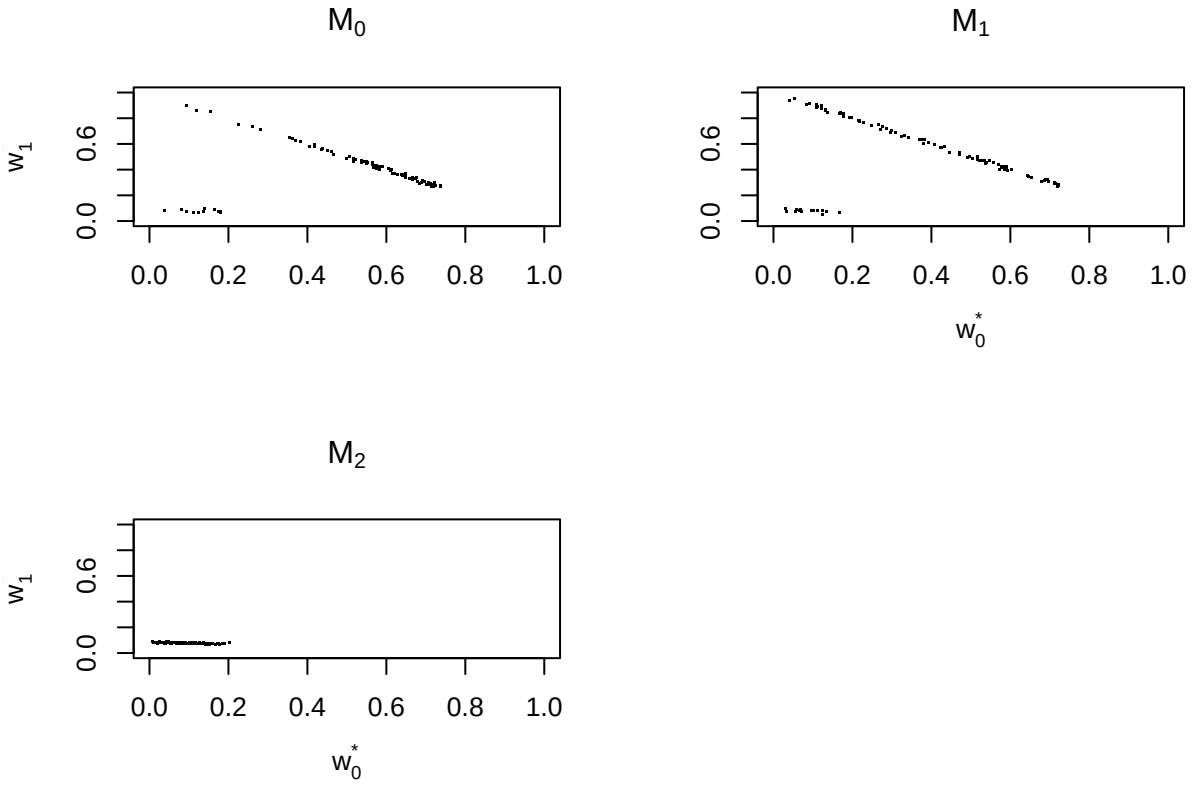
```r
summary(r2.bs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5553  0.5932  0.6064  0.6070  0.6205  0.6711
```

```r
sqrt(var(r2.bs))
```

```
## [1] 0.02142751
```

```
par(mfrow=c(2,2))
set.seed(13)
smpsz <- 100; jtr.x <- rnorm(smpsz,sd=.0075); jtr.y <- rnorm(smpsz,   sd=.0075)
ndx <- (rslt[,1]==1); ndx <- sample( (1:(3*NREP))[ndx], size=smpsz,   replace=F)
plot(rslt[ndx,3]+jtr.x,rslt[ndx,4]+jtr.y, xlim=c(0,1),ylim=c(0,1),        pch=".",
     xlab="", ylab=expression(w[1]^"*"))
title(expression(M[0]))
ndx <- (rslt[,1]==2); ndx <- sample( (1:(3*NREP))[ndx], size=smpsz,   replace=F)
plot(rslt[ndx,3]+jtr.x,rslt[ndx,4]+jtr.y, xlim=c(0,1),ylim=c(0,1),       pch=".",
   xlab=expression(w[0]^"*"),ylab="")
title(expression(M[1]))
ndx <- (rslt[,1]==3); ndx <- sample( (1:(3*NREP))[ndx], size=smpsz,     replace=F)
plot(rslt[ndx,3], rslt[ndx,4], xlim=c(0,1),ylim=c(0,1), pch=".",
     xlab=expression(w[0]^"*"), ylab=expression(w[1]^"*"))
title(expression(M[2]))
```




```
### range of limiting weight on M2 (when not zero)
summary(rslt[,5][rslt[,5]>0])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.7190  0.7940  0.8321  0.8292  0.8651  0.9172
```

```
### weight on M0 when M2 discredited and M0 true
summary(rslt[,3][(rslt[,5]==0)&(rslt[,1]==1)])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
## 0.02378 0.49321 0.60447 0.57088 0.69211 0.72918
```

```
### weight on M0 when M2 discredited and M1 true
summary(rslt[,3][(rslt[,5]==0)&(rslt[,1]==2)])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02411 0.25908 0.44455 0.42384 0.60351 0.72865
```