UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Psicologia dello Sviluppo e della Socializzazione

Corso di Laurea in Scienze psicologiche dello sviluppo, della personalità e delle relazioni interpersonali

Elaborato finale

Un approccio di Clustering funzionale per il text mining longitudinale: il corpus Sanremo.

Relatore
Prof. Calcagnì Antonio

Laureando: Samuele Lovera

Matricola: 1166671

ANNO ACCADEMICO 2019/2020

INDICE

Sommario

L'obiettivo di questo studio è utilizzare determinati algoritmi per estrarre informazioni da testi selezionati digitalizzati (*corpus*). Il procedimento consiste nell'identificare le parole-chiave che mostrano un andamento nel tempo simile fra loro. La grandezza presa in considerazione per la parole-chiave è la frequenza di presenza all'interno del testo analizzato (occorrenza). I testi analizzati sono quelli delle canzoni del festival di Sanremo dal 1951 al 2006. Il procedimento consiste nell'aggregare (*clusterizzare*), tramite appropriati algoritmi, le funzioni associate ad ogni parola-chiave. Così si ottengono dei *cluster* a cui sono associate con una certa probabilità funzioni tra loro simili (*pattern* temporali prototipici). L'analisi di quegli andamenti consente di estrarre informazioni più ad alto livello, come ad esempio, evoluzioni dei modi di comunicare e della cultura in generale, contesto storico, mode e altro ancora (Trevisani & Tuzzi, 2015). La possibilità di ottenere informazioni nascoste alla lettura umana rende l'estrazione computazionale del testo interessante ed utile per le sue applicazioni nelle discipline psicologiche. Considerando ad esempio l'enorme mole di dati testuali che si accumula ogni giorno nel campo della salute mentale, che si tratti di cartelle cliniche, forum di pazienti o social network, questo metodo sembra essere destinato ad occupare un posto importante nel panorama metodologico della ricerca psichiatrica (Abbe et al., 2016). In particolare, in questo elaborato è descritta una metodologia che permette di trattare dati testuali funzionali il cui andamento temporale è modellato mediante clustering funzionale. Tale proprietà rende il metodo particolarmente adatto allo studio dei *corpora* cronologici. Pensiamo, ad esempio, alla possibilità di processare i resoconti dei pazienti e di identificare le evoluzioni che subiscono nel corso del tempo come effetto dei trattamenti adottati.

CAPITOLO I

1.1 Text mining

Il *text mining* (TM) è l'analisi delle informazioni contenute in documenti testuali attraverso i metodi di *data mining*. All'interno di quest'ultimi, il TM è l'insieme di tecniche e modelli statistici specifici per i dati testuali (Tan, 1999). Nonostante il TM erediti parte dell'analisi propriamente qualitativa in cui entrano in gioco i significati contestualizzati delle parole (suo oggetto di studio), esso adotta un approccio di tipo statistico ed opera mediante strumenti quantitativi numerici (Tuzzi, 2003). La semplice lettura umana non è in grado di derivare informazioni nascoste, pattern e caratteristiche che fa invece emergere l'applicazione statistica computazionale ai dati testuali. Per questo motivo le applicazioni legate al TM sono molteplici soprattutto nell'ambito della ricerca scientifica.

1.2 Ambiente di lavoro: R

R è un linguaggio di programmazione ed ambiente di sviluppo specifico per l'analisi statistica dei dati. È un software gratuito, disponibile per diversi sistemi operativi (multipiattaforma) ed open source, cioè non protetto da copyright e liberamente modificabile dagli utenti (R Core Team, 2017). Uno degli aspetti che rende valido R è la vasta gamma di librerie *software* (nella terminologia R: pacchetti) di estensione inter-compatibili, fornite e gestite dagli stessi utenti. Inoltre, questi pacchetti possono essere installati facilmente e in sicurezza realizzando un ponte tra sviluppatori ed utenti di nuovi strumenti di analisi e rendendo R un ambiente di programmazione adatto per la collaborazione scientifica (Welbers et al., 2017). Questo fa dell'ambiente R una potente piattaforma per l'analisi computazionale del testo, rendendolo uno strumento prezioso per la ricerca sulla comunicazione.

1.2.1 I pacchetti utilizzati

The Comprehensive R Archive Network (CRAN) è una rete di siti che funge da servizio web principale per la distribuzione di sorgenti R come i suoi pacchetti di estensione (Crawley, 2013). I pacchetti sono un insieme di funzioni e *set* di dati che

aumentano la potenza di R migliorandone le funzionalità di base esistenti o aggiungendone di nuove. In questo specifico lavoro, per poter svolgere l'analisi di TM è stato utilizzato il pacchetto "tm" (Meyer, Hornik & Feinerer, 2008), che offre funzionalità per la gestione di documenti di testo astraendo il processo di manipolazione dei documenti e semplificando l'utilizzo di formati di testo eterogenei. Inoltre, i dati strutturati secondo la logica "tm" possono essere modellati attraverso il pacchetto "topicmodels" (Hornik & Grün, 2011) che fornisce l'infrastruttura di base per l'applicazione di modelli a *topic* latenti. Durante l'analisi è stato utilizzato anche il pacchetto "quanteda" (Benoit et al. 2018), che contiene una vasta gamma di funzioni specifiche per il trattamento e l'analisi di dati testuali. La libreria inoltre dispone di un pacchetto complementare per il caricamento dei testi, denominata "readtext" (Benoit & Obeng, 2017), la cui funzione principale permette di utilizzare in input uno o più file dal disco del calcolatore o da un URL e restituisce in output un tipo di dato in formato *data frame*. In particolare, "readtext" offre in un'unica funzione la possibilità di importare molti tipi di dati in un formato uniforme. Per implementare la tokenizzazione e lo *stemming* è stato utilizzato il pacchetto "SnowballC" (Bouchet-Valat, 2014). Per effettuare questo lavoro si è utillizato il pacchetto "data.table" (Dowle et al., 2019) che gestisce il data.frame mediante calcolo parallelo. Infine, è stato utilizzato il pacchetto "curvclust" (Giacofci et al., 2012) che implementa il modello statistico di raggruppamento delle parole-chiave (*clustering funzionale*), visualizzato con il pacchetto "pander".

1.3 Il corpus Sanremo

Un *corpus* è un insieme di testi omogenei raccolti per rispondere a una specifica domanda di ricerca. Il *corpus* di Sanremo contiene tutti ritornelli delle canzoni presentate al Festival della canzone italiana a partire dal 1951 fino al 2006. Il *corpus* è stato suddiviso in parti più piccole o sottoinsiemi del testo, cioè *sub-corpora* divisi in sequenze temporali di 5/6 anni. Ogni ritornello è composto da parole che sono sequenze di lettere isolate mediante separatori, ovvero spazi vuoti e segni di punteggiatura. Per elencare le parole contenute in un corpus, è necessario distinguere i concetti *word-type* e *word-token* (Trevisani & Tuzzi, 2015). Le *word-type* sono le parole distinte presenti in un testo. Mentre, il termine *word-token* si riferisce al numero totale di unità statistiche all'interno del corpus. Ad esempio, immaginando un corpus composto dall'unica frase: "un buon vino è un vino che ti piace",

un	buon	vino	è	un	vino	che	ti	piace	
1	2	3	4	5	6	7	8	9	*token*
1	2	3	4	1	3	5	6	7	parole distinte

stabilito che l'unità statistica di analisi è la parola, esso è formato da 9 *token*. Dati questi assunti viene assegnata automaticamente una variabile di classificazione delle unità statistiche che ha 7 unità, tante quante sono *word-type*. La frequenza di una parola distinta è il numero di *token* di parole corrispondenti nell'intero corpus analizzato. La lunghezza N del *corpus* è il numero totale di *token* e rappresenta la dimensione del testo in termini di occorrenze, cioè il numero di unità statistiche nel campione. Il numero V(N) delle parole distinte è la dimensione del *corpus* in termini di parole diverse e rappresenta il numero di modalità della variabile statistica associata all'unità di analisi. Il rapporto tra V / N fornisce una misura approssimativa della ricchezza lessicale (Tuzzi, 2003).

Una delle prime fasi della costruzione dei dati ha riguardato il problema dell'identificazione delle parole chiave su cui concentrare l'analisi. Una soluzione per rivelare i termini ai fini dell'analisi è quella di considerare le parole più frequenti. Questo metodo non tiene conto delle ridondanze e delle ambiguità insite nelle parole. Per tale motivo, durante la normalizzazione dei testi, è stata applicata al testo una procedura di *stemming*, ovvero il processo di riduzione della forma flessa di una parola alla sua forma radice, detta tema. In italiano, ad esempio, si riducono le parole "viaggiare" o "viaggiato" con il loro tema, cioè la parte in comune "viagg-"; Questa riduzione ha un duplice effetto benevolo sull'analisi del testo. In primo luogo, il minor numero di parole distinte limita la necessità di risorse sia in termini di tempo computazionale che di memoria fisica; un secondo effetto risiede nel fatto che accorpando parole con radice simile, e quindi presumibilmente con simile significato, l'analisi statistica può essere considerata più affidabile (Welbers et al., 2017).

Per identificare le parole-chiave è stato quindi utilizzato il dizionario comune per punti temporali. Eliminando ogni termine non presente in tutti i *sub-corpora*, sono state selezionate tutte le parole-chiave con frequenze uguali o superiori a 100 nel periodo 1951-2006. Questo ha permesso di avere informazioni sufficienti per tracciare nel tempo il

modello temporale delle frequenze delle singole parole-chiave. La matrice finale ha perciò 104 (parole-chiave) x 10 (punti temporali).

1.4 Primo trattamento dei dati con R

La preparazione dei dati è il punto di partenza per qualsiasi analisi. Per effettuare un'analisi statistica del testo utilizzando l'ambiente R è necessario infatti preparare il testo per l'analisi. È buona norma prima, di iniziare ogni sessione di lavoro su R, ripulire l'ambiente da eventuali oggetti/variabili presenti da sessioni precedenti; questo può essere fatto lanciando il comando: `rm (list = ls ())`. Dopo aver settato la working directory e caricato nell'ambiente di R i vari pacchetti/librerie da utilizzare per l'analisi è possibile procedere con il trattamento dei dati su R.

```
setwd ("~/Desktop/Elaborato Finale /")
library (readtext)
library (quanteda)
library (tm)
library (topicmodels)
library (SnowBallC)
```

1.4.1 Caricamento del Corpus

Prima di procedere con il caricamento effettivo del corpus è necessario caricare un file di supporto (*function.R*) che contiene alcune funzioni create *ad hoc* per alcuni passaggi dell'analisi preliminare:

```
source("~/Desktop/Elaborato Finale /functions.R")
```

Una volta selezionati i dati, il primo passo consiste quindi nell'importare il testo sull'ambiente R rendendolo un *corpus* grezzo:

```
t1 <- scan (file=paste0(getwd(),"/t1.txt"), fileEncoding='UTF-8',
what=character(), sep='\n', allowEscapes=T)
```

In questo modo, si è creato un oggetto su R (*t1*) contenente i dati che erano stati salvati in formato .txt. Il testo, adesso digitale, è rappresentato da una sequenza di caratteri, chiamate stringhe. In R, le stringhe sono rappresentate come oggetti chiamati tipi "*character*", che sono vettori di stringhe. In questo modo è possibile manipolare il testo digitale attraverso delle operazioni (ad esempio unione, divisione, l'estrazione di parti di stringhe) essenziali per l'analisi dei dati (Welbers, et al., 2017).

1.4.2 Pre-processamento

Ogni analisi testuale necessita di una fase di preparazione dei dati. Attraverso il pre-processamento il *corpus* viene pulito dal rumore, eliminando le informazioni superflue o parti di testo non necessarie:

```
testo <- scan(file=paste0(getwd(),"/t1.txt"),
          fileEncoding='UTF-8',
           what=character(), sep='\n', allowEscapes=T)
pos1=grep(pattern = '[****]',testo)
testo=testo[-pos1]
pos2=grep(pattern = '[++++]Titolo',testo)
testo=testo[-c(pos2,pos2+1)]
pos3=grep(pattern = '[++++]Testo',testo)
testo=testo[-c(pos3)]

testo=paste0(testo,collapse = ' ')
```

Pre-processare i dati, significa in questo caso *normalizzarli* manipolando il testo grezzo, con lo scopo di renderli meglio leggibili e analizzabili a livello computazionale. In quest'ottica, nel *corpus* Sanremo bisognerà quindi selezionare quelle unità testuali al netto delle locuzioni del linguaggio parlato che possono "sporcare" l'analisi, oscurando le eventuali unità testuali salienti. Un altro aspetto da considerare riguarda le preposizioni articolate che si hanno nel discorso parlato e che non destano particolare interesse per l'analisi che andremo a fare. Inoltre, le prestazioni di calcolo e l'accuratezza di molte tecniche di analisi del testo possono essere migliorate rimuovendo parole comuni non informative sul contenuto del testo designate in anticipo (dette *stopword*). Filtrare queste parole ha il vantaggio di ridurre la dimensione dei dati e il carico di calcolo, migliorandone la precisione. La rimozione è essenzialmente effettuata mediante l'utilizzo di elenchi predefiniti (Welbers, et al., 2017).

```
locs <- iconv(x = readLines(con = paste0("~/Desktop/Elaborato
 Finale/modi_di_dire_italiani_e_locuzioni.txt")))
prep_artic = read.table("~/Deskstop/ Elaborato Finale /
preposizioni_artic.csv", header = FALSE,sep = ",")


stpw_specific = iconv(readLines(con = paste0("~/Desktop/Elaborato Finale
 / stopwords specifiche.txt"), ok = TRUE))
stpw_specific = normalize_text(stpw_specific,preps.art = prep_artic,locs
 = locs)
stpw_specific = unlist(strsplit(stpw_specific,split = " "))
```

Una volta presi tutti gli accorgimenti si procede utilizzando il pacchetto "tm" che tramite i comandi seguenti:

```
doc.vec <- VectorSource(testo)
doc.corpus <- Corpus(doc.vec)
```

trasforma la matrice di testo in un *array* ad unica dimensione. Successivamente vengono applicate poi alcune trasformazioni per pulire il *corpus* dalle impurità che non verranno analizzate (ad esempio i connettivi logici, le punteggiature, numeri o percentuali). Il comando "*summary*" offre una panoramica dell'oggetto appena creato:

```
summary(doc.corpus)
##   Length Class             Mode
## 1 2      PlainTextDocument list
```

La normalizzazione è necessaria quando parole differenti hanno approssimativamente lo stesso significato, anche se sono scritte in modo leggermente diverso. Durante l'analisi, il testo è stato reso tutto in minuscolo attraverso il seguente comando:

```
doc.corpus <- tm_map(doc.corpus, tolower)
```

Questa trasformazione serve per rendere uguali, a livello computazionale, le parole che iniziano con la lettera maiuscola con le rispettive parole scritte in tutte in minuscolo.

Successivamente, sono stati lanciati i seguenti comandi: il primo permette di rimuovere la punteggiatura dal *corpus*, il secondo si occupa di eliminare i numeri.

```
doc.corpus <- tm_map(doc.corpus, removePunctuation)
doc.corpus <- tm_map(doc.corpus, removeNumbers)
```

Infine, attraverso la prossima funzione vengono rimosse dal *corpus* le *stopword* generali dell'italiano contenute nel pacchetto "tm".

```
doc.corpus <- tm_map(doc.corpus, removeWords, stopwords("italian"))
```

La tokenizzazzione consiste nella suddivisione del contenuto dei dati inziali in *token*, cioè una qualsiasi sequenza di caratteri (unità di testo) circondata da dei delimitatori (Welbers, et al., 2017). Spesso i *token* sono parole, perché questi sono i componenti semanticamente significativi più comuni dei testi. Un vantaggio è che riduce la dimensione del vocabolario, cioè l'intera gamma di funzionalità utilizzate nell'analisi, rendendo l'analisi più efficiente. In italiano la divisione dei testi per parole può essere eseguita principalmente grazie a chiari indicatori dei confini delle parole, come spazi bianchi e punteggiatura. Il processo di tokenizzazione e di *stemming* è stato implementato mediante il pacchetto "SnowBallC", come segue:

```
doc.corpus <- tm_map(doc.corpus, stemDocument)
doc.corpus <- tm_map(doc.corpus, stripWhitespace)
```

A questo punto è stato possibile raggiungere il primo obiettivo dell'analisi testuale: la creazione della *Term-Document-Matrix* (TDM), ossia una matrice di tipo termini x documenti che rappresenta la frequenza dei termini per ciascun documento all'interno di un oggetto *corpora*.

```
TDM <- TermDocumentMatrix(doc.corpus)
```

Da questa funzione si può creare la *Document-Term-Matrix* (DTM), ossia una rielaborazione della TDM, attraverso il seguente comando.

```
DTM <- DocumentTermMatrix(doc.corpus)
```

Una DTM è una matrice documenti x termini in cui le righe rappresentano i documenti, le colonne rappresentano i termini e le celle indicano l'occorrenza di un determinato termine all'interno di un documento. La DTM è uno dei formati più comuni per rappresentare un *corpus* di testo ignorando l'ordine delle parole, cioè tramite un approccio denominato *bag-of-word*. Il vantaggio di questa rappresentazione è quello di consentire l'analisi di dati mediante algebra lineare, (Welbers, et al., 2017). Nel nostro studio, la DTM parole-chiave x punti-temporali sarà l'input delle analisi successive. Risulta interessante estrapolare i termini che con maggiore frequenza vengono fuori da tale matrice; essendo le frequenze della TDM basse (ciò è dovuto al numero esiguo di documenti inseriti o comunque per scarsa associazione tra questi) si cercano quei termini che hanno una frequenza assoluta di ricorrenza pari ad un certo valore fissato (nel nostro caso, 200):

```
findFreqTerms(TDM, 200)
```

```
##  [1] "adesso" "amo" "amor" "ancora" "anima" "bella" "bene" "canzon"
##  [9] "cielo" "cosa" "così"  "cuor" "dentro" "dimmi" "dire" "donna"
## [17] "due"  "fare"  "fors"  "gent" "giorno"  "già"  "grand" "insiem"
## [25] "luna"  "mai"  "male" "mare" "mentr"  "mondo"  "nient"  "nott"
## [33] "occhi" "ogni"  "ora" "parol" "perchè" "poi"   "puoi"   "può"
## [41] "quando" "quel" "qui"  "sai"   "sempr"  "sento" "senza"  "sera"
## [49] "sola"  "sole"  "solo" "son"   "tanto"  "tempo" "terra"  "uomo"
## [57] "vai"  "vento"  "via" "vita"  "viver" "voglia" "voglio" "vorrei"
## [65] "vuoi"
```

1.5 Costruzione di un *corpus* longitudinale

Una volta caricato il testo grezzo, è stato creato un grafico che permetta di osservare la frequenza cumulata di *sub-corpus* negli anni:

```
      testo <- scan(file=paste0("~/Desktop/Elaborato Finale / t1.txt"), fil
eEncoding='UTF-8',
           what=character(), sep='\n', allowEscapes=T)
pos1=grep(pattern = '*Anno=',testo)
anni=testo[pos1]
anni=as.numeric(substr(unlist(sapply(strsplit(anni,split ='*Anno='),'[[',2))
,1,4))
plot(anni,xlab='Frequenza Cumulata',main='Frequenza cumulata di sub-corpus n
egli anni')
```

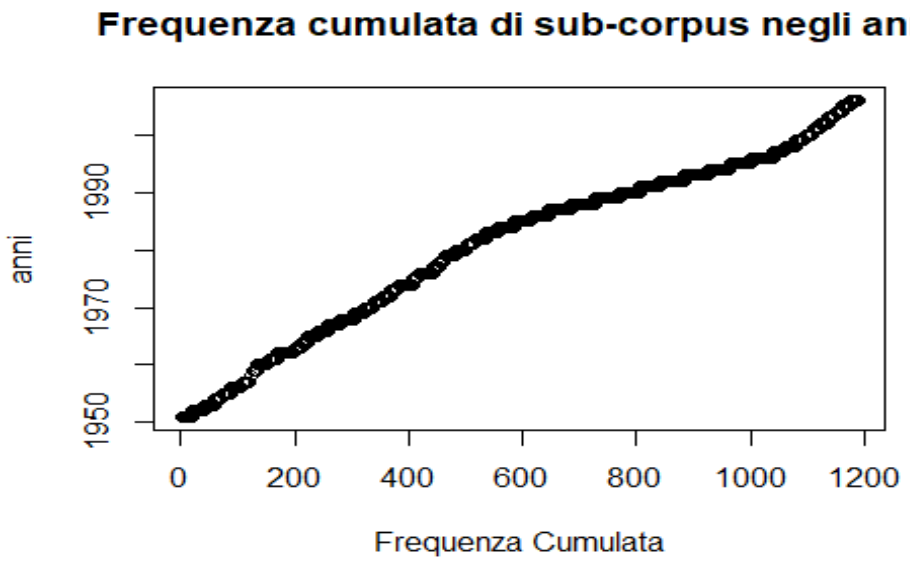


Figura 1

Si può osservare come l'andamento dell'integrale nel tempo delle frequenze assolute (frequenze cumulate) sia abbastanza lineare. L' analisi di questo andamento evidenzia un criterio di suddivisione del *corpus* attraverso la scelta del *range* di anni da considerare per ciascuna serie temporale. Per garantire una media di 100 unità statistiche per ogni *sub-corpora* è stato scelto di avere 10 *time-point*. Essendo l'intervallo di interesse di 55 anni è stato alternato un *range* da 5 anni con uno da 6.

```
classi=cut(anni,10)
table(classi)
```

```
## classi
## (1951,1956] (1956,1962] (1962,1968] (1968,1973] (1973,1978] (1978,1984]
##         107          92          80         102          81         123
## (1984,1990] (1990,1995] (1995,2000] (2000,2006]
##         185         232         107          80
```

Dal risultato emerge una profondità media di circa 100 unità statistiche per ogni sub-corpora. Per l'analisi è stato diviso il *corpus* principale secondo i seguenti *cut-point*, creando diversi sub-corpora:

```
cp=c(min(anni),as.numeric(substr(levels(classi),start = 7,stop = 10)))
cp
```

```
##  [1] 1951 1956 1962 1968 1973 1978 1984 1990 1995 2000 2006
```

Questi sono stati organizzati in una lista, ovvero una sequenza di elementi che, a differenza di vettori o matrici, possono contenere tipologia di dati diversi:

```
library(data.table)
CP=data.table(anni,pos1)
CP=CP[anni %in% cp,min(pos1),by=anni]
SC=list()
for(i in 1:(length(CP$V1)-1)){
      SC[[i]]=testo[CP$V1[i]:(CP$V1[i+1]-1)]  }
```

Adesso è necessario effettuare delle operazioni sulle stringhe affinché siano rimossi i caratteri speciali e le unità testuali non rilevanti per l'analisi. Questo viene fatto in *loop* per tutti i *sub-corpora*.

```
for(i in 1:length(SC)){
      pos1=grep(pattern = '[****]',SC[[i]])
      SC[[i]]=SC[[i]][-pos1]
      pos2=grep(pattern = '[++++]Titolo',SC[[i]])
      SC[[i]]=SC[[i]][-c(pos2,pos2+1)]
      pos3=grep(pattern = '[++++]Testo',SC[[i]])
      SC[[i]]=SC[[i]][-c(pos3)]
      SC[[i]]=paste0(SC[[i]],collapse = ' ') }
```

Una volta che i dati sono stati puliti, è stata creata una funzione ad *hoc* per effettuare tutte le operazioni necessarie al fine di normalizzare e pulire i *sub-corpora* in maniera corretta:

```
for(i in 1:length(SC)){
      SC[[i]] = normalize_text(text = SC[[i]],preps.art =
      prep_artic,locs = locs)

}
```

Il *corpus*, normalizzato e diviso in serie temporali, può essere rappresentato nella sua versione finale che verrà sottoposta all'analisi.

```
library(tm)
corpora = corpus(texts(x = c(SC[[1]],SC[[2]],SC[[3]],SC[[4]],SC[[5]],SC[[6]]
,
                             SC[[7]],SC[[8]],SC[[9]],SC[[10]])))
```

```
summary(corpora)

## Corpus consisting of 10 documents, showing 10 documents:
##
##    Text Types Tokens Sentences
##   text1  2614  14741        72
##   text2  2019  12437        23
##   text3  1944  15767        17
##   text4  1841  14650        35
##   text5  2249  13475        32
##   text6  2961  20141        44
##   text7  4762  41079        42
##   text8  5390  42619        19
##   text9  3736  30469        22
##  text10  2505  18438        30
```

Il comando "*summary*" ci offre una panoramica dei *sub-corpora* e delle relative frequenze dei *token*, delle parole distinte e delle frasi presenti in ogni punto temporale. Successivamente sono state rimosse le varie tipologie di *stopword* con i seguenti comandi:

```
stpw<-tm::stopwords(kind = "it")
stpw_tot<-c(stpw,stpw_specific)

corpora_tok<-tokens_select(x = tokens(corpora,remove_numbers=TRUE,remove_pun
ct=TRUE), pattern = stpw_tot,verbose=TRUE,padding=FALSE,selection = "remove"
,valuetype = "fixed")
```

1.6 La matrice dei dati longitudinali

In questo paragrafo verrà illustrato come creare la *Document-Feature-Matrix* (DFM) dei 10 *sub-corpora* elaborati fino a questo punto. La DFM diventerà l'unità analitica su cui sarà eseguita l'analisi del modello statistico. Una DFM è una matrice in cui le righe rappresentano i testi originali e le colonne le caratteristiche di quel testo (spesso *tokens*). Dopo aver creato la DFM, possiamo visualizzare l primi 10 elementi in colonna (*features*):

```
dfm_t1_t10 <- dfm(corpora_tok)
dfm_t1_t10[, 1:10]
```

```
## Document-feature matrix of: 10 documents, 10 features (22.0% sparse).
##        features
## docs    va serenata stasera ascoltare nessuno dovunque ricorderà baci vento può
##   text1 41        6       6         2       9        2         2   24     7  13
##   text2 21        2       2         0      10        1         0   27    21  18
##   text3 15        0      30         0      17        3         0   17     8  32
##   text4 60        4      15         0       8        0         0    0    16  25
##   text5 19        0       9         0      42        2         0    4    10  25
##   text6 73        8      20         2      16        1         0    0    34  12
## [ reached max_ndoc ... 4 more documents ]
```

Da cui è stato possibile creare il dizionario come segue:

```
freq.terms = data.frame(apply(dfm_t1_t10,2,sum),row.names = NULL)
dictionary = data.frame(colnames(dfm_t1_t10),freq.terms); names(dictionary)
= c("term","freq") #dictionary of corpora
dictionary = dictionary[order(dictionary$freq,decreasing = TRUE),] #sort dic
tionary

dictionaries = list()
for(i in 1:10){
  x = as.vector(dfm_t1_t10[i,])
  freq.terms = data.frame(x[x>0],row.names = NULL)
  dictionaries[[i]] = data.frame(colnames(dfm_t1_t10[i,])[x>0],freq.terms);
names(dictionaries[[i]]) = c("term","freq") #dictionary of corpora
  dictionaries[[i]] = dictionaries[[i]][order(dictionaries[[i]]$freq,decreas
ing = TRUE),] #sort dictionary
  colnames(dictionaries[[i]]) = c("terms","freq")
}
```

Di seguito è stato visualizzato quanto ottenuto finora sotto forma di "*summary*" delle frequenze riguardo al dizionario tracciato:

```
ttrs = textstat_lexdiv(x = dfm_t1_t10)
length_texts = sapply(X = dictionaries,FUN=nrow)
hapaxs = mapply(function(i)sum(dictionaries[[i]]$freq==1)/length_texts[i],1:
10)
```

Con il *tagging* della parte del discorso, classifichiamo una parola con la sua parte del discorso corrispondente. Di seguito viene fornito un esempio: "Le idee verdi incolori dormono furiosamente". Abbiamo due aggettivi (JJ), un sostantivo plurale (NNS), un verbo (VBP) e un avverbio (RB). L'analisi POS (*Part-Of-Speech*) può quindi essere utilizzata per prevedere

il POS dato lo stato corrente del testo, confrontando la grammatica di diversi testi, l'interazione uomo-calcolatore o la traduzione da una lingua all'altra. Il seguente approccio al *tagging* POS è molto simile a quello che si ha per la *sentiment analysis*. Abbiamo un dizionario POS e possiamo utilizzare una sorta di *inner join* per allegare le parole al loro POS (Bouveyron et al., 2019). Sfortunatamente, questo approccio è irrealisticamente semplicistico, poiché sarebbe necessario adottare ulteriori misure per garantire che le parole siano classificate correttamente. Ad esempio, senza ulteriori informazioni, non siamo in grado di dire se alcune parole vengono utilizzate come nomi o verbi (essere umano *vs.* essere una parte problematica del discorso). Tuttavia, questo esempio può servire come punto di partenza. Procediamo con i POS di ogni singolo discorso:

```
## POS su dizionario complessivo
t1_t10_pos = rdr_pos(x = paste(dictionary$term,collapse = " "),object = rdr_
model(language = "Italian", annotation = "UniversalPOS"))
head(t1_t10_pos)
```

```
##   doc_id token_id  token   pos
## 1     d1        1     te  PRON
## 2     d1        2  amore  NOUN
## 3     d1        3     me  PRON
## 4     d1        4 perchè SCONJ
## 5     d1        5  cuore  NOUN
## 6     d1        6   vita  NOUN
```

Di seguito sono state individuate le comunalità sul dizionario complessivo, indagando prima le comunalità ed unicità del dizionario:

```
comun_texts = compute_communality(dfmx = dfm_t1_t10)
print(comun_texts)
```

```
## $X
##
.. .. .. .. .. ..
## $overall
## [1] 0.0184972
```

dove l'output originale mostra tutti i valori delle possibili combinazioni di comunalità di dizionario tra i vari *sub-corpora*. "$overall" riporta l'indice di comunalità complessivo approssimabile in questo caso al 2%. Mentre, per quanto riguarda la comunalità rispetto

aggettivi ed avverbi, come riportato di seguito, l'indice di comunalità complessivo è approssimabile al 1% ai verbi ed al 2% rispetto agli aggettivi.

```
dfm_t1_t10_verb = dfm_keep(x=dfm_t1_t10, pattern=t1_t10_pos$token[t1_t10_pos
$pos=="VERB"])
dfm_t1_t10_adj = dfm_keep(x=dfm_t1_t10, pattern=t1_t10_pos$token[t1_t10_pos$
pos=="ADJ"])

comun_texts_verb = compute_communality(dfmx = dfm_t1_t10_verb)
print(comun_texts_verb)

## $X
##

## $overall
## [1] 0.01338072

comun_texts_adj = compute_communality(dfmx = dfm_t1_t10_adj)
print(comun_texts_adj)

## $X
##

## $overall
## [1] 0.01774398
```

L'ultimo passaggio relativo alla preparazione del *corpus* consiste nel ridefinire la DFM completa al netto di quei *token* che potrebbero non risultare utili ai fini dell'analisi. Dunque, sono state eliminate le parole non interpretabili (ad esempio avverbi e pronomi) e ridotta la sparsità nella matrice (numero di elementi con valore zero diviso per il numero totale di elementi), riducendo i termini con bassa frequenza.

```
dfm_t1_t10_refined = dfm_keep(x=dfm_t1_t10, pattern=t1_t10_pos$token[t1_t10_
pos$pos%in%c("VERB","ADJ","NOUN","PROPN","NUM")])

# riduciamo la sparsità
dfm_t1_t10_trimmed = dfm_trim(dfm_t1_t10_refined, min_termfreq = 100)
```

1.7 Analisi descrittiva del testo

Diverse statistiche possono essere utilizzate per descrivere, esplorare e analizzare un corpus di testo. Una tecnica popolare consente, ad esempio, di classificare il valore informativo delle parole all'interno di un *corpus* e quindi visualizzare quelle parole più

informative come una nuvola di parole (*wordcloud*). Ciò consente di ottenere una rapida descrizione dell'oggetto principale del corpus (Welbers et al., 2017). Il peso delle parole-chiave, che viene rappresentato con caratteri di dimensioni diverse, è inteso esclusivamente come frequenza di utilizzo all'interno del sito. In tale rappresentazione, più grande è il carattere, maggiore sarà la frequenza della parola-chiave. Di seguito viene riportato il *wordcloud* del *corpus* complessivo.

```
textplot_wordcloud(dfm_t1_t10_trimmed, min_count = 8, random_order = FALSE,
rotation = .25, color = c("deepskyblue","dodgerblue3","darkblue"))
```

Corpus Sanremo - 1951-2006

Il tema centrale di tutto il *corpus* ruota attorno all'unità testuale "amore" con dei gregari "cuore" e "vita". Tuttavia, risulta interessante andare a visualizzare lo stesso grafico nei singoli *sub-corpora*:

```r
# t1
textplot_wordcloud(dfm_t1_t10_trimmed[1,], min_count = 8, random_order =
FALSE, rotation = .25, color = c("deepskyblue","dodgerblue3","darkblue"))


# t2
textplot_wordcloud(dfm_t1_t10_trimmed[2,], min_count = 8, random_order = FAL
SE, rotation = .25, color = c("deepskyblue","dodgerblue3","darkblue"))



# t3
textplot_wordcloud(dfm_t1_t10_trimmed[3,], min_count = 8, random_order = FAL
SE, rotation = .25, color = c("deepskyblue","dodgerblue3","darkblue"))



# t4
textplot_wordcloud(dfm_t1_t10_trimmed[4,], min_count = 8, random_order = FAL
SE, rotation = .25, color = c("deepskyblue","dodgerblue3","darkblue"))



# t5
textplot_wordcloud(dfm_t1_t10_trimmed[5,], min_count = 8, random_order = FAL
SE, rotation = .25, color = c("deepskyblue","dodgerblue3","darkblue"))



# t6
textplot_wordcloud(dfm_t1_t10_trimmed[6,], min_count = 8, random_order = FAL
SE, rotation = .25, color = c("deepskyblue","dodgerblue3","darkblue"))



# t7
textplot_wordcloud(dfm_t1_t10_trimmed[7,], min_count = 8, random_order = FAL
SE, rotation = .25, color = c("deepskyblue","dodgerblue3","darkblue"))



# t8
textplot_wordcloud(dfm_t1_t10_trimmed[8,], min_count = 8, random_order = FAL
SE, rotation = .25, color = c("deepskyblue","dodgerblue3","darkblue"))



# t9
textplot_wordcloud(dfm_t1_t10_trimmed[9,], min_count = 8, random_order = FAL
SE, rotation = .25, color = c("deepskyblue","dodgerblue3","darkblue"))



# t10
textplot_wordcloud(dfm_t1_t10_trimmed[10,], min_count = 8, random_order = FA
LSE, rotation = .25, color = c("deepskyblue","dodgerblue3","darkblue"))
```

Sub-corpus 1 - 1951-1956

Sub-corpus 4 - 1968-1973

Sub-corpus 2 - 1956-1962

Sub-corpus 5 - 1973-1978

Sub-corpus 3 - 1962-1968

Sub-corpus 6 - 1978-1984

Sub-corpus 7 - 1984-1990

Sub-corpus 9 - 1995-2000

Sub-corpus 8 - 1990-1995

Sub-corpus 10 - 2000-2006

CAPITOLO II

Per l'analisi dei dati si propone un approccio di *clustering* funzionale. Tale metodo è un processo di apprendimento automatico non supervisionato dove un algoritmo crea un modello identificando determinati schemi nel testo e classificando i documenti di testo in gruppi simili definiti *cluster* (Welbers, et al., 2017). Il clustering favorisce l'individuazione di dati omogenei permettendo di valutare se esistono dei gruppi non casuali, cioè che mostrano delle regolarità e che condividono certe caratteristiche. L'unica influenza del ricercatore è la specificazione di alcuni parametri, come il numero di categorie in cui sono classificati i documenti. Esistono molti metodi di *clustering*, che vanno da approcci euristici come *k-means* e analisi di *linkage* a procedure più formali basate su modelli (James & Sugar, 2003). L'intento di questo elaborato è quello di applicare al corpus Sanremo l'approccio di Clustering di curve *model-based* (*Model-Based Curve Clustering,* MBCC) di analisi dei dati testuali funzionali (*Functional Textual Data Analysis,* FTDA) usato in Trevisani & Tuzzi (2015). Questo modello identifica e raggruppa parole chiave che mostrano andamenti temporali simili. Per fare questo si rappresentano e confrontano le singole traiettorie di ciascuna parola chiave impiegando una scomposizione basata sulle c.d. *wavelet* delle traiettorie del segnale e un modello funzionale per raggrupparle. Il seguente metodo risulta particolarmente efficace quando si è in presenza di dati con molte dimensioni e funzioni irregolari.

2.2 Descrizione del modello MBCC

L'evoluzione temporale di una parola chiave è espressa dalla sequenza delle sue occorrenze su un insieme di punti temporali. Questi dati discreti possono essere immaginati come l'osservazione discreta di una curva, cioè un'osservazione funzionale.
Si parla di analisi longitudinale quando si considera un campione di osservazioni sulle stesse *n* unità campionarie per *T* periodi temporali. Dunque, in questo caso si avranno le osservazioni $y_i(t)$, con $i = 1, 2, \ldots, n$ e $t = t1, t2, \ldots, tM$, con $M = 2^j$ e dove *i* indica l'unità campionaria di osservazioni e l'indice j indica il periodo temporale di osservazioni.

Le serie temporali, o meglio i dati funzionali così generati, pongono alcuni problemi: curve irregolari, alta variabilità interindividuale (inter-parola) e alta dimensionalità (Trevisani & Tuzzi, 2015). Nei *corpora* cronologici, i dati sono tipicamente scarsi nel tempo, quindi molte celle della tabella di contingenza hanno un conteggio ridotto o sono vuote. Nel *corpus* Sanremo questo è dovuto alla discrepanza tra il numero di parole distinte con il numero relativamente basso di *token* corrispondenti, ovvero la dimensione del *sub-corpora* del punto temporale. La ricchezza di informazioni e la regolarità dei segnali corrispondenti possono variare notevolmente nel tempo.

Le frequenze assolute delle parole chiave sono state sostituite con le rispettive frequenze relative. Così facendo si è diviso la frequenza di una parola su un punto temporale per il numero totale di *token* in tutti i ritornelli riferiti allo stesso punto temporale. Questo procedimento ha ridotto parzialmente l'effetto delle dimensioni del *sub-corpora*. Questo avviene perché una particolare parola chiave potrebbe avere una frequenza così bassa da suggere di non considerarla in quel punto temporale, questo porterebbe alla errata eliminazione della funzione associata a quella determinata parola.

2.3 Scomposizione *wavelet* delle traiettorie dei segnali

Nonostante esista una solida letteratura che ha studiato il *clustering* funzionale attraverso funzioni *spline* come base per la decomposizione del segnale (James & Sugar, 2003), le decomposizioni *wavelet-based* possono adattarsi a una gamma più ampia di forme funzionali risultando più flessibili delle *spline* e più efficienti dal punto di vista computazionale (Giacofci et al., 2013). Inoltre, le fluttuazioni specifiche delle parole, spesso

una delle principali fonti di variabilità, possono essere considerate effetti casuali delle parole, oltre alla componente fissa principale nel quadro del modello funzionale (Trevisani & Tuzzi, 2015). Per queste ragioni è stata applicata la classe dei *Functional Clustering Mixed-Model* (FCMM), *wavelet-based*, originariamente sviluppata da Giacofci et al. (2013) nel contesto dei problemi di *clustering* per la gestione dei *corpora* cronologici.
Essendo i dati da trattare discreti si è fatto ricorso alle Discrete Wavelet Transform (DWT) per esaminare le funzioni sull'insieme di *M* punti campionati. Esse restituiscono un vettore di dati della stessa lunghezza dell'input. In questo modo si decompone la funzione originale in un insieme di funzioni *wavelet* ortogonali (Giacofci et al., 2013) in grado, amplificandole o riducendole, di approssimare la funzione originale con maggiore efficienza e risparmiando risorse di calcolo. La rappresentazione *wavelet* si basa su un *wavelet* padre (o *scaling*) φ e un *wavelet* madre (o semplicemente *wavelet*) ψ.

La curva y_i(t) ha la seguente decomposizione:

$$y_i(t) = \sum_{k=0}^{2^{j_0}-1} c^*_{i,j_0k}\phi_{j_0k}(t) + \sum_{j\geq j_0}\sum_{k=0}^{2^j-1} d^*_{i,jk}\psi_{jk}(t)$$

con c_i e d_i sono i coefficienti per la decomposizione.

2.4 FCMM *wavelet-based*

Nel dominio *wavelet*, il FCMM fa riferimento a un modello lineare *mixed-effect* che può essere utilizzato per creare un algoritmo di clustering *model-based*. Per la stima della massima verosimiglianza si applica di solito l'algoritmo EM (*Expectation Maximization*), implementato all'interno del pacchetto R "curvclust" (Giacofci et al., 2012), che implementa il metodo proposto in Giacofci et al. (2013).

CAPITOLO III

Il *model-based clustering* presuppone che i dati siano stati generati da un modello e cerca di recuperare il modello originale dai dati. Questo modello andrà a definire quindi i cluster ed una loro assegnazione di documenti. Un criterio comunemente utilizzato per stimare i parametri del modello è la stima della massima verosimiglianza tramite l'algoritmo EM.

I dati discreti possono essere visualizzati come una tabella di contingenza (matrice) di parole chiave × punti temporali, dove le righe manifestano le osservazioni discrete di oggetti continui rappresentati da relazioni funzionali (analisi dei dati funzionali). Nel contesto di questo studio non vi era l'aspettativa di identificare e isolare automaticamente traiettorie di parole-chiave facilmente interpretabili. Se questa ipotesi fosse stata vera si sarebbero dovute osservare parole la cui presenza è cresciuta nel tempo e il cui trend crescente è in corso; parole che sono diventate meno frequenti e sono scomparse; parole che hanno avuto molto successo e parole che sono state costantemente presenti nel tempo. Al contrario, molte traiettorie mostrano picchi multipli che sono difficili da interpretare in termini di evoluzione cronologica, sebbene possano nascondere interessanti schemi di regolarità. La logica di analisi adottata è stata più di natura esplorativa, nonostante la tecnica statistica utilizzata presupponeva l'esistenza di modelli temporali prototipici delle parole-chiave.

3.2 Applicazione del MBCC in R

L'MBCC si basa su un modello di probabilità *finite-mixture* di distribuzioni multivariate, con particolare enfasi sulla famiglia più utilizzata di modelli, vale a dire miscele di distribuzioni normali multivariate. Per tenere in considerazione la diversa dimensione dei testi a disposizione per ciascun periodo di tempo, la forte asimmetria presente nella frequenza delle parole e la generale sparsità dei dati, non si può prescindere dal sottoporre i dati ad una trasformazione preliminare. Lo scopo di questo studio è esaminare come diverse trasformazioni agiscono sui risultati di *curve clustering* in termini di quantità e composizione dei gruppi di parole. Nel nostro caso useremo trasformazioni *Wavelet* utilizzando il comando `CCDred=getUnionCoef(CCD).` In particolare. dopo aver imposto che le varianze e le covarianze siano costanti attraverso il comando:

```
CCO["Gamma2.structure"]="constant"
```

si procede con l'esecuzione dell'algoritmo EM per la stima dei parametri dei modelli mediante il comando `CCR=getFCMM(CCDred,CCO).`
.Per quanto riguarda il numero di cluster da considerare nell'algoritmo EM, in questo lavoro si è ipotizzata l'esistenza di 4 cluster (sebbene un'appropriata *sensitivity analysis* potrebbe far evidenziare i limiti di tale scelta).

```
# fissiamo il numero di cluster K
K=4
# reshaping del dato
X=quanteda::as.matrix(dfm_t1_t10_trimmed)
fdat=split(X, rep(1:ncol(X), each = nrow(X)))
#data
CCD=new("CClustData",Y=fdat,filter.number=1)#smoothness
CCDred=getUnionCoef(CCD)
#options
CCO=new("CClustO")
CCO["nbclust"]=K
CCO["Gamma2.structure"]="constant"
CCR=getFCMM(CCDred,CCO)#Functional Clustering Mixed Models {FCMM,FCM,FMM}

## [ initialization with 100 burns using SEM ]
## [EM algorithm]
```

```
summary(CCR)
```

```
## Number of clusters  4
## cluster size  0.8380328 0.01923077 0.1331211 0.009615385
## labels
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   3   1   1   1   1   1   1   1   1   2   2   1   1   1   1   1   3   1   3   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   1   1   1   1   1   1   1   1   3   3   1   3   1   1   1   1   1   3   1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   1   3   3   4   1   3   3   3   1   1   1   1   1   1   1   1   1   1   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   1   1   1   1   3   1   1   1   1   1   1   1   1   1   1   1   1   3   1   1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 101 102 103 104
##   1   1   1   1
```

```
cluster=apply(CCR["Tau"],1,which.max)
```

Con i seguenti comandi è stato possibile visualizzare le prime 10 parole ottenendo i seguenti coefficienti:

```
library(pander)
panderOptions('knitr.auto.asis', FALSE)
c_i=sapply(CCDred@wavecoef,'[[',1)[1:10]
cat(paste0('Coefficienti Wavelet : C_i prime 10 parole'))
```

```
## Coefficienti Wavelet : C_i prime 10 parole
```

```
  pander(data.frame(id=1:10,C_i=c_i))
```

```
#
```

```
#
```

```
## ------------
##  id    C_i
## ---- -------
##  1     151
##
##  2    61.5
##
##  3    51.75
##
##  4    56.75
##
##  5    29.5
##
##  6    26.5
##
##  7    36.75
##
##  8    29.75
##
##  9    39.25
##
##  10   76.75
## ------------
```

```r
library(pander)
panderOptions('knitr.auto.asis', FALSE)
d_i=sapply(CCDred@wavecoef,'[[',2)[,1:10]
colnames(d_i)=colnames(X)[1:10]
rownames(d_i)=paste0('t_',1:10)
cat(paste0('Coefficienti Wavelet : D_i prime 10 parole in 10 follow-up'))
```

```
## Coefficienti Wavelet : D_i prime 10 parole in 10 follow-up
```

```r
  pander(d_i)
```

```
##
## ------------------------------------------------------------------------
##          va      vento      dire     sento    canto     vivo    volta
## ---------- -------- -------- -------- -------- -------- -------- --------
##  **t_1**    14.14    -9.899   -6.364   -11.31   9.899    -4.95    2.828
##
##  **t_2**    -31.82   -5.657   7.778    -4.243   -8.485   7.778    0.7071
##
##  **t_3**    -38.18   -16.97   -12.73     0      -2.121   0.7071   2.828
##
##  **t_4**    21.92    -6.364   4.243    6.364    -1.414   2.828    7.071
##
##  **t_5**    31.11     4.95    2.121    25.46    10.61    0.7071   2.828
##
##  **t_6**     -6.5      2        -4       6        4        -1       -3.5
##
##  **t_7**      41       31.5     19.5      36      11.5     19.5      18
##
##  **t_8**    -87.68   -27.93   -29.7    -1.061   -10.96   -1.061   -20.15
##
##  **t_9**    28.99    22.27    13.79    25.46    8.132    13.79    12.73
##
##  **t_10**    110       30     32.25    20.75      18       7      18.75
## ------------------------------------------------------------------------
##
## Table: Table continues below
## ---------------------------------------
##       felicitÃ     dolce     amor
## ---------- ---------- --------- --------
##  **t_1**     -4.243     1.414    -25.46
##
##  **t_2**    -0.7071     2.828    38.89
##
##  **t_3**     -21.92    -7.071    -10.61
##
##  **t_4**    -0.7071    -12.02    0.7071
##
##  **t_5**     1.414     -2.828      0
##
##  **t_6**      8.5        14       56.5
##
##  **t_7**       3         14        0
##
##  **t_8**     1.768     -0.3536   85.91
##
##  **t_9**     2.121      9.899      0
##
##  **t_10**    26.75      25.25    76.75
## ---------------------------------------
```

Con i seguenti codici è stato possibile visualizzare tutte le parole-chiave sottoposte all'analisi.

```
library(pander)
panderOptions('knitr.auto.asis', FALSE)
X_1=summary(dfm_t1_t10_trimmed)
X_1=data.table(X_1)
cl=sort(unique(cluster))
for(i in 1:length(cl)){
pos=as.numeric(names(which(cluster==cl[i])))
t1=paste0('X_1[j %in% pos,Cluster:=',cl[i],']')
eval(parse(text = t1))
}
X_1[,fx:=(x/sum(x)),by=Cluster]
pos=as.numeric(names(which(cluster==cl[1])))
Parole=colnames(X)[pos]
cat(paste0('Words : Group 1'))
```

Osserviamo infine i risultati del *clustering* ottenuto. I grafici mostrano in ascissa le sequenze temporali suddivise in intervalli di 5/6 anni a partire dal 1951 (1 = '51-'56; 2 = '56-'62; 3 = '62-'68; 4 = '68-73; 5 = '73-'78; 6 = '78-'84; 7 = '84-'90; 8 = '90-'95; 9 = '95-2000; 10 = 2000-2006) mentre in ordinata è descritta la frequenza relativa delle parole chiave.

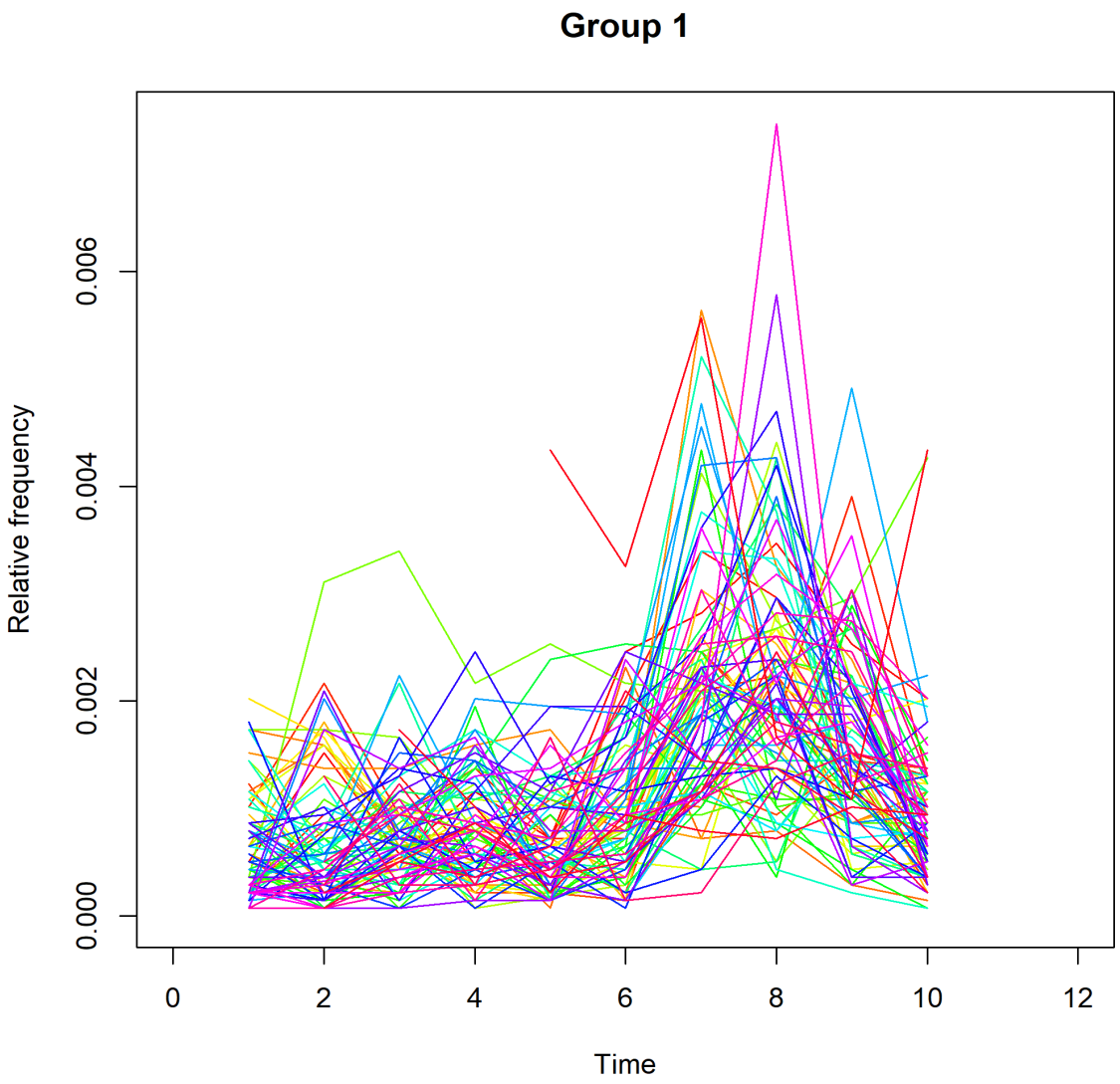
Il primo *Cluster* contiene le parole-chiave : *vento, dire, sento, canto, vivo, volta, felicità, dolce, bella, amare, mille, anni, età, sogno, sogno, mani, bacio, piano, , musica, luna, amo, parole, vedo, sorriso, porta, volte, stella, volo, vola, guarda, vai, luce, sola, detto, dio, altro, stelle, canzone, canta, sera, anima, altra, bello, buio, voce, penso, dimmi, grande, blu, città, strada, casa, sembra, storia, amico, acqua, insieme_a, gente, sa, donna, giorni, com, mano, vero, dolore, amori, morire, vieni, soli, uomo, paura, senti, voglia, terra, vivere, andare, forte, bisogno, stare, ami, pensare, basta, silenzio, verità.*
Il seguente codice permette di visualizzare queste parole-chiave come output di R:

```
## Words : Group 1

  pander(data.frame(Cluster=1,Parole))
```

Mentre i seguenti comandi mostrano il grafico per il *Cluster* 1;

```
# G1
# plot Clust1
plot(0,0,xlim = c(0,12),ylim = c(0,X_1[Cluster==1,max(fx)]),type = "n",main=
'Group 1',xlab='Time',ylab='Relative frequency')
words=unique(X_1[Cluster==1,j][])
N=length(words)
col1 <- rainbow(N)
for(k in 1:N){
a=X_1[Cluster==1 & j==words[k],.(x=i,y=fx)]
lines(a,type='l',col=col1[k])
}
```



Line Plot Group 1

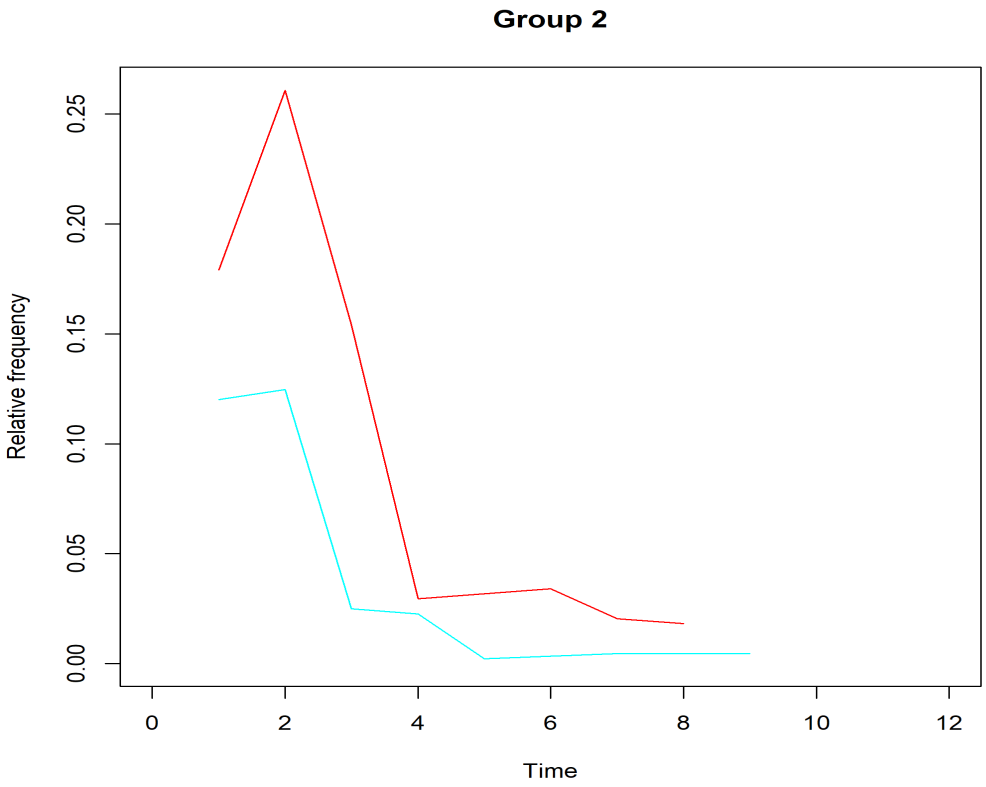
L'analisi è proseguita con i risultati del *clustering* per *Cluster* 2, che rappresentano le parole-chiave: *amor*, *cuor*. Di seguito sono mostrati i codici che permettono di visualizzare le parole-chiave ed il grafico relativo.

```
pos=as.numeric(names(which(cluster==cl[2])))
Parole=colnames(X)[pos]
cat(paste0('Words : Group 2'))

## Words : Group 2

  pander(data.frame(Cluster=2,Parole))

# G2
# plot Clust1
plot(0,0,xlim = c(0,12),ylim = c(0,X_1[Cluster==2,max(fx)]),type = "n",main=
'Group 2',xlab='Time',ylab='Relative frequency')
words=unique(X_1[Cluster==2,j][])
N=length(words)
col1 <- rainbow(N)
for(k in 1:N){
a=X_1[Cluster==2 & j==words[k],.(x=i,y=fx)]
lines(a,type='l',col=col1[k])
}
```



Line Plot Group 2

Il terzo *Cluster* contiene le parole-chiave: *va, tempo, vita, sole, mare, vuoi, cielo, occhi, giorno, notte, via, cuore, sai, mondo*.

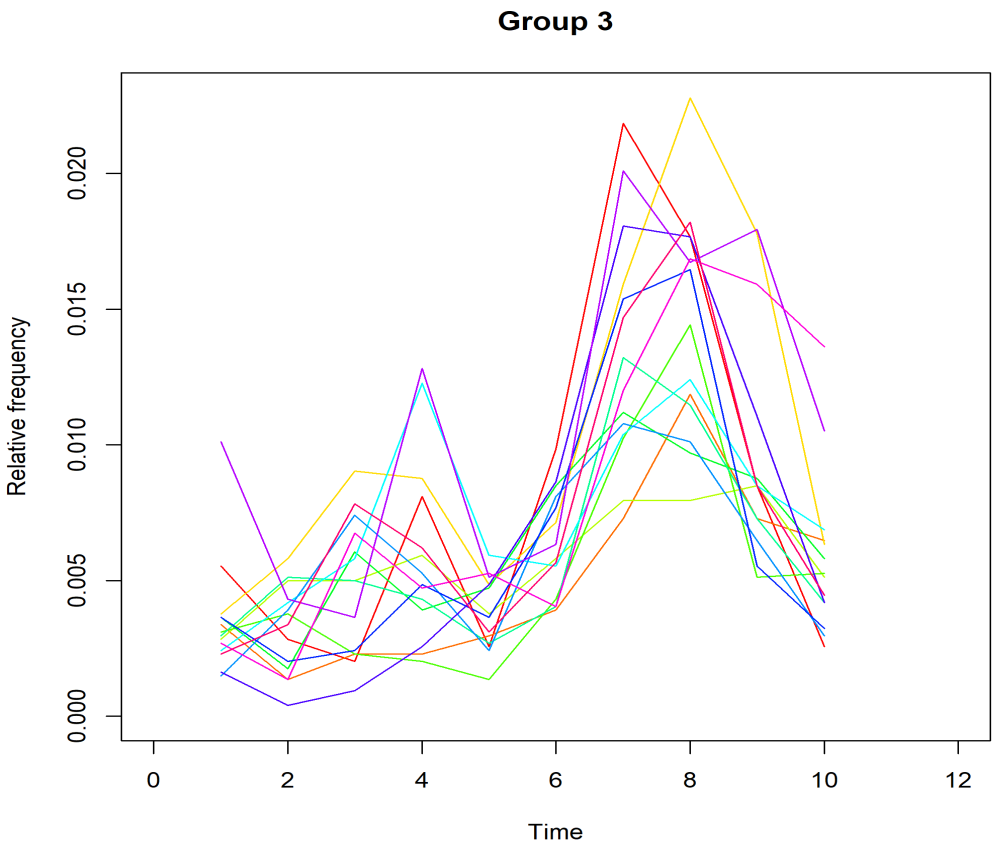
Di seguito i risultati del *clustering* per Cluster **3**:

```
pos=as.numeric(names(which(cluster==cl[3])))
Parole=colnames(X)[pos]
cat(paste0('Words : Group 3'))

## Words : Group 3

  pander(data.frame(Cluster=3,Parole))

# G3
# plot Clust1
plot(0,0,xlim = c(0,12),ylim = c(0,X_1[Cluster==3,max(fx)]),type = "n",main=
'Group 3',xlab='Time',ylab='Relative frequency')
words=unique(X_1[Cluster==3,j][])
N=length(words)
col1 <- rainbow(N)
for(k in 1:N){
a=X_1[Cluster==3 & j==words[k],.(x=i,y=fx)]
lines(a,type='l',col=col1[k])
}
```



Line Plot Group 3

Infine, il quarto *cluster* comprende solamente la parola-chiave: *amore*.

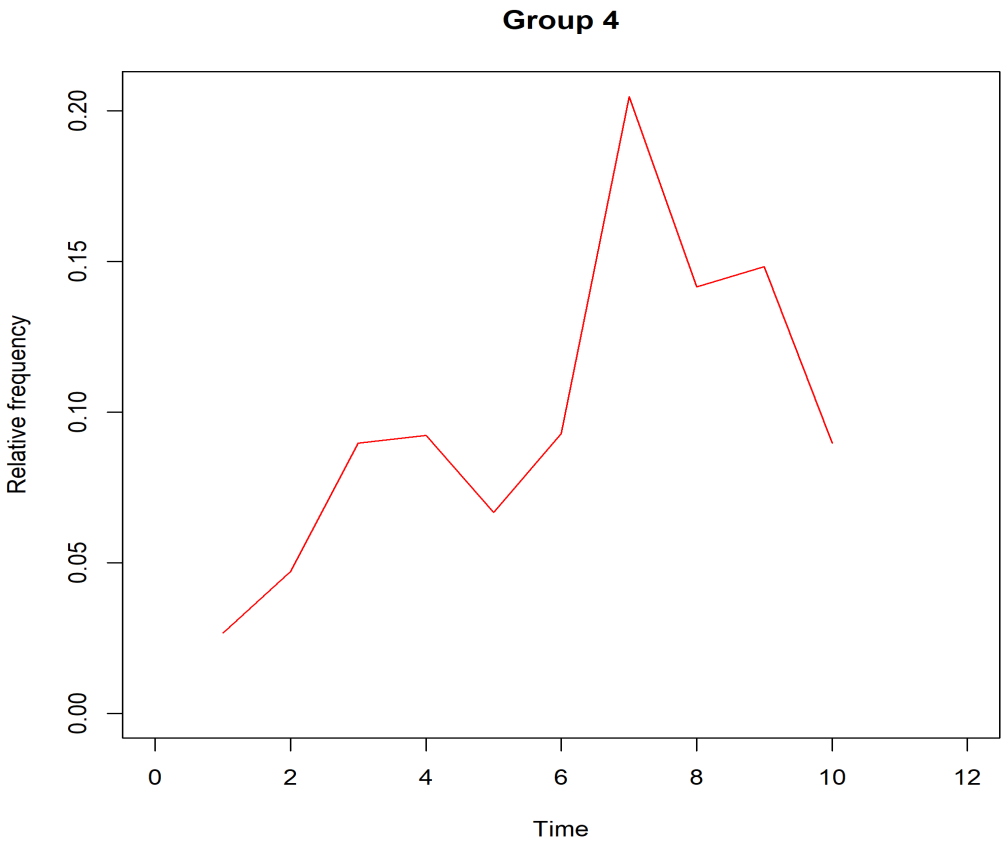
Di seguito sono presentati i codici con i risultati del *clustering* per Cluster **4** :

```
pos=as.numeric(names(which(cluster==cl[4])))
Parole=colnames(X)[pos]
cat(paste0('Words : Group 4'))

## Words : Group 4

  pander(data.frame(Cluster=4,Parole))

# G4
# plot Clust1
plot(0,0,xlim = c(0,12),ylim = c(0,X_1[Cluster==4,max(fx)]),type = "n",main=
'Group 4',xlab='Time',ylab='Relative frequency')
words=unique(X_1[Cluster==4,j][])
N=length(words)
col1 <- rainbow(N)
for(k in 1:N){
a=X_1[Cluster==4 & j==words[k],.(x=i,y=fx)]
lines(a,type='l',col=col1[k])
}
```



Line Plot Group 4

3.3 Interpretazione dei risultati

I grafici funzionali hanno in ascissa i punti temporali delimitati dai seguenti anni (cut point): 1951 – 1956 – 1962 – 1968 – 1973 – 1978 – 1984 – 1990 – 1995 – 2000 – 2006, mentre in ordinata vi è la frequenza relativa di ciascuna sequenza temporale di *token*. Come si può notare diversi cluster corrispondono a diversi pattern oppure a diverse grandezze delle frequenze relative.

ll primo cluster ha pattern con picco negli anni ’90 e parole caratterizzate da una frequenza relativa bassa e orientate a tematiche di amore. Il maggior numero di parole contenute in questo cluster in linea con il fatto che il festival di Sanremo è la vetrina di canzoni che, nella vasta maggioranza, trattano d’ amore.

Il secondo cluster, che contiene le parole-chiave “amor” e “cuor”, ha un *pattern* diverso dagli altri, in particolare, si rileva un picco negli anni ’50/’60, coerente con il linguaggio aulico/arcaico delle canzoni del tempo, e un andamento decrescente negli anni successivi. Questo andamento è confermato dalle *wordcloud*: le due parole decrescono di dimensione nel corso del tempo.

Il terzo cluster ha un andamento simile al primo (raggiunge l’apice negli anni ’90) ma con due differenze: la presenza di un picco secondario negli anni ’60 maggiormente definito e la frequenza relativa più alta. Inoltre, il significato delle parole in esso contenute si riferiscono più a viaggi e a luoghi piuttosto che a temi sentimentali. Infatti, osservando le *wordcloud* sono subito evidenti le parole-chiave “mare”, “terra”, vita”, “via” che sicuramente si distinguono dalle parole legate ai sentimenti viste sopra.

Il quarto cluster potrebbe essere interpretato come un artefatto dovuto ad un *k* troppo alto, che ha forzato l’algoritmo a trovare un quarto gruppo associandolo al termine più frequente, data la poca varianza di argomenti delle canzoni di Sanremo. Anche in questo caso, nonostante il *pattern* sia molto simile al terzo *cluster*, esso si distingue per il fatto di essere molto più frequente.

All’interno di ciascun cluster non si notano evidenti artefatti o curve isolate rispetto a quelle appartenenti allo stesso cluster.

3.4 Limiti dall'analisi e proposte future

L'applicazione del modello statistico MBCC descritto nel capitolo II al *corpus* Sanremo mostra l'opportunità di recuperare gli argomenti che sono apparsi in passato e che sono oggi ancora presenti nelle canzoni, mediante l'evoluzione delle parole-chiave nei ritornelli pubblicati dagli artisti nel corso del tempo. Tuttavia, il *corpus* Sanremo è un insieme di ritornelli di canzoni nel tempo, con diversa frequenza per gli anni presentati. Frazionare nel tempo le canzoni ad intervalli di tempo (anche regolari) di anni rischia di far emergere l'eterogeneità dei temi che tende a crearsi nei vari ritornelli. Questo effetto potrebbe essere amplificato dalla logica longitudinale e dalla frammentazione culturale osservata negli anni del Festival.

I risultati di FTDA e MBCC applicati al *corpus* Sanremo hanno la potenzialità di mettere in evidenza modelli temporali prototipici e *cluster* di parole chiave con modelli simili, in qualche modo la storia delle canzoni del Festival della canzone italiana. Tuttavia, ci sono vari aspetti della modellazione statistica su cui si potrebbe approfondire in futuro. Delle possibili proposte di lavoro implicherebbero: la scelta di sistemi di base alternativi alle *wavelet* per l'applicazione di una funzione di filtro il cui scopo è evidenziare i *pattern* significativi (*smoothing*) dei dati funzionali; l'opportunità di ridurre ulteriormente i dati mediante l'analisi delle componenti principali funzionali; l'uso di distribuzioni alternative a Normale per le curve di frequenza (Trevisani & Tuzzi, 2015). Inoltre, la MBCC potrebbe essere utilizzata efficacemente per seguire un *panel/focus group* formato appositamente dal ricercatore per osservare unità testuali dominanti nel tempo. In questo modo l'analisi non soffrirebbe della scarsa "numerosità campionaria" per parola-chiave rilevata nel seguente lavoro.

BIBLIOGRAFIA

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., & Matsuo, A. (2018). quanteda: An R package for the quantitative analysis of textual data. *Journal of Open Source Software*, *3*(30), 774. Retrieved from https://CRAN.R-project.org/package=readtext

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., & Matsuo, A. (2018). quanteda: An R package for the quantitative analysis of textual data. *Journal of Open Source Software*, *3*(30), 774. Retrieved from http://quanteda.io

Bouchet-Valat, M. (2014). SnowballC: Snowball stemmers based on the C libstemmer UTF-8 library. *R package version 0.5*, *1*. Retrieved from: https://cran.r-project.org/web/packages/SnowballC/index.html

Bouveyron, C., Celeux, G., Murphy, T. B., & Raftery, A. E. (2019). *Model-based clustering and classification for data science: with applications in R* (Vol. 50). Cambridge University Press.

Crawley, M. J. (2013). The R book. ed. *Wiley*, *5*, 744.

Dowle, M., Srinivasan, A., Gorecki, J., Chirico, M., Stetsenko, P., Short, T., ... & Ritchie, S. (2019). Package 'data. table'. *Extension of 'data. frame*. Retrieved from: https://cran.r-project.org/web/packages/data.table/index.html

Giacofci, M., Lambert-Lacroix, S., Marot, G., & Picard, F. (2012). Curvclust: curve clustering. *R package version 0.0*, *1*. Maggiori informazioni: http://www2.uaem.mx/r-mirror/web/packages/curvclust/curvclust.pdf

Giacofci, M., Lambert-Lacroix, S., Marot, G., & Picard, F. (2013). Wavelet-based clustering for mixed-effects functional models in high dimension. *Biometrics*, *69*(1), 31-40.

Hornik, K., & Grün, B. (2011). topicmodels: An R package for fitting topic models. *Journal of statistical software*, *40*(13), 1-30. Retrieved from: https://cran.r-project.org/web/packages/topicmodels/index.html

James, G. M., & Sugar, C. A. (2003). Clustering for sparsely sampled functional data. *Journal of the American Statistical Association*, *98*(462), 397-408.

Jockers, M. L., & Thalken, R. (2020). *Text Analysis with R*. Springer International Publishing.

Meyer, D., Hornik, K., & Feinerer, I. (2008). Text mining infrastructure in R. *Journal of statistical software*, *25*(5), 1-54. Retrived from https://cran.r-project.org/web/packages/tm/index.html

Team, R. C. (2017). R: A language and environment for statistical computing [Computer software; Version 3.3. 3]. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Tan, A. H. (1999, April). Text mining: The state of the art and the challenges. In *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases* (Vol. 8, pp. 65-70). sn.

Trevisani, M., & Tuzzi, A. (2015). A portrait of JASA: the History of Statistics through analysis of keyword counts in an early scientific journal. *Quality & Quantity*, *49*(3), 1287-1304.

Tuzzi, A. (2003). *L'analisi del contenuto: introduzione ai metodi e alle tecniche di ricerca*. Carocci editore.

Welbers, K., Van Atteveldt, W., & Benoit, K. (2017). Text analysis in R. *Communication Methods and Measures*, *11*(4), 245-265.