


# Basic of R for Actuaries - Corso SIA 9/5/2025

Manuel Caccone



9/5/2025

## 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 degli svantaggi relativi all'utilizzo del pacchetto  in ambito aziendale. La risposta alla domanda in *header* è: perchè non dovrebbe?


Partiamo dai vantaggi:

-  è un pacchetto in licenza gratuita *GNU*


---

<sup>1</sup>NorthEasternUniversity (2021)

# Parte 1 - Perché un attuario dovrebbe conoscere ?

In questa sezione andremo a discutere dei vantaggi e degli svantaggi relativi all'utilizzo del pacchetto  in ambito aziendale. La risposta alla domanda in *header* è: perchè non dovrebbe?


Partiamo dai vantaggi:

-  è un pacchetto in licenza gratuita *GNU*
- è tra i 10 linguaggi più conosciuti e più redditizi al mondo<sup>1</sup>


---

<sup>1</sup>NorthEasternUniversity (2021)

# Parte 1 - Perché un attuario dovrebbe conoscere ?

In questa sezione andremo a discutere dei vantaggi e degli svantaggi relativi all'utilizzo del pacchetto  in ambito aziendale. La risposta alla domanda in *header* è: perchè non dovrebbe?

Partiamo dai vantaggi:

-  è un pacchetto in licenza gratuita *GNU*
- è tra i 10 linguaggi più conosciuti e più redditizi al mondo<sup>1</sup>
- permette elevati livelli di personalizzazione e di flessibilità di calcolo, uno dei migliori strumenti di *back-end* di utilizzo aziendale;

---

<sup>1</sup>NorthEasternUniversity (2021)

# Parte 1 - Perché un attuario dovrebbe conoscere ?

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

# Parte 1 - Perché un attuario dovrebbe conoscere ?

- è un software **non** solo statistico, permette anche di sviluppare strumenti di *front-end* (R-Shiny);
- è il software più utilizzato nella comunità di statistici e attuari, vi sono 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 software **non** solo statistico, permette anche di sviluppare strumenti di *front-end* (R-Shiny);
- è il software più utilizzato nella comunità di statistici e attuari, vi sono 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 e i loro lavori sul personale *Github*.



# Parte 1 - Perché un attuario dovrebbe conoscere ?

Parliamo ora degli svantaggi:

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


# Parte 1 - Perché un attuario dovrebbe conoscere ?

Parliamo ora degli 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 degli 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** che ne costituiscono la sua materia. I vari tipi di “atomo” 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 con gli oggetti così come farebbe 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 *newbie* del mondo di , ho predisposto una guida semplicissima per muovere i primi passi<sup>2</sup>. Per poter utilizzare  è possibile:

- a) scaricare la versione “nuda” di R<sup>3</sup>
- b) scaricare il tool completo **RStudio**<sup>4</sup> di interfaccia grafica (consigliato).

---

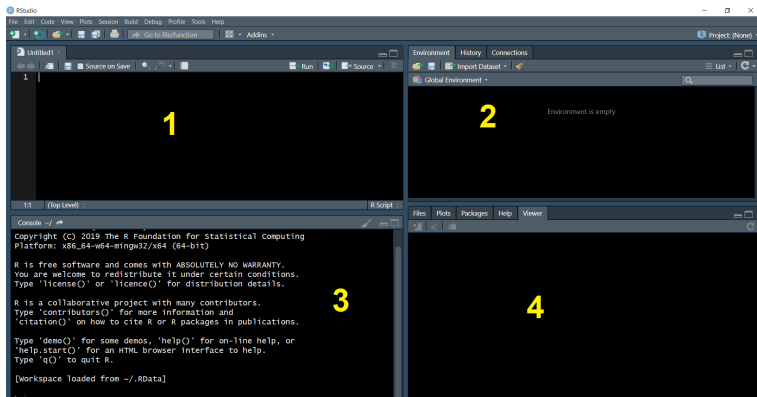
<sup>2</sup>Caccone (2021)

<sup>3</sup>CRAN-R-project (2021)

<sup>4</sup>RStudio (2021)

# Parte 1 - Un piccolo focus su RStudio

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

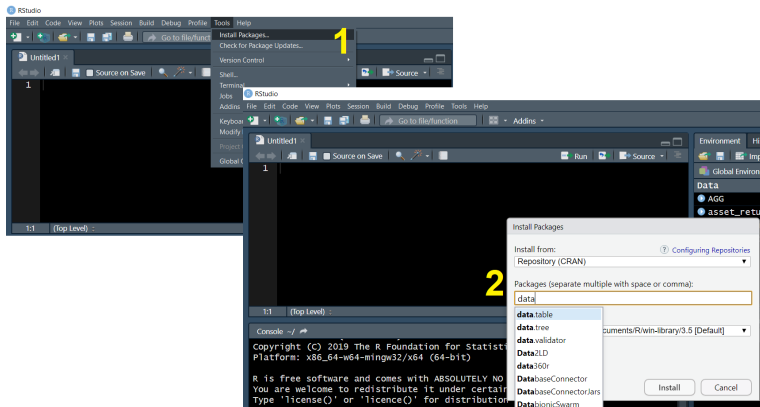


## Parte 1 - Un piccolo focus su RStudio

- La prima (1) sezione è lo spazio dedicato alla nostra “lavagna”, dove avremo 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. Nessuno possiede un  uguale ad un altro, ognuna avrà così la sua versione personalizzata. L'universo dei pacchetti è sconfinato e molti di questi 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 dell'utilizzo di  è la sua capacità di comunicare con più sistemi “alimentanti”. 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 *dataset* in formato *.sas7bdat*, tuttavia in  è possibile gestire direttamente file di questo formato: è possibile leggerli, modificarli e salvarli. Il pacchetto da utilizzare è **haven**<sup>5</sup>. Per questi passaggi vedremo l'esempio 1a

---

<sup>5</sup>Wickham (2021)

# R in dialogo con altri sistemi alimentanti - MATLAB

Grazie al pacchetto *R.matlab*<sup>6</sup> 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 poi ottenere gli output prodotti leggibili in .

Anche qui vedremo alcune applicazioni nell'esempio 1a.

---

<sup>6</sup>Bengtsson (2021)

# R in dialogo con altri sistemi alimentanti - Python

Grazie al pacchetto *reticulate*<sup>7</sup> possiamo:


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

Anche qui vedremo alcune applicazioni nell'esempio 1a.

---

<sup>7</sup>Ushey (2021)

# R in dialogo con altri sistemi alimentanti - Python

Ricordiamo che Python possiede più versioni, quella che utilizzeremo da  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 (famoso il caso *Covid Great Britain*<sup>8</sup>). Possiamo dunque, mediante il pacchetto **readxl**, caricare file *.xlsx*, modificarli e riscriverli nello stesso formato.

Anche qui vedremo alcune applicazioni nell'esempio 1a.

---

<sup>8</sup>Kelion (2020)



## R in dialogo con altri sistemi alimentanti - Excel

Capita spesso di voler utilizzare una procedura che abbiamo scritto in  in Excel, vorremmo in particolare che restituisca dei risultati direttamente nel nostro foglio di calcolo. Per cui si desidera NON usare Excel da  MA  da Excel. Per questo possiamo installare **RExcel**<sup>9</sup>, un *addin* di Excel che permette di stampare, ad esempio, un risultato di una procedura  direttamente in una cella A1 di *Prova.xlsx*.

---

<sup>9</sup>Statconn (2021)


## R in dialogo con altri sistemi alimentanti - Excel

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

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, una architettura Microsoft SQL Server oppure, più semplicemente, un DB Access (.accdB), questi si possono gestire direttamente da . Il pacchetto **DBI**<sup>10</sup> permette di:

- 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!

---

<sup>10</sup>Müller (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 in *uploading*. Il pacchetto **data.table**<sup>11</sup> possiede una funzione *fread* che permette l'*upload* massivo di file con un numero importante di righe

---

<sup>11</sup>Dowle (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” (vaporizzazione);
- 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), questi 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 vanno a “rimpicciolire” il peso specifico di grossi dataset.

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

Il pacchetto **fst**<sup>12</sup> 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.


---

<sup>12</sup>Klik (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

Quando estraiamo dati utilizzando il SAS, spesso facciamo largo uso di 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**<sup>13</sup> permette di attuare tutte le operazioni in linguaggio SQL, trattando i *data.frame* presenti nel *Global Environment* come tabelle di un DB. Ecco un semplice esempio per estrarre alcune informazioni preliminari:

```
#install.packages(c('sqldf', 'RH2'), dependencies = TRUE)
suppressPackageStartupMessages({
library(sqldf)
library(RH2)
})
```

```
## Warning: il pacchetto 'sqldf' è stato creato con R versione 3.5.1
```

```
## Warning: il pacchetto 'gsubfn' è stato creato con R versione 2.8.1
```

```
## Warning: il pacchetto 'proto' è stato creato con R versione 0.12.0
```

## 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**<sup>14</sup> offre delle soluzioni di logica incrementale sulla manipolazione dei dati. L'operatore “%>%” permette di aggiungere logiche di selezione del dato, di trasformazione, di calcolo e tanto altro. Facciamo qualche esempio.

---

<sup>14</sup>Hadley (2021)

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

Possiamo selezionare i dati in base ad un attributo:

```
suppressPackageStartupMessages({library(dplyr)})
```

```
## Warning: il pacchetto 'dplyr' è stato creato con R versio
```

```
head(starwars[,c('species')])
```

```
## # A tibble: 6 x 1
```

```
##   species
```

```
##   <chr>
```

```
## 1 Human
```

```
## 2 Droid
```

```
## 3 Droid
```

```
## 4 Human
```

```
## 5 Human
```

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

Possiamo selezionare, calcolare 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 ma con il vantaggio di attuare molte più opzioni e funzioni interne ad **R**, 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  81.3
```



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

## Parte 2 - Non-Life Reserving Package - metodi deterministici

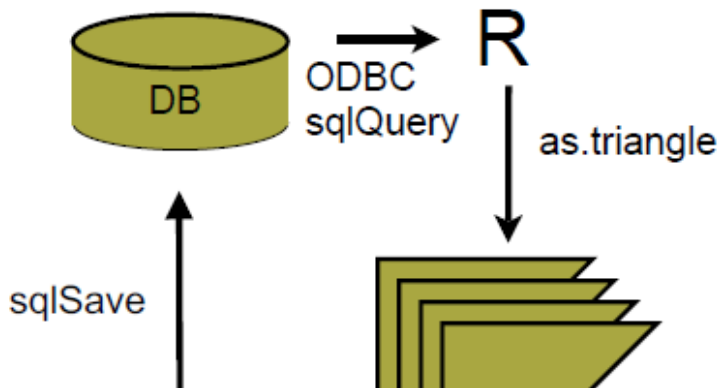
In questa sezione andremo a parlare dei metodi di riservazione deterministici in . Normalmente si ricevono delle “anagrafiche sinistri” che non permettono l’applicazione diretta del metodo mediante il formato “triangolare”. Tuttavia il pacchetto **ChainLadder**<sup>15</sup> mette a disposizione una serie di funzioni semplicissime da utilizzare, le quali semplificano tantissimo i passaggi che normalmente adottiamo in Excel.

---

<sup>15</sup>Gesmann (2021)

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

Inoltre possiamo immaginare un *framework*<sup>16</sup> del genere per la nostra procedura di calcolo:



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

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	1001.0359
## 2	2	2011	2012	986.6145
## 3	3	2012	2013	983.2484
## 4	4	2013	2014	995.4664
## 5	5	2014	2015	993.8868
## 6	6	2015	2016	1020.7103

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

Dunque 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:

```
# install.packages('ChainLadder',dependencies = T)
suppressPackageStartupMessages({library(ChainLadder)})
```

```
## Warning: il pacchetto 'ChainLadder' è stato creato con R versione 4.4.3
```

```
A=as.triangle(dati,origin = 'ANNO_ACC',dev = 'ANNO_SVIL',value = 'SINISTRO')
A
```

```
##           ANNO_SVIL
## ANNO_ACC  2011    2012    2013    2014    2015    2016    2017
## 2010 1001.036      NA      NA      NA      NA      NA      NA
## 2011      NA 986.6145      NA      NA      NA      NA      NA
## 2012      NA      NA 983.2484      NA      NA      NA      NA
## 2013      NA      NA      NA 995.4664      NA      NA      NA
## 2014      NA      NA      NA      NA 993.8868      NA      NA
## 2015      NA      NA      NA      NA      NA 1000.71      NA
```

## 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 gli *age-to-age factors* 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* dei parametri *age-to-age* stessi.

## 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  1e-10      1e-10      0      0      0      0
## 2011  1e-10      1e-10      0      0      0      0
## 2012  1e-10      1e-10      0      0      0      0
## 2013  1e-10      1e-10      0      0      0      0
## 2014  1e-10      1e-10      0      0      0      0
## 2015  1e-10      1e-10      0      0      0      0
## 2016  1e+03      1e+03      0      0      0      0
##
##                               Totals
## Latest:                    1,000
## Mean Ultimate:             1,000
```



## 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 direttamente in Excel oppure fare dei report direttamente in Word 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*<sup>17</sup> del relativo pacchetto.

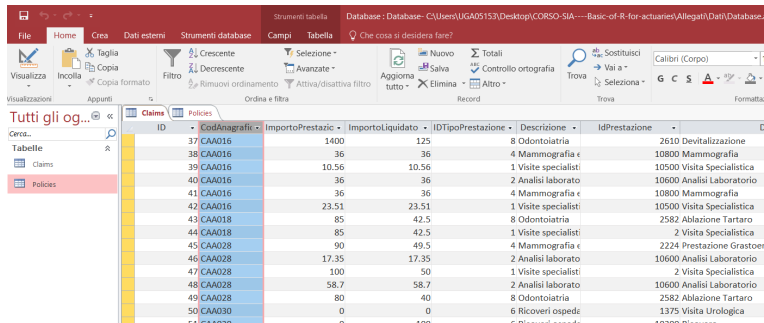
---

<sup>17</sup>Alessandro 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**<sup>18</sup>. 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*.



ID	CodAnagrafic	ImportoPrestazic	ImportoLiquidato	IDTipoPrestazione	Descrizione	IdPrestazione
37	CAA016	1400	125	8	Odontoiatria	2610 Devitalizzazione
38	CAA016	36	36	4	Mammografia e	10800 Mammografia
39	CAA016	10.56	10.56	1	Visite specialisti	10500 Visita Specialistica
40	CAA016	36	36	2	Analisi laborato	10600 Analisi Laboratorio
41	CAA016	36	36	4	Mammografia e	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 e	2224 Prestazione Grastro
46	CAA028	17.35	17.35	2	Analisi laborato	10600 Analisi Laboratorio
47	CAA028	100	50	1	Visite specialisti	2 Visita Specialistica
48	CAA028	58.7	58.7	2	Analisi laborato	10600 Analisi Laboratorio
49	CAA028	80	40	8	Odontoiatria	2582 Ablazione Tartaro
50	CAA030	0	0	6	Ricoveri ospede	1375 Visita Urologica
51	CAA030	0	0	6	Ricoveri ospede	1375 Visita Urologica

## 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*<sup>19</sup> mediante le funzioni del pacchetto **cplm**<sup>20</sup>.

---

<sup>19</sup>Tweedie (1984)

<sup>20</sup>Zhang (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 è un forte vantaggio). Rimandiamo, per approfondimenti di questa sezione, 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 Reddituale 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**

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

- 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.

Extra

## Extra - Il Decorator nelle funzioni di R

### ■ Cos'è un decoratore?

- In programmazione, un decoratore è una funzione che prende un'altra funzione come argomento e restituisce una nuova funzione.

### ■ Perché usare i decoratori?

- I decoratori possono essere usati per estendere il comportamento di una funzione esistente senza modificarla direttamente.

## Extra - Il Decorator nelle funzioni di R

### ■ Esempio di Decoratore in R:

```
# Definizione di un decoratore
decorate <- function(func) {
  function(...) {
    print("Before calling the function")
    result <- func(...)
    print("After calling the function")
    return(result)
  }
}

# Funzione da decorare
add <- function(a, b) {
  return(a + b)
}
```

# Extra - Copilot

## ■ Cos'è Copilot?

- Copilot è un assistente di programmazione basato su IA che aiuta a scrivere codice in modo più rapido e preciso.

## ■ Perché usare Copilot?

- Copilot può essere usato per scrivere codice in modo più rapido e preciso;
- aiuta la stesura del codice in maniera di esponenziale di produzione;
- può essere utilizzato per scrivere codice in maniera più efficiente.

# 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>.



## References II

- 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>.
- 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>.

## References III

Kelion, Leo. 2020. “Excel: Why Using Microsoft’s Tool Caused Covid-19 Results to Be Lost.” 2020.

<https://www.bbc.com/news/technology-54423988>.

Klik, Mark. 2021. “CRAN - FST.” 2021.

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

Müller, Kirill. 2021. “CRAN - DBI.” 2021.

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

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

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

## References IV

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>.

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/>.

## References V

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>.