Midterm Project

Chenxuan Xiong

2024-03-26

## Order Statistics

To derive the order statistics, we follow a general procedure:

1. Find the probability density function for the distribution .
2. Calculate the cumulative density function for the distribution .
3. For statistics , there must be samples less than , a sample equals to , and samples larger than .

* The probability samples less than :
* The probability a sample equals to :
* The probability samples larger than :
* Select sample out of to be less than is

1. Multiply all the terms we get the a gereral probability density function of

### 1. Uniform Distirbution

Probability density function: for

Cumulative density function:

Probability density function of :

Min:

Max:

Median:

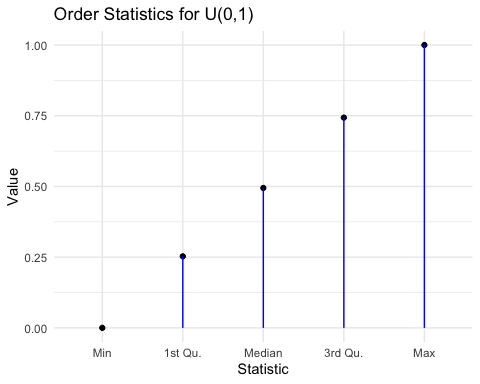
25 quantile:

75 quantile:

#### 

#### Simulation with U(0,1)

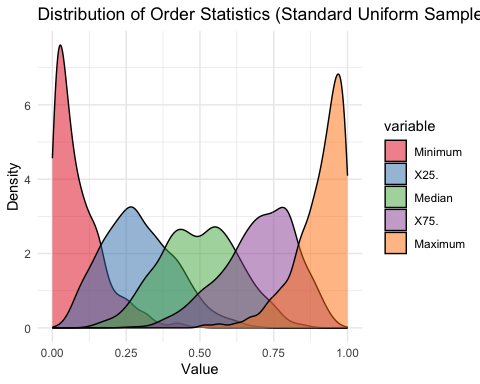
library(ggplot2)  
  
set.seed(123)  
  
sample\_size <- 10000  
  
uniform\_sample <- runif(sample\_size)  
  
order\_stats <- quantile(uniform\_sample, probs = c(0, 0.25, 0.5, 0.75, 1))  
  
  
plot\_data <- data.frame(  
 Statistic = c('Min', '1st Qu.', 'Median', '3rd Qu.', 'Max'),  
 Value = c(order\_stats[1], order\_stats[2], order\_stats[3], order\_stats[4], order\_stats[5])  
)  
  
ggplot(plot\_data, aes(x = reorder(Statistic, Value), y = Value)) +  
 geom\_point() +  
 geom\_segment(aes(xend = Statistic, yend = 0), color = 'blue') +  
 theme\_minimal() +  
 labs(title = "Order Statistics for U(0,1)", x = "Statistic", y = "Value")



library(ggplot2)  
library(reshape2)  
  
n <- 10  
iterations <- 1000  
  
min\_values <- numeric(iterations)  
q25\_values <- numeric(iterations)  
median\_values <- numeric(iterations)  
q75\_values <- numeric(iterations)  
max\_values <- numeric(iterations)  
  
set.seed(123)  
for (i in 1:iterations) {  
 sample <- runif(n, min = 0, max = 1)  
 min\_values[i] <- min(sample)  
 q25\_values[i] <- quantile(sample, 0.25)  
 median\_values[i] <- median(sample)  
 q75\_values[i] <- quantile(sample, 0.75)  
 max\_values[i] <- max(sample)  
}  
  
df <- data.frame(Minimum = min\_values, `25%` = q25\_values, Median = median\_values,   
 `75%` = q75\_values, Maximum = max\_values)  
  
df\_melted <- melt(df)

## No id variables; using all as measure variables

ggplot(df\_melted, aes(x = value, fill = variable)) +  
 geom\_density(alpha = 0.5) +  
 labs(title = "Distribution of Order Statistics (Standard Uniform Samples)",  
 x = "Value", y = "Density") +  
 theme\_minimal() +  
 scale\_fill\_brewer(palette = "Set1")



### 

### 2. Exponential Distribution

Probability density function: for

Cumulative density function:

Probability density function of :

Min:

Max:

Median:

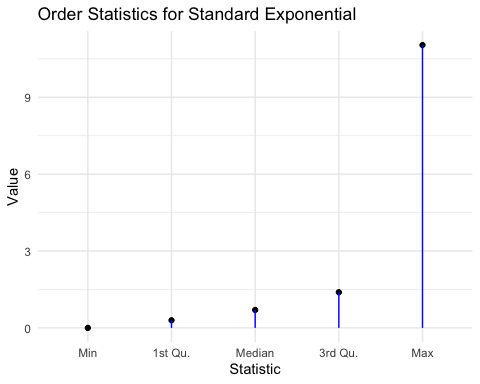
25 quantile:

75 quantile:

#### 

#### Simulation with standard exponential distribution

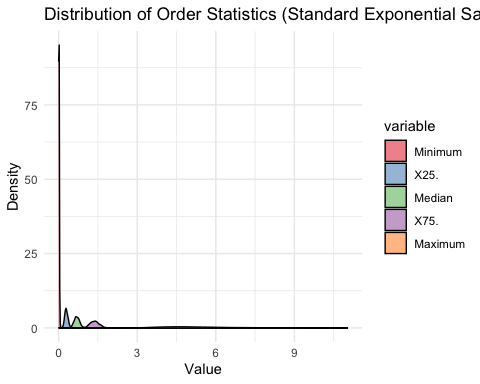
n <- 10000  
  
sample\_data <- rexp(n, rate = 1)   
min\_stat <- min(sample\_data)  
quantile\_25 <- quantile(sample\_data, probs = 0.25)  
median\_stat <- median(sample\_data)  
quantile\_75 <- quantile(sample\_data, probs = 0.75)  
max\_stat <- max(sample\_data)  
  
order\_stats <- c(min\_stat, quantile\_25, median\_stat, quantile\_75, max\_stat)  
  
plot\_data <- data.frame(  
 Statistic = c('Min', '1st Qu.', 'Median', '3rd Qu.', 'Max'),  
 Value = c(order\_stats[1], order\_stats[2], median\_stat, order\_stats[4], order\_stats[5])  
)  
ggplot(plot\_data, aes(x = reorder(Statistic, Value), y = Value)) +  
 geom\_point() +  
 geom\_segment(aes(xend = Statistic, yend = 0), color = 'blue') +  
 theme\_minimal() +  
 labs(title = "Order Statistics for Standard Exponential", x = "Statistic", y = "Value")



n <- 100  
iterations <- 1000  
  
min\_values <- numeric(iterations)  
q25\_values <- numeric(iterations)  
median\_values <- numeric(iterations)  
q75\_values <- numeric(iterations)  
max\_values <- numeric(iterations)  
  
set.seed(123)  
for (i in 1:iterations) {  
 sample <- rexp(n, rate = 1)  
 min\_values[i] <- min(sample)  
 q25\_values[i] <- quantile(sample, 0.25)  
 median\_values[i] <- median(sample)  
 q75\_values[i] <- quantile(sample, 0.75)  
 max\_values[i] <- max(sample)  
}  
  
df <- data.frame(Minimum = min\_values, `25%` = q25\_values, Median = median\_values,   
 `75%` = q75\_values, Maximum = max\_values)  
  
df\_melted <- melt(df)

## No id variables; using all as measure variables

ggplot(df\_melted, aes(x = value, fill = variable)) +  
 geom\_density(alpha = 0.5) +  
 labs(title = "Distribution of Order Statistics (Standard Exponential Samples)",  
 x = "Value", y = "Density") +  
 theme\_minimal() +  
 scale\_fill\_brewer(palette = "Set1")



### 

### 3. Normal Distribution

Probability density function:

Cumulative density function: - For standard normal distribution:

Cumulative density function for general normal distribution:

Probability density function of :

Min:

Max:

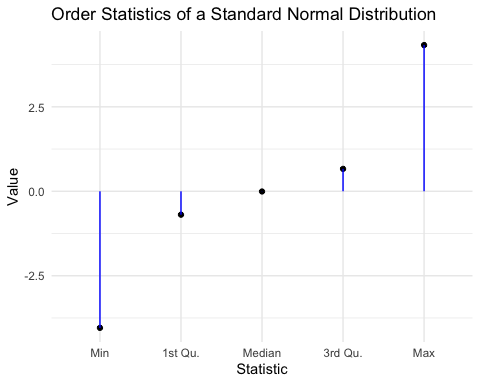
Median:

25 quantile:

75 quantile:

#### Simulation with standard normal distribution

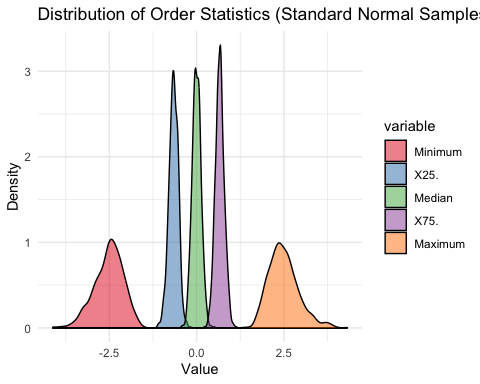
library(ggplot2)  
  
set.seed(42)  
  
sample\_size <- 10000  
simulated\_data <- rnorm(sample\_size, mean = 0, sd = 1)  
  
min\_val <- min(simulated\_data)  
first\_quartile <- quantile(simulated\_data, probs = 0.25)  
median\_val <- median(simulated\_data)  
third\_quartile <- quantile(simulated\_data, probs = 0.75)  
max\_val <- max(simulated\_data)  
  
order\_stats <- data.frame(  
 Statistic = c('Min', '1st Qu.', 'Median', '3rd Qu.', 'Max'),  
 Value = c(min\_val, first\_quartile, median\_val, third\_quartile, max\_val)  
)  
  
ggplot(order\_stats, aes(x = reorder(Statistic, Value), y = Value)) +  
 geom\_point() +  
 geom\_segment(aes(xend = Statistic, yend = 0), color = 'blue') +  
 theme\_minimal() +  
 labs(title = "Order Statistics of a Standard Normal Distribution",  
 x = "Statistic",  
 y = "Value")



n <- 100   
iterations <- 1000   
  
min\_values <- numeric(iterations)  
q25\_values <- numeric(iterations)  
median\_values <- numeric(iterations)  
q75\_values <- numeric(iterations)  
max\_values <- numeric(iterations)  
  
set.seed(123)   
for (i in 1:iterations) {  
 sample <- rnorm(n) # Generate a sample  
 min\_values[i] <- min(sample) # Minimum  
 q25\_values[i] <- quantile(sample, 0.25) # 25% Quantile  
 median\_values[i] <- median(sample) # Median  
 q75\_values[i] <- quantile(sample, 0.75) # 75% Quantile  
 max\_values[i] <- max(sample) # Maximum  
}  
  
df <- data.frame(Minimum = min\_values, `25%` = q25\_values, Median = median\_values,   
 `75%` = q75\_values, Maximum = max\_values)  
  
library(reshape2)  
df\_melted <- melt(df)

## No id variables; using all as measure variables

ggplot(df\_melted, aes(x = value, fill = variable)) +  
 geom\_density(alpha = 0.5) +  
 labs(title = "Distribution of Order Statistics (Standard Normal Samples)",  
 x = "Value",  
 y = "Density") +  
 theme\_minimal() +  
 scale\_fill\_brewer(palette = "Set1")



## 

## Markov Chain in Weather Forecasting

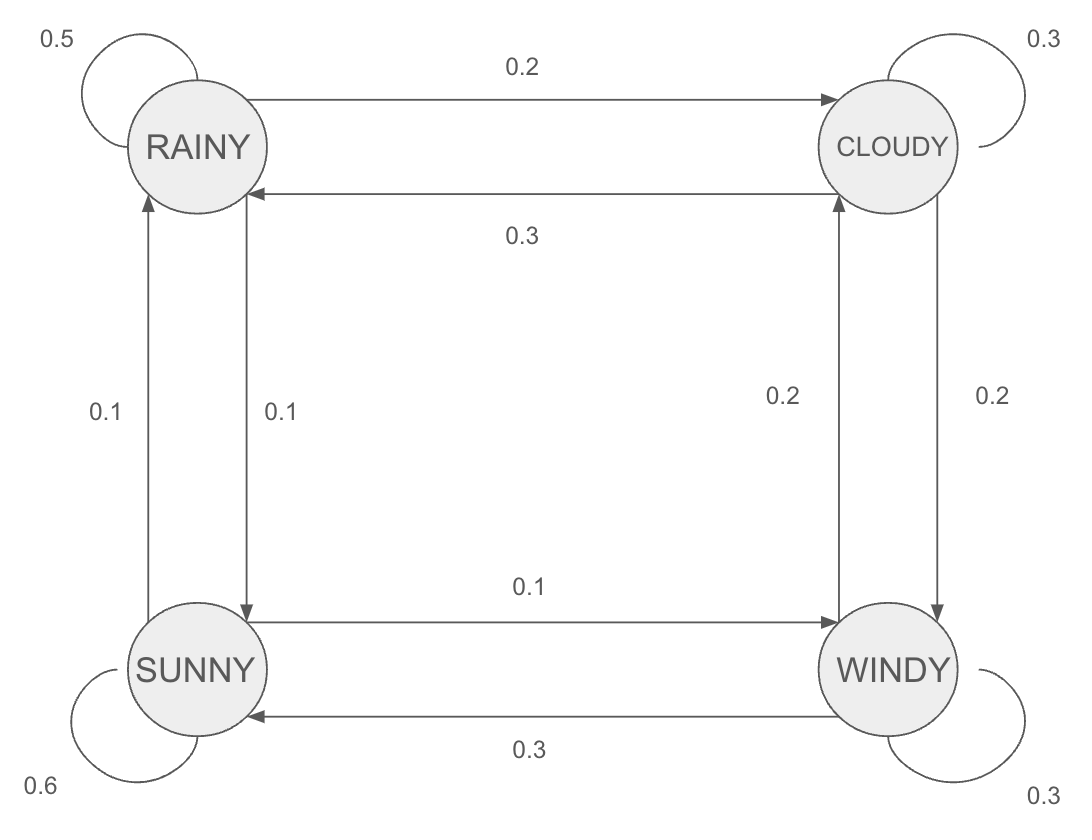
We know intuitively that if today is rainy then there is a good chance tomorrow will be rainy too. We might also expect it to be fairly likely to be cloudy, with only a small chance of being sunny. We can make similar judgements about what tomorrow’s weather will be like if it is cloudy or sunny today.

It can be present in a transition matrix:

| WEATHER TODAY | P(RAINY) | P(CLOUDY) | P(SUNNY) | P(WINDY) |
| --- | --- | --- | --- | --- |
| RAINY | 0.5 | 0.2 | 0.1 | 0.2 |
| CLOUDY | 0.3 | 0.3 | 0.2 | 0.2 |
| SUNNY | 0.1 | 0.2 | 0.6 | 0.1 |
| WINDY | 0.3 | 0.2 | 0.3 | 0.2 |

For example if today is sunny then there is a 10% chance of it being rainy tomorrow, 20% chance of it being cloudy, 60% chance of it being sunny again, and 10% chance of it being windy.

Graph representation:

 R

Simulation:

Assume the first day is a sunny day, predict weather of the following 14 days.

states <- c("RAINY", "CLOUDY", "SUNNY", "WINDY")  
transition\_matrix <- matrix(c(  
 0.5, 0.2, 0.1, 0.2,  
 0.3, 0.3, 0.2, 0.2,  
 0.1, 0.2, 0.6, 0.1,  
 0.3, 0.2, 0.3, 0.2  
), nrow = 4, byrow = TRUE)  
  
colnames(transition\_matrix) <- states  
rownames(transition\_matrix) <-states  
  
  
forecast\_next\_day <- function(current\_state, transition\_matrix) {  
  
 state\_index <- match(current\_state, rownames(transition\_matrix))  
   
 sample(rownames(transition\_matrix), size = 1, prob = transition\_matrix[state\_index, ])  
}  
  
current\_weather <- "SUNNY"  
  
# Forecast for the next 14 days  
forecast <- vector("character", length = 14)  
for (i in 1:14) {  
 next\_day\_weather <- forecast\_next\_day(current\_weather, transition\_matrix)  
 forecast[i] <- next\_day\_weather  
 current\_weather <- next\_day\_weather  
}  
  
forecast

## [1] "CLOUDY" "CLOUDY" "WINDY" "CLOUDY" "CLOUDY" "WINDY" "SUNNY" "SUNNY"   
## [9] "CLOUDY" "RAINY" "SUNNY" "SUNNY" "SUNNY" "SUNNY"

## 

## Irrigation Problem

To calculate the speed of the outer wheel, we use the formula:

The radius (r) is 1320 feet, and time is recorded in the .txt file.

data = read.table("rotation\_time.txt")  
time\_data <- unlist(data)  
speed\_data = 2\*pi\*1320/(time\_data)  
mean\_value <- mean(speed\_data)  
se <- sd(speed\_data) / sqrt(length(speed\_data))  
  
# Calculate the 90% confidence interval  
alpha <- 1 - 0.90  
z <- qt(1 - alpha / 2, df = length(speed\_data) - 1)  
lower <- mean\_value - z \* se  
upper <- mean\_value + z \* se  
  
ci <- c(lower, upper)  
print(ci)

## [1] 358.6738 369.7162

The 90% confidence interval for the speed is [358.6738, 369.7162].

Based on the data we’ve collected and our calculations, we are 90% certain that the true average speed falls between 358.7 and 369.7 feet per hour. This means that if we repeated this study many times, about 90 out of 100 times, the interval we calculated would contain the true average speed.

It’s a statistical way of saying we’re pretty sure, but not absolutely certain, about where the true value lies based on our sample. In more technical terms, the interval is calculated using the sample mean and the standard error, multiplied by the z-score that corresponds to the 90% confidence level from the student-t distribution. If we were to sample repeatedly from the same population and construct intervals in the same way, we’d expect 90% of those intervals to contain the population mean.