New service year data analysis

ZahraAfaghi

2024-12-01

part1 Data recall # After the salaries data from the car package has selected a random sample of jobs and stored their years of service in the yrs_service variable, then according to the value stored in my.id, the number of lines that are randomly selected is determined they do

```
my.id <- 20 + 63
set.seed(my.id)
library(car)
```

```
## Loading required package: carData
```

```
data(Salaries)
sample_data <- Salaries[sample(nrow(Salaries), my.id), ]
yrs_service <- sample_data$yrs.service
```

This code is used to calculate and display descriptive statistics of yrs.service from sample_data data, which includes mean, median, standard deviation, variance, range, and quartiles, and finally the results are printed in text form.

```
yrs_service <- sample_data$yrs.service

mean_yrs_service <- mean(yrs_service)
median_yrs_service <- median(yrs_service)
sd_yrs_service <- sd(yrs_service)
var_yrs_service <- var(yrs_service)
range_yrs_service <- range(yrs_service)
quartiles <- quantile(yrs_service)

print(paste("Mean:", mean_yrs_service))
```

```
## [1] "Mean: 16.3373493975904"
```

```
print(paste("Median:", median_yrs_service))
```

```
## [1] "Median: 14"
```

```
print(paste("Standard Deviation:", sd_yrs_service))
```

```
## [1] "Standard Deviation: 12.719561570556"
```

```
print(paste("Variance:", var_yrs_service))
```

```
## [1] "Variance: 161.787246547164"
```

```
print(paste("Range:", range_yrs_service[1], "to", range_yrs_service[2]))
```

```
## [1] "Range: 0 to 49"
```

```
print(paste("Quartiles:", quartiles))
```

```
## [1] "Quartiles: 0"  "Quartiles: 6"  "Quartiles: 14" "Quartiles: 23"
## [5] "Quartiles: 49"
```

#The third quartile, or Q3, is the value that has 75% of the data under it and actually defines the third cutoff value. This means that 25% of the data is above this value. The third quartile can indicate that here are employees with higher salaries than the majority of employees.

Here, using the e1071 library, we calculate the skewness and kurtosis of yrs.service and then print the resulting values.

```
library(e1071)
skewness_val <- skewness(yrs_service)
kurtosis_val <- kurtosis(yrs_service)
print(paste("Skewness:", skewness_val))
```

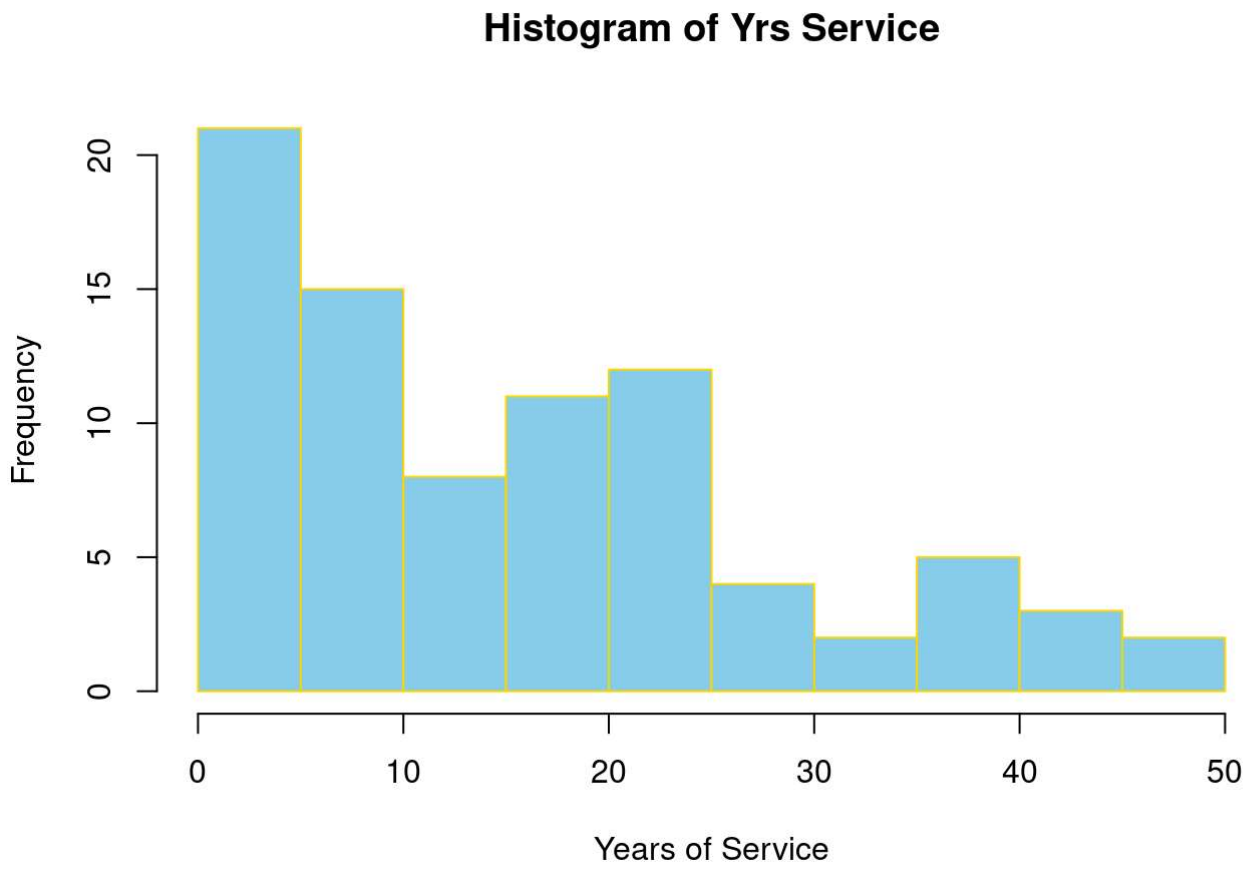
```
## [1] "Skewness: 0.76865579871953"
```

```
print(paste("Kurtosis:", kurtosis_val))
```

```
## [1] "Kurtosis: -0.25923239715185"
```

We draw the data in the form of a histogram, the horizontal axis shows the number of years of service

```
hist(yrs_service, main="Histogram of Yrs Service", xlab="Years of Service", col = "skyblue",border = "gold")
```



part2 Draw a box plot for salary data

```
install.packages("ggplot2")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
install.packages("car")
```

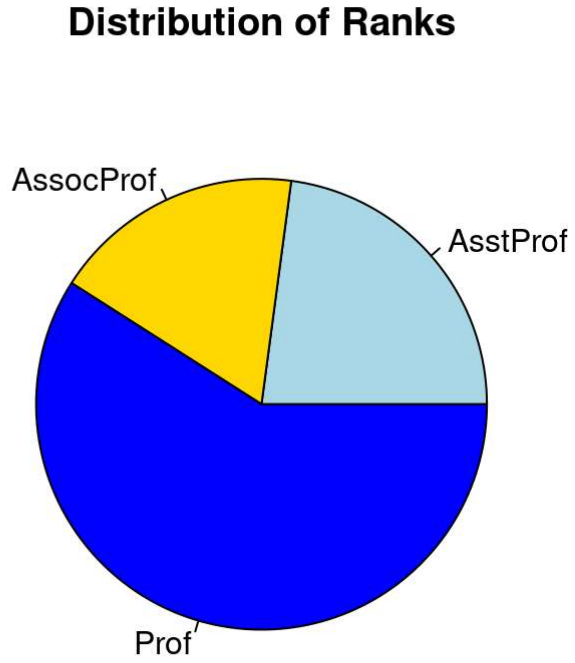
```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
library(car)
library(ggplot2)

my.id <- (20 + 63)
set.seed(my.id)

data("Salaries")
sample_data <- Salaries[sample(nrow(Salaries), my.id), ]

rank_count <- table(sample_data$rank)
colors <- c("lightblue", "gold", "blue")
pie(rank_count, main = "Distribution of Ranks", col = colors[1:length(rank_count)])
```




```
ggplot(sample_data, aes(x = salary)) +
  geom_histogram(bins = 30, fill = "skyblue", color = "gold") +
  labs(title = "Salary Distribution", x = "Salary", y = "Count") +
  theme_minimal()
```

part3 Find multiples of 6 #

Generates a series of random numbers and then rounds the numbers and selects only multiples of 6. The important thing is that using set.seed(my.id) the generation of random numbers can be repeated, so every time this code is executed Run, the same numbers will be generated.

```
my.id <- 63 + 20
set.seed(my.id)
numbers <- rnorm(100, mean=50, sd=10)
rounded_numbers <- round(numbers)
multiples_of_6 <- rounded_numbers[rounded_numbers %% 6 == 0]
print(multiples_of_6)
```

```
##  [1] 60 36 60 42 54 60 60 54 54 48 48 48 48 42 54 54
```

part4 # The following code divides the number of elements of a vector by the sum(x) of a function named f and returns the result. Then this function is applied to the salary variable from the Salaries dataframe.

```
my.id <- 20 + 63
set.seed(my.id)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:car':
##
##     recode
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
f <- function(x) {
  n <- length(x)
  result <- n / sum(x)
  return(result)
}

output <- f(Salaries$salary)


print(output)
```

```
## [1] 8.794575e-06
```

part5 # To calculate the integral with high accuracy and other calculations, we use integrate from the pracma package

```
library(pracma)
```

```
##
## Attaching package: 'pracma'
```

```
## The following object is masked from 'package:e1071':
##
##     sigmoid
```

```
## The following object is masked from 'package:car':
##
##     logit
```

```
integrand1 <- function(x) {
  return(1 + x^2)
}


result1 <- integrate(integrand1, 1, 3)
cat("The result of the first integral: ", result1$value, "\n")
```

```
## The result of the first integral:  10.66667
```

part5(1) # for each value y (from 0 to 1) that changes with a step of 1., the internal integral is calculated on the integrand2 function, which includes the variables x and y . Then the value of the inner integral for each y is added to the total result.

```
integrand2 <- function(x, y) {
  return(x + y^2 - 1)
}
result2 <- 0
for (y in seq(0, 1, by = 0.01)) {
  inner_integral <- integrate(function(x) integrand2(x, y), 3, 5)
  result2 <- result2 + inner_integral$value * 0.01
}

cat("The result of the second integral: ", result2, "\n")
```

```
## The result of the second integral:  6.7367
```

part6 # The above code creates a 3x3 matrix named A and then calculates and prints its determinant, also calculates and displays the inverse of the matrix A. After that, the values of the original diameter of the matrix are replaced by the zero vector (0, 0, 0) and the edited matrix is displayed.

```
A <- matrix(c(1, 6, 9, 2, 1, 5, 4, 3, 3), nrow = 3, byrow = TRUE)

det_A <- det(A)
cat("Determinants of the matrix A:", det_A, "\n")
```

```
## Determinants of the matrix A: 90
```

```
inv_A <- solve(A)
cat("Matrix inverse A:\n")
```

```
## Matrix inverse A:
```

```
print(inv_A)
```

```
##             [,1]       [,2]       [,3]
## [1,] -0.13333333  0.1000000  0.2333333
## [2,]  0.15555556 -0.3666667  0.1444444
## [3,]  0.02222222  0.2333333 -0.1222222
```

```
zero_vector <- c(0, 0, 0)
diag(A) <- zero_vector
cat("Matrix A after replacing the values on the main diameter with (0, 0, 0):\n")
```

```
## Matrix A after replacing the values on the main diameter with (0, 0, 0):
```

```
print(A)
```

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