Algorithms and Programming in R for Data Science

Dario Comanducci

I. Generazione del dataset

I parametri sono stati raggruppati in due liste, `infoSchool` e `infoStud`, per praticità e pulizia del codice: per quanto riguarda il numero di scuole per ciascun tipo (licei o istituti), in `infoSchool` avvaloriamo direttamente `N_Ls=30` e `N_It=70` dato che in totale le scuole sono 100 e sono in rapporto 30 a 70 tra licei ed istituti; il significato dei parametri in `infoStud` ricalca il testo del compito (in linea generale per tutto l'elaborato, variabili contententi 'Ls' afferiscono ai licei scientifici, quelle con 'It' agli istituti tecnici). Inoltre viene indicato il numero complessivo `n=200000` di studenti ed impostato un opportuno seme nella generazione dei numeri casuali (`set.seed(314)`), per la riproducibilità dei risultati.

La funzione `CreateDataFrame()`, con l'ausilio dell'altra funzione `AssignStudentsToSchool()` costituisce il Data Generating Process (DGP) dell'elaborato: poiché non è richiesta nessuna elaborazione sui dati di uno specifico studente, abbiamo evitato di associare ad ogni individuo un codice identidicativo, affidandoci piuttosto agli indici di posizione all'interno dei vettori di `n` elementi creati durante il DGP.

All'interno di `CreateDataFrame()` si imposta la distribuzione di probabilità che uno studente vada o al Liceo scientifico o all'Istituto tecnico: trattandosi di due sole alternative, la distribuzione è binomiale con probabilità `probLs` che uno studente frequenti un liceo scientifico, e probabilità `1-probLs` che invece vada all'Istituto tecnico; pertanto `isLyceum = rbinom(n,1, probLs)` genera un vettore di `n` elementi binari (0/1), dove 1 significa che il corrispondente studente va al liceo. La successiva istruzione `schoolType = ifelse(isLyceum == 1, 'Ls','It')` associa le stringhe `'Ls'` o `'It'` ai due tipi di scuola.

In maniera analoga si procede per assegnare il sesso (`'F'` e `'M'`) nel vettore `gender` per gli `n` studenti. Avremmo potuto ottenere un risultato analogo tramite la funzione `sample()`; ad esempio, sul tipo di scuole: `schoolType = sample(c('Ls','It'), size=n, replace=TRUE, prob=c(probLs,1-probLs))`

Nell'assegnare i voti si è ricorso alla funzione `which()` per recuperare gli indici degli studenti di liceo, e similmente per determinare quelli degli studenti d'istituto, così come per maschi e femmine: le quattro possibili intersezioni (`idLsM = intersect(idLs,idMales)`, ad esempio) indicano gli elementi del vettore `votes = rep(NA,n)` a cui assegnare un valore secondo la distribuzione di probabilità corrispondente (ad esempio `votes[idLsM] = rnorm(length(idLsM), infoStud$meanVoteLsByMales, infoStud$stdevVoteByMales)`); in mancanza di ulteriori indicazioni, si sono ritenuti validi anche voti esterni a $[0, 10]$, allo scopo di preservare la corrispondenza tra i dati empirici/sintetizzati e la distribuzione teorica.

Dopo aver generato i nomi fittizzi dei 30 licei (`schoolLs = sapply(1:infoSchool$N_Ls, function(k) sprintf("Ls%02d", k))` come `'Ls01' ...'Ls30'`, questi vengono assegnati rispettivamente agli studenti di liceo applicando un campionamento uniforme con reimbussolamento all'interno della funzione `AssignStudentsToSchool()` tramite la chiamata a `sample(x=schools, size=nStuds, replace=T)`. I vari licei vengono poi assegnati aglii alunni liceali, all'interno del vettore `goesTo=rep(NA,n)` precedentemente allocato, scorrendo gli indici in `idLs`. Analogo meccanismo per i 70 istituti tecnici `'It01' ...'It70'`.

Al termine `CreateDataFrame()` restituisce tutto nel dataframe `dfStudents`, avendo reso variabili categoriche il tipo di scuola (`as.factor(schoolType)`), la scuola frequentata (`as.factor(goesTo)`) ed il genere (`as.factor(gender)`) per ciascuno degli `n` studenti. Pertanto il dataset viene creato con `dfStudents = CreateDataFrame(n, infoStud, infoSchool)`.

II. Analisi del dataset

A.

La funzione `table()`, applicata prima alla coppia (`dfStudents$School, dfStudents$Gender`) e poi a (`dfStudents$SchoolType, dfStudents$Gender`), consente di generare le matrici di contingenza:

```
table(dfStudents\$School, dfStudents\$Gender)
table(dfStudents\$SchoolType,dfStudents\$Gender)
```

B.

Un modo di ottenere tali valori è quello di ricorrere alla libreria `dplyr` o in alternativa alla libreria `data.table`:

```
# con dplyr
recapVoteByGender = dfStudents %>%
        group_by(Gender) %>%
                summarise(avg=mean(Vote),
                med=median(Vote),
                q25=quantile(Vote,0.25),
                q75=quantile(Vote,0.75),
                sd=sd(Vote))
# con data.table
print(as.data.table(dfStudents)[,
        list(avg=mean(Vote),
                med=median(Vote),
                q25= quantile(Vote,0.25),
                q75= quantile(Vote,0.75),
                sd=sd(Vote)), by = .(Gender)])
```

In maniera analoga si ottengono le tabelle per tipo di scuola, con `SchoolType` al posto di `Gender`.

C.

Per visualizzare la distribuzione del voto, possiamo impiegare sia dei boxplot che degli istogrammi, avendo definito opportuni colori:

```
op = par('mfrow','mar','oma') #salvo default
# split immagine
par(mfrow=c(1,2), mar=c(4.5,4.0,1.5,0.5))
boxplot(formula=dfStudents$Vote ~
        dfStudents$Gender, data=dfStudents,
        frame.plot=F, col=gcol, border=bcol)

xM = dfStudents$Vote[dfStudents$Gender=='M']
xF = dfStudents$Vote[dfStudents$Gender=='F']
hist(x=xF, freq=F, col=colF_tr,
        xlim=c(min(xM),max(xM)))
hist(x=xM, freq=F, col=colM_tr, add=T)
```

Analizziamo il codice precedente:

- l'impiego di `formula = dfStudents$Vote ~ dfStudents$Gender` nella funzione `boxplot()` permette di creare tanti boxplot sui voti in `dfStudents$Vote` quanti sono gli elementi in `dfStudents$Gender`, partizionati secondo `dfStudents$Gender`;
- viceversa, nella funzione `hist()` i dati sono splittati "manualmente" tra maschi e femmine nei vettori `xM` e `xF`, sovrapponendo due distinti istogrammi.

La linea `par(mfrow=c(...), mar=c(...))` consente di impostare vari parametri di visualizzazione: `mfrow=c(...)` divide l'area per i grafici in due parti, aggiustando in entrambi i casi i margini della figura corrispondente (`mar=c(...)`); l'istruzione `op = par('mfrow', 'mar', 'oma')` consente di memorizzare i valori originali per i vari parametri grafici, poi ripristinati nell'istruzione `par(mfrow=op$mfrow, mar=op$mar, oma=op$oma)` (`oma` serve dopo, per i margini esterni dove inserire un titolo comune a più plot).

D.

Analogamente a quanto già fatto nel punto precedente, scambiando `dfStudents$Gender` con `dfStudents$SchoolType` abbiamo boxplot e istogrammi dei voti ripartiti tra i due tipi di scuole.

E.

La strategia per creare la variabile ordinale `LabeledVote` è stata quella di creare un vettore a valori discreti (interi, in particolare) e crescenti in corrispondenza delle soglie indicate per ciascun livello `'gravemente insuff.'`, `'insufficiente'`, `'sufficiente'`, `'buono'`, `'ottimo'`: il vettore `LabeledVote` così creato è stato poi aggiunto al dataframe `dfStudents` per completezza.

```
LabeledVote = as.ordered(0
        + 1*(dfStudents$Vote>=5)
        + 1*(dfStudents$Vote>=6)
        + 1*(dfStudents$Vote>=7)
        + 1*(dfStudents$Vote>=8.5))
levels(LabeledVote)=c('gravemente␣insuff.',
        'insufficiente','sufficiente',
        'buono','ottimo')
```

In alternativa, usando `cut()` con i tagli definiti in `breaks`:[1]

```
LabeledVote = as.ordered(cut(dfStudents$Vote,
        breaks=c(-Inf,5,6,7,8.5,Inf), right=F))
```

F.

Una rappresentazione concisa delle informazioni richieste è ottenibile con tabelle a due vie, ricorrendo di nuovo alla funzione `table()`: l'applicazione, tramite `apply()`, della funzione `cumsum()` lungo le colonne della tabella `absFreqsTable_simple` fornisce riga per riga la somma degli elementi in ciascuna colonna.

```
absFreqsTable_simple = as.data.frame.matrix(
        table(dfStudents$LabeledVote,
              dfStudents$Gender))
absFreqsTable_cumul=apply(absFreqsTable_simple,
                          2, cumsum)
divisor = colSums(absFreqsTable_simple)
relFreqsTable_simple = absFreqsTable_simple
        /divisor[col(absFreqsTable_simple)]
relFreqsTable_cumul = absFreqsTable_cumul
        /divisor[col(absFreqsTable_cumul)]
```

Le tabelle per le frequenze relative sono state ottenute dividendo rispettivamente ciascuna colonna delle frequenze assolute per il totale di maschi e femmine presi separatamente (attraverso `col()` ci assicuriamo che ciascuna colonna venga divisa per la rispettiva somma cumulata in `divisor`).

La rappresentazione visuale di tali informazioni si basa su dei grafici a barre: volendo un quadro sinottico, tali plot sono stati raggruppati in un'unica illustrazione attraverso la funzione `layout()`, più flessibile rispetto a `mfrow()` per creare una cella trasversale in basso dove riportare la legenda comune a tutti e quattro i grafici.

Grazie a `mtext("Labeled Vote Frequencies", side=3, outer=T, font=1.5, cex=1.5)` è stato inserito infine un titolo comune per l'intera figura nello spazio impostato da `oma()`.

G.

Riprendendo quanto già fatto, si forma una vettore binario `IsOptimus`, con cui `table()` crea la matrice di contingenza rispetto al tipo di scuola, divisa poi per la somma lungo le colonne come al punto precedente.

```
dfStudents$IsOptimus =
        as.numeric(dfStudents$Vote >= 8.5)
optimusTable = as.data.frame.matrix(
        table(dfStudents$IsOptimus,
              dfStudents$SchoolType))
divisor = colSums(optimusTable)
optimusTable = optimusTable
        /divisor[col(optimusTable)]
```

[1] Negli estremi a destra e a sinistra sono stati posti $\pm$`Inf` per gestire eventuali voti generati senza essere vincolati in $[0, 10]$.