Generazione automatica di tabelle con LaTeX e *Stata*

Rosa Gini

Pisa, 18 febbraio 2004

Sommario

Si descrive brevemente il linguaggio utilizzato per far comporre tabelle al LaTeX. Si descrivono degli esempi di generazione automatica di tabelle dall'interno del programma statistico *Stata* (versione 8).

Indice

Introduzione

Da quasi 30 anni in moltissimi ambienti editoriali, più frequentemente di ambito scientifico nell'area matematica-informatica, per comporre testi (articoli, lettere, libri, presentazioni...) si

usa un linguaggio di programmazione chiamato TEX e la sua versione semplificata LATEX.[1] La ragione per cui la lettera finale di queste parole si legge “k” e non “x” è che essa non è una lettera latina ma una lettera greca, la χ, che si legge *chi greca* e ha un suono gutturale simile appunto a “k”.

Il programma LATEX è così diffuso perché si tratta di un sistema di scrittura molto semplice che produce output di qualità estetica professionale. Questo permette a chi compone il testo di concentrarsi sui contenuti e ottenere comunque alla fine un testo ottimo dal punto di vista della fruibilità, pur senza avere nessuna competenza di tipo editoriale.[2]

Il fatto di essere un linguaggio di programmazione permette a LATEX di ottenere in modo semplice un input da un altro programma. Questo documento illustra come l’uso di LATEX permette a *Stata* di scrivere automaticamente delle tabelle, in un formato gradevole da vedere e inseribile all’interno di qualsiasi rapporto, pubblicazione... Questa possibilità è particolarmente vantaggiosa se si devono generare ripetutamente molte tabelle a partire da archivi strutturati sempre nello stesso modo ma con dati che cambiano, per esempio da un flusso amministrativo: si stabilisce una volta per tutte quali tabelle riassuntive si vogliano desumere dai dati, si scrive un file di programma per *Stata* che scrive le tabelle nel formato LATEX e poi si compila un documento LATEX che include tutte le tabelle. Se i dati cambiano basta rifare tutto il passaggio automatico, ma non è più necessario ripensare a come strutturare i dati, né bisogna fare alcun taglia-e-incolla di tabelle: il documento che contiene le tabelle esce dalla compilazione già pronto per essere esaminato. Anche in questo caso questo comporta la possibilità di concentrarsi maggiormente sui contenuti (in questo caso le analisi riassunte nelle tabelle) piuttosto che sul modo per presentarli.

Questa dispensa è strutturata in due parti: nella prima si descrive sommariamente come si genera un documento in LATEX, e ci si sofferma sul linguaggio che definisce le tabelle; nella seconda parte si descrive una funzionalità del linguaggio di programmazione di *Stata* e la si utilizza per dare qualche esempio di generazione automatica di tabelle a partire da un archivio di dati.

1 Scrivere tabelle in LATEX

1.1 Come scrivere in LATEX

Il LATEX somiglia, come concetto e non solo, all’HTML: esiste un file di testo, denominato *sorgente*, che dice a un *compilatore* di scrivere un file che poi viene letto da un programma di *lettura*: nel caso dell’HTML il programma che legge i file è un qualsiasi *browser*, tipo Netscape Navigator, Mozilla, MS Internet Explorer, Opera... Nel caso di testi LATEX ci sono vari formati di output e quello che useremo noi è il comune formato pdf, leggibile con Adobe Acrobat Reader e con molti altri lettori. Quando diciamo che LATEX “fa” qualcosa intendiamo riferirci

[1] Più precisamente TEX è un linguaggio di programmazione creato per gestire la redazione di testi e LATEX è un insieme di *macro*, cioè di piccoli programmi, scritti nel linguaggio TEX, che permette di gestire la redazione del testo da un livello più alto. Per esempio chi scrive non capisce un’acca di TEX, ma scrive con LATEX da anni senza aver bisogno di sapere nulla di più.

[2] Tuttavia, purtroppo, LATEX non può far nulla per rendere più intelligenti i contenuti.

al compilatore, che ha il ruolo di interpretare il file sorgente e di generare il file pdf. Quando diciamo che LaTeX "è" qualcosa intendiamo riferirci al linguaggio del LaTeX, cioè alle istruzioni che il compilatore comprende.

Chi scrive in LaTeX di fatto scrive il file sorgente. Per scriverlo può utilizzare un text editor o un word processor qualsiasi, perfino MS Word, anche se è consigliabile utilizzare programmi che non si prendono l'arbitrio di modificare il testo. Per esempio MS NotePad o MS WordPad vanno benissimo, ma ci sono alcuni editor gratuiti e liberi molto semplici e che supportano il linguaggio del LaTeX, nel senso che mettono in evidenza (per esempio cambiando colore) i simboli e i comandi specifici del linguaggio.

Questo documento è scritto con il Crimson Editor, un minuscolo text editor libero.

Il compilatore LaTeX esiste in molte distribuzioni, per tutti i sistemi operativi. La distribuzione più nota per i sistemi operativi Microsoft è il MiKTeX, quella per le distribuzioni Linux è TeTeX.

Non è nello scopo di questo documento dare indicazioni su come scrivere un documento LaTeX, se non quel minimo che serve per far compilare delle tabelle. In generale è utile sapere che LaTeX, come l'HTML, è un *linguaggio di markup*, cioè un linguaggio i cui il testo è accompagnato da parole che spiegano al compilatore come deve utilizzare il testo medesimo.

Esempio 1 Per esempio se vogliamo dire a LaTeX che stiamo per cominciare una sottosezione il cui titolo è "Alcuni esempi" gli scriviamo `\subsection{Alcuni esempi}`. Questa informazione induce LaTeX a fare varie cose: anzitutto a scrivere la frase "Alcuni esempi" in un certo stile (che qualche altro pezzo di codice gli ha già detto di adottare quando deve scrivere il titolo di una sottosezione), poi di numerare la sottosezione con un numero progressivo, sempre stabilito dal compilatore, poi di scrivere una riga di indice, e molte altre cose.

Attenzione che se un comando vale solo per *una parte* del testo, quella parte va circondata da parentesi graffe, oppure il comando deve essere della forma `\begin{comando}` ed `\end{comando}`. In questo secodno caso si dice che il comando definisce un *ambiente* (spesso citato con il nome inglese *environment*).

Esempio 2 Se vogliamo dire a LaTeX che un pezzo di testo va messo in grassetto gli scriviamo `\bf{una frase d'esemp`
e lui scriverà **una frase d'esempio**.

Esempio 3 Se vogliamo dire a LaTeX che un pezzo di testo va centrato gli scriviamo

```
\begin{center}  Pezzo da centrare.  \end{center}
```

e lui obbedendo scriverà

Pezzo da centrare.

Questo è un esempio di ambiente: l'ambiente *centrato*.

In particolare il concetto di *markup* vale per il documento stesso. Quindi per indurre LaTeX a compilare un file sorgente esso deve rispondere alle seguenti regolette:

- il file sorgente deve chiamarsi con l'estensione .tex;
- il file deve cominciare con la frase

  `\documentclass{article}`

  oppure `\documentclass{book}`, ... che dice a LaTeX se è nostra intenzione scrivere un articolo o un libro o altro;
- il testo vero e proprio del documento deve essere racchiuso tra le istruzioni

  `\begin{document}` ed `\end{document}`.

Dopodiché se si dà in pasto al compilatore LaTeX il file, tramite il comando `pdflatex`, esso produce un file pdf che ha lo stesso nome del file sorgente tex. La compilazione può essere fatta direttamente da una finestra DOS o da una Shell Unix, ma moltissimi editor di testi sono predisposti per creare un tasto o una macro che compila il file direttamente dalla finesta dell'editor medesimo. Per esempio il Crimson Editor è predisposto, e sul sito dell'editor è spiegato come fare per programmare la macro.

Esempio 4: breve documento completo. Se prendiamo un editor di testi, per esempio Crimson Editor o MS NotePad o OpenOffice Word, e ci copiamo questo testo:

```
\documentclass[12pt]{article}
\begin{document}
``Beh, nel {\em nostro} paese,'' disse Alice, ansimando ancora
lievemente, ``si arriverebbe da qualche parte se si corresse molto
velocemente e a lungo, come abbiamo fatto noi.''

``Che paese lento!'' disse la Regina. ``Guarda, {\em qui}, vedi,
devi correre al massimo della velocità che {\em tu} sembri poter
raggiungere, solo per restare nello stesso posto.  Se vuoi andare
da qualche parte, devi essere veloce almeno il doppio!''

\end{document}
```

e poi salviamo il file con il nome `alice.tex` e compiliamo (con il comando `pdflatex`), otteniamo un file di nome `alice.pdf` che, aperto con un normale lettore di pdf tipo Adobe Acrobat Reader, appare così:

"Beh, nel *nostro* paese," disse Alice, ansimando ancora lievemente, "si arriverebbe da qualche parte se si corresse molto velocemente e a lungo, come abbiamo fatto noi."

"Che paese lento!" disse la Regina. "Guarda, *qui*, vedi, devi correre al massimo della velocità che *tu* sembri poter raggiungere, solo per restare nello stesso posto. Se vuoi andare da qualche parte, devi essere veloce almeno il doppio!"

1.2 Alcuni esempi di `tabular`

Il comando che dice a LaTeX di scrivere una tabella è `tabular`, che definisce l'ambiente omonimo. Come funziona? È più facile farlo che dirlo: diamo qualche esempio, descrivendo mano a mano come si costruiscono.

Esempio 5: tabella semplice con linee.
Il codice

```
\begin{tabular}{r|c|c|}
&Maschi&Femmine\\ \hline
Giovani&5&14\\ \hline
Anziani&15&3\\ \hline
\end{tabular}
```

genera la tabella

	Maschi	Femmine
Giovani	5	14
Anziani	15	3

Esempio 6: tabella semplice senza linee, centrata.
Il codice

```
\begin{center}
\begin{tabular}{rcc}
&Maschi&Femmine\\
Giovani&5&14\\
Anziani&15&3\\
\end{tabular}
\end{center}
```

genera la tabella

	Maschi	Femmine
Giovani	5	14
Anziani	15	3

Da questi due esempi si capisce il funzionamento elementare del comando `tabular`. Dopo l'inizio dell'ambiente `tabular` bisogna elencare tra parentesi graffe quante colonne avrà la tabella, contemporaneamnete indicando se il contenuto sarà centrato o allineato a destra o a sinistra e se ci devono essere linee verticali: `{rcc}` significa che la tabella avrà 3 colonne, la prima allineata a destra e le seconde due centrate, invece `{r|llr}` significa che ci saranno 4 colonne, la prima e l'ultima allineate a destra, la seconda e la terza allineate a sinistra e che tra la prima e la seconda ci sarà una linea verticale... e così via.

Poi i dati vanno inseriti riga per riga, in uno stile del tutto simile al *comma-separated values*, con la differenza che la virgola e il simbolo di a capo sono sostituiti da altri simboli: al posto della virgola tra un elemento dal successivo di mette il carattere `&` e alla fine di ogni riga si mette il comando `\\`. Se dopo la fine di una riga si vuol tirare una linea orizzontale, prima di cominciare a inserire la riga successiva si scrive `\hline`.

Esempio 7: uso di titoli su più colonne.

Il codice

```
\begin{tabular}{lcccc}
&\multicolumn{2}{c}{Giovani}&\multicolumn{2}{c}{Anziani}\\
&Maschi&Femmine&Maschi&Femmine\\ \hline\hline
Simpatici&5&14&15&3\\
Antipatici&1&3&1&1\\
\end{tabular}
```

genera la tabella

	Giovani		Anziani	
	Maschi	Femmine	Maschi	Femmine
Simpatici	5	14	15	3
Antipatici	1	3	1	1

Esempio 8: uso di righe orizzontali su qualche colonna.

Il codice

```
\begin{tabular}{lcccc}
&\multicolumn{2}{c}{Giovani}&\multicolumn{2}{c}{Anziani}\\
&Maschi&Femmine&Maschi&Femmine\\ \cline{2-5}
Simpatici&5&14&15&3\\
Antipatici&1&3&1&1\\
\end{tabular}
```

genera la tabella

	Giovani		Anziani	
	Maschi	Femmine	Maschi	Femmine
Simpatici	5	14	15	3
Antipatici	1	3	1	1

Questi due esempi mostrano come si può spezzare la suddivisione in colonne di una riga: con il comando `\multicolumn{2}{c}{Anziani}` diciamo al LaTeX di scrivere il testo “Anziani” su due colonne e di centrarlo. Se avessimo scritto `\multicolumn{3}{l|}{Anziani}` gli

avremmo detto di scrivere quel testo su 3 colonne e di far seguire una linea verticale (lunga solo come la riga che stiamo scrivendo), e così via.

Il comando `\cline{2-5}` si mette al posto di `\hline` quando si vuole che la linea orizzontale non vada da un estremo all'altro della tabella ma solo dalla colonna 2 alla colonna 5.

Esempio 9: scrittura di testi in tabelle.

Il codice

```
\begin{tabular}{cp{3cm}p{3cm}p{3cm}}
Indicatore&Definizione&Numeratore&Denominatore\\ \hline
1&Tasso di simpatia di un ufficio&
Numero di colleghi d'ufficio simpatici&Numero di colleghi d'ufficio\\ \hline
2&Numero medio giornaliero di pause caffè {\em pro capite} &
Numero di pause caffè in un giorno&Numero di colleghi d'ufficio\\ \hline
\end{tabular}
```

genera la tabella

Indicatore	Definizione	Numeratore	Denominatore
1	Tasso di simpatia di un ufficio	Numero di colleghi d'ufficio simpatici	Numero di colleghi d'ufficio
2	Numero medio giornaliero di pause caffè *pro capite*	Numero di pause caffè in un giorno	Numero di colleghi d'ufficio

Esempio 10: colorare le righe.

```
\begin{tabular}{cccccc}
&&\multicolumn{4}{c}{\bf{Lavoratori }}\\
Azienda&&Biondi&Bruni&Rossi&Calvi\\ \hline
\rowcolor[gray]{.8}1&Non Simpatici&2&8&23&37\\
\rowcolor[gray]{.8}&Simpatici&148&294&630&388\\
\rowcolor[gray]{.9}2&Non Simpatici&2&7&31&22\\
\rowcolor[gray]{.9}&Simpatici&169&400&783&451\\
\rowcolor[gray]{.8}3&Non Simpatici&0&5&27&46\\
\rowcolor[gray]{.8}&Simpatici&169&395&882&592\\
\rowcolor[gray]{.9}4&Non Simpatici&0&9&25&30\\
\rowcolor[gray]{.9}&Simpatici&148&340&809&428\\
\end{tabular}
```

dà luogo alla tabella

		Lavoratori			
Azienda		Biondi	Bruni	Rossi	Calvi
1	Non Simpatici	2	8	23	37
	Simpatici	148	294	630	388
2	Non Simpatici	2	7	31	22
	Simpatici	169	400	783	451
3	Non Simpatici	0	5	27	46
	Simpatici	169	395	882	592
4	Non Simpatici	0	9	25	30
	Simpatici	148	340	809	428

E così via. Praticamente ogni tipo di tabella si può replicare nel linguaggio LaTeX: si possono colorare le colonne, si può ruotare il testo…

1.3 Le tabelle “svolazzanti”

Spesso in un documento non è bello mettere una tabella là dove è citata per la prima volta, perché questo obbliga magari a lasciare molto spazio bianco e cambiar pagina per poter far stare tutto l’oggetto. Per questo LaTeX si incarica di gestire gli oggetti *floating*, cioè “svolazzanti”, piazzandoli a suo gusto dove gli par meglio. Naturalmente per poter indicare questi oggetti è necessario che siano numerati e che abbiano una didascalia. Per esempio la tabella 1 a pagina 9 è generata dal codice seguente:

```
\begin{table}[ht]\centering\caption{Una tabella ‘‘svolazzante’’}
\label{svolazzante}
\begin{tabular}{|rccccc|}
\hline
&Strepitosi&Belli&Passabili&Così così& Ributtanti\\
Neri&$1434$ & $348$ & $307$ & $2214$ & $710$ \\
Gialli& $1747$ & $1774$ & $1220$ & $1340$ & $3740$ \\
Verdi&$2196$ & $3125$ & $309$ & $3123$ & $1016$ \\
Castani&$697$ & $888$ & $477$ & $666$ & $422$ \\
Magenta & $593$ & $885$ & $286$ & $522$ & $832$ \\ \\ \hline
\end{tabular}
\end{table}
```

e il riferimento al numero della tabella e alla pagina in cui essa appare (che è sconosciuto a chi scrive il testo, perché sarà il calcolatore a deciderlo quando il testo verrà compilato) è fatto in modo logico: il numero 1 è generato dal comando `\ref{svolazzante}` e il numero 9 è generato dal comando `\pageref{svolazzante}`.[3]

[3]Si noti che questi due comandi sono preziosissimi quando si stende un documento complesso. Infatti se nel corso della scrittura si aggiungono altre tabelle prima di questa, e quindi il numero della tabella cambia, o si sposta

Tabella 1: Una tabella “svolazzante”

	Strepitosi	Belli	Passabili	Così così	Ributtanti
Neri	1434	348	307	2214	710
Gialli	1747	1774	1220	1340	3740
Verdi	2196	3125	309	3123	1016
Castani	697	888	477	666	422
Magenta	593	885	286	522	832

Questo prezioso servizio di LaTeX ammette volentieri che l’utente avanzi delle preferenze circa il colocamento della tabella: l’opzione `[ht]` accanto all’inizio dell’ambiente `table` serve a comunicare a LaTeX che preferiremmo la tabella il più vicino possibile (opzione `h` per “here”) oppure, in subordine, in cima alla pagina (opzione `t` per “top”). Con la stessa forma si può specificare che preferiremmo la tabella in fondo alla pagina (opzione `b` per “bottom”). Queste opzioni sono facoltative, e possono essere messe in qualsiasi ordine e numero.

Ma l’ambiente `table` è comunque molto geloso del suo buon gusto nel piazzare le tabelle, e raramente obbedirà alla richiesta di mettere la tabella “qui” se avanzata con tanta gentilezza. Se si vuole insistere si può far precedere le opzioni da un punto esclamativo: la stringa `[!ht]` cercherà di persuare il LaTeX con più autorevolezza.

A volte può semplicemente far piacere assicurarsi che il compilatore metta gli oggetti “svolazzanti” *entro* una certa pagina, per esempio prima dell’inizio di un nuovo capitolo o di una nuova sezione. Per assicurarsi questo si può utilizzare il comando `\clearpage`, che obbliga LaTeX a stampare immediatamente tutti gli oggetti *floating* e a cominciare una nuova pagina.

Infine: è chiaro che l’ambiente `table` può essere scelto semplicemente perché fornisce una didascalia, e che può darsi che chi lo sceglie non abbia alcun desiderio di attivare la sua capacità di far “svolazzare” le tabelle. La possibilità di apporre didascalie agli oggetti in modo più libero, nonché di gestire più attivamente posizione e orientazione delle tabelle, è assicurata da molti pacchetti, per esempio `hvFloat`.

1.4 Tabelle larghe, tabelle lunghe, tabelle larghe e lunghe

Se una tabella contiene molte colonne si può provare a “schiacciarla” un po’ per farla stare tra i margini. Questo è possibile inserendo la tabella (dentro o fuori dall’eventuale ambiente `table`) in una “scatola” della dimensione del testo.

Esempio 11: tabella larga verticale. Il codice

```
\resizebox*{0.9\textwidth}{!}{
\begin{tabular}{lcccc}
\multicolumn{5}{c}{\bf Percentuali di antipatici nelle aree
```

di pagina la tabella, il riferimento logico continuerà a funzionare bene, e continuerà a mettere i numeri giusti. Questi comandi di *labelling* e *referencing* possono essere usati per qualsiasi altro oggetto numerato nel testo: i numeri delle sezioni, delle note, delle figure…

```
della Toscana, dal 1999 al 2002}\\ \hline
\bf Anni& \bf Area Centro &\bf  Area Sud-Est
& \bf Area Nord-Ovest& \bf Toscana\\ \hline
1999 &2.3\% (IC: 2.3-2.4)&2.6\% (IC: 2.5-2.7)
&2.8\% (IC: 2.8-2.9)&\bf 2.6\% (IC: 2.5-2.6)\\
2000 &2.6\% (IC: 2.6-2.7)&3.1\% (IC: 3.0-3.1)
&3.6\% (IC: 3.5-3.6)&\bf 3.1\% (IC: 3.0-3.1)\\
2001 &2.9\% (IC: 2.9-3.0)&3.6\% (IC: 3.5-3.6)
&4.2\% (IC: 4.1-4.3)&\bf 3.5\% (IC: 3.5-3.6)\\
2002 &3.1\% (IC: 3.0-3.1)&3.3\% (IC: 3.2-3.4)
&3.7\% (IC: 3.7-3.8)&\bf 3.4\% (IC: 3.3-3.4)\\
\hline\hline
\end{tabular}
}
```

genera la tabella seguente.

Percentuali di antipatici nelle aree della Toscana, dal 1999 al 2002

Anni	Area Centro	Area Sud-Est	Area Nord-Ovest	Toscana
1999	2.3% (IC: 2.3-2.4)	2.6% (IC: 2.5-2.7)	2.8% (IC: 2.8-2.9)	**2.6% (IC: 2.5-2.6)**
2000	2.6% (IC: 2.6-2.7)	3.1% (IC: 3.0-3.1)	3.6% (IC: 3.5-3.6)	**3.1% (IC: 3.0-3.1)**
2001	2.9% (IC: 2.9-3.0)	3.6% (IC: 3.5-3.6)	4.2% (IC: 4.1-4.3)	**3.5% (IC: 3.5-3.6)**
2002	3.1% (IC: 3.0-3.1)	3.3% (IC: 3.2-3.4)	3.7% (IC: 3.7-3.8)	**3.4% (IC: 3.3-3.4)**

Qui si vede che la tabella è stata schiacciata, mantenendo le proporzioni, alla larghezza che abbiamo chiesto, in questo caso `0.9\textwidth`, ovvero il 90% della larghezza del testo. Allo stesso modo avremmo potuto chiedere di stringere un po' di più o un po' di meno.

Tuttavia a volte le tabelle così ristrette risultano illeggibili, e si vorrebbe ruotarle di 90 gradi in modo da leggerele sul lato lungo della pagina. La Tabella 4 a pagina 22 è scritta nell'ambiente `sidewaystable`, che funziona esattamente come `table` solo che ruota la tabella e la didascalia di 90 gradi. Questo ambiente è ottenibile inserendo il pacchetto rotating o uno qualsiasi dei suoi numerosissimi derivati.

Esempio 12: tabella larga orizzontale. La tabella 4 rimane ancora un po' troppo larga, e quindi è infilata anche in una scatola come nell'esempio precedente. Il codice, che non riportiamo completamente perché è troppo lungo, è del tipo

```
\begin{sidewaystable}[ht]
\resizebox*{\textwidth}{!}{
\begin{tabular}{cc|cccc|cccc|cccc||cccc}
&Anno&\multicolumn{4}{c}{2000}&\multicolumn{4}{c}{2001}&\multicolumn{4}{c}{2002}
&\multicolumn{4}{c}{\bf{Totali azienda}}\\
Azienda&&45-64&65-74&75-84&85+&45-64&65-74&75-84
&85+&45-64&65-74&75-84&85+&45-64&65-74&75-84&85+\\
```

```
\hline1&Gravemente antipatici&91&73&52&22&82&53&57&18
&166&122&104&18&421&314&274&82\\
&Non gravemente antipatici&78&68&74&49&72&70
&81&59&174&154&176&50&368&357&390&192\\
...
\end{tabular}}
\caption{esempio di grossa tabella generata automaticamente (i dati sono fittizi)}
\label{generata}
\end{sidewaystable}
```

Finora abbiamo cercato di far stare le nostre tabelle in un'unica pagina. Tuttavia a volte questo non è comunque possibile, o non è preferibile considerando il tipo di documento che si vuole produrre. Se si ha una tabella lunga ma *non* larga, ovvero la cui larghezza è contenuta nella larghezza del documento, è sufficiente usare l'ambiente `longtable` associato al pacchetto omonimo: questo ambiente permette alla tabella di spezzarsi su più pagine successive, e ammette varie opzioni che permettono di ripetere ad ogni pagina il titolo della tabella, o le intestazioni delle colonne...

Se invece la tabella è lunga ma anche larga, non c'è altra possibilità che modificare completamente l'orientazione del documento. Il pacchetto rotating permette di generare l'ambiente landscape, che comincia una nuova pagina e dispone tutto il pezzo di documento successivo sulla pagina rovesciata. In particolare all'interno di questo ambiente si può inserire un `longtable`, e quindi la tabella si spezzerà su più pagine.

1.5 Separare i dati dal resto del documento

Spesso è conveniente estrarre dal file principale i frammenti che contengono i dati o comunque le tabelle. Questo è possibile usando il comando `\input`.

Esempio 13 Per esempio supponiamo di scrivere nel file `main.tex` il seguente codice:

```
\documentclass{article}
\begin{document}
\input{01.tex}
\end{document}
```

e poi di scrivere nel file `01.tex` il seguente codice:

```
\begin{tabular}{rcc}
&Maschi&Femmine\\
Giovani&5&14\\
Anziani&15&3\\
\end{tabular}
```

Quello che succederà quando compileremo `main.tex` sarà uguale a quello che succederebbe se compilassimo il file:

```
\documentclass{article}
\begin{document}
\begin{tabular}{rcc}
&Maschi&Femmine\\
Giovani&5&14\\
Anziani&15&3\\
\end{tabular}
\end{document}
```

e cioè verrà generata la stessa tabella dell'esempio 6 a pagina 5. Se però cambieranno i dati non sarà necessario metter mano al codice principale, cioè al file `main.tex`, basterà modificare il file specifico della tabella. Nel caso di questo esempio ovviamente non fa nessuna differenza, ma se il documento principale contiene molto testo e richiama molte tabelle il fatto che esse siano in file autonomi può tornare molto utile, specie se vogliamo modificarle in modo automatico.

1.6 Esportare e importare dati da varie sorgenti

Da una tabella contenuta in un documento pdf ovviamente non si possono estrarre dati da riversare in altri formati, e questo vale anche se il documento è stato ottenuto con il LaTeX. Se tuttavia del file avete anche il sorgente, dal testo del sorgente potete facilmente estrarre i dati, copiarli in un nuovo documento di testo, sostituire i simboli `&` con delle virgole e i simboli `\\` con degli a capo e avrete una tabella in formato csv, importabile in qualsiasi software che gestisce dati: software statistici, fogli di calcolo, gestori di data base...

Se viceversa si hanno dei dati già raccolti in un altro formato, e si vuole metterli rapidamente in formato LaTeX si può fare il procedimento sopra descritto a rovescio. Dopodiché bisogna racchiudere i dato tra `\begin{tabular}` ed `\end{tabular}`, specificando il simbolo che si desidera (`l`, `c`, `r`...) per ogni colonna eccetera.

Se si ha una tabella già formattata in un foglio di calcolo, con colori, righe orizzontali e verticali eccetera, si possono utilizzare vari piccolissimi software liberi, per esempio Spreadsheet2LaTeX o Excel2LaTeX, che danno un codice LaTeX già pronto per essere compilato.

2 Usare *Stata* per generare tabelle

2.1 I programmi già esistenti: `outtable` e `sutex`

I mondi di *Stata* e LaTeX hanno parecchio in comune, e soprattutto il fatto di aver entrambi generato una vera e propria *comunità di utenti*: chi usa questi programmi spesso fa dei piccoli miglioramenti che poi rende disponibili agli altri.

In particolare c'è già stato qualcuno che ha pensato di riversare l'output di alcuni comandi di *Stata* in un file di testo da inserire in un file LaTeX. Diversi comandi sono già disponibili e installabili dall'interno stesso di *Stata* con il comando `ssc install`.

Tra questi il più semplice è `outtable`: se `A` è una matrice in *Stata*, il comando

```
outtable using  dastata.tex, mat(A)
```

copia la matrice `A` nel file `dastata.tex`, pronto per essere incluso in un file sorgente tramite il comando `\input`.

Un altro utile comando è `sutex`, che fa uscire in formato LaTeX l'output del comando tradizionale `summarize`. Per esempio aprendo in *Stata* un archivio contenente le variabili `V1 V2 V3` e dando il comando

```
sutex V1 V2 V3, file(dastata.tex)
```

si fa generare a *Stata* il file `dastata.tex` con il seguente contenuto:

```
\begin{table}[htbp]\centering \caption{Summary statistics \label{sumstat}}
\begin{tabular}{l c c  c}\hline\hline
\multicolumn{1}{c}{\textbf{Variable}} & \textbf{Mean}
 & \textbf{Std. Dev.} & \textbf{N}\\ \hline
V1 & 0.105 & 0.058  & 14112\\
V2 & 0.09 & 0.058  & 14112\\
V3 & 0.093 & 0.063  & 10416\\
\hline\end{tabular}
\end{table}
```

Includendo il file in un documento con il comando `\input` e compilando si ottiene quindi questa tabella:

Variable	**Mean**	**Std. Dev.**	**N**
V1	0.105	0.058	14112
V2	0.09	0.058	14112
V3	0.093	0.063	10416

2.2 Come rendere *Stata* una diligente segretaria

Forse non tutti sanno che *Stata* può scrivere dei file. Ciò che gli permette di farlo sono i comandi `file open`, `file write` e `file close`. Se da dentro *Stata* si scrive

```
file open fiorellino using dastata.tex, write replace
file write  fiorellino "Ma che bella giornata!" _n
file close fiorellino
```

Stata si comporta come una brava segretaria: apre un file di nome `dastata.tex`, ci scrive dentro "Ma che bella giornata!" andando a capo alla fine (è questo l'effetto del simbolo `_n`)e infine chiude il file.

In questo esempio abbiamo richiamato la parola "fiorellino" 3 volte: la prima quando abbiamo aperto il file, la seconda quando ci abbiamo scritto e la terza quando l'abbiamo chiuso. Questo è un nome ausiliario che *Stata* utilizza quando scrive dei file. Naturalmente al posto di "fiorellino" avremmo potuto usare qualsiasi altro nome, la cosa importante è che il nome usato per aprire il file va poi richiamato quando ci si scrive dentro o lo si chiude. Attenzione a non confondere questo nome fittizio con il nome del file generato: questo viene nominato una sola volta, quando il file viene aperto.

2.3 Programmare *Stata*

Il programma *Stata* ha una facile modalità interattiva: si avvia il programma e del suo interno si compiono interattivamente delle operazioni, per esempio caricare un archivio e poi eseguire su di esso delle analisi. Queste operazioni interattive possono essere comandate sia dai vari menù a tendina e dai vari tasti della finestra del programma, sia inserendo i comandi come linee di testo nel riquadro inferiore della finestra del programma stesso. Quando i comandi si danno via menù essi compaiono automaticamente anche in forma di comando testuale: leggendo queste "traduzioni" si può rapidamente apprendere il linguaggio dei comandi di *Stata* .

In realtà *Stata* è molto più potente di così: infatti *Stata* ha un *secondo* modo di ricevere ordini: per mezzo di un file in cui i comandi vengono elencati. In questi file si può utilizzare un vero e proprio linguaggio di programmazione. Per esempio se si scrive (con il solito editor di testi, diciamo il Crimson Editor) un file con il seguente contenuto:

```
forvalues i=1/12{
file open fiorellino‘i’ using  dastata‘i’.tex, write replace
file write  fiorellino‘i’ "L’azienda USL ‘i’ "
file write  fiorellino‘i’ "è un’Azienda Sanitaria della Regione Toscana" _n
file close fiorellino‘i’
}
```

e si salva con il nome `programma.do`, dall'interno di *Stata* si potrà comandare interattivamente

```
do programma.do
```

Questo spingerà *Stata* a eseguire i comandi contenuti nel file `programma.do`. In questo esempio il file di programma è un semplice ciclo `for`, che dice al programma di fare 12 volte una cosa, ma a ogni passaggio il simbolo 'i' viene sostituito dal valore che il ha il parametro `i` in quel passaggio: la prima volta 1, la seconda 2, eccetera.[4]. Noto questo non è difficile capire cosa fa questo programma: fa scrivere a *Stata* 12 file, di nome `dastata1.tex`, `dastata2.tex`, `...`e in essi, rispettivamente, gli fa scrivere la frase "L'azienda USL 1 è un'Azienda Sanitaria della Regione Toscana", "L'azienda USL 2 è un'Azienda Sanitaria della Regione Toscana", ...

Detto questo, se invece di usare *Stata* come una segretaria immaginiamo di usarlo come programma statistico, può venirci in mente di caricare un archivio, contenente per esempio i dati di popolazione del territorio regionale, suddiviso per territorio delle Aziende USL. Supponiamo che tale archivio si chiami `popasl.dta`, e che 12 righe con le due variabili, `num_asl` e pop. Stavolta scriviamo il seguente programma.

```
use "popasl.dta"
forvalues i=1/12{
file open fiorellino‘i’ using  dastata‘i’.tex, write replace
file write  fiorellino‘i’ "L’azienda USL ‘i’ "
summarize pop if num_asl==‘i’
local pop‘i’=‘r(min)’
file write  fiorellino‘i’ "ha una popolazione di ‘pop‘i’’ abitanti" _n
```

[4]Nel linguaggio di programmazione di *Stata* il contenuto di una variabile (detta *macro*, vedi nota 5) viene indicato con il nome della variabile stessa circondato da virgolette singole.

```
file close fiorellino‘i’
}
```

In questo caso abbiamo usato altri due comandi di *Stata* :

- `summarize`, che dà delle brevi statistiche (numero di osservazioni valide, minimo, massimo, media, deviazione standard...) delle variabile numeriche elencate come argomento del comando, in questo caso di `pop`; poiché con l'istruzione `if num_ASL==‘i’` gli diciamo di restringere l'analisi solo all'azienda USL numero `‘i’`, *Stata* avrà a che fare con un'unica osservazione, che è appunto la popolazione di quella USL; *Stata* memorizza questo valore in una macro[5] che si chiama `r(mean)`;
- `local`, che genera una macro: in questo caso una macro ha nome `pop‘i’` che contiene il valore `r(mean)`, cioè la popolazione dell'azienda `‘i’`.

Quindi quello che farà la nostra segretaria-programma statistico è generare 12 file, come prima, ma stavolta ci scriverà rispettivamente "L'azienda USL 1 ha una popolazione di 199534 abitanti", "L'azienda USL 2 ha una popolazione di 216546 abitanti",... e questo senza che siamo noi a dover inserire il dato. Se viene modificato l'archivio `popasl.dta` con dati aggiornati, basterà che noi rifacciamo girare lo stesso programma ed esso riscriverà i 12 file con le cifre aggiornate, senza che noi muoviamo alcuna cellula cerebrale.

2.4 Far scrivere automaticamente tabelle a *Stata*

A questo punto il cerchio si chiude: per far generare a *Stata* delle tabelle bisogna in sostanza fargli scrivere dei file tex che contengono i dati e il codice che permette a LaTeX di scriverci delle tabelle.

Una cosa un po' sciocca da fare ma che funziona è provare a far scrivere a *Stata* i testi degli esercizi della prima parte di questa dispensa.

Esempio 14 Per far scrivere a *Stata* la tabella dell'esempio 5 bisogna far eseguire il seguente file di programma:

```
file open colpodigenio using dastata.tex, write replace
file write  colpodigenio "\begin{tabular}{r|c|c|}" _n
file write  colpodigenio "&Maschi&Femmine\\\ \hline" _n
file write  colpodigenio "Giovani&5&14\\\ \hline" _n
file write  colpodigenio "Anziani&15&3\\\ \hline" _n
file write  colpodigenio "\end{tabular}" _n
file close colpodigenio
```

[5] Nel linguaggio di *Stata* le variabili vengono chiamate *macro*, nome che solitamente in programmazione indica un'altra cosa, e cioè piccoli programmi (questo intendiamo noi, per esempio, nella nota 1 a pagina 2). La ragione per cui qui la parola macro cambia senso è che, essendo *Stata* anzitutto un programma statistico, la parola *variabile* ha già un altro significato, cioè quello di *variabile statistica*. Questa ambiguità di terminologia può generare un po' di confusione all'inizio, ma poi ci si fa l'abitudine...

Qui c'è un piccolo inghippo: *Stata* interpreta il simbolo \\ a modo proprio. **Per fargli stampare sul file il simbolo \\ bisogna scriverere \\\\.**

Ma naturalmente la cosa intelligente da fare è far leggere a *Stata* degli archivi o fargli calcolare dei valori a partire dagli archivi, e farglieli stampare in una forma tabellare elegante per mezzo di LaTeX.

Esempio 15 Il programma seguente fa caricare a *Stata* l'archivio `popasl.dta` e gli fa scrivere i dati in forma tabellare, con le righe colorate a colori alterni.

```
use "popasl.dta"
sort num_asl
local col0 ".8"
local col1 ".9"
file open unpofurbo using dastata.tex, write replace
file write  unpofurbo "\begin{tabular}{cr}" _n
file write  unpofurbo "{\bf Azienda}&{\bf Popolazione}\\\ \hline" _n
forvalues i=1/12{
local color‘i’=cond(mod(‘i’,2)==0,‘col0’,‘col1’)
local asl‘i’=num_asl[‘i’]
local pop‘i’=pop[‘i’]
file write  unpofurbo "\rowcolor[gray]{‘color‘i’’}Azienda USL ‘asl‘i’’ & ‘pop‘i’’ \\\ " _n
}
file write  unpofurbo "\end{tabular}" _n
file close unpofurbo
```

Il codice che risulta generato nel file `dastata.tex` è il seguente:

```
\begin{tabular}{cr}
{\bf Azienda}&{\bf Popolazione}\\ \hline
\rowcolor[gray]{.9}Azienda USL 1 & 199534 \\
\rowcolor[gray]{.8}Azienda USL 2 & 216546 \\
\rowcolor[gray]{.9}Azienda USL 3 & 269265 \\
\rowcolor[gray]{.8}Azienda USL 4 & 228027 \\
\rowcolor[gray]{.9}Azienda USL 5 & 317898 \\
\rowcolor[gray]{.8}Azienda USL 6 & 343037 \\
\rowcolor[gray]{.9}Azienda USL 7 & 252799 \\
\rowcolor[gray]{.8}Azienda USL 8 & 321725 \\
\rowcolor[gray]{.9}Azienda USL 9 & 215445 \\
\rowcolor[gray]{.8}Azienda USL 10 & 797058 \\
\rowcolor[gray]{.9}Azienda USL 11 & 216501 \\
\rowcolor[gray]{.8}Azienda USL 12 & 158557 \\
\end{tabular}
```

La tabella risultante è la tabella 2 a pagina 17.

Tabella 2: Una tabella a righe colorate generata da *Stata* a partire da un archivio

Azienda	**Popolazione**
Azienda USL 1	199534
Azienda USL 2	216546
Azienda USL 3	269265
Azienda USL 4	228027
Azienda USL 5	317898
Azienda USL 6	343037
Azienda USL 7	252799
Azienda USL 8	321725
Azienda USL 9	215445
Azienda USL 10	797058
Azienda USL 11	216501
Azienda USL 12	158557

Esempio 16 In questo caso usiamo le potenzialità di *Stata* e di LaTeX in modo un po' più furbo: il comando `cond` di *Stata* permette di attribuire a una macro un valore se una condizione è verificata e un altro se la condizione non è verificata. Nell'esempio la macro viene posta nella tabella subito prima del valore della popolazione, e contiene la stringa `\bf` se la popolazione è maggiore della media e la stringa vuota altrimenti.

```
use "popasl.dta"
sort num_asl
quietly summarize pop
local mean=r(mean)
file open piufurbo using dastata.tex, write replace
file write  piufurbo "\begin{tabular}{c|r}" _n
file write  piufurbo "{\bf Azienda}&{\bf Popolazione}\\\ \hline" _n
forvalues i=1/12{
local bf‘i’=cond(pop[‘i’]>‘mean’,"\bf","")
local asl‘i’=num_asl[‘i’]
local pop‘i’=pop[‘i’]
file write  piufurbo "Azienda USL ‘asl‘i’’ & { ‘bf‘i’’ ‘pop‘i’’} \\\ " _n
}
file write  piufurbo "\end{tabular}" _n
file close piufurbo
```

Quello che otteniamo è una tabella in cui la popolazione delle aziende con popolosità maggiore della media è scritta in grassetto. Il codice LaTeX generato da *Stata* è il seguente:

```
\begin{tabular}{c|r}
{\bf Azienda}&{\bf Popolazione}\\ \hline
Azienda USL 1 & {  199534} \\
Azienda USL 2 & {  216546} \\
Azienda USL 3 & {  269265} \\
```

Tabella 3: Una tabella in cui *Stata* e LaTeX evidenziano automaticamente alcuni valori

Azienda	**Popolazione**
Azienda USL 1	199534
Azienda USL 2	216546
Azienda USL 3	269265
Azienda USL 4	228027
Azienda USL 5	**317898**
Azienda USL 6	**343037**
Azienda USL 7	252799
Azienda USL 8	**321725**
Azienda USL 9	215445
Azienda USL 10	**797058**
Azienda USL 11	216501
Azienda USL 12	158557

```
Azienda USL 4 & {  228027} \\
Azienda USL 5 & { \bf 317898} \\
Azienda USL 6 & { \bf 343037} \\
Azienda USL 7 & {  252799} \\
Azienda USL 8 & { \bf 321725} \\
Azienda USL 9 & {  215445} \\
Azienda USL 10 & { \bf 797058} \\
Azienda USL 11 & {  216501} \\
Azienda USL 12 & {  158557} \\
\end{tabular}
```

e la tabella risultante è la tabella 3 a pagina 18.

2.5 Un esempio più complesso

A questo punto abbiamo accennato a tutti gli strumenti che possono essere utili per generare automaticamente tabelle. Gli esempi che abbiamo indicato finora sono piuttosto banali, e i risultati che si ottengono potrebbero essere replicabili con altri strumenti o manualmente senza troppa fatica.

Ma esistono obbiettivi che sono perseguibili con queste tecniche e *non facilmente raggiungibili* con altri strumenti già noti. Descriviamo un esempio.

Esempio 17 Vogliamo riportare in formata tabellare, in appendice a una pubblicazione, i dati da cui siamo partiti per generare 28 grafici presentati nella pubblicazione medesima. I dati sono contenuti in un archivio, `grezzi.dta`, che contiene i seguenti campi:

`anno`	Contiene l'anno cui si riferiscono i dati
`asl_num`	Contiene il codice dell'Azienda USL ai cui residenti si riferiscono i dati
`eta`	Contiene la classe di età dei soggetti cui si riferiscono i dati (4 classi in tutto)
`indic`	Contiene il codice dell'indicatore cui si riferiscono i dati
`ass`	Contiene il numero dei soggetti cui la prestazione sanitaria descritta dall'indicatore è stata assicurata
`nass`	Contiene il numero dei soggetti cui la prestazione sanitaria descritta dall'indicatore non è stata assicurata

Gli indicatori sono 14, quindi generare le 14 tabelle manualmente (per esempio selezionando i dati via SQL e copiando le tabelle ottenute su un word-processor tipo MS Word su cui poi regolare manualmente l'aspetto estetico delle tabelle) sarebbe un lavoro molto faticoso.

Inoltre l'archivio è a sua volta generato automaticamente con una cascata di query dai flussi di dati amministrativi regionali: quando i flussi verranno aggiornati il prossimo anno noi vorremmo non dover rifare tutto il lavoro, bensì avere una procedura già automatica che, a partire dall'archivio generato dalla cascata di query, generi le nuove 14 tabelle.

Scriviamo quindi un file `genera_tabelle.do` che apre l'archivio e che per ogni valore della variabile indicatore genera un file tex contenente il codice della tabella corrispondente. Usando il linguaggio delle macro di *Stata* possiamo facilmente associare a ogni indicatore un titolo, e la descrizione della prestazione, in modo che la tabella sia di facile lettura. Poi scriviamo un brevissimo file tex principale in cui includiamo in 14 file.

Non scriviamo qui il codice completo del programma `genera_tabelle.do`, ma la struttura sarà approssimativamente la seguente:

```
use grezzi.dta
/* definisce gli anni entro cui vogliamo rappresentare i dati*/
summarize anni
local anmin=r(min)
local anmax=r(max)
/* definisce i titoli delle 14 tabelle*/
local titolo1="Fratturati al femore operati entro 24 ore"
...
/* definisce le prestazioni in ciascuna delle 14 tabelle*/
local primo1="Gravemente antipatici"
local secondo1="Operati oltre le 24 ore"
...
/* scrive i 14 file*/
forvalues i=1/14{
file open indic‘i’ using tabella‘i’.tex
file write indic‘i’ "begin{table}\caption{‘titolo‘i’’}"_n
file write indic‘i’ "begin{tabular}{r|"
/*per ogni anno scrive quattro colonne, una per ogni classe d’età*/
forvalues j=‘anmin’/‘anmax’{
file write indic‘i’ "cccc|"
}
file write indic ‘i’ "}"_n
file write indic ‘i’ "&&"_n
/*scrive l’anno su quattro colonne, una per ogni classe d’età*/
```

```
forvalues j=‘anmin’/‘anmax’{
file write indic‘i’ "\multicolumn{4}{c}{‘j’}"
}
file write indic‘i’ "\\\ \hline"_n
*scrive scrive le 4 classi d’etè*/
file write indic‘i’ "&&"_n
forvalues j=‘anmin’/‘anmax’{
forvalues k=1/4{
file write indic‘i’ "Classe‘k’"
}
}
file write indic‘i’ "\\\ \hline"_n
/*poi scrive i dati: per ogni azienda, per ogni anno e per ogni classe d’età*/
forvalues s=1/12{
file write indic‘i’ "‘primo‘i’’&"
forvalues j=‘anmin’/‘anmax’{
forvalues k=1/4{
summarize ass if indic==‘i’ & anno==‘j’ & eta=‘k’
local e=r(mean)
file write indic‘i’ "‘e’&"
}
}
}
/*e così via...*/

}
```

Con un codice simile a quello elencato si generano automaticamente le 14 tabelle. Un esempio di tabella generata è riportato nella tabella 4 a pagina 22.

3 Conclusioni

Per sfruttare le potenzialità cui abbiamo fatto cenno in questo documento è necessario approfondire il linguaggio di programmazione di LaTeX e quello di *Stata* . Come sempre per imparare un liguaggio il consiglio più utile è mettersi a fare qualcosa con una consulenza amichevole a portata di mano. Tuttavia qualche risorsa bibliografica e il ricorso ad alcuni siti può essere preziosissimo. Per non parlare del fatto che sia *Stata* che LaTeX hanno fiorenti servizi di consulenza: *Stata* , che è un programma commerciale, ha per i suoi utenti un servizio che risponde in 24 ore; LaTeX, che è libero e gratuito, ha una comunità ricchissima, con siti che danno risposte a quesiti specifici, anche se in termini amichevoli e non commerciali (e quindi senza vincoli di tempo). Per ulteriori informazioni consultare il sito ufficiale del Gruppo Utilizzatori Italiani di TeX e LaTeX

http://www.guit.sssup.it

e il sito ufficiale di *Stata*

http://www.stata.com

Riferimenti bibliografici

[GMS94] Michel Goossens, Frank Mittelbach and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley Publishing Co. 1994.

[LL94] Leslie Lamport. *LaTeX– A Document Preparation System*. Addison-Wesley Publishing Co., 2nd edition, 1994.

Azienda	Anno	2000				2001				2002				**Totali azienda**			
		45-64	65-74	75-84	85+	45-64	65-74	75-84	85+	45-64	65-74	75-84	85+	45-64	65-74	75-84	85+
1	Gravemente antipatici	91	73	52	22	82	53	57	18	166	122	104	18	421	314	274	82
	Non gravemente antipatici	78	68	74	49	72	70	81	59	174	154	176	50	368	357	390	192
2	Gravemente antipatici	74	74	56	27	84	77	64	32	190	154	194	34	431	372	374	122
	Non gravemente antipatici	55	68	50	30	44	57	46	34	120	126	178	44	284	303	320	130
3	Gravemente antipatici	122	87	89	40	140	96	84	43	234	204	200	37	622	477	468	155
	Non gravemente antipatici	39	32	40	26	22	14	39	23	76	70	86	23	167	146	207	93
4	Gravemente antipatici	59	50	64	31	87	65	69	16	168	100	144	20	400	277	343	86
	Non gravemente antipatici	45	29	47	24	29	35	45	30	74	60	100	27	168	152	233	100
5	Gravemente antipatici	104	90	80	43	82	80	76	28	248	212	206	38	539	469	437	135
	Non gravemente antipatici	60	79	64	55	81	73	87	50	140	132	192	46	349	329	411	184
6	Gravemente antipatici	65	50	67	20	140	131	125	42	388	304	366	68	604	490	563	131
	Non gravemente antipatici	97	103	158	74	55	51	111	61	100	154	282	102	413	463	710	325
7	Gravemente antipatici	23	26	32	17	34	63	52	28	94	130	120	33	176	239	221	88
	Non gravemente antipatici	86	85	113	60	68	81	89	69	152	154	190	57	401	439	507	233
8	Gravemente antipatici	109	101	80	30	97	75	83	30	146	138	136	23	439	397	383	105
	Non gravemente antipatici	58	66	69	43	60	81	80	45	150	166	118	44	324	375	317	161
9	Gravemente antipatici	60	54	65	11	87	82	74	33	156	150	194	30	364	335	380	92
	Non gravemente antipatici	66	70	74	38	67	92	99	63	126	138	248	54	332	376	512	189
10	Gravemente antipatici	162	141	128	53	169	116	125	52	326	268	300	51	799	675	664	198
	Non gravemente antipatici	210	227	252	178	254	216	269	189	506	504	680	230	1178	1151	1425	744
11	Gravemente antipatici	91	76	71	31	75	76	70	25	158	184	176	24	407	423	393	114
	Non gravemente antipatici	19	28	22	24	22	36	32	15	48	56	78	18	117	151	167	80
12	Gravemente antipatici	7	5	2	1	12	6	9	3	24	18	10	2	51	38	29	9
	Non gravemente antipatici	74	80	99	39	84	85	96	48	194	150	214	54	443	412	496	185
Totali	Gravemente antipatici	**967**	**827**	**786**	**326**	**1089**	**920**	**888**	**350**	**2298**	**1984**	**2150**	**378**	**5253**	**4506**	**4529**	**1317**
	Non gravemente antipatici	**887**	**935**	**1062**	**640**	**858**	**891**	**1074**	**686**	**1860**	**1864**	**2542**	**749**	**4544**	**4654**	**5695**	**2616**

Tabella 4: esempio di grossa tabella generata automaticamente (i dati sono fittizi)