STA 6133: Homework 4

Ricardo Cortez & Ben Graf

Due 16 Apr 2021

1.

The one–sample t–test is a statistical procedure to test the hypotheses

$$H_0 : \mu = \mu_0 \quad \text{against} \quad H_1 : \mu \neq \mu_0$$

when the data are a random sample $X_1, \ldots, X_n$ from the $N(\mu, \sigma^2)$ distribution, with μ, σ^2 unknown and μ_0 a fixed number. It has been found that this test is robust to mild departures from normality. The goal of this problem is to assess the robustness of the test to strong departures from normality, by assuming the data are a random sample from one of the following distributions

$$N(1, 2), \quad \chi^2_{(1)}, \quad unif(0, 2), \quad exp(1).$$

(a)

Assume the nominal significance level is $\alpha = 0.1 \text{ or } 0.05$. Use Monte Carlo simulation to investigate whether the empirical type I error of the t–test above is approximately equal to the nominal significance level when the data are a random sample of size $n = 30 \text{ or } 300$ from a distribution above.

```r
mu_0 <- 1   # The 4 distributions all have mean 1
set.seed(1)
M <- 100000
type1err <- data.frame("Normal" = rep(0,4), "Chi_squared" = rep(0,4), "Uniform" = rep(0,4),
                       "Exponential" = rep(0,4), "alpha" = rep(0,4), "n" = rep(0,4))
config <- 0   # Counts different alpha and n configurations
for (alpha in c(0.1, 0.05)) {
  for (n in c(30, 300)) {
    config <- config + 1
    #print(paste0("Configuration ",config))
    type1err[config,]$alpha <- alpha
    type1err[config,]$n <- n
    (crit_val <- qt(1-alpha/2, df = n-1))   # Critical value for t
    data1 <- matrix(rnorm(n*M, mean = 1, sd = sqrt(2)), nrow = M, ncol = n)
    data2 <- matrix(rchisq(n*M, df = 1), nrow = M, ncol = n)
    data3 <- matrix(runif(n*M, min = 0, max = 2), nrow = M, ncol = n)
    data4 <- matrix(rexp(n*M, rate = 1/1), nrow = M, ncol = n)
    t_test_stat1 <- (apply(data1, 1, mean) - mu_0) / (apply(data1, 1, sd) / sqrt(n))
    t_test_stat2 <- (apply(data2, 1, mean) - mu_0) / (apply(data2, 1, sd) / sqrt(n))
    t_test_stat3 <- (apply(data3, 1, mean) - mu_0) / (apply(data3, 1, sd) / sqrt(n))
    t_test_stat4 <- (apply(data4, 1, mean) - mu_0) / (apply(data4, 1, sd) / sqrt(n))
    type1err[config,1] <- mean(abs(t_test_stat1) > crit_val)
    type1err[config,2] <- mean(abs(t_test_stat2) > crit_val)
    type1err[config,3] <- mean(abs(t_test_stat3) > crit_val)
    type1err[config,4] <- mean(abs(t_test_stat4) > crit_val)
  }
}

knitr::kable(type1err, format = "html", table.attr = "style='width:50%;'")
```

Normal	Chi_squared	Uniform	Exponential	alpha	n
0.10151	0.13800	0.10064	0.12008	0.10	30
0.09888	0.10319	0.10130	0.10363	0.10	300
0.05008	0.09219	0.04993	0.07308	0.05	30
0.05013	0.05592	0.05025	0.05287	0.05	300

For the first two rows in the table above, the nominal type I error is 0.1, and for the latter two rows, the nominal type I error is 0.05, so those are the “targets” for our investigation. For the Normal and Uniform distributions, all four configurations of n and α are approximately equal to the nominal type I error. It does not appear to matter which sample size (n) is used. The Chi-squared and Exponential distributions, on the other hand, are *not* approximately equal to the nominal type I error; they do especially poorly for $n = 30$ and better, but still not great, for $n = 300$. The t-test on these two distributions results in rejecting the null hypothesis too aggressively (the type I error is higher than desired).

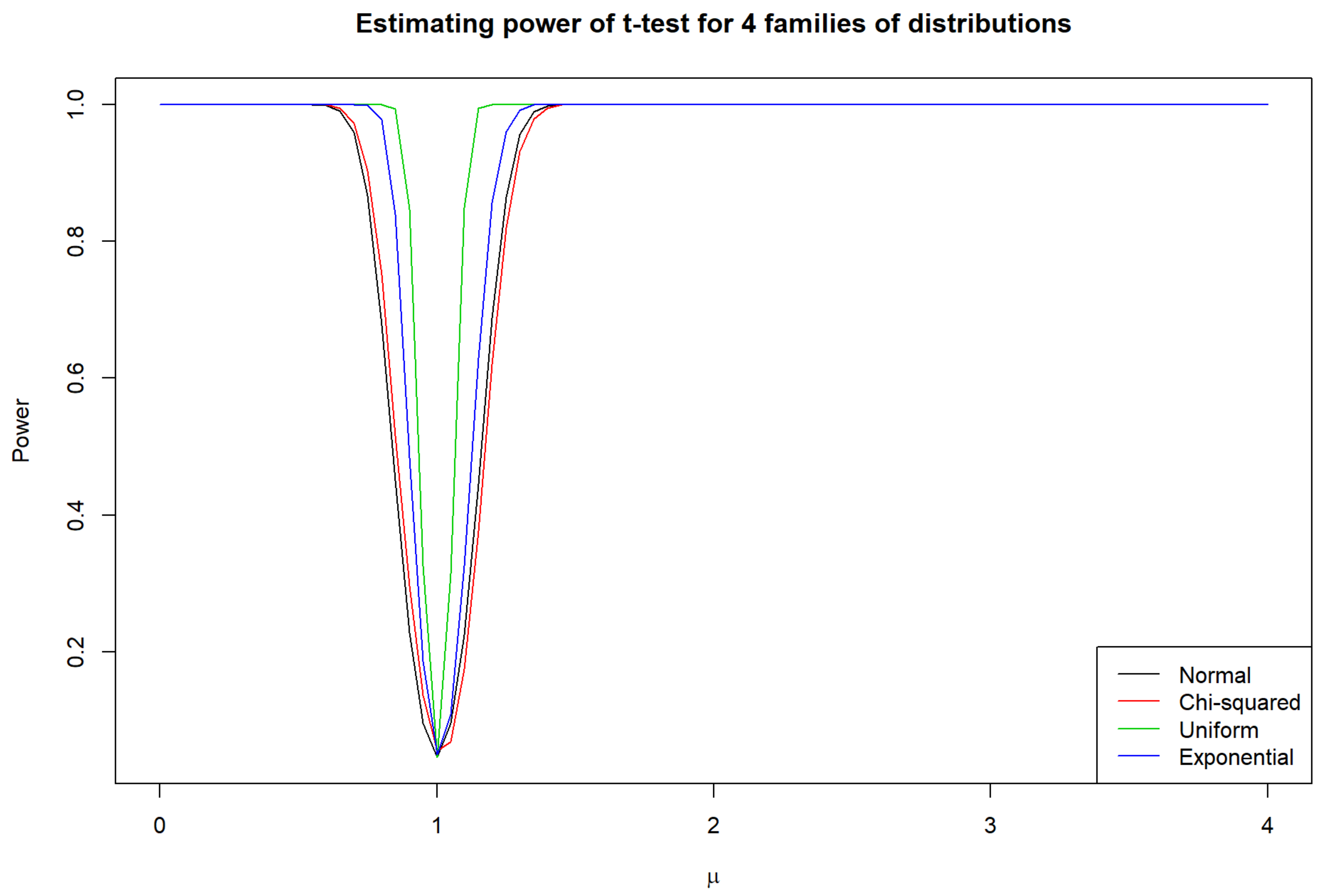
(b)

For the t–test with nominal significance level $\alpha = 0.05$ and $n = 300$, use Monte Carlo simulation to estimate the power functions when the data are from a family of distributions like those above, but with mean $\mu \in (0, 4)$.

```r
alpha <- 0.05
n <- 300
set.seed(1)
M <- 10000
mu_range <- seq(from = 0, to = 4, by = 0.05)   # Range of "true" mu to estimate power for
crit_val <- qt(1-alpha/2, df = n-1)   # Critical value for t
power <- data.frame("norm" = rep(0,length(mu_range)), "chisq" = rep(0,length(mu_range)),
                    "unif" = rep(0,length(mu_range)), "exp" = rep(0,length(mu_range)), "mu" = mu
_range)

for (j in 1:length(mu_range)) {
  # The 4 functions have to be shifted to the current mean=mu being evaluated
  data1 <- matrix(rnorm(n*M, mean = mu_range[j], sd = sqrt(2)), nrow = M, ncol = n)
  data2 <- matrix(rchisq(n*M, df = mu_range[j]), nrow = M, ncol = n)
  data3 <- matrix(runif(n*M, min = mu_range[j]-1, max = mu_range[j]+1), nrow = M, ncol = n)
  data4 <- matrix(rexp(n*M, rate = 1/mu_range[j]), nrow = M, ncol = n)
  t_test_stat1 <- (apply(data1, 1, mean) - mu_0) / (apply(data1, 1, sd)/sqrt(n))
  t_test_stat2 <- (apply(data2, 1, mean) - mu_0) / (apply(data2, 1, sd)/sqrt(n))
  t_test_stat3 <- (apply(data3, 1, mean) - mu_0) / (apply(data3, 1, sd)/sqrt(n))
  t_test_stat4 <- (apply(data4, 1, mean) - mu_0) / (apply(data4, 1, sd)/sqrt(n))
  power[j,1] <- mean(abs(t_test_stat1) > crit_val)
  power[j,2] <- mean(abs(t_test_stat2) > crit_val)
  power[j,3] <- mean(abs(t_test_stat3) > crit_val)
  power[j,4] <- mean(abs(t_test_stat4) > crit_val)
  #if (j %% 8 == 0) {
  #  print(j)
  #}
}

plot(x = power$mu, y = power$norm, type = "l", lty = 1, col = 1,
     xlab = expression(mu), ylab = "Power",
     main = "Estimating power of t-test for 4 families of distributions")
lines(x = power$mu, y = power$chisq, type = "l", lty = 1, col = 2)
lines(x = power$mu, y = power$unif, type = "l", lty = 1, col = 3)
lines(x = power$mu, y = power$exp, type = "l", lty = 1, col = 4)
legend("bottomright", c("Normal", "Chi-squared", "Uniform", "Exponential"), lty = 1, col = 1:4)
```



With $n = 300$, the power functions are more similar than they would be at $n = 30$. The Uniform has the tightest dip (in terms of width) at $\mu = 1$, followed by the Exponential. The Normal and Chi-squared are very similar in width. The t-test is more powerful as μ nears 1 for the Uniform and Exponential distributions than for Normal.

2.

Let $X_1, \dots, X_n$ be a random sample from a distribution that is symmetric about θ, meaning that $X - \theta = \theta - X$. For continuous random variables this means that for every $x \in \mathbb{R}$ the cdf of X satisfies $F(\theta - x) = 1 - F(\theta + x)$ and the pdf satisfies $f(\theta - x) = f(\theta + x)$. For any of these distributions we have $E_\theta(X) = med_\theta(X) = \theta$ (provided $E_\theta(X)$ exists), so two natural estimators of θ are $\overline{x}$ and $M =$ sample median of $X_1, \dots, X_n$. A third estimator of θ is the α–trimmed mean, defined as

$$\overline{X}_\alpha = \frac{1}{n - 2k} \sum_{i=k+1}^{n-k} X_{(i)}$$

where $\alpha \in (0, 1/2), k = \lfloor n\alpha \rfloor = (\text{integer part of } n\alpha)$ and $X_{(1)}, \dots, X_{(n)}$ are the order statistics. For the questsion below use as a sample size $n = 30$ and as simluation size $M = 100000$.

(a)

Consider the following families of distributions:

$$\{N(\theta, 3) : \theta \in \mathbb{R}\}, \quad \{Dexp(\theta, \sqrt{3/2}) : \theta \in \mathbb{R}\}, \quad \{t_3(\theta, 1) : \theta \in \mathbb{R}\}.$$

For each of these families compute the mean square error of the estimators $\overline{X}, M, \text{and } \overline{X}_{0.1}$, and comment on how these estimators compare.

```r
n <- 30
set.seed(1)
M <- 100000
theta <- 0   # MSEs should be constant for theta, so picking 0 for convenience

# Normal
data_a <- matrix(rnorm(n*M, mean = theta, sd = sqrt(3)), nrow = M, ncol = n)
# Double exponential (Dexp)
data_b <- matrix(rdexp(n*M, location = theta, scale = sqrt(3/2)), nrow = M, ncol = n)
# Scaled T - for this function, sd is defined as "Scale factor for the shifted, scaled distribut
ion"
data_c <- matrix(rt.scaled(n*M, df = 3, mean = theta, sd = 1), nrow = M, ncol = n)

# Function to estimate theta 3 ways and calculate MSE for each
MSEvec.func <- function(data, theta) {
  # data: Matrix of random data with each row containing a size n sample.
  #       Has M such rows and will estimate theta across these M Monte Carlo runs.
  # theta: The true value of the parameter being estimated.

  est_mean <- apply(data, 1, mean)
  est_median <- apply(data, 1, median)
  est_trimmean <- apply(data, 1, mean, trim = 0.1)
  MSE_mean <- mean((est_mean - theta)^2)
  MSE_median <- mean((est_median - theta)^2)
  MSE_trimmean <- mean((est_trimmean - theta)^2)

  # Returns vector of the MSE of 3 estimates of theta:  mean, median, and trimmed mean (alpha=0.
1)
  return(c(MSE_mean, MSE_median, MSE_trimmean))
}

MSEdf <- as.data.frame(rbind(MSEvec.func(data_a, theta),
                             MSEvec.func(data_b, theta),
                             MSEvec.func(data_c, theta)))
names(MSEdf) <- c("MSE_mean", "MSE_median", "MSE_trimmean")
rownames(MSEdf) <- c("Normal", "DExp", "Scaled T")
knitr::kable(MSEdf, format = "html", table.attr = "style='width:40%;'")
```

	MSE_mean	MSE_median	MSE_trimmean
Normal	0.1004040	0.1502965	0.1059112
DExp	0.1005668	0.0637666	0.0769642
Scaled T	0.0986050	0.0607286	0.0556150

For the Normal distribution, the best estimator of θ appears to be the sample mean, with the trimmed mean nearly matching it; the median does a poor job by comparison. For the Exponential, the median is the clear winner, and the trimmed mean outperforms the sample mean noticeably. For the Scaled t_3, the trimmed mean is best, with the median not far behind and the sample mean bringing up the rear; this is likely due to the the heavier tails of the t distribution.

(b)

Consider now the family of distributions:

$$F_\theta(x) = (1-\epsilon)\Phi(\frac{x-\theta}{\sqrt{3}}) + \epsilon G_\theta(x),\ \theta \in \mathbb{R}$$

where G_θ is the cdf of the $t_3(\theta, 1)$ distribution and $\epsilon \in (0, 1)$. This mixture model, usually called a contamination model, indicates that on average $100(1-\epsilon)$% of the observation come from the $N(\theta, 3)$ distribution, while the rest come from the $t_3(\theta, 1)$ distribution. In this case we also have $E_\theta(X) = med_\theta(X) = \theta$. Compute the mean square error of the estimators $\overline{X}, M, \text{and } \overline{X}_{0.1}$,when ϵ is 0.1 and 0.3, and comment on how these estimators compare.

```
set.seed(1)
epsilon_range <- c(0.1, 0.3)
MSEmix <- matrix(nrow = length(epsilon_range), ncol = 3)
for (k in 1:length(epsilon_range)) {
  mixing <- rbinom(M, size = 1, prob = epsilon_range[k])   # First draw from mixing distribution
  data_d <- matrix(nrow = M, ncol = n)
  for (i in 1:M) {
    if (mixing[i]) {   # If mixing drew a 1, draw from Scaled T
      data_d[i,] <- rt.scaled(n, df = 3, mean = theta, sd = 1)
    } else {   # If mixing drew a 0, draw from Normal
      data_d[i,] <- rnorm(n, mean = theta, sd = sqrt(3))
    }
  }
  MSEmix[k,] <- MSEvec.func(data_d, theta)   # Calculate 3 MSEs using function from (a)
}
MSEmix <- as.data.frame(MSEmix)
names(MSEmix) <- c("MSE_mean", "MSE_median", "MSE_trimmean")
rownames(MSEmix) <- c("Epsilon_0.1", "Epsilon_0.3")
knitr::kable(MSEmix, format = "html", table.attr = "style='width:50%;'")
```

	MSE_mean	MSE_median	MSE_trimmean
Epsilon_0.1	0.1002968	0.1412649	0.1009672
Epsilon_0.3	0.0993781	0.1241358	0.0908865

When $\epsilon = 0.1$, the mixture is mostly Normal, so, as we found in (a), the sample mean is the best estimator of θ. By $\epsilon = 0.3$, though, the mixture has enough Scaled t_3 in it that the trimmed mean overtakes the sample mean as the best estimator of θ. (Interestingly, the MSE of all three estimators is lower for $\epsilon = 0.3$.)

3.

Suppose you have $k \geq 2$ independent random samples $X_{11}, X_{12}, \ldots, X_{1n_1};\ X_{21}, X_{22}, \ldots, X_{2n_2};\ \ldots\ldots;\ X_{k1}, X_{k2}, \ldots, X_{kn_k}$, with $n_i \geq 2$ and $\sigma_i^2 = \operatorname{var}(X_{ij}),\ i = 1, \ldots, k;\ j = 1, \ldots, n_i$. Consider testing the hypothesis of equality of variances:

$$H_0 : \sigma_1^2 = \ \ldots\ = \sigma_k^2,$$

against the alternative $H_1 : \sigma_{i_1}^2 \neq \sigma_{i_2}^2$ for some $i_1 \neq i_2$. A test to do this with (approximate) significance level α is *Bartlett's test*; see `bartlett.test` in `R`. The theory supporting this test assumes all the samples are normally distributed. It has been found that this test is non-robust to deviations form normality, and rejection of H_0 is often due to either differences in variance or non-normality. A more robust test is *Levene's test*; see `levene.test` in the `R` package `lawstat`. Suppose $k = 3,\ (n_1, n_2, n_3) = (10, 10, 20),\ \alpha = 0.05$, and all the random samples are from one of three families: Normal, t$_3$, or exponential. Without lost of generality, for the Normal and t$_3$ families set the all the means to zero. For the questions below use as simulation size $m = 10000$.

(a)

For the null models $(\sigma_1^2, \sigma_2^2, \sigma_3^2) = (1, 1, 1) \text{ and } (10, 10, 10)$ estimate the significance level of Bartlett's and Levene's tests for each of the three families of distributions.

```r
k <- 3
n <- c(10, 10, 20)
alpha <- 0.05
M <- 10000


# (a)

groups <- c(rep(1,n[1]), rep(2,n[2]), rep(3,n[3]))
bart.test <- function(vec, groups, alpha) {
  return(bartlett.test(vec ~ groups)$p.value < alpha)   # If p-value < alpha, reject null hypoth
esis
}
lev.test <- function(vec, groups, alpha) {
  return(levene.test(vec, groups, location = "mean")$p.value < alpha)   # If p-value < alpha, re
ject null hypothesis
}

set.seed(1)

results <- matrix(nrow = 3, ncol = 4)

# Normal
var_0 <- c(1,1,1)
dat1 <- matrix(rnorm(n[1]*M, mean = 0, sd = sqrt(var_0[1])), nrow = M, ncol = n[1])
dat2 <- matrix(rnorm(n[2]*M, mean = 0, sd = sqrt(var_0[2])), nrow = M, ncol = n[2])
dat3 <- matrix(rnorm(n[3]*M, mean = 0, sd = sqrt(var_0[3])), nrow = M, ncol = n[3])
comb_dat <- cbind(dat1, dat2, dat3)
results[1,1] <- mean(apply(comb_dat, 1, bart.test, groups, alpha))   # Should approximate alpha
 = 0.05
results[1,3] <- mean(apply(comb_dat, 1, lev.test, groups, alpha))   # Should approximate alpha =
0.05

var_0 <- c(10,10,10)
dat1 <- matrix(rnorm(n[1]*M, mean = 0, sd = sqrt(var_0[1])), nrow = M, ncol = n[1])
dat2 <- matrix(rnorm(n[2]*M, mean = 0, sd = sqrt(var_0[2])), nrow = M, ncol = n[2])
dat3 <- matrix(rnorm(n[3]*M, mean = 0, sd = sqrt(var_0[3])), nrow = M, ncol = n[3])
comb_dat <- cbind(dat1, dat2, dat3)
results[1,2] <- mean(apply(comb_dat, 1, bart.test, groups, alpha))   # Should approximate alpha
 = 0.05
results[1,4] <- mean(apply(comb_dat, 1, lev.test, groups, alpha))   # Should approximate alpha =
0.05

# Scaled T
var_0 <- c(1,1,1)
dat1 <- matrix(rt.scaled(n[1]*M, df = 3, mean = 0, sd = sqrt(var_0[1])), nrow = M, ncol = n[1])
dat2 <- matrix(rt.scaled(n[2]*M, df = 3, mean = 0, sd = sqrt(var_0[2])), nrow = M, ncol = n[2])
dat3 <- matrix(rt.scaled(n[3]*M, df = 3, mean = 0, sd = sqrt(var_0[3])), nrow = M, ncol = n[3])
comb_dat <- cbind(dat1, dat2, dat3)
results[2,1] <- mean(apply(comb_dat, 1, bart.test, groups, alpha))   # Should approximate alpha
 = 0.05
results[2,3] <- mean(apply(comb_dat, 1, lev.test, groups, alpha))   # Should approximate alpha =
0.05
```

```
var_0 <- c(10,10,10)
dat1 <- matrix(rt.scaled(n[1]*M, df = 3, mean = 0, sd = sqrt(var_0[1])), nrow = M, ncol = n[1])
dat2 <- matrix(rt.scaled(n[2]*M, df = 3, mean = 0, sd = sqrt(var_0[2])), nrow = M, ncol = n[2])
dat3 <- matrix(rt.scaled(n[3]*M, df = 3, mean = 0, sd = sqrt(var_0[3])), nrow = M, ncol = n[3])
comb_dat <- cbind(dat1, dat2, dat3)
results[2,2] <- mean(apply(comb_dat, 1, bart.test, groups, alpha))   # Should approximate alpha
 = 0.05
results[2,4] <- mean(apply(comb_dat, 1, lev.test, groups, alpha))   # Should approximate alpha =
0.05

# Exponential
var_0 <- c(1,1,1)
dat1 <- matrix(rexp(n[1]*M, rate = 1/sqrt(var_0[1])), nrow = M, ncol = n[1])
dat2 <- matrix(rexp(n[2]*M, rate = 1/sqrt(var_0[2])), nrow = M, ncol = n[2])
dat3 <- matrix(rexp(n[3]*M, rate = 1/sqrt(var_0[3])), nrow = M, ncol = n[3])
comb_dat <- cbind(dat1, dat2, dat3)
results[3,1] <- mean(apply(comb_dat, 1, bart.test, groups, alpha))   # Should approximate alpha
 = 0.05
results[3,3] <- mean(apply(comb_dat, 1, lev.test, groups, alpha))   # Should approximate alpha =
0.05

var_0 <- c(10,10,10)
dat1 <- matrix(rexp(n[1]*M, rate = 1/sqrt(var_0[1])), nrow = M, ncol = n[1])
dat2 <- matrix(rexp(n[2]*M, rate = 1/sqrt(var_0[2])), nrow = M, ncol = n[2])
dat3 <- matrix(rexp(n[3]*M, rate = 1/sqrt(var_0[3])), nrow = M, ncol = n[3])
comb_dat <- cbind(dat1, dat2, dat3)
results[3,2] <- mean(apply(comb_dat, 1, bart.test, groups, alpha))   # Should approximate alpha
 = 0.05
results[3,4] <- mean(apply(comb_dat, 1, lev.test, groups, alpha))   # Should approximate alpha =
0.05

# Results
results <- as.data.frame(results)
names(results) <- c("Bartlett (1,1,1)", "Bartlett (10,10,10)", "Levene (1,1,1)", "Levene (10,10,
10)")
rownames(results) <- c("Normal", "Scaled t", "Exponential")
knitr::kable(results, format = "html", table.attr = "style='width:60%;'")
```

	Bartlett (1,1,1)	**Bartlett (10,10,10)**	**Levene (1,1,1)**	**Levene (10,10,10)**
Normal	0.0524	0.0477	0.0592	0.0588
Scaled t	0.3438	0.3339	0.0777	0.0816
Exponential	0.3442	0.3372	0.1816	0.1853

Bartlett’s test appears superior when the data is in fact Normal; the significance levels for Levene’s test are a bit higher than they should be ideally. However, for the other two distributions, Bartlett’s test is a disaster; it’s non-robustness is borne out. Levene’s test is not terrible for the Scaled t_3 distribution. It is not good for Exponential, but it is far superior to Bartlett’s.

(b)

For the alternative models $(\sigma_1^2, \sigma_2^2, \sigma_3^2) = (1, 1, 3) \text{ and } (1, 2, 6)$ estimate the power of Bartlett's and Levene's tests for each of the three families of distributions.

```r
set.seed(1)
power3 <- matrix(nrow = 3, ncol = 4)

# Normal
var_0 <- c(1,1,3)
dat1 <- matrix(rnorm(n[1]*M, mean = 0, sd = sqrt(var_0[1])), nrow = M, ncol = n[1])
dat2 <- matrix(rnorm(n[2]*M, mean = 0, sd = sqrt(var_0[2])), nrow = M, ncol = n[2])
dat3 <- matrix(rnorm(n[3]*M, mean = 0, sd = sqrt(var_0[3])), nrow = M, ncol = n[3])
comb_dat <- cbind(dat1, dat2, dat3)
power3[1,1] <- mean(apply(comb_dat, 1, bart.test, groups, alpha))   # Gives power
power3[1,3] <- mean(apply(comb_dat, 1, lev.test, groups, alpha))   # Gives power

var_0 <- c(1,2,6)
dat1 <- matrix(rnorm(n[1]*M, mean = 0, sd = sqrt(var_0[1])), nrow = M, ncol = n[1])
dat2 <- matrix(rnorm(n[2]*M, mean = 0, sd = sqrt(var_0[2])), nrow = M, ncol = n[2])
dat3 <- matrix(rnorm(n[3]*M, mean = 0, sd = sqrt(var_0[3])), nrow = M, ncol = n[3])
comb_dat <- cbind(dat1, dat2, dat3)
power3[1,2] <- mean(apply(comb_dat, 1, bart.test, groups, alpha))   # Gives power
power3[1,4] <- mean(apply(comb_dat, 1, lev.test, groups, alpha))   # Gives power

# Scaled T
var_0 <- c(1,1,3)
dat1 <- matrix(rt.scaled(n[1]*M, df = 3, mean = 0, sd = sqrt(var_0[1])), nrow = M, ncol = n[1])
dat2 <- matrix(rt.scaled(n[2]*M, df = 3, mean = 0, sd = sqrt(var_0[2])), nrow = M, ncol = n[2])
dat3 <- matrix(rt.scaled(n[3]*M, df = 3, mean = 0, sd = sqrt(var_0[3])), nrow = M, ncol = n[3])
comb_dat <- cbind(dat1, dat2, dat3)
power3[2,1] <- mean(apply(comb_dat, 1, bart.test, groups, alpha))   # Gives power
power3[2,3] <- mean(apply(comb_dat, 1, lev.test, groups, alpha))   # Gives power

var_0 <- c(1,2,6)
dat1 <- matrix(rt.scaled(n[1]*M, df = 3, mean = 0, sd = sqrt(var_0[1])), nrow = M, ncol = n[1])
dat2 <- matrix(rt.scaled(n[2]*M, df = 3, mean = 0, sd = sqrt(var_0[2])), nrow = M, ncol = n[2])
dat3 <- matrix(rt.scaled(n[3]*M, df = 3, mean = 0, sd = sqrt(var_0[3])), nrow = M, ncol = n[3])
comb_dat <- cbind(dat1, dat2, dat3)
power3[2,2] <- mean(apply(comb_dat, 1, bart.test, groups, alpha))   # Gives power
power3[2,4] <- mean(apply(comb_dat, 1, lev.test, groups, alpha))   # Gives power

# Exponential
var_0 <- c(1,1,3)
dat1 <- matrix(rexp(n[1]*M, rate = 1/sqrt(var_0[1])), nrow = M, ncol = n[1])
dat2 <- matrix(rexp(n[2]*M, rate = 1/sqrt(var_0[2])), nrow = M, ncol = n[2])
dat3 <- matrix(rexp(n[3]*M, rate = 1/sqrt(var_0[3])), nrow = M, ncol = n[3])
comb_dat <- cbind(dat1, dat2, dat3)
power3[3,1] <- mean(apply(comb_dat, 1, bart.test, groups, alpha))   # Gives power
power3[3,3] <- mean(apply(comb_dat, 1, lev.test, groups, alpha))   # Gives power

var_0 <- c(1,2,6)
dat1 <- matrix(rexp(n[1]*M, rate = 1/sqrt(var_0[1])), nrow = M, ncol = n[1])
dat2 <- matrix(rexp(n[2]*M, rate = 1/sqrt(var_0[2])), nrow = M, ncol = n[2])
dat3 <- matrix(rexp(n[3]*M, rate = 1/sqrt(var_0[3])), nrow = M, ncol = n[3])
comb_dat <- cbind(dat1, dat2, dat3)
power3[3,2] <- mean(apply(comb_dat, 1, bart.test, groups, alpha))   # Gives power
power3[3,4] <- mean(apply(comb_dat, 1, lev.test, groups, alpha))   # Gives power
```

```
# Results
power3 <- as.data.frame(power3)
names(power3) <- c("Bartlett (1,1,3)", "Bartlett (1,2,6)", "Levene (1,1,3)", "Levene (1,2,6)")
rownames(power3) <- c("Normal", "Scaled t", "Exponential")
knitr::kable(power3, format = "html", table.attr = "style='width:60%;'")
```

	Bartlett (1,1,3)	**Bartlett (1,2,6)**	**Levene (1,1,3)**	**Levene (1,2,6)**
Normal	0.5006	0.8015	0.4294	0.6736
Scaled t	0.6394	0.7882	0.2308	0.3970
Exponential	0.6335	0.7831	0.4061	0.5649

Bartlett’s test is consistently more powerful than Levene’s test across all three distributions and both alternative models. Its power is similar between Scaled t_3 and Exponential. Levene’s test is more powerful for Normal than it is for Exponential or for Scaled t_3 (least of all).

(c)

Summarize in a concise way your findings in (a) and (b).

Bartlett’s test is preferred if you are confident your data is from a Normal distribution. It is the more powerful test, but it rejects *far* too aggressively if the distribution is anything other than Normal. That aggressiveness is a double-edged sword, responsible both for it being non-robust and for its higher power. Levene’s test achieves some robustness by being less aggressive, but its power suffers by comparison.

4.

The following are observations on the time (in hours) between failures of an air conditioning equipment:

$$3 \quad 5 \quad 7 \quad 18 \quad 43 \quad 85 \quad 91 \quad 98 \quad 100 \quad 130 \quad 230 \quad 487$$

(a)

Let μ be the expected value of the failure time. Assuming these data are a random sample from the exp(μ) distribution, use bootstrap to estimate the bias and standard deviation of the MLE of $g(\mu) = 1/\mu$.

```
fail_times <- c(3, 5, 7, 18, 43, 85, 91, 98, 100, 130, 230, 487)
xbar <- mean(fail_times)   # Sample mean
#xsd <- sd(fail_times)   # Sample standard deviation
n <- length(fail_times)   # Sample size
M <- 100000   # Simulation size

# (a)
set.seed(1)
pboot <- matrix(rexp(n*M, rate = 1/xbar), nrow = M, ncol = n)   # Parametric bootstrap sample
T.pboot <- 1/apply(pboot, 1, mean)   # M draws from bootstrap distribution of g(mu)=1/mu
#hist(T.pboot)
(bias_est <- mean(T.pboot) - (1/xbar))
```

```
## [1] 0.0008399412
```

```
(sd_est <- sd(T.pboot))
```

```
## [1] 0.003202506
```

```
a<- data.frame("Bias" = bias_est,"SD" = sd_est)
knitr::kable(a,format = "html", table.attr = "style='width:30%;'")
```

Bias	SD
0.0008399	0.0032025

The estimates of the bias and standard deviation of the MLE of $g(\mu) = 1/\mu$ are above.

(b)

Under the same assumption as in (a), compute exact, asymptotic and bootstrap 90% confidence intervals for $P_\mu(X > 100)$. Comment on the differences.

```r
alpha <- 0.1   # 90% confidence intervals

# Exact
exact_mu_CI <- c(xbar/qgamma(1-alpha/2, shape = n, scale = 1/n), xbar/qgamma(alpha/2, shape = n,
scale = 1/n))
ex <- exp(-100/exact_mu_CI)   # Transform exact CI of mu

# Asymptotic
asymp_mu_CI <- c(xbar+qnorm(alpha/2)*xbar/sqrt(n), xbar+qnorm(1-alpha/2)*xbar/sqrt(n))
as <- exp(-100/asymp_mu_CI)   # Transform asymptotic CI of mu

# Bootstrap
gt_100 <- function(x) {
  # x is a vector
  return(mean(x>100))   # Returns percentage of elements of x that are >100
}


gt_100_exp <- function(x) {
  # x is a vector
  return(exp(-100/mean(x)))   # Returns transformation of estimate of mu
}

gt_exp_fail <- gt_100_exp(fail_times)

Tb.pboot <- apply(pboot, 1, gt_100_exp)   # M draws from parametric bootstrap distribution of ex
p(-100/mean(x))
v <- unname(quantile(Tb.pboot, probs = c(1-alpha/2, alpha/2)))
bt <- c(2*gt_exp_fail - v[1], 2*gt_exp_fail - v[2])   # Bootstrap CI of P(X>100)

a <- data.frame(rbind(ex,as,bt),row.names = c("Exact CI", "Asymptotic CI", "Bootstrapped CI"))

knitr::kable(a,format = "html", table.attr = "style='width:40%;'",col.names = c("Lower","Upper"
))
```

	Lower	Upper
Exact CI	0.2456574	0.5863351
Asymptotic CI	0.1717470	0.5340128
Bootstrapped CI	0.2498122	0.5915037

The Exact confidence interval has the smallest width, which makes sense since it has the strongest assumptions. The confidence interval that was estimated with bootstrap is only slightly larger, approximating the exact interval quite well. The asymptotic confidence interval is somewhat larger and shifted to the left of the other two.

(c)

Assume now the data are a random sample from an unknown cdf F. compute the bootstrap-t percentile and BC$_a$ 90% confidence intervals for $P_F(X > 100)$. Comment on the differences.

```r
npboot <- matrix(sample(fail_times, size = n*M, replace = TRUE), nrow = M, ncol = n)   # Non-par
ametric bootstrap sample
Tc.npboot <- apply(npboot, 1, gt_100)   # M draws from non-parametric bootstrap distribution of
 P(X>100), this is a list of probabilities

# Bootstrap-t
#V <- apply(npboot, 1, sd)/sqrt(n)
gt_fail <- gt_100(fail_times)

V <- sqrt(Tc.npboot * (1-Tc.npboot) / n)   # Because T is a probability, its standard deviation
 is average of n Bernoullis = sqrt(pq/n)
u <- unname(quantile((Tc.npboot - gt_fail)/V, probs = c(1-alpha/2, alpha/2)))   # Bootstrap-t qu
antiles
sd_obs <- sqrt(gt_fail * (1-gt_fail) / n)   # Standard deviation of observed data is sqrt(pq/n)
 where p=P(X>100) for observed
Boot.t <- c(gt_fail - u[1]*sd_obs, gt_fail - u[2]*sd_obs)   # Bootstrap-t CI of P(X>100)

# Percentile
Percentile <- unname(quantile(Tc.npboot, probs = c(alpha/2, 1-alpha/2)))   # Bootstrap percentil
e CI of P(X>100)

# BCa
w <- qnorm(mean(Tc.npboot < gt_fail))
#k <- pnorm(2*w - qnorm(c(1-alpha/2, alpha/2)))
#quantile(Tc.npboot, probs = k)   # Bootstrap bias-corrected percentile CI of P(X>100)
TJ <- c()
for(i in 1:n) {
  TJ[i] <- gt_100(fail_times[-i])
}
TJm <- mean(TJ)
a <- (1/6)*sum((TJm - TJ)^3) / sum((TJm - TJ)^2)^1.5
z <- qnorm(c(alpha/2, 1-alpha/2))
l <- pnorm(w + (w + z)/(1 - a*(w + z)))

BCa <- quantile(Tc.npboot, probs = l)   # Bootstrap BCa CI of P(X>100)
BCa<-unname(BCa)

knitr::kable(data.frame(rbind(Boot.t,Percentile,BCa)),col.names = c("Lower","Upper"),format = "h
tml", table.attr = "style='width:40%;'")
```

	Lower	Upper
Boot.t	0.0334936	0.5111165
Percentile	0.0833333	0.5000000
BCa	0.0000000	0.4166667

As expected, all three non-parametric confidence intervals are larger than the parametric ones from (b) above. The Bootstrap-t interval is the largest of the three. In fact, the widths of the other two are the same. The BCa (which adjusts for bias) is shifted closer to 0.