Use of R's Data Frames: an Illustration with Chinese Dialects Data

Norm Matloff
Department of Computer Science
University of California at Davis
Davis, CA 95616 USA
matloff@cs.ucdavis.edu

October 5, 2010

Aspects of R Illustrated Here

- data frames
- filtering
- string manipulation
- **lapply()**, **sapply()**
- **which()**, **split()**, **merge()**

The Chinese Languages

- Standard Chinese is Mandarin/*putonghua*/*guoyu*, pretty much *lingua franca* status.

The Chinese Languages

- Standard Chinese is Mandarin/*putonghua*/*guoyu*, pretty much *lingua franca* status.
- But dialects still going strong, e.g. Cantonese, Shanghainese.

The Chinese Languages

- Standard Chinese is Mandarin/*putonghua*/*guoyu*, pretty much *lingua franca* status.
- But dialects still going strong, e.g. Cantonese, Shanghainese.
- Mandarin speaker from Beijing who wishes do business in Hong Kong may get "closer" to clients by speaking Cantonese.

The Chinese Languages

- Standard Chinese is Mandarin/*putonghua*/*guoyu*, pretty much *lingua franca* status.
- But dialects still going strong, e.g. Cantonese, Shanghainese.
- Mandarin speaker from Beijing who wishes do business in Hong Kong may get "closer" to clients by speaking Cantonese.
- HKer with weak Mandarin may want to improve.

The Chinese Languages

- Standard Chinese is Mandarin/*putonghua*/*guoyu*, pretty much *lingua franca* status.
- But dialects still going strong, e.g. Cantonese, Shanghainese.
- Mandarin speaker from Beijing who wishes do business in Hong Kong may get "closer" to clients by speaking Cantonese.
- HKer with weak Mandarin may want to improve.
- Thus some rules, "mappings," that will show correspondences between the dialects would be a useful learning aid.

The Chinese Languages

- Standard Chinese is Mandarin/*putonghua*/*guoyu*, pretty much *lingua franca* status.
- But dialects still going strong, e.g. Cantonese, Shanghainese.
- Mandarin speaker from Beijing who wishes do business in Hong Kong may get "closer" to clients by speaking Cantonese.
- HKer with weak Mandarin may want to improve.
- Thus some rules, "mappings," that will show correspondences between the dialects would be a useful learning aid. **Find them with R!**

How Different Are the Dialects from One Another?

- Differences between pronunciations sometimes striking.

How Different Are the Dialects from One Another?

- Differences between pronunciations sometimes striking.
- E.g. the character for "down," 下, is pronounced

How Different Are the Dialects from One Another?

- Differences between pronunciations sometimes striking.
- E.g. the character for "down," 下, is pronounced
  - *xia* in Mandarin,

How Different Are the Dialects from One Another?

- Differences between pronunciations sometimes striking.
- E.g. the character for "down," 下, is pronounced
  - *xia* in Mandarin,
  - *ha* in Cantonese

How Different Are the Dialects from One Another?

- Differences between pronunciations sometimes striking.
- E.g. the character for "down," 下, is pronounced
  - *xia* in Mandarin,
  - *ha* in Cantonese
  - and *wu* in Shanghainese.

How Different Are the Dialects from One Another?

- Differences between pronunciations sometimes striking.
- E.g. the character for "down," 下, is pronounced
  - *xia* in Mandarin,
  - *ha* in Cantonese
  - and *wu* in Shanghainese.
- Some differences in grammar too.

How Different Are the Dialects from One Another?

- Differences between pronunciations sometimes striking.
- E.g. the character for "down," 下, is pronounced
  - *xia* in Mandarin,
  - *ha* in Cantonese
  - and *wu* in Shanghainese.
- Some differences in grammar too.
- (For simplicity, will not discuss tones.)

How Can R Help?

- There ARE some patterns.

How Can R Help?

- There ARE some patterns.
- E.g. in Mandarin *xia* and Cantonese *ha* above, the $x \longrightarrow h$ correspondence is common,

How Can R Help?

- There ARE some patterns.
- E.g. in Mandarin *xia* and Cantonese *ha* above, the $x \rightarrow h$ correspondence is common, e.g. 香 ("fragrant"), is pronounced *xiang* in Mandarin and *heung* in Cantonese.

How Can R Help?

- There ARE some patterns.
- E.g. in Mandarin *xia* and Cantonese *ha* above, the $x \rightarrow h$ correspondence is common, e.g. 香 ("fragrant"), is pronounced *xiang* in Mandarin and *heung* in Cantonese.
- But the correspondence $x \rightarrow y$ is also common, e.g. 休 ("to rest") is *xiu* in Mandarin, *yau* in Cantonese.

How Can R Help?

- There ARE some patterns.
- E.g. in Mandarin *xia* and Cantonese *ha* above, the $x \rightarrow h$ correspondence is common, e.g. 香 ("fragrant"), is pronounced *xiang* in Mandarin and *heung* in Cantonese.
- But the correspondence $x \rightarrow y$ is also common, e.g. 休 ("to rest") is *xiu* in Mandarin, *yau* in Cantonese.
- Also $x \rightarrow s$, etc.

Our Goal

We wish to write R code to input dialect data frame, output a list of correspondences like $x \longrightarrow h$.

Our Goal

We wish to write R code to input dialect data frame, output a list of correspondences like $x \longrightarrow h$.
Example: Mandarin speaker wants to learn Cantonese.

Our Goal

We wish to write R code to input dialect data frame, output a list of correspondences like $x \longrightarrow h$.
Example: Mandarin speaker wants to learn Cantonese.
She wonders, what does an initial Mandarin x map to?

Our Goal

We wish to write R code to input dialect data frame, output a list of correspondences like $x \longrightarrow h$.
Example: Mandarin speaker wants to learn Cantonese.
She wonders, what does an initial Mandarin x map to?
The R code tells her that the rules $x \longrightarrow h$ and $x \longrightarrow s$ are the most common ones for initial consonant x.

Our Goal

We wish to write R code to input dialect data frame, output a list of correspondences like $x \longrightarrow h$.
Example: Mandarin speaker wants to learn Cantonese.
She wonders, what does an initial Mandarin x map to?
The R code tells her that the rules $x \longrightarrow h$ and $x \longrightarrow s$ are the most common ones for initial consonant x.
It also lists *all* mappings for x, i.e. the Cantonese pronunciations for all characters pronounced x- in Mandarin.

Our Goal

We wish to write R code to input dialect data frame, output a list of correspondences like $x \longrightarrow h$.
Example: Mandarin speaker wants to learn Cantonese.
She wonders, what does an initial Mandarin x map to?
The R code tells her that the rules $x \longrightarrow h$ and $x \longrightarrow s$ are the most common ones for initial consonant x.
It also lists *all* mappings for x, i.e. the Cantonese pronunciations for all characters pronounced x- in Mandarin.
Example: Cantonese speaker wants to learn Mandarin. R tells him that the Cantonese ending *-im* maps most often to a Mandarin *-ian* or *-an*. Etc.

Example of Use

Our main function's name is **mapsound()**.
Example input data frame:

Example of Use

Our main function's name is **mapsound()**.
Example input data frame:

```
> head(canman8)
 Ch char    Can    Man Can cons Can sound Can tone Man cons Man sound Man tone
1      一   yat1    yi1        y        at        1        y         i        1
2      丁  ding1  ding1        d       ing        1        d       ing        1
3      七  chat1    qi1       ch        at        1        q         i        1
4      丈 jeung6 zhang4        j      eung        6       zh       ang        4
5      上 seung5 shang3        s      eung        5       sh       ang        3
6      下    ha5   xia4        h         a        5        x        ia        4
```

Example, cont'd.

Example call:

```
> m2cx <- mapsound(canman8,c("Man cons","Can cons"),"x")
```

Example, cont'd.

Example call:

```
> m2cx <- mapsound(canman8,c("Man cons","Can cons"),"x")
```

Here's one component of the output:

Example, cont'd.

Example call:

```
> m2cx <- mapsound(canman8,c("Man cons","Can cons"),"x")
```

Here's one component of the output:

```
> m2cx$counts
ch  f  g  h  j  k kw  n  s  y
15  2  1 87 12  4  2  1 81 21
```

Example, cont'd.

Example call:

```
> m2cx <- mapsound(canman8,c("Man cons","Can cons"),"x")
```

Here's one component of the output:

```
> m2cx$counts
ch  f  g  h  j  k kw  n  s  y
15  2  1 87 12  4  2  1 81 21
```

So, if we see a Mandarin *x*, it probably maps to *h* or *s* in Cantonese. Not a perfect rule, but helps a lot.

Example, cont'd.

Example call:

```
> m2cx <- mapsound(canman8,c("Man cons","Can cons"),"x")
```

Here's one component of the output:

```
> m2cx$counts
ch  f  g  h  j  k kw  n  s  y
15  2  1 87 12  4  2  1 81 21
```

So, if we see a Mandarin *x*, it probably maps to *h* or *s* in Cantonese. Not a perfect rule, but helps a lot.
Let's see some of the images of the mappings, say the one for *ch*:

Example, cont'd.

Example call:

```
> m2cx <- mapsound(canman8,c("Man cons","Can cons"),"x")
```

Here's one component of the output:

```
> m2cx$counts
ch  f  g  h  j  k kw  n  s  y
15  2  1 87 12  4  2  1 81 21
```

So, if we see a Mandarin *x*, it probably maps to *h* or *s* in Cantonese. Not a perfect rule, but helps a lot.

Let's see some of the images of the mappings, say the one for *ch*:

```
> head(m2cx$images[["ch"]])
     Ch char   Can  Man Can cons Can sound Can tone Man cons Man sound Man tone
613       嗅 chau3 xiu4       ch        au        3        x        iu        4
982       尋 cham4 xin2       ch        am        4        x        in        2
1050      巡 chun3 xun2       ch        un        3        x        un        2
1173      徐 chui4  xu2       ch        ui        4        x         u        2
1184      循 chun3 xun2       ch        un        3        x        un        2
1566      斜  che4 xie2       ch         e        4        x        ie        2
```

Overview of the R code:

Overview of the R code:

- **mapsound()**: Finds the actual mappings, as seen above.

Overview of the R code:

- **mapsound()**: Finds the actual mappings, as seen above.
- **merge2fy()**: Combines two 1-dialect data frames to produce a 2-dialect frame, which is input to **mapsound()**.

Overview of the R code:

- **mapsound()**: Finds the actual mappings, as seen above.
- **merge2fy()**: Combines two 1-dialect data frames to produce a 2-dialect frame, which is input to **mapsound()**.
- **sepsoundtone()**: Takes a character's pronunciation, e.g. *tian1*, and breaks it into 3 sound components, e.g. *t*, *ian* and 1. Called by **merge2fy()**.

Code for **mapsound()**

```
1  mapsound <- function(df,cols,sourceval) {
2     fromcol <- cols[1]
3     tocol <- cols[2]
4     # find row numbers correspond value to be mapped
5     base <- which(df[[fromcol]] == sourceval)
6     # extract data frame for those rows
7     basedf <- df[base,]
8     # determine which rows of basedf correspond to the various mapped
9     # values
10    sp <- split(1:nrow(basedf),basedf[[tocol]])
11    retval <- list()
12    # call R's length() function on each of the mapped vectors, thereby
13    # counts of each mapping
14    retval$counts <- sapply(sp,length)
15    # get the characters for each mapping
16    retval$images <- lapply(sp,function(mappedvec) basedf[mappedvec,])
17    return(retval)
18 }
```

Code to Merge Data of Two Dialects

But there is prep work that must be done:

Code to Merge Data of Two Dialects

But there is prep work that must be done:

Recall the example input data frame:

	Ch char	Can	Man	Can cons	Can sound	Can tone	Man cons	Man sound	Man tone
1	一	yat1	yi1	y	at	1	y	i	1
2	丁	ding1	ding1	d	ing	1	d	ing	1

...

Code to Merge Data of Two Dialects

But there is prep work that must be done:
Recall the example input data frame:

	Ch char	Can	Man	Can cons	Can sound	Can tone	Man cons	Man sound	Man tone
1	一	yat1	yi1	y	at	1	y	i	1
2	丁	ding1	ding1	d	ing	1	d	ing	1
...									

This came from merging two dfs, Cantonese and Mandarin. Here's part of the Cantonese df (Mandarin one is similar):

Code to Merge Data of Two Dialects

But there is prep work that must be done:
Recall the example input data frame:

```
  Ch char    Can    Man Can cons Can sound Can tone Man cons Man sound Man tone
1      一   yat1    yi1        y         at        1        y         i 1
2      丁  ding1  ding1        d        ing        1        d       ing 1
...
```

This came from merging two dfs, Cantonese and Mandarin. Here's part of the Cantonese df (Mandarin one is similar):

```
> head(can8)
  Ch char   Can
1      一  yat1
2      乙 yuet3
3      丁 ding1
...
```

Code to Merge Data of Two Dialects

But there is prep work that must be done:
Recall the example input data frame:

```
  Ch char    Can    Man Can cons Can sound Can tone Man cons Man sound Man tone
1      一   yat1    yi1        y         at        1        y         i 1
2      丁  ding1  ding1        d        ing        1        d       ing 1
...
```

This came from merging two dfs, Cantonese and Mandarin. Here's part of the Cantonese df (Mandarin one is similar):

```
> head(can8)
  Ch char   Can
1      一  yat1
2      乙 yuet3
3      丁 ding1
...
```

I wrote the function **merge2fy()** to merge the two dfs—and split the pronunciations into 3 sound components.

Code for **merge2fy()**

```
1   merge2fy <- function(fy1,fy2) {
2      outdf <- merge(fy1,fy2)
3      # separate tone from sound, and create new columns
4      for (fy in list(fy1,fy2)) {
5         # saplout will be a matrix, init consonants in row 1, remaining
6         # sounds in row 2, and tones in row 3
7         saplout <- sapply((fy[[2]]),sepsoundtone)
8         # convert it to a data frame
9         tmpdf <- data.frame(fy[,1],t(saplout),row.names=NULL,
10           stringsAsFactors=F)
11        # add names to the columns
12        consname <- paste(names(fy)[[2]]," cons",sep="")
13        restname <- paste(names(fy)[[2]]," sound",sep="")
14        tonename <- paste(names(fy)[[2]]," tone",sep="")
15        names(tmpdf) <- c("Ch char",consname,restname,tonename)
16        # need to use merge(), not cbind(), due to possibly different
17        # ordering of fy, outdf
18        outdf <- merge(outdf,tmpdf)
19     }
20     return(outdf)
21  }
```

Code for **sepsoundtone()**

```
1   sepsoundtone <- function(pronun) {
2      nchr <- nchar(pronun)
3      vowels <- c("a","e","i","o","u")
4      # how many initial consonants?
5      numcons <- 0
6      for (i in 1:nchr) {
7         ltr <- substr(pronun,i,i)
8         if (!ltr %in% vowels) numcons <- numcons + 1 else break
9      }
10     cons <- if (numcons > 0) substr(pronun,1,numcons) else NA
11     tone <- substr(pronun,nchr,nchr)
12     # final character will be the tone, if any
13     numtones <- if (tone %in% letters)) 0 else 1
14     if (numtones == 0) tone <- NA
15     therest <- substr(pronun,numcons+1,nchr-numtones)
16     return(c(cons,therest,tone))
17  }
```

More Work to Be Done

- Find mappings if initial letter is a vowel.

More Work to Be Done

- Find mappings if initial letter is a vowel.
- Some characters have multiple readings.

More Work to Be Done

- Find mappings if initial letter is a vowel.
- Some characters have multiple readings.
- Try to map the tones.