R Programming Basics :

Topics :

Basic Functions : Code #1
Plots : Code #2
Data Manipulation : Code #3 (mutation, summarize, group_by, filter, ungroup)
Values Imputation : Code #4
Mock Questions : Code #5
GGPlot Dummy Code : Code #56

#1 Some Basics Functions :

Function	Description
`abs(x)`	Absolute value
`sqrt(x)`	Square root
`log(x)`	Natural logarithm (base e)
`log10(x)`	Logarithm with base 10
`exp(x)`	Exponential funciton e^x
`cos(x), sin(x), ...`	Trigonometric functions
`ceiling(x)`	Round up: `ceiling(6.475)` is 7
`floor(x)`	ound down: `floor(6.489)` is 6
`trunc(x)`	Cut decimals: `trunc(2.99)` is 2
`round(x, digits=n)`	Regular rounding: `round(7.657, 2)` yields 7.67

Statistical Functions :

Function	Description
`mean(x, na.rm=FALSE)`	Arithmetic mean of object x
`sd(x)`	Standard deviation of object x
`sd(x)`	Variance of object x
`mad(x)`	Median absolute deviation of values in x
`median(x)`	Median value of object x
`quantile(x, probs)`	Quantiles where x is the numeric vector whose quantiles are desired and probs is a numeric vector with probabilities in $[0, 1]$
`range(x)`	Range
`sum(x)`	Sum
`min(x)`	Minimum
`max(x)`	Maximum

Function	Description
`sort(x)`	Sort elements
`order(x)`	Indices of elements in sorted order
`unique(x)`	Vector of unique elements (removes duplicates)
`duplicated(x)`	Which elements of x are duplicates?
`which.min(x)`	Index of smallest element
`which.max(x)`	Index of largest element
`which(x)`	Indices of elements in x which are TRUE

Dataframes

Data frames are lists where every components has the same length:

```
> x = data.frame(
+   id = 1:4,
+   name = c("Max", "Sophie", "Jack", "Ted"),
+   grade = c(5.0, 5.0, 4.0, 5.0))
> x
```

```
> sprintf("x = %i, y = %f", x, y) # %f for fixed point decimal notation [-]mmm.ddd
## [1] "x = 10, y = 19.453533"
```

#2 Plots Basics :

Bar Plot :

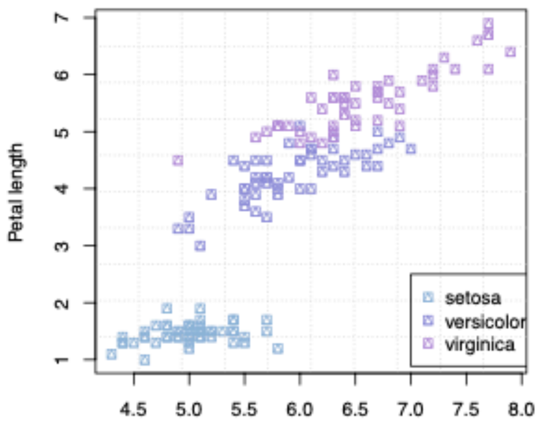
```
> tab = table(mtcars$cyl)
> barplot(tab, cex.axis = 0.7, main = "Distribution of the nr. of cylinders")
```

Scatter Plot :

plot(feature_1, feature_2, xlab = "Sepal length", ylab = "Petal length")

Scatter plot and adding different colours to different species / types

```
> cols = c("#8DB3D9", "#8D8DD9", "#B38DD9") # colors in RGB format
> plot(iris$Sepal.Length, iris$Petal.Length,
+   xlab = "Sepal length", ylab = "Petal length",
+   col = cols[iris$Species], pch = 14) # color by species (factor)
> grid(nx = 10, ny = 10, col = "lightgray", lty = "dotted", equilogs = TRUE)
> legend(7, 2.5, legend = c("setosa","versicolor","virginica"),
+   col = cols, pch = 14)
```



Saving Plots : Panda ULEOR-03 handout : Page 66/110

#3 Data Manipulation : Tidyverse

Data Mutation, adds a new column with the provided condition.

mutate(diamonds, ratio = (price*carat)/depth, excellent = (cut >= "Premium") & (color == "E"))

This will add new columns in the database.

Summarize() : summarize the whole column into a single value.

summarize(diamonds, avg_price = mean(price))
Or,

```r
df_summary <- df %>%
  group_by(y) %>%
  summarize(mean_x = mean(x), .groups = "drop")
```

group_by() :

filter() : Filter out rows with some specific values

Temp = filter(diamonds, clarity %in% c(“I1”, “SI2”, “IF”))

```r
library(dplyr)
ds <- diamonds %>% filter(color %in% c("F", "G", "H", "I"))
```

This does the same thing as:

```r
ds = subset(diamonds, color %in% c("F", "G", "H", "I"))
```

arrange() : helps to sort the data.
 Example usage :
arrange(cut , desc(price_mean)) , this will sort the data frame so that it is first ordered in ascending order by the column cut,
Then within each cur group, it the orders the rows in descending order by price_mean.

unite() and seprate():

x = diamonds %>% unite(index, cut, color, sep = "-") , will unite cut and colour into same column index with - in between

x %>% separate(index, into = c("cut", "color"), sep = "-") , separates index to cut and colour from -

Sorting Data :

df_sorted <- df %>% arrange(desc(x))

#4 Imputation Techniques

Imputation Techniques

Mean Imputation: Replace NA with the mean (ignoring NAs).

r Copy

```
df_imputed <- df %>% mutate(x = ifelse(is.na(x), mean(x, na.rm = TRUE), x))
```

MEAN IMPUTATION :

r Copy

```
library(dplyr)
# Suppose df is your data frame and 'var' is the variable with missing values.
df_imputed <- df %>%
  mutate(var = ifelse(is.na(var), mean(var, na.rm = TRUE), var))
```

REGRESSION IMPUTATION

```
# Suppose df is your data frame, and you want to impute missing values in 'y' usi
# Fit a linear model on complete cases.
model <- lm(y ~ x, data = df, na.action = na.exclude)

# Predict missing values for 'y'
missing_idx <- which(is.na(df$y))
df$y[missing_idx] <- predict(model, newdata = df[missing_idx, ])
```

#5 Mock Questions

Mock Question

The `mpg` dataset contains fuel economy data for 234 cars. Using *tidyverse* tools, perform the following steps:

1. **Subset** the data so it only includes cars made in **2008 or later** (i.e., `year >= 2008`).
2. **Group** the resulting data by **manufacturer** and **drive type** (`drv`).
3. **Summarize** by computing two values per group: the **average highway mpg** (`mean_hwy`) and the **maximum city mpg** (`max_cty`).
4. **Arrange** the summarized table in **descending order** of the average highway mpg.
5. **Create** a bar plot (using **ggplot2**) that shows the average highway mpg on the y-axis, manufacturer on the x-axis, and uses fill color to distinguish the drive type (`drv`). Add a suitable title.

```
# 1) Load Packages and Data
library(dplyr)    # For data manipulation
library(ggplot2)  # For the mpg dataset and for plotting

# The 'mpg' dataset comes automatically with ggplot2
data(mpg)

# 2) Subset the data to include only rows where year >= 2008
mpg_filtered <- mpg %>%
  filter(year >= 2008)

# 3) Group by manufacturer and drv (drive type), then summarize
mpg_summary <- mpg_filtered %>%
  group_by(manufacturer, drv) %>%
  summarize(
    mean_hwy = mean(hwy),  # average highway mpg
    max_cty  = max(cty),   # maximum city mpg
    .groups  = "drop"      # ungroup after summarizing
  )

# 4) Arrange the summarized table in descending order of mean_hwy
mpg_arranged <- mpg_summary %>%
  arrange(desc(mean_hwy))

# Print to check the table
print(mpg_arranged)

# 5) Create a bar plot of mean_hwy vs. manufacturer, colored by drive type
ggplot(mpg_arranged, aes(x = manufacturer, y = mean_hwy, fill = drv)) +
  geom_col(position = "dodge") +        # bar plot with side-by-side bars
  labs(
```

```
    title = "Average Highway MPG by Manufacturer and Drive Type (Year >=
2008)",
    x     = "Manufacturer",
    y     = "Mean Highway MPG"
  ) +
  theme_minimal() +
  coord_flip()  # optional: flip coordinates for readability
```

Mock Question

The `msleep` dataset contains information on the sleep habits of different mammals, including their taxonomic order, sleep duration, and various physiological attributes. Using **tidyverse** tools, do the following:

1. **Filter** the data to keep only mammals whose **body weight** (`bodywt`) is less than **50 kg.**
2. **Group** the resulting data by **taxonomic order** (`order`).
3. **Summarize** each group by computing:
    - The **average** total sleep time (`mean_sleep = mean(sleep_total)`)
    - The **maximum** REM sleep time (`max_rem = max(sleep_rem)`)
4. **Arrange** the summary in **ascending order** of the average total sleep (`mean_sleep`).
5. **Create a scatter plot** of `mean_sleep` (x-axis) vs. `max_rem` (y-axis), labeling points by their taxonomic order. Add an appropriate title.

```
# 1) Load Packages and Data
library(dplyr)    # For data manipulation
library(ggplot2)  # msleep dataset is included with ggplot2

data(msleep)      # load the msleep dataset

# 2) Filter the data: keep only rows where bodywt < 50
msleep_filtered <- msleep %>%
  filter(bodywt < 50)

# 3) Group by 'order' and summarize
msleep_summary <- msleep_filtered %>%
  group_by(order) %>%
  summarize(
    mean_sleep = mean(sleep_total, na.rm = TRUE),  # average total sleep
    max_rem    = max(sleep_rem, na.rm = TRUE),     # maximum REM sleep
    .groups    = "drop"                            # ungroup after summarizing
  )

# 4) Arrange in ascending order of mean_sleep
```

```
msleep_arranged <- msleep_summary %>%
  arrange(mean_sleep)

# Print to check the table
print(msleep_arranged)

# 5) Create a scatter plot of mean_sleep vs max_rem, labeling by order
ggplot(msleep_arranged, aes(x = mean_sleep, y = max_rem, label = order)) +
  geom_point(color = "blue", size = 3) +
  geom_text(vjust = -1, size = 3) +  # add text labels above points
  labs(
    title = "Mean Total Sleep vs. Max REM Sleep (Bodywt < 50)",
    x = "Mean Total Sleep (hrs)",
    y = "Max REM Sleep (hrs)"
  ) +
  theme_minimal()
```

Question:

You are given the `mpg` dataset. Perform the following tasks using tidyverse:

1. Filter the dataset to include only vehicles with engine displacement (`displ`) less than 4.
2. Create a new variable `efficiency` defined as the ratio of city mileage (`cty`) to highway mileag (`hwy`).
3. Group the data by `manufacturer` and compute:
   - The average `efficiency` (name it `avg_efficiency`)
   - The total number of observations (`n_obs`)
4. Arrange the results in descending order of `avg_efficiency`.
5. Create a bar plot of `avg_efficiency` by `manufacturer` (bars colored by manufacturer).

```
library(dplyr)
library(ggplot2)

# Step 1: Filter the data
mpg_filtered <- mpg %>% filter(displ < 4)

# Step 2: Create the new variable
mpg_filtered <- mpg_filtered %>% mutate(efficiency = cty / hwy)

# Step 3: Group and summarize
mpg_summary <- mpg_filtered %>%
  group_by(manufacturer) %>%
  summarize(
    avg_efficiency = mean(efficiency, na.rm = TRUE),
    n_obs = n(),
    .groups = "drop"
  )

# Step 4: Arrange in descending order of avg_efficiency
mpg_summary <- mpg_summary %>% arrange(desc(avg_efficiency))
print(mpg_summary)

# Step 5: Create a bar plot
ggplot(mpg_summary, aes(x = reorder(manufacturer, -avg_efficiency), y = avg_efficienc
  geom_col() +
  labs(title = "Average Efficiency by Manufacturer (displ < 4)",
       x = "Manufacturer", y = "Average Efficiency") +
  theme_minimal()
```

Mock Question 2

Question:

Using the `airquality` dataset, perform the following:

1. Remove all rows with missing values.
2. Create a new variable `Temp_C` that converts the `Temp` variable from Fahrenheit to Celsius using the formula:
   $Temp\_C = (Temp - 32) \times \frac{5}{9}$.
3. Group the data by `Month` and calculate:
   - The average Ozone level (`avg_Ozone`)
   - The average `Temp_C` (`avg_Temp_C`)
4. Identify outliers in the `Ozone` variable using the 1.5 * IQR rule and remove them from the dataset.
5. Create a line plot showing the trend of average Ozone levels by Month after outlier removal.

Copy

```r
library(dplyr)
library(ggplot2)

# Step 1: Remove rows with missing values
airq_clean <- airquality %>% filter(complete.cases(.))

# Step 2: Convert temperature to Celsius
airq_clean <- airq_clean %>%
  mutate(Temp_C = (Temp - 32) * 5/9)

# Step 3: Group by Month and summarize
airq_summary <- airq_clean %>%
  group_by(Month) %>%
  summarize(
    avg_Ozone = mean(Ozone),
    avg_Temp_C = mean(Temp_C),
    .groups = "drop"
  )

# Step 4: Remove outliers from Ozone using the 1.5*IQR rule
iqr_ozone <- IQR(airq_clean$Ozone)
q1 <- quantile(airq_clean$Ozone, 0.25)
q3 <- quantile(airq_clean$Ozone, 0.75)
lower_bound <- q1 - 1.5 * iqr_ozone
upper_bound <- q3 + 1.5 * iqr_ozone

airq_no_out <- airq_clean %>% filter(Ozone >= lower_bound, Ozone <= upper_bound)
```

```r
# Recalculate monthly average Ozone after outlier removal
airq_summary_no_out <- airq_no_out %>%
  group_by(Month) %>%
  summarize(
    avg_Ozone = mean(Ozone),
    .groups = "drop"
  )

# Step 5: Create a line plot of average Ozone levels across months
ggplot(airq_summary_no_out, aes(x = Month, y = avg_Ozone)) +
  geom_line(color = "blue") +
  geom_point(color = "red", size = 3) +
  labs(title = "Trend of Average Ozone Levels by Month (Outliers Removed)",
       x = "Month", y = "Average Ozone Level") +
  theme_minimal()
```

Mock Question 1: Tidyverse Data Manipulation with mpg Dataset

Question:

Using the `mpg` dataset, perform the following tasks:

1. Filter the data to include only vehicles with highway mileage (`hwy`) greater than 25.
2. Create a new variable `efficiency_ratio` defined as the ratio of city mileage (`cty`) to highway mileage (`hwy`).
3. Group the data by `manufacturer` and calculate:
   - The average `efficiency_ratio` (name it `avg_efficiency`)
   - The total count of vehicles (`vehicle_count`)
4. Sort the results in descending order of `avg_efficiency`.
5. Create a bar plot showing `avg_efficiency` for each manufacturer.

```r
library(dplyr)
library(ggplot2)

# 1. Filter dataset: vehicles with hwy > 25
mpg_filtered <- mpg %>% filter(hwy > 25)

# 2. Create new variable: efficiency_ratio
mpg_filtered <- mpg_filtered %>% mutate(efficiency_ratio = cty / hwy)

# 3. Group by manufacturer and summarize
mpg_summary <- mpg_filtered %>%
  group_by(manufacturer) %>%
  summarize(
    avg_efficiency = mean(efficiency_ratio, na.rm = TRUE),
    vehicle_count = n(),
    .groups = "drop"
  )

# 4. Sort the summary table in descending order of avg_efficiency
mpg_summary <- mpg_summary %>% arrange(desc(avg_efficiency))
print(mpg_summary)

# 5. Create a bar plot of avg_efficiency by manufacturer
ggplot(mpg_summary, aes(x = reorder(manufacturer, -avg_efficiency), y = avg_efficienc
  geom_col() +
  labs(
    title = "Average Efficiency Ratio by Manufacturer",
    x = "Manufacturer",
    y = "Average Efficiency Ratio (cty/hwy)"
  ) +
  theme_minimal()
```

Question:

Using the `airquality` dataset, complete the following tasks:

1. Remove rows with any missing values.
2. Create a new variable `Temp_C` that converts the temperature (`Temp`) from Fahrenheit to Celsius using the formula $Temp\_C = (Temp - 32) \times \frac{5}{9}$.
3. Group the cleaned data by `Month` and compute the average `Ozone` level (name it `avg_Ozone`) and average `Temp_C` (name it `avg_Temp_C`).
4. Identify and remove outliers from the `Ozone` variable using the 1.5 * IQR rule.
5. Plot a line chart showing the trend of the average `Ozone` levels across months after outlier removal.

```r
library(dplyr)
library(ggplot2)

# 1. Remove rows with missing values
airq_clean <- airquality %>% filter(complete.cases(.))

# 2. Convert temperature to Celsius
airq_clean <- airq_clean %>%
  mutate(Temp_C = (Temp - 32) * 5/9)

# 3. Group by Month and summarize average Ozone and Temp_C
airq_summary <- airq_clean %>%
  group_by(Month) %>%
  summarize(
    avg_Ozone = mean(Ozone, na.rm = TRUE),
    avg_Temp_C = mean(Temp_C, na.rm = TRUE),
    .groups = "drop"
  )

# 4. Outlier removal for Ozone using 1.5*IQR rule
ozone_iqr <- IQR(airq_clean$Ozone)
ozone_q1 <- quantile(airq_clean$Ozone, 0.25)
ozone_q3 <- quantile(airq_clean$Ozone, 0.75)
lower_bound <- ozone_q1 - 1.5 * ozone_iqr
upper_bound <- ozone_q3 + 1.5 * ozone_iqr

airq_no_out <- airq_clean %>% filter(Ozone >= lower_bound, Ozone <= upper_bound)

# Recalculate monthly average Ozone after outlier removal
airq_summary_no_out <- airq_no_out %>%
  group_by(Month) %>%
  summarize(
    avg_Ozone = mean(Ozone, na.rm = TRUE),
    .groups = "drop"
  )
```

```r
# 5. Plot the trend of average Ozone levels by Month after outlier removal
ggplot(airq_summary_no_out, aes(x = Month, y = avg_Ozone)) +
  geom_line(color = "blue", size = 1) +
  geom_point(color = "red", size = 3) +
  labs(
    title = "Trend of Average Ozone Levels by Month (Outliers Removed)",
    x = "Month",
    y = "Average Ozone Level"
  ) +
  theme_minimal()
```

#6 ggplot()

Gg plot scatter plot :

```r
# Load ggplot2 package
library(ggplot2)

# Use a built-in dataset (mtcars) for demonstration
data(mtcars)

# Create a scatter plot: weight vs. miles per gallon
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(size = 3, color = "blue") +    # Scatter points
  labs(
    title = "Scatter Plot of MPG vs. Weight",
    x = "Weight (1000 lbs)",
    y = "Miles per Gallon"
  ) +
  theme_minimal()                          # Minimal theme for a clean look
```

code for copying

```
ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(size = 3, color = "blue") +    # Scatter points
  labs(
    title = "Scatter Plot of MPG vs. Weight",
    x = "Weight (1000 lbs)",
    y = "Miles per Gallon"
  ) +
  theme_minimal()
```

BAR PLOT :

2. Bar Plot (mtcars: count by number of cylinders)

r Copy

```r
library(ggplot2)
data(mtcars)
ggplot(mtcars, aes(x = factor(cyl))) +
  geom_bar(fill = "steelblue") +
  labs(
    title = "Count of Cars by Cylinder Number",
    x = "Number of Cylinders",
    y = "Count"
  ) +
  theme_minimal()
```

Line Plot :

3. Line Plot (airquality: Average Ozone by Month)

r Copy

```r
library(ggplot2)
library(dplyr)
data(airquality)

# Summarize mean Ozone per Month
aq_summary <- airquality %>%
  group_by(Month) %>%
  summarize(mean_Ozone = mean(Ozone, na.rm = TRUE), .groups = "drop")

ggplot(aq_summary, aes(x = Month, y = mean_Ozone)) +
  geom_line(color = "red", size = 1) +
  geom_point(color = "blue", size = 2) +
  labs(
    title = "Average Ozone Levels by Month",
    x = "Month",
    y = "Mean Ozone"
  ) +
  theme_minimal()
```

HISTOGRAM :

4. Histogram (mtcars: Distribution of MPG)

r

Copy

```r
library(ggplot2)
data(mtcars)
ggplot(mtcars, aes(x = mpg)) +
  geom_histogram(binwidth = 2, fill = "darkgreen", color = "black") +
  labs(
    title = "Histogram of Miles per Gallon",
    x = "MPG",
    y = "Count"
  ) +
  theme_minimal()
```

BOXPLOT :

. Boxplot (mtcars: MPG by Cylinder)

r

Copy

```r
library(ggplot2)
data(mtcars)
ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot(fill = "orange") +
  labs(
    title = "Boxplot of MPG by Number of Cylinders",
    x = "Number of Cylinders",
    y = "MPG"
  ) +
  theme_minimal()
```