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 $f(x)$.
2. Calculate the cumulative density function for the distribution $F(x)$.
3. For k^{th} statistics $X_{(k)}$, there must be $k-1$ samples less than $X_{(k)}$, a sample equals to $X_{(k)}$, and $n-k$ samples larger than $X_{(k)}$.

- The probability $k-1$ samples less than $X_{(k)}$: $F(x)^{k-1}$
- The probability a sample equals to $X_{(k)}$: $f(x)$
- The probability $n-k$ samples larger than $X_{(k)}$: $[1-F(x)]^{n-k}$
- Select $k-1$ sample out of n to be less than x is $\binom{n}{k-1}$

4. Multiply all the terms we get the a gereral probability density function of $X_{(k)} = \binom{n}{k-1}F(x)^{k-1}f(x)[1-F(x)]^{n-k}$

1. Uniform Distirbution

Probability density function: $f(x) = \frac{1}{b-a}$ for $a \leq x \leq b$

Cumulative density function: $F(x) = \int_a^x f\,(t)\,dt = \int_a^x \frac{1}{b-a}\,dt = \frac{x-a}{b-a}$

Probability density function of $X_{(k)}$: $f\left(X_{(k)}\right) = \frac{n!}{(k-1)!(n-k)!}\left(\frac{x-a}{b-a}\right)^{k-1}\left(1-\frac{x-a}{b-a}\right)^{n-k}\frac{1}{b-a}$

Min: $f\left(X_{(1)}\right) = \frac{n}{b-a}\left(\frac{b-x}{b-a}\right)^{n-1}$

Max: $f\left(X_{(n)}\right) = \frac{n}{b-a}\left(\frac{x-a}{b-a}\right)^{n-1}$

Median: $f\left(X_{(n/2)}\right) = \frac{n!}{(n/2-1)!(n/2)!}\left(\frac{x-a}{b-a}\right)^{n/2-1}\left(1-\frac{x-a}{b-a}\right)^{n/2}\frac{1}{b-a}$

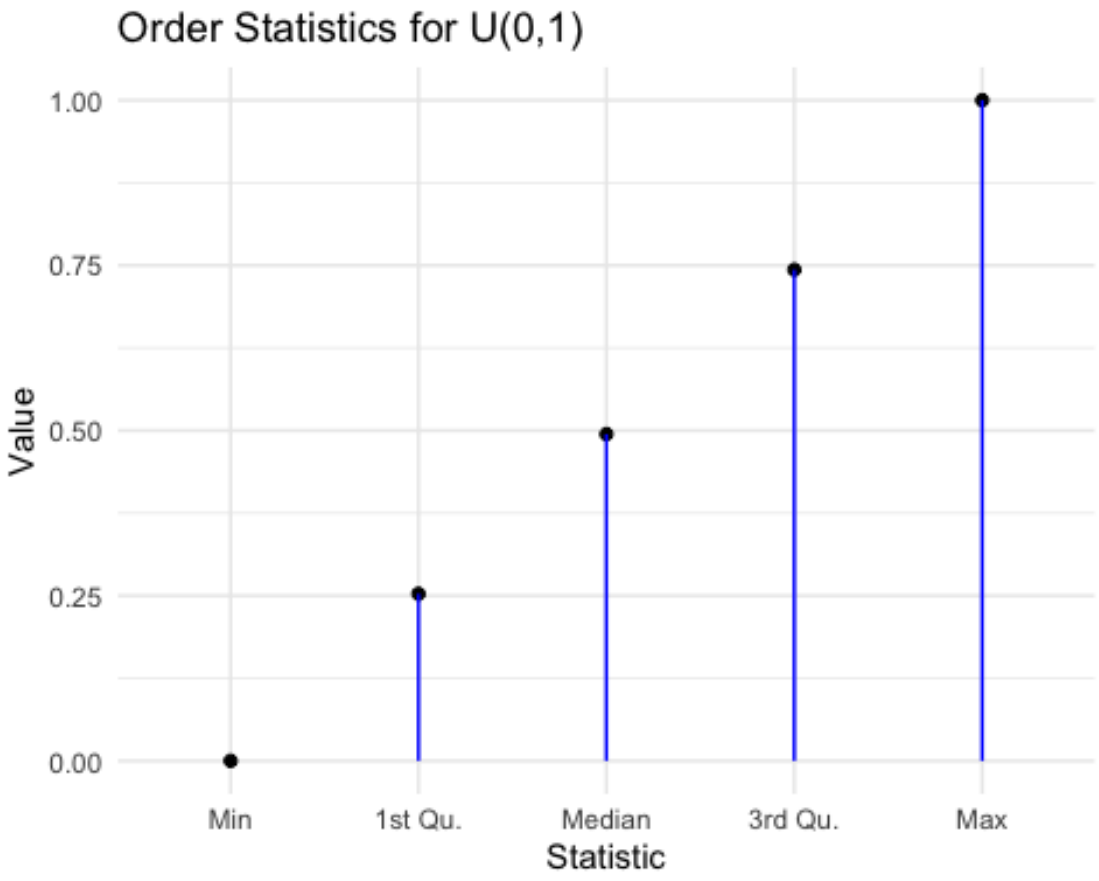
25 quantile: $f\left(X_{(n/4)}\right) = \frac{n!}{(n/4-1)!(3n/4)!}\left(\frac{x-a}{b-a}\right)^{n/4-1}\left(1-\frac{x-a}{b-a}\right)^{3n/4}\frac{1}{b-a}$

75 quantile: $f\left(X_{(3n/4)}\right) = \frac{n!}{(3n/4-1)!(n/4)!}\left(\frac{x-a}{b-a}\right)^{3n/4-1}\left(1-\frac{x-a}{b-a}\right)^{n/4}\frac{1}{b-a}$

Simulation with U(0,1)

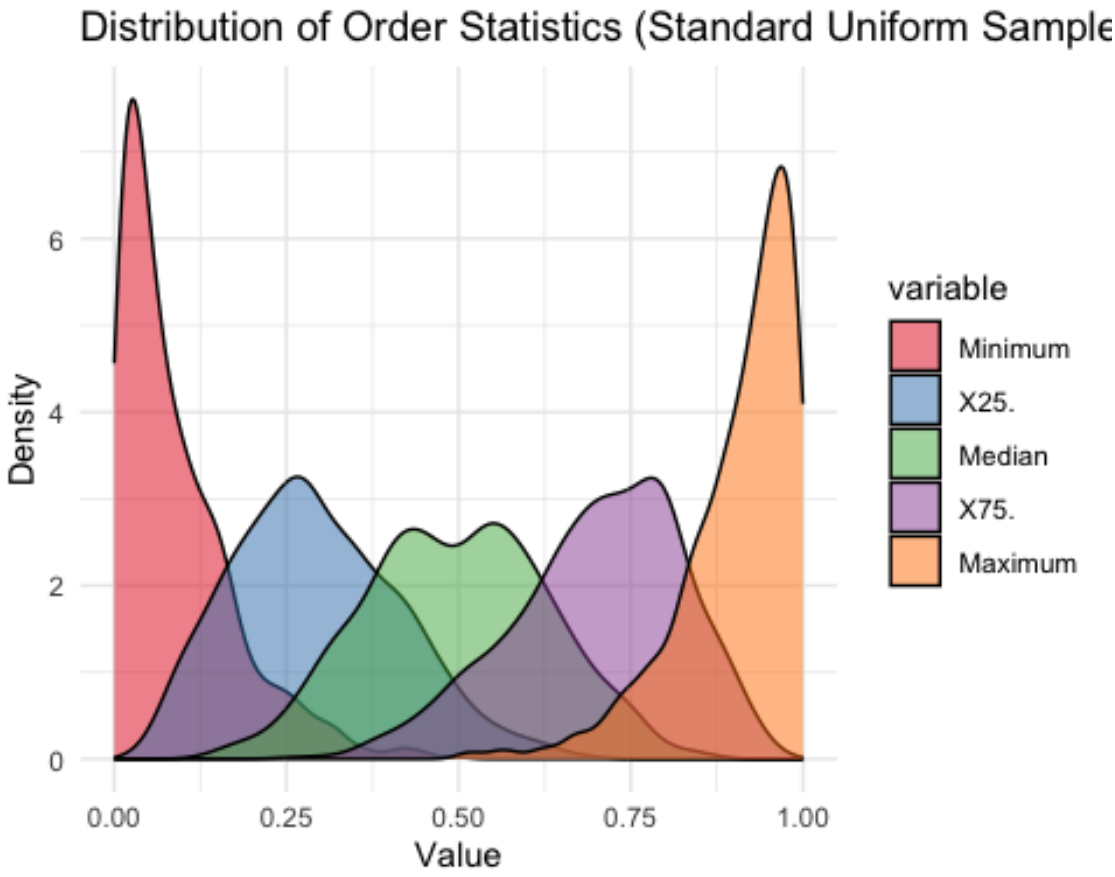
```
library(ggplot2)

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)
```

```r
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: $f(x) = \lambda e^{-\lambda x}$ for $x \geq 0$

Cumulative density function: $F(x) = \int_0^x \lambda\, e^{-\lambda t}\, dt = 1 - e^{\lambda x}$

Probability density function of $X_{(k)}$: $f\left(X_{(k)}\right) = \frac{n!}{(k-1)!(n-k)!}\left(1 - e^{\lambda x}\right)^{k-1} \lambda e^{-\lambda x}\left(e^{\lambda x}\right)^{n-k} = \frac{n!}{(k-1)!(n-k)!} e^{-\lambda n x} x^{k-1}$

Min: $f\left(X_{(1)}\right) = ne^{-\lambda n x}$

Max: $f(X_{(n)}) = ne^{-\lambda nx}x^{n-1}$

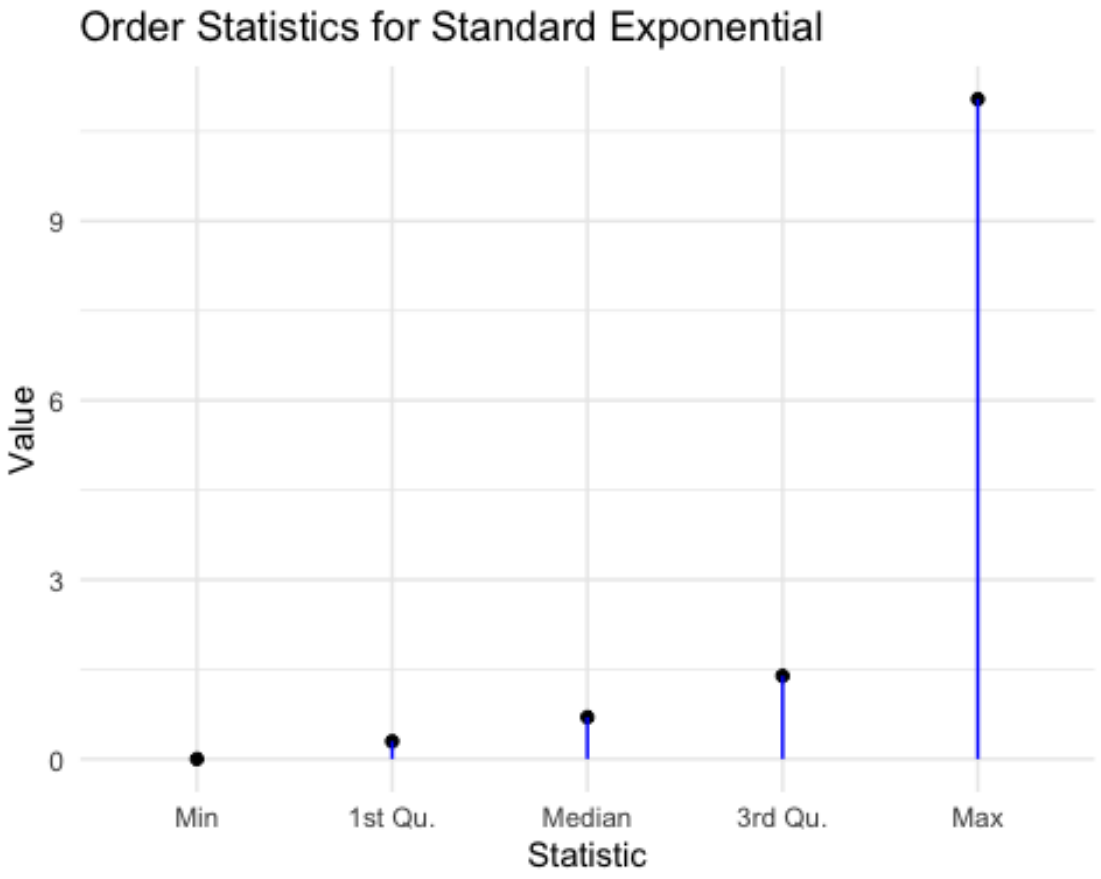
Median: $f(X_{(n/2)}) = \frac{n!}{(n/2-1)!(n/2)!}e^{-\lambda nx}x^{n/2-1}$

25 quantile: $f(X_{(n/4)}) = \frac{n!}{(n/4-1)!(3n/4)!}e^{-\lambda nx}x^{n/4-1}$

75 quantile: $f(X_{(3n/4)}) = \frac{n!}{(3n/4-1)!(n/4)!}e^{-\lambda nx}x^{3n/4-1}$

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")
```




```r
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: $f(x|\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

Cumulative density function: - For standard normal distribution: $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt$

Cumulative density function for general normal distribution: $F(x) = \Phi\left(\frac{x-\mu}{\sigma}\right)$

Probability density function of $X_{(k)}$: $f\left(X_{(k)}\right) = \frac{n!}{(k-1)!(n-k)!}\left(\Phi\left(\frac{x-\mu}{\sigma}\right)\right)^{k-1} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \left(1 - \Phi\left(\frac{x-\mu}{\sigma}\right)\right)^{n-k}$

Min: $f\left(X_{(1)}\right) = n\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \left(1 - \Phi\left(\frac{x-\mu}{\sigma}\right)\right)^{n-1}$

Max: $f\left(X_{(n)}\right) = n\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \left(\Phi\left(\frac{x-\mu}{\sigma}\right)\right)^{n-1}$

Median: $f\left(X_{(n/2)}\right) = \frac{n!}{(n/2-1)!(n/2)!}\left(\Phi\left(\frac{x-\mu}{\sigma}\right)\right)^{n/2-1} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \left(1 - \Phi\left(\frac{x-\mu}{\sigma}\right)\right)^{n/2}$

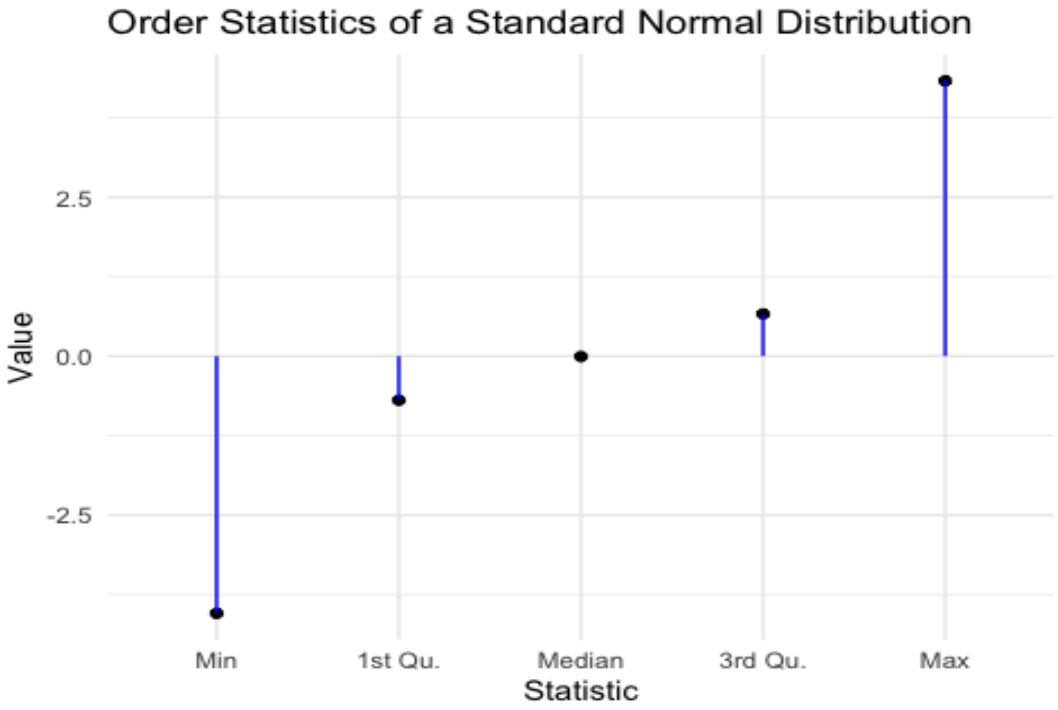
25 quantile: $f\left(X_{(n/4)}\right) = \frac{n!}{(n/4-1)!(3n/4)!}\left(\Phi\left(\frac{x-\mu}{\sigma}\right)\right)^{n/4-1} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \left(1 - \Phi\left(\frac{x-\mu}{\sigma}\right)\right)^{3n/4}$

75 quantile: $f\left(X_{(3n/4)}\right) = \frac{n!}{(3n/4-1)!(n/4)!}\left(\Phi\left(\frac{x-\mu}{\sigma}\right)\right)^{3n/4-1} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \left(1 - \Phi\left(\frac{x-\mu}{\sigma}\right)\right)^{n/4}$

Simulation with standard normal distribution

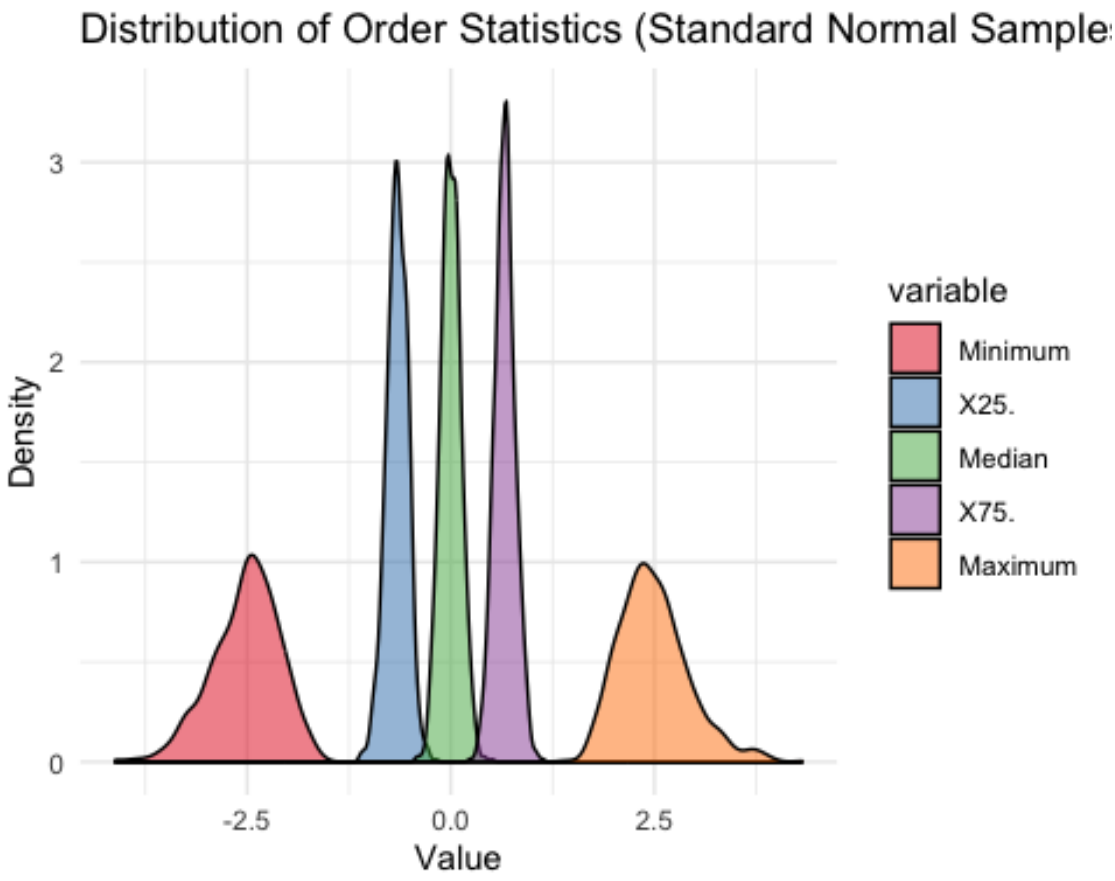
```r
library(ggplot2)

set.seed(42)

sample_size <- 10000
simulated_data <- rnorm(sample_size, mean = 0, sd = 1)
```

```r
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")
```



```r
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

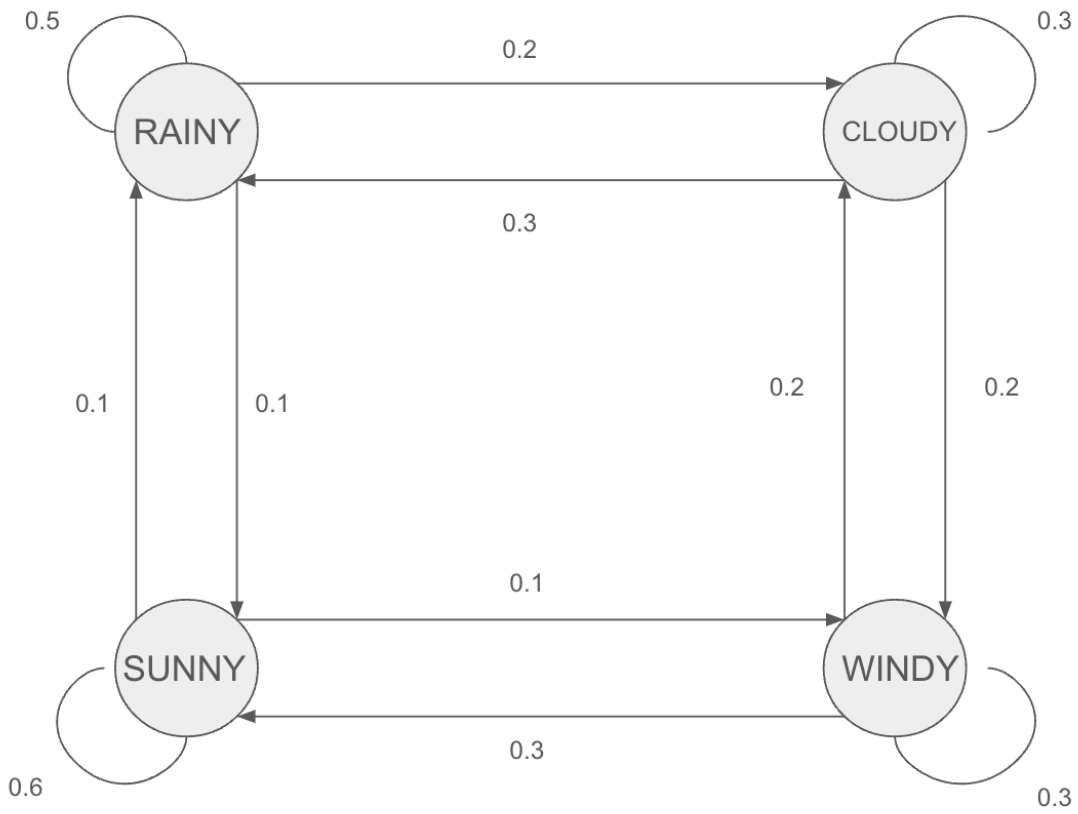
We know intuitively that if today is rainy then there is a good chance tomorrow will be rainy too. We might also expect it to be fairly likely to be cloudy, with only a small chance of being sunny. We can make similar judgements about what tomorrow's weather will be like if it is cloudy or sunny today.

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.

```r
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:

$$v = \frac{2\pi r}{t}$$

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.