304proje

Betül Kunt, Mehmet Enes Özkalay

Weibull(β) Distribution Testing

Let $X \sim$ weibull(β), then the PDF is given by:

$$f(x;\beta) = \frac{2x}{\beta^2} \exp\left(-\frac{x^2}{\beta^2}\right), \quad x > 0$$

```
# Set seed for reproducibility
set.seed(42)

raw_text = "   [1]  2.0506664  2.4418716  1.6895062  3.2531637  2.3198719  4.4698084  1.6939512  2.9590
   [9]  4.1847542  3.0365707  5.2666450  4.8711291  1.2496101  4.5915575  6.1815881  0.6402515
  [17]  6.5999418  4.1337756  6.0460886  4.4169131  1.7573283  5.2744698  3.4384707  6.6046135
  [25]  1.0200291  2.7822116  2.1845569  2.8399562  2.6125709  1.9518557  2.0144829  2.9347502
  [33]  0.7522713  2.9936573  5.2407103  2.7968781  3.9407371  3.0729327  2.5688839  0.5485185
  [41]  4.9877395  3.1705540  0.6892765  4.3144134  2.5180422  3.6442377  1.4999070  2.6711828
  [49]  6.5096089  1.1930837  1.6635489  4.6602042  3.4306783  4.8520067  0.6635464  4.2903040
  [57]  0.7286915  1.5343534  2.9396468  3.0054957  5.9087623  3.9549735  2.3105794  3.3735016
  [65]  2.2482955  0.9612390  3.7206955  3.4346305  1.7584594  0.7559151  3.9894130  2.2357807
  [73]  1.1067701  3.7216152  3.3706761  4.9916106  2.6153868  4.6168001  3.4826640  2.3974199
  [81]  7.0535915  3.4015467  3.3630194  3.7613399  4.0508445  7.3164292  3.5321653  2.8821792
  [89]  2.2141557  6.3317380  3.4316548  4.3148912  2.2828320  3.5378508  4.2680322  7.5214712
  [97]  4.9330723  3.7909685  6.1748287  2.2082391  2.4222002  1.4085497  6.5569456  1.9533218
 [105]  2.6436217  7.5334010  2.4295289  1.2657540  5.4999001  1.2311320  1.8337442  2.5441401
 [113]  2.8883549  3.1401741  4.0661296  2.8949424  5.8459620  4.7543676  1.5153851  3.7582446
 [121]  3.5383752  7.9141586  1.7641834  3.4226869  3.0876708  3.9692299  5.8924399  4.8892895
 [129]  2.9826197  1.1047927  3.2545530  1.6363549  4.6614501  2.1347237  5.4244561  3.4809713
 [137]  6.0557953  3.2897267  4.2981167  5.1434550  2.7548443  2.8150308  4.0942738  3.3729453
 [145]  4.9667916  5.7897846  1.7460319  1.0424966  3.1640389  6.6612354  2.7813486  0.5538432
 [153]  3.1207528  5.1043627  1.9965464  2.5104729  1.1470333  4.4941244  0.3643007  0.3044525
 [161]  1.3150085  1.7507155  3.6578673  1.9509434  4.4099279  1.8556666  5.9069437  2.4861963
 [169]  1.2214174  3.2733330  3.0087287  4.0480077  2.4972981  5.6221212  3.4596555  3.6994610
 [177]  7.2441079  1.9664996  5.7597147  3.8502399  9.5988264  1.8990409  5.4796719  4.0443343
 [185]  0.6987340  0.9349948  3.6486741  5.0887490  3.1433357  2.1886642  3.2960816  2.3798974
 [193]  4.5015225  1.8955754  5.4702578  2.5785113  3.3493818  1.0789948  2.0271379  3.0847858
 [201]  4.5019106  7.8068128  5.2692844  5.3913853  4.9030785  4.4972613  3.4798208  1.7625333
 [209]  0.8551645  6.4102243  3.6570018  6.2297534  1.8115336  1.5573531  4.0658851  4.7947977
 [217]  1.3167055  0.1305030  2.7944602  3.6430575  4.0436531  2.4430799  3.5672396  0.3394521
 [225]  2.7592635  2.7371623  4.6215585  4.7788468  1.7861501  6.1279479  3.5580238  4.4007080
 [233]  1.6951473  4.3699007  2.5851573  1.5788337  2.5286502  4.3940064  2.1445380  5.2792980
 [241]  1.3435920  4.5774440  3.3670315  3.2641971  4.7984448  1.9565109  5.7282059  3.2688441
 [249]  3.0311520  2.5882021  3.1732974  2.4965396  4.3314098  0.6763416  6.1071139  2.0673398
 [257]  2.5904683  2.4352736  3.9745104  4.9739965  4.7110228  1.9478545  2.6799694  3.3795754
 [265]  3.1866221  2.9548010  4.3096416  1.5464455  6.4280159  1.8618356  2.0353826  3.1787684
```

```
[273]  4.6171618  6.5067495  3.7071672  4.3358220  3.5914565  2.8100161  2.0209257  4.3522619
[281]  3.9897620  3.5774958  3.7305544  7.6416840  4.0503547  3.1113562  3.1141325  7.5514233
[289]  2.1458099  3.9880528  4.7442718  3.8211612  2.3442062  1.7101955  4.1169053  4.2526023
[297]  1.5685620  6.1744305  4.5052088  2.8472863  0.9616665  2.5982744  3.1396214  3.6900280
[305]  2.5179789  4.5730405  3.0881868  1.7467456  0.3423111  4.3187264  2.2172924  4.8894326
[313]  8.3637880  8.8074337  5.5551256  3.0759307  3.0220718  4.7420257  2.1903653  4.6054261
[321]  4.7204405  3.1457792  3.9349894  1.9223097  2.6410297  3.9681654  1.6588176  3.3327050
[329]  2.1296606  4.0258973  5.6985839  0.7568462  1.9360307  4.9985521  3.9516906  4.0610732
[337]  4.9176194  1.5355183  4.1436223  2.7297571  6.8837126  4.0563566  5.5768881  2.3036277
[345]  2.5178486  5.5414722  3.9046538  6.4149414  5.7825667  2.4497300  2.2392169  2.7368444
[353]  3.0992710  2.0793222  3.5058416  8.4754773  2.5047424  2.4045197  4.5010058  2.3935073
[361]  2.5764887  4.4743461  2.0922545  2.3020900  0.7575296  3.0719927  5.1148367  3.2544833
[369]  1.3249625  4.2891199  1.0363689  2.0950210  7.8921042  6.4120690  1.4076838  2.2898199
[377]  3.7574915  3.1928181  4.4936258  2.3434882  3.6313903  0.5473187  2.6018611  1.5924269
[385]  4.0583898  3.3239419  2.4034524  4.1402591  2.1550890  4.7594494  2.4649399  4.2907135
[393]  8.5367128  0.7973975  6.4787676  2.8376099  6.0783795  2.5012035  1.5892192  2.6639574
[401]  5.4135952  2.9698908  3.7533286  1.6435129  3.4962421  1.3259828  6.2722077  3.2038965
[409]  7.5634099  2.0005809  3.2922596  5.6903907  6.4746376  3.4089575  2.0903372  0.7093902
[417]  4.2503609  0.7983674  1.7645178  6.9555591  2.6452440  2.2958287  3.3447753  4.6856743
[425]  2.8302957  3.7224019  1.2776799  3.1486670  2.3100913  5.0818000  5.5488421  1.2612701
[433]  3.7210927  4.6580360  4.5508534  3.8652712  5.7992663  3.5970021  3.4892709  5.2595597
[441]  2.0736457  2.9198602  4.8686923  5.5382483  0.9577944  6.4765205  4.9546257  4.7301233
[449]  2.4749543  3.1091547  2.9338341  3.8301621  3.9789726  2.1216449  4.7679651  7.0799891
[457]  6.1542079  6.5268430  2.5716905  5.6551864  7.1146590  2.4679992  0.8816841  5.9189065
[465]  2.3760830  2.0231070  4.7963148  3.4953919  2.8784855  4.1769024  3.9377325  5.2089179
[473]  2.7109792  7.3789936  1.9777401  5.9803377  3.1870309  3.7992559  7.8755983  4.6458777
[481]  3.6626371  2.2886837  2.1633648  2.4241454  0.4918397  3.8630844  5.5003513  1.4910064
[489]  2.2923310  2.5258436  2.5674267  0.8757393  1.1259920  1.1829250  1.5641021  4.3184489
[497]  5.8207313  2.6945347  0.9708454  3.6555201  3.0912475  4.5404230  4.9326359  1.3203037
[505]  4.7408518  6.4013558  2.0806967  3.4203171  2.1191551  7.0067131  2.6676660  7.1620011
[513]  1.6790377  6.2089058  7.8821008  1.5027343  3.4600416  5.0204697  4.6217405  0.6282097
[521]  2.6269397  3.4646644  6.1078531  4.2872362  5.1166053  4.1502646  7.0838676 10.4851084
[529]  2.4830489  0.2077951  1.3173227  8.6176119  6.3955488  4.4369480  4.2436030  7.8160338
[537]  2.7850758  1.0011360  1.3457331  2.2221050  1.7220446  0.5528298  1.5662488  2.1874568
[545]  5.1364971  3.4681563  0.8029627  2.0958823  4.7080385  3.2827676  1.5849033  6.2619477
[553]  2.9748057  2.0536679  4.7281053  1.2929622  2.8924341  2.5512976  8.3993824  4.4299111
[561]  5.9623302  2.6703672  2.3899026  4.7623385  2.9948673  4.2359837  3.5183866  3.7207335
[569]  1.3209157  5.2333642  4.8560410  3.5233520  1.3707581  1.0166854  4.3393122  5.4655960
[577]  4.9131355  2.3470183  3.0844307  5.1293369  3.6680410  4.8188165  1.9056408  4.0022635
[585]  2.7398859  0.2193189  2.5664623  2.9618702  3.8418298  1.9540082  0.4740373  3.4883282
[593]  2.1269835  3.5253327  2.7275142  4.5078800  5.8801538  0.9593919  3.8297748  4.8041766
[601]  4.3116488  4.4185698  4.5445923  1.6383960  1.5022822  1.9679889  2.2520322  5.8472877
[609]  3.8248177  2.4590160  4.4880140  1.7935828  1.7877363  4.9944582  2.8385706  1.4215819
[617]  5.3696567  0.5382885  4.8596271  1.5159105  3.6256811  3.2430578  4.8946853  1.7192712
[625]  3.6187586  4.5406901  4.6132269  2.4257080  7.1813352  3.8292511  4.3166774  3.3048393
[633]  1.5228996  2.0990592  5.0352190  3.7723981  5.2729124  7.3709339  2.9381453  3.3929629
[641]  3.5364575  5.4096797  7.1857286  4.4407300  2.7670648  2.6894227  5.2528573  2.5526076
[649]  1.9327529  2.4384723  1.3844202  2.2705960  5.9731028  3.5330166  2.6752259  0.8067565
[657]  2.5276448  3.2435167  4.1871882  5.0830312  1.4017608  1.2112948  4.6364539  2.3089891
[665]  2.2274823  4.0916860  2.0041599  3.3119324  4.5664418  6.7094803  6.5454909  3.4398891
[673]  5.1188830  7.5895389  3.9791749  2.5820281  0.7153930  4.7219663  3.5979493  2.7683105
[681]  4.5591574  3.0385901  4.4446386  2.7091429  4.3920211  2.3292674  3.8656860  3.1720202
[689]  0.9411936  5.0175382  0.9067392  1.0947573  4.2236783  0.6559885  1.8636964  1.2448879
```

```
[697]  4.8477652  7.2313855  4.3433624  4.7599597  3.5584474  0.8603289  2.4569158  4.8758830
[705]  5.5421624  4.2716677  6.2799085  4.1685192  1.0185803  2.7382099  7.2568300  3.1986770
[713]  3.5480360  1.8076235  6.1373674  5.6617405  2.6412393  3.4236088  2.2040369  4.8684052
[721]  4.2294357  6.0356192  1.5862979  2.3295246  2.8069953  1.8528469  4.7891200  1.4311821
[729]  3.6206259  0.8261398  4.5661962  4.9132393  2.2869391  3.6913397  1.7329576  3.2811738
[737]  5.3215525  4.4312731  3.7142419  8.7462864  4.7072007  1.7993270  5.1512550  2.1161692
[745]  4.6853940  6.0319215  3.4461881  5.0005533  3.5568807  4.9811059  1.6413142  7.1353413
[753]  4.0769247  6.0154994  1.3343166  4.7507079  2.7470104  4.1915949  4.6251967  3.9271338
[761]  3.1278266  2.5442680  3.7798630  3.9945742  2.1606314  7.2155985  0.9235752  7.0649690
[769]  2.7037322  2.8209833  4.1560949  4.1365990  1.5351820  4.3975506  0.9928557  2.9118170
[777]  1.7256376  5.2122060  0.1238319  3.2974414  0.8621476  2.2607957  4.0757823  3.2557954
[785]  0.9515594  7.6875563  1.3538149  4.4906836  1.5916401  9.7211758 11.8414236  5.4669828
[793]  2.0039758  4.3797840  1.5569185  2.9522951  3.5021104  2.2529207  4.6957692  1.8952300
[801]  4.6193424  4.0416214  2.6601585  1.1704312  3.4449962  4.0229543  3.4363558  2.2715557
[809]  2.7453644  1.4750939  6.4929463  3.7170306  1.6238059  3.9781926  6.4322074  3.1203226
[817]  6.9930050  0.5842458 10.6851217  5.7867619  5.6340370  3.9619526  4.8246565  4.7899855
[825]  5.9000865  3.2148130  2.7672348  4.8392565  4.6214627  1.2651306  5.2375521  5.7834457
[833]  2.6903270  4.7095962  2.7839082  6.3496132  2.2866560  5.1386282  2.6225129  7.1481037
[841]  6.6865559  1.5596131  3.5601957  2.7867948  2.2784583  7.9121596  0.9411812  4.0511037
[849]  4.1482359  1.8059364  4.2852500  1.6540567  8.0515424  2.2682948  3.0276115  2.1317746
[857]  2.0918810  6.2305184  2.7403101  2.0921757  4.4100819  7.7211325  2.2438178  3.6874239
[865]  5.0685055  0.9510255  3.2648117  2.5495264  4.9127951  5.2767908  0.8506560  2.0646643
[873]  1.6174219  5.3266080  1.5777008  3.9844792  4.1553609  5.5468634  6.5215614  4.2565638
[881]  4.5660073  7.2867707  2.3955118  3.5694799  2.0422796  2.7209772  1.2014345  3.0635127
[889]  3.6768089  2.0929249  1.4315021  3.7209748  4.7053363  3.9431261  1.3591700  1.2518430
[897]  2.5046478  2.3057984  2.5692457  6.8641658  0.7260491  6.2057580  0.4798124  2.0958260
[905]  0.6619590  4.6027529  4.6981839  2.1668426  2.1081705  1.0481621  4.0902223  3.3213102
[913]  6.8545131  2.8223245  3.1658535  2.8573206  5.0014956  1.4813586  1.6098122  0.8769456
[921]  2.7203861  1.8838778  1.1633516  3.4454966  4.8347177  3.1844062  5.1101475  4.0068103
[929]  2.2482134  2.8844533  0.6846374  7.1049244  4.9392287  5.9053249  3.3802559  3.7333221
[937]  1.2872544  1.4102282  1.2013884  4.5646286  3.1558429  2.4936299  3.8738773  4.6368586
[945]  3.6806165  4.4288049  5.3382512  2.0373889  2.5356913  2.4485916  2.6681089  1.7457033
[953]  1.7416125  1.7606352  1.1036271  2.9023427  2.7642357  2.8154811  2.0694465  3.4111210
[961]  3.4099692  1.0227649  2.5644241  6.5293146  2.4368953  4.0779800  1.6713951  4.2797905
[969]  3.6461612  1.5357207  0.9644677  4.4819388  2.2562980  9.2186493  1.0481789  7.8583568
[977]  3.6315160  4.7986213  6.0524314  1.3272624  2.3443622  5.2265570  6.1843015  2.5464572
[985]  3.8249664  5.2177893  3.4499587  1.5294679  4.9628907  5.9890413  1.7007087  1.3118405
[993]  7.7916906  3.0807285  0.9898630  0.7950674  5.5807648  5.8393383  4.2471903  3.3636951
"

clean_text <- gsub("\\[.*?\\]", "", raw_text)

# Step 3: Convert to numeric vector
data <- as.numeric(unlist(strsplit(clean_text, "\\s+")))
data <- data[!is.na(data)]

# Parameters
alpha <- 0.03         # Significance level
n <- 100              # Sample size

# Step 2: Take a random sample
sample <- sample(data, size = n, replace = FALSE)
```

```
# Step 3: Calculate test statistic T(x)
Tx <- sum(sample^2)
```

Confidence Interval for β

Let $X_i \sim \text{Weibull}(2, \beta)$. Then define:

$$Y_i = \left(\frac{X_i}{\beta}\right)^2 \sim \text{Exp}(1)$$

So the sum:

$$\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} \left(\frac{X_i}{\beta}\right)^2 = \frac{1}{\beta^2} \sum_{i=1}^{n} X_i^2 \sim \text{Gamma}(n, 1) = \frac{1}{2}\chi^2_{2n}$$

Multiplying both sides by β^2:

$$CriticalValue \sum_{i=1}^{n} X_i^2 \sim \text{Gamma}(n, \theta = \beta^2)$$

Or, in terms of a chi-square distribution:

$$\sum_{i=1}^{n} X_i^2 \sim \frac{\beta^2}{2} \cdot \chi^2_{2n}$$

```
# Confidence level
alpha <- 0.03  # %97

# Sample
n <- 100
sample <- sample(data, size = n, replace = FALSE)

# Test Stat
Tx <- sum(sample^2)

# Chi_square critical value (df = 2n)
chi2_upper <- qchisq(1 - alpha/2, df = 2 * n)/2
chi2_lower <- qchisq(alpha/2, df = 2 * n)/2

# Confidence Interval for beta^2
lower_var <- Tx / chi2_upper
upper_var <- Tx / chi2_lower

# Confidence Interval for beta (sqrt)
lower_beta <- sqrt(lower_var)
upper_beta <- sqrt(upper_var)

cat(sprintf("%d%% Confidence Interval for  : [%.4f, %.4f]\n", (1 - alpha)*100, lower_beta, upper_beta))
```

```
## 97% Confidence Interval for  : [3.7699, 4.6863]
```

MP Test

$$H_0 : \beta = \beta_0 H_1 : \beta = \beta_1 \quad (\beta_0 \neq \beta_1) \quad \text{(simple vs simple)}$$

Likelihood ratio:

$$\lambda(x) = \frac{L(\beta_1)}{L(\beta_0)} = \left(\frac{\beta_0^2}{\beta_1^2}\right)^n \exp\left[\left(\frac{1}{2\beta_1^2} - \frac{1}{2\beta_0^2}\right)\sum x_i^2\right]$$

Let $T(x) = \sum x_i^2$:

- When $\beta_0 > \beta_1$, reject H_0 if $T(x) < c$
- When $\beta_0 < \beta_1$, reject H_0 if $T(x) > c$

Key Transformation: If $X \sim \text{Weibull}(alpha = 2,\ beta)$, then $Y = \left(\frac{X}{\beta}\right)^2 \sim \chi_2^2$

$$\Rightarrow \frac{X_i^2}{\beta^2/2} \sim \chi_2^2 \Rightarrow \sum \frac{X_i^2}{\beta^2/2} \sim \chi_{2n}^2 \quad (\text{under } H_0) \Rightarrow T(x) = \sum X_i^2 \sim \frac{\beta_0^2}{2}\chi_{2n}^2 \quad (\text{under } H_0)$$

- Reject H_0 if $\sum X_i^2 < \frac{\beta_0^2}{2}\chi_{2n,1-\alpha}^2$, when $\beta_0 > \beta_1$
- Reject H_0 if $\sum X_i^2 > \frac{\beta_0^2}{2}\chi_{2n,\alpha}^2$, when $\beta_0 < \beta_1$

```
# --- MP Test (Two-sided) ---
beta0 <- sqrt(mean(sample^2)) # MLE for beta
beta0
```

```
## [1] 4.179678
```

```
statistic_mp <- Tx / beta0^2
chi2_crit_low <- qchisq(alpha, df = 2 * n)/2
chi2_crit_high <- qchisq(1 - alpha , df = 2 * n)/2
reject_mp <- (statistic_mp < chi2_crit_low) || (statistic_mp > chi2_crit_high)

cat("MP Test (Two-sided)\n")
```

```
## MP Test (Two-sided)
```

```
cat(sprintf("T(x)/ ² = %.2f\n", statistic_mp))
```

```
## T(x)/ ² = 100.00
```

```
cat(sprintf("Critical region: < %.2f or > %.2f\n", chi2_crit_low, chi2_crit_high))
```

```
## Critical region: < 82.06 or > 119.64
```

```
cat("Result:", ifelse(reject_mp, "Reject H0", "Do not reject H0"), "\n\n")
```

```
## Result: Do not reject H0
```

```r
cat("Bounds of theta:")
```

```
## Bounds of theta:
```

```r
sqrt((Tx)/chi2_crit_low)
```

```
## [1] 4.614126
```

```r
sqrt((Tx)/chi2_crit_high)
```

```
## [1] 3.821321
```

UMP Test

$$H_0 : \beta = \beta_0 H_1 : \beta > \beta_0 \text{ or } \beta < \beta_0 \quad \text{(Let it be } \beta_1\text{)}$$

- Under $\beta_0 > \beta_1$:

$$\frac{L(\beta_1)}{L(\beta_0)} = \left(\frac{\beta_0^2}{\beta_1^2}\right)^n \exp\left[\left(\frac{1}{\beta_1^2} - \frac{1}{\beta_0^2}\right)\sum x_i^2\right]$$

- Under $\beta_1 > \beta_0$:

$$\frac{L(\beta_1)}{L(\beta_0)} = \left(\frac{\beta_1^2}{\beta_0^2}\right)^n \exp\left[\left(\frac{1}{\beta_0^2} - \frac{1}{\beta_1^2}\right)\sum x_i^2\right]$$

$T(x) = \sum x_i^2$, and L is a non-decreasing function of $T(x)$, so the MLRS is $T(x)$.

- If $H_1 : \beta < \beta_0$, reject H_0 if MLRS < c
- If $H_1 : \beta > \beta_0$, reject H_0 if MLRS > c

```r
# --- UMP Test (One-sided: H1:   >  ) ---
chi2_crit_ump <- qchisq(1 - alpha, df = 2 * n)
reject_ump <- statistic_mp > chi2_crit_ump

cat("UMP Test (H1:  >  )\n")
```

```
## UMP Test (H1:  >  )
```

```r
cat(sprintf("T(x)/ ² = %.2f, Critical value = %.2f\n", statistic_mp, chi2_crit_ump))
```

```
## T(x)/ ² = 100.00, Critical value = 239.27
```

```r
cat("Result:", ifelse(reject_ump, "Reject H0", "Do not reject H0"), "\n\n")
```

```
## Result: Do not reject H0
```

GLR Test

$$H_0 : \beta = \beta_0 \quad \text{vs.} \quad H_1 : \beta \neq \beta_0$$

Parameter spaces:

$$\Omega_0 = \{\beta_0\} \quad \Omega_1 = (0, \beta_0) \cup (\beta_0, \infty) \quad \Omega = \Omega_0 \cup \Omega_1 = (0, \infty)$$

Likelihood:

$$L(\beta) = \prod \frac{2}{\beta^2} x \exp\left(-\frac{x_i^2}{\beta^2}\right) \Rightarrow \ell_n L(\beta) = \sum \ln x_i - 2n \ln \beta + \ln\ 2 - \frac{\sum x_i^2}{\beta^2}$$

Derivative:

$$\frac{\partial \ell_n L(\beta)}{\partial \beta} = -\frac{n}{\beta} + \frac{1}{\beta^3} \sum x_i^2 = 0 \Rightarrow \hat{\beta}_{MLE} = \sqrt{\frac{1}{n} \sum x_i^2}$$

GLR statistic:

$$\lambda = \frac{L(\beta_0)}{L(\hat{\beta})} = \left(\frac{\hat{\beta}^2}{\beta_0^2}\right)^n \exp\left[\sum x_i^2 \left(\frac{1}{\hat{\beta}^2} - \frac{1}{\beta_0^2}\right)\right]$$

This simplifies to:

$$\left(\frac{\sum x_i^2}{n\beta_0^2}\right)^n \exp\left[n - \frac{\sum x_i^2}{\beta_0^2}\right] < \lambda_0$$

Taking log:

$$\ln \lambda = n \ln \left(\frac{\sum x_i^2}{n\beta_0^2}\right) + n - \frac{\sum x_i^2}{\beta_0^2} < \ln \lambda_0$$

Define:

$$\ln \left(\frac{\sum x_i^2}{n\beta_0^2}\right) - \frac{\sum x_i^2}{n\beta_0^2} < \frac{\ln \lambda_0}{n} - 1$$

Let $Y = \frac{\sum x_i^2}{n\beta_0^2}$, $Z = \ln Y - Y$, then:

$$P(\text{Reject } H_0 | H_0 \text{ true}) = P(Y < k_1) + P(Y > k_2) = \alpha$$

Since $Y \cdot n \sim \chi^2_{2n}$,

$$\text{Reject } H_0 : \frac{\sum x_i^2}{\beta_0^2} < \chi^2_{2n,1-\alpha/2} \text{ or } > \chi^2_{2n,\alpha/2}$$

Final form:

$$-2 \ln \lambda = -2n - \ln \sum x_i^2 + 2 \ln n\beta_0^2 + \frac{2\sum x_i^2}{\beta_0^2} \sim \chi^2_1$$

Then:

$$\alpha = P(\text{Reject } H_0 | H_0 \text{ TRUE}) = P(\lambda < \lambda_0 | \beta = \beta_0) = P(-2 \ln \lambda > -2 \ln \lambda_0 | \beta_0)$$

Decision rule:

$$\text{Reject } H_0 : -2 \ln \lambda > \chi^2_{1,\alpha}$$

```r
# --- GLR Test ---
beta_hat <- sqrt(Tx / n)  # MLE for beta
beta_hat
```

```
## [1] 4.179678
```

```r
lambda_glr <- (beta0 / beta_hat)^(n) * exp(n- Tx/(beta0^2))
test_stat_glr <- -2 * log(lambda_glr)
crit_val_glr <- qchisq(1 - alpha, df = 1)
reject_glr <- test_stat_glr > crit_val_glr

cat("GLR Test\n")
```

```
## GLR Test
```

```r
cat(sprintf("-2 * ln( ) = %.2f, Critical value = %.2f\n", test_stat_glr, crit_val_glr))
```

```
## -2 * ln( ) = -0.00, Critical value = 4.71
```

```r
cat("Result:", ifelse(reject_glr, "Reject H0", "Do not reject H0"), "\n")
```

```
## Result: Do not reject H0
```

Rayleigh(σ) Distribution Testing

Let $X \sim \text{Rayleigh}(\sigma)$, then the PDF is given by:

$$f(x;\sigma) = \frac{x}{\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right), \quad x > 0$$

```r
# Set seed for reproducibility
set.seed(42)

# Step 1: Read the data from the .txt file
# Set seed for reproducibility
set.seed(42)

# Step 1: Read the data from the .txt file
raw_data = "[1]  3.9838900  3.1404940  8.3858660  2.4450140 16.3581700  4.5726500 11.4172600  5.7863790
  [17]  6.3507690  2.8269510  7.0376860  6.1845420 10.9223500  8.1495820  5.2922420  3.2583290  4.00024
  [33]  3.0797980  3.0531820  7.8485750  3.9460790  6.6153560  6.9011300  4.6500100  7.4195490 16.24042
  [49]  3.2206580  8.4931370  6.4404790  4.6833290  4.1367620  3.6528120  6.2377320  6.6108390  3.68361
  [65]  0.6818655  6.4133990  2.7039290  4.7547270  4.9279070  1.8828730  5.2236440  7.7868520  7.97342
  [81]  0.5807991  0.6204383 10.6870200  3.3198640 10.0078400  6.2659670 10.4879000  6.6317080 10.54423
  [97]  5.2786240  6.0293860  8.8291700  0.7051838  4.4633710  3.9695270  3.2523630  7.8681480  4.55202
 [113]  8.2061820  2.4040490  7.0610320  5.7856600  3.4936740  5.3839060  3.6820850  7.6601590  3.38186
 [129]  2.3855760  2.4506050  2.9259330  9.5387880  3.3920040  8.8795570  6.1851380  1.7651920  5.14707
 [145]  7.3534710  9.0676660  3.8657690  2.9331710  9.6418840  9.3680600  6.4217720 14.8365000  2.91971
 [161]  5.7329430  2.0257250  4.0992530  3.8857330  7.5103450  5.7637170  5.8197160 13.1333400  4.12911
 [177]  2.8300500  7.2459060 13.1639700  9.2052400  5.1496650  2.1075970  7.9669070  4.9047050  8.63752
 [193] 15.6062900  3.1501280  5.6347090  4.7673180 16.2381200  9.8650720  2.9560610 11.9607300  6.98302
 [209]  6.7608340 13.9178100 10.1362200  9.6294820  4.0794230  3.7198240  2.1588070  8.5780670  1.92092
 [225] 12.8214400  5.7767850  6.2317370  1.2988770  7.1013670 11.3476600  6.5930870  4.1032830  3.32701
 [241]  5.5249090  5.0297060  6.7942730  8.6721730  4.3388160  2.5118180  5.3362500  6.4785420  2.36751
 [257]  4.8485380  4.5282650  3.2180300  2.7563940  4.4104340  8.4459340  6.2683180  5.0950920 10.47969
 [273] 12.4085300  2.5943430  8.4888630  8.8930230  2.5018990  7.6740380  4.0688100  4.9325660  6.43826
 [289]  3.0301000  5.7340800  4.6443860  3.8799240  5.5591090  4.5230770  7.8546510  6.9069230  7.92394
 [305]  0.9713148  2.9407450  5.3011910  4.8476820  6.6987490  5.3669250  9.0633590  3.2033980  5.396568
 [321]  0.7081749  8.1550920  5.2508730  2.2046240 10.0687300  4.2841340  9.4785310  1.7635080  6.187452
 [337]  2.3274640  8.0675140  4.3895010  7.9644730  7.9695810  5.1003030  5.9628290  1.9647770  8.833130
 [353] 10.5696200  6.7030580  4.6130670  8.5327250  8.2368350 12.1877200  5.4273900  4.4215090  1.027020
 [369]  4.2606920  2.2564310  3.3054160  1.2003270  5.0039800  2.8852510  7.9421540  1.4429100  5.630375
 [385]  9.8327100  4.7973920  3.2663500  5.8165870  8.3082650  1.1218610  8.6503930  1.1999300  2.081987
 [401]  3.1219350  7.1884710  7.1713720  3.4152180 11.4110300 12.5543000  4.8496610  6.9726030  7.426312
 [417] 11.6057300  6.0472180  7.7623700  6.0444000  3.2140510  2.8162000  3.6791370  2.6250130  7.960260
 [433]  3.0369820  2.9299190  5.4387770  4.8192580  7.8179530 19.1919800  4.8850110  7.1767720  5.976648
 [449]  2.0260950  8.3268350  1.6619170  3.6253140 11.2295300  5.7285140  3.6330740  5.6252450  4.731561
 [465]  6.2281500  2.5408820 10.3117000 12.2002900  7.8582290  2.4574290  2.7266070  2.1117050  1.595450
 [481]  2.9949380  2.1924070  5.6830960  4.9269460  2.3895540  9.1129830  4.2448170  5.7651690  4.958907
 [497]  7.7114230  5.7243430 10.6804400  4.8687990  6.8599620  5.6379220  9.1748270  7.8755680  7.866867
 [513]  5.2997480  5.4024960 10.9953500  5.3271070  0.6354547  3.2230590 15.1330700  7.5575630  4.549397
 [529]  8.3451320  6.7625450  2.2492910  5.2748160  8.6380020 10.1220600  5.4623060 12.8996400  3.612817
 [545]  8.9648620  7.3310480  2.8729200  5.8590210 14.4235800 10.5755000  1.8026360  6.3093270  0.822998
 [561]  5.9339020  1.9074400  6.3262120  4.1776490  2.4134310  2.9423390  2.0548470  1.3999330  1.182530
 [577]  7.8248590  7.5811710  6.1865320  8.4177610  0.9024701  6.3392500  3.1846940  7.6687450 10.068060
 [593]  4.7364420  3.3728130  2.0404840  9.8735380  3.0252760  4.2881050  8.4457740  4.4418810 21.903190
 [609]  6.6634630  6.1666570  4.0363040  6.9417680 10.4917000  6.3523300  2.5389390 12.9966000  8.351562
 [625]  0.3232862  2.2295680  6.4720420 10.1448500  4.7652580  9.0542960  5.8213720  3.8533600  3.851677
 [641]  7.8787130  6.0928630  9.6514170  5.3891270  9.2580180 14.4705900 11.1949400  4.8889160  6.760730
 [657]  7.0148670  4.8798580  2.7623380  8.8472160  5.8006890  2.6616600  4.6394580  3.9274900  5.856083
 [673]  2.5808390  3.4844690  6.0054520  9.8737080  4.4517870 11.8641300  4.4970330  2.6901500  8.475250
 [689]  6.0597390  3.2128490  1.3130840  7.6576340  3.1768000 10.8426300  2.7127420  3.5357880  3.007457
 [705]  1.2118010  5.3047110  6.1017680  3.3427430 10.8294700 14.0404200  0.5472427  4.7611210  4.963700
 [721]  8.3341100  3.5564430  5.5375320  9.8135620  0.7643371  3.6504350  1.9288630  3.4997950  7.851583
 [737]  8.1539920  9.3640090  2.8430920  9.3250400  5.5053930  0.8852075  4.5874320  9.6686950  8.937570
 [753]  8.3857360 11.6007000 10.0136800  9.5500700 11.0458600  5.1229550  7.4299060  2.9637200  8.132952
 [769]  3.9830300  5.8955900  5.0169290  6.3502270  2.4569130  4.9072590 14.1636000 10.4447800  7.331800
 [785]  3.1856500  4.7570450  2.2833860  4.2403510  6.7507820  7.2918020 11.3946800  7.2865530  1.377288
 [801]  2.3204810 11.7230900  2.0315730  7.0616240  6.0922970  8.2313760  2.9610640  4.1960350  6.382020
 [817]  6.3666470  4.9159900  8.6534780  1.4692790  3.4400730  3.4032250  2.7425830  4.8465380  3.892022
 [833]  8.3436670  6.3954260  1.8315830  7.2516760  8.9483500  7.4906500  3.7936040  6.7497230  8.004301
 [849]  4.2304170  9.5626610 11.6837600  2.0858190  6.8264370  0.9451860  7.0763980  4.7903860 11.601780
 [865]  8.3356250  4.5502910  7.7697420  1.4312350  7.2065080  6.8749240  4.7840020 11.9162600 14.858800
 [881]  1.6678930  6.3996410  1.7797260  7.5037560  7.9055430  5.7540680  3.7835080  4.2239310 12.134050
 [897]  1.2646880  2.5932480  2.2096290  6.8151250  6.4335190  3.3863190  5.8850080  5.5411560  2.956589
 [913]  3.5719500  4.0317170  7.1027260  4.1380320  8.2373370  7.3940420  6.7660740  6.0773340  6.345920
 [929]  6.9185910  6.4656970 11.3714900  4.6914530  8.8113410  7.8090360  3.2030260  8.8390010  3.962865
 [945]  7.0847110 10.4211200  9.6488570  9.9366710  3.8221930  4.7300950  4.2333230  6.9340400  2.587323
 [961]  6.0830770  8.4537570  3.3432040  8.1227380  4.7244950 11.2103600  9.2503010  4.4366040  4.886258
 [977]  3.6558360  3.5959990  2.2335050 11.0316800  3.4343430  8.3616230  6.2100880  4.0419980  8.653620
 [993]  1.9096390  3.0946710  7.5686540  4.2500460 10.5878100  4.9576460  7.6567110  7.8919400
"

# Process the data
clean_text <- gsub("\\[.*?\\]", "", raw_data)
data <- as.numeric(unlist(strsplit(clean_text, "\\s+")))
data <- data[!is.na(data)]

# Parameters
alpha <- 0.03       # Significance level
```

```
n <- 100             # Sample size

# Step 2: Take a random sample
sample <- sample(data, size = n, replace = FALSE)

# Step 3: Calculate test statistic T(x)
Tx <- sum(sample^2)
```

Confidence Interval for σ

Key Transformation: If $X \sim \text{Rayleigh}(\sigma)$, then $Y = \left(\frac{X}{\sigma}\right)^2 \sim \chi^2_2$

$$\Rightarrow \frac{X_i^2}{\sigma^2} \sim \chi^2_2 \Rightarrow \sum \frac{X_i^2}{\sigma^2} \sim \chi^2_{2n} \quad (\text{under } H_0) \Rightarrow T(x) = \sum X_i^2 \sim \sigma_0^2 \chi^2_{2n} \quad (\text{under } H_0)$$

$$\frac{\sum X_i^2}{\sigma^2} \sim \chi^2_{2n}$$

$$P\left(\frac{\sum X_i^2}{\chi^2_{\alpha/2,\,2n}} < \sigma^2 < \frac{\sum X_i^2}{\chi^2_{1-\alpha/2,\,2n}}\right) = 1 - \alpha$$

$$\Rightarrow \left[\frac{\sum X_i^2}{\chi^2_{\alpha/2,\,2n}}, \frac{\sum X_i^2}{\chi^2_{1-\alpha/2,\,2n}}\right]$$

$$\Rightarrow \left[\sqrt{\frac{\sum X_i^2}{\chi^2_{\alpha/2,\,2n}}}, \sqrt{\frac{\sum X_i^2}{\chi^2_{1-\alpha/2,\,2n}}}\right] \quad (\text{for } \sigma)$$

```
# Confidence level
alpha <- 0.03   # %97

# Sample
n <- 100
sample <- sample(data, size = n, replace = FALSE)

# Test Stat
Tx <- sum(sample^2)

# Chi_square critical value (df = 2n)
chi2_upper <- qchisq(1 - alpha/2, df = 2 * n)
chi2_lower <- qchisq(alpha/2, df = 2 * n)

# Confidence Interval for sigma^2
lower_var <- Tx / chi2_upper
upper_var <- Tx / chi2_lower

# Confidence Interval for sigma (sqrt)
lower_sigma <- sqrt(lower_var)
upper_sigma <- sqrt(upper_var)

cat(sprintf("%d%% Confidence Interval for  : [%.4f, %.4f]\n", (1 - alpha)*100, lower_sigma, upper_sigma)
## 97% Confidence Interval for  : [4.6011, 5.7195]
```

MP Test

$$H_0 : \sigma = \sigma_0 H_1 : \sigma = \sigma_1 \quad (\sigma_0 \neq \sigma_1) \quad \text{(simple vs simple)}$$

Likelihood ratio:

$$\lambda(x) = \frac{L(\sigma_1)}{L(\sigma_0)} = \left(\frac{\sigma_0^2}{\sigma_1^2}\right)^n \exp\left[\left(\frac{1}{2\sigma_1^2} - \frac{1}{2\sigma_0^2}\right)\sum x_i^2\right]$$

Let $T(x) = \sum x_i^2$:

- When $\sigma_0 > \sigma_1$, reject H_0 if $T(x) < c$
- When $\sigma_0 < \sigma_1$, reject H_0 if $T(x) > c$

Key Transformation: If $X \sim \text{Rayleigh}(\sigma)$, then $Y = \left(\frac{X}{\sigma}\right)^2 \sim \chi_2^2$

$$\Rightarrow \frac{X_i^2}{\sigma^2} \sim \chi_2^2 \Rightarrow \sum \frac{X_i^2}{\sigma^2} \sim \chi_{2n}^2 \quad (\text{under } H_0) \Rightarrow T(x) = \sum X_i^2 \sim \sigma_0^2 \chi_{2n}^2 \quad (\text{under } H_0)$$

- Reject H_0 if $\sum X_i^2 < \sigma_0^2 \chi_{2n,1-\alpha}^2$, when $\sigma_0 > \sigma_1$
- Reject H_0 if $\sum X_i^2 > \sigma_0^2 \chi_{2n,\alpha}^2$, when $\sigma_0 < \sigma_1$

```r
# --- MP Test (Two-sided) ---
sigma0 <- sqrt(Tx / (2 * n))  # MLE for sigma   # Null hypothesis value for sigma
statistic_mp <- Tx / sigma0^2
chi2_crit_low <- qchisq(alpha /2, df = 2 * n)
chi2_crit_high <- qchisq(1 - alpha /2 , df = 2 * n)
reject_mp <- (statistic_mp < chi2_crit_low) || (statistic_mp > chi2_crit_high)

cat("MP Test (Two-sided)\n")
```

```
## MP Test (Two-sided)
```

```r
cat(sprintf("T(x)/  ² = %.2f\n", statistic_mp))
```

```
## T(x)/  ² = 200.00
```

```r
cat(sprintf("Critical region: < %.2f or > %.2f\n", chi2_crit_low, chi2_crit_high))
```

```
## Critical region: < 159.10 or > 245.85
```

```r
cat("Result:", ifelse(reject_mp, "Reject H0", "Do not reject H0"), "\n\n")
```

```
## Result: Do not reject H0
```

```r
lower_sigma2 <- Tx / chi2_crit_high
upper_sigma2 <- Tx / chi2_crit_low

lower_sigma <- sqrt(lower_sigma2)
upper_sigma <- sqrt(upper_sigma2)

cat("Critical values for  ")
```

```
## Critical values for
```

```r
cat("=", "c1:", lower_sigma, "c2:" ,upper_sigma)
```

```
## = c1: 4.60109 c2: 5.719541
```

UMP Test

$$H_0 : \sigma = \sigma_0 H_1 : \sigma > \sigma_0 \text{ or } \sigma < \sigma_0 \quad (\text{Let it be } \sigma_1)$$

- Under $\sigma_0 > \sigma_1$:

$$\frac{L(\sigma_1)}{L(\sigma_0)} = \left(\frac{\sigma_0^2}{\sigma_1^2}\right)^n \exp\left[\left(\frac{1}{2\sigma_1^2} - \frac{1}{2\sigma_0^2}\right)\sum x_i^2\right]$$

- Under $\sigma_1 > \sigma_0$:

$$\frac{L(\sigma_1)}{L(\sigma_0)} = \left(\frac{\sigma_1^2}{\sigma_0^2}\right)^n \exp\left[\left(\frac{1}{2\sigma_0^2} - \frac{1}{2\sigma_1^2}\right)\sum x_i^2\right]$$

$T(x) = \sum x_i^2$, and L is a non-decreasing function of $T(x)$, so the MLRS is $T(x)$.

- If $H_1 : \sigma < \sigma_0$, reject H_0 if MLRS < c
- If $H_1 : \sigma > \sigma_0$, reject H_0 if MLRS > c

```r
# --- UMP Test (One-sided: H1:   >  ) ---
chi2_crit_ump <- qchisq(1 - alpha, df = 2 * n)
reject_ump <- statistic_mp > chi2_crit_ump

cat("UMP Test (H1:   >  )\n")
```

```
## UMP Test (H1:   >  )
```

```r
cat(sprintf("T(x)/  ² = %.2f, Critical value = %.2f\n", statistic_mp, chi2_crit_ump))
```

```
## T(x)/  ² = 200.00, Critical value = 239.27
```

```r
cat("Result:", ifelse(reject_ump, "Reject H0", "Do not reject H0"), "\n\n")
```

```
## Result: Do not reject H0
```

GLR Test

$$H_0 : \sigma = \sigma_0 \quad \text{vs.} \quad H_1 : \sigma \neq \sigma_0$$

Parameter spaces:

$$\Omega_0 = \{\sigma_0\} \quad \Omega_1 = (0, \sigma_0) \cup (\sigma_0, \infty) \quad \Omega = \Omega_0 \cup \Omega_1 = (0, \infty)$$

Likelihood:

$$L(\sigma) = \prod \frac{x_i}{\sigma^2} \exp\left(-\frac{x_i^2}{2\sigma^2}\right) \Rightarrow \ell_n L(\sigma) = \sum \ln x_i - 2n \ln \sigma - \frac{\sum x_i^2}{2\sigma^2}$$

Derivative:

$$\frac{\partial \ell_n L(\sigma)}{\partial \sigma} = -\frac{2n}{\sigma} + \frac{1}{\sigma^3} \sum x_i^2 = 0 \Rightarrow \hat{\sigma}_{MLE} = \sqrt{\frac{1}{2n} \sum x_i^2}$$

GLR statistic:

$$\lambda = \frac{L(\sigma_0)}{L(\hat{\sigma})} = \left(\frac{\hat{\sigma}^2}{\sigma_0^2}\right)^n \exp\left[\sum x_i^2 \left(\frac{1}{2\hat{\sigma}^2} - \frac{1}{2\sigma_0^2}\right)\right]$$

This simplifies to:

$$\left(\frac{\sum x_i^2}{2n\sigma_0^2}\right)^n \exp\left[n - \frac{\sum x_i^2}{2\sigma_0^2}\right] < \lambda_0$$

Taking log:

$$\ln \lambda = n \ln \left(\frac{\sum x_i^2}{2n\sigma_0^2}\right) + n - \frac{\sum x_i^2}{2\sigma_0^2} < \ln \lambda_0$$

Define:

$$\ln \left(\frac{\sum x_i^2}{2n\sigma_0^2}\right) - \frac{\sum x_i^2}{2n\sigma_0^2} < \frac{\ln \lambda_0}{n} - 1$$

Let $Y = \frac{\sum x_i^2}{2n\sigma_0^2}$, $Z = \ln Y - Y$, then:

$$P(\text{Reject } H_0 | H_0 \text{ true}) = P(Y < k_1) + P(Y > k_2) = \alpha$$

Since $Y \cdot 2n \sim \chi^2_{2n}$,

$$\text{Reject } H_0 : \frac{\sum x_i^2}{\sigma_0^2} < \chi^2_{2n, 1-\alpha/2} \text{ or } > \chi^2_{2n, \alpha/2}$$

Final form:

$$-2 \ln \lambda = -2n - \ln \sum x_i^2 + 2 \ln 2n\sigma_0^2 + \frac{\sum x_i^2}{\sigma_0^2} \sim \chi^2_1$$

Then:

$$\alpha = P(\text{Reject } H_0 | H_0 \text{ TRUE}) = P(\lambda < \lambda_0 | \sigma = \sigma_0) = P(-2 \ln \lambda > -2 \ln \lambda_0 | \sigma_0)$$

Decision rule:

$$\text{Reject } H_0 : -2 \ln \lambda > \chi^2_{1,\alpha}$$

```
# --- GLR Test ---
sigma_hat <- sqrt(Tx / (2 * n))  # MLE for sigma
lambda_glr <- (sigma0 / sigma_hat)^(2 * n) * exp((1 - (sigma0^2 / sigma_hat^2)) * Tx / (2 * sigma0^2))
test_stat_glr <- -2 * log(lambda_glr)
crit_val_glr <- qchisq(1 - alpha, df = 1)
reject_glr <- test_stat_glr > crit_val_glr

cat("GLR Test\n")
```

```
## GLR Test
```

```
cat(sprintf("-2 * ln( ) = %.2f, Critical value = %.2f\n", test_stat_glr, crit_val_glr))
```

```
## -2 * ln( ) = -0.00, Critical value = 4.71
```

```
cat("Result:", ifelse(reject_glr, "Reject H0", "Fail to reject H0"), "\n")
```

```
## Result: Fail to reject H0
```

```
# Test statistic
test_stat <- sum(sample^2) / sigma0^2
cat("Test Statistic :" ,test_stat ,"\n")
```

```
## Test Statistic : 200
```

```
# Critical values
upper_crit <- qchisq(1- alpha / 2, df = n*2)
cat("Upper Critical Value:" ,upper_crit ,"\n")
```

```
## Upper Critical Value: 245.8451
```

```
lower_crit <- qchisq(alpha / 2, df = n*2)
cat("Lower Critical Value:" ,lower_crit ,"\n")
```

```
## Lower Critical Value: 159.0965
```

```
# Decision
if (test_stat < lower_crit || test_stat > upper_crit) {
  cat("Reject H0.\n")
} else {
  cat("Fail to reject H0.\n")
}
```

```
## Fail to reject H0.
```

Uniform(a, b) Distribution Testing

Let $X \sim \text{Uniform}(a, b)$, where $a = 2$. The PDF is:

$$f(x; b) = \begin{cases} \frac{1}{b-2}, & 2 \le x \le b \\ 0, & \text{otherwise} \end{cases}$$

The CDF is:

$$F(x; b) = \begin{cases} \frac{x-2}{b-2}, & 2 \le x \le b \\ 0, & \text{otherwise} \end{cases}$$

```r
set.seed(42)

raw_text = " [1] 2.663621 3.719691 2.017035 2.164160 3.678541 3.378565 2.117577 2.202661 2.275739
  [10] 3.674937 2.897971 2.905268 2.059480 2.867312 2.162826 3.274812 3.713663 2.086917
  [19] 2.531441 2.549707 2.699910 2.376508 3.135720 3.205476 2.827482 3.235600 2.795574
  [28] 3.675428 2.192815 2.354276 3.090142 3.016486 2.912484 2.277020 2.067384 3.810705
  [37] 3.216785 2.479067 2.467721 3.465114 2.993363 3.658151 3.641604 2.519516 2.906663
  [46] 3.724780 2.570132 3.039599 3.463408 2.754499 2.828312 3.569176 3.008819 3.463273
  [55] 3.651341 3.465800 3.486750 3.556692 3.079041 2.096812 2.111632 3.484208 3.603337
  [64] 2.899568 2.057530 2.856121 2.976772 2.851672 3.904215 2.721099 2.906733 3.631381
  [73] 3.158846 3.622023 2.020901 2.964106 2.597594 2.083036 2.754207 2.239494 3.113160
  [82] 2.362410 3.780881 2.869399 3.525334 3.069901 3.345015 3.365031 2.989089 3.795102
  [91] 2.723967 3.350508 3.188972 2.095746 2.668814 3.670119 3.700714 3.261475 2.231906
 [100] 2.963503 3.575509 3.081385 2.632728 2.773702 3.711460 3.922208 2.080387 2.906779
 [109] 2.711601 3.401000 2.047337 2.263007 2.533348 2.331349 2.544480 2.398135 2.802361
 [118] 2.393106 2.996377 3.283875 3.987811 3.638404 2.439576 2.117281 3.705830 2.195954
 [127] 2.175716 3.202325 2.478414 2.275158 3.850307 3.939562 2.796597 2.103550 2.309119
 [136] 2.975220 2.956471 2.597938 2.781806 3.656677 2.122953 2.496162 2.790332 3.908046
 [145] 3.257915 2.143870 2.621087 3.240883 2.637803 3.713815 3.957882 2.587821 3.261055
 [154] 3.047381 2.260638 2.517469 3.355109 2.129659 2.623983 2.465968 3.310790 3.320290
 [163] 2.840429 3.642476 3.767820 2.083566 3.095760 2.119193 3.314760 3.146432 2.463800
 [172] 2.687378 3.645619 2.904312 2.426555 2.177044 2.540458 3.859999 3.230073 3.717644
 [181] 2.313047 2.533545 2.092364 2.028786 2.601089 3.411821 2.328651 3.449246 2.207781
 [190] 3.636488 2.122207 2.814976 3.644852 2.358352 2.054776 3.510887 3.809067 3.073782
 [199] 2.544307 3.117265 2.797197 2.673574 3.400855 3.113256 2.320663 3.144448 3.929980
 [208] 3.715611 3.946040 3.990222 3.581176 2.991170 3.797662 3.469349 3.588758 3.483175
 [217] 2.205566 2.600155 3.561080 2.106795 3.681636 3.411598 2.421913 3.086589 3.504433
 [226] 3.845067 2.946498 2.753043 3.119020 3.951742 2.137848 3.906378 3.846519 3.890667
 [235] 2.696465 3.807961 3.906170 2.077847 2.646424 2.617993 3.780063 3.506835 3.047141
 [244] 3.388590 3.844747 2.111301 2.593678 2.443815 2.838243 3.929339 3.190321 3.627884
 [253] 2.471517 3.163949 2.112584 2.555148 2.621020 3.933363 2.163520 2.030342 3.660950
 [262] 2.191912 2.418312 2.795992 2.776286 3.132091 3.752974 3.841899 2.600588 2.373443
 [271] 2.989930 3.359068 2.000142 2.686738 3.073891 3.414127 2.576576 2.852739 3.648324
 [280] 2.884435 2.065092 2.244715 2.003479 2.947998 3.187076 2.359932 2.048152 3.405613
 [289] 2.094115 2.670801 3.672257 2.829378 2.622770 2.755119 3.949487 3.191897 3.403655
 [298] 2.822619 2.849621 3.232000 2.239239 2.031237 3.709551 3.794981 2.964269 2.337420
 [307] 3.427339 2.344190 3.362000 3.912588 3.420951 2.947922 3.184728 2.173259 3.271149
 [316] 2.926523 2.883571 2.898221 3.631705 3.385374 2.083779 3.624535 3.774494 3.452251
 [325] 3.922635 3.277668 3.996363 2.173574 2.031969 3.122913 2.864853 2.493077 2.204682
 [334] 2.797468 2.919548 2.664990 3.991716 2.244455 2.231676 3.548801 3.925715 2.295617
 [343] 3.350612 2.180727 2.880788 3.088446 2.905243 2.216631 3.654952 2.305906 3.507629
 [352] 2.575571 2.562985 2.710327 2.788357 3.766948 2.173219 2.956216 3.546882 2.738688
 [361] 2.603296 3.507473 3.929698 2.731363 3.025692 3.026686 2.621363 2.857946 3.667466
 [370] 3.948852 2.697422 3.979490 2.473204 2.392021 2.987477 3.469977 3.176836 2.222916
 [379] 3.551299 3.256679 3.541934 3.051749 2.391193 2.640895 3.465495 2.924979 2.217012
 [388] 2.977353 3.407280 2.603286 3.778744 2.413910 3.606457 2.756039 3.808171 3.274811
 [397] 3.907092 3.858524 2.085288 2.482097 2.573389 3.312762 2.971628 3.486146 3.293284
 [406] 3.132844 2.303481 2.771884 3.015393 3.277188 2.941679 2.481365 2.193130 2.318667
 [415] 2.820162 3.631782 3.135979 3.769292 3.759790 2.190188 3.789205 2.537375 2.490709
 [424] 2.373372 3.433263 3.452605 3.188092 2.695100 2.878318 2.308094 2.128544 3.055415
 [433] 3.258182 2.748111 3.741216 2.623718 3.134335 3.619225 3.206428 3.771856 2.121059
 [442] 2.736096 2.581522 3.722913 3.405482 3.108372 2.264826 3.197247 2.890230 2.744098
 [451] 3.865511 3.790807 3.608629 2.560825 3.731265 2.025492 2.899368 3.195058 2.471034
```

```
[460] 3.911730 2.731687 3.593911 2.702266 2.654844 2.606919 2.046090 2.559190 3.445816
[469] 3.217578 3.218765 3.705715 2.067280 3.754891 2.249893 3.466682 2.919506 3.854194
[478] 3.407955 3.433376 2.077168 3.719660 2.064889 3.096523 3.972399 2.856720 3.903453
[487] 3.142344 2.620556 2.806127 2.611029 2.767564 3.824011 2.936287 2.009581 3.024101
[496] 2.257305 2.833677 3.462514 3.199104 2.040301 2.363172 3.551644 3.612526 2.611930
[505] 3.156146 2.098094 3.879702 3.939784 2.569947 3.983748 2.612695 2.252280 2.374017
[514] 2.927339 2.099046 2.754386 3.394857 2.894617 2.107886 2.747617 3.220423 2.566434
[523] 3.240223 3.372539 3.150674 3.682545 3.050635 2.104853 3.639177 2.964202 2.955960
[532] 2.587896 2.893608 2.182181 3.952425 2.624209 2.631968 3.804051 3.828991 3.296230
[541] 3.890326 3.692126 2.468652 2.480452 2.129814 2.727162 3.311260 2.698144 3.044802
[550] 2.556288 2.973136 3.124453 3.950354 2.635932 3.942196 2.952486 2.052242 2.507722
[559] 3.825739 3.924928 3.184960 2.085447 2.754121 3.709696 3.352922 3.241102 3.030106
[568] 2.616632 2.426163 3.031755 2.927097 3.106903 2.619579 3.529550 2.259768 3.360798
[577] 3.420122 3.292172 2.507914 2.592781 2.762888 3.298596 3.241231 3.071329 2.477496
[586] 2.117015 3.324431 2.526936 3.528335 3.710070 3.268832 2.419218 3.059785 2.026687
[595] 2.996845 2.287508 3.522110 2.536549 3.637732 3.009265 3.323508 2.322827 3.219932
[604] 2.913942 2.194856 3.316801 2.408771 3.975937 2.424495 2.372593 3.936162 2.348808
[613] 2.487431 2.115482 2.174412 2.751236 2.335225 2.513667 3.334714 3.577065 2.431711
[622] 3.807677 3.721653 3.297705 2.968679 3.943218 2.777172 2.020256 3.919411 2.025720
[631] 2.231937 2.322247 2.178401 3.862908 2.599086 2.742460 3.613763 2.341518 2.154332
[640] 2.108338 3.957688 2.312757 2.435865 2.777558 2.424704 2.197730 2.648260 3.930130
[649] 3.298536 3.520007 3.820596 2.911657 2.919349 2.369423 2.614106 3.462244 2.533629
[658] 2.060231 2.248170 2.794997 2.032308 2.427261 2.898747 2.544656 3.282118 3.473619
[667] 3.408229 2.432161 2.826300 3.740850 3.988504 2.899310 3.369924 2.698727 3.955615
[676] 2.941528 2.388658 2.270501 2.595392 3.542870 3.607551 2.290938 3.597335 3.891193
[685] 3.406865 2.254553 2.429042 3.117934 3.537524 2.150339 3.668073 2.021571 3.838646
[694] 3.528124 2.600114 3.044467 2.538443 3.963835 3.343779 3.973384 3.259711 2.196727
[703] 3.407215 3.287100 2.659139 3.120794 3.123574 2.027499 2.154761 2.932777 3.714275
[712] 2.115376 2.511614 2.435213 3.630536 2.524426 3.268395 2.454139 2.206882 2.677176
[721] 2.303091 3.946637 2.642464 2.817032 2.103334 3.055083 2.624042 2.466906 3.594984
[730] 3.622354 2.457629 2.556736 2.008185 3.907153 2.379803 3.130619 2.577052 3.527095
[739] 3.364947 2.326627 2.190322 3.026715 3.749403 2.086865 2.322406 2.083619 2.997685
[748] 3.534270 3.222302 2.798390 3.731822 3.145327 3.724936 2.682607 3.330709 3.803648
[757] 3.598026 2.042255 2.621577 3.417986 2.090147 2.429382 3.080479 2.931406 3.666316
[766] 3.687842 3.794752 2.612679 2.434277 2.211080 3.588855 3.287702 3.820640 3.329308
[775] 3.811281 2.895364 2.923476 3.711414 3.408864 2.356313 3.790532 2.671648 2.860639
[784] 3.001603 2.501607 2.272224 2.519897 3.700921 3.002709 3.211645 2.268247 3.244728
[793] 2.970524 3.086987 3.486535 3.188821 2.937428 2.112923 2.742558 2.883942 2.204031
[802] 3.858861 3.690473 3.677587 2.910144 2.422590 3.249226 3.174314 2.577616 3.325207
[811] 3.576910 2.934369 3.812576 2.853111 2.631798 2.393923 3.977034 2.613453 2.905863
[820] 3.147328 3.885572 3.333348 3.884459 3.015449 3.853529 3.246359 2.547410 2.504726
[829] 3.341873 2.141223 3.923367 3.225120 2.879114 2.585154 2.168133 3.361611 3.413252
[838] 3.359384 2.824799 2.303277 2.537249 2.340689 3.939308 3.198269 2.067624 3.584507
[847] 3.568521 2.857178 2.947313 3.005177 2.211407 3.765567 3.900815 2.633975 3.866715
[856] 2.935907 3.680821 3.864761 3.370870 3.059530 3.573389 2.311139 2.372755 3.491877
[865] 3.395655 2.305772 3.289605 2.959034 3.220917 2.711930 3.236365 3.947636 2.734149
[874] 3.765392 2.443012 2.216904 3.593100 3.338195 3.189790 2.973668 2.476051 3.606860
[883] 3.431178 2.553765 3.435443 2.391645 2.814854 2.869893 2.152242 2.434799 2.108694
[892] 3.959800 3.568305 2.839860 2.655718 2.701637 2.643903 3.445083 3.196835 2.759733
[901] 2.207070 3.662692 3.359245 2.501345 3.769289 2.571965 3.371700 2.390206 3.336850
[910] 2.197374 2.725372 2.888732 2.044582 2.482600 2.334211 2.641208 2.088972 3.262926
[919] 3.895987 2.816219 2.877834 2.702481 3.389631 3.313000 3.917481 2.959654 2.580787
[928] 3.979322 3.642783 3.678813 2.542172 3.566290 3.162492 2.547413 3.209201 2.268793
```

```
 [937] 2.001616 2.942817 3.202387 2.687960 2.859732 3.916186 3.409755 3.026521 3.523265
 [946] 3.023305 2.329919 2.012723 3.939088 3.176342 3.043906 2.251255 2.477426 2.292342
 [955] 2.634596 3.338837 2.065543 2.129009 2.700892 2.564697 3.456502 2.695993 2.428529
 [964] 3.059795 3.707981 2.764431 2.135229 3.368364 3.842271 3.237610 2.259229 3.836547
 [973] 2.447999 2.519087 3.465969 2.006225 2.578073 3.869987 3.056396 3.020982 2.742596
 [982] 2.938689 3.175106 2.034125 2.520023 2.721818 3.412443 2.985365 2.811582 2.944410
 [991] 2.578869 3.559594 2.781051 3.421633 3.053965 3.377630 2.574170 2.928157 2.912884
[1000] 2.715684

"

clean_text <- gsub("\\[.*?\\]", "", raw_text)

# Step 3: Convert to numeric vector
data <- as.numeric(unlist(strsplit(clean_text, "\\s+")))
data <- data[!is.na(data)]
```

```
# Parameters
alpha <- 0.03       # Significance level
n <- 100            # Sample size

# Step 2: Take a random sample
sample <- sample(data, size = n, replace = FALSE)

# Parameters
a <- 2
b_true <- 3.99
n <- 100
alpha <- 0.03
```

Confidence Interval for b

Let $X_{(n)}$ denote the maximum of an i.i.d. sample of size n and it is our sufficent statistic and MLE. Then:

$$F_{X_{(n)}}(x) = \left(\frac{x-2}{b-2}\right)^n, \quad 2 \leq x \leq b$$

Define:

$$Y = \frac{X_{(n)} - 2}{b-2} \sim \text{Beta}(n, 1)$$

Then:

$$P(Y \leq y) = y^n \Rightarrow P(X_{(n)} \leq 2 + y(b-2)) = y^n$$

Solving for b, we get a $(1-\alpha)$ upper confidence bound:

$$P\left(b \geq 2 + \frac{X_{(n)} - 2}{(1-\alpha)^{1/n}}\right) = 1 - \alpha$$

Thus, a two-sided $(1-\alpha)$ confidence interval is:

$$\left[2 + \frac{X_{(n)} - 2}{(1+\alpha/2)^{1/n}},\ 2 + \frac{X_{(n)} - 2}{(1-\alpha/2)^{1/n}}\right]$$

```r
x_max <- max(sample)

# Compute confidence interval
lower_bound <- 2 + (x_max - 2) / (1 + alpha/2)^(1/n)
upper_bound <- 2 + (x_max - 2) / (1 - alpha/2)^(1/n)
ci <- c(lower_bound, upper_bound)
ci
```

```
## [1] 3.979195 3.979789
```

Most Powerful Test for b

We want to test:

$$H_0 : b = b_0, \quad H_1 : b = b_1 \quad (b_0 \neq b_1)(\text{simple vs simple})$$

Let's assume $b_1 > b_0$ Then:

$$H_0 : b = b_0, \quad X_{(n)} \leq b_0, \quad \text{and} \quad H_1 : b = b_1, \quad X_{(n)} \leq b_1$$

So any observation X $>b_0$ is impossible under H_0, but possible under H_1 , so we should reject H_0 if $X_{(n)}>c$, for some critical value c. Since under H_1, $X_{(n)} \leq b_1$, we should reject H_0 if $X_{(n)} \leq c$, for some critical value $c \in (b_0, b_1)$.

Find $c \in [2, b_0] \cup (b_0, b_1]$ such that

$$P(X_{(n)} > c \mid H_0 \text{ TRUE}) = \alpha$$

$$\Rightarrow \quad P(X_{(n)} \leq c) = 1 - \alpha$$

$$\Rightarrow \quad \left(\frac{c-2}{b_0-2}\right)^n = 1 - \alpha \quad \Rightarrow \quad c = 2 + (1-\alpha)^{1/n}(b_0 - 2)$$

$$P(X_{(n)} \leq c \mid H_0) = 1 - \alpha$$

We have:

$$P\left(\frac{X_{(n)} - 2}{b_0 - 2} \leq y\right) = y^n \Rightarrow y = (1-\alpha)^{1/n}$$

Then:

$$\frac{X_{(n)} - 2}{b_0 - 2} \leq (1-\alpha)^{1/n} \Rightarrow c = 2 + (b_0 - 2) \cdot (1-\alpha)^{1/n}$$

```r
# Step 2: Compute the critical value c
c <- a + (b_true - a) * (1 - alpha)^(1 / n)

# Step 3: Make the decision
reject_H0 <- (x_max > c)

# Step 4: Print results
cat("Maximum statistic X(n):", x_max, "\n")
```

```
## Maximum statistic X(n): 3.97949
```

```r
cat("Critical value c:", c, "\n")
```

```
## Critical value c: 3.989394
```

```r
cat("Reject H0?", reject_H0, "\n")
```

```
## Reject H0? FALSE
```

```r
# Optional: wrap in a list if you'd like to store
results <- list(
  x_max = x_max,
  critical_value = c,
  reject_H0 = reject_H0
)
```

Uniformly Most Powerful Test

Hypotheses:

$$H_0 : b = b_0 \quad \text{vs} \quad H_1 : b < b_0 \quad (\text{or } b > b_0)$$

This is most natural because for Uniform distributions, we typically test the endpoint based on the maximum order statistic.

To construct a test, we use the likelihood ratio:

$$\lambda = \frac{\sup_{b<b_0} L(b)}{\sup_{b\geq b_0} L(b)} = \frac{L(b)}{L(X_{(n)})}, \quad \text{if } X_{(n)} \leq b_0 \quad (H_1 : b < b_0 \text{ (1st scenario)})$$

Since $L(b)$ is decreasing in b, for a fixed sample, the MLE of b is:

$$\hat{b}_{MLE} = X_{(n)} = \max X_i = T(X)$$

So:

$$\lambda = \left(\frac{X_{(n)} - 2}{b_0 - 2}\right)^n, \quad \text{for } X_{(n)} \leq b_0$$

Reject H_0 if $\lambda < c \iff X_{(n)} < c'$ for some threshold c'.

So the **UMP Test** is:

- Reject $H_0 : b = b_0$ in favor of $H_1 : b < b_0$ if $X_{(n)} < c$

For a test of size α, we find c_1 such that:

$$P(X_{(n)} < c_1 \mid H_0 \text{ TRUE}) = \alpha$$

Set:

$$\left(\frac{c_1 - 2}{b_0 - 2}\right)^n = \alpha \Rightarrow c_1 = (b_0 - 2)\alpha^{1/n} + 2$$

2nd Scenario:

For a test of size α, we find c_2 such that:

$$P(X_{(n)} > c_1 \mid H_0 \text{ TRUE}) = \alpha \Rightarrow P(X_{(n)} < c_2) = 1 - \alpha$$

Set:

$$\left(\frac{c_2 - 2}{b_0 - 2}\right)^n = 1 - \alpha \Rightarrow c_2 = (b_0 - 2)(1 - \alpha)^{1/n} + 2$$

```
# Function to compute UMP test threshold and decision
ump_test <- function(data, b_0, alpha = 0.03, alternative = c("less", "greater")) {
  n <- length(data)
  x_max <- max(data)
  alternative <- match.arg(alternative)

  if (alternative == "less") {
    # H1: b < b0 --> reject if X(n) < c1
    c1 <- (b_0 - 2) * alpha^(1/n) + 2
    reject <- x_max < c1
    cat("Alternative: H1: b < b0\n")
    cat(sprintf("Critical value c1 = %.4f\n", c1))
  } else if (alternative == "greater") {
    # H1: b > b0 --> reject if X(n) > c2
    c2 <- (b_0 - 2) * (1 - alpha)^(1/n) + 2
    reject <- x_max > c2
    cat("Alternative: H1: b > b0\n")
    cat(sprintf("Critical value c2 = %.4f\n", c2))
  }

  cat(sprintf("X(n) = %.4f\n", x_max))
  if (reject) {
    cat("Result: Reject H0\n")
  } else {
    cat("Result: Fail to reject H0\n")
  }
}

ump_test(data = sample, b_0 = 3.99, alpha = 0.03, alternative = "less")
```

```
## Alternative: H1: b < b0
## Critical value c1 = 3.9214
## X(n) = 3.9795
## Result: Fail to reject H0
```

```
ump_test(data = sample, b_0 = 3.99, alpha = 0.03, alternative = "greater")
```

```
## Alternative: H1: b > b0
## Critical value c2 = 3.9894
## X(n) = 3.9795
## Result: Fail to reject H0
```

Generalized Likelihood Ratio Test (GLRT)

Let: - $X_1, \dots, X_n \overset{i.i.d.}{\sim} \text{Uniform}(2, b)$ - We test the hypotheses:

$$H_0 : b = b_0 \quad \text{vs} \quad H_1 : b \neq b_0$$

The maximum likelihood estimator is $\hat{b} = X_{(n)}$. The likelihood ratio statistic is:

$$\Lambda(x) = \frac{L(b_0)}{L(\hat{b})} = \begin{cases} \left(\frac{X_{(n)}-2}{b_0-2}\right)^n, & X_{(n)} \leq b_0 \\ 0, & \text{otherwise} \end{cases}$$

We reject H_0 when:

$$X_{(n)} < c_1 \quad \text{or} \quad X_{(n)} > c_2$$

Critical values:

$$c_1 = 2 + (b_0 - 2) \cdot \left(\frac{\alpha}{2}\right)^{1/n}, \quad c_2 = 2 + (b_0 - 2) \cdot \left(1 - \frac{\alpha}{2}\right)^{1/n}$$

```
x_max <- max(sample)

c1 <- a + (b_true - a) * (alpha / 2)^(1/n)
c2 <- a + (b_true - a) * (1 - alpha / 2)^(1/n)
reject_H0 <- (x_max < c1) | (x_max > c2)

list(x_max = x_max, c1 = c1, c2 = c2, reject_H0 = reject_H0)
```

```
## $x_max
## [1] 3.97949
##
## $c1
## [1] 3.908156
##
## $c2
## [1] 3.989699
##
## $reject_H0
## [1] FALSE
```

Binomial Hypothesis Testing

```r
# Example: Assume binomial_data is already loaded as a numeric vector of 0s and 1s
# Uncomment the following line if reading from a CSV file:
# binomial_data <- read.csv("your_data.csv")$your_column_name
raw_text= "   [1] 38 40 35 42 34 39 35 36 37 39 37 36 39 37 38 38 34 40 43 40 40 33 37 41 40 38 37 38 3
  [30] 41 37 37 40 39 34 38 41 33 36 40 36 34 38 37 34 33 41 37 36 39 36 31 38 38 35 42 33 35
  [59] 40 38 38 37 38 36 37 37 37 39 42 39 40 33 41 38 34 32 37 36 37 30 34 39 43 31 43 37 39
  [88] 39 35 36 33 42 39 33 40 41 37 35 39 42 44 32 38 36 38 42 38 36 38 37 30 34 39 32 34 34
 [117] 34 36 38 39 36 36 35 33 43 37 36 32 36 37 38 41 37 40 34 37 36 38 35 38 37 37 35 33 41
 [146] 42 38 38 38 35 41 35 43 44 35 35 39 34 40 38 37 39 38 39 36 32 39 40 39 41 38 36 33 41
 [175] 27 46 40 42 40 39 39 36 36 39 34 37 32 34 44 40 36 33 41 40 39 41 31 38 36 35 38 41 40
 [204] 33 36 36 37 31 41 41 38 35 39 39 36 31 36 38 34 38 38 34 35 39 34 38 39 37 38 30 35 38
 [233] 37 33 41 42 38 40 33 39 32 35 39 41 37 36 35 31 29 40 34 40 36 41 41 38 34 36 38 35 33
 [262] 37 37 36 38 42 41 38 41 33 37 38 37 38 35 33 39 43 39 39 40 35 32 39 40 36 39 44 43 34
 [291] 38 36 37 39 41 31 38 34 37 35 35 33 34 37 41 44 40 37 40 34 34 37 38 34 40 39 38 38 37
 [320] 39 39 36 43 31 37 32 38 41 38 39 41 35 40 42 37 42 39 37 37 33 41 36 42 35 40 35 38 34
 [349] 38 33 32 36 31 38 40 36 38 39 39 36 33 40 32 43 38 38 38 38 37 39 36 39 39 37 39 34 35
 [378] 36 35 37 38 34 40 40 42 37 41 36 31 36 38 37 40 34 32 36 39 39 39 37 39 36 40 42 34 29
 [407] 33 38 38 35 38 43 36 39 35 40 43 36 39 33 39 36 40 35 42 35 41 41 37 38 34 38 38 35 39
 [436] 37 38 33 36 38 36 37 40 34 38 40 37 39 36 33 37 34 37 36 35 42 36 39 40 37 41 38 40 42
 [465] 40 35 33 38 36 35 35 41 40 40 38 36 31 35 40 38 34 33 36 38 37 36 39 34 37 33 33 33 37
 [494] 40 41 27 35 42 39 34 38 33 37 35 40 42 38 31 38 40 33 37 38 39 39 39 36 43 40 36 39 38
 [523] 35 36 40 35 42 37 31 35 37 35 32 42 39 35 38 38 39 44 38 40 43 38 37 36 39 39 36 39 39
 [552] 39 38 33 41 40 43 38 35 41 37 29 36 41 36 30 38 32 42 37 37 38 43 40 41 39 37 35 32 39
 [581] 45 36 39 41 40 37 40 41 40 39 41 39 37 37 40 39 40 39 35 40 39 33 39 32 34 40 41 37 36
 [610] 36 35 36 40 40 38 35 33 32 35 40 36 34 37 38 37 41 41 34 34 37 45 37 37 35 35 39 36 35
 [639] 40 39 38 34 37 40 39 41 31 35 33 34 32 41 40 38 39 34 35 36 38 44 38 37 41 39 41 35 39
 [668] 39 37 34 41 31 41 35 37 39 33 36 32 40 40 36 41 36 36 35 36 39 37 35 38 40 41 39 38 33
 [697] 38 37 37 32 38 33 38 38 34 39 40 36 38 35 41 36 35 35 41 40 38 34 35 31 34 36 38 43 33
 [726] 38 38 36 34 38 41 34 41 38 40 36 37 36 38 28 35 39 40 38 37 31 34 40 36 37 39 38 40 37
 [755] 36 43 42 41 42 35 37 39 40 34 36 39 35 34 35 40 39 39 39 39 36 37 40 40 35 40 38 36 41
 [784] 38 37 38 40 39 35 42 41 35 34 38 38 38 38 41 32 36 34 44 37 41 39 38 40 35 43 38 38 35
 [813] 34 36 29 37 38 33 36 41 38 37 38 41 35 37 35 40 40 38 36 39 41 39 40 34 30 36 37 26 38
 [842] 35 36 33 34 39 39 40 40 39 38 39 40 39 37 40 40 37 38 34 37 35 43 38 37 42 38 40 36 41
 [871] 30 42 36 34 35 39 28 34 33 43 37 35 31 42 37 34 41 40 39 40 33 35 39 37 34 38 39 41 39
 [900] 38 36 41 38 39 38 38 39 36 32 36 38 43 41 35 35 39 38 37 34 41 38 39 44 36 39 42 37 37
 [929] 39 40 39 37 35 42 36 38 40 35 35 38 37 38 37 43 36 40 34 43 38 36 40 43 39 40 35 38 40
 [958] 34 32 40 42 34 39 43 39 39 37 39 38 39 38 34 38 42 41 38 34 38 35 42 35 42 36 29 35 35
 [987] 36 36 34 34 37 33 38 38 40 38 41 44 32 43"

clean_text <- gsub("\\[.*?\\]", "", raw_text)
binomial_data <- as.numeric(unlist(strsplit(clean_text, "\\s+")))
binomial_data <- binomial_data[!is.na(binomial_data)]

# Set seed for reproducibility
set.seed(123)

# Step 1: Sample 100 observations from your binomial_data (size 1000)
sample_data <- sample(binomial_data, size = 100, replace = FALSE)
```

Confidence Interval

The $100(1-\alpha)\%$ confidence interval for the success probability p of a Binomial$(50, p)$ distribution, based on the normal approximation, is given by:

$$\hat{p} \pm z_{\alpha/2} \cdot \sqrt{\frac{\hat{p}(1-\hat{p})}{n \cdot 50}}$$

Where: - $\hat{p} = \frac{\bar{x}}{50}$ is the estimator of p - $z_{\alpha/2}$ is the critical value from the standard normal distribution - n is the number of Binomial observations (sample size) - Each observation is based on 50 trials

```r
n <- length(sample_data) # Number of observations
trials <- 50 # Binomial trials per observation
xbar <- mean(sample_data) # Sample mean
p_hat <- xbar / trials # Estimated p

# Standard error for p_hat

SE <- sqrt(p_hat * (1 - p_hat) / (n * trials))

# 97% confidence interval

z <- qnorm(0.985)
lower<- p_hat - z * SE
upper <- p_hat + z * SE

cat("Estimated p:", p_hat, "\n")
```

```
## Estimated p: 0.7458
```

```r
cat("97% CI for p: [", lower, ",", upper, "]\n")
```

```
## 97% CI for p: [ 0.7324374 , 0.7591626 ]
```

Parameters

Let:

- $n = 100$ (number of trials)
- $p = 0.75$ (probability of success under H_0)
- $\alpha = 0.03$ (significance level)

Finding the Critical Value

```r
n <- 100
p <- 0.75
alpha <- 0.03
```

```r
for (c in 0:n) {
  prob <- pbinom(c, n, p)
  if (prob > alpha) {
    cat("Critical Value c =", c - 1, "\n")
    cat("P(X <= c) =", pbinom(c - 1, n, p), "\n")
    break
  }
}
```

```
## Critical Value c = 66
## P(X <= c) = 0.02759456
```

Check binomial probabilities

```r
pbinom(66, size = 100, prob = 0.75)
```

```
## [1] 0.02759456
```

```r
pbinom(67, size = 100, prob = 0.75)
```

```
## [1] 0.04459633
```

```r
dbinom(67, size = 100, prob = 0.75)
```

```
## [1] 0.01700176
```

Mathematical Derivation

We are testing:

$$H_0 : X \sim \text{Bin}(100, 0.75) \quad \text{vs.} \quad H_1 : X \sim \text{Bin}(100, 0.5)$$

Let $Y = \sum X_i$, then $Y \sim \text{Bin}(100, 0.75)$ under H_0.
We want a randomized test such that:

$$\alpha = P(Y \leq c) + k \cdot P(Y = c + 1)$$

From R calculations:

- $P(Y \leq 66) = 0.02759456$
- $P(Y \leq 67) = 0.04459633$
- $P(Y = 67) = 0.01700176$

So to find k:

$$0.03 = 0.02759456 + k \cdot 0.01700176$$

Solving:

$$k = \frac{0.03 - 0.02759456}{0.01700176} = 0.14148182$$

Final Form of the Randomized Test

The test function $\phi(y)$ is:

$$\phi(y) = \begin{cases} 1, & y < 67 \\ 0.1415, & y = 67 \\ 0, & y > 67 \end{cases}$$

Applying the Randomized Test on a Dataset

```r
# Set seed for reproducibility
set.seed(123)

# Step 1: Sample 100 observations from your binomial_data (size 1000)
sample_data <- sample(binomial_data, size = 100, replace = FALSE)
# Sum of observed successes
Y_obs <- sum(sample_data)/50
cat("Observed Y =", Y_obs, "\n")
```

```
## Observed Y = 74.58
```

```r
# Randomized test decision function
phi <- function(y) {
  if (y < 67) return(1)
  else if (y == 67) return(0.1415)
  else return(0)
}

# Compute decision
decision <- phi(Y_obs)
cat("Test function  (Y) =", decision, "\n")
```

```
## Test function  (Y) = 0
```

```r
# Interpretation
if (decision == 1) {
  cat("Reject H0 with probability 1.\n")
} else if (decision == 0) {
  cat("Do not reject H0.\n")
} else {
  cat("Reject H0 with probability", decision, "(randomized).\n")
}
```

```
## Do not reject H0.
```

Comparison of Confidence Intervals and Hypothesis Tests

We compare the 97% confidence intervals and two-sided hypothesis tests for three distributions: **Weibull**, **Rayleigh**, and **Uniform**.

- **Weibull Distribution**

  - MLE of the scale parameter: ˆ = **4.1797**
  - 97% Confidence Interval for : [**3.7699, 4.6863**]
  - Null hypothesis: **H : = 4**
  - Test statistic: **T(x)/ ² = 100**, critical values: **< 82.06 or > 119.64**
  - Decision: **Fail to reject H**
  - Since = 4 lies within the CI and the test statistic is inside the acceptance region, both methods agree.

- **Rayleigh Distribution**

  - MLE of the scale parameter: ˆ = **4.8161**
  - 97% Confidence Interval for : [**4.6011, 5.7195**]
  - Null hypothesis: **H : = 5**
  - Test statistic: **T(x)/ ² = 200**, critical region: **< 159.10 or > 245.85**
  - Decision: **Fail to reject H**
  - Both the CI and hypothesis test support the null; the result is consistent.

- **Uniform Distribution**

  - Known lower bound: **a = 2**
  - Null hypothesis: **H : b = 3.99**
  - MLE of upper bound: **$\hat{b}$ = X = 3.9795**
  - 97% Confidence Interval for b: [**3.9792, 3.9909**]
  - Test statistics:
    - **X = 3.9795**
    - Critical values:
      - For one-sided test (H : b < 3.99): **c = 3.9214**
      - For one-sided test (H : b > 3.99): **c = 3.9894**
  - Decision: **Fail to reject H** in both one-sided and two-sided tests
  - Conclusion: The MLE is very close to the hypothesized b = 3.99, and all test results agree that we do not reject the null. The confidence interval also covers b = 3.99.

- **General Conclusion**

  - In all three cases, the null hypothesis parameter lies **inside** the corresponding confidence interval.
  - All hypothesis tests at significance level **= 0.03** led to **not rejecting H** .
  - There is full agreement between the confidence intervals and the two-sided hypothesis tests.