HW7

Will Tirone

4/4/23

4.3

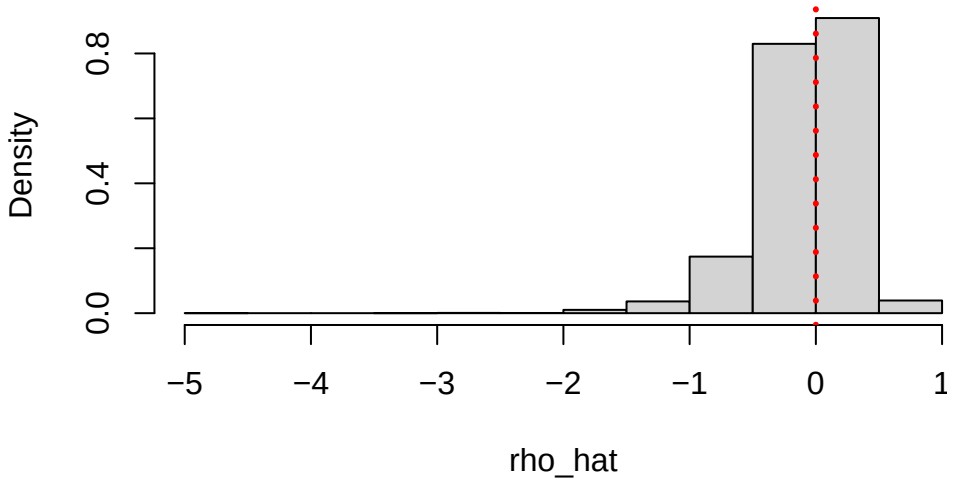
f)

From 4.2, we are setting n = m + k, so dividing this into two groups results in m = k = 1929. Assuming my n = 3858 from part d) is correct, it looks like the null hypothesis case where $\phi_1 = phi_2 = 0.01$ is very poorly approximated by the asymptotic limit, while $\phi_1 = \phi_2 = 0.99$ looks fairly close to normal. The same is true under the alternative, the less extreme case of $\phi_1 = 0.01, \phi_2 = 0.03$ is skewed left while $\phi_1 = 0.01, \phi_2 = 0.99$ approximates the normal limit well.

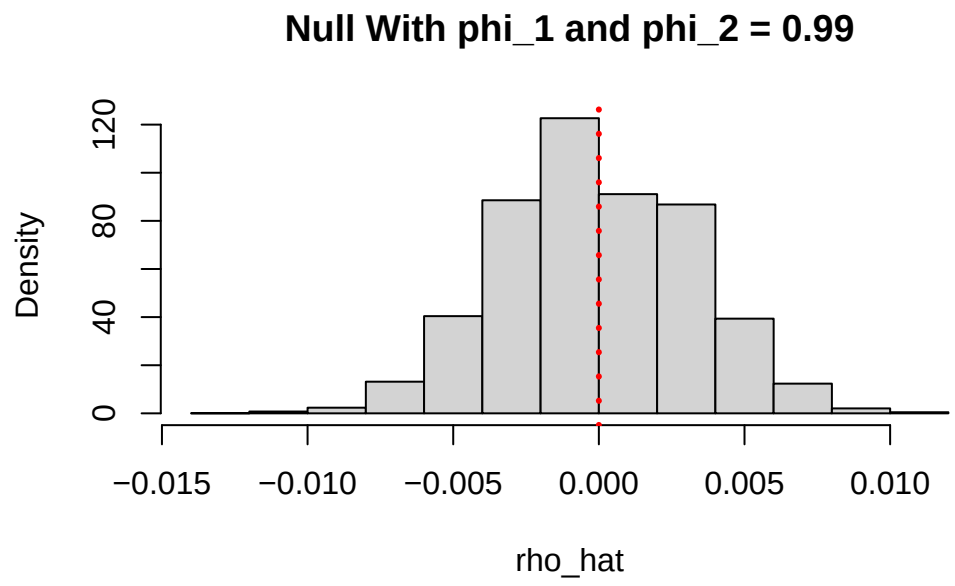
Also, since our $\hat{\rho} \sim N(\rho, \frac{\sigma^2}{m+k})$ we would expect to see the sampling distribution approximately centered at the true rho (a dotted red line in the plots) used to sample the data. In every case, it does seem like a fairly good approximation.

```
set.seed(5)


normal_compare = function(phi_1, phi_2, m, k, title, S){

  # n calculated in part d)
  n = 3858
  m = k = n/2

  # init
  rho_hat = c()

  for (i in 1:S){
    # sampling for our problem
    X_1 = rbinom(m, 1, phi_1)
```

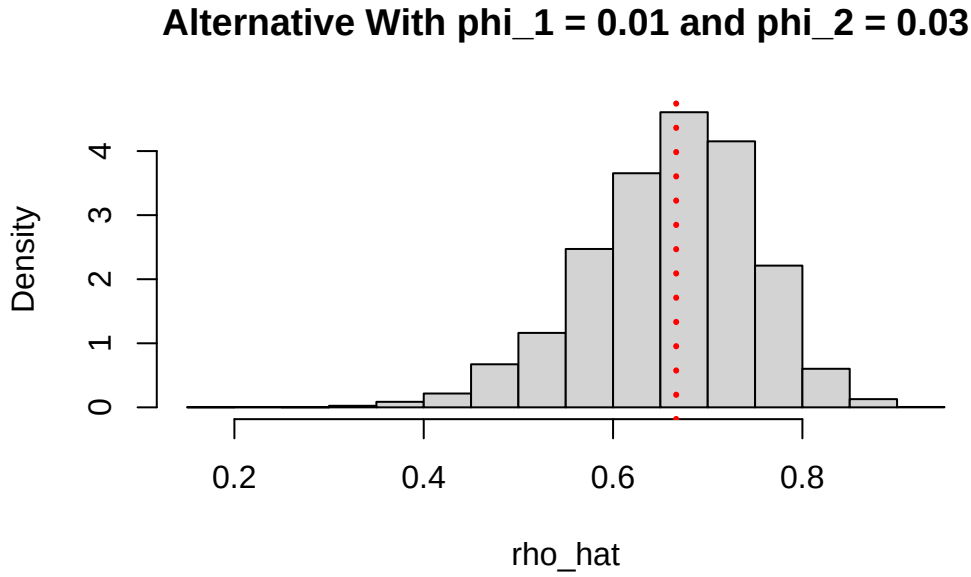
```
    X_2 = rbinom(k, 1, phi_2)

    Y_1_bar = mean(X_1)
    Y_2_bar = mean(X_2)

    rho_temp = 1 - (Y_1_bar/Y_2_bar)
    rho_hat = c(rho_hat, rho_temp)
  }

  true_rho = 1 - (phi_1/phi_2)

  hist(rho_hat, main = title, prob = TRUE)
  abline(v=true_rho, col="red", lwd = 3, lty=3)

}

normal_compare(0.01, 0.01, m, k,
               title="Null With phi_1 and phi_2 = 0.01",
               S=10000)
```



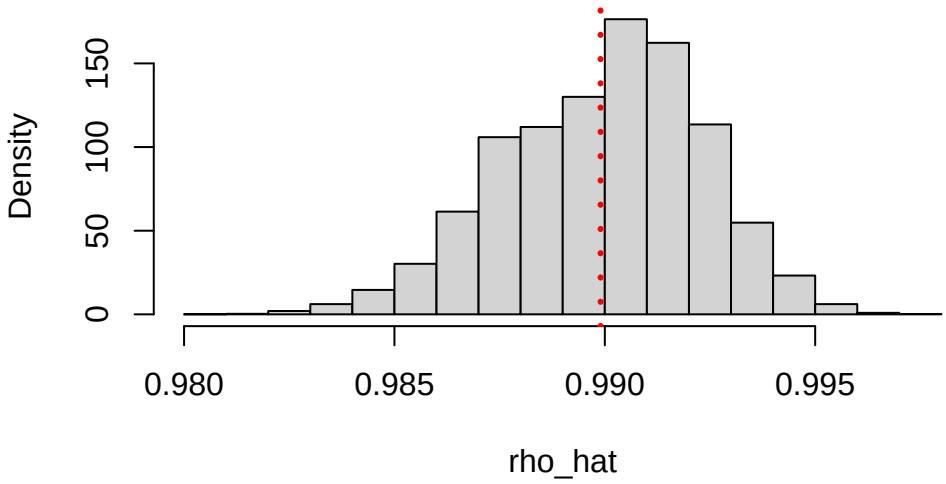

```
normal_compare(0.99, 0.99, m, k,
               title="Null With phi_1 and phi_2 = 0.99",
               S=10000)
```




```
normal_compare(0.01, 0.03, m, k,
               title="Alternative With phi_1 = 0.01 and phi_2 = 0.03",
               S=10000)
```




```
normal_compare(0.01, 0.99, m, k,
               title="Alternative With phi_1 = 0.01 and phi_2 = 0.99",
               S=10000)
```



g)

Set Up

The set up code is in the first chunk below, then the alpha and beta values are computed below that. The set up is similar to the above, only here with each draw of Y_1, Y_2 we compute the test statistic, compare to $z_{99\%}$ and average across the draws.

Results All of the below are done with 100,000 repetitions with m = k = 1929. With $\phi_1, \phi_2 = 0.01$, $\alpha \approx 0.04105$. With $\phi_1, \phi_2 = 0.99$, $\alpha \approx 0.01$ so it actually does seem to have the correct alpha with larger values of the parameters.

In the alternative hypothesis case, with $\phi_1 = 0.01, \phi_2 = 0.10$, $\beta = 0$ so the type II control doesn't seem to hold in the more reasonable case. However, with values that are closer like $\phi_1 = 0.015, \phi_2 = 0.017$, $\beta = 0.90$ so it doesn't seem to work in either case.

```
check_a_b = function(phi_1, phi_2, S=10000){

  # n calculated in part d)
  n = 3858
  m = k = n/2

  # init
  test_stat = c()

  for (i in 1:S){
```

```
    # sampling for our problem
    X_1 = rbinom(m, 1, phi_1)
    X_2 = rbinom(k, 1, phi_2)

    Y_1_bar = mean(X_1)
    Y_2_bar = mean(X_2)

    rho_temp = 1 - (Y_1_bar/Y_2_bar)
    sigma_temp = sqrt(variance(Y_1_bar, Y_2_bar))

    # store results
    test_temp = sqrt(n) * (rho_temp / sigma_temp)
    test_stat = c(test_stat, test_temp)
  }

  if (phi_1 == phi_2){
    alpha = mean(test_stat >= 2.33)
    print("alpha : ")
    return(alpha)}
  else{
    print("beta : ")
    beta = mean(test_stat < 2.33)
    return(beta)}
}
```

Null Simulations:

```
# reasonable null example
check_a_b(0.01,0.01,S=10000)
```

```
[1] "alpha : "
```

```
[1] 0.0397
```

```
# extreme null example
check_a_b(0.99,0.99,S=10000)
```

```
[1] "alpha : "
```

```
[1] 0.0092
```

Alternative Simulations:

```
# moderate alternative example
check_a_b(0.01,0.10,S=10000)
```

```
[1] "beta : "
```

```
[1] 0
```

```
# extreme alternative example
check_a_b(0.015,0.017,S=10000)
```

```
[1] "beta : "
```

```
[1] 0.9129
```

4.5

b)

Computing the C.I. for the given data, which is [0.7710499, 0.8129501]

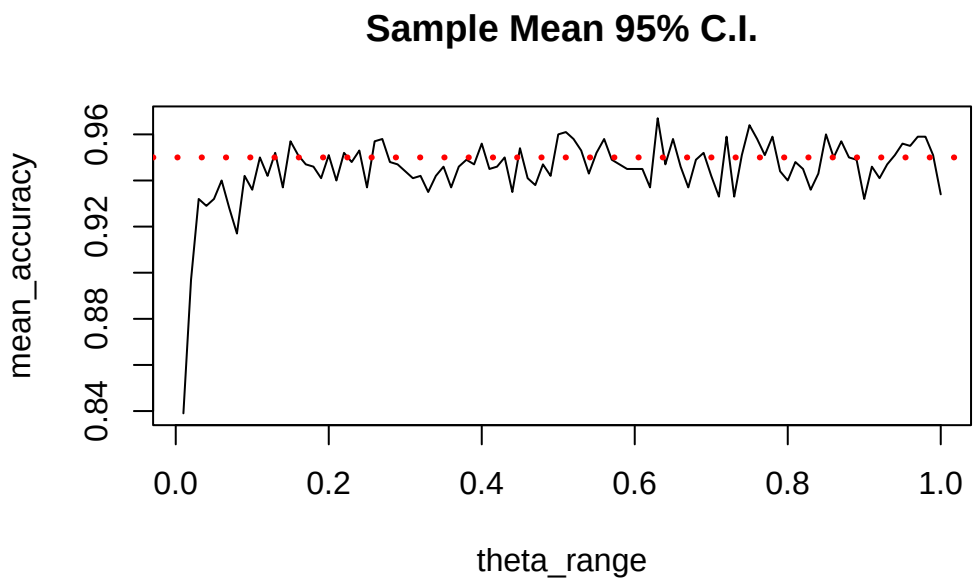
```
ratings_count_data = rep(c(5,4,3,2,1), c(119, 59, 17, 5, 0))
n = length(ratings_count_data)
X = (ratings_count_data - 0.5) / 5
X_bar = mean(X)
S = sqrt(var(X))

CI = c(X_bar -1.96 * S/sqrt(n), X_bar + 1.96 * S/sqrt(n))
cat("95% CI for 4.5 b) : ", "[", CI, "]")
```

```
95% CI for 4.5 b) :  [ 0.7710499 0.8129501 ]
```

c)

The distribution we have is $Beta(\theta, 1)$, comparing draws of this to our confidence interval below, we can see the 95% is a fairly good approximation if $\theta > 0.1$ or so. The horizontal red line is again at 0.95.

```r
theta_range = seq(0.01, 1, .01)
mean_accuracy = c()

for (theta in theta_range){

  contained = c()

  for (i in 1:1000){

    # draw data
    data = rbeta(200, theta, 1)

    # compute our statistics
    x_bar = mean(data)
    s_err = sqrt(var(data)/200)
    draw_ci = c(x_bar - 1.96*s_err, x_bar + 1.96*s_err)
    stat = theta / (1+theta)

    # check if contained in interval
    test = between(stat, draw_ci[1], draw_ci[2])

    # update test value
    contained = c(contained, test)
  }
  mean_accuracy = c(mean_accuracy, mean(contained))
}

plot(theta_range, mean_accuracy, type='l', main="Sample Mean 95% C.I.")
abline(h=0.95, col='red', lwd=3, lty=3)
```

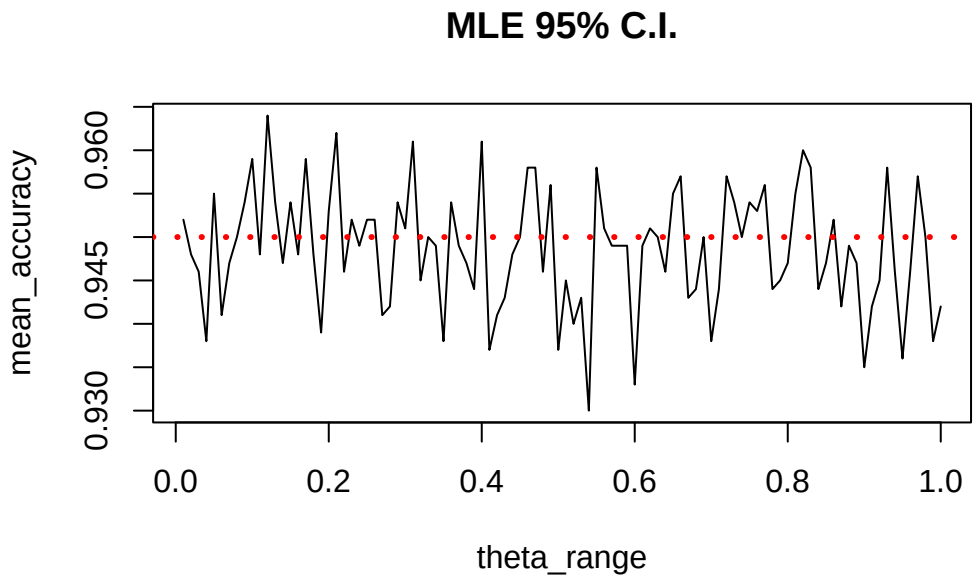


e)

Repeating for the M.L. rule here, with supporting calculations done in quesiton d). The rule appears to give a good C.I. since it is around 95% at all true θ values.

```
theta_range = seq(0.01, 1, .01)
mean_accuracy = c()

for (theta in theta_range){

  contained = c()

  for (i in 1:1000){

    # draw data
    data = rbeta(200, theta, 1)
    # (1 /(sqrt(mle) * (mle +1)^4))
    # compute our statistics, now using MLE
    mle = -200/sum(log(data))
    s_err = (mle / (mle+1)^2) / sqrt(200)
    mu_mle = mle / (1 + mle)
    draw_ci = c(mu_mle - 1.96*s_err, mu_mle + 1.96*s_err)

    # our stat
```

```
    stat = theta / (1+theta)

    # check if contained in interval
    test = between(stat, draw_ci[1], draw_ci[2])

    # update test value
    contained = c(contained, test)
  }
  mean_accuracy = c(mean_accuracy, mean(contained))
}

plot(theta_range, mean_accuracy, type='l', main='MLE 95% C.I.')
abline(h=0.95, col='red', lwd=3, lty=3)
```



f)

It seems the MLE approach is better since we move around 0.95 even at low values of θ, while the sample average approach really only approaches 0.95 when $\theta > 2$. Since we don't know the true population θ, it's important that we have good coverage at all possible values $\theta \in \Theta$.