Statistics with R

Yi Wang

2023-09-03

Contents

4 Probability

4.1 Basic concepts of probability

In this code:

- We calculate the probability of drawing a Heart from a sample space (a deck of cards).
- We simulate random events such as a coin toss and rolling a six-sided die.
- We simulate multiple die rolls and visualize the resulting probability distribution.
- We calculate the probability of a specific outcome (rolling a 3).

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

# Define a sample space (e.g., a deck of cards)
sample_space <- c("Hearts", "Diamonds", "Clubs", "Spades")

# Calculate the probability of drawing a Heart from the sample space
probability_heart <- sum(sample_space == "Hearts") / length(sample_space)

cat("Probability of drawing a Heart:", probability_heart, "\n")
```

```
## Probability of drawing a Heart: 0.25
```

```r
# Simulate a random event (e.g., coin toss)
coin_toss <- sample(c("Heads", "Tails"), size = 1)
```

```r
cat("Result of a random coin toss:", coin_toss, "\n")
```

```
## Result of a random coin toss: Heads
```

```r
# Simulate rolling a six-sided die
die_roll <- sample(1:6, size = 1)

cat("Result of rolling a die:", die_roll, "\n")
```

```
## Result of rolling a die: 5
```

```r
# Simulate multiple die rolls and visualize the probability distribution
num_rolls <- 1000
die_rolls <- sample(1:6, size = num_rolls, replace = TRUE)

# Calculate the relative frequencies for each outcome
relative_frequencies <- table(die_rolls) / num_rolls
relative_frequencies
```

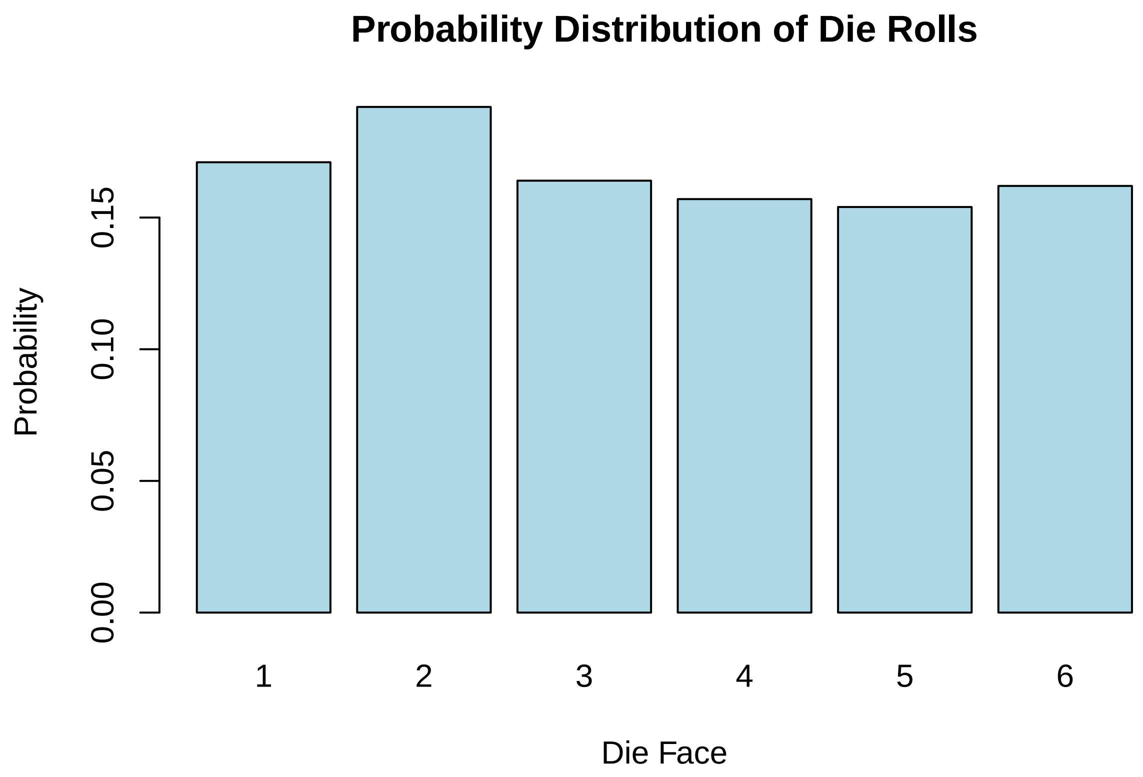
```
## die_rolls
##     1     2     3     4     5     6
## 0.171 0.192 0.164 0.157 0.154 0.162
```

```r
# Calculate the probability of rolling a 3
probability_roll_3 <- relative_frequencies[3]

cat("Probability of rolling a 3:", probability_roll_3, "\n")
```

```
## Probability of rolling a 3: 0.164
```

```r
# Visualize the probability distribution with a bar plot
barplot(relative_frequencies, main = "Probability Distribution of Die Rolls",
        xlab = "Die Face", ylab = "Probability", col = "lightblue")
```



4.2 Addition rule and multiplication rule

4.3 Complements, conditional probability, and Bayes' theorem

4.4 Counting

4.4.1 Calculate factorial $n!$

R provides a built-in function to calculate factorial. You can use the `factorial()` function in R to compute the factorial of a number. Here's how you can use it:

```
n <- 5
factorial_result <- factorial(n)
cat("Factorial of", n, "is", factorial_result, "\n")
```

```
## Factorial of 5 is 120
```

Replace the value of `n` with the number for which you want to calculate the factorial, and the `factorial()` function will return the result.

4.4.2 Find all permutaitons and the number of all permutaions

To do this, we can use the `permutations` function from the **gtools** package. For any list of size `n`, this function computes all the different combinations we can get when we select `r` items. Here are all the ways we can choose two numbers from a list consisting of `1,2,3`:

```
library(gtools)
permutations(3, 2)
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    1    3
## [3,]    2    1
## [4,]    2    3
## [5,]    3    1
## [6,]    3    2
```

Notice that the order matters here: 3,1 is different than 1,3. Also, note that (1,1), (2,2), and (3,3) do not appear because once we pick a number, it can't appear again.

To get the actual number of permutations, one can use the R-function `nrow()` to find the total number of rows in the output of `permutations`:

```
library(gtools)
nrow(permutations(3,2))
```

```
## [1] 6
```

Alternatively, we can add a vector `v` to indicate the objects that a permutation is performed on. If you want to see five random seven digit phone numbers out of all possible phone numbers (without repeats), you can type:

```
all_phone_numbers <- permutations(10, 7, v = 0:9) # Use digits 0, 1, ..., 9
n <- nrow(all_phone_numbers)
cat("total number of phone numbers n = ", n, "\n")
```

```
## total number of phone numbers n =  604800
```

```
# Randomly sample 5 phone numbers
print("Randomly sample 5 phone numbers:")
```

```
## [1] "Randomly sample 5 phone numbers:"
```

```
index <- sample(n, 5)
all_phone_numbers[index,]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]    8    9    5    1    6    0    3
## [2,]    4    0    2    3    5    7    8
## [3,]    0    4    6    1    3    2    7
## [4,]    5    8    2    6    4    0    3
## [5,]    7    5    1    0    9    2    6
```

Instead of using the numbers 1 through 10, the default, it uses what we provided through `v`: the digits 0 through 9.

4.4.3 Find all combinations and the number of all combinations

How about if the order doesn't matter? For example, in Blackjack if you get an Ace and a face card in the first draw, it is called a *Natural 21* and you win automatically. If we wanted to compute the probability of this happening, we would enumerate the *combinations*, not the permutations, since the order does not matter.

```
combinations(3,2)
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    1    3
## [3,]    2    3
```

In the second line, the outcome does not include (2,1) because (1,2) already was enumerated. The same applies to (3,1) and (3,2).

To get the actual number of combinations, one can do

```r
nrow(combinations(3,2))
```

```
## [1] 3
```

(**optional**) Of course, one can define a R-function to calculate a permutation number.

```r
# Function to calculate permutation (nPr)
nPr <- function(n, r) {
  if (n < r) {
    return(0)
  } else {
    return(factorial(n) / factorial(n - r))
  }
}
nPr(3,2)
```

```
## [1] 6
```

```r
# Function to calculate combination (nCr)
nCr <- function(n, r) {
  if (n < r) {
    return(0)
  } else {
    return(factorial(n) / (factorial(r) * factorial(n - r)))
  }
}
nCr(3,2)
```

```
## [1] 3
```

5 Discrete probability distribution

5.1 Calculate mean, standard deviation and variance with equal probability

You can use R to calculate the mean, standard deviation, and variance of a given data set using built-in functions like `mean()`, `sd()`, and `var()`. Here's some sample R code to do that:

```r
# Sample data set
data_set <- c(12, 15, 18, 21, 24, 27, 30, 33, 36, 39)

# Calculate the mean
mean_value <- mean(data_set)
cat("Mean:", mean_value, "\n")
```

```
## Mean: 25.5
```

```r
# Calculate the standard deviation
std_deviation <- sd(data_set)
cat("Standard Deviation:", std_deviation, "\n")
```

```
## Standard Deviation: 9.082951
```

```r
# Calculate the variance
variance <- var(data_set)
```

```r
cat("Variance:", variance, "\n")
```

```
## Variance: 82.5
```

Just replace the data_set vector with your actual data, and this code will compute and print the mean, standard deviation, and variance for your data set. Note the results calculated by `mean()`, `sd()` and `var()` assumes each data points occurs with the equal probability $1/n$, where n is the number of data points.

5.2 Expectation and standard deviation with a given probability distribution

By definition,

```r
# Define the possible values and their corresponding probabilities
values <- c(1, 2, 3, 4, 5)
probabilities <- c(0.1, 0.2, 0.3, 0.2, 0.2)

# Calculate the mean (expected value)
mean_value <- sum(values * probabilities)

# Print the result
cat("Mean (Expected Value) =", mean_value, "\n")
```

```
## Mean (Expected Value) = 3.2
```

Or one can use the following built-in function:

```r
wt <- c(5,  5,  4,  1)/15
x <- c(3.7,3.3,3.5,2.8)
xm <- weighted.mean(x, wt)
xm
```

```
## [1] 3.453333
```

To calculate the variance of a probability distribution in R, you can use the Here's how you can do it:

```r
# Define the values of the random variable (x_i)
values <- c(1, 2, 3, 4, 5)

# Define the probabilities (P(x_i))
probabilities <- c(0.2, 0.3, 0.1, 0.2, 0.2)

# Calculate the mean (expected value) of the random variable
mean_x <- sum(values * probabilities)

# Calculate the variance using the formula
variance <- sum((values - mean_x)^2 * probabilities)

# Print the variance
cat("Variance:", variance, "\n")
```

```
## Variance: 2.09
```

5.3 Median

```r
# Create a sample vector
data_vector <- c(12, 45, 23, 67, 8, 34, 19)

# Calculate the median
```

```
median_value <- median(data_vector)

# Print the median
cat("Median:", median_value, "\n")
```

```
## Median: 23
```

5.4 Binomial probability distributions

You can generate a data set with a binomial distribution in R using the rbinom() function. This function simulates random numbers following a binomial distribution. Here's an example code to generate a data set with a binomial distribution:

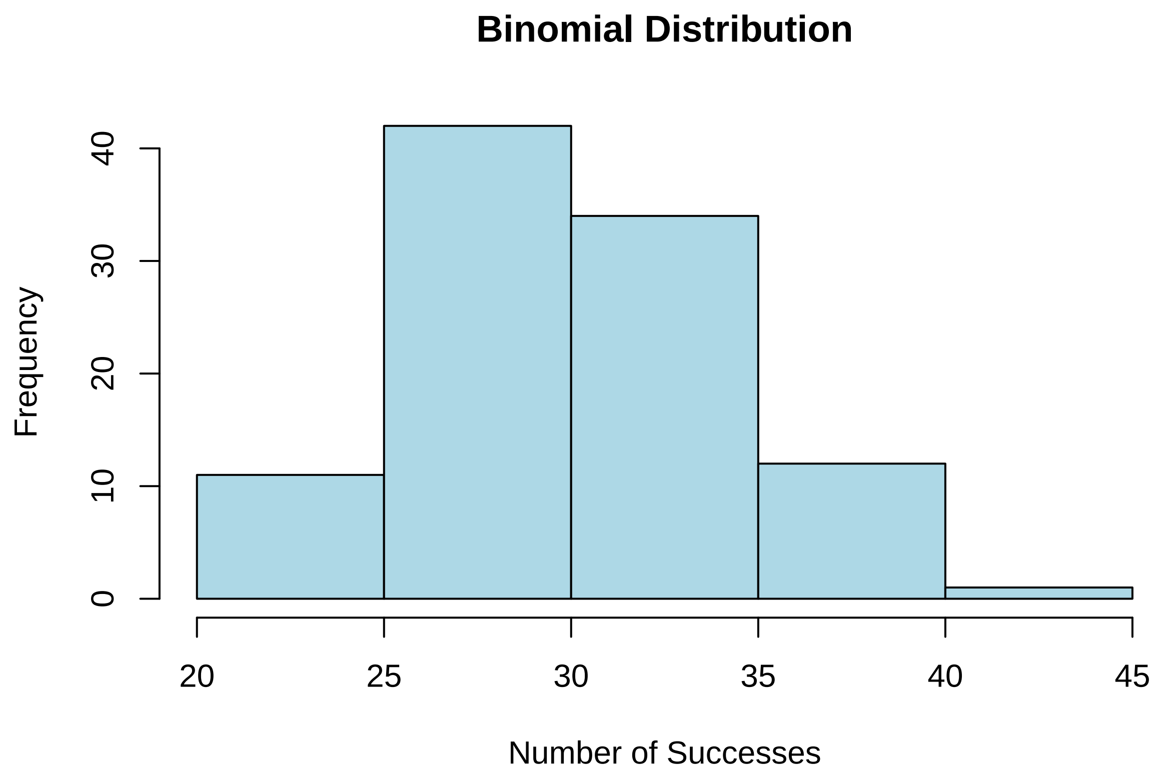
```
# Set the parameters for the binomial distribution
n <- 100    # Number of trials
p <- 0.3    # Probability of success in each trial

# Generate a dataset with a binomial distribution
binomial_data <- rbinom(n, size = n, prob = p)

# Print the generated dataset
print(binomial_data)
```

```
##   [1] 27 27 39 27 33 24 28 32 37 39 30 36 30 35 29 30 35 29 29 35 31 32 27 36 28
##  [26] 31 35 30 28 30 30 27 24 31 32 31 30 32 30 29 35 25 37 34 30 37 32 29 30 32
##  [51] 38 24 38 35 26 34 33 28 33 34 36 32 27 29 24 26 35 32 35 25 24 24 27 31 33
##  [76] 44 35 27 30 31 27 32 37 31 28 22 30 27 35 27 26 36 29 34 24 20 27 28 29 29
```

```
# Create a histogram to visualize the data
hist(binomial_data, main = "Binomial Distribution", xlab = "Number of Successes", ylab = "Frequency", c
```



```r
# verify the mean =np, and var=npq
# Sample mean
mean(binomial_data)
```

```
## [1] 30.64
```

```r
# Theoretical mean
n*p
```

```
## [1] 30
```

```r
# Sample variance
var(binomial_data)
```

```
## [1] 18.51556
```

```r
# Theoretical variance
n*p*(1-p)
```

```
## [1] 21
```

You can calculate the probability of specific outcomes in a binomial distribution in R using the dbinom() function, which calculates the *probability mass function* (PMF) of the binomial distribution. Here's how to use it:

```r
# Set the parameters for the binomial distribution
x <- 2     # Number of successes (the outcome you want to calculate the probability for)
n <- 10    # Number of trials
p <- 0.3   # Probability of success in each trial
```

```r
# Calculate the probability of getting 'x' successes in 'n' trials
probability <- dbinom(x, size = n, prob = p)

# Print the calculated probability
cat("Probability of", x, "successes in", n, "trials:", probability, "\n")
```

```
## Probability of 2 successes in 10 trials: 0.2334744
```

The `pbinom()` function in R is used to calculate cumulative probabilities for a binomial distribution. Specifically, it calculates the cumulative probability that a random variable following a binomial distribution is less than or equal to a specified value. In other words, it gives you the *cumulative distribution function* (CDF) for a binomial distribution.

Here's the basic syntax of the `pbinom()` function:

```r
pbinom(q, size, prob, lower.tail = TRUE)
```

`q`: The value for which you want to calculate the cumulative probability.

`size`: The number of trials or events in the binomial distribution.

`prob`: The probability of success in each trial.

`lower.tail`: A logical parameter that determines whether you want the cumulative probability for values less than or equal to `q` (`TRUE`) or greater than `q` (`FALSE`). By default, it is set to `TRUE`.

The pbinom() function returns the cumulative probability for the specified value q based on the given parameters.

Here's an example of how to use `pbinom()`:

```r
# Calculate the cumulative probability that X is less than or equal to 3
cumulative_prob <- pbinom(3, size = 10, prob = 0.3)

# Print the cumulative probability
cat("Cumulative Probability:", cumulative_prob, "\n")
```

```
## Cumulative Probability: 0.6496107
```

In this example, we're calculating the cumulative probability that a random variable following a binomial distribution with parameters `size = 10` and `prob = 0.3` is less than or equal to `3`. The result is stored in the cumulative_prob variable and printed to the console.

You can use the `pbinom()` function to answer questions like "What is the probability of getting at most 3 successes in 10 trials with a success probability of 0.3?" by specifying the appropriate values for q, size, and prob.

5.5 Poisson probability distributions (Optional)

To generate a data set with a Poisson distribution in R, you can use the `rpois()` function. The Poisson distribution is often used to model the number of events occurring in a fixed interval of time or space when the events happen with a known constant mean rate. Here's how you can use `rpois()`:

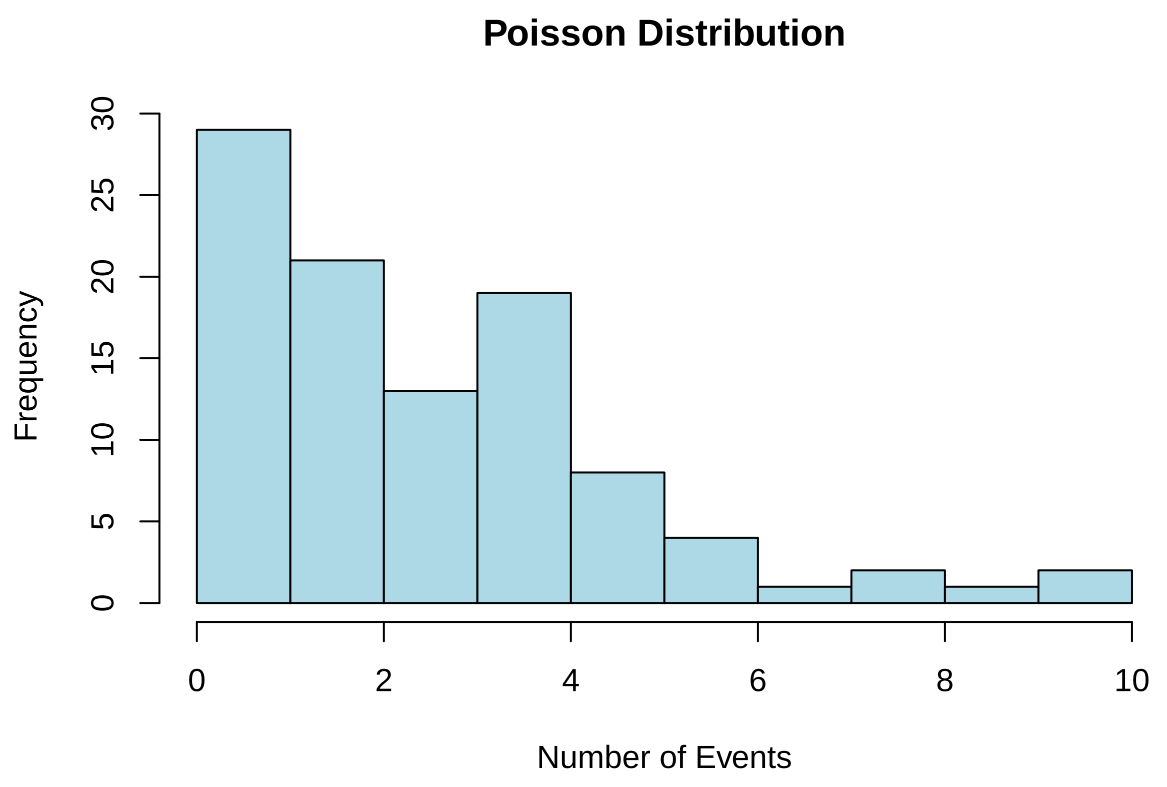
```r
# Set the parameters for the Poisson distribution
lambda <- 3  # Mean (average) rate of events

# Generate a dataset with a Poisson distribution
poisson_data <- rpois(n = 100, lambda = lambda)
```

```r
# Print the generated dataset
print(poisson_data)
```

```
##   [1]  3  0  1  3  4  4  4  1  3  4  2  8  1  3  2  2  2  9  4  4  5  4  4  6  1
##  [26]  6  6  1  5  4  3  2  0  2  1  2  3  0  4  4  1  1 10  5  0  4  2  0  2  1
##  [51]  2  5  2  2  4  0  1 10  0  4  3  2  3  8  3  7  4  4  3  1  4  2  0  3  1
##  [76]  1  3  1  2  2  5  0  1  6  4  5  2  1  1  2  2  1  3  4  1  2  2  5  1  5
```

```r
# Create a histogram to visualize the data
hist(poisson_data, main = "Poisson Distribution", xlab = "Number of Events", ylab = "Frequency", col =
```



```r
# Verify the theoretical mean and variance
mean(poisson_data)
```

```
## [1] 2.93
```

```r
#Theoretical mean = lambda

var(poisson_data)
```

```
## [1] 4.732424
```

```r
#Theoretical variance = lambda
```

To calculate the probability of a specific value occurring in a Poisson distribution in R, you can use the dpois() function. This function calculates the *probability mass function* (PMF) of the Poisson distribution. Here's how to use it:

```r
# Set the parameters for the Poisson distribution
x <- 2     # The specific value for which you want to calculate the probability
lambda <- 3  # Mean (average) rate of events

# Calculate the probability of getting exactly 'x' events
probability <- dpois(x, lambda)

# Print the calculated probability
cat("Probability of", x, "events:", probability, "\n")
```

```
## Probability of 2 events: 0.2240418
```

To calculate the *cumulative distribution function* (CDF) for a Poisson distribution in R, you can use the ppois() function. This function calculates the cumulative probability that a Poisson random variable is less than or equal to a specified value. Here's how to use it:

```r
# Set the parameters for the Poisson distribution
x <- 2     # The specific value for which you want to calculate the cumulative probability
lambda <- 3  # Mean (average) rate of events

# Calculate the cumulative probability of getting less than or equal to 'x' events
cumulative_prob <- ppois(x, lambda)

# Print the calculated cumulative probability
cat("Cumulative Probability of less than or equal to", x, "events:", cumulative_prob, "\n")
```

```
## Cumulative Probability of less than or equal to 2 events: 0.4231901
```