


Basic of R for Actuaries - Corso SIA 25/03/2021

Manuel Caccone



25/03/2021

Programma del corso




Parte 1 - Programma - *Overview* su R

-  negli usi attuariali: perché un attuario dovrebbe conoscere questo potente strumento.





Parte 1 - Programma - *Overview* su R

-  negli usi attuariali: perché un attuario dovrebbe conoscere questo potente strumento.
-  in dialogo con altri sistemi alimentanti: R+SAS, R+MATLAB, R+EXCEL, R+ACCESS, R+SQL-ORACLE


Parte 1 - Programma - *Overview* su R

-  negli usi attuariali: perché un attuario dovrebbe conoscere questo potente strumento.
-  in dialogo con altri sistemi alimentanti: R+SAS, R+MATLAB, R+EXCEL, R+ACCESS, R+SQL-ORACLE
-  e i *Big Data*: quali strumenti per non perdere potenza di calcolo;



Parte 1 - Programma - *Overview* su R

-  negli usi attuariali: perché un attuario dovrebbe conoscere questo potente strumento.
-  in dialogo con altri sistemi alimentanti: R+SAS, R+MATLAB, R+EXCEL, R+ACCESS, R+SQL-ORACLE
-  e i *Big Data*: quali strumenti per non perdere potenza di calcolo;
-  e le *query* : logica SQL e logica NO-SQL con l'utilizzo di **dplyr**



Parte 2 - Programma - Applicazioni nel mondo *Life* e *Non-Life*

- **Non-Life Reserving Package:** l'utilizzo di *ChainLadder* e l'applicazione di metodi deterministici e stocastici in ;

Parte 2 - Programma - Applicazioni nel mondo *Life* e *Non-Life*


- **Non-Life Reserving Package:** l'utilizzo di *ChainLadder* e l'applicazione di metodi deterministici e stocastici in ;
- **Non-Life Premium Package:** partendo da una logica “anagrafica sinistri e premi”, costruzione di un *framework*  per un modello *frequency-severity* basato su un GLM;


Parte 2 - Programma - Applicazioni nel mondo *Life* e *Non-Life*

- **Non-Life Reserving Package:** l'utilizzo di *ChainLadder* e l'applicazione di metodi deterministici e stocastici in ;
- **Non-Life Premium Package:** partendo da una logica “anagrafica sinistri e premi”, costruzione di un *framework*  per un modello *frequency-severity* basato su un GLM;
- **Life Package:** l'insieme dei passaggi necessari per arrivare alla valutazione di un portafoglio di polizze mediante *Profit-Testing* deterministico.


Parte 1 - Overview


Parte 1 - Perché un attuario dovrebbe conoscere ?

In questa sezione andremo a discutere dei vantaggi e svantaggi relativi all'utilizzo del pacchetto  in ambito aziendale. Prima ancora dovremmo chiederci perchè no? Partiamo dai vantaggi:

-  è un pacchetto in licenza gratuita *GNU*


Parte 1 - Perché un attuario dovrebbe conoscere ?


In questa sezione andremo a discutere dei vantaggi e svantaggi relativi all'utilizzo del pacchetto  in ambito aziendale. Prima ancora dovremmo chiederci perchè no? Partiamo dai vantaggi:

-  è un pacchetto in licenza gratuita *GNU*
- è tra i 10 linguaggi più conosciuti e più redditizi al mondo¹

¹NorthEasternUniversity (2021)

Parte 1 - Perché un attuario dovrebbe conoscere ?

In questa sezione andremo a discutere dei vantaggi e svantaggi relativi all'utilizzo del pacchetto  in ambito aziendale. Prima ancora dovremmo chiederci perchè no? Partiamo dai vantaggi:

-  è un pacchetto in licenza gratuita *GNU*
- è tra i 10 linguaggi più conosciuti e più redditizi al mondo¹
- permette elevati livelli di personalizzazione e di flessibilità di calcolo, uno dei migliori strumenti di *back-end* di utilizzo aziendale;

¹NorthEasternUniversity (2021)

Parte 1 - Perché un attuario dovrebbe conoscere ?

- è un pacchetto **non** solo statistico, permette anche di sviluppare strumenti di *front-end* (R-Shiny);

Parte 1 - Perché un attuario dovrebbe conoscere ?

- è un pacchetto **non** solo statistico, permette anche di sviluppare strumenti di *front-end* (R-Shiny);
- è il software più utilizzato nella comunità di statistici e attuari, infiniti pacchetti nel CRAN: il “cervello” mondiale continuamente revisionato da una comunità di accademici e non che mettono a disposizione il loro sapere;

Parte 1 - Perché un attuario dovrebbe conoscere ?

- è un pacchetto **non** solo statistico, permette anche di sviluppare strumenti di *front-end* (R-Shiny);
- è il software più utilizzato nella comunità di statistici e attuari, infiniti pacchetti nel CRAN: il “cervello” mondiale continuamente revisionato da una comunità di accademici e non che mettono a disposizione il loro sapere;
- numerosissimi esponenti del mondo attuariale pubblicano il loro pacchetto od i loro lavori sul personale *Github*.

Parte 1 - Perché un attuario dovrebbe conoscere ?

Parliamo ora dei svantaggi:

- non è garantita una assistenza in caso di *bug* nei pacchetti pubblicati nel CRAN;


Parte 1 - Perché un attuario dovrebbe conoscere ?

Parliamo ora dei svantaggi:


- non è garantita una assistenza in caso di *bug* nei pacchetti pubblicati nel CRAN;
- necessita di *expertise* per il suo utilizzo;

Parte 1 - Perché un attuario dovrebbe conoscere ?

Parliamo ora dei svantaggi:

- non è garantita una assistenza in caso di *bug* nei pacchetti pubblicati nel CRAN;
- necessita di *expertise* per il suo utilizzo;
- il continuo *versioning* di  stesso e dei relativi pacchetti prevede una possibile obsolescenza in caso di aggiornamento di procedure sedimentate, nella progettazione ma mai nel funzionamento.

Parte 1 - La base della base

La programmazione in  è la sedimentazione di **oggetti**. I vari tipi di “atomo” che si possono trovare sono:

- vettore numerico

```
c(1,2,3)
```

```
## [1] 1 2 3
```

- vettore stringa

```
c('a','b','c')
```

```
## [1] "a" "b" "c"
```

Parte 1 - La base della base

- matrice come insieme di vettori numerici

```
matrix(c(1,2,3,4,5,6),ncol=2)
```

```
##      [,1] [,2]  
## [1,]    1    4  
## [2,]    2    5  
## [3,]    3    6
```

- *data-frame* come insieme misto di vettori numerici e stringa

```
data.frame(A=c('a','b','c'),B=c(1,2,3))
```

```
##    A B  
## 1 a 1  
## 2 b 2  
## 3 c 3
```

Parte 1 - La base della base

- *lista* come insieme misto di vettori, matrici e *data-frame*:

```
A=data.frame(A=c('a','b','c'),B=c(1,2,3))  
B=c('a','b','c')  
list(A,B)
```

```
## [[1]]
```

```
##      A B
```

```
## 1 a 1
```

```
## 2 b 2
```


```
## 3 c 3
```

```
##
```

```
## [[2]]
```

```
## [1] "a" "b" "c"
```


Parte 1 - La base della base

Inoltre  ragiona sugli oggetti così come l'Algebra: si possono considerare uno spazio su cui fare *applicazioni* chiamate *funzioni*:

```
parabola=function(x) return(x^2)
parabola(2)
```

```
## [1] 4
```

Parte 1 - La base della base

Inoltre  mette a disposizione due strumenti di gestione del calcolo, per renderlo più veloce ed efficiente:

- *For Loop*

$$A = \sum_i^4 x_i = 10 \quad x_i : 1, 2, 3, 4$$

```
a=c(1,2,3,4)
ris=c()
for(i in 1:3) ris[i]=a[i+1]+a[i]
```


Parte 1 - La base della base



■ *Vectorized Function*

$$f(a_i) = a_i^2 : a_i = 1, 2, 3$$

```
a=c(1,2,3)
lapply(a,function(x) return(x^2))
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 4
##
## [[3]]
## [1] 9
```

Parte 1 - La base della base

Per chi è nuovissimo nel mondo di , ho predisposto una guida semplicissima per muovere i primi passi². Per poter utilizzare  è possibile:

- a) scaricare la versione “nuda” di R³
- b) scaricare il tool completo **RStudio**⁴ di interfaccia grafica.

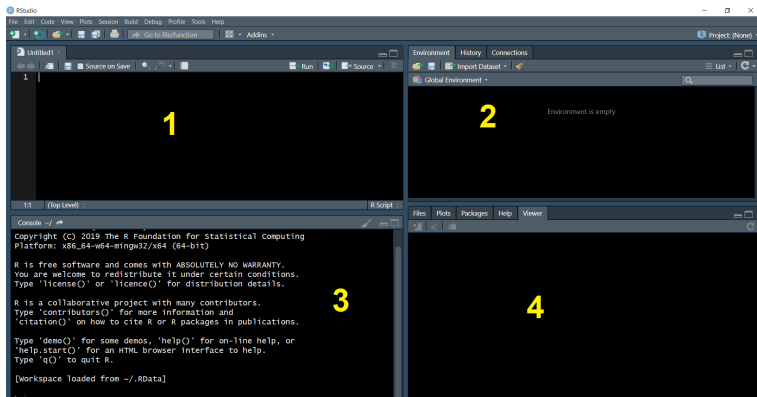
²Caccone (2021)

³CRAN-R-project (2021)

⁴RStudio (2021)

Parte 1 - Un piccolo focus su RStudio

Qui di seguito abbiamo una immagine della suddivisione dell'interfaccia RStudio:

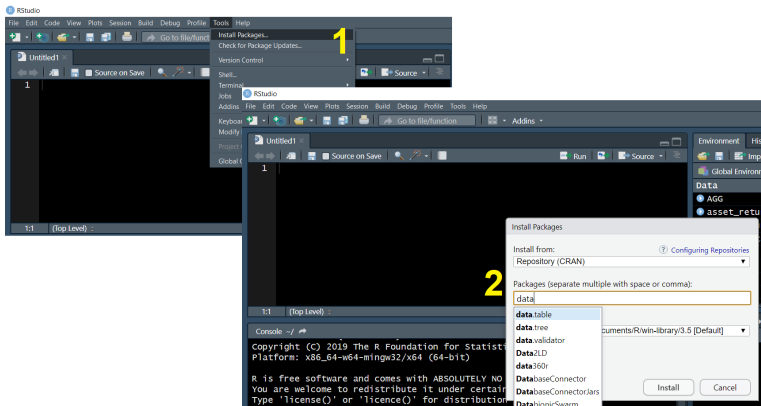


Parte 1 - Un piccolo focus su RStudio

- La prima (1) sezione è lo spazio dedicato alla nostra “lavagna”, dove scriveremo il nostro *script*
- La seconda (2) è quella del *Global Environment*, il “baule” ove risiedono gli oggetti di cui abbiamo parlato
- la *console* (3) che riceve i comandi/codice per procedere al calcolo e alla restituzione dei risultati/errori
- lo spazio (4) che chiameremo *other*, dove si possono visualizzare i risultati diversi dal calcolo: grafici, help file, viewer et *alia*.

Parte 1 - Un piccolo focus su RStudio

Altro passaggio importantissimo è quello relativo al *download* di un pacchetto dal CRAN:




Parte 1 - Un piccolo focus su RStudio

RStudio permette di fare questo passaggio tramite il menu a tendina oppure tramite codice da imputare a **console**:


```
install.packages('data.table',dependencies = T)
```

Parte 1 - Un piccolo focus su RStudio

Ricordiamo che  possiede la base (insieme necessario e sufficiente per l'insieme degli utilizzi basilari) e tutto il resto è uno “zoccolo duro” di pacchetti da scaricare. L'universo dei pacchetti è sconfinato e molti pacchetti dipendono tra loro: attenzione alle dipendenze quando installi un pacchetto.


Parte 1 - R in dialogo con altri sistemi alimentanti

R in dialogo con altri sistemi alimentanti

Uno dei vantaggi più rilevanti di  è la sua capacità di comunicare con più sistemi “alimentati”. In azienda spesso ci ritroviamo ad utilizzare:

- SAS
- Matlab
- Python
- Excel
- Access e Db Oracle - My SQL


R in dialogo con altri sistemi alimentanti - SAS

In SAS vengono prodotti prodotti dataset in formato *.sas7bdat*, in  è possibile gestire direttamente file di questo formato, leggerli, modificarli e salvarli. Il pacchetto da utilizzare è **haven**.⁵ Per questi passaggi vedremo l'esempio 1a

⁵Wickham (2021)

R in dialogo con altri sistemi alimentanti - MATLAB

Grazie al pacchetto *R.matlab*⁶ possiamo

- creare un collegamento diretto con i programmi che scriviamo in Matlab
- leggere i file *.mat* che salviamo
- lanciare procedure Matlab ed attendere che queste finiscano per ottenere gli output in .

Anche qui vedremo alcune applicazioni nell'esempio 1a.

⁶Bengtsson (2021)

R in dialogo con altri sistemi alimentanti - Python

Grazie al pacchetto *reticulate*⁷ possiamo:

- eseguire Python all'interno di R come unico “corpo”
- sfruttare da R librerie di *Machine Learning* e *Deep Learning* non previsti in R!
- utilizzare procedure già scritte in Python *.py* ed unirle a procedure in R.

Anche qui vedremo alcune applicazioni nell'esempio 1a.

⁷Ushey (2021)

R in dialogo con altri sistemi alimentanti - Python





Ricordiamo che Python possedere più versioni, quella che utilizzeremo da R è la versione *miniconda*. Il pacchetto prevede in unico comando **reticulate::install_miniconda()** l'installazione *freeware* della versione prevista, con la possibilità di utilizzo da RStudio.

R in dialogo con altri sistemi alimentanti - Excel

Uno dei sistemi alimentanti più utilizzati è sicuramente Microsoft Excel. Spesso riceviamo grossi file Excel e non riusciamo a gestirli opportunamente in quanto vi è un limite di righe. Possiamo dunque, mediante il pacchetto **readxl** riusciamo a caricare file `.xls/x`, modificarli e riscriverli nello stesso formato.




Anche qui vedremo alcune applicazioni nell'esempio 1a.

R in dialogo con altri sistemi alimentanti - Excel

Capita spesso che una procedura che abbiamo scritto in , vorremmo ci restituisca dei risultati direttamente in Excel. Per cui non vorremmo usare Excel da  ma  da Excel. Per questo possiamo installare **RExcel**⁸, un *addin* di Excel che permette di stampare, ad esempio, un risultato di una procedura  direttamente in una cella A1 di *Prova.xlsx*.


⁸Statconn (2021)

R in dialogo con altri sistemi alimentanti - Excel

Riusciamo inoltre su  a creare *ex novo* un foglio Excel che contenga output di procedure  scritte, creare un *template* ed incollare anche grafici che abbiamo programmato in .

Anche qui vedremo alcune applicazioni nell'esempio 1a.

R in dialogo con altri sistemi alimentanti - ACCESS, ORACLE, MYSQL

Se in azienda possediamo un Database Oracle, oppure un database più semplicemente in Access (.accdb), oppure una architettura Microsoft SQL Server, posso gestire le tabelle del database direttamente da . Il pacchetto **DBI**⁹ permette:


- stabilire connessioni ODBC con Driver Microsoft ODBC 32-bit e 64-bit
- leggere tutte le tabelle presenti sul Db e importarle come `data.frame`;
- modificare le tabelle e scrivere risultati direttamente come *DBO*

Insomma possiamo gestire il nostro DB senza passaggi intermedi!

⁹Müller (2021)

Parte 1 - R e i Big Data: quali strumenti per non perdere potenza di calcolo


Parte 1 - R e i Big Data - upload massivi

Spesso si ritiene che il SAS sia lo strumento più idoneo per gestire dati particolarmente “pesanti”. In realtà possiamo gestire in  file, ad esempio .csv, con milioni di righe e senza problemi di tempo di *uploading*. Il pacchetto **data.table**¹⁰ possiede una funzione *fread* che permette l'*upload* massivo di file con un numero importante di righe

¹⁰Dowle (2021)

Parte 1 - R e i Big Data - upload massivi

La funzione possiede particolari proprietà informatiche, in particolare:


- considera il file con un insieme scomponibile di unità elementari, dunque frammenta il file in più parti per poterlo gestire in maniera “diffusa”;
- una volta scomposto il file, questo viene richiamato in  con funzioni di *parallel importing*. In poche parole immaginando il nostro pc come una fattoria con 4 mulini (*core), questo viene “macinato” contemporaneamente per avere più in fretta il risultato.

Parte 1 - R e i Big Data - applicazioni di funzioni



Considerate operazioni anche semplici su grossi dataset, possiamo avere difficoltà in termini di *time machine* ogni qual volta si ottiene un campo calcolato. Lo strumento *data.table* del pacchetto omonimo è una evoluzione dello strumento base *data.frame*. In particolare:

- gestisce le colonne del dataset in maniera “diffusa” (come già ribadito);
- permette l'uso combinato di *vectorization* e *paralleling* delle funzioni di calcolo;
- permette di ridurre i tempi di elaborazione in maniera significativa come conseguenza delle logiche menzionate.

Parte 1 - R e i Big Data - gestione della memoria virtuale

Spesso quando si utilizza il SAS si fa riferimento al PDV, quale magazzino di memoria temporanea e centro di elaborazione di un qualsiasi calcolo. Quando si gestisce un dataset della categoria *Big Data*, un qualsiasi software possiede il suo PDV. In  viene utilizzata la memoria temporanea del computer, tuttavia è possibile utilizzare alcuni strumenti che “rimpiccioliscono” il peso specifico di grossi dataset.


Parte 1 - R e i Big Data - gestione della memoria virtuale

Il pacchetto **fst**¹¹ permette la compressione dei Big Data caricati in , ciò permetterà la gestione dei calcoli nel PDV di  mediante un carico più “alleggerito” e dunque più efficiente sia da un punto di vista del tempo-macchina, sia per la fluidità della procedura che creeremo.

¹¹Klik (2021)

Parte 1 - R e le query : logica SQL e logica NO-SQL con l'utilizzo di dplyr

Parte 1 - R e le query : logica SQL e logica NO-SQL con l'utilizzo di dplyr

Spesso ci ritroviamo ad utilizzare il SAS spesso estraiamo i dati mediante query PROC-SQL. Tuttavia anche in  possiamo utilizzare il linguaggio SQL per estrarre dati, innestare tabelle con le varie INNER JOIN, LEFT-JOIN etc.

Parte 1 - R e le query : logica SQL - pacchetto **sqldf**

Il pacchetto **sqldf**¹² permette di attuare tutte le operazioni in linguaggio SQL, trattando i `data.frame` presenti nel *Global Environment* come tabelle di un DB. Sd esempio estrarre alcune informazioni preliminari:

```
#install.packages(c('sqldf','RH2'),dependencies = TRUE)
suppressPackageStartupMessages({
  library(sqldf)
  library(RH2)
})
sqldf("select * from iris limit 5")
```

##	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
## 1	5.1	3.5	1.4	0.2	setosa
## 2	4.9	3.0	1.4	0.2	setosa
## 3	4.7	3.2	1.3	0.2	setosa

Parte 1 - R e le query : logica SQL - pacchetto **sqldf**

Oppure ottenere dei veri e propri campi calcolati:

```
DF <- data.frame(a = 1:5, b = letters[1:5])  
sqldf("select * from DF")
```

```
##    a b  
## 1 1 a  
## 2 2 b  
## 3 3 c  
## 4 4 d  
## 5 5 e
```

```
sqldf("select avg(a) mean, var_samp(a) var from DF")
```

```
##    mean var  
## 1      3 2.5
```

Parte 1 - R e le query : logica NO-SQL - pacchetto **dplyr**

Se volessimo abbandonare la logica SQL, possiamo senz'altro utilizzare una logica NO-SQL. In particolare il pacchetto **dplyr**¹³ offre delle soluzioni di logica incrementale sulla manipolazione dei dati. L'operatore "`%>%`" permette di aggiungere logiche di selezione del dato. Facciamo qualche esempio.

¹³Hadley (2021)

Parte 1 - R e le query : logica NO-SQL - pacchetto **dplyr**

Possiamo selezionare i dati in base ad un attributo:

```
suppressPackageStartupMessages({library(dplyr)})  
head(starwars[,c('species')])
```

```
## # A tibble: 6 x 1  
##   species  
##   <chr>  
## 1 Human  
## 2 Droid  
## 3 Droid  
## 4 Human  
## 5 Human  
## 6 Human
```

```
starwars %>%  
  filter(species == "Droid")
```

Parte 1 - R e le query : logica NO-SQL - pacchetto **dplyr**

Possiamo selezionare i dati in base ad un attributo:

```
starwars %>%
  filter(species == "Droid")
```

```
## # A tibble: 6 x 14
##   name    height    mass hair_color skin_color eye_color birth_year
##   <chr>    <int> <dbl> <chr>      <chr>      <chr>      <dbl>
## 1 C-3PO      167     75 <NA>      gold        yellow      1938
## 2 R2-D2       96     32 <NA>      white, bl~ red        1938
## 3 R5-D4       97     32 <NA>      white, red red        1938
## 4 IG-88      200    140 none      metal       red        1938
## 5 R4-P~       96     NA none      silver, r~ red, blue   1938
## 6 BB8        NA     NA none      none        black       1938
## # ... with 5 more variables: homeworld <chr>, species <chr>,
## #   vehicles <list>, starships <list>
```


Parte 1 - R e le query : logica NO-SQL - pacchetto **dplyr**

Possiamo selezionare e rinominare un campo in base ad un caratteristica:

```
starwars_1= starwars %>%  
  mutate(name, bmi = mass / ((height / 100) ^ 2)) %>%  
  select(name:mass, bmi)  
head(starwars_1,3)
```

```
## # A tibble: 3 x 4  
##   name          height  mass  bmi  
##   <chr>         <int> <dbl> <dbl>  
## 1 Luke Skywalker    172    77  26.0  
## 2 C-3P0             167    75  26.9  
## 3 R2-D2              96    32  34.7
```

Parte 1 - R e le query : logica NO-SQL - pacchetto **dplyr**

Possiamo inoltre combinare più operazioni, proprio come nel linguaggio SQL, con il vantaggio di attuare molte più opzioni e funzione interne ad , avendo un linguaggio fluido come un SQL ma “potente”:

```
starwars_2=starwars %>%  
  group_by(species) %>%  
  summarise(  
    n = n(),  
    mass = mean(mass, na.rm = TRUE)  
  ) %>%  
  filter(  
    n > 1,  
    mass > 50  
  )
```



Parte 1 - R e le query : logica NO-SQL - pacchetto **dplyr**

```
head(starwars_2,3)
```

```
## # A tibble: 3 x 3  
##   species      n  mass  
##   <chr>    <int> <dbl>  
## 1 Droid         6  69.8  
## 2 Gungan        3   74  
## 3 Human       35  82.8
```

Parte 2 - Non-Life Reserving Package - metodi deterministici e stocastici

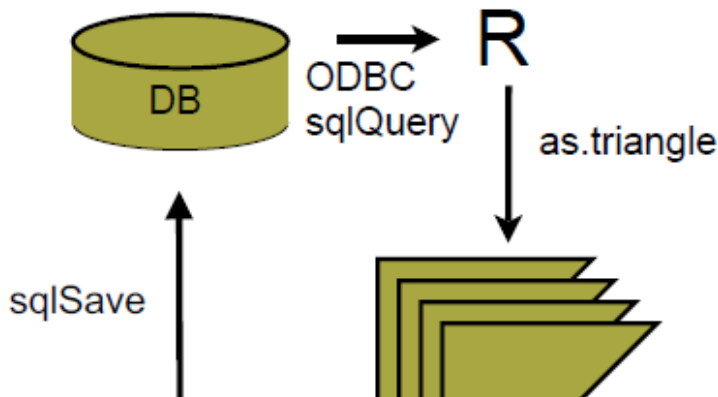
Parte 2 - Non-Life Reserving Package - metodi deterministici

In questa sezione andremo a parlare dei metodi di riservazione deterministici in . Normalmente si ricevono delle “anagrafiche sinistri” che non permettono l'applicazione diretta del metodo mediante il formato “triangolare”. Tuttavia il pacchetto **ChainLadder**¹⁴ mette a disposizione una serie di funzioni semplicissime da utilizzare, le quali semplificano tantissimo i passaggi che normalmente adottiamo in Excel.

¹⁴Gesmann (2021)

Parte 2 - Non-Life Reserving Package - organizzazione del dato

Inoltre possiamo immaginare un *framework*¹⁵ del genere per la nostra procedura di calcolo:



Parte 2 - Non-Life Reserving Package - organizzazione del dato

In particolare ci può risultare particolarmente utile la trasformazione di una anagrafica sinistri del tipo:

```
dati=data.frame(ID=1:7,ANNO_ACC=2010:2016,ANNO_SVIL=2011:2017,SINISTRO=rnorm(7,1000,10))  
head(dati)
```

##	ID	ANNO_ACC	ANNO_SVIL	SINISTRO
## 1	1	2010	2011	982.9477
## 2	2	2011	2012	993.1023
## 3	3	2012	2013	989.5693
## 4	4	2013	2014	1007.1285
## 5	5	2014	2015	1006.7664
## 6	6	2015	2016	1002.0789

In particolare l'esempio di dati di cui alla precedente slide è un caso classico (e semplicissimo) di anagrafica sinistri, quale estrazione del DataBase alimentante. La funzione *ChainLadder::as.triangle* del pacchetto menzionato trasforma automaticamente i dati in triangolare:

##	ANNO_SVIL							
##	ANNO_ACC	2011	2012	2013	2014	2015	2016	2017
##	2010	982.9477	NA	NA	NA	NA	NA	NA
##	2011	NA	993.1023	NA	NA	NA	NA	NA
##	2012	NA	NA	989.5693	NA	NA	NA	NA
##	2013	NA	NA	NA	1007.129	NA	NA	NA
##	2014	NA	NA	NA	NA	1006.766	NA	NA
##	2015	NA	NA	NA	NA	NA	1002.079	NA
##	2016	NA	NA	NA	NA	NA	NA	1005.324

Parte 2 - Non-Life Reserving Package - ChainLadder

Una volta ottenuta una rappresentazione triangolare dei sinistri risulta molto facile mediante la funzione *chainladder()* calcolare quanto necessario per la valutazione della riserva sinistri (una volta operato un aggiustamento sul nostro Db)

```
A[is.na(A)]<-10^(-10)
res=ChainLadder::chainladder(A)
summary(res)
```

```
##           Length Class      Mode
## Models      6    -none-    list
## Triangle  49    triangle numeric
## delta       6    -none-    numeric
## weights  49    triangle numeric
```

Parte 2 - Non-Life Reserving Package - ChainLadder

L'elemento di risultato è un oggetto di tipo lista e contiene tutti i modelli necessari quanti sono gli anni di sviluppo per poterne ottenere il fatto di sviluppo e la triangolare completa. Interessanti sono le possibilità della funzione **chainladder**

```
chainladder(Triangle = ,weights = ,delta = )
```

In particolare la possibilità di scegliere i pesi da associare alla matrice triangolare, inoltre il vantaggio di ottenere un *weighting* anche dei parametri *age-to-age*.

Parte 2 - Non-Life Reserving Package - BootChainLadder

Passando ai metodi stocastici, restiamo ancorati alla versione stocastica più semplice collegata al metodo deterministico più semplice (il ChainLadder per l'appunto). Con la sola funzione *ChainLadder::BootChainLadder()* è possibile calcolare in un unico colpo tutti gli output necessari, scegliendo una distribuzione di tipo Gamma oppure Over-dispersed Poisson:

```
res=ChainLadder::BootChainLadder(A,R = 999,process.distr = 'od.pois')
res
```

```
## ChainLadder::BootChainLadder(Triangle = A, R = 999, process.distr = "od.pois")
##
##           Latest Mean Ultimate Mean IBNR IBNR.S.E IBNR 75% IBNR 95%
## 2010 1.00e-10      1.00e-10      0      0      0      0
## 2011 1.00e-10      1.00e-10      0      0      0      0
## 2012 1.00e-10      1.00e-10      0      0      0      0
## 2013 1.00e-10      1.00e-10      0      0      0      0
## 2014 1.00e-10      1.00e-10      0      0      0      0
## 2015 1.00e-10      1.00e-10      0      0      0      0
## 2016 1.01e+03      1.01e+03      0      0      0      0
##
##           Totals
## Latest:      1,005
## Mean Ultimate: 1,005
## Mean IBNR:    0
```

Parte 2 - Non-Life Reserving Package - BootChainLadder

Ovviamente con pochi e semplici comandi riusciremmo ad impostare delle analisi di *sensitivity* al variare di alcuni parametri, al variare della base dati in maniera parametrica rispetto al flusso in entrata. Con le possibilità esplorate nelle *slide* precedenti, risulterà possibile staccare questi output in direttamente in Excel, oppure fare dei report in Word direttamente mediante RMarkdown.

Parte 2 - Non-Life Reserving Package - BootChainLadder

Per poter osservare dei risultati più plausibili rispetto al nostro dataset “giocattolo”, rimandiamo all'esercitazione 2a. Questa prende ampio spunto dalle esercitazioni contenute nelle *vignette*¹⁶ del relativo pacchetto.

¹⁶Alessandro Carrato and Zhang (2021)

Parte 2 - Non-Life Premium Package

Parte 2 - Non-Life Premium Package - un modello *frequency-severity*

In questa sezione andremo ad osservare le opportunità espresse dal pacchetto **RODBC**¹⁷. Con questo pacchetto includiamo alle possibilità già menzionate nelle sezioni precedenti, quella di sfruttare un DB Access per trarne un *pricing* di tipo *frequency-severity*.

Database : Database- C:\Users\UGA05153\Desktop\CORSO-SIA----Basic-of-R-for-actuaries\Allegati\Dat\Bases...


File Home Crea Dati esterni Strumenti database Campi Tabella Che cosa si desidera fare?

Visualizza Incolla Taglia Copia Copia formato Filtro Crescente Decrescente Rimuovi ordinamento Selezione Avanzate Aggiorna tutto Nuovo Salva Elimina Totali Controllo ortografia Trova Sostituisci Vai a Seleziona Calibri (Corpo)

Tutti gli oggetti... Claims Policies

ID	CodAnagrafici	ImportoPrestazioni	ImportoLiquidato	IDTipoPrestazione	Descrizione	IdPrestazione
37	CAA016	1400	125	8	Odontoiatria	2610 Devitalizzazione
38	CAA016	36	36	4	Mammografia €	10800 Mammografia
39	CAA016	10.56	10.56	1	Visite specialisti	10500 Visita Specialistica
40	CAA016	36	36	2	Analisi laboratorio	10600 Analisi Laboratorio
41	CAA016	36	36	4	Mammografia €	10800 Mammografia
42	CAA016	23.51	23.51	1	Visite specialisti	10500 Visita Specialistica
43	CAA018	85	42.5	8	Odontoiatria	2582 Ablazione Tartaro
44	CAA018	85	42.5	1	Visite specialisti	2 Visita Specialistica
45	CAA028	90	49.5	4	Mammografia €	2224 Prestazione Grastroe
46	CAA028	17.35	17.35	2	Analisi laboratorio	10600 Analisi Laboratorio
47	CAA028	100	50	1	Visite specialisti	2 Visita Specialistica
48	CAA028	58.7	58.7	2	Analisi laboratorio	10600 Analisi Laboratorio
49	CAA028	80	40	8	Odontoiatria	2582 Ablazione Tartaro
50	CAA030	0	0	6	Ricoveri ospedale	1375 Visita Urologica
51	CAA030	0	100	6	Ricoveri ospedale	10300 Ricovero
52	CAA033	120	60	1	Visite specialisti	2 Visita Specialistica
53	CAA033	120	66	4	Mammografia €	2223 Esame gravidanza
54	CAA033	20	60	4	Mammografia €	2 Visita Specialistica

Parte 2 - Non-Life Premium Package - **RODBC**

Con questo pacchetto abbiamo la possibilità di gestire una architettura DB (semplice come nell'immagine precedente) e trarre un collegamento ODBC diretto con . In particolare è possibile da questi agganciare tutte le logiche SQL e No-SQL che abbiamo visto in precedenza.


Parte 2 - Non-Life Premium Package - GLM

Inoltre nell'esercitazione di seguito a questa sezione, avremo la possibilità di fare *pricing* mediante l'applicazione dei *Generalized Linear Model* nel caso di dati del settore *Health*. In particolare attueremo un modello *Tweedie*¹⁸ mediante le funzioni del pacchetto **cplm**¹⁹.

¹⁸Tweedie (1984)

¹⁹Zhang (2021)

Parte 2 - Non-Life Premium Package - GLM

La possibilità di poter applicare in  un modello così “ricercato” e flessibile, forse non è disponibile in altri software di *statistical modeling*, dunque rimandiamo all'esercitazione 2b.

Parte 2 - Life Package

Parte 2 - Life Package - Valutazioni attuariali in R

Quando si parla di valutazione di portafoglio si intende la valutazione attuariale, riferibile ai flussi relativi ad un contratto assicurativo, siano essi positivi (ad es. premi) o negativi (ad es. prestazioni, spese). Nel loro insieme tali flussi sono riconducibili al problema della valutazione di operazioni finanziarie aleatorie. Nel caso delle assicurazioni sulla vita, si tratta di flussi generati da contratti di media-lunga durata, e la valutazione attuariale deve tenere conto delle due componenti di differimento e incertezza che caratterizzano tali flussi.

Parte 2 - Life Package - Valutazioni attuariali in R

Tali componenti sono trattate ricorrendo a strumenti tipici rispettivamente della matematica finanziaria e del calcolo delle probabilità; in particolare, ai fini della valutazione risulta necessario definire opportune basi tecniche. La scelta della base tecnica è solitamente determinata dallo scopo della valutazione attuariale:

- Base tecnica del I ordine: b.t. “prudenziiale” utilizzata ai fini di pricing e reserving;
- Base tecnica del II ordine: b.t. “realistica” utilizzata ai fini del profit testing, Embedded Value, IAS, Solvency II.

Parte 2 - Life Package - Valutazioni attuariali in R

Non esiste un unico metodo di valutazione ma, qualunque sia l'obiettivo di valutazione, esiste una impostazione comune basata sul concetto di Valore Attuale Netto dei flussi futuri – VAN (o Net Present Value - NPV).

Una specifica implementazione richiede di stabilire:

- il tipo di flussi da attualizzare;
- l'orizzonte temporale su cui i flussi stessi si estendono;
- il tasso al quale effettuare l'attualizzazione;
- la struttura probabilistica mediante la quale quantificare la componente aleatoria dei flussi.

Parte 2 - Life Package - Valutazioni attuariali in R

La scelta del tipo di flussi da attualizzare corrisponde all'adozione di uno specifico "criterio di valutazione". I criteri comunemente adottati sono: criterio di Cassa (considera i flussi monetari dati da entrate e uscite) ed il criterio Reddittuale o di competenza (considera i flussi degli utili periodali).

Parte 2 - Life Package - Valutazioni attuariali in R

Si consideri un portafoglio costituito da una generazione di N contratti identici su teste di età x . Sia

- x l'età di ingresso
- n durata del contratto
- i il tasso tecnico di gestione
- ${}_h p_x$ la probabilità di sopravvivenza h anni per testa di età x ;
- ${}_h \frac{1}{1} q_x$ probabilità di decesso tra h e $h + 1$ per testa di età x

Parte 2 - Life Package - Valutazioni attuariali in R

Siano inoltre

- P_{t-1}^T : premio di tariffa
- E_{t-1} : le spese pagate all'epoca $t - 1$
- C_t : il capitale pagato in caso di morte
- S_t : la somma assicurata alla scadenza n
- R_t : il valore di riscatto in caso di uscita anticipata V_t : la riserva costituita in t

Parte 2 - Life Package - Valutazioni attuariali in R

Abbiamo inoltre le basi tecniche del II ordine:

- i_t^* : livello realistico di rendimento atteso
- ${}_t p_x^*$: probabilità realistica di sopravvivenza
- ${}_h \frac{1}{1} q_x^*$: probabilità di morte realistica tra h e $h + 1$
- r_t^* : probabilità realistica di riscatto/abbandono
- ${}_t \lambda_x^*$: probabilità realistica di permanenza nel portafoglio

Parte 2 - Life Package - Valutazioni attuariali in R

A livello di singolo contratto si avrà il cash-flow atteso:

$$f_t^* = (P_{t-1}^T - E_{t-1})(1+i^*) - C_t q_{x+t-1}^* - R_t p_{x+t-1}^* \quad \forall \quad t = 1, 2, \dots, n-1$$

se $t = n$ allora

$$f_t^* = (P_{n-1}^T - E_{n-1})(1+i^*) - C_n q_{x+n-1}^* - S_n p_{x+n-1}^*$$

Parte 2 - Life Package - Valutazioni attuariali in R

Sia N_t il numero aleatorio di contratti tale per cui

$$N_t = E[N_t] = N \cdot {}_t\lambda_x^*$$

Allora il cash-flow annuo atteso, *emerging cost* è dato da

$$F_t^I = N_{t-1} f_t^*$$

con cash-flow totale

$$F_{(0,n)}^I = \sum_{t=1}^n F_t^I (1 + i^*)^{-t}$$

Parte 2 - Life Package - Valutazioni attuariali in R

In seguito potremo valutare inoltre la scomposizione dell'utile:

- utile **finanziario**

$${}_F u_t^* = (V_{t-1} + P_{t-1})(i^* - i) - V_t j_t^V p_{x+t-1}^*$$

- margine per **mortalità**

$${}_M u_t^* = (C_t - V_t^-)(q_{x+t-1} - q_{x+t-1}^*)$$


- margine per **riscatti e abbandoni**

$${}_R u_t^* = (V_t - R_t)p_{x+t-1}^* r_t^*$$

Laddove l'**utile tecnico** sarà

$$UT = {}_E u_t^* + {}_M u_t^* + {}_R u_t^*$$

Parte 2 - Life Package - Profit Testing in - FINE PRESENTAZIONE

Rimandiamo all'ultima esercitazione in base alle formule viste in precedenza. Faremo una esercitazione in Excel ed una omologa in  utilizzando l'allegato 2c.

References I

Alessandro Carrato, Markus Gesmann, Fabio Concina, and Wayne Zhang. 2021. "Vignette - Chainladder." 2021. <https://cran.r-project.org/web/packages/ChainLadder/vignettes/ChainLadder.html>.

Bengtsson, Henrik. 2021. "CRAN - R.matlab." 2021. <https://CRAN.R-project.org/package=R.matlab>.

Caccone, Manuel. 2021. "Appunti Di Base." 2021. <https://github.com/manuelcaccone/CORSO-SIA---Basic-of-R-for-actuaries/raw/main/Allegati/appunti%20di%20base%20di%20R.pdf>.

CRAN-R-project. 2021. "Download Diretto Ultima Versione." 2021. <https://cran.r-project.org/bin/windows/base/R-4.0.4-win.exe>.

Dowle, Matt. 2021. "CRAN - Data.table." 2021. <https://CRAN.R-project.org/package=data.table>.

References II

G., Grothendieck. 2021. "CRAN - Sqldf." 2021.

<https://CRAN.R-project.org/package=sqldf>.

Gesmann, Markus. 2021. "CRAN - Chainladder." 2021.

<https://CRAN.R-project.org/package=ChainLadder>.

Hadley, Wickham. 2021. "CRAN - Dplyr." 2021.

<https://CRAN.R-project.org/package=dplyr>.

Klik, Mark. 2021. "CRAN - Fst." 2021.

<https://CRAN.R-project.org/package=fst>.

Müller, Kirill. 2021. "CRAN - Rdbi." 2021.

<https://CRAN.R-project.org/package=Rdbi>.

References III

NorthEasternUniversity. 2021. "The 10 Most Popular Programming Languages to Learn in 2021." 2021.

<https://www.northeastern.edu/graduate/blog/most-popular-programming-languages.htm>.

Ripley, Brian. 2021. "CRAN - Rodbc." 2021.

<https://CRAN.R-project.org/package=RODBC>.

RStudio. 2021. "Download Diretto Ultima Versione." 2021. <https://download1.rstudio.org/desktop/windows/RStudio-1.4.1106.exe>.

Statconn. 2021. "Download Addin." 2021.

<http://www.statconn.com/products.html>.

References IV

Tweedie, Maurice CK. 1984. "An Index Which Distinguishes Between Some Important Exponential Families." In *Statistics: Applications and New Directions: Proc. Indian Statistical Institute Golden Jubilee International Conference*, 579:579–604.

Ushey, Kevin. 2021. "CRAN - Reticulate." 2021.
<https://rstudio.github.io/reticulate/>.

Wickham, Hadley. 2021. "CRAN - Haven." 2021.
<https://CRAN.R-project.org/package=haven>.

Zhang, Yanwei (Wayne). 2021. "CRAN - Cplm." 2021.
<https://CRAN.R-project.org/package=cplm>.