import pandas as pd

import matplotlib.pyplot as plt

file ='/home/ex5/Downloads/Sample Datasets-20230926/sales2.csv'

df=pd.read\_csv(file)

print(df)

# Display the first few records

print(df.head())

# Describe the dataset

print(df.describe())

# Check for NA values

print(df.isna().sum())

# Plot a bar graph for ord\_date Vs Purchase amount

df.plot(x='ord\_date', y='purch\_amt',kind='bar')

plt.show()

# Fetch the details of customer id 3001

customer\_3001 = df[df['customer\_id'] == 3001]

print(customer\_3001)

#Update the purchase amount of order Id 70012 as 500

df.loc[df['ord\_no']==70012,'purch\_amt']=500

#Retrieve the order no and customer id of the maximum purchase amount

max\_purchase = df[df['purch\_amt'] == df['purch\_amt'].max()]

ord\_no = max\_purchase['ord\_no'].values[0]

customer\_id = max\_purchase['customer\_id'].values[0]

print(ord\_no, customer\_id)

# Retrieve the order no, customer id and purchase amount whose value is greater than 1000

high\_value\_orders = df[df['purch\_amt'] > 1000]

print(high\_value\_orders[['ord\_no', 'customer\_id', 'purch\_amt']])

# Fetch the order details that are placed during the month of October

october\_orders = df[df['ord\_date'].str.startswith('Oct')]

print(october\_orders)

# Split the dataset into groups on customer\_id to summarize purch\_amt with respect to each salesman id

grouped\_df = df.groupby('customer\_id')['purch\_amt'].sum()

print(grouped\_df)

# Calculate percentage of purch\_amt in each group and rename purch\_amt to purchaseAmount

grouped\_df = grouped\_df / grouped\_df.sum() \* 100

grouped\_df.rename("purchaseAmount", inplace=True)

print(grouped\_df)

import pandas as pd

import matplotlib.pyplot as plt

file ='/home/ex5/Downloads/Sample Datasets-20230926/iris.csv'

iris\_df=pd.read\_csv(file)

# Load the iris dataset

iris\_df = pd.read\_csv(file)

# Display the shape of the dataset

print("Shape of the dataset:", iris\_df.shape)

# Display a sample of 10 records

print("Sample of 10 records:")

print(iris\_df.head(10))

# Display records from the 10th row to the 20th row

print("Records from 10th row to 20th row:")

print(iris\_df[10:21])

# Drop duplicate records

iris\_df = iris\_df.drop\_duplicates()

# Count the number of unique species

unique\_species = iris\_df['species'].nunique()

print("Number of unique species:", unique\_species)

# Sort the data based on sepal length

iris\_df = iris\_df.sort\_values(by='Sepal.Length')

# Calculate sum, mean, and mode of petal length

petal\_length\_sum = iris\_df['petal\_length'].sum()

petal\_length\_mean = iris\_df['petal\_length'].mean()

petal\_length\_mode = iris\_df['petal\_length'].mode().values[0]

print("Sum of petal length:", petal\_length\_sum)

print("Mean of petal length:", petal\_length\_mean)

print("Mode of petal length:", petal\_length\_mode)

# Extract minimum and maximum value from sepal width

sepal\_width\_min = iris\_df['sepal\_width'].min()

sepal\_width\_max = iris\_df['sepal\_width'].max()

print("Minimum sepal width:", sepal\_width\_min)

print("Maximum sepal width:", sepal\_width\_max)

# Fetch the records with petal length > 4cm

petal\_length\_gt\_4 = iris\_df[iris\_df['petal\_length'] > 4.0]

print("Records with petal length > 4cm:")

print(petal\_length\_gt\_4)

# Split the data based on species type and find the mean sepal length

species\_mean\_sepal\_length = iris\_df.groupby('species')['sepal\_length'].mean()

print("Mean sepal length by species:")

print(species\_mean\_sepal\_length)

# Add a new attribute 'total' that stores the sum of sepal width and petal width

iris\_df['total'] = iris\_df['sepal\_width'] + iris\_df['petal\_width']

# Set the values of the rows 10 to 19 of the column 'petal\_length' to NaN

iris\_df.loc[10:19, 'petal\_length'] = float('NaN')

# Fill the NA values with its mean

iris\_df['petal\_length'].fillna(iris\_df['petal\_length'].mean(), inplace=True)

# Plot a histogram to show the species distribution

plt.figure(figsize=(8, 6))

iris\_df['species'].value\_counts().plot(kind='bar', color=['blue', 'green', 'red'])

plt.title("Species Distribution")

plt.xlabel("Species")

plt.ylabel("Count")

plt.xticks(rotation=0)

plt.show()

# File: DataFormats.R

# Course: R: An Introduction (with RStudio)

# DATA TYPES ###############################################

# Numeric

n1 <- 15 # Double precision by default

n1

typeof(n1)

n2 <- 1.5

n2

typeof(n2)

# Character

c1 <- "c"

c1

typeof(c1)

c2 <- "a string of text"

c2

typeof(c2)

# Logical

l1 <- TRUE

l1

typeof(l1)

l2 <- F

l2

typeof(l2)

# DATA STRUCTURES ##########################################

## Vector ##################################################

v1 <- c(1, 2, 3, 4, 5)

v1

is.vector(v1)

v2 <- c("a", "b", "c")

v2

is.vector(v2)

v3 <- c(TRUE, TRUE, FALSE, FALSE, TRUE)

v3

is.vector(v3)

## Matrix ##################################################

m1 <- matrix(c(T, T, F, F, T, F), nrow = 2)

m1

m2 <- matrix(c("a", "b",

"c", "d"),

nrow = 2,

byrow = T)

m2

## Array ###################################################

# Give data, then dimemensions (rows, columns, tables)

a1 <- array(c( 1:24), c(4, 3, 2))

a1

## Data frame ##############################################

# Can combine vectors of the same length

vNumeric <- c(1, 2, 3)

vCharacter <- c("a", "b", "c")

vLogical <- c(T, F, T)

dfa <- cbind(vNumeric, vCharacter, vLogical)

dfa # Matrix of one data type

df <- as.data.frame(cbind(vNumeric, vCharacter, vLogical))

df # Makes a data frame with three different data types

## List ####################################################

o1 <- c(1, 2, 3)

o2 <- c("a", "b", "c", "d")

o3 <- c(T, F, T, T, F)

list1 <- list(o1, o2, o3)

list1

list2 <- list(o1, o2, o3, list1) # Lists within lists!

list2

# COERCING TYPES ###########################################

## Automatic coercion ######################################

# Goes to "least restrictive" data type

(coerce1 <- c(1, "b", TRUE))

# coerce1 # Parenthese around command above make this moot

typeof(coerce1)

## Coerce numeric to integer ###############################

(coerce2 <- 5)

typeof(coerce2)

(coerce3 <- as.integer(5))

typeof(coerce3)

## Coerce character to numeric #############################

(coerce4 <- c("1", "2", "3"))

typeof(coerce4)

(coerce5 <- as.numeric(c("1", "2", "3")))

typeof(coerce5)

## Coerce matrix to data frame #############################

(coerce6 <- matrix(1:9, nrow= 3))

is.matrix(coerce6)

(coerce7 <- as.data.frame(matrix(1:9, nrow= 3)))

is.data.frame(coerce7)

# CLEAN UP #################################################

# Clear environment

rm(list = ls())

# Clear console

cat("\014") # ctrl+L

# Clear mind :)

—--------------------------------------------------------------------------------------------------------------

# File: Describe.R

# Course: R: An Introduction (with RStudio)

# INSTALL AND LOAD PACKAGES ################################

library(datasets) # Load base packages manually

# Installs pacman ("package manager") if needed

if (!require("pacman")) install.packages("pacman")

# Use pacman to load add-on packages as desired

pacman::p\_load(pacman, psych)

# LOAD DATA ################################################

head(iris)

# PSYCH PACKAGE ############################################

# Get info on package

p\_help(psych) # Opens package PDF in browser

p\_help(psych, web = F) # Opens help in R Viewer

# DESCRIBE() ###############################################

# For quantitative variables only.

describe(iris$Sepal.Length) # One quantitative variable

describe(iris) # Entire data frame

# CLEAN UP #################################################

# Clear environment

rm(list = ls())

# Clear packages

p\_unload(all) # Remove all add-ons

detach("package:datasets", unload = TRUE) # For base

# Clear console

cat("\014") # ctrl+L

# Clear mind :)

—---------------------------------------------------------------------------------------------

# File: BarCharts.R

# Course: R: An Introduction (with RStudio)

# LOAD DATASETS PACKAGES ###################################

library(datasets)

# LOAD DATA ################################################

?mtcars

head(mtcars)

# BAR CHARTS ###############################################

barplot(mtcars$cyl) # Doesn't work

# Need a table with frequencies for each category

cylinders <- table(mtcars$cyl) # Create table

barplot(cylinders) # Bar chart

plot(cylinders) # Default X-Y plot (lines)

# CLEAN UP #################################################

# Clear environment

rm(list = ls())

# Clear packages

detach("package:datasets", unload = TRUE) # For base

# Clear plots

dev.off() # But only if there IS a plot

# Clear console

cat("\014") # ctrl+L

# Clear mind :)

========================================================================

# File: OverlayingPlots.R

# Course: R: An Introduction (with RStudio)

# INSTALL AND LOAD PACKAGES ################################

library(datasets) # Load/unload base packages manually

# LOAD DATA ################################################

# Annual Canadian Lynx trappings 1821-1934

?lynx

head(lynx)

# HISTOGRAM ################################################

# Default

hist(lynx)

# Add some options

hist(lynx,

breaks = 14, # "Suggests" 14 bins

freq = FALSE, # Axis shows density, not freq.

col = "thistle1", # Color for histogram

main = paste("Histogram of Annual Canadian Lynx",

"Trappings, 1821-1934"),

xlab = "Number of Lynx Trapped")

# Add a normal distribution

curve(dnorm(x, mean = mean(lynx), sd = sd(lynx)),

col = "thistle4", # Color of curve

lwd = 2, # Line width of 2 pixels

add = TRUE) # Superimpose on previous graph

# Add two kernel density estimators

lines(density(lynx), col = "blue", lwd = 2)

lines(density(lynx, adjust = 3), col = "purple", lwd = 2)

# Add a rug plot

rug(lynx, lwd = 2, col = "gray")

# CLEAN UP #################################################

# Clear packages

detach("package:datasets", unload = TRUE) # For base

# Clear plots

dev.off() # But only if there IS a plot

# Clear console

cat("\014") # ctrl+L

# Clear mind :)

—----------------------------------------------------------------------------------------------------------------------------

# File: Summary.R

# Course: R: An Introduction (with RStudio)

# INSTALL AND LOAD PACKAGES ################################

library(datasets) # Load/unload base packages manually

# LOAD DATA ################################################

head(iris)

# SUMMARY() ################################################

summary(iris$Species) # Categorical variable

summary(iris$Sepal.Length) # Quantitative variable

summary(iris) # Entire data frame

# CLEAN UP #################################################

# Clear packages

detach("package:datasets", unload = TRUE) # For base

# Clear console

cat("\014") # ctrl+L

# Clear mind :)

—----------------------------------------------------------------------------------------------------------------------------

# File: Scatterplots.R

# Course: R: An Introduction (with RStudio)

# LOAD DATASETS PACKAGES ###################################

library(datasets) # Load/unload base packages manually

# LOAD DATA ################################################

?mtcars

head(mtcars)

# PLOTS ####################################################

# Good to first check univariate distributions

hist(mtcars$wt)

hist(mtcars$mpg)

# Basic X-Y plot for two quantitative variables

plot(mtcars$wt, mtcars$mpg)

# Add some options

plot(mtcars$wt, mtcars$mpg,

pch = 19, # Solid circle

cex = 1.5, # Make 150% size

col = "#cc0000", # Red

main = "MPG as a Function of Weight of Cars",

xlab = "Weight (in 1000 pounds)",

ylab = "MPG")

# CLEAN UP #################################################

# Clear packages

detach("package:datasets", unload = TRUE) # For base

# Clear plots

dev.off() # But only if there IS a plot

# Clear console

cat("\014") # ctrl+L

# Clear mind :)