ESERCIZI INFORMATICA

➔ Esercizio 1

Contare il numero di occorrente del carattere T nella stringa string. Contare successivamente il numero di occorrenze della sottostringa GC nella stringa string

```
>sommaT <- 0
>string <- c("TSDJSGCGCTTGCTISWN")
>for (i in 1:nchar(string)) {
        if (substr (string, i, i) == "T"){
                sommaT <- sommaT + 1
        }
}
>sommaGC <- 0
>for (i in 1:nchar(string)-1) {
        if (substr (string, i, i+1) == "GC"){
                sommaGC <- sommaGC + 1
        }
}
```

➔ Esercizio 2

Scrivere uno script stats_iris.R che carichi il dataset Iris, e costruisca una lista contenente: una matrice 3x2, dove righe sono le specie (setosa, versicolor, virginica), le colonne la lunghezza e la larghezza media dei petali calcolate per ciascuna sottospecie separatamente; un vettore (named) che contenga il numero di valori compresi nel range interquartile per ciascuna colonna numerica del data frame.

```
>data(iris)
>specie1 <- iris[iris[,5] == "setosa"]
>specie1_len <- mean(specie1[,"Petal.Length"])
>specie1_wid <- mean(specie1[,"Petal.Width"])
>specie2 <- iris[iris[,5] == "versicolor"]
>specie2_len <- mean(specie2[,"Petal.Length"])
>specie2_wid <- mean(specie2[,"Petal.Width"])
```

```
>specie3 <- iris[iris[,5] == "verginica"]
>specie3_len <- mean(specie3[,"Petal.Length"])
>specie3_wid <- mean(specie3[,"Petal.Width"])
>matrice <- matrix(c(specie1_len, specie2_len, specie3_len, specie1_wid, specie2_wid, specie3_wid), nrow=3, ncol=2)
>rownames(matrice) <- c("setosa", "versicolor", "verginica")
>colnames(matrice) <- c("lunghezza", "larghezza")

>vettore <- iris[c("Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width")]   ← ho selezionato tutte e quattro le colonne di lunghezza e larghezza sepali e petali
>count_IQR <- c()   ←vettore vuoto che dovrò riempire
# Colonna 1
>q1 <- quantile(iris[,1], 0.25)
>q3 <- quantile(iris[,1], 0.75)
>count_IQR["Sepal.Length"] <- sum(iris[,1] >= q1 & iris[,1] <= q3)   ←ho calcolato prima i quantili q1 e q3. Ho iniziato a riempire il vettore assegnando un nome alla prima postazione, poi ho sommato tutti i valori della prima colonna che sono compresi tra q1 e q3
# Colonna 2
>q1 <- quantile(iris[,2], 0.25)
>q3 <- quantile(iris[,2], 0.75)
>count_IQR["Sepal.Width"] <- sum(iris[,2] >= q1 & iris[,2] <= q3)
# Colonna 3
>q1 <- quantile(iris[,3], 0.25)
>q3 <- quantile(iris[,3], 0.75)
>count_IQR["Petal.Length"] <- sum(iris[,3] >= q1 & iris[,3] <= q3)
# Colonna 4
>q1 <- quantile(iris[,4], 0.25)
>q3 <- quantile(iris[,4], 0.75)
>count_IQR["Petal.Width"] <- sum(iris[,4] >= q1 & iris[,4] <= q3)
>print(count_IQR)
>risultato <- list(matrice, count_IQR)
>print (risultato)
```

➔ Esercizio 3

Scrivere una funzione che presa in input una matrice M di due colonne, salvi in disp.png e bar.png rispettivamente il diagramma di dispersione delle due colonne e un barplot contenente entrambe le colonne di M. Nominare gli assi in entrambe le figure. Controllare che la matrice abbia esattamente 2 colonne (if). Chiamare la funzione sulla matrice ottenuta leggendo i dati da SpiderTable.txt

```
>Matrice <- read.table("SpiderTable.txt", header=T)

>salva_grafici <- function(Matrice) {
        if(ncol(Matrice) != 2){                    ←verifico che la matrice sia di due colonne
                stop("la matrice deve avere due colonne")
                }
        png("disp.png")
        plot(Matrice[,1], Matrice[,2], xlab="colonna 1", ylab="colonna 2", main="grafico di
        dispersione", col="pink")
        dev.off()
        png("bar.png")
        barplot(t(Matrice), beside=T, col=c("pink", "red"), xlab ="Osservazioni", ylab ="Valori",
        las=2, legend.text=C("colonna1", "colonna2"), names.arg=1:nrow(Matrice))
        dev.off()
}
salva_grafici(Matrice)
```

➔ Esercizio 4

Scrivere una funzione R conta_estremi che presa in input una matrice M, calcoli il numero di valori estremi in ciascuna colonna, e li restituisca all'ambiente chiamante in un vettore. Suggerimento: scrivere prima una funzione che riceve un vettore numerico e ne restituisce il numero di valori estremi, quindi chiamarla all'interno di conta_estremi.

```
>conta_estremi_vettori <- function(vett) {
        q1 <- quantile(vett, 0.25)
        q3 <- quantile (vett, 0.75)
        IQR <- IQR(vett)
        limite_inf <- q1 - 1.5 *IQR
```

```
    limite_sup <- q3 + 1.5 *IQR
    sum(vett < limite_inf || vett > limite_sup)    ←conta i valori estremi superiore o inferiore
}
>conta_estremi <- function(Matrice) {        ← creo la funzione conta_estremi
    apply(Matrice, 2, conta_estremi_vettore)    ← applica la funzione conta_estremi_vettori ad ogni colonna della matrice
}
```

➔ Esercizio 5

Scrivere una funzione conta_occorrenze, che riceve due vettori, parole e cerca: entrambi contengono delle stringhe, e la funzione deve contare ciascuna stringa in cerca quante volte compare in parole, quindi restituire un vettore (lungo quanto cerca) con le occorrenze trovate per ciascuna stringa in cerca. Nota: gli operatori usati per vettori numerici (which, %in%, etc) sono gli stessi da usare per vettori di stringhe.

```
>conta_occorrenze <- function(parole, cerca) {
    if (!is.character(parole) || !is.character(cerca)) {
        stop("Entrambi gli argomenti devono essere vettori di stringhe.")
    }
    conteggi <- numeric(length(cerca))  ←creo un vettore di zeri lungo quanto "cerca"
    names (conteggi) <- cerca
    for(i in 1:length(cerca)){      ← Per ogni parola in cerca, conta quante volte appare in parole
        conteggi[i] <- sum(parole == cerca[i])
    }
return(conteggi)
}
>conta_occorrenze(parole, cerca)
```

➔ Esercizio 6