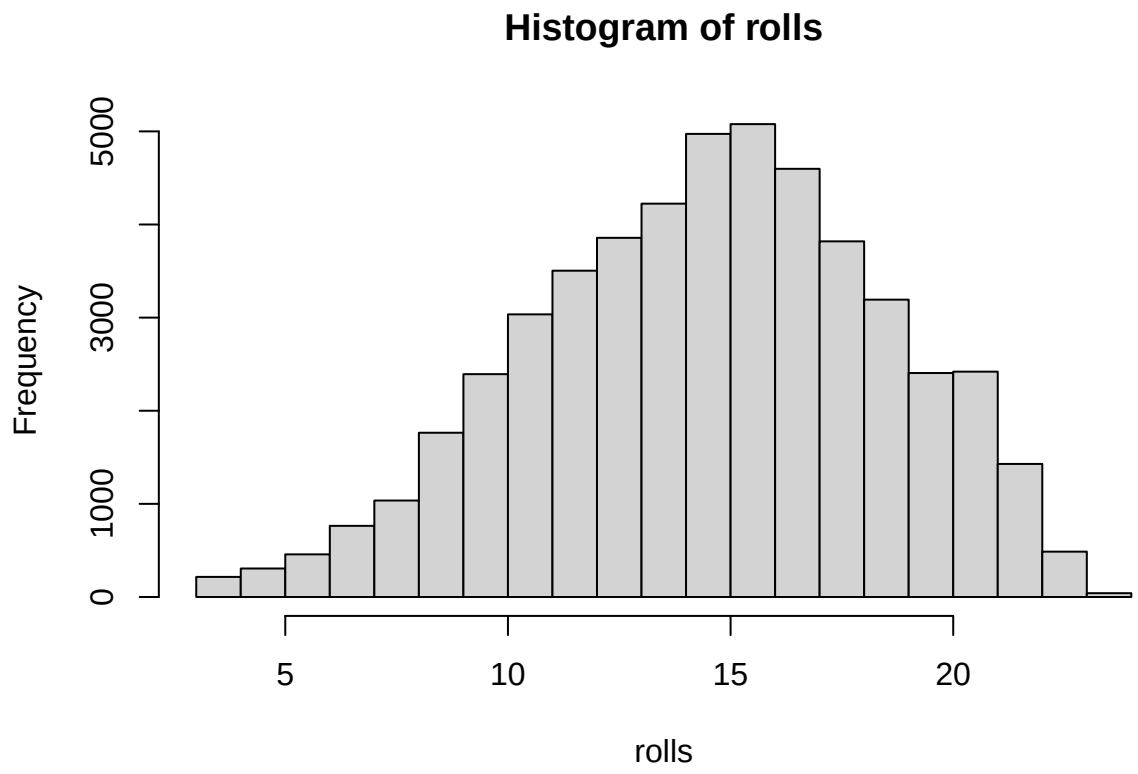
functions

Tyler Kephart

2024-08-06

Question 1. Create a function that produces a histogram of 50,000 rolls of three 8 sided dice. Each die is loaded so that the number 7 has a higher probability of being rolled than the other numbers, assume all other sides of the die have a 1/10 probability of being rolled.

```
roll <- function() {
  die <- 1:8
  # 7 in favor at 3/10 prob, all else 1/10
  dice <- sample(die, size = 3, replace = TRUE,
    prob = c(1/10, 1/10, 1/10, 1/10, 1/10, 1/10, 3/10, 1/10))
  sum(dice)
}
plot_50k_rolls <- function() {
  # roll 50K times then plot on histogram
  rolls <- replicate(50000, roll())
  hist(rolls)
}
plot_50k_rolls()
```



Question 2. Write a function, rescale01(), that recieves a vector as an input and checks that the inputs are all numeric. If the input vector is numeric, map any -Inf and Inf values to 0 and 1, respectively. If the input vector is non-numeric, stop the function and return the message "inputs must all be numeric".

```
rescale01 <- function(x) {
  if(all(is.na(as.numeric(x)))){
    stop("inputs must all be numeric")
  } else {
    rng <- range(x, na.rm=TRUE, finite=TRUE)
    y <- (x - rng[1]) / (rng[2] - rng[1])
    y[y == -Inf] <- 0
    y[y == Inf] <- 1
    y
  }
}
x <- c(-Inf,0,-111,Inf,420,2222,-6996)
rescale01(x)
```

```
## [1] 0.0000000 0.7589499 0.7469082 1.0000000 0.8045129 1.0000000 0.0000000
```

Question 3. Write a function that takes two vectors of the same length and returns the number of positions that have an NA in both vectors. If the vectors are not the same length, stop the function and return the message "vectors must be the same length".

```
both_na <- function(x, y) {
  if (length(x) != length(y)) {
    stop("vectors must be the same length")
  } else {
    # NA is TRUE, TRUE = 1, FALSE = 0
    # multiply and add up to get number of positions both NA
    sum(is.na(x) * is.na(y))
  }
}
x <- c(1,2,NA,4,NA,NA)
y <- c(1,NA,NA,NA,NA,6)
both_na(x, y)
```

```
## [1] 2
```

Question 4. Implement a fizzbuzz function. It takes a single number as input. If the number is divisible by three, it returns "fizz". If it's divisible by five it returns "buzz". If it's divisible by three and five, it returns "fizzbuzz". Otherwise, it returns the number.

```
fizzbuzz <- function(x) {
  # verify x is a single whole number
  if ((!is.numeric(x)) || (length(x) > 1)) {
    stop("input one whole number")
    # divisible by 3 and 5
  } else if ((x %% 3 == 0) && (x %% 5 == 0)) {
    print("fizzbuzz")
    # divisible by 5
  } else if (x %% 5 == 0) {
    print("buzz")
```

```
    # divisible by 3
  } else if (x %% 3 == 0) {
    print("fizz")
    # otherwise itself
  } else {
    print(x)
  }
}
fizzbuzz(1)
```

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

```
fizzbuzz(6)
```

```
## [1] "fizz"
```

```
fizzbuzz(10)
```

```
## [1] "buzz"
```

```
fizzbuzz(15)
```

```
## [1] "fizzbuzz"
```

Question 5. Rewrite the function below using `cut()` to simplify the set of nested if-else statements.

```
if (temp <= 0) {
  "freezing"
  } else if (temp <= 10) {
  "cold"
  } else if (temp <= 20) {
  "cool"
  } else if (temp <= 30) {
  "warm"
  } else {
  "hot"
  }
```

```
temp_feeling <- function(temp) {
  # freezing <= 0, cold <= 10, cool <= 20
  # warm <= 30, hot > 30
  cut(temp,
      breaks=c(-Inf,0,10,20,30,Inf),
      labels=c("freezing","cold","cool","warm","hot"))
}
temp_feeling(-5)
```

```
## [1] freezing
## Levels: freezing cold cool warm hot
```

```
temp_feeling(0)
```

```
## [1] freezing
## Levels: freezing cold cool warm hot
```

```r
temp_feeling(5)
```

```
## [1] cold
## Levels: freezing cold cool warm hot
```

```r
temp_feeling(10)
```

```
## [1] cold
## Levels: freezing cold cool warm hot
```

```r
temp_feeling(15)
```

```
## [1] cool
## Levels: freezing cold cool warm hot
```

```r
temp_feeling(20)
```

```
## [1] cool
## Levels: freezing cold cool warm hot
```

```r
temp_feeling(25)
```

```
## [1] warm
## Levels: freezing cold cool warm hot
```

```r
temp_feeling(30)
```

```
## [1] warm
## Levels: freezing cold cool warm hot
```

```r
temp_feeling(35)
```

```
## [1] hot
## Levels: freezing cold cool warm hot
```