# R (autore: Vittorio Albertoni)

## Premessa

Viene generalmente catalogato tra i linguaggi di programmazione ma io, come molti altri, preferisco vederlo come un ambiente, un environment, all'interno del quale possiamo fare soltanto determinate cose: nel caso specifico calcoli statistici di qualsiasi tipo.

Dal momento che per fare queste cose occorre richiamare funzioni che hanno un nome ed una sintassi di uso, hanno ragione anche quelli che lo vedono come un linguaggio.

Sta di fatto che, se con i veri e propri linguaggi di programmazione detti all purpose, come C, Pascal, Pyton, Java, Julia, LISP, ecc., possiamo fare tutto ciò che vogliamo, con R possiamo solo fare calcoli, soprattutto calcoli statistici.

Con il vantaggio di poterli fare in maniera generalmente più semplice che non ricorrendo ai linguaggi all purpose.

Dobbiamo tuttavia riconoscere che Python e Julia vantano strumenti che nulla hanno da invidiare a R, come semplicità di uso e performance, in tema di analisi dei dati e di calcoli statistici.

La stessa semplicità non si ritrova in altri linguaggi, soprattutto se andiamo ai tempi in cui l'unico linguaggio disponibile per programmare calcoli era il Fortran.

E' per questo motivo che a metà anni '70 dello scorso secolo, nei Bells Laboratories, venne sviluppato il linguaggio statistico S, reso disponibile al di fuori dei Bells Laboratories nel 1980 e tuttora disponibile nella versione S plus, con licenza proprietaria.

E' da questo linguaggio che, nel 1996, lo statistico neozelandese Ross Ihaka e il matematico e statistico canadese Robert Gentleman hanno derivato R, da loro definito "a language for data analysis and graphics", rendendolo software libero open source.

Per l'analisi statistica, il calcolo numerico, la manipolazione dei dati e la visualizzazione grafica non c'è di meglio, salvo ricorrere, come ho detto, a linguaggi all purpose come Python e Julia, il cui uso sta soppiantando quello di R.

In questo manualetto mi propongo semplicemente di mostrare a grandi linee il funzionamento di R, con rimandi a strumenti che meglio di quanto possa fare io descrivono nello specifico la sintassi dei vari comandi.

# Indice

# 1 Installazione

All'indirizzo *https://cran.mirror.garr.it/CRAN/* troviamo i file per installare R sul nostro computer. Essi sono disponibili per i sistemi operativi Linux, Windows e Mac. Chi usa Linux può trovare quanto serve anche nel repository della sua distro.

L'installazione che avviene utilizzando questi file è già abbastanza ricca e comprende il package base, il package per la grafica e il package per il dataset.

Per le cose più sofisticate abbiamo a disposizione molti altri packages. All'indirizzo
*https://cran.r-project.org/web/packages/available_packages_by_name.html*
ne troviamo l'elenco e, cliccando sul nome che ci interessa, veniamo indirizzati a pagine che illustrano il singolo pacchetto, con indicazioni sulle modalità di uso.

L'installazione di un pacchetto avviene scrivendo il comando
`install.packages("<nome_pacchetto>", dependencies=TRUE)`
nella shell di R.
Dando invio dopo avere scritto il comando `install.packages()` senza indicare gli argomenti apriamo un elenco dal quale possiamo scegliere il pacchetto da installare.

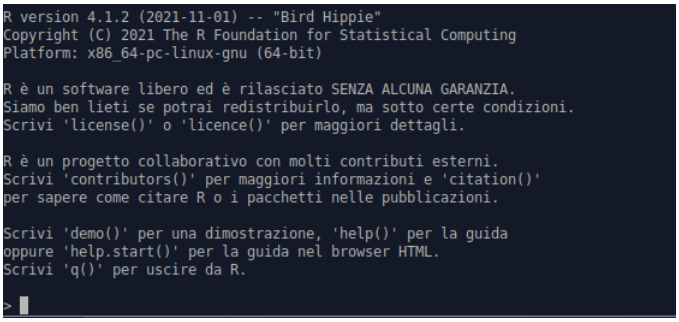
Per la disinstallazione di un pacchetto usiamo il comando
`remove.packages("<nome_pacchetto>")`.

Con il comando
`library()`
possiamo vedere l'elenco dei pacchetti installati.

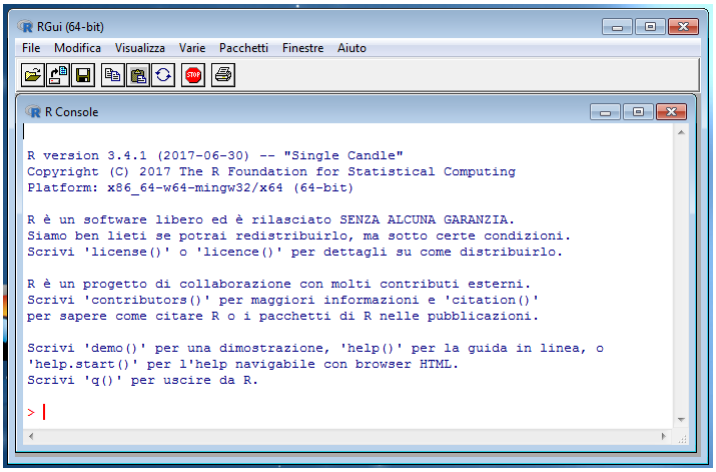
# 2 Come funziona

Per lavorare con R si utilizza normalmente il terminale. Quello che in Linux e Mac si chiama proprio Terminale e che in Windows è sostituito dal così detto Prompt dei comandi, che non è utilizzabile per R.

Se lavoriamo su Linux o Mac, scrivendo a terminale il comando `R` si apre, nel terminale stesso, la shell di R, altrimenti detta console.




Se lavoriamo su Windows abbiamo a disposizione un launcer per aprire la R GUI



che contiene la console in un ambiente arricchito da un menu e da una barra di strumenti per fare alcune cose, tra cui, molto comoda, la voce PACCHETTI, che ci aiuta a installare i pacchetti.

In corrispondenza al prompt > si scrivono i comandi.

Ci viene ricordato che con il comando `q()` usciamo da R. Possiamo anche farlo con `ctrl-D`.

Ci vengono altresì suggeriti i comandi `demo()` e `help()`, convenientemente sostituibile da `help.start()`, molto utili per familiarizzare con R.
Con `demo()` ci vengono elencate le demo disponibili e per accedere alla demo dobbiamo indicarne il nome tra le parentesi.
Con `help()` ci viene spiegato come funziona l'help e inserendo il nome di un comando tra le parentesi ci viene spiegato il comando.
Con `help.start()` il sistema di help viene trasferito nel nostro browser web predefinito, dove, tra l'altro, troviamo un ottimo manuale introduttivo a R in lingua inglese.

Altro simpatico comando di didattica è `example()`. Se all'interno delle parentesi scriviamo il nome di un comando ci vengono proposte esemplificazioni del suo utilizzo.

Una volta scritto il comando in corrispondenza al prompt, alla pressione del tasto INVIO vediamo comparire il risultato fornito dal comando stesso.

Possiamo indicare più comandi sulla stessa riga separandoli con un punto e virgola (`;`) e alla pressione del tasto INVIO verranno eseguiti uno via l'altro mostrando i risultati.

Possiamo anche inserire nella shell di R uno script, cioè una serie di comandi che formano un programma, racchiudendo i comandi stessi da parentesi graffe ( `{` e `}` ) e scrivendoli su righe separate.

Infine possiamo produrre uno script scrivendone i comandi con un editor di testo e salvare il tutto in un file con estensione `.R`. Lo script potrà essere eseguito quante volte vogliamo con il comando
`source("<file.R>")`
in corrispondenza al prompt > nella shell di R.
Se non siamo nella directory dove è stato salvato il file dobbiamo indicare tra gli apici il path al file stesso.

* * *

A questo punto subentra la necessità di conoscere i comandi da inserire nella shell o nello script e, per non ripetere inutilmente cose da altri scritte meglio di come possa fare io, suggerisco i seguenti due indirizzi web:

*https://www.agnesevardanega.eu/wiki/r/start*

Vi troviamo un percorso completo per l'apprendimento di ciò che si può fare con R, proposto in lingua italiana a principianti niente meno che da una insegnante, Agnese Vardanega (meglio di così...) e fatto molto bene.
I contenuti che troviamo qui sono gli stessi proposti in un testo della stessa autrice «Ricerca sociale con R. Concetti e funzioni di base per l'analisi esplorativa dei dati», pubblicato attraverso StreetLib e reperibile in tutte le librerie on-line.

*cran.r-project.org › doc › contrib › Frascati-FormularioStatisticaR*

Vi troviamo il catalogo completo delle funzioni statistiche, con l'indicazione del package di appartenenza, della formula sottostante la funzione, con la descrizione dell'output e con alcune esemplificazioni di utilizzo.

Molto altro si trova sul web su R, in lingua inglese ma spesso anche in lingua italiana.

## 3 RStudio

La modalità di lavorare con R in console vista nel precedente capitolo è la più ruspante e richiede una completa padronanza dei comandi e delle funzioni, dato che, mentre si lavora in

console, diventa scomodo consultare l'help, salvo avere a disposizione un supporto cartaceo di consultazione.

Per agevolare non solo i principianti ma anche i professionisti c'è stato chi ha inventato una interfaccia grafica per lavorare con R, che si chiama RStudio.

Si tratta di una IDE (Integrated Development Environment) sviluppata a partire dal 2010 e rilasciata, anche con licenza libera, in prima versione il 1 novembre 2016.

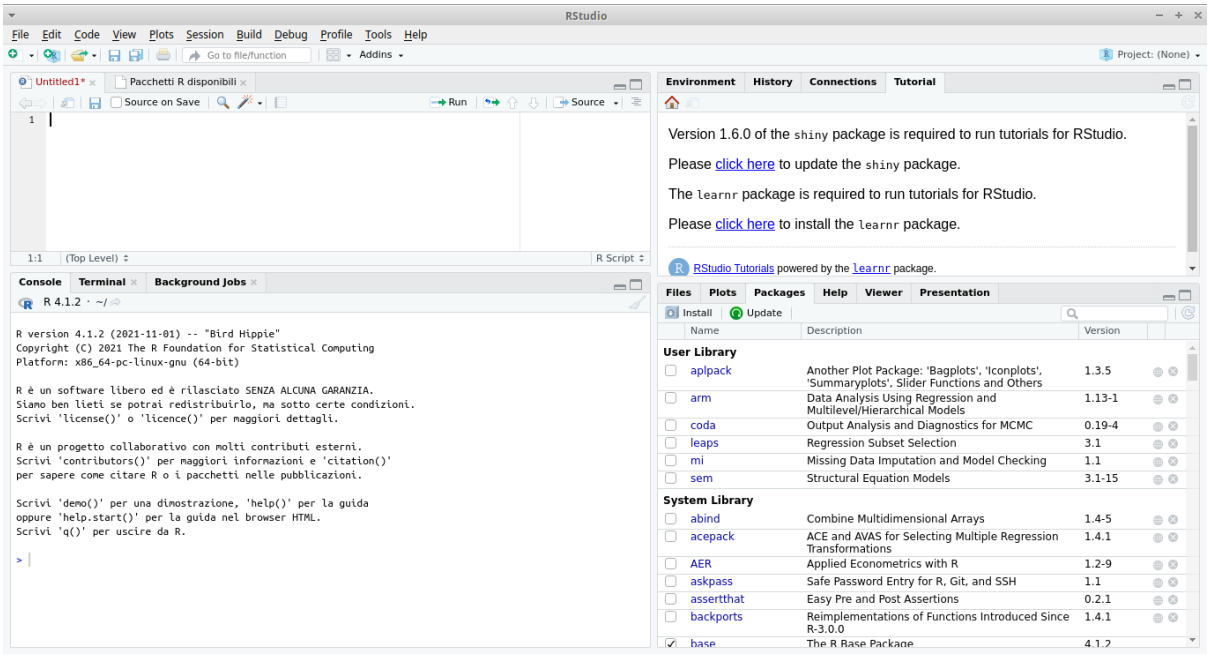
Il suo uso presenta il grande vantaggio del controllo e dell'auto-completamento dei comandi che scriviamo, dell'immediatezza nell'accesso e nella consultazione dell'aiuto sulle funzioni e nella facilitazione nella gestione di oggetti e file di dati dall'area di lavoro.

Troviamo RStudio all'indirizzo *www.rstudio.com* e, se vogliamo risparmiare, dalla pagina DOWNLOAD possiamo scegliere RSTUDIO DESKTOP OPEN SOURCE LICENSE FREE DOWNLOAD e scaricare gli installer per il nostro sistema operativo (Linux, Windows o Mac).

Possiamo apprezzare il valore del software se diamo un'occhiata al prezzo dell'abbonamento annuale per le versioni commerciali corredate di vari livelli di assistenza e garanzia.

La versione gratuita libera è sempre rilasciata SENZA ALCUNA GARANZIA ma funziona bene pure lei se siamo in grado di farla funzionare, o solo con la nostra capacità o con l'aiuto di volontari appassionati che possiamo trovare sul web.

La finestra di lavoro di RStudio si presenta così




e, sotto la barra dei menu e degli strumenti, abbiamo quattro quadranti.

La barra dei menu è autoesplicativa.

Scorrendo il mouse sulle icone della sottostante barra degli strumenti abbiamo la descrizione di ciò che avviene cliccando sull'icona.

Il quadrante in basso a sinistra può aprirsi su tre schede.
Per default si apre sulla prima, denominata CONSOLE, che altro non è che la shell di cui abbiamo parlato nel precedente capitolo e come quella funziona.
Cliccando sulla linguetta di intestazione della seconda, denominata TERMINAL, apriamo il nostro terminale, dandoci la possibilità di fare altre cose intanto che lavoriamo su R.
Cliccando sulla linguetta di intestazione della terza, denominata BACKGROUND JOBS, apriamo una finestra nella quale possiamo lanciare elaborazioni in background. Il lancio avviene cliccando sul pulsante in alto a sinistra della finestra START BACKGROUND JOB e scegliendo il job da lanciare utilizzando la finestra di dialogo che si apre.

Il quadrante in alto a sinistra è un editor di script e vi possiamo inserire una serie di comandi per realizzare un progetto complesso.

E' dotato di una barra degli strumenti sulla quale abbiamo due pulsanti per eseguire lo script, in tutto o in parte.
Cliccando sul pulsante con la scritta RUN eseguiamo il contenuto della riga corrente o della parte selezionata dello script.
Cliccando sul pulsante con la scritta SOURCE eseguiamo tutto lo script.
Il risultato dell'elaborazione di parte o di tutto lo script compare nella sottostante CONSOLE.

Quando scriviamo i comandi in una di queste due finestre, Console e Editor, siamo aiutati dalla code completion e dalla comparsa di un help che ci guida all'inserimento dei parametri richiesti dalla funzione che stiamo attivando.

Nel quadrante in alto a destra si possono aprire quattro finestre.
La prima è denominata ENVIRONMENT ed è molto utile. La sua barra degli strumenti contiene alcune icone che si descrivono al passaggio del mouse. Aprendo quella denominata IMPORT DATASET abbiamo modo di organizzare l'importazione di dati da varie fonti. Sotto la barra degli strumenti abbiamo due pulsanti: il primo ci dà modo di scegliere se rimanere in R, posizione di default, o se dobbiamo collegarci a oggetti Python; il secondo ci dà modo di scegliere se il contenuto dell'environment debba essere evidenziato in modo globale, scelta di default, o limitatamente a un package. Nella zona sottostante verrà via via evidenziato ciò che succede nell'ambiente (environment) in cui stiamo lavorando: variabili create e loro valore, ecc.
La seconda è denominata HISTORY e vi si registra tutto ciò che facciamo.
La terza è denominata CONNECTIONS e serve per organizzare connessioni con basi di dati per alimentare il dataset.
La quarta è denominata TUTORIAL e serve per accedere a materiale educativo su R.

Abbiamo infine il ricco quadrante in basso a destra che si può aprire su sei finestre.
La prima, denominata FILES, ci dà modo di lavorare sul nostro file system per creare, spostare, cancellare files.
La seconda, denominata PLOTS, è dedicata a mostrare eventuali grafici prodotti nel corso del nostro lavoro.
La terza, denominata PACKAGES, elenca tutti i packages che abbiamo a disposizione: quelli che sono preceduti da un segno di spunta sono direttamente utilizzabili e quelli che sono preceduti da un quadratino vuoto vanno preventivamente caricati cliccando sul pulsante INSTALL (se produciamo uno script è bene che carichiamo il package che ci serve attraverso il comando `library(nome_pacchetto)` nella prima riga dello script stesso).
La quarta, denominata HELP, viene utilizzata per mostrare istruzioni sull'uso dei vari comandi e funzioni quando richiamate in approfondimento dell'help collegato alla code completion.
Le due rimanenti riguardano sofisticazioni per professionisti esperti.

Quello qui presentato è RStudio funzionante su un sistema Linux e chi lavora su Windows può trovare qualche differenza: la più vistosa è la mancanza della scheda Terminale nel quadrante in basso a sinistra (il Terminale esiste infatti solo in Linux e Mac).

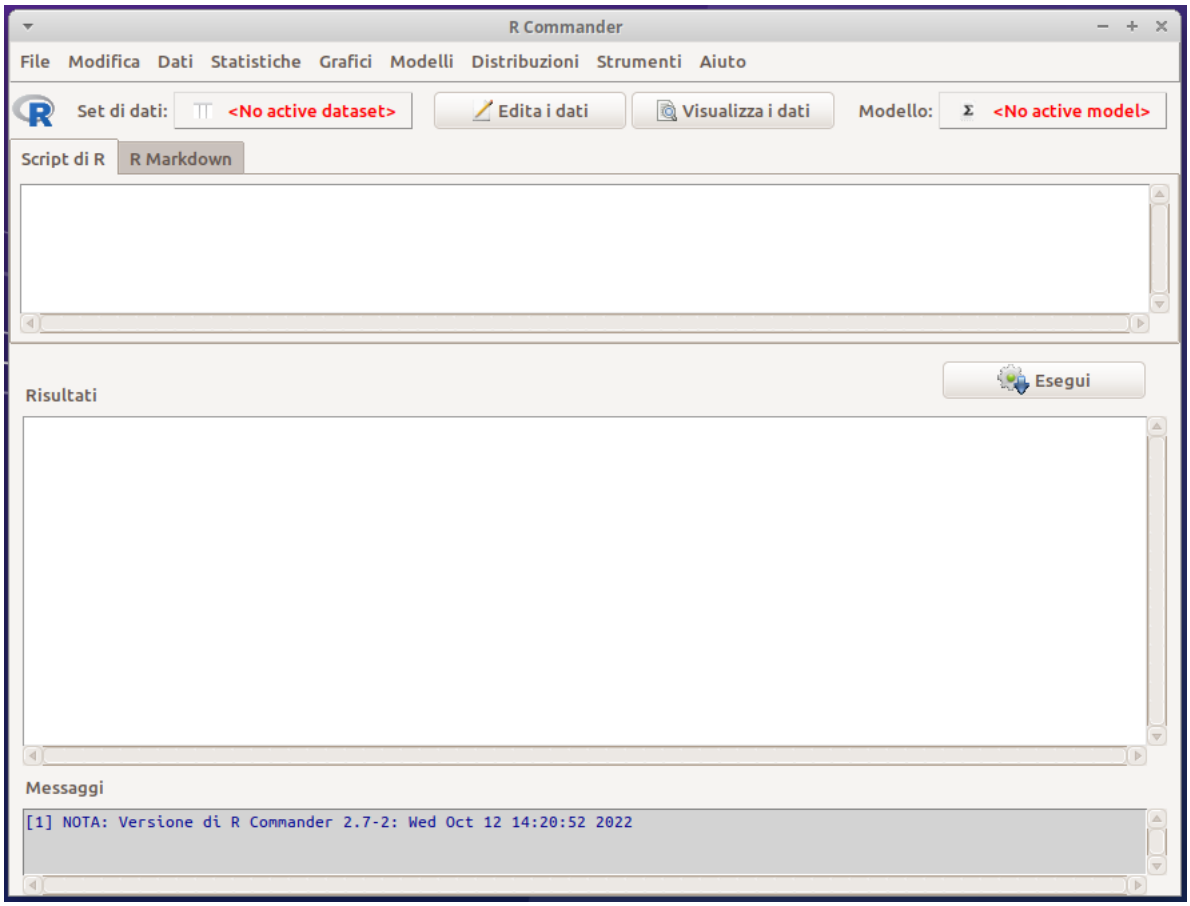
# 4 R Commander

R Commander è una interfaccia grafica che si propone di consentire di lavorare con R, almeno di fare con R tantissime cose, senza conoscerne i comandi e la relativa sintassi.

Si tratta di un package di R, che va ovviamente installato dopo che è stato installato R, con il comando
`install.packages("Rcmdr", dependencies=TRUE)`.
Chi lavora su Windows può aiutarsi con il menu PACCHETTI ▷ INSTALLA pacchetti della R Gui.
Chi lavora su Linux trova molto probabilmente il pacchetto nel repository della sua distro con il nome `rcmdr` e lo può installare con il gestore dei pacchetti.
Chi lavora su Mac deve innanzi tutto verificare che sul suo computer sia installato il sistema a finestre X11 (nella cartella UTILITY in APPLICAZIONI compare il file `X11.app`). Se X11.app

manca, è possibile installarlo dal disco di installazione di Mac OS X. Inoltre è necessario successivamente installare Tcl/Tk procurandosi il file `TclTk-8.5.5-x11.dmg` all'indirizzo *https://cran.r-project.org/bin/macosx/tools/*.
Infine si può procedere all'installazione di R Commander come visto fare per Linux e Windows.

La finestra di lavoro di R Commander è la seguente




Abbiamo una barra di menu autoesplicativa e, sotto, una barra di strumenti altrettanto autoesplicativa.
Ancora sotto abbiamo due finestre, una sopra l'altra. La prima, per default aperta sulla scheda SCRIPT DI R, la seconda dedicata ai RISULTATI.
Tra le due un pulsante ESEGUI, cliccando sul quale viene eseguito il contenuto della riga corrente nella finestra dello script o quanto selezionato dello script stesso, con evidenziazione di quanto fatto e del risultato nella finestra dei risultati.

Per esempio, se scriviamo nella finestra dello script la più elementare delle istruzioni

```
3 + 5
```

e con il cursore posizionato sulla stessa riga o avendo selezionato la riga stessa clicchiamo su ESEGUI, vedremo comparire nella finestra dei risultati quanto segue

```
> 3 + 5
[1] 8
```

cioè l'operazione svolta e il suo risultato, come se avessimo operato nella console.

In questo modo possiamo operare come se non utilizzassimo R Commander, scrivendo nella finestra dello script come se fossimo in R Studio.

Per lavorare con R Commander dobbiamo necessariamente avere caricato un set di dati: al lancio di R Commander notiamo nella finestrella SET DI DATI la scritta in rosso `<No active dataset>` e se scorriamo i menu troviamo la maggior parte delle voci disattivate.

Il caricamento dei dati avviene agendo sul menu DATI che ci dà modo di generare un set di dati o di importarlo (l'importazione da Excel richiede proprio un file Excel e se lavoriamo con Open Office o Libre Office dobbiamo salvare il file in quel formato per poterlo utilizzare).

A dati caricati tutte le voci dei menu sono attivate e possiamo eseguire i nostri calcoli avvalendoci di esse. Provvede R Commander a tradurre le nostre scelte in comandi riconosciuti, a compilare lo script, mostrandolo nella relativa finestra, e ad eseguirlo.

La citata guida a R della prof. Agnese Vardanega contiene un capitolo dedicato a R Commander.

Indubbiamente questo software alleggerisce di molto la fatica necessaria per usare R e consente di operare anche senza essere conoscitori dei vari comandi e della relativa sintassi ma, a volte, combina qualche scherzo e bisogna ricorrere a rimedi che comportano questa conoscenza.

Se, per esempio, abbiamo la seguente tabella, in un file `clientela.csv`, che evidenzia il numero dei clienti della nostra azienda nelle quattro città in cui opera

| citta | clienti |
|---|---|
| Milano | 780 |
| Roma | 458 |
| Torino | 413 |
| Genova | 342 |

e vogliamo descrivere la distribuzione dei clienti in un grafico a torta utilizzando R Commander, dobbiamo importare i dati agendo da menu DATI ▷ IMPORTA DATI ▷ DA FILE DI TESTO... e scegliere il file clientela.csv.

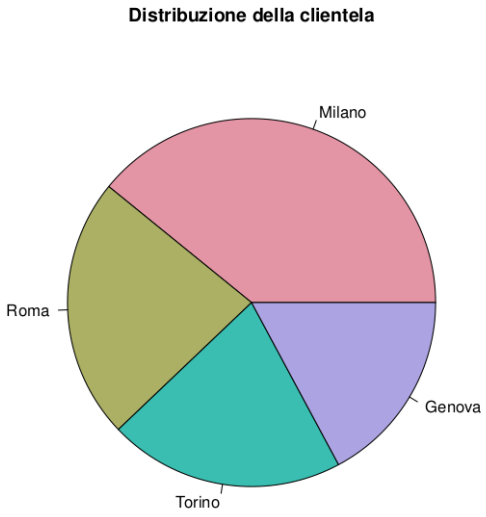
Agendo da menu GRAFICI ▷ GRAFICO A TORTA... siamo forzati a generare il seguente comando

```
with(Dataset, piechart(citta, xlab="", ylab="", main="Distribuzione della clientela", col=rainbow_hcl(4), scale="percent"))
```

che non serve a nulla e genera un grafico insignificante.
Sapendo come funziona, sostituiamo a quel comando il seguente

```
with(clientela, pie(clienti, labels = citta, main="Distribuzione della clientela", col=rainbow_hcl(4), scale="percent"))
```

ed otteniamo il grafico desiderato




Peccato! Perché quello della generazione di grafici è un grande punto di forza di R e proprio qui si trova una topicca di R Commander; spero non ce ne siano tante altre. Ciò serve comunque a dimostrare che gli automatismi spinti a volte creano qualche problema.

Tra l'altro avverto che R Commander è un software molto pesante e occupa quasi un Giga di spazio su disco.

## 5 Spigolature

Come ho detto, sul web esiste una vasta documentazione su R, anche in lingua italiana. Quella che ho richiamato specificamente nel Capitolo 2 è più che sufficiente per acquisire dimestichez-

za con questo software, avendo ovviamente anche la preparazione matematica e statistica per sapere che cosa si sta facendo.

Esplorando qua e là ho tuttavia notato scarsità di sottolineatura di alcune cosette abbastanza importanti che possiamo fare con R: la prima riguarda la facilità con sui possiamo creare grafici di funzioni matematiche, la seconda riguarda la possibilità di conferire interfaccia grafica a R attraverso il linguaggio Tcl/Tk (l'interfaccia di R Commander che abbiamo visto nel precedente capitolo è programmata in Tcl/Tk).

## 5.1 Grafici di funzione

La funzione `plot()` è delegata a produrre gran parte della vasta quantità di tipi di grafico che costituiscono uno dei punti di forza di R e il suo utilizzo richiede la conoscenza del significato di una miriade di parametri attraverso i quali ottenere grafici e loro caratteristiche.

Essa può essere utilizzata in modo facile, con pochi parametri, per creare grafici di funzione con la seguente sintassi:

```
plot(funzione, ascissa_iniziale, ascissa_finale, col = "...", main = "..."
```

in cui

`funzione` è il nome della funzione,

`ascissa_iniziale` e `ascissa_finale` indicano il campo di riferimento del grafico,

`col` indica il colore del grafico,

`main` indica il titolo del grafico,

solo i primi tre parametri sono necessari.

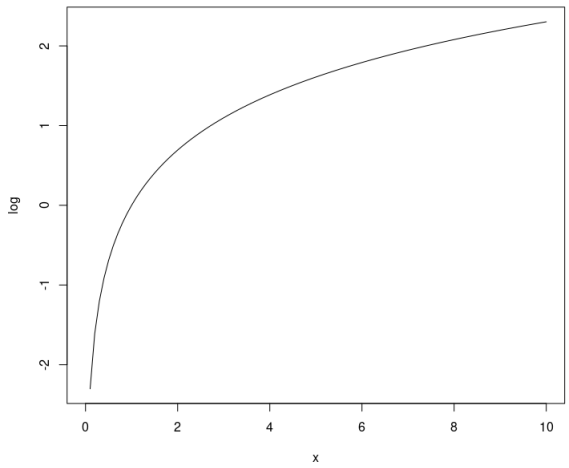
La funzione deve avere un nome per poterla richiamare come parametro.

Se la funzione è una di quelle preconfezionate in R, come sin, cos, log, ecc. abbiamo già il nome per richiamarla.

Il comando

```
plot(log,0,10)
```

produce questo grafico




Se la funzione non è tra quelle conosciute da R la creiamo noi assegnandola a un oggetto richiamabile con un nome.

La sintassi per creare l'oggetto funzione è

<nome> `= function(x){`

    <espressione>

    `}`

dove

<nome> è il nome dell'oggetto funzione

<espressione> è la $f(x)$ di $y = f(x)$.

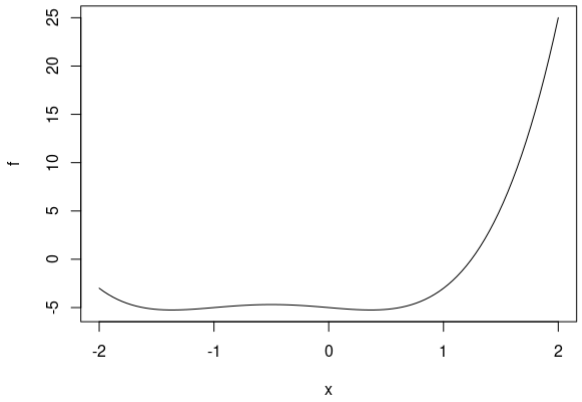
Se vogliamo tracciare il grafico della funzione
$y = x^4 + 2x^3 - x - 5$
creiamo la funzione `f` con

```
f = function(x){
    x^4+2*x^3-x-5
    }
```

e la tracciamo con

```
plot(f,-2,2)
```

ottenendo il grafico




Possiamo aggiungere una griglia al grafico utilizzando, insieme alla funzione `plot()` la funzione `grid()`, che ha la seguente sintassi

```
grid(nx=NULL, ny=NULL, col="...").
```

Con il seguente script

```
f <- function(x){
   x^3-5*x-3
   }
plot(f, -2, 2, col="red", main="Grafico di y = x^3-5x-3")
grid(nx=NULL, ny=NULL, col="green")
```

otteniamo il grafico

## 5.2 Interfaccia grafica

Per quanto si possa considerare superfluo e inutilmente laborioso, R è attrezzato per potersi arricchire di interfaccia grafica, la così detta GUI.

A questo scopo esiste il package `tcltk` che utilizza il linguaggio di programmazione Tcl/Tk[1].

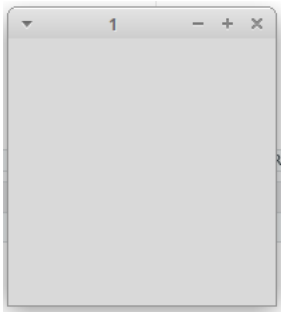
Un bell'esempio di applicazione R con GUI costruita con tcltk è R Commander, che abbiamo visto nel precedente capitolo.

Il package tcltk è presente nella dotazione di base di R ma deve essere caricato per essere utilizzato (è uno di quelli che nell'elenco gestito da RStudio ha il quadratino senza spunta).

Il suo utilizzo comporta ovviamente la conoscenza del linguaggio Tcl/Tk, anche se la sintassi dei vari comandi nell'embedding con R è alquanto diversa.

La finestra radice si crea con lo script

```
library(tcltk)
r = tktoplevel()
```




Aggiungendo allo script le istruzioni

```
tkwm.title(r,"PROVA")
f = tkframe(r, width=200, height=100, bg="yellow")
tkpack(f)
```

diamo un nome alla finestra e inseriamo nella finestra stessa un frame con colore di fondo giallo, largo 200 pixel e alto 100 pixel


Aggiungendo allo script le istruzioni

```
l = tklabel(f, text="CIAO", width = 20, padx=10, pady=10, bg="yellow", fg="red")
b = tkbutton(f, text="ESCI", bg="cyan", fg="blue")
tkpack(l, b)
```

otteniamo


Come si vede, con la sintassi sopra esemplificata e facilmente estensibile a tutti gli widget, li possiamo creare e collocare osservando che il loro nome e i comandi per collocarli sono gli stessi del linguaggio Tcl/Tk preceduti dal prefisso `tk` (`button` diventa `tkbutton`, `pack` diventa `tkpack`, ecc.).

[1] Per una introduzione a questo linguaggio consiglio il manualetto «tcl_tk» allegato al mio articolo «Tcl/Tk: un tesoro nascosto» pubblicato nel dicembre 2019 sul mio blog all'indirizzo *www.vittal.it*. Lo stesso contenuto si trova nel volumetto «Programmare con Tcl/Tk» acquistabile su Amazon.

Per quanto riguarda i dati da trattare, il linguaggio Tcl/Tk si presta all'acquisizione dei dati dalla tastiera ma, per poterli lavorare con le funzioni di R dobbiamo applicare il seguente trucco

. creiamo una variabile per l'ambiente Tcl con la funzione di R `tclVar()`

`x = tclVar(8)`

crea la variabile x per l'ambiente Tcl e vi inserisce il valore `8`

. per poter utilizzare nell'ambiente R questo valore ricorriamo alla funzione di R `tclvalue()`

`y = tclvalue(x)`

crea la variabile `y` per l'ambiente R e, dal momento che in Tcl tutto è stringa, vi inserisce il valore stringa `"8"`.

Per inserire nella variabile y questo valore come numero dobbiamo ricorrere al casting

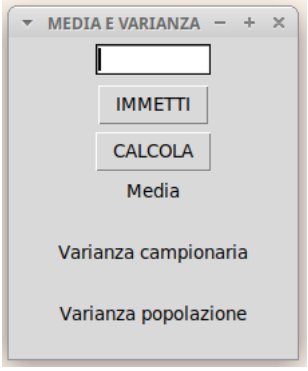
`y = as.numeric(tclvalue(x))`

Il seguente script esemplifica un po' tutto

```
library(tcltk)
r = tktoplevel()
tkwm.title(r, "MEDIA E VARIANZA")
x = tclVar("")
v = c()
e = tkentry(r, width=10, textvariable=x)
tkpack(e, pady=5)
tkfocus(e)
immetti = function(){
    xx = as.numeric(tclvalue(x))
    v <<- c(v, xx)
    tkdelete(e, 0, 10)
    tkfocus(e)
}
b = tkbutton(r, text="IMMETTI", command=immetti)
tkpack(b, pady=2)
calcola = function(){
    m <- mean(v)
    vc <- var(v)
    vp <- vc*(length(v)-1)/length(v)
    tkconfigure(l2, text=m)
    tkconfigure(l4, text=vc)
    tkconfigure(l6, text=vp)
    tkmessageBox(title="DATI", message=paste(v))
}
b2 = tkbutton(r, text="CALCOLA", command=calcola)
tkpack(b2, pady=2)
l1 = tklabel(r, text="Media")
tkpack(l1, pady=2)
l2 = tklabel(r)
tkpack(l2)
l3 = tklabel(r, text="Varianza campionaria")
tkpack(l3, padx=35, pady=2)
l4 = tklabel(r)
tkpack(l4)
l5 = tklabel(r, text="Varianza popolazione")
tkpack(l5, pady=2)
l6=tklabel(r)
tkpack(l6)
```

Con questo script possiamo calcolare la media e la varianza di una serie di dati con il software R senza conoscere nulla di questo software.

Memorizzato lo script in un file `prova.R`, se lo lanciamo dalla console compare la seguente finestra




L'utente inserisce i dati della serie di cui vuole calcolare media e varianza, uno per uno, nella finestrella cliccando sul pulsante IMMETTI per ciascuno di essi. Alla fine, cliccando sul pulsante CALCOLA, ottiene quanto richiesto, con un messaggio riepilogativo dei dati immessi.

Se, per esempio, inseriamo la serie 1 2 3 4 5 otteniamo