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()`
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!**
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
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.)
There ARE some patterns.

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

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

E.g. in Mandarin \textit{xia} and Cantonese \textit{ha} above, the \( x \rightarrow h \) correspondence is common, e.g. 香 ("fragrant"), is pronounced \textit{xiang} in Mandarin and \textit{heung} in Cantonese.

But the correspondence \( x \rightarrow y \) is also common, e.g. 休 ("to rest") is \textit{xiu} in Mandarin, \textit{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 $x \rightarrow h$.

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 $x \rightarrow h$ and $x \rightarrow s$ 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.
Our Goal

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

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 $x \rightarrow h$.

**Example:** Mandarin speaker wants to learn Cantonese. She wonders, what does an initial Mandarin $x$ map to?

Cantonese speaker wants to learn Mandarin. R tells him that the Cantonese ending $-im$ maps most often to a Mandarin $-ian$ or $-an$. Etc.
Our Goal

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

**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 \( x \rightarrow h \) and \( x \rightarrow s \) 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 $x \rightarrow h$.

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

**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 $x \rightarrow h$ and $x \rightarrow s$ 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.
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:

```r
> 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 call:

```r
> m2cx <- mapsound(canman8,c("Man cons","Can cons"),"x")
```
Example, cont’d.

Example call:

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

Here’s one component of the output:
Example, cont’d.

Example call:

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

Here’s one component of the output:

```r
> m2cx$counts
      ch  f  g  h  j  k  kw  n  s  y
1  15  2  1  87 12  4  2  1  81 21
```
Example, cont’d.

Example call:

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

Here’s one component of the output:

```r
> 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:

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

Here’s one component of the output:

```r
> 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 call:

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

Here's one component of the output:

```r
> 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:

```r
> head(m2cx$images[["ch"]])
```

<table>
<thead>
<tr>
<th>Ch</th>
<th>char</th>
<th>Can</th>
<th>Man</th>
<th>Can cons</th>
<th>Can sound</th>
<th>Can tone</th>
<th>Man cons</th>
<th>Man sound</th>
<th>Man tone</th>
</tr>
</thead>
<tbody>
<tr>
<td>613</td>
<td>嗅</td>
<td>chau3 xiu4</td>
<td>ch</td>
<td>au</td>
<td>3</td>
<td>x</td>
<td>iu</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>982</td>
<td>尋</td>
<td>cham4 xin2</td>
<td>ch</td>
<td>am</td>
<td>4</td>
<td>x</td>
<td>in</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1050</td>
<td>巡</td>
<td>chun3 xun2</td>
<td>ch</td>
<td>un</td>
<td>3</td>
<td>x</td>
<td>un</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1173</td>
<td>徐</td>
<td>chui4 xu2</td>
<td>ch</td>
<td>ui</td>
<td>4</td>
<td>x</td>
<td>u</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1184</td>
<td>循</td>
<td>chun3 xun2</td>
<td>ch</td>
<td>un</td>
<td>3</td>
<td>x</td>
<td>un</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1566</td>
<td>斜</td>
<td>che4 xie2</td>
<td>ch</td>
<td>e</td>
<td>4</td>
<td>x</td>
<td>ie</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
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()`. 
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()**.
```r
mapsound <- function(df, cols, sourceval) {
  fromcol <- cols[1]
  tocol <- cols[2]
  # find row numbers correspond value to be mapped
  base <- which(df[[fromcol]] == sourceval)
  # extract data frame for those rows
  basedf <- df[base,]
  # determine which rows of basedf correspond to the various mapped 
  # values
  sp <- split(1:nrow(basedf), basedf[[tocol]])
  retval <- list()
  # call R’s length() function on each of the mapped vectors, thereby 
  # counts of each mapping
  retval$counts <- sapply(sp, length)
  # get the characters for each mapping
  retval$images <- lapply(sp, function(mappedvec) basedf[mappedvec,])
  return(retval)
}
```
But there is prep work that must be done:
But there is prep work that must be done:
Recall the example input data frame:

<table>
<thead>
<tr>
<th>Ch</th>
<th>char</th>
<th>Can</th>
<th>Man</th>
<th>Can cons</th>
<th>Can sound</th>
<th>Can tone</th>
<th>Man cons</th>
<th>Man sound</th>
<th>Man tone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yat1</td>
<td>yil</td>
<td>y</td>
<td>at</td>
<td>1</td>
<td>y</td>
<td>i1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ding1</td>
<td>ding1</td>
<td>d</td>
<td>ing</td>
<td>1</td>
<td>d</td>
<td>ing1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

```r
head(can8)
```

I wrote the function `merge2fy()` to merge the two dfs—and split the pronunciations into 3 sound components.
But there is prep work that must be done:
Recall the example input data frame:

<table>
<thead>
<tr>
<th>Ch</th>
<th>char</th>
<th>Can</th>
<th>Man</th>
<th>Can cons</th>
<th>Can sound</th>
<th>Can tone</th>
<th>Man cons</th>
<th>Man sound</th>
<th>Man tone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>一</td>
<td>yat1</td>
<td>yi1</td>
<td>y</td>
<td>at</td>
<td>1</td>
<td>y</td>
<td>i1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>丁</td>
<td>ding1</td>
<td>ding1</td>
<td>d</td>
<td>ing</td>
<td>1</td>
<td>d</td>
<td>ing1</td>
<td></td>
</tr>
</tbody>
</table>

This came from merging two dfs, Cantonese and Mandarin. Here’s part of the Cantonese df (Mandarin one is similar):
But there is prep work that must be done:
Recall the example input data frame:

<table>
<thead>
<tr>
<th>Ch char</th>
<th>Can</th>
<th>Man</th>
<th>Can cons</th>
<th>Can sound</th>
<th>Can tone</th>
<th>Man cons</th>
<th>Man sound</th>
<th>Man tone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 一</td>
<td>yat1</td>
<td>yi1</td>
<td>y  at</td>
<td>1</td>
<td>y</td>
<td>i 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 丁</td>
<td>ding1</td>
<td>ding1</td>
<td>d  ing</td>
<td>1</td>
<td>d</td>
<td>ing 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

```r
> head(can8)
   Ch char  Can
1   一  yat1
2  丁  yuet3
3  丁 ding1
```
But there is prep work that must be done:

Recall the example input data frame:

<table>
<thead>
<tr>
<th>Ch</th>
<th>char</th>
<th>Can</th>
<th>Man</th>
<th>Can cons</th>
<th>Can sound</th>
<th>Can tone</th>
<th>Man cons</th>
<th>Man sound</th>
<th>Man tone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yat1</td>
<td>y</td>
<td>at</td>
<td>1</td>
<td>y</td>
<td>i1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ding1</td>
<td>ding1</td>
<td>d</td>
<td>ing</td>
<td>1</td>
<td>d</td>
<td>ing</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

```r
> head(can8)
  Ch char  Can
1  — 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()`

```r
merge2fy <- function(fy1, fy2) {
  outdf <- merge(fy1, fy2)
  # separate tone from sound, and create new columns
  for (fy in list(fy1, fy2)) {
    # saplout will be a matrix, init consonants in row 1, remaining
    # sounds in row 2, and tones in row 3
    saplout <- sapply(fy[[2]], sepsoundtone)
    # convert it to a data frame
    tmpdf <- data.frame(fy[, 1], t(saplout), row.names=NULL,
                        stringsAsFactors=F)
    # add names to the columns
    consname <- paste(names(fy)[[2]], " cons", sep=" ")
    restname <- paste(names(fy)[[2]], " sound", sep=" ")
    tonename <- paste(names(fy)[[2]], " tone", sep=" ")
    names(tmpdf) <- c("Ch char", consname, restname, tonename)
    # need to use merge(), not cbind(), due to possibly different
    # ordering of fy, outdf
    outdf <- merge(outdf, tmpdf)
  }
  return(outdf)
}
```
```r
sepsoundtone <- function(pronun) {
  nchr <- nchar(pronun)
  vowels <- c("a","e","i","o","u")
  # how many initial consonants?
  numcons <- 0
  for (i in 1:nchr) {
    ltr <- substr(pronun,i,i)
    if (!ltr %in% vowels) numcons <- numcons + 1 else break
  }
  cons <- if (numcons > 0) substr(pronun,1,numcons) else NA
  tone <- substr(pronun,nchr,nchr)
  # final character will be the tone, if any
  numtones <- if (tone %in% letters) 0 else 1
  if (numtones == 0) tone <- NA
  therest <- substr(pronun,numcons+1,nchr-numtones)
  return(c(cons,therest,tone))
}
```
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.