

Business Analytics Programming

Lab 7 - Structuring and Analyzing a Data Frame

Dr. Wajahat Gilani

Rutgers Business School

December 8, 2019

Table Information

We now have the `resdf` table, lets view the basic structure:

```
1 resdf.info()
```

Data columns (total 41 columns): 10 Km 5905 non-null object

5 Mi 5905 non-null object

5 Mile 21118 non-null object

A 3949 non-null object

AG 9831 non-null object

Ag 56398 non-null object

DIV /TOT 6208 non-null object

Div /Tot 38951 non-null object

Div /Tot 12434 non-null object

Div/Tot 5237 non-null object

GUN 3623 non-null object

GUN TIM 3017 non-null object

Gun 12210 non-null object

Gun Ti 3949 non-null object

Gun Tim 25812 non-null object

Table Information - Year and Name Columns

Name 60347 non-null object
Net 12210 non-null object
Net Ti 3949 non-null object
Net Tim 20575 non-null object
Num 56019 non-null object
PACE 3191 non-null object
PLACE 9831 non-null object
Pace 48516 non-null object
Pace0 5905 non-null object
Pace1 5905 non-null object
Place 60347 non-null object
S 10514 non-null object
Split 5277 non-null object
TIME 3191 non-null object
Time 18376 non-null object
Year 70178 non-null int64
g Hometown 3949 non-null object

m 3949 non-null object
m0 3949 non-null object
dtypes: int64(1), object(40)

View First Row

```
1 dd = resdf.iloc[0]
```

Spyder (Python 3.7)		dd - Series	
Index		0	
HOMETOWN		nan	
Hometown		Kenya	
Hometown	Net Tim	nan	
NAME		nan	
NET		nan	
NET TIM		nan	
NUM		nan	
Name		Allan Kiprono	
Net		nan	
Net Ti		nan	
Net Tim		nan	
Num		9	
PACE		nan	
PLACE		nan	
Pace		4:32	
Pace0		nan	
Pace1		nan	

It's a mess (2 names)

View First Row

```
1 dd = resdf.iloc[0]
```

Spyder (Python 3.7)		dd - Series	
Index		0	
HOMETOWN		nan	
Hometown		Kenya	
Hometown	Net Tim	nan	
NAME		nan	
NET		nan	
NET TIM		nan	
NUM		nan	
Name		Allan Kiprono	
Net		nan	
Net Ti		nan	
Net Tim		nan	
Num		9	
PACE		nan	
PLACE		nan	
Pace		4:32	
Pace0		nan	
Pace1		nan	

It's a mess (2 names)

Reduce the Columns

There are a lot of repeating columns, but the best place to get context about the table in terms of size, is with columns that are **mandatory**. In this case, the runner's **name** and the **year** they ran is a must (we created the year column).

```
resdf[ 'Year' ].shape (70178, 41)
resdf[ resdf[ 'Name' ].notnull() ].shape (60347, 41)
resdf[ resdf[ 'NAME' ].notnull() ].shape (9831, 41)
```

The two name columns add up to 70,178, so we can merge them into one column.

```
resdf.loc[ resdf[ 'Name' ].isnull(), 'Name' ] = resdf[ 'NAME' ]
```

```
resdf[ resdf[ 'Name' ].notnull() ].shape (70178, 41)
```

```
del resdf[ 'NAME' ]
resdf.shape (70178, 40)
```

Reduce the Columns - Time

Index	0
10 Km	nan
5 Mi	nan
5 Mile	22:32
A	nan
AG	nan
Ag	22
DIV /TOT	nan
Div /Tot	1/347
Div /Tot	nan
Div/Tot	nan
GUN	nan
GUN TIM	nan
Gun	nan
Gun Ti	nan
Gun Tim	nan
HOMETOWN	nan
Hometown	Kenya

There is a 10 Km column, but two 5 "mile" columns. Which one(s) should we keep? Are they relevant?

Reduce the Columns - Time

```
resdf[resdf['10 Km'].notnull()].shape (5905, 40)
resdf[resdf['5 Mi'].notnull()].shape (5905, 40)
resdf[resdf['5 Mile'].notnull()].shape (21118, 40)
```

The numbers don't add up.

```
resdf[resdf['10 Km'].notnull()].groupby('Year')['Year'].count()
```

Year

2008 5905

```
resdf[resdf['5 Mi'].notnull()].groupby('Year')['Year'].count()
```

Year

2008 5905

```
resdf[resdf['5 Mile'].notnull()].groupby('Year')['Year'].count()
```

Year

2010 6912

2011 7012

2012 7194

Reduce the Columns - Time

```
resdf['Year'].unique()
```

```
array([2012, 2011, 2010, 2009, 2008,  
       2007, 2006, 2005, 2004, 2003, 2002,  
       2001, 2000, 1999])
```

Lets choose a year that doesn't have one of those columns, and see what values it has for time.

```
df2007 = resdf[resdf['Year']==2007]  
df2007=df2007[df2007.columns[~df2007.isnull().all()]]
```

This allows us to view just one particular year, with its particular columns.

Index	Hometown	Name	Num	Pace	Place	S	Split	Time	Year
33674	hiopia	Tadesse...	13	4:37	1		28:47	46:01	2007
33675	anzania	John Yu...	7	4:37	2		28:48	46:04	2007
33676	enya	John Ko...	2001	4:38	3		28:48	46:11	2007
33677	enya	Nichola...	21	4:38	4		28:47	46:12	2007

The Time column is what we want, lets check to see if the other dates have it.

```
resdf[resdf['Time'].notnull()].shape  
resdf[resdf['TIME'].notnull()].shape
```

```
(18376, 40)  
(3191, 40)
```

Reduce the Columns - Time

```
1 resdf.loc[resdf['Time'].notnull() | resdf['TIME'].notnull(), 'Year'].unique()  
  
array([2012, 2008, 2007, 1999])
```

We can cheat a little bit and view the first 2 rows for every year.

TIME	Time	Year
nan	45:15	2012
nan	46:28	2012
nan	nan	2011
nan	nan	2011

```
1 resdf['time']=resdf['Time']  
2 resdf.loc[resdf['TIME'].notnull(), 'time']=resdf['TIME']  
3  
4 resdf[resdf['time'].notnull()].groupby('Year')['Year'].count()  
()
```

Year 1999 3191 2007 5277 2008 5905 2012 7194

Reduce the Columns - Time

```
1 resdf.loc[resdf['Net Tim'].notnull(), 'time']=resdf['Net Tim']
2 resdf[resdf['time'].notnull()].groupby('Year')['Year'].count
()
```

Year
1999 3191
2007 5277
2008 5905
2009 6651
2010 6912
2011 7012
2012 7194

The only dates missing are: 2006, 2005, 2004, 2003, 2002, 2001, and 2000

We can filter our **dfs** table for those years.

```
1 dfs = dfs.loc[dfs.Year.isin([2006, 2005,
2 2004, 2003, 2002, 2001, 2000])]
```

```
1 resdf.loc[resdf['Gun Tim'].notnull(), 'time']=resdf['Gun Tim']
2 resdf.loc[resdf['Net'].notnull(), 'time']=resdf['Net']
3
4 resdf[resdf['time'].notnull()].groupby('Year')['Year'].count
()
```

1999 3191 **2002** 3725 **2004** 4157 **2005** 4328 **2006** 5237 **2007** 5277 **2008**
5905 **2009** 6651 **2010** 6912 **2011** 7012 **2012** 7194

Reduce the Columns - Time

```
1 resdf.loc[resdf['Net Tim'].notnull(), 'time']=resdf['Net Tim']
2 resdf[resdf['time'].notnull()].groupby('Year')['Year'].count
()
```

Year
1999 3191
2007 5277
2008 5905
2009 6651
2010 6912
2011 7012
2012 7194

The only dates missing are: 2006, 2005, 2004, 2003, 2002, 2001, and 2000

We can filter our **dfs** table for those years.

```
1 dfs = dfs.loc[dfs.Year.isin([2006, 2005,
2 2004, 2003, 2002, 2001, 2000])]
```

```
1 resdf.loc[resdf['Gun Tim'].notnull(), 'time']=resdf['Gun Tim']
2 resdf.loc[resdf['Net'].notnull(), 'time']=resdf['Net']
3
4 resdf[resdf['time'].notnull()].groupby('Year')['Year'].count
()
```

1999 3191 **2002** 3725 **2004** 4157 **2005** 4328 **2006** 5237 **2007** 5277 **2008**
5905 **2009** 6651 **2010** 6912 **2011** 7012 **2012** 7194

Reduce the Columns - Time

```
1 resdf.loc[resdf['NET'].notnull(), 'time']=resdf['NET']
2 resdf.loc[resdf['NET TIM'].notnull(), 'time']=resdf['NET TIM']
3 resdf.loc[resdf['Gun Ti'].notnull(), 'time']=resdf['Gun Ti']
4
5 resdf[resdf['time'].notnull()].groupby('Year')['Year'].count
  ()
```

We now have all the times for all the year, we delete the other columns.

```
1 del resdf['Time']
2 del resdf['TIME']
3 del resdf['Net Tim']
4 del resdf['Gun Tim']
5 del resdf['Net']
6 del resdf['NET']
7 del resdf['NET TIM']
8 del resdf['Gun Ti']
9 del resdf['10 Km']
10 del resdf['5 Mi']
11 del resdf['5 Mile']
```

Reduce the Columns - Location

```
dfs = resdf.groupby('Year').head(2).reset_index(drop=True)
```

HOMETOWN	Hometown	Hometown	Net Tim
nan	Ethiopia	nan	
nan	Tanzania	nan	
nan	nan	Kenya	47:24
nan	nan	Kenya	47:34
nan	Kenya	nan	
nan	Kenya	nan	
nan	KEN	nan	
nan	KEN	nan	
nan	nan	nan	
nan	nan	nan	
nan	Ken	nan	
nan	Ken	nan	
Kenya	nan	nan	
Kenya	nan	nan	
Kenya	nan	nan	
Kenya	nan	nan	
Ethiopia	nan	nan	

looks like we have 2 columns in one.

Names are not standard, and

Reduce the Columns - str.extract

The column name has unknown spaces, so it's best to just copy and paste it.

```
resdf.columns
```

```
Index(['A', 'AG', 'Ag', 'DIV /TOT', 'Div /Tot', 'Div /Tot', 'Div/Tot',  
'GUN', 'GUN TIM', 'Gun', 'HOMETOWN', 'Hometown', 'Hometown Net  
Tim', 'NUM', 'Name', 'Net Ti', 'Num', 'PACE', 'PLACE', 'Pace', 'Pace0',  
'Pace1', 'Place', 'S', 'Split', 'Year', 'g Hometown', 'm', 'm0', 'time'],  
dtype='object')
```

```
resdf['Hometown2'] = resdf['Hometown Net Tim'].str.  
extract(pat = '([A-Z a-z]+)')
```

```
dfs = resdf.groupby('Year').head(2).reset_index(drop=True)
```

Year	g Hometown	m	m0	time	Hometown2
2007	nan	nan	nan	46:04	nan
2006	nan	nan	nan	47:25	Kenya
2006	nan	nan	nan	47:35	Kenya
2005	nan	nan	nan	46:54	nan
2005	nan	nan	nan	46:57	nan
2004	nan	nan	nan	48:12	nan
2004	nan	nan	nan	48:12	nan
2003	7 KEN	6	5	46:5	nan