Duffy *et al.* - sample size estimator + code to generate and analyse simulated datasets

2021-08-08

The `EnvStats` package should be installed for functions that are called directly. Generating simulation datasets sampled from skewed distributions requires the `sn` package. `ggplot2` is used for plotting of results.

```
require(ggplot2)
require(sn)
require(parallel)
```

The below function to subsample each population (for subsamples where n 3) and use a modified Two One-Sided T-test (TOST) procedure to test for statistical equivalence between the subsample and source populations. If the total number of unique subsamples `nsub` is 10 000 (i.e. when n is low), all combinations are tested. If `nsub` is > 10 000, 10 000 random combinations are tested. First a `chenTTest` (from the `EnvStats` package) is tried, which returns NA if skew is not detected. Where skew is not detected a standard t-test is used instead. Equivalence between variances is calculated using two one-sided chi-squared tests (`EnvStats::varTest`).

As demonstrated in the below examples, argument `x` should be either a vector of trait values or a single numeric index for simulated populations, for which the argument `set` must also be defined. The `equiv_margin` argument should be a single number indicating the desired equivalence margin for testing (e.g. `equiv_margin = 0.25` for 'negligble error', ±0.25 °C, or `equiv_margin = 1` for 'moderate error', ±1.00 °C).

```
TOSTer <- function(x, equiv_margin, set = NULL) {

    if (is.null(set))
        {
            spacdat <- x
        }  #if used on empirical data (i.e. one vector of CT data)
    if (is.numeric(set)) {
        spacdat <- skew_dists[[set]][x][[1]]
    }
    # if used on simulated data where the 'set' is required
    # for subsetting

    ChenTOST <- function(sub, spacdat, full_mn, full_sd, equiv_margin) {
        # modified CHEN TOST procedure run all combinations
        # if < 10 000 possible combinations
        if (choose(length(spacdat), sub) < 10000) {
            nsub <- choose(length(spacdat), sub)
            comX <- combn(1:length(spacdat), sub)
            smpl_mn <- split(comX, rep(1:ncol(comX), each = nrow(comX)))
        } else {
            # if there are more than 10 000 possible
            # combinations, run a random subset of 10 000
```

```
    nsub <- 10000
    smpl_mn <- lapply(1:nsub, function(x) {
        sample(1:length(spacdat), sub)
    })
}

equiv <- sapply(smpl_mn, function(x) {
    tryCatch({

        # EnvStats::chenTTest #two one-sided
        # t-tests for mean equivalence try a Chen
        # t-test on one end of distribution
        mn_upr <- EnvStats::chenTTest(spacdat[x], alternative = "greater",
          mu = full_mn - equiv_margin)$p.value[3]
        # if data are not sufficiently skewed for
        # Chen t-test, run a standard t-test
        # instead
        if (is.na(mn_upr)) {
          mn_upr <- t.test(spacdat[x], alternative = "greater",
            mu = full_mn - equiv_margin)$p.value
        }
        mn_lwr <- t.test(spacdat[x], alternative = "less",
          mu = full_mn + equiv_margin)$p.value  #standard t-test for other end
        mn_rez <- c(mn_upr, mn_lwr)

        # EnvStats::varTest
        var_upr <- EnvStats::varTest(spacdat[x], alternative = "greater",
          sigma.squared = ifelse((full_sd - equiv_margin) >
            0, (full_sd - equiv_margin)^2, 1e-05), conf.level = 0.95)$p.value
        var_lwr <- EnvStats::varTest(spacdat[x], alternative = "less",
          sigma.squared = (full_sd + equiv_margin)^2,
          conf.level = 0.95)$p.value
        var_rez <- c(var_upr, var_lwr)

        mnvar_equiv <- c(NA, NA)
        ifelse(sum(is.na(mn_rez)) > 0, {
          mnvar_equiv[1] <- NA
        }, {
          mnvar_equiv[1] <- max(mn_rez) < 0.05
        })
        ifelse(sum(is.na(var_rez)) > 0, {
          mnvar_equiv[2] <- NA
        }, {
          mnvar_equiv[2] <- max(var_rez) < 0.05
        })

        return(mnvar_equiv)
        # error handling if TOST procedure fails
        # (e.g. if data contain too many non-unique
        # values)\t
    }, error = function(z) {
        return(c(NA, NA))
    })
```

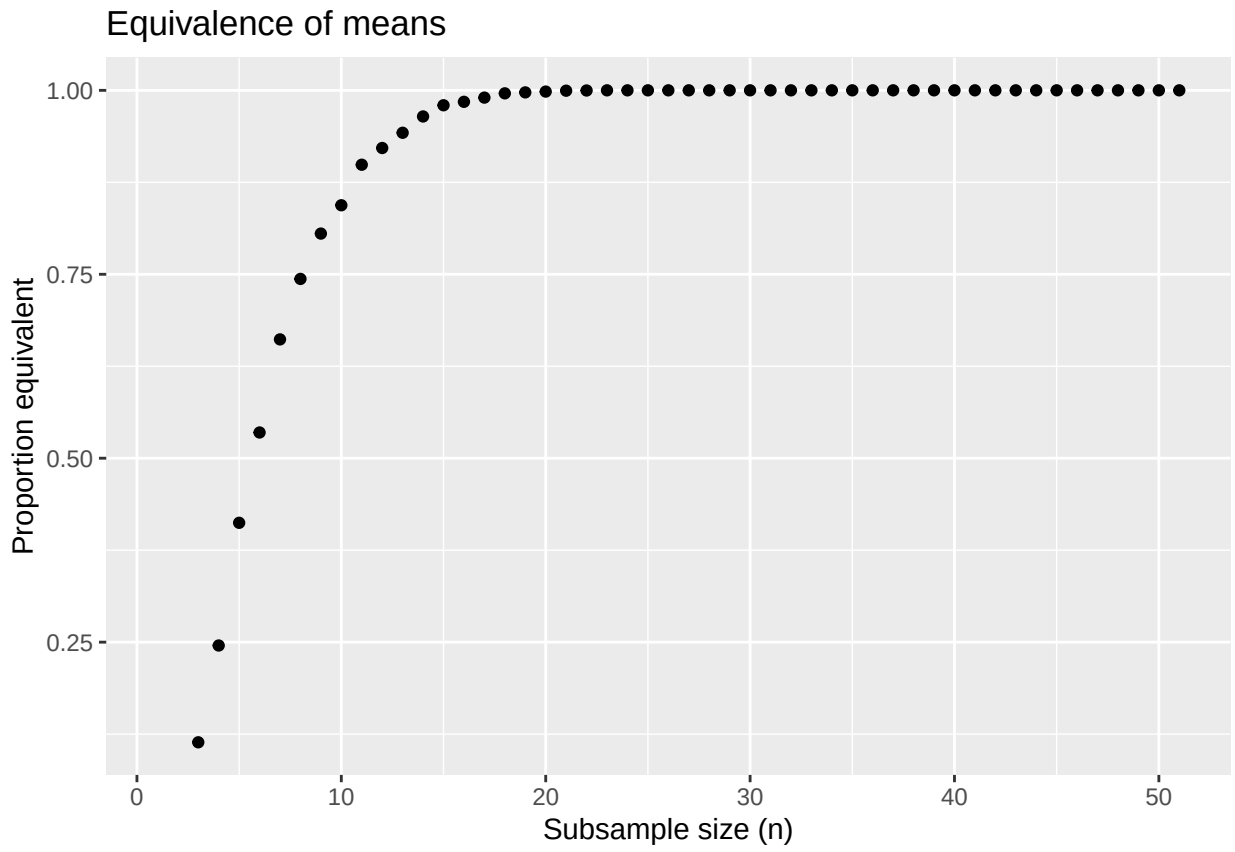
```
        })

        mn_eq <- equiv[1, ]
        var_eq <- equiv[2, ]
        # calculate proportion of succesful tests
        return(c(length(mn_eq[which(mn_eq == T)])/sum(is.finite(mn_eq)),
            length(var_eq[which(var_eq == T)])/sum(is.finite(var_eq))))
    }

    res <- do.call(cbind, lapply(3:length(spacdat), ChenTOST,
        spacdat = spacdat, full_mn = mean(spacdat), full_sd = sd(spacdat),
        equiv_margin = equiv_margin))  #only doing subsets where n > 3
    res2return <- cbind(1:length(spacdat), t(cbind(matrix(NA,
        nrow = 2, ncol = 2), res)))
    colnames(res2return) <- c("n", "mn_eq", "var_eq")
    return(data.frame(res2return))
}
```

Example using CTmax data from *Desoria trispinata*, passed as a vector of individual CT values.

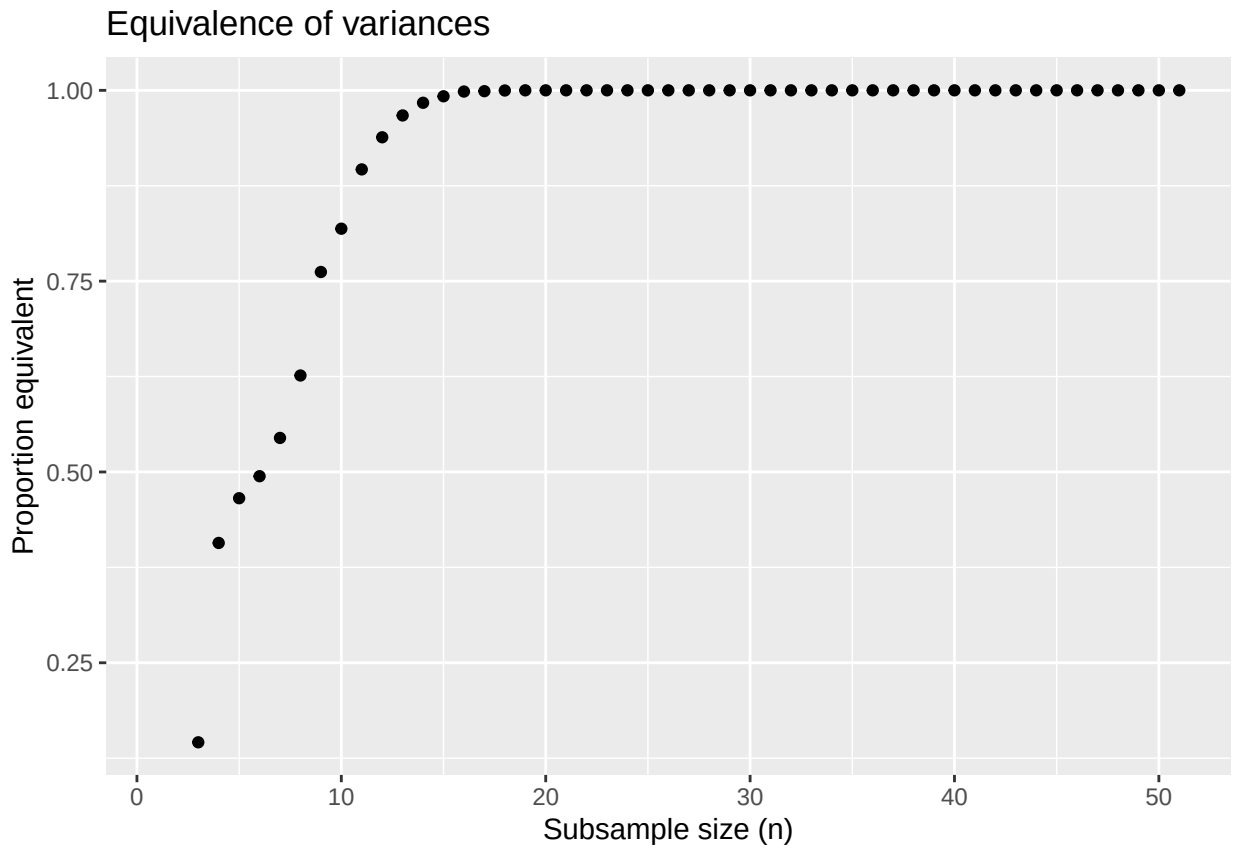
```
desoria_ctmax <- c(37.5, 37.5, 37.5, 37.5, 37.5, 37.5, 37.5,
    37.5, 37.5, 37.9, 37.9, 37.9, 37.9, 37.9, 38.2, 38.2, 38.2,
    37.7, 37.7, 37.7, 37.8, 37.8, 37.8, 37.8, 37.8, 37.8, 37.8,
    37.8, 37.8, 37.8, 37.8, 37.8, 37.8, 37.8, 37.8, 37.6, 37.6,
    37.6, 37.6, 37.6, 37.6, 37.6, 37.6, 37.6, 38, 38, 38, 38.4,
    38.4, 38.4, 38.4)

des_max_TOSTed_0p25 <- TOSTer(desoria_ctmax, equiv_margin = 0.25,
    set = NULL)
# run TOSTer function on vector of CT values with a 0.25 °C
# equivalence margin.

head(des_max_TOSTed_0p25)
```

```
##   n     mn_eq    var_eq
## 1 1        NA        NA
## 2 2        NA        NA
## 3 3 0.1138915 0.1458140
## 4 4 0.2454481 0.4070013
## 5 5 0.4122007 0.4655915
## 6 6 0.5349605 0.4944483
```

```
ggplot(data = des_max_TOSTed_0p25, aes(x = n, y = mn_eq)) + geom_point() +
    labs(x = "Subsample size (n)", y = "Proportion equivalent") +
    ggtitle("Equivalence of means")
```




```
ggplot(data = des_max_TOSTed_0p25, aes(x = n, y = var_eq)) +
    geom_point() + labs(x = "Subsample size (n)", y = "Proportion equivalent") +
    ggtitle("Equivalence of variances")
```



Example generating and using simulated data. These data are parameterised using mean, variance, and skew values from the published literature, which can be replaced/updated using preliminary studies on the species and trait of interest. For demonstration purposes, seeds are generated for 50 reproducible randomly generated populations.

```
pop_seeds <- c(809560L, 854479L, 16380L, 856738L, 42793L, 949638L,
    349154L, 680772L, 904048L, 32917L, 846553L, 23701L, 673766L,
    538887L, 373079L, 888918L, 244202L, 598545L, 671795L, 530864L,
    944901L, 489945L, 21730L, 380536L, 854547L, 482305L, 628216L,
    892896L, 198276L, 988513L, 296083L, 262567L, 209831L, 76535L,
    257664L, 198760L, 523672L, 174167L, 129254L, 916874L, 288318L,
    408110L, 212368L, 18343L, 963653L, 102179L, 652356L, 779200L,
    109124L, 253919L)
```

Set parameters for source population(s) and equivalence testing. Change as required.

```
mn <- 40.0296951  #mean of population(s)
stdev <- 1  #standard deviation of population(s)
skews <- c(0, 1, 2, 10, 50)  #skew of population(s)

equiv_margin <- 0.25  #equivalence margin for equivalence testing
```

Generate randomly drawn population(s) comprising 100 individuals from normal/skewed distributions using the `sn` package.

```
skew_dists <- lapply(skews, function(skw, stdev, seeds) {
    lapply(seeds, function(seed) {
        set.seed(seed)
        rsn(100, xi = mn, omega = stdev, alpha = skw)
    })
}, stdev = stdev, seeds = pop_seeds)
```

Run the `TOSTer` function across all simulated populations. Simulation and testing will take a considerable amount of time (~16hrs on a 2.6Ghz quad-core desktop). The process can be distributed across cores and run in parallel using `parallel::mclapply` if desired (substitute in `mclapply` for one of the `lapply` loops in the `TOSTer` function). Results will be outputted to a list, with each element containing results for each skew variant.

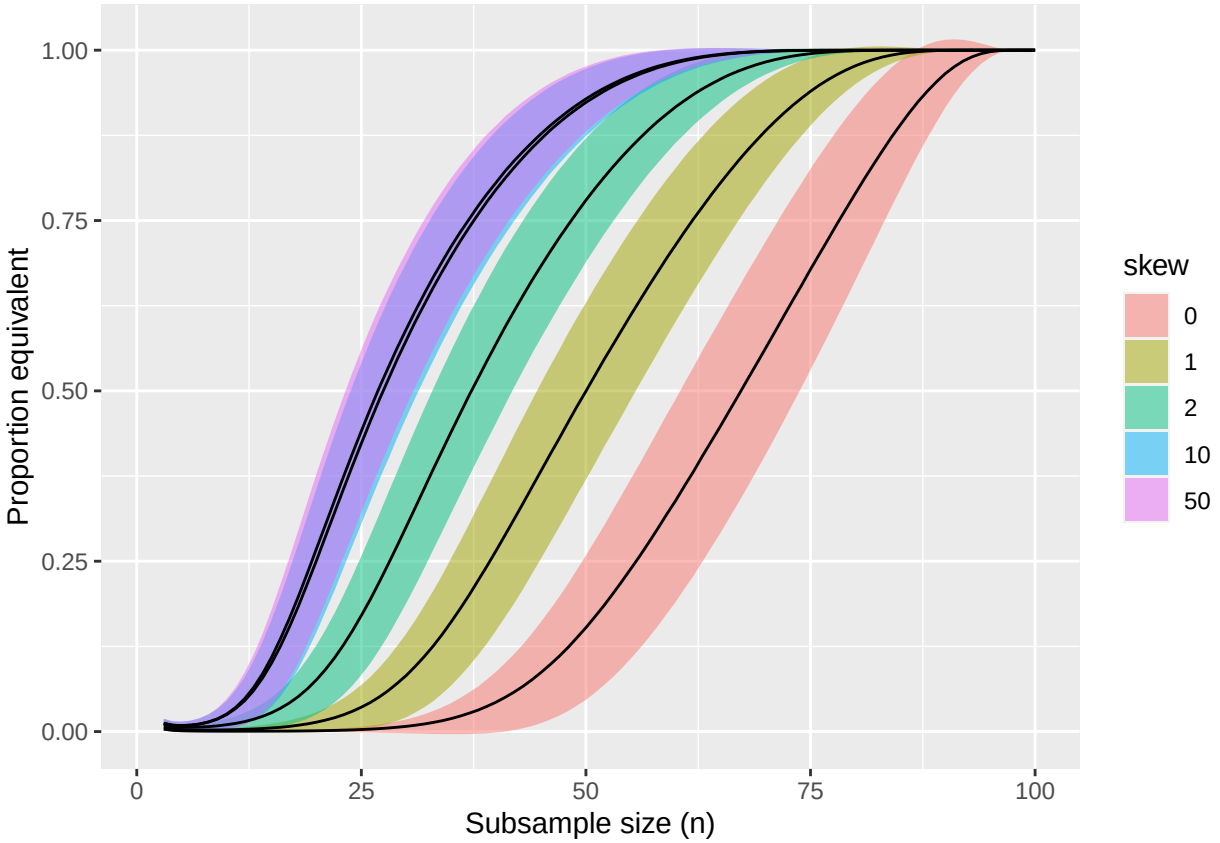
```
tost_res <- lapply(1:length(skews), function(x) {
    do.call(rbind, lapply(1:length(pop_seeds), TOSTer, equiv_margin = equiv_margin,
        set = x))
})
# name each element in the list for future reference
names(tost_res) <- paste0("sd", sub("\\.", "p", stdev), "_skw",
    sub("\\.", "p", skews), "_eqm", sub("\\.", "p", equiv_margin))
```

Summarise and format data for plotting.

```
plot_dat <- do.call(rbind, lapply(1:length(skews), function(x) {
    data.frame(nsamp = 1:100, mn = sapply(1:100, function(n) {
        mean(tost_res[[x]][which(tost_res[[x]][, "n"] == n),
            "mn_eq"])
    }), stdev = sapply(1:100, function(n) {
        sd(tost_res[[x]][which(tost_res[[x]][, "n"] == n), "mn_eq"])
    }), grp = skews[x])
}))
```

Create plot from `plot_dat`.

```
ggplot(data = plot_dat, mapping = aes(x = nsamp, y = mn, ymin = mn -
    stdev, ymax = mn + stdev, fill = as.factor(grp))) + geom_ribbon(alpha = 0.5) +
    geom_line() + labs(x = "Subsample size (n)", y = "Proportion equivalent",
    fill = "skew")
```


1.00
0.75
0.50
0.25
0.00
Proportion equivalent
0
25
50
75
100
Subsample size (n)
skew
0
1
2
10
50