betaMC: Internal Tests

Ivan Jacob Agaloos Pesigan

Tests

```
#> test-betaMC-beta-mc-est-mi
#> Call:
#> BetaMC(object = mc)
#>
#> Standardized regression slopes
#> type = "mvn"
#>            est     se R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> NARTIC  0.4951 0.0734 5 0.4295 0.4301 0.4329 0.5945 0.5956 0.5959
#> PCTGRT  0.3915 0.0677 5 0.2056 0.2063 0.2092 0.3539 0.3542 0.3543
#> PCTSUPP 0.2632 0.0807 5 0.1635 0.1661 0.1775 0.3601 0.3617 0.3621
#> Call:
#> BetaMC(object = mc)
#>
#> Standardized regression slopes
#> type = "mvn"
#> Test passed
#> Call:
#> BetaMC(object = mc)
#>
#> Standardized regression slopes
#> type = "mvn"
#>           est     se R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> NARTIC 0.7622 0.0793 5 0.6755 0.6766 0.6815 0.8774 0.8837 0.8851
#> Call:
#> BetaMC(object = mc)
#>
#> Standardized regression slopes
#> type = "mvn"
#> Test passed

#> test-betaMC-beta-mc-est

#> Call:
#> BetaMC(object = mc)
#>
```

```
#> Standardized regression slopes
#> type = "mvn"
#>            est     se R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> NARTIC  0.4951 0.0701 5 0.3652 0.3657 0.3678 0.5235 0.5250 0.5254
#> PCTGRT  0.3915 0.0558 5 0.3502 0.3517 0.3585 0.4947 0.4976 0.4982
#> PCTSUPP 0.2632 0.0931 5 0.1234 0.1247 0.1305 0.3566 0.3630 0.3645
#> Call:
#> BetaMC(object = mc)
#>
#> Standardized regression slopes
#> type = "mvn"
#> Test passed
#> Call:
#> BetaMC(object = mc)
#>
#> Standardized regression slopes
#> type = "mvn"
#>           est     se R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> NARTIC 0.7622 0.0793 5 0.6755 0.6766 0.6815 0.8774 0.8837 0.8851
#> Call:
#> BetaMC(object = mc)
#>
#> Standardized regression slopes
#> type = "mvn"
#> Test passed

#> test-betaMC-delta-r-sq-mc-est-mi

#> Call:
#> DeltaRSqMC(object = mc)
#>
#> Improvement in R-squared
#> type = "mvn"
#>            est     se R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> NARTIC  0.1859 0.0750 5 0.0805 0.0812 0.0845 0.2688 0.2771 0.2789
#> PCTGRT  0.1177 0.0302 5 0.0244 0.0248 0.0267 0.0964 0.0970 0.0972
#> PCTSUPP 0.0569 0.0361 5 0.0186 0.0193 0.0221 0.1141 0.1176 0.1183
#> Call:
#> DeltaRSqMC(object = mc)
#>
#> Improvement in R-squared
#> type = "mvn"
#> Test passed
#> Test passed

#> test-betaMC-delta-r-sq-mc-est

#> Call:
```

```
#> DeltaRSqMC(object = mc)
#>
#> Improvement in R-squared
#> type = "mvn"
#>            est     se R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> NARTIC  0.1859 0.0584 5 0.0450 0.0460 0.0505 0.1960 0.2004 0.2014
#> PCTGRT  0.1177 0.0355 5 0.0870 0.0879 0.0916 0.1754 0.1758 0.1759
#> PCTSUPP 0.0569 0.0440 5 0.0106 0.0107 0.0113 0.1125 0.1185 0.1199
#> Call:
#> DeltaRSqMC(object = mc)
#>
#> Improvement in R-squared
#> type = "mvn"
#> Test passed
#> Test passed
```

#> test-betaMC-diff-beta-mc-est-mi

```
#> Call:
#> DiffBetaMC(object = mc)
#>
#> Differences of standardized regression slopes
#> type = "mvn"
#>                    est     se R   0.05%    0.5%    2.5%  97.5%  99.5% 99.95%
#> NARTIC-PCTGRT  0.1037 0.1170 5  0.0798  0.0803  0.0827 0.3467 0.3523 0.3536
#> NARTIC-PCTSUPP 0.2319 0.1419 5  0.0874  0.0876  0.0888 0.4057 0.4160 0.4183
#> PCTGRT-PCTSUPP 0.1282 0.1110 5 -0.1362 -0.1349 -0.1290 0.1481 0.1606 0.1634
#> Call:
#> DiffBetaMC(object = mc)
#>
#> Differences of standardized regression slopes
#> type = "mvn"
#> Test passed
#> Test passed
```

#> test-betaMC-diff-beta-mc-est

```
#> Call:
#> DiffBetaMC(object = mc)
#>
#> Differences of standardized regression slopes
#> type = "mvn"
#>                    est     se R   0.05%    0.5%    2.5%  97.5%  99.5% 99.95%
#> NARTIC-PCTGRT  0.1037 0.1073 5 -0.1330 -0.1319 -0.1269 0.1136 0.1157 0.1161
#> NARTIC-PCTSUPP 0.2319 0.1332 5  0.1016  0.1017  0.1021 0.3930 0.4003 0.4020
#> PCTGRT-PCTSUPP 0.1282 0.1350 5 -0.0142 -0.0107  0.0046 0.3101 0.3110 0.3112
#> Call:
```

```
#> DiffBetaMC(object = mc)
#>
#> Differences of standardized regression slopes
#> type = "mvn"
#> Test passed
#> Test passed
```

#> test-betaMC-mc-fixed-x-mi

```
#> MCMI(object = object, mi = mi, R = R, type = "mvn", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "adf", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc0", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc1", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc2", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc3", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc4", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc4m", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc5", fixed_x = TRUE)
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "adf", fixed_x = TRUE)
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc0", fixed_x = TRUE)
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc1", fixed_x = TRUE)
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc2", fixed_x = TRUE)
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc3", fixed_x = TRUE)
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc4", fixed_x = TRUE)
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc4m", fixed_x = TRUE)
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn", fixed_x = TRUE)
#> MCMI(object = object, mi = mi, R = R, type = "hc5", fixed_x = TRUE)
#> Test passed
#> Test passed
```

#> test-betaMC-mc-fixed-x

```
#> MC(object = object, R = R, type = "mvn", fixed_x = TRUE)
```

```
#> MC(object = object, R = R, type = "adf", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc0", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc1", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc2", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc3", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc4", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc4m", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc5", fixed_x = TRUE)
#> Test passed
#> MC(object = object, R = R, type = "mvn", fixed_x = TRUE)
#> MC(object = object, R = R, type = "adf", fixed_x = TRUE)
#> Test passed
#> MC(object = object, R = R, type = "mvn", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc0", fixed_x = TRUE)
#> Test passed
#> MC(object = object, R = R, type = "mvn", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc1", fixed_x = TRUE)
#> Test passed
#> MC(object = object, R = R, type = "mvn", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc2", fixed_x = TRUE)
#> Test passed
#> MC(object = object, R = R, type = "mvn", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc3", fixed_x = TRUE)
#> Test passed
#> MC(object = object, R = R, type = "mvn", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc4", fixed_x = TRUE)
#> Test passed
#> MC(object = object, R = R, type = "mvn", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc4m", fixed_x = TRUE)
#> Test passed
#> MC(object = object, R = R, type = "mvn", fixed_x = TRUE)
#> MC(object = object, R = R, type = "hc5", fixed_x = TRUE)
#> Test passed
#> Call:
#> MC(object = object, R = 5L, decomposition = "chol", fixed_x = TRUE)
#> The first set of simulated parameter estimates
#> and model-implied covariance matrix.
#>
#> $coef
#> [1] 0.4758462 0.4987145
#>
#> $sigmasq
#> [1] 0.5337949
#>
#> $vechsigmacapx
```

```
#> [1] 1.000000e+00 5.848434e-17 1.000000e+00
#>
#> $sigmacapx
#>              [,1]         [,2]
#> [1,] 1.000000e+00 5.848434e-17
#> [2,] 5.848434e-17 1.000000e+00
#>
#> $sigmaysq
#> [1] 1.008941
#>
#> $sigmayx
#> [1] 0.4758462 0.4987145
#>
#> $sigmacap
#>           [,1]         [,2]         [,3]
#> [1,] 1.0089407 4.758462e-01 4.987145e-01
#> [2,] 0.4758462 1.000000e+00 5.848434e-17
#> [3,] 0.4987145 5.848434e-17 1.000000e+00
#>
#> $pd
#> [1] TRUE
#>
#> Call:
#> MC(object = object, R = 5L, decomposition = "svd", fixed_x = TRUE)
#> The first set of simulated parameter estimates
#> and model-implied covariance matrix.
#>
#> $coef
#> [1] 0.4659576 0.5142679
#>
#> $sigmasq
#> [1] 0.5359596
#>
#> $vechsigmacapx
#> [1] 1.000000e+00 5.848434e-17 1.000000e+00
#>
#> $sigmacapx
#>              [,1]         [,2]
#> [1,] 1.000000e+00 5.848434e-17
#> [2,] 5.848434e-17 1.000000e+00
#>
#> $sigmaysq
#> [1] 1.017548
#>
#> $sigmayx
```

```
#> [1] 0.4659576 0.5142679
#>
#> $sigmacap
#>           [,1]         [,2]         [,3]
#> [1,] 1.0175476 4.659576e-01 5.142679e-01
#> [2,] 0.4659576 1.000000e+00 5.848434e-17
#> [3,] 0.5142679 5.848434e-17 1.000000e+00
#>
#> $pd
#> [1] TRUE
#>
#> Test passed
```

#> test-betaMC-mc-mi

```
#> MCMI(object = object, mi = mi, R = R, type = "mvn")
#> MCMI(object = object, mi = mi, R = R, type = "adf")
#> MCMI(object = object, mi = mi, R = R, type = "hc0")
#> MCMI(object = object, mi = mi, R = R, type = "hc1")
#> MCMI(object = object, mi = mi, R = R, type = "hc2")
#> MCMI(object = object, mi = mi, R = R, type = "hc3")
#> MCMI(object = object, mi = mi, R = R, type = "hc4")
#> MCMI(object = object, mi = mi, R = R, type = "hc4m")
#> MCMI(object = object, mi = mi, R = R, type = "hc5")
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn")
#> MCMI(object = object, mi = mi, R = R, type = "adf")
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn")
#> MCMI(object = object, mi = mi, R = R, type = "hc0")
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn")
#> MCMI(object = object, mi = mi, R = R, type = "hc1")
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn")
#> MCMI(object = object, mi = mi, R = R, type = "adf")
#> MCMI(object = object, mi = mi, R = R, type = "hc2")
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn")
#> MCMI(object = object, mi = mi, R = R, type = "hc3")
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn")
#> MCMI(object = object, mi = mi, R = R, type = "hc4")
#> Test passed
#> MCMI(object = object, mi = mi, R = R, type = "mvn")
#> MCMI(object = object, mi = mi, R = R, type = "hc4m")
#> Test passed
```

```
#> MCMI(object = object, mi = mi, R = R, type = "mvn")
#> MCMI(object = object, mi = mi, R = R, type = "hc5")
#> Test passed
```

#> test-betaMC-mc

```
#> MC(object = object, R = R, type = "mvn")
#> MC(object = object, R = R, type = "adf")
#> MC(object = object, R = R, type = "hc0")
#> MC(object = object, R = R, type = "hc1")
#> MC(object = object, R = R, type = "hc2")
#> MC(object = object, R = R, type = "hc3")
#> MC(object = object, R = R, type = "hc4")
#> MC(object = object, R = R, type = "hc4m")
#> MC(object = object, R = R, type = "hc5")
#> Test passed
#> MC(object = object, R = R, type = "mvn")
#> MC(object = object, R = R, type = "adf")
#> Test passed
#> MC(object = object, R = R, type = "mvn")
#> MC(object = object, R = R, type = "hc0")
#> Test passed
#> MC(object = object, R = R, type = "mvn")
#> MC(object = object, R = R, type = "hc1")
#> Test passed
#> MC(object = object, R = R, type = "mvn")
#> MC(object = object, R = R, type = "hc2")
#> Test passed
#> MC(object = object, R = R, type = "mvn")
#> MC(object = object, R = R, type = "hc3")
#> Test passed
#> MC(object = object, R = R, type = "mvn")
#> MC(object = object, R = R, type = "hc4")
#> Test passed
#> MC(object = object, R = R, type = "mvn")
#> MC(object = object, R = R, type = "hc4m")
#> Test passed
#> MC(object = object, R = R, type = "mvn")
#> MC(object = object, R = R, type = "hc5")
#> Test passed
#> Call:
#> MC(object = object, R = 5L, decomposition = "chol")
#> The first set of simulated parameter estimates
#> and model-implied covariance matrix.
#>
#> $coef
#> [1] 0.4688013 0.4660633
```

```
#>
#> $sigmasq
#> [1] 0.5331191
#>
#> $vechsigmacapx
#> [1]  0.9497147989 -0.0003547646  1.0075414443
#>
#> $sigmacapx
#>               [,1]          [,2]
#> [1,]  0.9497147989 -0.0003547646
#> [2,] -0.0003547646  1.0075414443
#>
#> $sigmaysq
#> [1] 0.9605405
#>
#> $sigmayx
#> [1] 0.4450622 0.4694118
#>
#> $sigmacap
#>           [,1]          [,2]          [,3]
#> [1,] 0.9605405  0.4450622080  0.4694118229
#> [2,] 0.4450622  0.9497147989 -0.0003547646
#> [3,] 0.4694118 -0.0003547646  1.0075414443
#>
#> $pd
#> [1] TRUE
#>
#> Call:
#> MC(object = object, R = 5L, decomposition = "svd")
#> The first set of simulated parameter estimates
#> and model-implied covariance matrix.
#>
#> $coef
#> [1] 0.4580301 0.4850555
#>
#> $sigmasq
#> [1] 0.5497989
#>
#> $vechsigmacapx
#> [1] 1.038296916 0.009804769 1.068440723
#>
#> $sigmacapx
#>             [,1]        [,2]
#> [1,] 1.038296916 0.009804769
#> [2,] 0.009804769 1.068440723
#>
```

```
#> $sigmaysq
#> [1] 1.023363
#>
#> $sigmayx
#> [1] 0.480327 0.522744
#>
#> $sigmacap
#>          [,1]        [,2]        [,3]
#> [1,] 1.023363 0.480327047 0.522743966
#> [2,] 0.480327 1.038296916 0.009804769
#> [3,] 0.522744 0.009804769 1.068440723
#>
#> $pd
#> [1] TRUE
#>
#> Test passed
```

```
#> test-betaMC-p-cor-mc-est-mi
```

```
#> Call:
#> PCorMC(object = mc)
#>
#> Squared partial correlations
#> type = "mvn"
#>            est     se R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> NARTIC  0.4874 0.1085 5 0.3270 0.3271 0.3274 0.5766 0.5863 0.5885
#> PCTGRT  0.3757 0.0746 5 0.1301 0.1314 0.1369 0.3121 0.3149 0.3155
#> PCTSUPP 0.2254 0.1079 5 0.0849 0.0871 0.0966 0.3698 0.3780 0.3799
#> Call:
#> PCorMC(object = mc)
#>
#> Squared partial correlations
#> type = "mvn"
#> Test passed
#> Test passed
```

```
#> test-betaMC-p-cor-mc-est
```

```
#> Call:
#> PCorMC(object = mc)
#>
#> Squared partial correlations
#> type = "mvn"
#>            est     se R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> NARTIC  0.4874 0.1162 5 0.2344 0.2354 0.2401 0.5105 0.5169 0.5184
#> PCTGRT  0.3757 0.0736 5 0.3104 0.3122 0.3203 0.4926 0.4933 0.4934
#> PCTSUPP 0.2254 0.1260 5 0.0543 0.0553 0.0598 0.3601 0.3785 0.3826
```

```
#> Call:
#> PCorMC(object = mc)
#>
#> Squared partial correlations
#> type = "mvn"
#> Test passed
#> Test passed
```

#> test-betaMC-r-sq-mc-est-mi

```
#> Call:
#> RSqMC(object = mc)
#>
#> R-squared and adjusted R-squared
#> type = "mvn"
#>        est     se R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> rsq 0.8045 0.0312 5 0.7508 0.7517 0.7556 0.8343 0.8367 0.8373
#> adj 0.7906 0.0334 5 0.7330 0.7340 0.7382 0.8224 0.8251 0.8256
#> Call:
#> RSqMC(object = mc)
#>
#> R-squared and adjusted R-squared
#> type = "mvn"
#> Test passed
#> Test passed
```

#> test-betaMC-r-sq-mc-est

```
#> Call:
#> RSqMC(object = mc)
#>
#> R-squared and adjusted R-squared
#> type = "mvn"
#>        est     se R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> rsq 0.8045 0.0346 5 0.7565 0.7574 0.7614 0.8496 0.8526 0.8533
#> adj 0.7906 0.0371 5 0.7391 0.7401 0.7444 0.8388 0.8421 0.8428
#> Call:
#> RSqMC(object = mc)
#>
#> R-squared and adjusted R-squared
#> type = "mvn"
#> Test passed
#> Test passed
```

#> test-betaMC-s-cor-mc-est-mi

```
#> Call:
#> SCorMC(object = mc)
```

```
#>
#> Semipartial correlations
#> type = "mvn"
#>            est     se R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> NARTIC  0.4312 0.0911 5 0.2837 0.2849 0.2900 0.5175 0.5261 0.5281
#> PCTGRT  0.3430 0.0642 5 0.1560 0.1572 0.1624 0.3105 0.3115 0.3118
#> PCTSUPP 0.2385 0.0752 5 0.1364 0.1381 0.1458 0.3371 0.3427 0.3440
#> Call:
#> SCorMC(object = mc)
#>
#> Semipartial correlations
#> type = "mvn"
#> Test passed
#> Test passed
```

#> test-betaMC-s-cor-mc-est

```
#> Call:
#> SCorMC(object = mc)
#>
#> Semipartial correlations
#> type = "mvn"
#>            est     se R  0.05%   0.5%   2.5%  97.5%  99.5% 99.95%
#> NARTIC  0.4312 0.0892 5 0.2120 0.2139 0.2224 0.4423 0.4476 0.4487
#> PCTGRT  0.3430 0.0497 5 0.2950 0.2963 0.3019 0.4187 0.4193 0.4194
#> PCTSUPP 0.2385 0.0944 5 0.1029 0.1035 0.1061 0.3329 0.3438 0.3462
#> Call:
#> SCorMC(object = mc)
#>
#> Semipartial correlations
#> type = "mvn"
#> Test passed
#> Test passed
```

#> test-betaMC

```
#> Test passed
```

#> test-zzz-coverage

```
#>               beta1    beta2    beta3 sigmasq   sigmax1x1  sigmax2x1  sigmax3x1
#> sigmaysq   909.1981 257.2976 276.0367       1 0.007091036 0.03637752 0.01896371
#> sigmayx1  3507.1691 471.2058 510.5430       0 0.084208291 0.21599726 0.11260003
#> sigmayx2   471.2058 333.2295 150.9121       0 0.000000000 0.08420829 0.00000000
#> sigmayx3   510.5430 150.9121 554.4386       0 0.000000000 0.00000000 0.08420829
#> sigmax1x1    0.0000   0.0000   0.0000       0 1.000000000 0.00000000 0.00000000
#> sigmax2x1    0.0000   0.0000   0.0000       0 0.000000000 1.00000000 0.00000000
#> sigmax3x1    0.0000   0.0000   0.0000       0 0.000000000 0.00000000 1.00000000
```

```
#> sigmax2x2    0.0000   0.0000   0.0000       0 0.000000000 0.00000000 0.00000000
#> sigmax3x2    0.0000   0.0000   0.0000       0 0.000000000 0.00000000 0.00000000
#> sigmax3x3    0.0000   0.0000   0.0000       0 0.000000000 0.00000000 0.00000000
#>             sigmax2x2 sigmax3x2  sigmax3x3
#> sigmaysq  0.04665482 0.0486426 0.01267877
#> sigmayx1  0.00000000 0.0000000 0.00000000
#> sigmayx2  0.21599726 0.1126000 0.00000000
#> sigmayx3  0.00000000 0.2159973 0.11260003
#> sigmax1x1 0.00000000 0.0000000 0.00000000
#> sigmax2x1 0.00000000 0.0000000 0.00000000
#> sigmax3x1 0.00000000 0.0000000 0.00000000
#> sigmax2x2 1.00000000 0.0000000 0.00000000
#> sigmax3x2 0.00000000 1.0000000 0.00000000
#> sigmax3x3 0.00000000 0.0000000 1.00000000
#>               beta1    beta2    beta3 sigmasq
#> sigmaysq   909.1981 257.2976 276.0367       1
#> sigmayx1  3507.1691 471.2058 510.5430       0
#> sigmayx2   471.2058 333.2295 150.9121       0
#> sigmayx3   510.5430 150.9121 554.4386       0
#> sigmax1x1    0.0000   0.0000   0.0000       0
#> sigmax2x1    0.0000   0.0000   0.0000       0
#> sigmax3x1    0.0000   0.0000   0.0000       0
#> sigmax2x2    0.0000   0.0000   0.0000       0
#> sigmax3x2    0.0000   0.0000   0.0000       0
#> sigmax3x3    0.0000   0.0000   0.0000       0
#>               beta1    beta2    beta3       rsq   sigmax1x1  sigmax2x1
#> sigmaysq   909.1981 257.2976 276.0367 -126.0843 0.007091036 0.03637752
#> sigmayx1  3507.1691 471.2058 510.5430    0.0000 0.084208291 0.21599726
#> sigmayx2   471.2058 333.2295 150.9121    0.0000 0.000000000 0.08420829
#> sigmayx3   510.5430 150.9121 554.4386    0.0000 0.000000000 0.00000000
#> sigmax1x1    0.0000   0.0000   0.0000    0.0000 1.000000000 0.00000000
#> sigmax2x1    0.0000   0.0000   0.0000    0.0000 0.000000000 1.00000000
#> sigmax3x1    0.0000   0.0000   0.0000    0.0000 0.000000000 0.00000000
#> sigmax2x2    0.0000   0.0000   0.0000    0.0000 0.000000000 0.00000000
#> sigmax3x2    0.0000   0.0000   0.0000    0.0000 0.000000000 0.00000000
#> sigmax3x3    0.0000   0.0000   0.0000    0.0000 0.000000000 0.00000000
#>             sigmax3x1  sigmax2x2 sigmax3x2  sigmax3x3
#> sigmaysq  0.01896371 0.04665482 0.0486426 0.01267877
#> sigmayx1  0.11260003 0.00000000 0.0000000 0.00000000
#> sigmayx2  0.00000000 0.21599726 0.1126000 0.00000000
#> sigmayx3  0.08420829 0.00000000 0.2159973 0.11260003
#> sigmax1x1 0.00000000 0.00000000 0.0000000 0.00000000
#> sigmax2x1 0.00000000 0.00000000 0.0000000 0.00000000
#> sigmax3x1 1.00000000 0.00000000 0.0000000 0.00000000
#> sigmax2x2 0.00000000 1.00000000 0.0000000 0.00000000
#> sigmax3x2 0.00000000 0.00000000 1.0000000 0.00000000
```

```
#> sigmax3x3 0.00000000 0.00000000 0.0000000 1.00000000
#>               beta1    beta2    beta3       rsq
#> sigmaysq   909.1981 257.2976 276.0367 -126.0843
#> sigmayx1  3507.1691 471.2058 510.5430    0.0000
#> sigmayx2   471.2058 333.2295 150.9121    0.0000
#> sigmayx3   510.5430 150.9121 554.4386    0.0000
#> sigmax1x1    0.0000   0.0000   0.0000    0.0000
#> sigmax2x1    0.0000   0.0000   0.0000    0.0000
#> sigmax3x1    0.0000   0.0000   0.0000    0.0000
#> sigmax2x2    0.0000   0.0000   0.0000    0.0000
#> sigmax3x2    0.0000   0.0000   0.0000    0.0000
#> sigmax3x3    0.0000   0.0000   0.0000    0.0000
#> Test passed
#> [[1]]
#> [[1]][[1]]
#> [[1]][[1]]$value
#> [[1]][[1]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[1]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[2]]
#> [[1]][[2]]$value
#> [[1]][[2]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[2]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[3]]
#> [[1]][[3]]$value
#> [[1]][[3]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[3]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[4]]
#> [[1]][[4]]$value
```

```
#> [[1]][[4]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[4]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[5]]
#> [[1]][[5]]$value
#> [[1]][[5]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[5]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[6]]
#> [[1]][[6]]$value
#> [[1]][[6]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[6]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[7]]
#> [[1]][[7]]$value
#> [[1]][[7]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[7]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[8]]
#> [[1]][[8]]$value
#> [[1]][[8]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[8]]$visible
```

```
#> [1] TRUE
#>
#>
#> [[1]][[9]]
#> [[1]][[9]]$value
#> [[1]][[9]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[9]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[10]]
#> [[1]][[10]]$value
#> [[1]][[10]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[10]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[11]]
#> [[1]][[11]]$value
#> [[1]][[11]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[11]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[12]]
#> [[1]][[12]]$value
#> [[1]][[12]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[12]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[13]]
#> [[1]][[13]]$value
```

```
#> [[1]][[13]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[13]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[14]]
#> [[1]][[14]]$value
#> [[1]][[14]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[14]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[15]]
#> [[1]][[15]]$value
#> [[1]][[15]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[15]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[16]]
#> [[1]][[16]]$value
#> [[1]][[16]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[16]]$visible
#> [1] TRUE
#>
#>
#> [[1]][[17]]
#> [[1]][[17]]$value
#> [[1]][[17]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[17]]$visible
```

```
#> [1] TRUE
#>
#>
#> [[1]][[18]]
#> [[1]][[18]]$value
#> [[1]][[18]]$value[[1]]
#> [1] TRUE
#>
#>
#> [[1]][[18]]$visible
#> [1] TRUE
```

Environment

```
ls()

#> [1] "nas1982" "root"
```

Class

```
#> [[1]]
#> [1] "data.frame"
#>
#> [[2]]
#> [1] "root_criterion"
```

References

Pesigan, I. J. A., & Cheung, S. F. (2023). Monte Carlo confidence intervals for the indirect effect with missing data. *Behavior Research Methods*, *56*(3), 1678–1696. https://doi.org/10.3758/s13428-023-02114-4

R Core Team. (2024). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/