calculate MAPE estimators

Alex Pate

2024-12-13

### Set working directory  
rm(list=ls())  
  
### Set working directory  
setwd("/mnt/bmh01-rds/mrc-multi-outcome/Project\_8.2/")  
  
### Source functions  
R.func.sources <- list.files("R", full.names = TRUE)  
sapply(R.func.sources, source)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

## Loading required package: Hmisc

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':  
##   
## src, summarize

## The following objects are masked from 'package:base':  
##   
## format.pval, units

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

##   
## Attaching package: 'DescTools'

## The following objects are masked from 'package:Hmisc':  
##   
## %nin%, Label, Mean, Quantile

##   
## Attaching package: 'mice'

## The following object is masked from 'package:stats':  
##   
## filter

## The following objects are masked from 'package:base':  
##   
## cbind, rbind

## R/sim\_functions.R R/sim\_load\_packages.R  
## value ? character,19   
## visible FALSE FALSE

### Load functions  
source("R/sim\_functions.R")  
  
### Call libraries  
library(ggplot2)  
  
### Read in data  
metadata <- readRDS("data/heuristic\_shrinkage\_sim\_metadata.rds")  
metadata <- subset(metadata, C.pop != 0)  
metadata <- subset(metadata, cat == FALSE)  
  
### Add SE for the diff  
metadata$S.VH.diff.se <- metadata$S.VH.diff.sd/sqrt(250)  
metadata$S.boot.diff.se <- metadata$S.boot.diff.sd/sqrt(250)  
  
### Write a function to get the quantiles for a given range of C-statistics  
get\_quantiles <- function(C1,C2,correlated = TRUE){  
   
 df <- subset(metadata, C.pop > C1 & C.pop <= C2 & corr == correlated)  
 quantiles.S.VH.MCE <- quantile(as.numeric(df$S.VH.diff.se), p = c(0.5,0.9,0.99))  
 quantiles.S.boot.MCE <- quantile(as.numeric(df$S.boot.diff.se), p = c(0.5,0.9,0.99))  
   
 return(list("S.VH" = quantiles.S.VH.MCE,  
 "S.boot" = quantiles.S.boot.MCE))  
}  
  
### Write a function to get the number of observations with a given range of C-statistics  
get\_N <- function(C1,C2,correlated = TRUE){  
   
 df <- subset(metadata, C.pop > C1 & C.pop <= C2 & corr == correlated)  
 n <- nrow(df)  
 paste("N=", n)  
   
}  
  
### Define cut-offs for C-statistic  
quantiles.list <- list(c(0.6, 0.65),  
 c(0.65, 0.7),  
 c(0.7, 0.75),  
 c(0.75, 0.8),  
 c(0.8, 0.85),  
 c(0.85, 0.9))

# Get results for correlated data

### Get quantiles  
MCE.by.quantiles <- lapply(quantiles.list, function(x) {get\_quantiles(x[1],x[2])})  
MCE.all <- get\_quantiles(0,1, correlated = FALSE)  
  
### Put into a table  
MCE.by.quantiles <- lapply(MCE.by.quantiles, function(x) {  
 out <- rbind(x[[1]], x[[2]])  
 rownames(out) <- c("S\_VH", "S\_boot")  
 return(out)  
})  
MCE.all <- rbind(MCE.all[[1]], MCE.all[[2]])  
rownames(MCE.all) <- c("S\_VH", "S\_boot")  
  
### Print S\_pop by quantiles  
knitr::kable(MCE.all, digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.008 | 0.021 | 0.073 |
| S\_boot | 0.006 | 0.013 | 0.033 |

knitr::kable(MCE.by.quantiles[[1]], digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.014 | 0.053 | 0.160 |
| S\_boot | 0.012 | 0.025 | 0.142 |

knitr::kable(MCE.by.quantiles[[2]], digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.010 | 0.033 | 0.086 |
| S\_boot | 0.009 | 0.018 | 0.035 |

knitr::kable(MCE.by.quantiles[[3]], digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.008 | 0.019 | 0.034 |
| S\_boot | 0.007 | 0.013 | 0.019 |

knitr::kable(MCE.by.quantiles[[4]], digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.006 | 0.014 | 0.021 |
| S\_boot | 0.005 | 0.009 | 0.013 |

knitr::kable(MCE.by.quantiles[[5]], digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.006 | 0.012 | 0.015 |
| S\_boot | 0.005 | 0.007 | 0.009 |

knitr::kable(MCE.by.quantiles[[6]], digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.005 | 0.012 | 0.015 |
| S\_boot | 0.004 | 0.007 | 0.008 |

### Get N values  
get\_N(0,1)

## [1] "N= 14161"

lapply(quantiles.list, function(x) {get\_N(x[1],x[2])})

## [[1]]  
## [1] "N= 818"  
##   
## [[2]]  
## [1] "N= 1469"  
##   
## [[3]]  
## [1] "N= 2488"  
##   
## [[4]]  
## [1] "N= 4140"  
##   
## [[5]]  
## [1] "N= 4295"  
##   
## [[6]]  
## [1] "N= 598"

# Get results for uncorrelated data

### Get quantiles  
MCE.by.quantiles <- lapply(quantiles.list, function(x) {get\_quantiles(x[1],x[2], correlated = FALSE)})  
MCE.all <- get\_quantiles(0,1, correlated = FALSE)  
  
### Put into a table  
MCE.by.quantiles <- lapply(MCE.by.quantiles, function(x) {  
 out <- rbind(x[[1]], x[[2]])  
 rownames(out) <- c("S\_VH", "S\_boot")  
 return(out)  
})  
  
MCE.all <- rbind(MCE.all[[1]], MCE.all[[2]])  
rownames(MCE.all) <- c("S\_VH", "S\_boot")  
  
### Print MCE by quantiles  
knitr::kable(MCE.all, digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.008 | 0.021 | 0.073 |
| S\_boot | 0.006 | 0.013 | 0.033 |

knitr::kable(MCE.by.quantiles[[1]], digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.014 | 0.050 | 0.106 |
| S\_boot | 0.011 | 0.022 | 0.049 |

knitr::kable(MCE.by.quantiles[[2]], digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.010 | 0.027 | 0.051 |
| S\_boot | 0.008 | 0.014 | 0.023 |

knitr::kable(MCE.by.quantiles[[3]], digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.007 | 0.017 | 0.025 |
| S\_boot | 0.006 | 0.010 | 0.013 |

knitr::kable(MCE.by.quantiles[[4]], digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.006 | 0.013 | 0.018 |
| S\_boot | 0.005 | 0.007 | 0.009 |

knitr::kable(MCE.by.quantiles[[5]], digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.005 | 0.013 | 0.016 |
| S\_boot | 0.004 | 0.006 | 0.007 |

knitr::kable(MCE.by.quantiles[[6]], digits = 3)

|  | 50% | 90% | 99% |
| --- | --- | --- | --- |
| S\_VH | 0.007 | 0.011 | 0.013 |
| S\_boot | 0.005 | 0.005 | 0.005 |

### Get N values  
get\_N(0,1)

## [1] "N= 14161"

lapply(quantiles.list, function(x) {get\_N(x[1],x[2])})

## [[1]]  
## [1] "N= 818"  
##   
## [[2]]  
## [1] "N= 1469"  
##   
## [[3]]  
## [1] "N= 2488"  
##   
## [[4]]  
## [1] "N= 4140"  
##   
## [[5]]  
## [1] "N= 4295"  
##   
## [[6]]  
## [1] "N= 598"

# Take a look at those with really high SE

### Get quantiles  
quantile(metadata$S.VH.diff.se, p = c(0.5,0.9,0.99,1))

## 50% 90% 99% 100%   
## 0.007457534 0.019580888 0.068335056 0.562254210

quantile(metadata$S.boot.diff.se, p = c(0.5,0.9,0.99,1))

## 50% 90% 99% 100%   
## 5.811538e-03 1.265768e-02 3.281955e-02 1.356329e+02

### Take a look at those scenarios with high SE  
metadata.high.S.VH.MCE <- subset(metadata, S.VH.diff.se > 0.1)  
metadata.high.S.VH.MCE[1:30, c("C.pop", "P.meas", "P.unmeas", "nreq")]

## C.pop P.meas P.unmeas nreq  
## 162 0.6165844 3 15 132  
## 487 0.6316011 1 18 211  
## 427 0.5963263 2 6 354  
## 498 0.5725901 2 28 563  
## 368 0.5414126 3 9 1719  
## 1712 0.6000919 3 17 402  
## 4224 0.5581973 1 24 1161  
## 4623 0.5284979 1 25 4122  
## 4926 0.6763486 2 21 137  
## 3530 0.5588134 1 28 825  
## 733 0.6867150 3 10 115  
## 4431 0.6430352 2 16 145  
## 4634 0.6360370 2 22 102  
## 2180 0.5221814 2 23 4417  
## 4252 0.6292728 1 11 193  
## 956 0.6196501 1 20 277  
## 4192 0.5524239 1 27 1462  
## 2567 0.6162128 1 6 454  
## 974 0.6274347 5 28 161  
## 4272 0.5845477 1 17 794  
## 3473 0.6560089 4 22 110  
## 4773 0.6342751 2 7 124  
## 11120 0.5923986 3 24 272  
## 3991 0.6098008 3 7 221  
## 1398 0.5654102 2 19 1284  
## 2996 0.5511533 3 24 3630  
## 27107 0.5889784 1 3 379  
## 47107 0.5893509 3 10 623  
## 32130 0.6291517 3 8 155  
## 29121 0.5587197 1 4 999

metadata.high.S.boot.MCE <- subset(metadata, S.boot.diff.se > 0.1)  
metadata.high.S.boot.MCE[1:30, c("C.pop", "P.meas", "P.unmeas", "nreq")]

## C.pop P.meas P.unmeas nreq  
## 152 0.5353218 1 1 2344  
## 342 0.5843770 1 6 1087  
## 487 0.6316011 1 18 211  
## 1816 0.5426665 1 23 1654  
## 4020 0.6126617 1 5 580  
## 4224 0.5581973 1 24 1161  
## 4623 0.5284979 1 25 4122  
## 3530 0.5588134 1 28 825  
## 4849 0.5732570 1 20 856  
## 4180 0.5693344 1 1 1395  
## 4252 0.6292728 1 11 193  
## 4192 0.5524239 1 27 1462  
## 4459 0.5718597 1 25 720  
## 663 0.5599906 1 30 2301  
## 5067 0.5353600 1 11 4978  
## 3290 0.5936210 1 6 757  
## 1089 0.5987992 1 28 439  
## 4392 0.5845642 1 9 384  
## 4997 0.5687348 1 30 770  
## 2409 0.6159694 1 29 312  
## 27107 0.5889784 1 3 379  
## 40119 0.5888051 1 2 1225  
## 1608 0.5397044 1 18 1584  
## 48125 0.5558199 1 11 725  
## 4609 0.6549188 1 19 129  
## 30127 0.6292842 1 7 111  
## 39130 0.5271602 1 2 2621  
## 22153 0.5754706 1 9 2654  
## 30136 0.6140671 1 1 987  
## 49139 0.5988977 1 18 295