LDT Size Bias Analyses

Dani

2022-08-16

Study 1: Direct Replication

Participant Level:

```
df.PL = read.csv("Participant.csv")
```

RT

```
t.PL.RT = t.test(df.PL$Small_RT, df.PL$Large_RT, paired = T)
t.PL.RT
```

```
##
##  Paired t-test
##
## data:  df.PL$Small_RT and df.PL$Large_RT
## t = 0.95292, df = 107, p-value = 0.3428
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -1.791438  5.107937
## sample estimates:
## mean difference
##         1.65825
```

```
t2d(t.PL.RT$statistic, n=108)
```

```
##         t
## 0.1833901
```

Error Rate

```
t.PL.ER = t.test(df.PL$Small_ER, df.PL$Large_ER, paired = T)
t.PL.ER
```

```
##
##  Paired t-test
##
## data:  df.PL$Small_ER and df.PL$Large_ER
## t = 2.4269, df = 107, p-value = 0.0169
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  0.001356791 0.013458024
## sample estimates:
## mean difference
##     0.007407407
```

```
t2d(t.PL.ER$statistic, n=108)
```

```
##         t
## 0.4670597
```

Word-Pair Level:

```
df.WP = read.csv("WordPair.csv")
```

RT

```
t.WP.RT = t.test(df.WP$Small_RT, df.WP$Large_RT, paired = T)
t.WP.RT
```

```
##
##  Paired t-test
##
## data:  df.WP$Small_RT and df.WP$Large_RT
## t = 0.45592, df = 44, p-value = 0.6507
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -9.859799 15.625059
## sample estimates:
## mean difference
##         2.88263
```

```
t2d(t.WP.RT$statistic, n=108)
```

```
##          t
## 0.08774232
```

Error Rate

```
t.WP.ER = t.test(df.WP$Small_ER, df.WP$Large_ER, paired = T)
t.WP.ER
```

```
##
##  Paired t-test
##
## data:  df.WP$Small_ER and df.WP$Large_ER
## t = 1.0577, df = 44, p-value = 0.296
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -0.006418108  0.020594353
## sample estimates:
## mean difference
##     0.007088123
```

```
t2d(t.WP.ER$statistic, n=108)
```

```
##         t
## 0.2035492
```

Study 2: Norming Differences

```
df.Norm = read.csv("Norming.csv")

df.Norm.scaled = df.Norm[,1:3]
for(cn in colnames(df.Norm[4:17])){
  df.Norm.scaled[,cn] = scale(df.Norm[,cn])
}

df.Norm.WP = pivot_wider(df.Norm[,2:10],
                         id_cols = c("WordPair"),
                         names_from = c("size"),
                         values_from = c(3:9))
```

ICCs (may need to switch `s` to "average" rather than "single")

```
m = "two"
t = "agreement"
s = "single"

size_icc  = icc(cbind(df.Norm.scaled$PU.SIZE, df.Norm.scaled$GN.SIZE),
                model = m, type = t, unit = s)
gend_icc  = icc(cbind(df.Norm.scaled$PU.GEND, df.Norm.scaled$GN.GEND),
                model = m, type = t, unit = s)
val_icc   = icc(cbind(df.Norm.scaled$PU.VAL, df.Norm.scaled$GN.VAL),
                model = m, type = t, unit = s)
conc_icc  = icc(cbind(df.Norm.scaled$PU.CNC, df.Norm.scaled$GN.CNC),
```

```
              model = m, type = t, unit = s)
img_icc  = icc(cbind(df.Norm.scaled$PU.IMAG, df.Norm.scaled$GN.IMAG),
              model = m, type = t, unit = s)
fam_icc   = icc(cbind(df.Norm.scaled$PU.FAM, df.Norm.scaled$GN.FAM),
              model = m, type = t, unit = s)
arou_icc  = icc(cbind(df.Norm.scaled$PU.AROU, df.Norm.scaled$GN.AROU),
              model = m, type = t, unit = s)


ests = c(size_icc$value, val_icc$value, gend_icc$value, img_icc$value,
         conc_icc$value, arou_icc$value, fam_icc$value)
lbs = c(size_icc$lbound, val_icc$lbound, gend_icc$lbound, img_icc$lbound,
        conc_icc$lbound, arou_icc$lbound, fam_icc$lbound)
ubs = c(size_icc$ubound, val_icc$ubound, gend_icc$ubound, img_icc$ubound,
        conc_icc$ubound, arou_icc$ubound, fam_icc$ubound)
sigs = c(size_icc$p.value, val_icc$p.value, gend_icc$p.value, img_icc$p.value,
         conc_icc$p.value, arou_icc$p.value, fam_icc$p.value)

forkable = data.frame(ests, lbs, ubs, sigs)
rownames(forkable) = c("size", "valence", "gender", "imageability",
                       "concreteness", "aorusal", "familiarity")


print(knitr::kable(forkable[order(forkable$ests, decreasing = T),], bookend = T, digits = 3))
```

```
##
##
## |             |  ests|   lbs|   ubs| sigs|
## |:------------|-----:|-----:|-----:|----:|
## |size         | 0.967| 0.950| 0.978|    0|
## |gender       | 0.946| 0.919| 0.965|    0|
## |valence      | 0.934| 0.901| 0.956|    0|
## |concreteness | 0.822| 0.740| 0.880|    0|
## |imageability | 0.720| 0.600| 0.808|    0|
## |familiarity  | 0.628| 0.481| 0.740|    0|
## |aorusal      | 0.609| 0.458| 0.726|    0|
```

Paired t-tests Comparing GN and PU

```
forkable = data.frame(Dimension = c(""),
                      t.value = c(0),
                      p.value = c(0))
forkable = forkable[-1,]

for(d in 4:10){
  d.PU = d
  d.GN = d+7

  v = unlist(strsplit(names(df.Norm.scaled)[d.PU],"\\."))[2]

  tmp.t = t.test(df.Norm.scaled[,d.PU], df.Norm.scaled[,d.GN],
```

```
                paired = T)

  forkable[d,] = cbind(v, round(tmp.t$statistic,3), round(tmp.t$p.value,3))
}

print(knitr::kable(na.omit(forkable[order(forkable$t.value),]), bookend = T, row.names = F, digits = 3))
```

```
##
##
## |Dimension |t.value |p.value |
## |:---------|:-------|:-------|
## |AROU      |-0.006  |0.995   |
## |VAL       |-0.038  |0.97    |
## |FAM       |-0.097  |0.923   |
## |SIZE      |-0.223  |0.824   |
## |IMAG      |0.128   |0.899   |
## |CNC       |0.252   |0.802   |
## |GEND      |0.43    |0.668   |
```

Paired t-tests Comparing Small and Large Words

```
forkable = data.frame(Dimension = c(""),
                      `S_Mean_PU` = c(0),
                      `L_Mean_PU` = c(0),
                      `S_95CI_PU` = c(0),
                      `L_95CI_PU` = c(0),
                      t.value.PU = c(0),
                      p.value.PU = c(0)
                      )
forkable = forkable[-1,]

for(d in 1:7){
  d.s.PU = d*2
  d.l.PU = d*2+1

  v = unlist(strsplit(names(df.Norm.WP)[d.s.PU],"\\.|_"))[2]

  sm.PU      = mean(df.Norm.WP[[d.s.PU]], na.rm = T)
  lm.PU      = mean(df.Norm.WP[[d.l.PU]], na.rm = T)
  sm.PU.95CI = 1.96*(sd(df.Norm.WP[[d.s.PU]], na.rm = T) / sqrt(length(df.Norm.WP[[d.s.PU]])))
  lm.PU.95CI = 1.96*(sd(df.Norm.WP[[d.l.PU]], na.rm = T) / sqrt(length(df.Norm.WP[[d.l.PU]])))
  tmp.t.PU = t.test(df.Norm.WP[[d.l.PU]], df.Norm.WP[[d.s.PU]], paired = T)

  forkable[d,] = cbind(v, round(sm.PU,3), round(lm.PU,3), round(sm.PU.95CI,4), round(lm.PU.95CI,4), rou
                       )
}

print(knitr::kable(forkable[order(forkable$t.value.PU),], bookend = T, row.names = F, digits = 3))
```

```
##
##
```

```
## |Dimension |S_Mean_PU |L_Mean_PU |S_95CI_PU |L_95CI_PU |t.value.PU |p.value.PU |
## |:---------|:---------|:---------|:---------|:---------|:----------|:----------|
## |IMAG      |5.961     |5.941     |0.1915    |0.1674    |-0.164     |0.87       |
## |CNC       |5.508     |5.392     |0.18      |0.2312    |-0.758     |0.453      |
## |FAM       |5.863     |5.505     |0.1395    |0.17      |-3.601     |0.001      |
## |AROU      |3.113     |3.315     |0.2148    |0.1621    |1.431      |0.16       |
## |VAL       |4.445     |4.771     |0.3392    |0.2301    |1.487      |0.144      |
## |SIZE      |2.431     |5.947     |0.1536    |0.1395    |29.214     |0          |
## |GEND      |2.745     |3.495     |0.2274    |0.2081    |4.989      |0          |
```

Study 3: Continuous Predictors and Multi-Level Mediation Analyses

```
df.MLM = read.csv("MLM.csv")
```

Base RT ~ Size Regression

```
mlm.sizeOnly = lmer(RT ~ SIZE + (1|SubID), data = df.MLM)
summary(mlm.sizeOnly)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: RT ~ SIZE + (1 | SubID)
##    Data: df.MLM
##
## REML criterion at convergence: 102819.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.6287 -0.6657 -0.0896  0.5470  6.4364
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  SubID    (Intercept) 3853     62.07
##  Residual             6456     80.35
## Number of obs: 8820, groups:  SubID, 108
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)  528.9315     6.3483  130.9597  83.319  < 2e-16 ***
## SIZE          -1.5196     0.4688 8711.3687  -3.241  0.00119 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr)
## SIZE -0.311
```

A paths

```
mlm.sizePredGend = lmer(GEND ~ SIZE + (1|SubID), data = df.MLM)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
summary(mlm.sizePredGend)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: GEND ~ SIZE + (1 | SubID)
##    Data: df.MLM
##
## REML criterion at convergence: 19737.3
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.3359 -0.6814 -0.0252  0.6275  2.4203
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  SubID    (Intercept) 0.0000   0.0000
##  Residual             0.5478   0.7402
## Number of obs: 8820, groups:  SubID, 108
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept) 2.271e+00  1.979e-02 8.818e+03  114.73   <2e-16 ***
## SIZE        2.049e-01  4.316e-03 8.818e+03   47.47   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr)
## SIZE -0.917
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```
mlm.sizePredFam = lmer(FAM ~ SIZE + (1|SubID), data = df.MLM)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
summary(mlm.sizePredFam)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: FAM ~ SIZE + (1 | SubID)
##    Data: df.MLM
##
## REML criterion at convergence: 13815.3
```

```
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.9588 -0.5616  0.2249  0.7389  1.6784
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  SubID    (Intercept) 0.0000   0.000
##  Residual             0.2799   0.529
## Number of obs: 8820, groups:  SubID, 108
##
## Fixed effects:
##               Estimate Std. Error         df t value Pr(>|t|)
## (Intercept)  6.083e+00  1.415e-02  8.818e+03  429.98   <2e-16 ***
## SIZE        -9.165e-02  3.085e-03  8.818e+03  -29.71   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr)
## SIZE -0.917
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

Size Suppressed by Familiarity

```
mlm.sizeAndFam = lmer(RT ~ SIZE + FAM + (1|SubID), data = df.MLM)
summary(mlm.sizeAndFam)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: RT ~ SIZE + FAM + (1 | SubID)
##    Data: df.MLM
##
## REML criterion at convergence: 102585.5
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.8036 -0.6544 -0.0925  0.5414  6.6994
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  SubID    (Intercept) 3840     61.97
##  Residual             6288     79.29
## Number of obs: 8820, groups:  SubID, 108
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)  677.7867    11.5957 1348.2250  58.452  < 2e-16 ***
## SIZE          -3.7630     0.4853 8710.3860  -7.754 9.88e-15 ***
## FAM          -24.4703     1.5973 8710.4266 -15.320  < 2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr) SIZE
## SIZE -0.413
## FAM  -0.838  0.302
```

Size Confounded with Gender

```
mlm.sizeAndGend = lmer(RT ~ SIZE + GEND + (1|SubID), data = df.MLM)
summary(mlm.sizeAndGend)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: RT ~ SIZE + GEND + (1 | SubID)
##    Data: df.MLM
##
## REML criterion at convergence: 102810.9
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.5795 -0.6643 -0.0917  0.5441  6.4890
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  SubID    (Intercept) 3853     62.07
##  Residual             6452     80.32
## Number of obs: 8820, groups:  SubID, 108
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)  535.8021     6.8698  179.4424  77.994  < 2e-16 ***
## SIZE          -0.9001     0.5251 8710.3042  -1.714  0.08653 .
## GEND          -3.0248     1.1563 8710.3182  -2.616  0.00891 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr) SIZE
## SIZE -0.084
## GEND -0.382 -0.451
```

Double Mediation of Size Effect by both Size and Familiarity

```
mlm.Full = lmer(RT ~ SIZE + FAM + GEND + (1|SubID), data = df.MLM)
summary(mlm.Full)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
```

```
## Formula: RT ~ SIZE + FAM + GEND + (1 | SubID)
##    Data: df.MLM
##
## REML criterion at convergence: 102541.8
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.7249 -0.6550 -0.0858  0.5464  6.8612
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  SubID    (Intercept) 3838     61.95
##  Residual             6258     79.11
## Number of obs: 8820, groups:  SubID, 108
##
## Fixed effects:
##              Estimate Std. Error        df t value Pr(>|t|)
## (Intercept)  710.1395    12.6135 1805.0914  56.300  < 2e-16 ***
## SIZE          -2.4423     0.5256 8709.3024  -4.647 3.42e-06 ***
## FAM          -26.9641     1.6397 8709.4805 -16.445  < 2e-16 ***
## GEND          -7.5648     1.1718 8709.3658  -6.456 1.14e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr) SIZE   FAM
## SIZE -0.194
## FAM  -0.840  0.178
## GEND -0.397 -0.389  0.236
```

'bmlm' version

Size Suppressed by Familiarity

```
mlm.fam = mlm(d = df.MLM,
              id = "SubID",
              x  = "SIZE",
              m  = "FAM",
              y  = "RT")
```

```
## Estimating model, please wait.
```

```
##
## SAMPLING FOR MODEL 'bmlm' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.004 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 40 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1379.21 seconds (Warm-up)
## Chain 1:                764.52 seconds (Sampling)
## Chain 1:                2143.73 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bmlm' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.003 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 30 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1346.23 seconds (Warm-up)
## Chain 2:                763.328 seconds (Sampling)
## Chain 2:                2109.56 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'bmlm' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.003 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 30 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
```

```
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 949.597 seconds (Warm-up)
## Chain 3:                762.501 seconds (Sampling)
## Chain 3:                1712.1 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'bmlm' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.003 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 30 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 1247.83 seconds (Warm-up)
## Chain 4:                762.199 seconds (Sampling)
## Chain 4:                2010.03 seconds (Total)
## Chain 4:

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

```
mlm_summary(mlm.fam)
```

```
##   Parameter   Mean   SE Median   2.5%  97.5% n_eff Rhat
## 1         a  -0.09 0.00  -0.09  -0.10  -0.09  7171    1
## 2         b -24.46 1.60 -24.46 -27.77 -21.38  4664    1
## 3        cp  -3.76 0.51  -3.76  -4.76  -2.77  6471    1
## 4        me   2.24 0.17   2.24   1.92   2.57  5156    1
```

```
## 5         c  -1.52 0.50  -1.52  -2.49  -0.56  8394    1
## 6       pme  -1.67 3.66  -1.47  -4.04  -0.86  4059    1
```

Size Confounded with Gender

```
mlm.gend = mlm(d = df.MLM,
               id = "SubID",
               x  = "SIZE",
               m  = "GEND",
               y  = "RT")
```

```
## Estimating model, please wait.

##
## SAMPLING FOR MODEL 'bmlm' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.003 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 30 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 859.452 seconds (Warm-up)
## Chain 1:                382.301 seconds (Sampling)
## Chain 1:                1241.75 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'bmlm' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.004 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 40 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 1259.79 seconds (Warm-up)
## Chain 2:                383.352 seconds (Sampling)
## Chain 2:                1643.14 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'bmlm' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.003 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 30 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1021.62 seconds (Warm-up)
## Chain 3:                764.889 seconds (Sampling)
## Chain 3:                1786.51 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'bmlm' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.003 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 30 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 840.071 seconds (Warm-up)
## Chain 4:                411.102 seconds (Sampling)
## Chain 4:                1251.17 seconds (Total)
## Chain 4:
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quanti
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

```
mlm_summary(mlm.gend)
```

```
##   Parameter  Mean   SE Median  2.5% 97.5% n_eff Rhat
## 1         a  0.20 0.00   0.20  0.20  0.21  4779    1
## 2         b -3.02 1.39  -3.00 -5.78 -0.28  2675    1
## 3        cp -0.90 0.55  -0.90 -1.99  0.19  6064    1
## 4        me -0.62 0.29  -0.62 -1.19 -0.06  2678    1
## 5         c -1.52 0.51  -1.52 -2.53 -0.54  6064    1
## 6       pme  0.41 3.76   0.41  0.03  1.24  3750    1
```