DEPARTMENT OF EPIDEMIOLOGY AND BIOSTATISTICS,
SCHULICH SCHOOL OF MEDICINE & DENTISTRY,
WESTERN UNIVERSITY

A Detailed Reproduction on the Preliminary Tests of Variances with Statistical Inference

Jiaqi Bi

September 29, 2022

1 INTRODUCTION

This paper reproduced the simulation study conducted by Zimmerman (2004) with the argument on significance tests including the two-sample Student t test, Welch t test, following by a preliminary Levene tests that introduced by Levene (1960). The significance tests are widely used within interdisciplinary research, such as A/B testing in psychological articles, hypothesis testing within social sciences, medical research, and etc. However, the algorithm and weakness of hypothesis testing when the equality of variances are usually not strictly considered if the testing was conducted by non-statisticians as Zimmerman (2004) mentioned. It is very common that most of testing procedures do not involve preliminary test of variances. Thus, in this paper, the simulation study on significance tests and preliminary tests are reproduced with Type I error rates are considered as the outcome for the comparison. The simulation algorithm in this paper will combine some statistical inferences as well, to represent the integrated algorithm.

2 METHOD

The simulation procedure contains normal random number generations, where some pseudo-random numbers are generated in each condition using the method invented by Becker et al. (1988) and Johnson et al. (1995). The condition includes several different divisions of standard errors of two groups. The sample size of two groups is either 30 or 60, where there will be 50, 40, 20, or 10 for each group when the total sample size is 60; there will be 25, 20, 10, or 5 for each group when the total sample size is 30. The standard error ratio is determined manually, starting from 1.0, i.e. $\sigma_1/\sigma_2 = 1.0$, and will range from 1.0 to 2.5 with increases of

0.2 or 0.5. Each group of simulation will have 50000 replications, and there will be four tests in total for each replication. The preliminary test of equality of variances introduced by Levene (1960) will firstly be simulated, and the conditional t test either Student t test or Welch t test when the simulation passes the Levene tests. Then the simulation will use unconditional t tests including those two tests mentioned.

The Levene's test, is the hypothesis test on the equality of variances, i.e. it has the null hypothesis where $H_0 : \sigma_1^2 = \sigma_2^2$ and the alternative hypothesis where $H_a : \sigma_1^2 \neq \sigma_2^2$. The test statistic introduced by Levene (1960) is defined as

$$W = \frac{(N-k)}{(k-1)} \cdot \frac{\sum_{i=1}^{k} N_i (\bar{Z}_{i.} - \bar{Z}_{..})^2}{\sum_{i=1}^{k} \sum_{j=1}^{N_i} (Z_{ij} - \bar{Z}_{i.})^2} \tag{2.1}$$

where N is the sample size divided into k subgroups, and in this case, the paper has either 30 or 60 total sample size that divided into two groups as introduced in Introduction section. N_i is the sample size of the ith group, and Z_{ij} is defined using the mean of the ith group:

$$Z_{ij} = |Y_{ij} - \bar{Y}_{i.}| \tag{2.2}$$

where Y is the variable of interest in the simulation study with sample size N. $\bar{Z}_{i.}$ denotes the group means of Z_{ij}, $\bar{Z}_{..}$ denotes the overall mean of Z_{ij}. The Levene's test will reject the null hypothesis when

$$W > F_{\alpha, k-1, N-k}$$

such that F distribution has $k-1$ and $N-k$ degrees of freedom at significance level α. In this case, α will be determined using 0.5 or 0.1.

When the replication passes the Levene's test, that the replication is determined whether the variances are equal, the Student t test or Welch t test will be generated. The Student t test has the null hypothesis on two groups of mean are equal assuming two variances are equal, while Welch t test assumes two variances are unequal, i.e. $H_0 : \mu_1 = \mu_2$, and the alternative hypothesis is $H_1 : \mu_1 \neq \mu_2$. Then the computation of test statistic is performed via

$$T = \frac{\mu_1 - \mu_2}{\sqrt{s^2/n_1 + s^2/n_2}} \tag{2.3}$$

where μ_i represents the mean of each sample n_i and s^2 is an estimator of variance of two samples, and $T \overset{H_0}{\sim} N(0, \sigma^2)$ as each replication produces the pseudo-random number with mean at 0:

$$s^2 = \frac{\sum (x - \mu_1)^2 + \sum (x - \mu_2)^2}{n_1 + n_2 - 2} \tag{2.4}$$

Then we take the p-value to check if the test statistic is as extreme or more extreme than what is observed under H_0. The p-value in each replication is calculated as

$$\mathbb{P}(T > |T^*| \big| H_0)$$

The type I error for each test is calculated through

$$\mathbb{P}(\text{Reject } H_0 | H_0) = \alpha \tag{2.5}$$

With above settings, the simulation results are referred to Table 1.

3 Results

Table 3.1 illustrates the results of Type I error rates for each test, including Student t test, Welch t test, and preliminary Levene's test with conditions on both t tests, as well as the Levene's test itself. As shown in Table 3.1, the large sample has a closer Student t test Type I error to the α for both 0.01 and 0.05, however the value decreases as the increase in the standard deviation ratio. However, the Welch t test has all values close to the significance, except when the sample is extremely small where there is a slight increases as shown in Table 3.1 while $n_1 = 5$. As the standard error is 1.0, the Levene's test is close to the significance level, and close to 0 when $\alpha = 0.01$. The Levene's test has the function to test the equality of variances, so this is expected. When the condition of Levene's test rejects the null hypothesis, Welch t test is performed, so when the power of Levene's test is extremely lower than others, the conditional t tests have closer outcome to Student t tests; When the power of Levene's test is large enough, say near 1, the conditional t tests have more similar Type I error rates to Welch t tests. The table also shows that when the total sample size is small, there are more extreme increasing in the conditional t tests than larger total sample size.

As shown in Figure 3.1 and Figure 3.2, the plots show a detailed comparison between three t tests on the Type I error rates, as the standard error ranges from 1.0 to 2.4 with 0.2 increment each. The conditional t test on both graphs has a curve, it either increases in the beginning, then decreases gradually in the end, or vice versa. The Student t test, on the other hand, increases along with the ratio of standard errors. The Welch t test remains quite stable as the ratio of standard errors change and it always stays very close to the significance line (as shown a grey dotted line).

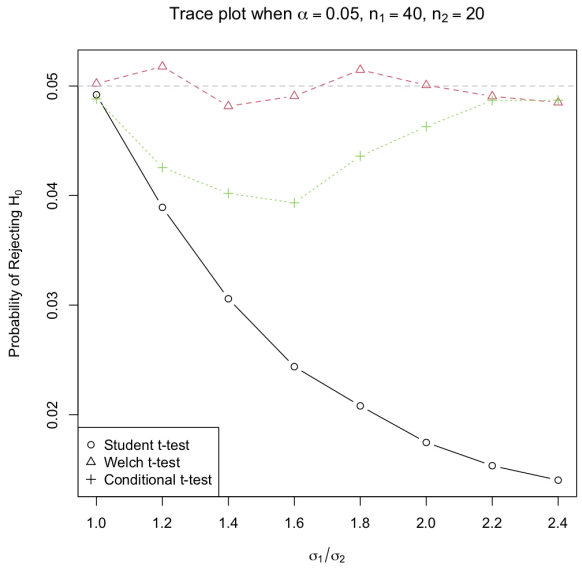

Figure 3.1: Type I Error rates of Student t test, Welch t test, and conditional t test with preliminary Levene's test. Larger sample size of n_1

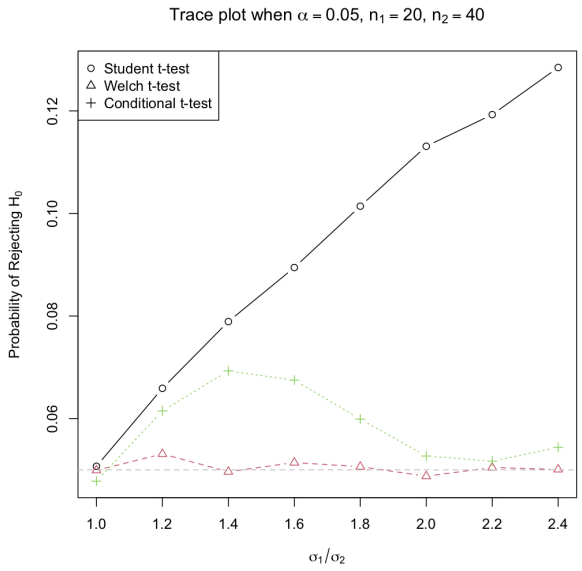

Figure 3.2: Type I Error rates of Student t test, Welch t test, and conditional t test with preliminary Levene's test. Smaller sample size of n_1

4 Discussion

This paper majorly reproduces the research work done by Zimmerman (2004) while there are more ad hoc mathematical interpretations within the method section. The software used within this paper is R programming, while the original paper used other softwares for the simulation. Results for both papers are similar, that both papers introduce the findings on the reason of abandoning the preliminary tests. As the Levene's test is a foregoing hypothesis testing method on the equality of variances prior to t tests, this causes the procedure to be fussy while two procedures are involved in a statistical hypothesis testing.

Table 4.1: Probability of rejecting H_0 for various combinations of sample sizes, significance levels, and SE ratios. S - Student t test, W - Welch t test, C - Choice conditional on preliminary Levene's test, L - Power of Levene's test.

			$\alpha = 0.01$				$\alpha = 0.05$			
n_1	n_2	σ_1/σ_2	S	W	C	L	S	W	C	L
50	10	1.0	0.011	0.011	0.011	0.007	0.050	0.051	0.052	0.042
		1.5	0.001	0.010	0.003	0.043	0.011	0.051	0.032	0.231
		2.0	0	0.012	0.004	0.175	0.004	0.051	0.039	0.523
		2.5	0	0.011	0.006	0.384	0.001	0.049	0.046	0.830
40	20	1.0	0.010	0.010	0.010	0.007	0.051	0.050	0.050	0.041
		1.5	0.003	0.010	0.005	0.139	0.026	0.051	0.040	0.402
		2.0	0.002	0.010	0.006	0.349	0.018	0.049	0.044	0.848
		2.5	0.001	0.011	0.008	0.849	0.014	0.049	0.049	0.978
20	40	1.0	0.010	0.010	0.009	0.007	0.050	0.050	0.051	0.042
		1.5	0.024	0.010	0.021	0.218	0.084	0.049	0.066	0.436
		2.0	0.038	0.011	0.020	0.676	0.110	0.050	0.057	0.864
		2.5	0.045	0.010	0.014	0.915	0.129	0.051	0.053	0.979
10	50	1.0	0.010	0.010	0.010	0.007	0.050	0.053	0.053	0.042
		1.5	0.047	0.010	0.041	0.147	0.136	0.051	0.100	0.299
		2.0	0.092	0.011	0.056	0.487	0.201	0.049	0.093	0.678
		2.5	0.133	0.012	0.044	0.753	0.253	0.050	0.071	0.879
25	5	1.0	0.011	0.016	0.010	0.006	0.050	0.059	0.055	0.030
		1.5	0.001	0.013	0.002	0.033	0.013	0.054	0.023	0.088
		2.0	0	0.012	0.001	0.121	0.004	0.052	0.022	0.191
		2.5	0	0.011	0.001	0.028	0.002	0.051	0.023	0.310
20	10	1.0	0.010	0.010	0.011	0.006	0.052	0.051	0.050	0.038
		1.5	0.004	0.009	0.004	0.068	0.027	0.050	0.034	0.169
		2.0	0.003	0.010	0.003	0.259	0.019	0.049	0.034	0.428
		2.5	0.002	0.010	0.004	0.251	0.015	0.048	0.038	0.666
10	20	1.0	0.010	0.011	0.010	0.006	0.050	0.050	0.050	0.039
		1.5	0.024	0.011	0.022	0.068	0.088	0.052	0.077	0.207
		2.0	0.038	0.011	0.032	0.259	0.114	0.050	0.081	0.508
		2.5	0.049	0.010	0.036	0.481	0.133	0.050	0.072	0.742
5	25	1.0	0.011	0.016	0.010	0.002	0.050	0.057	0.055	0.031
		1.5	0.046	0.017	0.046	0.030	0.129	0.057	0.125	0.108
		2.0	0.092	0.015	0.089	0.131	0.206	0.056	0.159	0.301
		2.5	0.134	0.014	0.110	0.281	0.262	0.054	0.157	0.498

REFERENCES

[1] Zimmerman D. W. (2004). A note on preliminary tests of equality of variances. *The British journal of mathematical and statistical psychology*, 57(Pt 1), 173–181. https://doi.org/10.1348/000711004849222

[2] Levene, H. (1960) Robust Tests for Equality of Variances. In: Olkin, I., Ed., *Contributions to Probability and Statistics*, Stanford University Press, Palo Alto, 278-292.

[3] Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

[4] Johnson, N. L., Kotz, S. and Balakrishnan, N. (1995) *Continuous Univariate Distributions*, volume 1, chapter 13. Wiley, New York.

APPENDIX

R CODE USED IN THIS SIMULATION STUDY

```
alpha_0.01 <- 0.01
alpha_0.05 <- 0.05
n.rep <- 50000
n1_60 <- c(50, 40, 20, 10)
n2_60 <- c(60-n1_60)
n1_30 <- c(25, 20, 10, 5)
n2_30 <- c(30-n1_30)
sd_num <- seq(1, 2.5, by=0.5)



######## Simulation n1 = c(50, 40, 20, 10), n2 = c(10, 20, 40, 50),
######## sigma1/sigma2 = 1.0, 1.5, 2.0, 2.5
##### Student T-Test n1+n2=60 #####

# alpha = 0.01 #
TypeIError_S0.01sd <- list(NULL, NULL, NULL, NULL)
TypeIError_S0.01 <- list(TypeIError_S0.01sd,
                         TypeIError_S0.01sd,
                         TypeIError_S0.01sd,
                         TypeIError_S0.01sd)
for (i in 1:4) {
  for (j in 1:4) {
    TypeIError_S0.01[[i]][[j]] <- replicate(n.rep,
                                           t.test(rnorm(n1_60[i], sd = sd_num[j]),
                                                  rnorm(n2_60[i], sd = 1),
                                                  alternative = "two.sided",
                                                  var.equal = TRUE)$p.value)
  }
}

mean_TypeIErrorS0.01_60 <- c()
for (i in 1:4) {
  for (j in 1:4) {
    mean_TypeIErrorS0.01_60 <- c(mean_TypeIErrorS0.01_60,
                                 mean(TypeIError_S0.01[[i]][[j]] < 0.01))
  }
}

# alpha = 0.05 #
TypeIError_S0.05sd <- list(NULL, NULL, NULL, NULL)
TypeIError_S0.05 <- list(TypeIError_S0.05sd,
                         TypeIError_S0.05sd,
                         TypeIError_S0.05sd,
                         TypeIError_S0.05sd)
```

```
for (i in 1:4) {
  for (j in 1:4) {
    TypeIError_S0.05[[i]][[j]] <- replicate(n.rep,
                                            t.test(rnorm(n1_60[i], sd = sd_num[j]),
                                                   rnorm(n2_60[i], sd = 1),
                                                   alternative = "two.sided",
                                                   var.equal = TRUE)$p.value)
  }
}

mean_TypeIErrorS0.05_60 <- c()
for (i in 1:4) {
  for (j in 1:4) {
    mean_TypeIErrorS0.05_60 <- c(mean_TypeIErrorS0.05_60,
                                 mean(TypeIError_S0.05[[i]][[j]] < 0.05))
  }
}

##### Welch T-Test n1+n2=60#####
# alpha = 0.01 #
TypeIError_W0.01sd <- list(NULL, NULL, NULL, NULL)
TypeIError_W0.01 <- list(TypeIError_W0.01sd,
                         TypeIError_W0.01sd,
                         TypeIError_W0.01sd,
                         TypeIError_W0.01sd)
for (i in 1:4) {
  for (j in 1:4) {
    TypeIError_W0.01[[i]][[j]] <- replicate(n.rep,
                                            t.test(rnorm(n1_60[i], sd = sd_num[j]),
                                                   rnorm(n2_60[i], sd = 1),
                                                   alternative = "two.sided",
                                                   var.equal = FALSE)$p.value)
  }
}

mean_TypeIErrorW0.01_60 <- c()
for (i in 1:4) {
  for (j in 1:4) {
    mean_TypeIErrorW0.01_60 <- c(mean_TypeIErrorW0.01_60,
                                 mean(TypeIError_W0.01[[i]][[j]] < 0.01))
  }
}

# alpha = 0.05 #
TypeIError_W0.05sd <- list(NULL, NULL, NULL, NULL)
TypeIError_W0.05 <- list(TypeIError_W0.05sd,
                         TypeIError_W0.05sd,
                         TypeIError_W0.05sd,
```

```
                          TypeIError_W0.05sd)
for (i in 1:4) {
  for (j in 1:4) {
    TypeIError_W0.05[[i]][[j]] <- replicate(n.rep,
                                           t.test(rnorm(n1_60[i], sd = sd_num[j]),
                                                  rnorm(n2_60[i], sd = 1),
                                                  alternative = "two.sided",
                                                  var.equal = FALSE)$p.value)
  }
}

mean_TypeIErrorW0.05_60 <- c()
for (i in 1:4) {
  for (j in 1:4) {
    mean_TypeIErrorW0.05_60 <- c(mean_TypeIErrorW0.05_60,
                                 mean(TypeIError_W0.05[[i]][[j]] < 0.05))
  }
}

##### Levene Test & Choice Conditional n1+n2=60, alpha = 0.01#####

# sd1/sd2 = 2 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.01_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
            {
    a <- rnorm(n1_60[i], sd = 2)
    b <- rnorm(n2_60[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.01_prelevene[[i]] <- c(TypeIError_0.01_prelevene[[i]],
                                        ifelse(
                                          leveneTest(
                                            values ~ ind, data = stacked)$`Pr(>F)` <
                              t.test(a, b, alternative = "two.sided", var.equal = FALSE)$
                              t.test(a, b, alternative = "two.sided", var.equal = TRUE)$p
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind,
                                               data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.01_prelevene2.0 <- c()
mean.TIE0.01_2.0 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.01_prelevene[[i]] <- TypeIError_0.01_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.01_2.0 <- c(mean.TIE0.01_2.0, mean(results[[i]] < 0.01))
  mean.TIE0.01_prelevene2.0 <- c(mean.TIE0.01_prelevene2.0,
```

```
                                  mean(TypeIError_0.01_prelevene[[i]] < 0.01))
}

# sd1/sd2 = 1.5 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.01_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_60[i], sd = 1.5)
    b <- rnorm(n2_60[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.01_prelevene[[i]] <- c(TypeIError_0.01_prelevene[[i]],
                                        ifelse(
                                          leveneTest(
                                            values ~ ind, data = stacked)$`Pr(>F)` < (
                                            t.test(a, b, alternative = "two.sided", va
                                            t.test(a, b, alternative = "two.sided", va
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.01_prelevene1.5 <- c()
mean.TIE0.01_1.5 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.01_prelevene[[i]] <- TypeIError_0.01_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.01_1.5 <- c(mean.TIE0.01_1.5, mean(results[[i]] < 0.01))
  mean.TIE0.01_prelevene1.5 <- c(mean.TIE0.01_prelevene1.5,
                                 mean(TypeIError_0.01_prelevene[[i]] < 0.01))
}

# sd1/sd2 = 1.0 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.01_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_60[i], sd = 1)
    b <- rnorm(n2_60[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.01_prelevene[[i]] <- c(TypeIError_0.01_prelevene[[i]],
                                        ifelse(
                                          leveneTest(
                                            values ~ ind, data = stacked)$`Pr(>F)` < (
                                            t.test(a, b, alternative = "two.sided", va
                                            t.test(a, b, alternative = "two.sided", va
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
```

```
  }
}

mean.TIE0.01_prelevene1.0 <- c()
mean.TIE0.01_1.0 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.01_prelevene[[i]] <- TypeIError_0.01_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.01_1.0 <- c(mean.TIE0.01_1.0, mean(results[[i]] < 0.01))
  mean.TIE0.01_prelevene1.0 <- c(mean.TIE0.01_prelevene1.0,
                                 mean(TypeIError_0.01_prelevene[[i]] < 0.01))
}

# sd1/sd2 = 2.5 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.01_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_60[i], sd = 2.5)
    b <- rnorm(n2_60[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.01_prelevene[[i]] <- c(TypeIError_0.01_prelevene[[i]],
                                        ifelse(
                                          leveneTest(
                                            values ~ ind, data = stacked)$`Pr(>F)` < 0
                                            t.test(a, b, alternative = "two.sided", va
                                            t.test(a, b, alternative = "two.sided", va
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.01_prelevene2.5 <- c()
mean.TIE0.01_2.5 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.01_prelevene[[i]] <- TypeIError_0.01_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.01_2.5 <- c(mean.TIE0.01_2.5, mean(results[[i]] < 0.01))
  mean.TIE0.01_prelevene2.5 <- c(mean.TIE0.01_prelevene2.5,
                                 mean(TypeIError_0.01_prelevene[[i]] < 0.01))
}


##### Levene Test & Choice Conditional n1+n2=60, alpha = 0.05#####

# sd1/sd2 = 2 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.05_prelevene <- list(NULL, NULL, NULL, NULL)
```

```
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_60[i], sd = 2)
    b <- rnorm(n2_60[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.05_prelevene[[i]] <- c(TypeIError_0.05_prelevene[[i]],
                                        ifelse(
                                          leveneTest(
                                            values ~ ind, data = stacked)$`Pr(>F)` < (
                                            t.test(a, b, alternative = "two.sided", va
                                            t.test(a, b, alternative = "two.sided", va
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.05_prelevene2.0 <- c()
mean.TIE0.05_2.0 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.05_prelevene[[i]] <- TypeIError_0.05_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.05_2.0 <- c(mean.TIE0.05_2.0, mean(results[[i]] < 0.05))
  mean.TIE0.05_prelevene2.0 <- c(mean.TIE0.05_prelevene2.0,
                                 mean(TypeIError_0.05_prelevene[[i]] < 0.05))
}

# sd1/sd2 = 1.5 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.05_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_60[i], sd = 1.5)
    b <- rnorm(n2_60[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.05_prelevene[[i]] <- c(TypeIError_0.05_prelevene[[i]],
                                        ifelse(leveneTest(values ~ ind, data = stacked
                                        t.test(a, b, alternative = "two.sided", var.eq
                                        t.test(a, b, alternative = "two.sided", var.eq
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.05_prelevene1.5 <- c()
mean.TIE0.05_1.5 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.05_prelevene[[i]] <- TypeIError_0.05_prelevene[[i]][!is.na(TypeIError_0
```

```
  mean.TIE0.05_1.5 <- c(mean.TIE0.05_1.5, mean(results[[i]] < 0.05))
  mean.TIE0.05_prelevene1.5 <- c(mean.TIE0.05_prelevene1.5,
                                 mean(TypeIError_0.05_prelevene[[i]] < 0.05))
}

# sd1/sd2 = 1.0 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.05_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_60[i], sd = 1)
    b <- rnorm(n2_60[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.05_prelevene[[i]] <- c(TypeIError_0.05_prelevene[[i]],
                                        ifelse(
                                          leveneTest(
                                            values ~ ind, data = stacked)$`Pr(>F)` < (
                                            t.test(a, b, alternative = "two.sided", va
                                            t.test(a, b, alternative = "two.sided", va
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.05_prelevene1.0 <- c()
mean.TIE0.05_1.0 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.05_prelevene[[i]] <- TypeIError_0.05_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.05_1.0 <- c(mean.TIE0.05_1.0, mean(results[[i]] < 0.05))
  mean.TIE0.05_prelevene1.0 <- c(mean.TIE0.05_prelevene1.0,
                                 mean(TypeIError_0.05_prelevene[[i]] < 0.05))
}

# sd1/sd2 = 2.5 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.05_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_60[i], sd = 2.5)
    b <- rnorm(n2_60[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.05_prelevene[[i]] <- c(TypeIError_0.05_prelevene[[i]],
                                        ifelse(
                                          leveneTest(
                                            values ~ ind, data = stacked)$`Pr(>F)` < (
                                            t.test(a, b, alternative = "two.sided", va
```

```
                                                t.test(a, b, alternative = "two.sided", va
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.05_prelevene2.5 <- c()
mean.TIE0.05_2.5 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.05_prelevene[[i]] <- TypeIError_0.05_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.05_2.5 <- c(mean.TIE0.05_2.5, mean(results[[i]] < 0.05))
  mean.TIE0.05_prelevene2.5 <- c(mean.TIE0.05_prelevene2.5,
                                 mean(TypeIError_0.05_prelevene[[i]] < 0.05))
}
```

```
######## Simulation n1 = c(25, 20, 10, 5), n2 = c(5, 10, 20, 25),
######## sigma1/sigma2 = 1.0

##### Student T-Test n1+n2=30#####
# alpha = 0.01 #
TypeIError_S0.01sd <- list(NULL, NULL, NULL, NULL)
TypeIError_S0.01 <- list(TypeIError_S0.01sd,
                         TypeIError_S0.01sd,
                         TypeIError_S0.01sd,
                         TypeIError_S0.01sd)
for (i in 1:4) {
  for (j in 1:4) {
    TypeIError_S0.01[[i]][[j]] <- replicate(n.rep,
                                            t.test(rnorm(n1_30[i], sd = sd_num[j]),
                                                   rnorm(n2_30[i], sd = 1),
                                                   alternative = "two.sided",
                                                   var.equal = TRUE)$p.value)
  }
}

mean_TypeIErrorS0.01_30 <- c()
for (i in 1:4) {
  for (j in 1:4) {
    mean_TypeIErrorS0.01_30 <- c(mean_TypeIErrorS0.01_30,
                                 mean(TypeIError_S0.01[[i]][[j]] < 0.01))
```

```
  }
}
# alpha = 0.05 #
TypeIError_S0.05sd <- list(NULL, NULL, NULL, NULL)
TypeIError_S0.05 <- list(TypeIError_S0.05sd,
                         TypeIError_S0.05sd,
                         TypeIError_S0.05sd,
                         TypeIError_S0.05sd)
for (i in 1:4) {
  for (j in 1:4) {
    TypeIError_S0.05[[i]][[j]] <- replicate(n.rep,
                                            t.test(rnorm(n1_30[i], sd = sd_num[j]),
                                                   rnorm(n2_30[i], sd = 1),
                                                   alternative = "two.sided",
                                                   var.equal = TRUE)$p.value)
  }
}

mean_TypeIErrorS0.05_30 <- c()
for (i in 1:4) {
  for (j in 1:4) {
    mean_TypeIErrorS0.05_30 <- c(mean_TypeIErrorS0.05_30,
                                 mean(TypeIError_S0.05[[i]][[j]] < 0.05))
  }
}

##### Welch T-Test n1+n2=30#####
# alpha = 0.01 #
TypeIError_W0.01sd <- list(NULL, NULL, NULL, NULL)
TypeIError_W0.01 <- list(TypeIError_W0.01sd,
                         TypeIError_W0.01sd,
                         TypeIError_W0.01sd,
                         TypeIError_W0.01sd)
for (i in 1:4) {
  for (j in 1:4) {
    TypeIError_W0.01[[i]][[j]] <- replicate(n.rep,
                                            t.test(rnorm(n1_30[i], sd = sd_num[j]),
                                                   rnorm(n2_30[i], sd = 1),
                                                   alternative = "two.sided",
                                                   var.equal = FALSE)$p.value)
  }
}

mean_TypeIErrorW0.01_30 <- c()
for (i in 1:4) {
  for (j in 1:4) {
    mean_TypeIErrorW0.01_30 <- c(mean_TypeIErrorW0.01_30,
                                 mean(TypeIError_W0.01[[i]][[j]] < 0.01))
```

```
  }
}
# alpha = 0.05 #
TypeIError_W0.05sd <- list(NULL, NULL, NULL, NULL)
TypeIError_W0.05 <- list(TypeIError_W0.05sd,
                         TypeIError_W0.05sd,
                         TypeIError_W0.05sd,
                         TypeIError_W0.05sd)
for (i in 1:4) {
  for (j in 1:4) {
    TypeIError_W0.05[[i]][[j]] <- replicate(n.rep,
                                            t.test(rnorm(n1_30[i], sd = sd_num[j]),
                                                   rnorm(n2_30[i], sd = 1),
                                                   alternative = "two.sided",
                                                   var.equal = FALSE)$p.value)
  }
}

mean_TypeIErrorW0.05_30 <- c()
for (i in 1:4) {
  for (j in 1:4) {
    mean_TypeIErrorW0.05_30 <- c(mean_TypeIErrorW0.05_30,
                                 mean(TypeIError_W0.05[[i]][[j]] < 0.05))
  }
}

##### Levene Test & Choice Conditional n1+n2=30, alpha = 0.01 #####
# sd1/sd2 = 2 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.01_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_30[i], sd = 2)
    b <- rnorm(n2_30[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.01_prelevene[[i]] <- c(TypeIError_0.01_prelevene[[i]],
                                        ifelse(
                                          leveneTest(
                                            values ~ ind, data = stacked)$`Pr(>F)` < 
                                            t.test(a, b, alternative = "two.sided", va
                                            t.test(a, b, alternative = "two.sided", va
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.01_prelevene2.0_30 <- c()
mean.TIE0.01_2.0_30 <- c()
```

```
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.01_prelevene[[i]] <- TypeIError_0.01_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.01_2.0_30 <- c(mean.TIE0.01_2.0_30, mean(results[[i]] < 0.01))
  mean.TIE0.01_prelevene2.0_30 <- c(mean.TIE0.01_prelevene2.0_30,
                                    mean(TypeIError_0.01_prelevene[[i]] < 0.01))
}

# sd1/sd2 = 1.5 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.01_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_30[i], sd = 1.5)
    b <- rnorm(n2_30[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.01_prelevene[[i]] <- c(TypeIError_0.01_prelevene[[i]],
                                        ifelse(
                                          leveneTest(
                                            values ~ ind, data = stacked)$`Pr(>F)` < 0
                                            t.test(a, b, alternative = "two.sided", va
                                            t.test(a, b, alternative = "two.sided", va
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.01_prelevene1.5_30 <- c()
mean.TIE0.01_1.5_30 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.01_prelevene[[i]] <- TypeIError_0.01_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.01_1.5_30 <- c(mean.TIE0.01_1.5_30, mean(results[[i]] < 0.01))
  mean.TIE0.01_prelevene1.5_30 <- c(mean.TIE0.01_prelevene1.5_30,
                                    mean(TypeIError_0.01_prelevene[[i]] < 0.01))
}

# sd1/sd2 = 1.0 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.01_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_30[i], sd = 1)
    b <- rnorm(n2_30[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.01_prelevene[[i]] <- c(TypeIError_0.01_prelevene[[i]],
                                        ifelse(leveneTest(values ~ ind, data = stacked
```

```
                                        t.test(a, b, alternative = "two.sided", var.eq
                                        t.test(a, b, alternative = "two.sided", var.eq
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.01_prelevene1.0_30 <- c()
mean.TIE0.01_1.0_30 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.01_prelevene[[i]] <- TypeIError_0.01_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.01_1.0_30 <- c(mean.TIE0.01_1.0_30, mean(results[[i]] < 0.01))
  mean.TIE0.01_prelevene1.0_30 <- c(mean.TIE0.01_prelevene1.0_30,
                                    mean(TypeIError_0.01_prelevene[[i]] < 0.01))
}

# sd1/sd2 = 2.5 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.01_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_30[i], sd = 2.5)
    b <- rnorm(n2_30[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.01_prelevene[[i]] <- c(TypeIError_0.01_prelevene[[i]],
                                        ifelse(leveneTest(values ~ ind, data = stacked
                                        t.test(a, b, alternative = "two.sided", var.eq
                                        t.test(a, b, alternative = "two.sided", var.eq
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.01_prelevene2.5_30 <- c()
mean.TIE0.01_2.5_30 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.01_prelevene[[i]] <- TypeIError_0.01_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.01_2.5_30 <- c(mean.TIE0.01_2.5_30, mean(results[[i]] < 0.01))
  mean.TIE0.01_prelevene2.5_30 <- c(mean.TIE0.01_prelevene2.5_30,
                                    mean(TypeIError_0.01_prelevene[[i]] < 0.01))
}

##### Levene Test & Choice Conditional n1+n2=30, alpha = 0.05 #####
# sd1/sd2 = 1.0 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.05_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
```

```
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_30[i], sd = 1)
    b <- rnorm(n2_30[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.05_prelevene[[i]] <- c(TypeIError_0.05_prelevene[[i]],
                                        ifelse(leveneTest(values ~ ind, data = stacked
                                        t.test(a, b, alternative = "two.sided", var.eq
                                        t.test(a, b, alternative = "two.sided", var.eq
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.05_prelevene1.0_30 <- c()
mean.TIE0.05_1.0_30 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.05_prelevene[[i]] <- TypeIError_0.05_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.05_1.0_30 <- c(mean.TIE0.05_1.0_30, mean(results[[i]] < 0.05))
  mean.TIE0.05_prelevene1.0_30 <- c(mean.TIE0.05_prelevene1.0_30,
                                    mean(TypeIError_0.05_prelevene[[i]] < 0.05))
}

# sd1/sd2 = 1.5 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.05_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_30[i], sd = 1.5)
    b <- rnorm(n2_30[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.05_prelevene[[i]] <- c(TypeIError_0.05_prelevene[[i]],
                                        ifelse(
                                          leveneTest(
                                            values ~ ind, data = stacked)$`Pr(>F)` < 0
                                            t.test(a, b, alternative = "two.sided", va
                                            t.test(a, b, alternative = "two.sided", va
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.05_prelevene1.5_30 <- c()
mean.TIE0.05_1.5_30 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.05_prelevene[[i]] <- TypeIError_0.05_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.05_1.5_30 <- c(mean.TIE0.05_1.5_30, mean(results[[i]] < 0.05))
```

```
  mean.TIE0.05_prelevene1.5_30 <- c(mean.TIE0.05_prelevene1.5_30,
                                    mean(TypeIError_0.05_prelevene[[i]] < 0.05))
}

# sd1/sd2 = 2.0 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.05_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_30[i], sd = 2.0)
    b <- rnorm(n2_30[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.05_prelevene[[i]] <- c(TypeIError_0.05_prelevene[[i]],
                                        ifelse(leveneTest(values ~ ind, data = stacked
                                        t.test(a, b, alternative = "two.sided", var.eq
                                        t.test(a, b, alternative = "two.sided", var.eq
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`
  }
}

mean.TIE0.05_prelevene2.0_30 <- c()
mean.TIE0.05_2.0_30 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.05_prelevene[[i]] <- TypeIError_0.05_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.05_2.0_30 <- c(mean.TIE0.05_2.0_30, mean(results[[i]] < 0.05))
  mean.TIE0.05_prelevene2.0_30 <- c(mean.TIE0.05_prelevene2.0_30,
                                    mean(TypeIError_0.05_prelevene[[i]] < 0.05))
}

# sd1/sd2 = 2.5 #
results <- list(NULL, NULL, NULL, NULL)
TypeIError_0.05_prelevene <- list(NULL, NULL, NULL, NULL)
for (i in 1:4) {
  for (j in 1:n.rep)
  {
    a <- rnorm(n1_30[i], sd = 2.5)
    b <- rnorm(n2_30[i], sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.05_prelevene[[i]] <- c(TypeIError_0.05_prelevene[[i]],
                                        ifelse(leveneTest(values ~ ind, data = stacked
                                        t.test(a, b, alternative = "two.sided", var.eq
                                        t.test(a, b, alternative = "two.sided", var.eq
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`
  }
}
```

```
mean.TIE0.05_prelevene2.5_30 <- c()
mean.TIE0.05_2.5_30 <- c()
for (i in 1:4) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.05_prelevene[[i]] <- TypeIError_0.05_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.05_2.5_30 <- c(mean.TIE0.05_2.5_30, mean(results[[i]] < 0.05))
  mean.TIE0.05_prelevene2.5_30 <- c(mean.TIE0.05_prelevene2.5_30,
                                    mean(TypeIError_0.05_prelevene[[i]] < 0.05))
}

########## For the GRAPH! ###############
### Student t-test SE ratio 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4 ###
## 40 20 ##
graph_se <- seq(1.0, 2.4, by = 0.2)
TypeIError_S0.05_n1 <- list(NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL)
for (i in 1:8) {

    TypeIError_S0.05_n1[[i]] <- replicate(n.rep,
                                          t.test(rnorm(40, sd = graph_se[i]),
                                                 rnorm(20, sd = 1),
                                                 alternative = "two.sided",
                                                 var.equal = TRUE)$p.value)

}

mean_TypeIErrorS0.05_n1 <- c()
for (i in 1:8) {

    mean_TypeIErrorS0.05_n1 <- c(mean_TypeIErrorS0.05_n1,
                                 mean(TypeIError_S0.05_n1[[i]] < 0.05))

}
## 20 40 ##
TypeIError_S0.05_n2 <- list(NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL)
for (i in 1:8) {

  TypeIError_S0.05_n2[[i]] <- replicate(n.rep,
                                        t.test(rnorm(20, sd = graph_se[i]),
                                               rnorm(40, sd = 1),
                                               alternative = "two.sided",
                                               var.equal = TRUE)$p.value)

}

mean_TypeIErrorS0.05_n2 <- c()
for (i in 1:8) {

  mean_TypeIErrorS0.05_n2 <- c(mean_TypeIErrorS0.05_n2,
```

```
                              mean(TypeIError_S0.05_n2[[i]] < 0.05))

}



### Welch t-test SE ratio 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4 ###
## 40 20 ##
TypeIError_W0.05_n1 <- list(NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL)
for (i in 1:8) {

  TypeIError_W0.05_n1[[i]] <- replicate(n.rep,
                                        t.test(rnorm(40, sd = graph_se[i]),
                                               rnorm(20, sd = 1),
                                               alternative = "two.sided",
                                               var.equal = FALSE)$p.value)

}

mean_TypeIErrorW0.05_n1 <- c()
for (i in 1:8) {

  mean_TypeIErrorW0.05_n1 <- c(mean_TypeIErrorW0.05_n1,
                               mean(TypeIError_W0.05_n1[[i]] < 0.05))

}
## 20 40 ##
TypeIError_W0.05_n2 <- list(NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL)
for (i in 1:8) {

  TypeIError_W0.05_n2[[i]] <- replicate(n.rep,
                                        t.test(rnorm(20, sd = graph_se[i]),
                                               rnorm(40, sd = 1),
                                               alternative = "two.sided",
                                               var.equal = FALSE)$p.value)

}

mean_TypeIErrorW0.05_n2 <- c()
for (i in 1:8) {

  mean_TypeIErrorW0.05_n2 <- c(mean_TypeIErrorW0.05_n2,
                               mean(TypeIError_W0.05_n2[[i]] < 0.05))

}

### Levene Conditional t-test SE ratio 1.0, 1.2, 1.4, 1.6, 1.8, 2.0, 2.2, 2.4 ###
## 40 20 ##
```

```
graph_se <- seq(1.0, 2.4, by = 0.2)
results <- list(NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL)
TypeIError_0.05_prelevene <- list(NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL)
for (i in 1:8) {
  for (j in 1:n.rep) {
    a <- rnorm(40, sd = graph_se[i])
    b <- rnorm(20, sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.05_prelevene[[i]] <- c(TypeIError_0.05_prelevene[[i]],
                                        ifelse(
                                          leveneTest(
                                            values ~ ind, data = stacked)$`Pr(>F)` < (
                                            t.test(a, b, alternative = "two.sided", va
                                            t.test(a, b, alternative = "two.sided", va
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.05_prelevene_n1 <- c()
mean.TIE0.05_n1 <- c()
for (i in 1:8) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.05_prelevene[[i]] <- TypeIError_0.05_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.05_n1 <- c(mean.TIE0.05_n1, mean(results[[i]] < 0.05))
  mean.TIE0.05_prelevene_n1 <- c(mean.TIE0.05_prelevene_n1,
                                 mean(TypeIError_0.05_prelevene[[i]] < 0.05))
}
plot(mean.TIE0.05_prelevene_n1)
## 20 40 ##
graph_se <- seq(1.0, 2.4, by = 0.2)
results <- list(NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL)
TypeIError_0.05_prelevene <- list(NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL)
for (i in 1:8) {
  for (j in 1:10000) {
    a <- rnorm(20, sd = graph_se[i])
    b <- rnorm(40, sd = 1)
    stacked <- stack(list(a = a, b = b))
    TypeIError_0.05_prelevene[[i]] <- c(TypeIError_0.05_prelevene[[i]],
                                        ifelse(
                                          leveneTest(
                                            values ~ ind, data = stacked)$`Pr(>F)` < (
                                            t.test(a, b, alternative = "two.sided", va
                                            t.test(a, b, alternative = "two.sided", va
    results[[i]] <- c(results[[i]], leveneTest(values ~ ind, data = stacked)$`Pr(>F)`)
  }
}

mean.TIE0.05_prelevene_n2 <- c()
```

```
mean.TIE0.05_n2 <- c()
for (i in 1:8) {
  results[[i]] <- results[[i]][!is.na(results[[i]])]
  TypeIError_0.05_prelevene[[i]] <- TypeIError_0.05_prelevene[[i]][!is.na(TypeIError_0
  mean.TIE0.05_n2 <- c(mean.TIE0.05_n2, mean(results[[i]] < 0.05))
  mean.TIE0.05_prelevene_n2 <- c(mean.TIE0.05_prelevene_n2,
                                 mean(TypeIError_0.05_prelevene[[i]] < 0.05))
}

## 40 20 graph ##
graph_n1 <- data.frame(mean_TypeIErrorS0.05_n1,
                       mean_TypeIErrorW0.05_n1,
                       mean.TIE0.05_prelevene_n1) %>%
  mutate(ratio = graph_se) %>%
  rename(Student = mean_TypeIErrorS0.05_n1,
         Welch = mean_TypeIErrorW0.05_n1,
         Conditional = mean.TIE0.05_prelevene_n1)
x <- cbind(graph_n1$ratio, graph_n1$ratio, graph_n1$ratio)
y <- cbind(graph_n1$Student, graph_n1$Welch, graph_n1$Conditional)
matplot(x, y, type = "b", pch=c(1,2,3), xlab = TeX("$\\sigma_1/\\sigma_2$"),
        ylab = TeX("Probability of Rejecting $H_0$"),
        main= TeX("Trace plot when $\\alpha=0.05$, $n_1=40$, $n_2=20$"))
abline(h=0.05, col="grey", lty = 2)
legend("bottomleft", legend=c(TeX("Student $t$-test"),
                              TeX("Welch $t$-test"),
                              TeX("Conditional $t$-test")),
       pch=c(1,2,3))


## 20 40 graph ##
graph_n2 <- data.frame(mean_TypeIErrorS0.05_n2,
                       mean_TypeIErrorW0.05_n2,
                       mean.TIE0.05_prelevene_n2) %>%
  mutate(ratio = graph_se) %>%
  rename(Student = mean_TypeIErrorS0.05_n2,
         Welch = mean_TypeIErrorW0.05_n2,
         Conditional = mean.TIE0.05_prelevene_n2)
x2 <- cbind(graph_n2$ratio, graph_n2$ratio, graph_n2$ratio)
y2 <- cbind(graph_n2$Student, graph_n2$Welch, graph_n2$Conditional)
matplot(x2, y2, type = "b", pch=c(1,2,3), xlab = TeX("$\\sigma_1/\\sigma_2$"),
        ylab = TeX("Probability of Rejecting $H_0$"),
        main= TeX("Trace plot when $\\alpha=0.05$, $n_1=20$, $n_2=40$"))
abline(h=0.05, col="grey", lty = 2)
legend("topleft", legend=c(TeX("Student $t$-test"),
                           TeX("Welch $t$-test"),
                           TeX("Conditional $t$-test")), pch=c(1,2,3))
```