R Programming Exercises

---

1. a.) Write R program to find R-Mean, Median & Mode with the sample data.

Program:

```
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
result.mean <- mean(x)
print(result.mean)

x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
result.mean <- mean(x,trim = 0.3)
print(result.mean)

 x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)
 result.mean <- mean(x)
 print(result.mean)
 result.mean <- mean(x,na.rm = TRUE)
 print(result.mean)

 x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
 median.result <- median(x)
 print(median.result)

 getmode <- function(v) {
   uniqv <- unique(v)
   uniqv[which.max(tabulate(match(v, uniqv)))]
 }
```

```
v <- c(2,1,2,3,1,2,3,4,1,5,5,3,2,3)
result <- getmode(v)
print(result)
charv <- c("o","it","the","it","it")
result <- getmode(charv)
print(result)
```

Output:

```
x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
> result.mean <- mean(x)
> print(result.mean)
[1] 8.22
>
> x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
> result.mean <- mean(x,trim = 0.3)
> print(result.mean)
[1] 5.55
>
>  x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)
>  result.mean <- mean(x)
>  print(result.mean)
[1] NA
>  result.mean <- mean(x,na.rm = TRUE)
>  print(result.mean)
[1] 8.22
>
>  x <- c(12,7,3,4.2,18,2,54,-21,8,-5)
>  median.result <- median(x)
>  print(median.result)
[1] 5.6
>
>  getmode <- function(v) {
```

```
+    uniqv <- unique(v)
+    uniqv[which.max(tabulate(match(v, uniqv)))]
+  }
>
>  v <- c(2,1,2,3,1,2,3,4,1,5,5,3,2,3)
>  result <- getmode(v)
>  print(result)
[1] 2
>  charv <- c("o","it","the","it","it")
>  result <- getmode(charv)
>  print(result)
[1] "it"
```

Explanation:

Breaking down the getmode function line by line:

```
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}
```

Line 1: getmode <- function(v) {

- This **defines a new function** named getmode that takes one argument v.
- v is expected to be a vector (numeric, character, or other types).
- The function will return the **mode** — the most frequently occurring value in v.

Line 2: uniqv <- unique(v)

- unique(v) returns a vector of the **unique values** present in v — that is, no duplicates.
- These unique values are stored in the variable uniqv.
- For example, if v = c(2, 2, 3, 1, 3), then uniqv = c(2, 3, 1).

Line 3: uniqv[which.max(tabulate(match(v, uniqv)))]

This is the key line where the mode is found, let's break it down from the inside out:

- match(v, uniqv):
    - This returns a vector of the **positions** of each element of v in uniqv.

For example, if v = c(2, 2, 3, 1, 3) and uniqv = c(2, 3, 1), then:

```
 match(v, uniqv)
# returns: c(1, 1, 2, 3, 2)
```

- tabulate(match(v, uniqv)):
    - tabulate counts the **frequency** of each integer in the vector passed to it.

Given the previous example vector c(1,1,2,3,2), it counts how many times 1, 2, and 3 occur:

```
 tabulate(c(1,1,2,3,2))
# returns: c(2, 2, 1)
```

    - So here, the first unique value (2) appears 2 times, the second unique value (3) appears 2 times, and the third unique value (1) appears 1 time.
- which.max(tabulate(...)):
    - which.max returns the **index** of the maximum value in the frequency counts.
    - For the example above, which.max(c(2, 2, 1)) would return 1 (the first maximum).

    - So, it returns the index of the most frequent value in uniqv.
- Finally, uniqv[...]:
    - This selects the unique value corresponding to that index.
    - So it returns the **mode** of v.

---

1. b) Write R program to find Analysis and Covariance with the sample data and visualize the regression graphically.

Program:

```
input <- mtcars[,c("am","mpg","hp")]
print(head(input))
# Get the dataset.
input <- mtcars
# Create the regression model.
result <- aov(mpg~hp*am,data = input)
print(summary(result))
# Create the regression model.
result <- aov(mpg~hp+am,data = input)
print(summary(result))
# Create the regression models.
result1 <- aov(mpg~hp*am,data = input)
result2 <- aov(mpg~hp+am,data = input)
# Compare the two models.
print(anova(result1,result2))

install.packages("ggplot2", type = "binary")
library(ggplot2)
```

```
ggplot(input, aes(x = hp, y = mpg, color = factor(am))) +
  geom_point(size = 3) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "ANCOVA: mpg ~ hp + am",
       x = "Horsepower",
       y = "Miles per Gallon",
       color = "Transmission (0 = Auto, 1 = Manual)")
mtcars
```

Output:

```
 input <- mtcars[,c("am","mpg","hp")]
> print(head(input))
                  am  mpg  hp
Mazda RX4          1 21.0 110
Mazda RX4 Wag      1 21.0 110
Datsun 710         1 22.8  93
Hornet 4 Drive     0 21.4 110
Hornet Sportabout  0 18.7 175
Valiant            0 18.1 105
> # Get the dataset.
> input <- mtcars
> # Create the regression model.
> result <- aov(mpg~hp*am,data = input)
> print(summary(result))
            Df Sum Sq Mean Sq F value   Pr(>F)
hp           1  678.4   678.4  77.391 1.50e-09 ***
am           1  202.2   202.2  23.072 4.75e-05 ***
hp:am        1    0.0     0.0   0.001    0.981
Residuals   28  245.4     8.8
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> # Create the regression model.
```

```
> result <- aov(mpg~hp+am,data = input)
> print(summary(result))
            Df Sum Sq Mean Sq F value   Pr(>F)
hp           1  678.4   678.4   80.15 7.63e-10 ***
am           1  202.2   202.2   23.89 3.46e-05 ***
Residuals   29  245.4     8.5
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> # Create the regression models.
> result1 <- aov(mpg~hp*am,data = input)
> result2 <- aov(mpg~hp+am,data = input)
> # Compare the two models.
> print(anova(result1,result2))
Analysis of Variance Table

Model 1: mpg ~ hp * am
Model 2: mpg ~ hp + am
  Res.Df    RSS Df  Sum of Sq      F Pr(>F)
1     28 245.43
2     29 245.44 -1 -0.0052515 6e-04 0.9806
>
> install.packages("ggplot2", type = "binary")
WARNING: Rtools is required to build R packages but is not currently installed.
Please download and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into ‘C:/Users/student/Documents/R/win-library/4.1’
(as ‘lib’ is unspecified)

  There is a binary version available (and will be
  installed) but the source version is later:
       binary source
ggplot2  3.4.2  4.0.0
```

also installing the dependencies ‘utf8’, ‘fansi’, ‘magrittr’, ‘pillar’, ‘pkgconfig’, ‘tibble’

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/utf8_1.2.3.zip'
Content type 'application/zip' length 210162 bytes (205 KB)
downloaded 205 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/fansi_1.0.4.zip'
Content type 'application/zip' length 366039 bytes (357 KB)
downloaded 357 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/magrittr_2.0.3.zip'
Content type 'application/zip' length 238677 bytes (233 KB)
downloaded 233 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/pillar_1.9.0.zip'
Content type 'application/zip' length 658514 bytes (643 KB)
downloaded 643 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/pkgconfig_2.0.3.zip'
Content type 'application/zip' length 22488 bytes (21 KB)
downloaded 21 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/tibble_3.2.1.zip'
Content type 'application/zip' length 702025 bytes (685 KB)
downloaded 685 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/ggplot2_3.4.2.zip'
Content type 'application/zip' length 4295743 bytes (4.1 MB)
downloaded 4.1 MB

package ‘utf8’ successfully unpacked and MD5 sums checked
package ‘fansi’ successfully unpacked and MD5 sums checked
package ‘magrittr’ successfully unpacked and MD5 sums checked
package ‘pillar’ successfully unpacked and MD5 sums checked

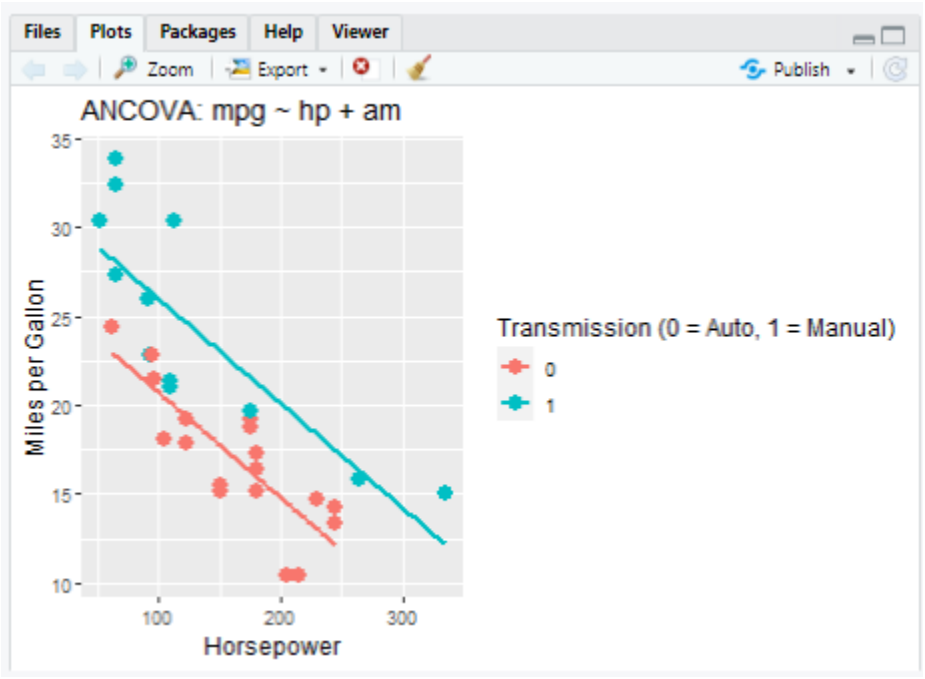
```
package ‘pkgconfig’ successfully unpacked and MD5 sums checked
package ‘tibble’ successfully unpacked and MD5 sums checked
package ‘ggplot2’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\student\AppData\Local\Temp\RtmpgXDUUf\downloaded_packages
> library(ggplot2)
Warning message:
package ‘ggplot2’ was built under R version 4.1.3
>
> ggplot(input, aes(x = hp, y = mpg, color = factor(am))) +
+   geom_point(size = 3) +
+   geom_smooth(method = "lm", se = FALSE) +
+   labs(title = "ANCOVA: mpg ~ hp + am",
+        x = "Horsepower",
+        y = "Miles per Gallon",
+        color = "Transmission (0 = Auto, 1 = Manual)")
`geom_smooth()` using formula = 'y ~ x'
> mtcars
                   mpg cyl  disp  hp drat    wt  qsec vs
Mazda RX4          21.0   6 160.0 110 3.90 2.620 16.46  0
Mazda RX4 Wag      21.0   6 160.0 110 3.90 2.875 17.02  0
Datsun 710         22.8   4 108.0  93 3.85 2.320 18.61  1
Hornet 4 Drive     21.4   6 258.0 110 3.08 3.215 19.44  1
Hornet Sportabout  18.7   8 360.0 175 3.15 3.440 17.02  0
Valiant            18.1   6 225.0 105 2.76 3.460 20.22  1
Duster 360         14.3   8 360.0 245 3.21 3.570 15.84  0
Merc 240D          24.4   4 146.7  62 3.69 3.190 20.00  1
Merc 230           22.8   4 140.8  95 3.92 3.150 22.90  1
Merc 280           19.2   6 167.6 123 3.92 3.440 18.30  1
Merc 280C          17.8   6 167.6 123 3.92 3.440 18.90  1
Merc 450SE         16.4   8 275.8 180 3.07 4.070 17.40  0
Merc 450SL         17.3   8 275.8 180 3.07 3.730 17.60  0
Merc 450SLC        15.2   8 275.8 180 3.07 3.780 18.00  0
```

Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0		
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0		
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0		
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1		
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1		
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1		
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1		
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0		
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0		
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0		
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0		
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1		
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0		
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1		
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0		
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0		
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0		
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1		

	am	gear	carb
Mazda RX4	1	4	4
Mazda RX4 Wag	1	4	4
Datsun 710	1	4	1
Hornet 4 Drive	0	3	1
Hornet Sportabout	0	3	2
Valiant	0	3	1
Duster 360	0	3	4
Merc 240D	0	4	2
Merc 230	0	4	2
Merc 280	0	4	4
Merc 280C	0	4	4
Merc 450SE	0	3	3
Merc 450SL	0	3	3
Merc 450SLC	0	3	3
Cadillac Fleetwood	0	3	4
Lincoln Continental	0	3	4

Chrysler Imperial 0 3 4
Fiat 128 1 4 1
Honda Civic 1 4 2
Toyota Corolla 1 4 1
Toyota Corona 0 3 1
Dodge Challenger 0 3 2
AMC Javelin 0 3 2
Camaro Z28 0 3 4
Pontiac Firebird 0 3 2
Fiat X1-9 1 4 1
Porsche 914-2 1 5 2
Lotus Europa 1 5 2
Ford Pantera L 1 5 4
Ferrari Dino 1 5 6
Maserati Bora 1 5 8
Volvo 142E 1 4 2
> \

Graph Plotted:



Explanation:

"Am","mpg","hp" columns from MTcars dataset is taken as the input - Head means first 6 - so, first 6 rows are printed as the output

After that, entire dataset is taken as the input

Step-by-step explanation for

*result <- aov(mpg ~ hp * am, data = input)*
print(summary(result))

1. `aov()` → This is used in R to do an **Analysis of Variance (ANOVA)**.
   It helps check if certain factors (independent variables) have a significant effect on another variable (dependent variable).
2. `mpg ~ hp * am` → This is the formula.
   - `mpg` = the dependent variable (miles per gallon).
   - `hp` = independent variable (horsepower).
   - `am` = independent variable (transmission type, usually 0 = automatic, 1 = manual).
   - `*` = means "consider both hp, am, and their interaction (hp:am)".
3. So, this model asks:
   - Does **horsepower** affect mpg?
   - Does **transmission type** affect mpg?
   - Does the **combination of horsepower and transmission type** affect mpg?
4. `data = input` → Tells R that the variables come from the dataset named `input`.
5. `summary(result)` → Prints the ANOVA table, showing which factors (hp, am, hp:am) significantly affect mpg.

Step-by-step explanation for

result <- aov(mpg ~ hp + am, data = input)

print(summary(result))

1. **`aov()`** → Runs **Analysis of Variance (ANOVA)**.
2. **`mpg ~ hp + am`** → This is the formula.
   - `mpg` = dependent variable (miles per gallon).
   - `hp` = independent variable (horsepower).
   - `am` = independent variable (transmission type).
   - `+` = means "consider hp and am **separately**, but not their interaction."
3. So, this model asks:
   - Does **horsepower** affect mpg?
   - Does **transmission type** affect mpg?
     (But it does **not** check if the *combination* of hp and am together has an effect.)
4. **`data = input`** → Uses the dataset `input`.
5. **`summary(result)`** → Prints the ANOVA table, telling you whether hp and am are statistically significant in explaining mpg.

👉 Difference from the earlier one (`hp * am`):

- `hp * am` → checks **hp, am, and their interaction together**.
- `hp + am` → checks only **hp and am separately**.

Step-by-step explanation:

print(anova(result1, result2))

1. **`result1` and `result2`** → These are two models you created earlier (for example, one with `hp + am` and one with `hp * am`).
2. **`anova(result1, result2)`** → Compares the two models using an ANOVA test.

It checks if the **bigger model** (the one with more terms, like including the interaction `hp:am`) is actually a **significant improvement** over the smaller model.

3. **`print()`** → Just displays the results.

In very simple words:
This code asks: *"Does adding extra terms (like the interaction between horsepower and transmission) really improve the model, or is the simpler model good enough?"*

If the p-value is **small (usually < 0.05)** → the bigger model is better.
If the p-value is **large** → the extra terms don't add much, so the simpler model is fine.

Here, **p = 0.9806**, which is **very large** (almost 1).

In simple words:

The interaction term (`hp:am`) **does not improve the model at all**.

So, the simpler model (`mpg ~ hp + am`) is just as good as the bigger one.

Explanation for graph plotting

What this code does overall: it installs/loads the `ggplot2` plotting package and then draws a scatterplot of `mpg` vs `hp`, colors points by transmission (`am`), and adds linear regression lines (one per transmission group) — a simple visual ANCOVA.

install.packages("ggplot2", type = "binary")

- Installs the `ggplot2` package from CRAN.
- `type = "binary"` tells R to prefer a precompiled binary package (useful on Windows so you don't need compilation tools like Rtools).
- **Caveat:** if a binary for your R version is not available, R will try to install from source and then may require Rtools (or the install can fail). If install fails, try `install.packages("ggplot2")` (or install Rtools on Windows).

library(ggplot2)

- Loads the `ggplot2` package into the session so you can use `ggplot()` and its geoms.
- If this errors, it means `ggplot2` is not installed or failed to load.

ggplot(input, aes(x = hp, y = mpg, color = factor(am))) +

- `ggplot(input, ...)` starts a ggplot using the data frame `input` (here `input` should have columns `hp`, `mpg`, and `am`, e.g. `mtcars`).
- `aes(...)` defines aesthetic mappings:
  - `x = hp` → horsepower on the x-axis.
  - `y = mpg` → miles per gallon on the y-axis.
    `color = factor(am)` → color the points (and lines) by `am`, but `factor(am)` **converts the numeric `am` to a categorical factor** so ggplot treats it as discrete groups (usually 0 = auto, 1 = manual).
- The `+` chains layers onto the plot.

geom_point(size = 3) +

- `geom_point()` adds the scatterplot points.
- `size = 3` controls the point size (larger number → bigger points). This is a layer-level aesthetic not mapped to data (constant size for all points).

geom_smooth(method = "lm", se = FALSE) +

- `geom_smooth()` adds smoothed lines (regression lines here).
- `method = "lm"` fits **linear models** (`y ~ x`) and draws the fitted lines.
- Because you mapped `color = factor(am)` in the main `ggplot(...)` call, `geom_smooth()` will automatically fit **one linear model per `am` group** and draw a separate line for each group. That gives you the group-wise regression lines needed for visual ANCOVA.
- `se = FALSE` disables the shaded confidence interval around each fitted line. Set `se = TRUE` if you want the confidence bands shown.

labs(title = "ANCOVA: mpg ~ hp + am",

x = "Horsepower",

y = "Miles per Gallon",

color = "Transmission (0 = Auto, 1 = Manual)")

- `labs()` sets human-readable labels:
  - `title` gives the plot title.
  - `x` and `y` set axis labels.
  - `color = "..."` sets the **legend title** for the color mapping (so instead of showing `factor(am)` as the legend heading, it shows your descriptive text).
