Brunner-Munzel Validation #1

Basic Validation using R Data

Aaron R. Caldwell

Table of contents

This document serves as a validation that the `TOSTER` package matches, or at least approximately matches, the provided results of a Brunner-Munzel test compared to the functions implemented in the `nparcomp`, `brunnermunzel`, and `lawstat`.

Sleep Data

Two-Sample

```
test_that("Two-sample test (t-stat)",{
tost_res = brunner_munzel(x = subset(sleep, group == 2)$extra,
                          y = subset(sleep, group == 1)$extra,
                          paired = FALSE, perm = FALSE, alternative = "two.sided")

nparcomp_res = nparcomp::npar.t.test(data = sleep, info = FALSE,
                                     extra ~ group, alternative = "two.sided",
                                     rounds = 5, method = "t.app")
expect_equal(tost_res$p.value, nparcomp_res$Analysis$p.Value[1],
             tolerance = .0001)

bm_res = brunnermunzel.test(y = subset(sleep, group == 2)$extra,
                          x = subset(sleep, group == 1)$extra,
                          alternative = "two.sided")

expect_equal(abs(tost_res$p.value - bm_res$p.value),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$estimate) - unname(bm_res$estimate)),
             0,
             tolerance = .0001)
expect_equal(abs(unname(tost_res$conf.int[[1]])-unname(bm_res$conf.int[1])),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$conf.int[[2]]) - unname(bm_res$conf.int[2])),
             0,
             tolerance = .01)

law_res = brunner.munzel.test(y = subset(sleep, group == 2)$extra,
                          x = subset(sleep, group == 1)$extra,
                          alternative = "two.sided")

expect_equal(abs(tost_res$p.value - law_res$p.value),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$estimate) - unname(law_res$estimate)),
             0,
```

```
              tolerance = .0001)
  expect_equal(abs(unname(tost_res$conf.int[[1]])-unname(law_res$conf.int[1])),
              0,
              tolerance = .01)
  expect_equal(abs(unname(tost_res$conf.int[[2]]) - unname(law_res$conf.int[2])),
              0,
              tolerance = .01)

  })
```

```
Test passed
```

Two-sample permuted

```
test_that("Two-sample permutation test (t-stat)",{

set.seed(1728622)
tost_res = brunner_munzel(x = subset(sleep, group == 2)$extra,
                          y = subset(sleep, group == 1)$extra,
                          paired = FALSE, perm = TRUE, alternative = "two.sided",
                          max_n_perm = 40000)
set.seed(1728622)
nparcomp_res = nparcomp::npar.t.test(data = sleep, info = FALSE,
                                     extra ~ group, alternative = "two.sided",
                                     rounds = 5, method = "permu",
                                     nperm = 40000)
expect_equal(abs(tost_res$p.value - nparcomp_res$Analysis$p.value[1]),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$estimate) - unname(nparcomp_res$Analysis$Estimator[1])),
             0,
             tolerance = .0001)
expect_equal(abs(unname(tost_res$conf.int[[1]])-unname(nparcomp_res$Analysis$Lower[1])),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$conf.int[[2]]) - unname(nparcomp_res$Analysis$Upper[1])),
             0,
             tolerance = .01)


bm_res = brunnermunzel.permutation.test(y = subset(sleep, group == 2)$extra,
                          x = subset(sleep, group == 1)$extra,
                          alternative = "two.sided")

expect_equal(abs(tost_res$p.value - bm_res$p.value),
             0,
             tolerance = .01)




})
```

```
Test passed
```

Paired

```
test_that("paired test (t-stat)",{
tost_res = brunner_munzel(x = subset(sleep, group == 2)$extra,
                          y = subset(sleep, group == 1)$extra,
                          paired = TRUE, perm = FALSE, alternative = "two.sided")

nparcomp_res = nparcomp::npar.t.test.paired(data = sleep, info = FALSE,
                                    extra ~ group, alternative = "two.sided",
                                    rounds = 5,
                                    nperm = 40000,
                                    plot.simci = FALSE)
expect_equal(abs(tost_res$p.value - nparcomp_res$Analysis[1,5]),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$estimate) - unname(nparcomp_res$Analysis[1,2])),
             0,
             tolerance = .0001)
expect_equal(abs(unname(tost_res$conf.int[[1]])-unname(nparcomp_res$Analysis[1,1])),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$conf.int[[2]]) - unname(nparcomp_res$Analysis[1,3])),
             0,
             tolerance = .01)


set.seed(220975298)
tost_res = brunner_munzel(x = subset(sleep, group == 2)$extra,
                          y = subset(sleep, group == 1)$extra,
                          paired = TRUE, perm = TRUE, alternative = "two.sided",
                          max_n_perm = 40000)
set.seed(220975298)
nparcomp_res = nparcomp::npar.t.test.paired(data = sleep, info = FALSE,
                                    extra ~ group, alternative = "two.sided",
                                    plot.simci = FALSE,
                                    rounds = 5,
                                    nperm = 40000)
expect_equal(abs(tost_res$p.value - nparcomp_res$Analysis[2,5]),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$estimate) - unname(nparcomp_res$Analysis[2,2])),
```

```
              0,
              tolerance = .0001)
  expect_equal(abs(unname(tost_res$conf.int[[1]])-unname(nparcomp_res$Analysis[2,1])),
              0,
              tolerance = .02)
  expect_equal(abs(unname(tost_res$conf.int[[2]]) - unname(nparcomp_res$Analysis[2,3])),
              0,
              tolerance = .02)
  })
```

```
Test passed
```

PGI Data

Two-Sample

```
test_that("Two-sample test (t-stat)",{
tost_res = brunner_munzel(x = subset(PGI, timepoint == "base")$PGIscore,
                          y = subset(PGI, timepoint == "week4")$PGIscore,
                          paired = FALSE, perm = FALSE, alternative = "two.sided")

nparcomp_res = nparcomp::npar.t.test(data = PGI, info = FALSE,
                                     PGIscore ~ timepoint, alternative = "two.sided",
                                     rounds = 5, method = "t.app")
expect_equal(tost_res$p.value, nparcomp_res$Analysis$p.Value[1],
             tolerance = .0001)

bm_res = brunnermunzel.test(y = subset(PGI, timepoint == "base")$PGIscore,
                          x = subset(PGI, timepoint == "week4")$PGIscore,
                          alternative = "two.sided")

expect_equal(abs(tost_res$p.value - bm_res$p.value),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$estimate) - unname(bm_res$estimate)),
             0,
             tolerance = .0001)
expect_equal(abs(unname(tost_res$conf.int[[1]])-unname(bm_res$conf.int[1])),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$conf.int[[2]]) - unname(bm_res$conf.int[2])),
             0,
             tolerance = .01)

law_res = brunner.munzel.test(y = subset(PGI, timepoint == "base")$PGIscore,
                          x = subset(PGI, timepoint == "week4")$PGIscore,
                          alternative = "two.sided")

expect_equal(abs(tost_res$p.value - law_res$p.value),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$estimate) - unname(law_res$estimate)),
             0,
```

```
              tolerance = .0001)
  expect_equal(abs(unname(tost_res$conf.int[[1]])-unname(law_res$conf.int[1])),
              0,
              tolerance = .01)
  expect_equal(abs(unname(tost_res$conf.int[[2]]) - unname(law_res$conf.int[2])),
              0,
              tolerance = .01)

  })
```

```
Test passed
```

Two-sample permuted

```
test_that("Two-sample permutation test (t-stat)",{

set.seed(1728622)
tost_res = brunner_munzel(x = subset(PGI, timepoint == "week4")$PGIscore,
                          y = subset(PGI, timepoint == "base")$PGIscore,
                          paired = FALSE, perm = TRUE, alternative = "two.sided",
                          max_n_perm = 40000)
set.seed(1728622)
nparcomp_res = nparcomp::npar.t.test(data = PGI, info = FALSE,
                                     PGIscore ~ timepoint, alternative = "two.sided",
                                     rounds = 5, method = "permu",
                                     nperm = 40000)
expect_equal(abs(tost_res$p.value - nparcomp_res$Analysis$p.value[1]),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$estimate) - unname(nparcomp_res$Analysis$Estimator[1])),
             0,
             tolerance = .0001)
expect_equal(abs(unname(tost_res$conf.int[[1]])-unname(nparcomp_res$Analysis$Lower[1])),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$conf.int[[2]]) - unname(nparcomp_res$Analysis$Upper[1])),
             0,
             tolerance = .01)


bm_res = brunnermunzel.permutation.test(y = subset(PGI, timepoint == "week4")$PGIscore,
                          x = subset(PGI, timepoint == "base")$PGIscore,
                          alternative = "two.sided")

expect_equal(abs(tost_res$p.value - bm_res$p.value),
             0,
             tolerance = .01)




})
```

```
-- Warning (<text>:27:1): Two-sample permutation test (t-stat) -----------------
```

```
Sample number is too large. Using 'brunnermunzel.test'

Backtrace:
 1. brunnermunzel::brunnermunzel.permutation.test(...)
 2. brunnermunzel:::brunnermunzel.permutation.test.default(...)
```

Paired

```
test_that("paired test (t-stat)",{
tost_res = brunner_munzel(x = subset(PGI, timepoint == "week4")$PGIscore,
                          y = subset(PGI, timepoint == "base")$PGIscore,
                          paired = TRUE, perm = FALSE, alternative = "two.sided")

nparcomp_res = nparcomp::npar.t.test.paired(data = PGI, info = FALSE,
                                    PGIscore ~ timepoint, alternative = "two.sided",
                                    rounds = 5,
                                    nperm = 40000,
                                    plot.simci = FALSE)
expect_equal(abs(tost_res$p.value - nparcomp_res$Analysis[1,5]),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$estimate) - unname(nparcomp_res$Analysis[1,2])),
             0,
             tolerance = .0001)
expect_equal(abs(unname(tost_res$conf.int[[1]])-unname(nparcomp_res$Analysis[1,1])),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$conf.int[[2]]) - unname(nparcomp_res$Analysis[1,3])),
             0,
             tolerance = .01)


set.seed(220975298)
tost_res = brunner_munzel(x = subset(PGI, timepoint == "week4")$PGIscore,
                          y = subset(PGI, timepoint == "base")$PGIscore,
                          paired = TRUE, perm = TRUE, alternative = "two.sided",
                          max_n_perm = 10000)
set.seed(220975298)
nparcomp_res = nparcomp::npar.t.test.paired(data = PGI, info = FALSE,
                                    PGIscore ~ timepoint, alternative = "two.sided",
                                    plot.simci = FALSE)
expect_equal(abs(tost_res$p.value - nparcomp_res$Analysis[2,5]),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$estimate) - unname(nparcomp_res$Analysis[2,2])),
             0,
             tolerance = .001)
```

```
expect_equal(abs(unname(tost_res$conf.int[[1]])-unname(nparcomp_res$Analysis[2,1])),
             0,
             tolerance = .02)
expect_equal(abs(unname(tost_res$conf.int[[2]]) - unname(nparcomp_res$Analysis[2,3])),
             0,
             tolerance = .02)
})
```

```
Test passed
```

mtcars Data

Two-Sample

```
test_that("Two-sample test (t-stat)",{
tost_res = brunner_munzel(x = subset(mtcars, am == 1)$mpg,
                          y = subset(mtcars, am == 0)$mpg,
                          paired = FALSE, perm = FALSE, alternative = "two.sided")

nparcomp_res = nparcomp::npar.t.test(data = mtcars, info = FALSE,
                                     mpg ~ am, alternative = "two.sided",
                                     rounds = 5, method = "t.app")
expect_equal(abs(tost_res$p.value- nparcomp_res$Analysis$p.Value[1]),
             0,
             tolerance = .0001)

bm_res = brunnermunzel.test(y = subset(mtcars, am == 1)$mpg,
                          x = subset(mtcars, am == 0)$mpg,
                          alternative = "two.sided")

expect_equal(abs(tost_res$p.value - bm_res$p.value),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$estimate) - unname(bm_res$estimate)),
             0,
             tolerance = .0001)
expect_equal(abs(unname(tost_res$conf.int[[1]])-unname(bm_res$conf.int[1])),
             0,
             tolerance = .01)
```

```r
expect_equal(abs(unname(tost_res$conf.int[[2]]) - unname(bm_res$conf.int[2])),
             0,
             tolerance = .01)

law_res = brunner.munzel.test(y = subset(mtcars, am == 1)$mpg,
                              x = subset(mtcars, am == 0)$mpg,
                              alternative = "two.sided")

expect_equal(abs(tost_res$p.value - law_res$p.value),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$estimate) - unname(law_res$estimate)),
             0,
             tolerance = .0001)
expect_equal(abs(unname(tost_res$conf.int[[1]])-unname(law_res$conf.int[1])),
             0,
             tolerance = .01)
expect_equal(abs(unname(tost_res$conf.int[[2]]) - unname(law_res$conf.int[2])),
             0,
             tolerance = .01)

})
```

```
Test passed
```