Author: Akash Prasad

Date: October 5, 2025

Environment reset (run first)

```
rm(list = ls())                 # clear objects
graphics.off()                  # close plots
cat("\014")                     # clear console
```

Step 1: Computing the requested arithmetic and logical expressions

```
123 * 453
```

```
## [1] 55719
```

```
5^2 * 40
```

```
## [1] 1000
```

```
TRUE & FALSE
```

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

```
TRUE | FALSE
```

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

```
75 %% 10
```

```
## [1] 5
```

```
75 / 10
```

```
## [1] 7.5
```

Step 2: Creating a vector with c() function and storing it in variable "first_vector"

```
first_vector <- c(17, 12, -33, 5)
first_vector
```

```
## [1]  17  12 -33   5
```

Step 3: Creating a 5-multiples vector with c() function and storing it in variable "counting_by_fives"

```
counting_by_fives <- c(5, 10, 15, 20, 25, 30, 35)
counting_by_fives
```

```
## [1]  5 10 15 20 25 30 35
```

Step 4: Creating a descending vector from 20 to 1 using range operator ":" and storing it in "second_vector"

```
second_vector <- 20:1
second_vector
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

Step 5: Creating a vector from 5 to 15 using range operator ":" and storing it in "counting_vector"

```
counting_vector <- 5:15
counting_vector
```

```
##  [1]  5  6  7  8  9 10 11 12 13 14 15
```

Step 6: Creating a vector of grades and storing it in "grades"

```
grades <- c(96, 100, 85, 92, 81, 72)
grades
```

```
## [1]  96 100  85  92  81  72
```

Step 7: Adding 3 bonus points to grades and storing it in "bonus_points_added"

```
bonus_points_added <- grades + 3
bonus_points_added
```

```
## [1]  99 103  88  95  84  75
```

Step 8: Creating a vector of numbers 1–100 using range operator ":" and storing it in "one_to_one_hundred"

```
one_to_one_hundred <- 1:100
one_to_one_hundred
```

```
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  2
##  [40]  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  6
##  [79]  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
```

Step 9: Performing addition, multiplication, and logical comparisons on "second_vector"

```
#'Reading "second_vector"
second_vector
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

```
#'Adding 20 to "second_vector" elements
second_vector + 20
```

```
##  [1] 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21
```

```
#'Reading "second_vector"
second_vector
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

```
#'Multiplying 20 to "second_vector" elements
second_vector * 20
```

```
##  [1] 400 380 360 340 320 300 280 260 240 220 200 180 160 140 120 100  80  60  40  20
```

```
#'Reading "second_vector"
second_vector
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

```
#'Checking if "second_vector" elements are greater or equal than 20
second_vector >= 20
```

```
##  [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
#'Reading "second_vector"
second_vector
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

```
#'Checking if "second_vector" elements are not equal to 20
second_vector != 20
```

```
##  [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
#'Reading "second_vector"
second_vector
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

Since we are only computing arithmetic and logical expressions and not updating the "second_vector", its value remains unchanged

Step 10: Calculating sum of numbers from 1–100 and storing it in "total"

```
total <- sum(one_to_one_hundred)
total
```

```
## [1] 5050
```

Step 11: Calculating average of numbers from 1–100 and storing it in "average_value"

```
average_value <- mean(one_to_one_hundred)
average_value
```

```
## [1] 50.5
```

Step 12: Calculating median of numbers from 1–100 and storing it in "median_value"

```
median_value <- median(one_to_one_hundred)
median_value
```

```
## [1] 50.5
```

Step 13: Finding maximum value from 1–100 and storing it in "max_value"

```
max_value <- max(one_to_one_hundred)
max_value
```

```
## [1] 100
```

Step 14: Finding minimum value from 1–100 and storing it in "min_value"

```
min_value <- min(one_to_one_hundred)
min_value
```

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

Step 15: Extracting first element from “second_vector” and storing it in “first_value”

```
first_value <- second_vector[1]
first_value
```

```
## [1] 20
```

Step 16: Extracting first three elements from “second_vector” and storing them in “first_three_values”

```
first_three_values <- second_vector[1:3]
first_three_values
```

```
## [1] 20 19 18
```

Step 17: Extracting 1st, 5th, 10th, and 11th elements from “second_vector” and storing in “vector_from_brackets”

```
vector_from_brackets <- second_vector[c(1, 5, 10, 11)]
vector_from_brackets
```

```
## [1] 20 16 11 10
```

Step 18: Using Boolean indexing to extract selected elements from “first_vector” and storing in “vector_from_boolean_brackets”

```
vector_from_boolean_brackets <- first_vector[c(FALSE, TRUE, FALSE, TRUE)]
vector_from_boolean_brackets
```

```
## [1] 12  5
```

Only elements where the index is True is filtered and stored in new variable “vector_from_boolean_brackets”

Step 19: Checking which elements in “second_vector” are greater than or equal to 10

```
second_vector >= 10
```

```
##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
```

```
second_vector
```

```
##  [1] 20 19 18 17 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

First 10 elements of the "second_vector" are greater or equal to 10, meanwhile the "second_vector" is unchanged

Step 20: Filtering "one_to_one_hundred" to keep only values >= 20

```
one_to_one_hundred[one_to_one_hundred  >=  20]
```

```
##  [1]  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43
## [41]  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83
## [81] 100
```

```
one_to_one_hundred
```

```
##  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  2
## [40]  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  6
## [79]  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
```

Only 81 elements are selected only for viewing (and not updated) out of 100, with filter condition of "one_to_one_hundred" elements greater or equal to 20.

Step 21: Filtering "grades" to keep only values greater than 85 and storing in "lowest_grades_removed"

```
lowest_grades_removed <- grades[grades > 85]
lowest_grades_removed
```

```
## [1]  96 100  92
```

Step 22: Removing 3rd and 4th elements from "grades" and storing in "middle_grades_removed"

```
middle_grades_removed <- grades[-c(3, 4)]
middle_grades_removed
```

```
## [1]  96 100  81  72
```

Step 23: Removing 5th and 10th elements from "second_vector" and storing in "fifth_vector"

```
fifth_vector <- second_vector[-c(5, 10)]
fifth_vector
```

```
##  [1] 20 19 18 17 15 14 13 12 10  9  8  7  6  5  4  3  2  1
```

Step 24: Setting seed and generating 10 random numbers between 0 and 1000 using runif(), storing in "random_vector"

```
set.seed(5)
random_vector <- runif(n = 10, min = 0, max = 1000)
random_vector
```

```
##  [1] 200.2145 685.2186 916.8758 284.3995 104.6501 701.0575 527.9600 807.9352 956.5001 110.4530
```

Step 25: Calculating total of "random_vector" and storing in "sum_vector"

```
sum_vector <- sum(random_vector)
sum_vector
```

```
## [1] 5295.264
```

Step 26: Calculating cumulative sum of "random_vector" and storing in "cumsum_vector"

```
cumsum_vector <- cumsum(random_vector)
cumsum_vector
```

```
##   [1]    200.2145    885.4330 1802.3088 2086.7083 2191.3584 2892.4159 3420.3759 4228.3111 5184.8112 5295.
```

Step 27: Calculating mean of "random_vector" and storing in "mean_vector"

```
mean_vector <- mean(random_vector)
mean_vector
```

```
## [1] 529.5264
```

Step 28: Calculating standard deviation of "random_vector" and storing in "sd_vector"

```
sd_vector <- sd(random_vector)
sd_vector
```

```
## [1] 331.3606
```

Step 29: Rounding values of "random_vector" to nearest integers and storing in "round_vector"

```
round_vector <- round(random_vector)
round_vector
```

```
##   [1] 200 685 917 284 105 701 528 808 957 110
```

Step 30: Sorting values of "random_vector" in ascending order and storing in "sort_vector"

```
sort_vector <- sort(random_vector)
sort_vector
```

```
##   [1] 104.6501 110.4530 200.2145 284.3995 527.9600 685.2186 701.0575 807.9352 916.8758 956.5001
```

Step 31: Downloading the datafile ds_salaries.csv from Canvas and Saving it in same directory as this project

Step 32: Reading dataset "ds_salaries.csv" into R and storing it as "first_dataframe"

```
first_dataframe <- read.csv("ds_salaries.csv")
first_dataframe
```

```
##        X work_year experience_level employment_type                    job_title
## 1      0      2020                MI              FT                Data Scientist
## 2      1      2020                SE              FT      Machine Learning Scientist
## 3      2      2020                SE              FT         Big Data Engineer
## 4      3      2020                MI              FT       Product Data Analyst
## 5      4      2020                SE              FT      Machine Learning Engineer
## 6      5      2020                EN              FT              Data Analyst
## 7      6      2020                SE              FT         Lead Data Scientist
## 8      7      2020                MI              FT                Data Scientist
## 9      8      2020                MI              FT       Business Data Analyst
## 10     9      2020                SE              FT         Lead Data Engineer
## 11    10      2020                EN              FT                Data Scientist
## 12    11      2020                MI              FT                Data Scientist
## 13    12      2020                EN              FT                Data Scientist
## 14    13      2020                MI              FT         Lead Data Analyst
## 15    14      2020                MI              FT              Data Analyst
## 16    15      2020                MI              FT              Data Analyst
## 17    16      2020                EN              FT             Data Engineer
## 18    17      2020                SE              FT         Big Data Engineer
## 19    18      2020                EN              FT     Data Science Consultant
## 20    19      2020                MI              FT         Lead Data Engineer
## 21    20      2020                MI              FT      Machine Learning Engineer
## 22    21      2020                MI              FT       Product Data Analyst
## 23    22      2020                SE              FT             Data Engineer
## 24    23      2020                MI              FT          BI Data Analyst
## 25    24      2020                MI              FT         Lead Data Scientist
## 26    25      2020                EX              FT   Director of Data Science
## 27    26      2020                EN              FT         Research Scientist
## 28    27      2020                SE              FT             Data Engineer
## 29    28      2020                EN              CT       Business Data Analyst
## 30    29      2020                SE              FT      Machine Learning Manager
## 31    30      2020                MI              FT   Data Engineering Manager
## 32    31      2020                EN              FT         Big Data Engineer
## 33    32      2020                SE              FT                Data Scientist
## 34    33      2020                MI              FT         Research Scientist
## 35    34      2020                MI              FT              Data Analyst
## 36    35      2020                MI              FT             Data Engineer
## 37    36      2020                MI              FT     Data Science Consultant
## 38    37      2020                EN              FT      Machine Learning Engineer
## 39    38      2020                EN              FT              Data Analyst
## 40    39      2020                EN              FT      Machine Learning Engineer
## 41    40      2020                MI              FT                Data Scientist
##    salary
## 1   70000
## 2  260000
## 3   85000
## 4   20000
## 5  150000
## 6   72000
## 7  190000
## 8  11000000
## 9  135000
## 10 125000
## 11  45000
## 12 3000000
## 13  35000
## 14  87000
## 15  85000
## 16   8000
## 17 4450000
## 18 100000
## 19 423000
## 20  56000
## 21 299000
## 22 450000
## 23  42000
## 24  98000
## 25 115000
## 26 325000
## 27  42000
## 28 720000
## 29 100000
## 30 157000
## 31  51999
## 32  70000
## 33  60000
## 34 450000
## 35  41000
## 36  65000
## 37 103000
## 38 250000
## 39  10000
## 40 138000
## 41  45760
```

##	42	41	2020	EX	FT	Data Engineering Manager	70000
##	43	42	2020	MI	FT	Machine Learning Infrastructure Engineer	44000
##	44	43	2020	MI	FT	Data Engineer	106000
##	45	44	2020	MI	FT	Data Engineer	88000
##	46	45	2020	EN	PT	ML Engineer	14000
##	47	46	2020	MI	FT	Data Scientist	60000
##	48	47	2020	SE	FT	Data Engineer	188000
##	49	48	2020	MI	FT	Data Scientist	105000
##	50	49	2020	MI	FT	Data Engineer	61500
##	51	50	2020	EN	FT	Data Analyst	450000
##	52	51	2020	EN	FT	Data Analyst	91000
##	53	52	2020	EN	FT	AI Scientist	300000
##	54	53	2020	EN	FT	Data Engineer	48000
##	55	54	2020	SE	FL	Computer Vision Engineer	60000
##	56	55	2020	SE	FT	Principal Data Scientist	130000
##	57	56	2020	MI	FT	Data Scientist	34000
##	58	57	2020	MI	FT	Data Scientist	118000
##	59	58	2020	SE	FT	Data Scientist	120000
##	60	59	2020	MI	FT	Data Scientist	138350
##	61	60	2020	MI	FT	Data Engineer	110000
##	62	61	2020	MI	FT	Data Engineer	130800
##	63	62	2020	EN	PT	Data Scientist	19000
##	64	63	2020	SE	FT	Data Scientist	412000
##	65	64	2020	SE	FT	Machine Learning Engineer	40000
##	66	65	2020	EN	FT	Data Scientist	55000
##	67	66	2020	EN	FT	Data Scientist	43200
##	68	67	2020	SE	FT	Data Science Manager	190200
##	69	68	2020	EN	FT	Data Scientist	105000
##	70	69	2020	SE	FT	Data Scientist	80000
##	71	70	2020	MI	FT	Data Scientist	55000
##	72	71	2020	MI	FT	Data Scientist	37000
##	73	72	2021	EN	FT	Research Scientist	60000
##	74	73	2021	EX	FT	BI Data Analyst	150000
##	75	74	2021	EX	FT	Head of Data	235000
##	76	75	2021	SE	FT	Data Scientist	45000
##	77	76	2021	MI	FT	BI Data Analyst	100000
##	78	77	2021	MI	PT	3D Computer Vision Researcher	400000
##	79	78	2021	MI	CT	ML Engineer	270000
##	80	79	2021	EN	FT	Data Analyst	80000
##	81	80	2021	SE	FT	Data Analytics Engineer	67000
##	82	81	2021	MI	FT	Data Engineer	140000
##	83	82	2021	MI	FT	Applied Data Scientist	68000

		company_location	company_size
##	1	DE	L
##	2	JP	S
##	3	GB	M
##	4	HN	S
##	5	US	L
##	6	US	L
##	7	US	S
##	8	HU	L
##	9	US	L
##	10	NZ	S
##	11	FR	S

## 12	IN	L
## 13	FR	M
## 14	US	L
## 15	US	L
## 16	PK	L
## 17	JP	S
## 18	GB	S
## 19	IN	M
## 20	US	M
## 21	CN	M
## 22	IN	L
## 23	GR	L
## 24	US	M
## 25	AE	L
## 26	US	L
## 27	NL	L
## 28	MX	S
## 29	US	L
## 30	CA	L
## 31	DE	S
## 32	US	L
## 33	US	L
## 34	US	M
## 35	FR	L
## 36	AT	L
## 37	US	L
## 38	US	L
## 39	NG	S
## 40	US	S
## 41	US	S
## 42	ES	L
## 43	PT	M
## 44	US	L
## 45	GB	L
## 46	DE	S
## 47	GB	S
## 48	US	L
## 49	US	L
## 50	FR	L
## 51	IN	S
## 52	US	L
## 53	DK	S
## 54	DE	L
## 55	US	S
## 56	DE	M
## 57	ES	M
## 58	US	M
## 59	US	L
## 60	US	M
## 61	US	L
## 62	US	M
## 63	IT	S
## 64	US	L
## 65	HR	S

```
## 66                 DE              S
## 67                 DE              S
## 68                 US              M
## 69                 US              S
## 70                 AT              S
## 71                 LU              S
## 72                 FR              S
## 73                 GB              L
## 74                 US              L
## 75                 US              L
## 76                 FR              L
## 77                 US              M
## 78                 IN              M
## 79                 US              L
## 80                 US              M
## 81                 DE              L
## 82                 US              L
## 83                 CA              L
##  [ reached 'max' / getOption("max.print") -- omitted 524 rows ]
```

Step 33: Generating summary statistics of the dataset "first_dataframe"

```
summary(first_dataframe)
```

```
##        X           work_year     experience_level   employment_type    job_title            salary
##  Min.   :  0.0   Min.   :2020   Length:607         Length:607         Length:607         Min.   :
##  1st Qu.:151.5   1st Qu.:2021   Class :character   Class :character   Class :character   1st Qu.:
##  Median :303.0   Median :2022   Mode  :character   Mode  :character   Mode  :character   Median :  1
##  Mean   :303.0   Mean   :2021                                                           Mean   :  3
##  3rd Qu.:454.5   3rd Qu.:2022                                                           3rd Qu.:  1
##  Max.   :606.0   Max.   :2022                                                           Max.   :304
##   remote_ratio    company_location   company_size
##  Min.   :  0.00   Length:607         Length:607
##  1st Qu.: 50.00   Class :character   Class :character
##  Median :100.00   Mode  :character   Mode  :character
##  Mean   : 70.92
##  3rd Qu.:100.00
##  Max.   :100.00
```

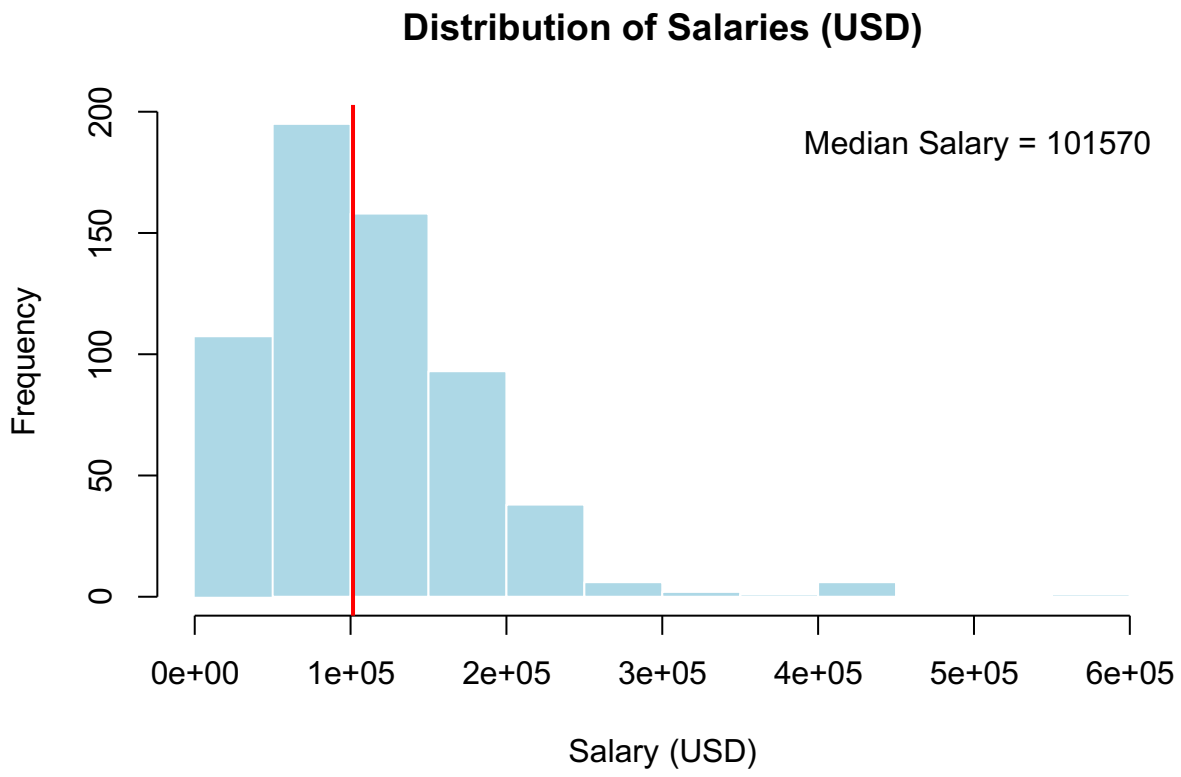
Visualizations + Recommendations

1. Salary Distribution (Histogram)

Reason: Helps visualize how salaries are spread, showing skewness and outliers.

```
hist(first_dataframe$salary_in_usd,
     main = "Distribution of Salaries (USD)",
     xlab = "Salary (USD)",
```

```
      col = "lightblue",
      border = "white")
abline(v = median(first_dataframe$salary_in_usd, na.rm = TRUE),
       col = "red", lwd = 2)
legend("topright", legend = c(paste("Median Salary =", round(median(first_dataframe$salary_in_usd, na.r
       bty = "n")
```



2. Salary by Experience Level (Boxplot)

Reason: Shows how pay changes with experience, revealing career-level pay gaps.

```
boxplot(salary_in_usd ~ experience_level, data = first_dataframe,
        main = "Salary by Experience Level",
        xlab = "Experience Level",
        ylab = "Salary (USD)",
        col = "lightgreen")
```

3. Salary by Company Size (Boxplot)

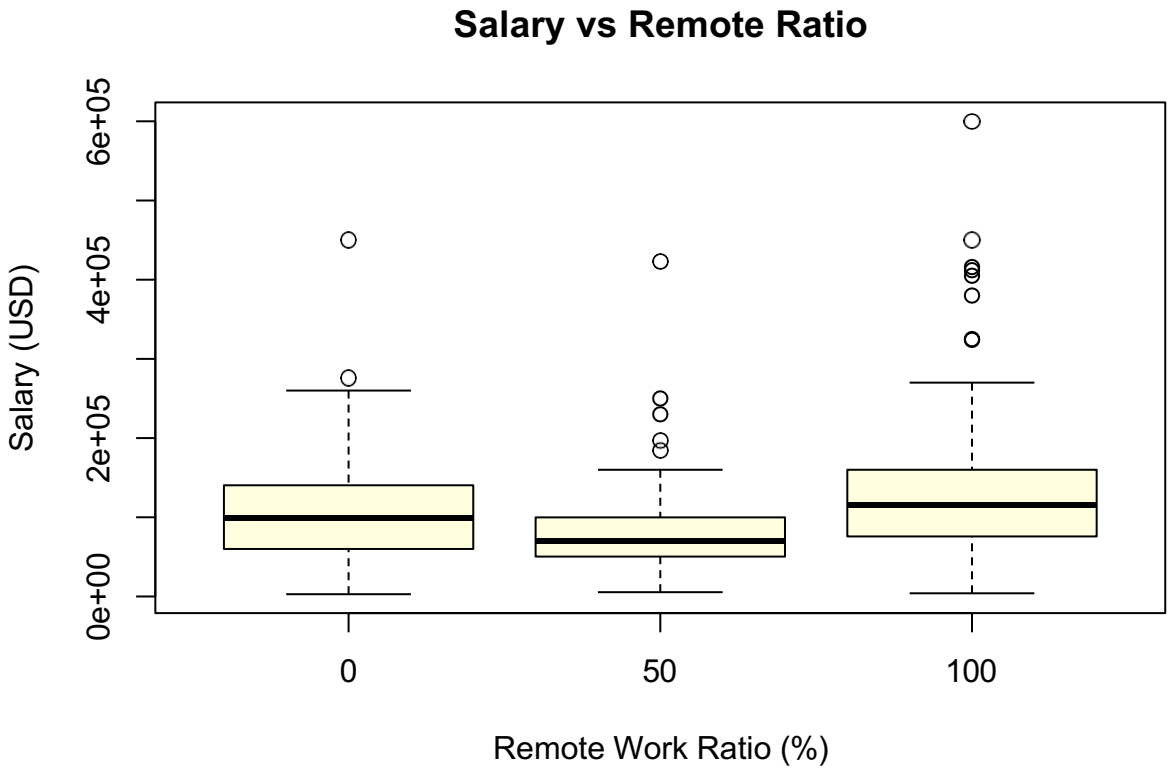
Reason: Highlights differences in pay trends across small, medium, and large companies.

```
boxplot(salary_in_usd ~ company_size, data = first_dataframe,
        main = "Salary by Company Size",
        xlab = "Company Size",
        ylab = "Salary (USD)",
        col = "lightpink")
```

4. Remote Ratio vs Salary (Boxplot)

Reason: Shows how compensation differs between on-site, hybrid, and fully remote roles.

```
boxplot(salary_in_usd ~ remote_ratio, data = first_dataframe,
        main = "Salary vs Remote Ratio",
        xlab = "Remote Work Ratio (%)",
        ylab = "Salary (USD)",
        col = "lightyellow")
```



Findings & Insights

The visualizations show that salaries in the dataset are right-skewed, with most professionals earning around the median while a few high-paying roles drive up the average. Pay clearly increases with experience, moving from entry- to executive-level positions. Larger companies tend to offer higher compensation than small or mid-sized ones, reflecting resource differences. Finally, remote roles show comparable or slightly higher pay, suggesting flexibility is increasingly valued without reducing earnings potential.

References

R Core Team (2025). *R: A Language and Environment for Statistical Computing*.
Xie, Y. (2023). *knitr: A General-Purpose Package for Dynamic Report Generation in R.*
All code written by the author.

END