Q&A Code

Wei Chieh Chen

2025-03-03

Task:

We want to generate a random sample of size n from a Normal distribution with fix mean mu and fix standard deviation sigma. We then explore R's built-in cut() function (without using any additional packages) to categorize the data. Next, we compute and compare empirical (sample) quantiles and the corresponding theoretical (population) quantiles, plot them against each other in a scatterplot, and finally create a normal QQ plot using only base R functions to assess how closely the sampled data follow the specified Normal distribution.

Solution

Step 1: Set Parameters

We need to specify the sample size n as well as the mean μ and standard deviation σ of the Normal distribution. Code Explanation: n determines how many random values to draw. mu and sigma define the normal distribution we're sampling from. set.seed(…) ensures you'll get the same random sample each time the code runs (useful for debugging and comparing results).

```
set.seed(2023) # Sets random generator seed for consistent output

# ---------------------------------------------------------
# Set parameters and generate sample data
# ---------------------------------------------------------
n <-  sample(1:3000, 1)     # Number of observations
mu <- 5        # Fix Mean
sigma <- 2     # Fix Standard deviation
```

Step 2: Generate a Random Sample Data

Code Explanation: rnorm(n, mean, sd) draws n samples from a $N(\mu, \sigma)$ distribution. The result is a numeric vector stored in sample_data.

```
sample_data <- rnorm(n, mean = mu, sd = sigma)
```

Step 3: Explore the cut function

Code Explanation: cut(…) takes a numeric vector (here sample_data) and "cuts" it into intervals (bins). breaks = c(-Inf, …, Inf) ensures you capture very low and very high values. The output (category_factor) is a factor whose levels correspond to the intervals you defined.

```
# Define breakpoints: break the data into 7 categories: from mu +/- 3*sigma
# (-Inf, mu-3*sigma]    (mu-3*sigma, mu-2*sigma]     (mu-2*sigma, mu-sigma]
# (mu-sigma,mu]      (mu,mu+sigma]      (mu+sigma,mu+2*sigma]
# (mu+2*sigma,mu+3*sigma]  (mu+3*sigma, Inf]
```

```
break_points <- c(-Inf, mu - 3*sigma, mu - 2*sigma, mu - sigma, mu,
                  mu + sigma, mu + 2 *sigma, mu + 3*sigma, Inf)

categorical_factor  <- cut(sample_data, breaks = break_points)

# See how many data points fall into each category
print(table(categorical_factor))
```

```
## categorical_factor
## (-Inf,-1]    (-1,1]     (1,3]     (3,5]     (5,7]     (7,9]    (9,11] (11, Inf]
##         3        34       274       695       628       236        36         3
```

Step 4: Compute Empirical (Sample) Quantiles

SOLUTION 1: We define the empirical distribution function (CDF) for any t by:

$$\hat{F}(t) \;=\; \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{x_i \le t\}.$$

In practice, we sort the data in ascending order, $x_{(1)} \le x_{(2)} \le \cdots \le x_{(n)}$. The inverse CDF (or *empirical quantile*) at probability p is given by:

$$\hat{F}^{-1}(p) \;=\; \min\{x : \hat{F}(x) \ge p\},$$

which can be implemented simply by:

$$\hat{F}^{-1}(p) \;=\; x_{(\lceil p\,n \rceil)}.$$

To see how well our sample aligns with its assumed distribution, we compute:

$$\text{empirical quantiles: } \hat{F}^{-1}(p), \quad \text{theoretical quantiles: } Q_{\text{theory}}(p).$$

For a normal distribution $N(\mu, \sigma)$,

$$Q_{\text{theory}}(p) \;=\; \mu \;+\; \sigma\, z_p,$$

where z_p is the standard normal p-quantile.

SOLUTION 2: Reference URL: https://library.virginia.edu/data/articles/understanding-q-q-plots

Code Explanation: quantile(sample_data, probs = probs) calculates the quantiles of sample_data at the probability points.

```
# ---------------------------------------------------------
# Empirical CDF F_hat(t) and its inverse F_hat_inv(p)
# ---------------------------------------------------------
# Sort the data
sample_data_sorted <- sort(sample_data)

# Empirical CDF function: F_hat(t)
F_hat <- function(sample_data, t) {
  return(sum(sample_data <= t) / length(sample_data))
}

# Empirical CDF inverse:
#    Given p in [0,1], return the smallest x_i s.t. F_hat(x_i) >= p.
#    (One simple way is to choose x_sorted[ceiling(p * n)] if p>0;
#     for p=0, we can return the minimum, etc.)
F_hat_inv <- function(sample_data_sorted, p) {
```

```r
  n <- length(sample_data_sorted)
  if (p <= 0) return(sample_data_sorted[1])
  if (p >= 1) return(sample_data_sorted[n])
  idx <- ceiling(p * n)
  return(sample_data_sorted[idx])
}

# ----------------------------------------------
# Compare empirical vs. theoretical quantiles
# ----------------------------------------------
# Let's pick a set of probabilities:
probs <- seq(0, 1, 0.01)

# a) Empirical quantiles using F_hat_inv() "by hand"
sample_quantiles <- sapply(probs, function(p) F_hat_inv(sample_data_sorted, p))

# b) Theoretical quantiles from N(mu, sigma)
population_quantiles <- qnorm(probs, mean = mu, sd = sigma)

cat("\nEmpirical quantiles (by hand):\n")
```

```
##
## Empirical quantiles (by hand):
```

```r
print(sample_quantiles)
```

```
##   [1] -1.7607140  0.2571545  1.0517958  1.3599619  1.5217927  1.6919067
##   [7]  1.8577537  2.0328229  2.1524379  2.2630514  2.3824761  2.4597833
##  [13]  2.5664762  2.6950819  2.7867701  2.8630137  2.9678179  3.0384189
##  [19]  3.1067576  3.1783361  3.2761595  3.3454963  3.4116395  3.4654243
##  [25]  3.5328357  3.5774657  3.6278342  3.6880578  3.7509660  3.8027053
##  [31]  3.8496742  3.8946235  3.9327710  3.9927934  4.0577136  4.1220269
##  [37]  4.1742888  4.2393652  4.2879666  4.3327176  4.3902252  4.4403309
##  [43]  4.4777016  4.5321859  4.5915676  4.6406406  4.6916000  4.7400337
##  [49]  4.7809792  4.8305644  4.8742604  4.9079177  4.9715126  5.0109704
##  [55]  5.0590018  5.1245491  5.1611192  5.2107890  5.2631392  5.3156483
##  [61]  5.3807635  5.4559889  5.5273798  5.5577202  5.6082023  5.6682208
##  [67]  5.7150401  5.7775036  5.8210650  5.8718964  5.9345384  5.9984042
##  [73]  6.0814234  6.1568278  6.2121589  6.2769852  6.3254384  6.3777473
##  [79]  6.4301415  6.5044200  6.5522069  6.6630726  6.7441519  6.8008073
##  [85]  6.8638943  6.9474745  7.0355599  7.1256065  7.2206267  7.3111120
##  [91]  7.4210003  7.4909485  7.6089521  7.7342125  7.8502739  8.0114091
##  [97]  8.2691869  8.5292909  9.0075679  9.4494742 11.9290263
```

```r
cat("\nTheoretical quantiles:\n")
```

```
##
## Theoretical quantiles:
```

```r
print(population_quantiles)
```

```
##   [1]      -Inf 0.3473043 0.8925022 1.2384128 1.4986279 1.7102927 1.8904528
##   [8] 2.0484179 2.1898569 2.3184899 2.4368969 2.5469438 2.6500264 2.7472177
##  [15] 2.8393613 2.9271332 3.0110842 3.0916695 3.1692698 3.2442074 3.3167575
##  [22] 3.3871575 3.4556136 3.5223063 3.5873949 3.6510205 3.7133092 3.7743740
##  [29] 3.8343170 3.8932306 3.9511990 4.0082993 4.0646024 4.1201737 4.1750737
```

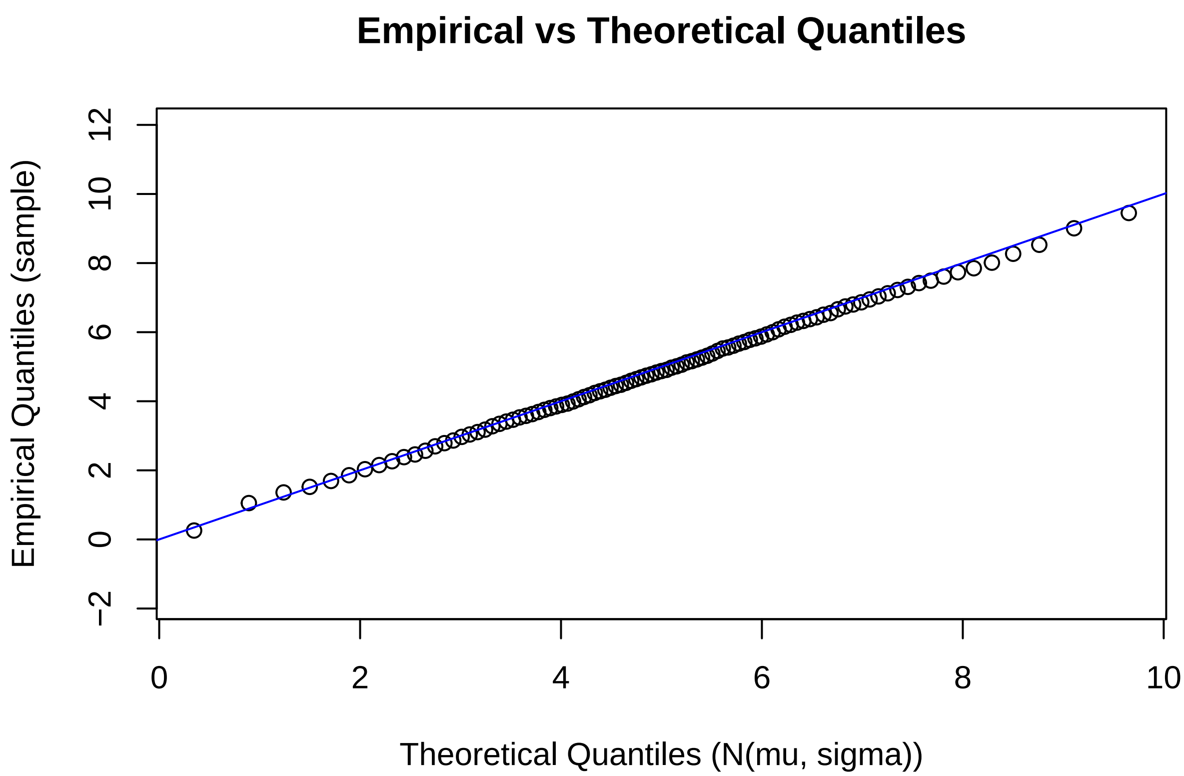
```
##  [36] 4.2293591 4.2830824 4.3362933 4.3890384 4.4413619 4.4933058 4.5449100
##  [43] 4.5962130 4.6472517 4.6980616 4.7486773 4.7991326 4.8494603 4.8996928
##  [50] 4.9498622 5.0000000 5.0501378 5.1003072 5.1505397 5.2008674 5.2513227
##  [57] 5.3019384 5.3527483 5.4037870 5.4550900 5.5066942 5.5586381 5.6109616
##  [64] 5.6637067 5.7169176 5.7706409 5.8249263 5.8798263 5.9353976 5.9917007
##  [71] 6.0488010 6.1067694 6.1656830 6.2256260 6.2866908 6.3489795 6.4126051
##  [78] 6.4776937 6.5443864 6.6128425 6.6832425 6.7557926 6.8307302 6.9083305
##  [85] 6.9889158 7.0728668 7.1606387 7.2527823 7.3499736 7.4530562 7.5631031
##  [92] 7.6815101 7.8101431 7.9515821 8.1095472 8.2897073 8.5013721 8.7615872
##  [99] 9.1074978 9.6526957       Inf
```

```r
################ Sol 2 ################################
#--------------------------------------------------------
# Compute sample quantiles
#--------------------------------------------------------
# Generates the quantiles for a normal distribution from 0 to 1 by increments of 0.01
# probs <- seq(0, 1, 0.01)
# sample_quantiles <- quantile(sample_data, probs = probs)
#--------------------------------------------------------
# Compute theoretical (population) quantiles
#--------------------------------------------------------
# For a Normal(mu, sigma), the quantiles are qnorm(probs, mean = mu, sd = sigma)
# population_quantiles <- qnorm(probs, mean = mu, sd = sigma)
```

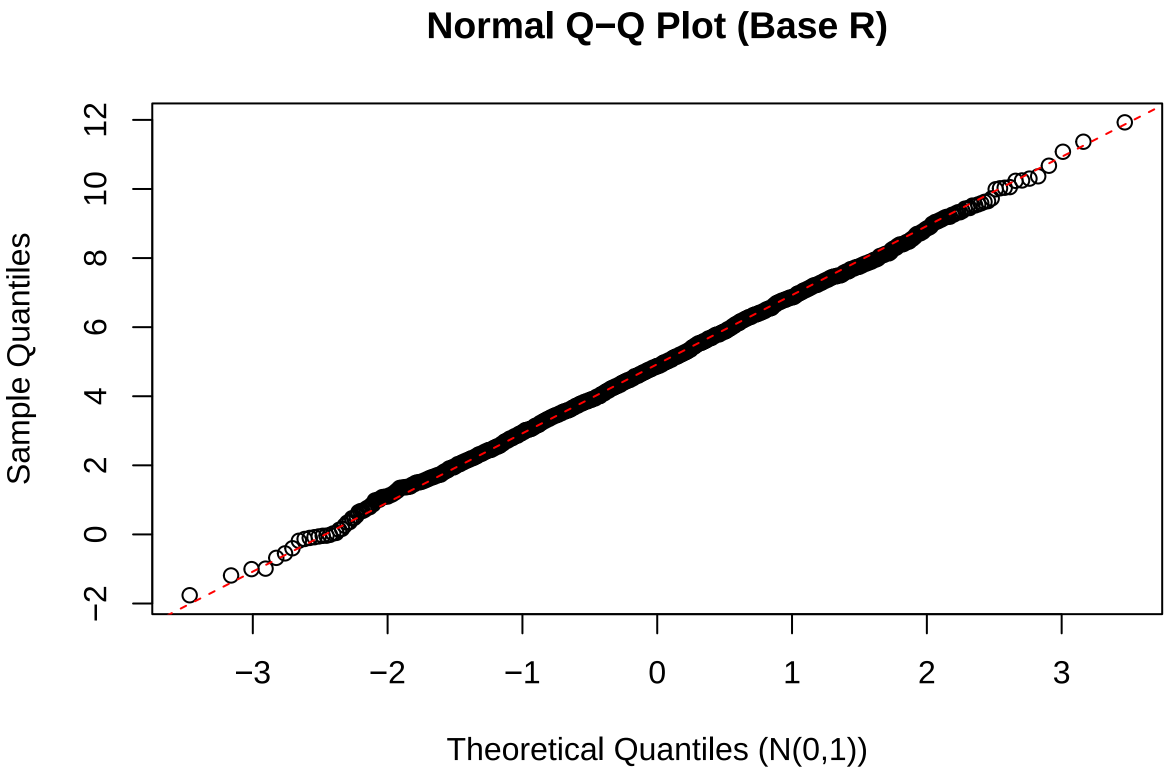
Scatterplot & QQ Plot

Code Explanation: Using base R's qqnorm() compares the sample quantiles of the data to the theoretical quantiles of $N(0,1)$. If the data truly come from a Normal distribution (potentially with shift/scale), the points should lie near the straight line qqline().

```r
#--------------------------------------------------------
# Create a scatterplot of sample vs. theoretical quantiles
#--------------------------------------------------------
plot(population_quantiles, sample_quantiles,
     main = "Empirical vs Theoretical Quantiles",
     xlab = "Theoretical Quantiles (N(mu, sigma))",
     ylab = "Empirical Quantiles (sample)")
abline(0, 1, col = "blue")  # Add reference line
```




```r
#------------------------------------------------------
# Create a normal QQ plot (base R)
#------------------------------------------------------
# Using qqnorm (centers and scales for standard normal)
qqnorm(sample_data,
       main = "Normal Q-Q Plot (Base R)",
       xlab = "Theoretical Quantiles (N(0,1))",
       ylab = "Sample Quantiles")
qqline(sample_data, col = "red", lty = 2)
```


Normal Q−Q Plot (Base R)
Sample Quantiles
Theoretical Quantiles (N(0,1))
−2
0
2
4
6
8
10
12
−3
−2
−1
0
1
2
3