Sri Lanka Institute of Information

Technology



Lab Submission

Lab sheet 05

**IT24104301**

**Karannagoda K. V. R.**

**Probability and Statistics | IT2120**

B.Sc. (Hons) in Information Technology

**Year 2 Semester 1 (2025)**

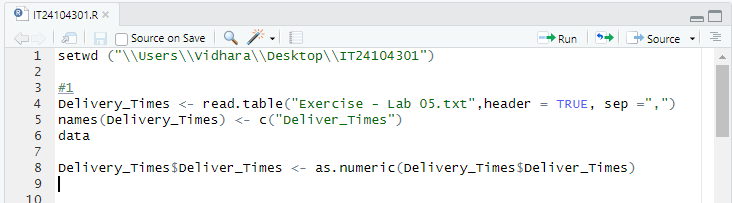
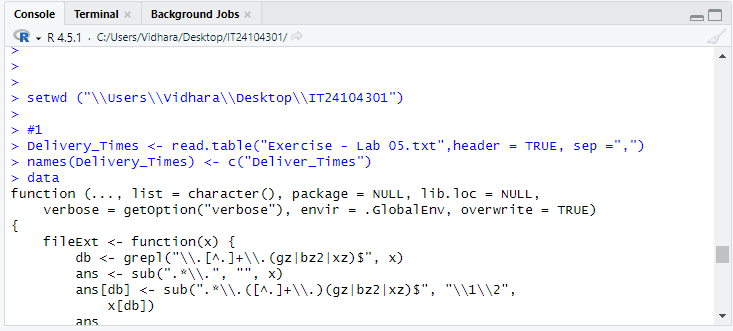


**Faculty of Computing**

IT2120 - Probability and Statistics Lab Sheet 05

**Exercise**

**Instructions:** Create a folder in your desktop with your registration number (Eg: ”IT.......”). You need to save the R script file and take screenshots of the command prompt with answers and save it in a word document inside the folder. Save both R script file and word document with your registration number (Eg: ”IT........”). After you finish the exercise, zip the folder and upload the zip file to the submission link.

1. Import the dataset (’Exercise – Lab 05.txt’) into R and store it in a data frame called ”Delivery Times”.

setwd ("\\Users\\Vidhara\\Desktop\\IT24104301")

>

> #1

> Delivery\_Times <- read.table("Exercise - Lab 05.txt",header = TRUE, sep =",")

> names(Delivery\_Times) <- c("Deliver\_Times")

> data

function (..., list = character(), package = NULL, lib.loc = NULL,

verbose = getOption("verbose"), envir = .GlobalEnv, overwrite = TRUE)

{

fileExt <- function(x) {

db <- grepl("\\.[^.]+\\.(gz|bz2|xz)$", x)

ans <- sub(".\*\\.", "", x)

ans[db] <- sub(".\*\\.([^.]+\\.)(gz|bz2|xz)$", "\\1\\2",

x[db])

ans

}

my\_read\_table <- function(...) {

lcc <- Sys.getlocale("LC\_COLLATE")

on.exit(Sys.setlocale("LC\_COLLATE", lcc))

Sys.setlocale("LC\_COLLATE", "C")

read.table(...)

}

stopifnot(is.character(list))

names <- c(as.character(substitute(list(...))[-1L]), list)

if (!is.null(package)) {

if (!is.character(package))

stop("'package' must be a character vector or NULL")

}

paths <- find.package(package, lib.loc, verbose = verbose)

if (is.null(lib.loc))

paths <- c(path.package(package, TRUE), if (!length(package)) getwd(),

paths)

paths <- unique(normalizePath(paths[file.exists(paths)]))

paths <- paths[dir.exists(file.path(paths, "data"))]

dataExts <- tools:::.make\_file\_exts("data")

if (length(names) == 0L) {

db <- matrix(character(), nrow = 0L, ncol = 4L)

for (path in paths) {

entries <- NULL

packageName <- if (file\_test("-f", file.path(path,

"DESCRIPTION")))

basename(path)

else "."

if (file\_test("-f", INDEX <- file.path(path, "Meta",

"data.rds"))) {

entries <- readRDS(INDEX)

}

else {

dataDir <- file.path(path, "data")

entries <- tools::list\_files\_with\_type(dataDir,

"data")

if (length(entries)) {

entries <- unique(tools::file\_path\_sans\_ext(basename(entries)))

entries <- cbind(entries, "")

}

}

if (NROW(entries)) {

if (is.matrix(entries) && ncol(entries) == 2L)

db <- rbind(db, cbind(packageName, dirname(path),

entries))

else warning(gettextf("data index for package %s is invalid and will be ignored",

sQuote(packageName)), domain = NA, call. = FALSE)

}

}

colnames(db) <- c("Package", "LibPath", "Item", "Title")

footer <- if (missing(package))

paste0("Use ", sQuote(paste("data(package =", ".packages(all.available = TRUE))")),

"\n", "to list the data sets in all \*available\* packages.")

else NULL

y <- list(title = "Data sets", header = NULL, results = db,

footer = footer)

class(y) <- "packageIQR"

return(y)

}

paths <- file.path(paths, "data")

for (name in names) {

found <- FALSE

for (p in paths) {

tmp\_env <- if (overwrite)

envir

else new.env()

if (file\_test("-f", file.path(p, "Rdata.rds"))) {

rds <- readRDS(file.path(p, "Rdata.rds"))

if (name %in% names(rds)) {

found <- TRUE

if (verbose)

message(sprintf("name=%s:\t found in Rdata.rds",

name), domain = NA)

objs <- rds[[name]]

lazyLoad(file.path(p, "Rdata"), envir = tmp\_env,

filter = function(x) x %in% objs)

break

}

else if (verbose)

message(sprintf("name=%s:\t NOT found in names() of Rdata.rds, i.e.,\n\t%s\n",

name, paste(names(rds), collapse = ",")),

domain = NA)

}

files <- list.files(p, full.names = TRUE)

files <- files[grep(name, files, fixed = TRUE)]

if (length(files) > 1L) {

o <- match(fileExt(files), dataExts, nomatch = 100L)

paths0 <- dirname(files)

paths0 <- factor(paths0, levels = unique(paths0))

files <- files[order(paths0, o)]

}

if (length(files)) {

for (file in files) {

if (verbose)

message("name=", name, ":\t file= ...", .Platform$file.sep,

basename(file), "::\t", appendLF = FALSE,

domain = NA)

ext <- fileExt(file)

if (basename(file) != paste0(name, ".", ext))

found <- FALSE

else {

found <- TRUE

switch(ext, R = , r = {

library("utils")

sys.source(file, chdir = TRUE, envir = tmp\_env)

}, RData = , rdata = , rda = load(file, envir = tmp\_env),

TXT = , txt = , tab = , tab.gz = , tab.bz2 = ,

tab.xz = , txt.gz = , txt.bz2 = , txt.xz = assign(name,

my\_read\_table(file, header = TRUE, as.is = FALSE),

envir = tmp\_env), CSV = , csv = , csv.gz = ,

csv.bz2 = , csv.xz = assign(name, my\_read\_table(file,

header = TRUE, sep = ";", as.is = FALSE),

envir = tmp\_env), found <- FALSE)

}

if (found)

break

}

if (verbose)

message(if (!found)

"\*NOT\* ", "found", domain = NA)

}

if (found)

break

}

if (!found) {

warning(gettextf("data set %s not found", sQuote(name)),

domain = NA)

}

else if (!overwrite) {

for (o in ls(envir = tmp\_env, all.names = TRUE)) {

if (exists(o, envir = envir, inherits = FALSE))

warning(gettextf("an object named %s already exists and will not be overwritten",

sQuote(o)))

else assign(o, get(o, envir = tmp\_env, inherits = FALSE),

envir = envir)

}

rm(tmp\_env)

}

}

invisible(names)

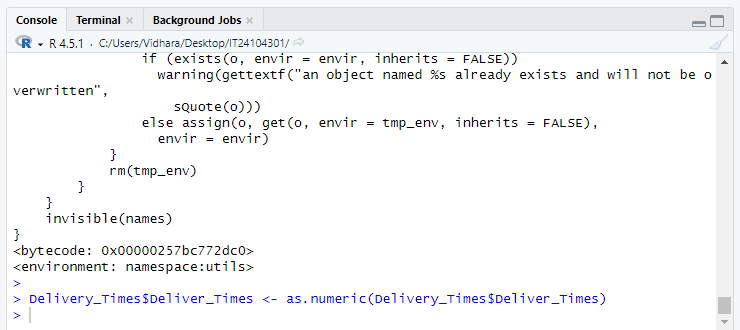
}

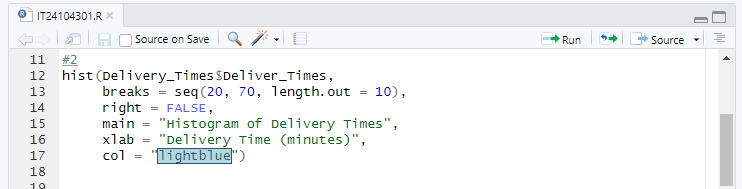
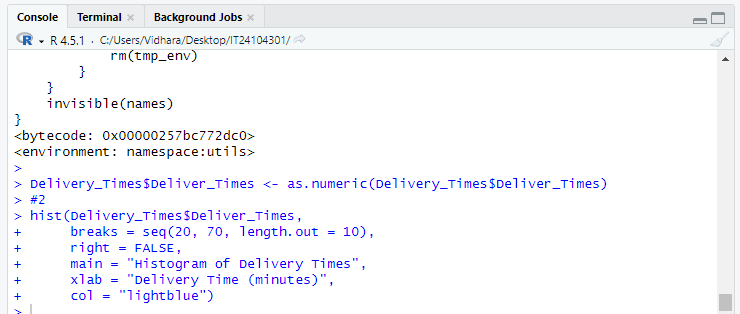
<bytecode: 0x00000257bc772dc0>

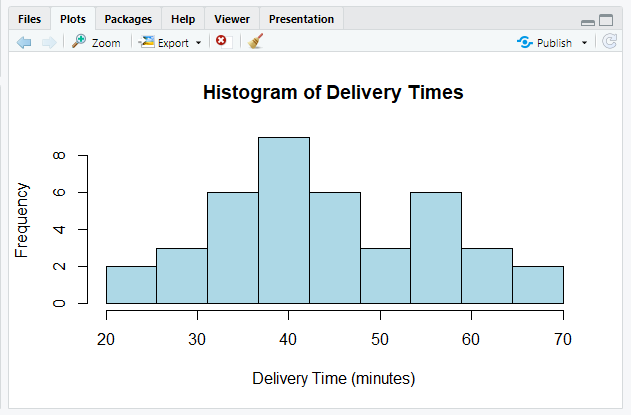
<environment: namespace:utils>

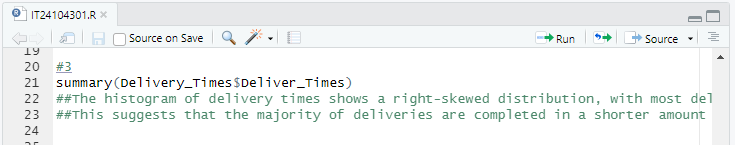
>

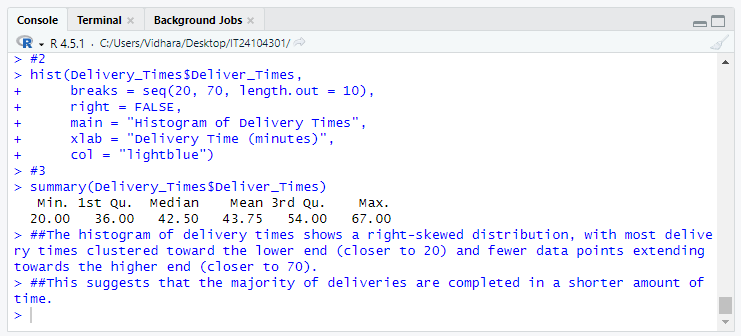
> Delivery\_Times$Deliver\_Times <- as.numeric(Delivery\_Times$Deliver\_Times)



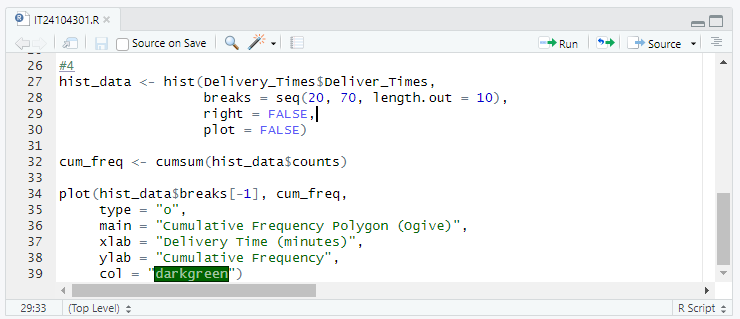
1. Draw a histogram for deliver times using nine class intervals where the lower limit is 20 and upper limit is 70. Use right open intervals.

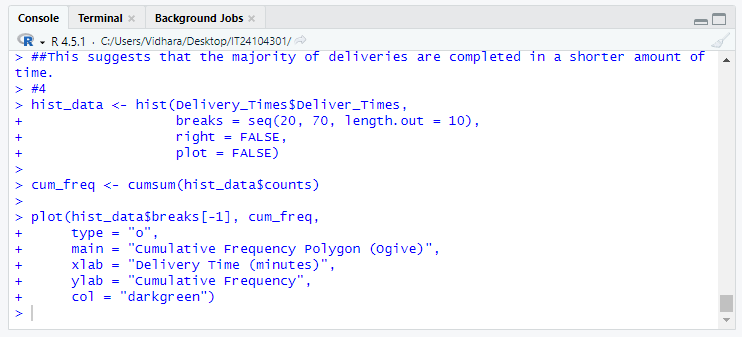


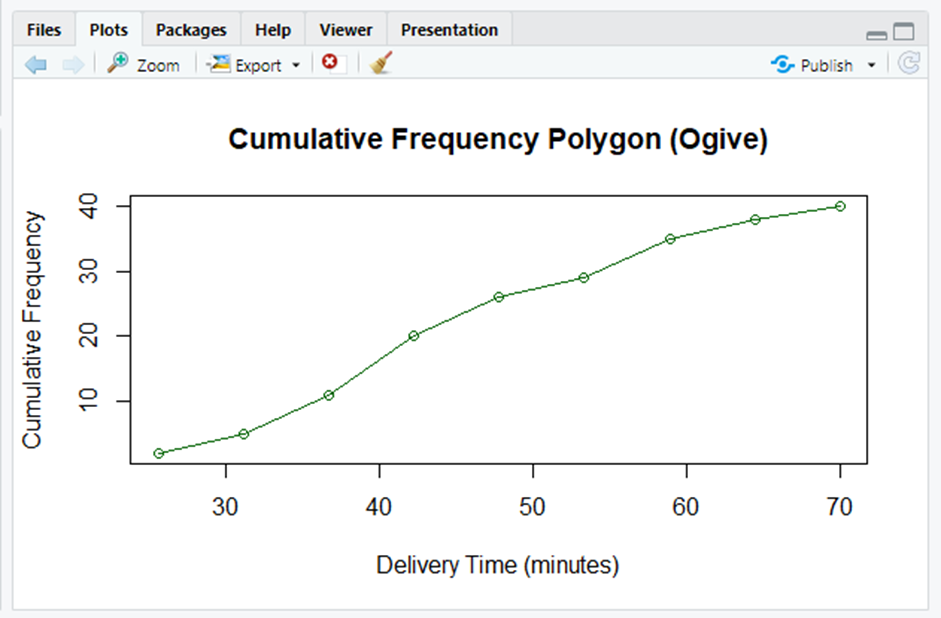
1. Comment on the shape of the distribution.



1. Draw a cumulative frequency polygon (ogive) for the data in a separate plot.

****

****

****