SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

Aug 2024

Data Munging Seminar 5

ICT233 Data Programming

Veronica Hu huhe001@suss.edu.sg

RECAP

S4 Key Learning Objectives

1) Assemble datasets together for analysis using Pandas
2) Understand the needs of concatenating datasets and performing the operations on them
3) Understand the needs of merging datasets and performing the operations on them
4) Learn what missing data are and how they are created
5) Work with data issues such as missing and incomplete data during analysis
6) Learn how to use pivot, melt, and normalization operations on datasets

SEMINAR OVERVIEW

Data Munging – LEARNING OBJECTIVES

1) Determine data types in Pandas and convert them between various forms
2) Handle and manipulate string and text data in Pandas
3) Understand how to apply built-in or self-defined functions on Pandas vector
4) Appreciate the use of techniques to group and combine Pandas data for calculation and analytical purposes

ETL Process

Seminar 5

Extract

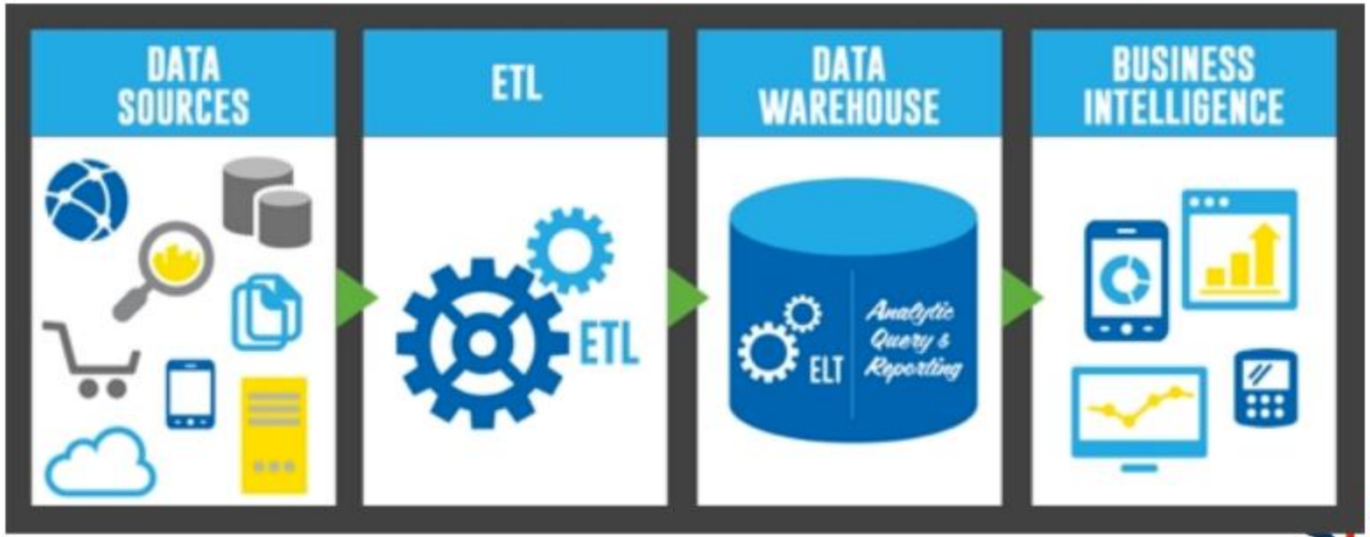
- One or more source systems containing customer, financial, or product data (CRM, Accounting system, Warehouse, MES)
- Files types - Flat files, XML, Oracle, IBM DB2, SQL Server,, IBM Websphere MQ, ODBC, JDBC, Hadoop Distributed File System (HDFS), Hive/HCatalog, JSON, Mainframe (IBM z/OS), Salesforce.com, SAP/R3

Transform

- Applying business rules, cleansing, and validating the data.
- Aggregation, Copy, Join, Sort, Merge, Partition, Filter, Reformat, Lookup
- Mathematical: +, -, x, /, Abs, IsValidNumber, Mod, Pow, Rand, Round, Sqrt, ToNumber, Truncate, Average, Min, Max
- Logical: And, Or, Not, IfThenElse, RegEx, Variables
- Text: Concatenate, CharacterLengthOf, LengthOf, Pad, Replace, ToLower, ToText, ToUpper, Translate, Trim, Hash
- Date: DateAdd, DateDiff, DateLastDay, DatePart, IsValidDate
- Format: ASCII, EBCDIC, Unicode

Load

- Load the results into one or more target systems such as a data warehouse, datamart, or business intelligence reporting system.
- Output: Flat files, XML, Oracle, IBM DB2, SQL Server, Teradata, Sybase, Vertica, Netezza, Greenplum, ODBC, JDBC, Hadoop Distributed File System (HDFS), Hive/HCatalog, Mainframe (IBM z/OS), Salesforce.com, Tableau, QlikView



Chapter 1: Working with Data Types

1.1 Introduction

- Raw data comes in various shapes and sizes, requiring organization for analysis
- Data munging transforms raw data into a usable format for analytics
  - Focuses on handling and converting data types in Pandas
  - Proper data type conversion is necessary for accurate data processing and analysis

Comparison of Pandas vs Python types

Pandas Type	Python Type	Description
`object`	`string`	Most common data type
`int64`	`int`	Whole number
`float64`	`float`	Numbers with decimals
`datetime64`	`datetime`	date and time format (need to be imported this from Python lib)

Chapter 1: Working with Data Types

1.1 Introduction

- Determine Data Types
- Converting Data Types
  - from category to object/string
  - from float to object/string
  - from object/string to numeric value

```
display(tips.head())

# display types of each column
print(tips.dtypes)
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
total_bill     float64
tip            float64
sex           category
smoker        category
day           category
time          category
size             int64
dtype: object
```

Chapter 1: Working with Data Types

1.2 String and Text Manipulation

- Subsetting and slicing strings
- Combining Strings in Pandas

```
string1 = 'python'
string2 = 'programming'
combined_str = string1 + ' ' + string2
print(combined_str)
```

```
python programming
```

```
mystring = 'python programming'

print(mystring[0]) # print the first character
print(mystring[0:5]) # print 1st to 5th characters
print(mystring[5:10]) # print 6th to 10th characters
print(mystring[-1]) # print last character
print(len(mystring)) # print the length of the string
```

```
p
pytho
n pro
g
18
```

```
mystring[::-1]
```

```
'gnimmargorp nohtyp'
```

Chapter 1: Working with Data Types

1.3 Regular Expressions (RegEx)

- Searching for regular expressions
- Extracting data using regular expressions
- Tidying up Data

```
import re
# the pattern to match: a number that is either 8 or 9, followed by 7 more digits
p = re.compile('[89]\d{7}') #This is pattern
mystring = 'The number 98989898 is in this string, \
    the other number is 8191192, \
    this number 33341231 is not a phone number.'

# findall(): search a given string and return a list of identified numbers,

match = p.findall(mystring)

print(match)
print(len(match)) # to return the number of items in result
```

```
['98989898']
1
```

```
# str.extract(): to extract out the numeric part of the searched pattern
df[0].str.extract(r'\w+-(\d+)-\w+')
```

Chapter 2: Applying Functions

2.1 Functions

- apply ()
  - Basics
  - Functions are applied on a column or row vector
  - Column-wise Operations (axis = 0)
  - Row-wise Operations (axis = 1)

```
# To apply a function across a column
display(df['b'].apply(my_sq))
```

```
0     4
1     9
2    16
Name: b, dtype: int64
```

```
# To apply a function across a row
display(df.iloc[1].apply(my_sq)) # applied across 2nd row
```

```
a    400
b      9
Name: 1, dtype: int64
```

```
# Column-wise Operations, defult axis=0
# apply the function across the entire dataframe
df.apply(my_sq,axis=0)
```

	a	b
0	100	4
1	400	9
2	900	16

```
display(df['a'].apply(my_sq)) # applied across column
```

```
0    100
1    400
2    900
Name: a, dtype: int64
```

Chapter 2: Applying Functions

2.2 Manipulating Functions

- Vectorizing a Function
- Lambda Functions

```
def ave(x,y):
    if x < 15:
        return ('No Result')
    else:
        return (x + y) / 2
```

```
import numpy as np

# pass np.vectorize on the function we
# want to vectorise, to create a new function.
new_ave = np.vectorize(ave)
print(new_ave(df['a'], df['b']))
```

```
['No Result' '11.5' '17.0']
```

```
def my_sq(x):
    return x ** 2

# lambda x: expression
# the above my_sq function
# can be rewritten in one
# line like this:

lambda x: x ** 2
```

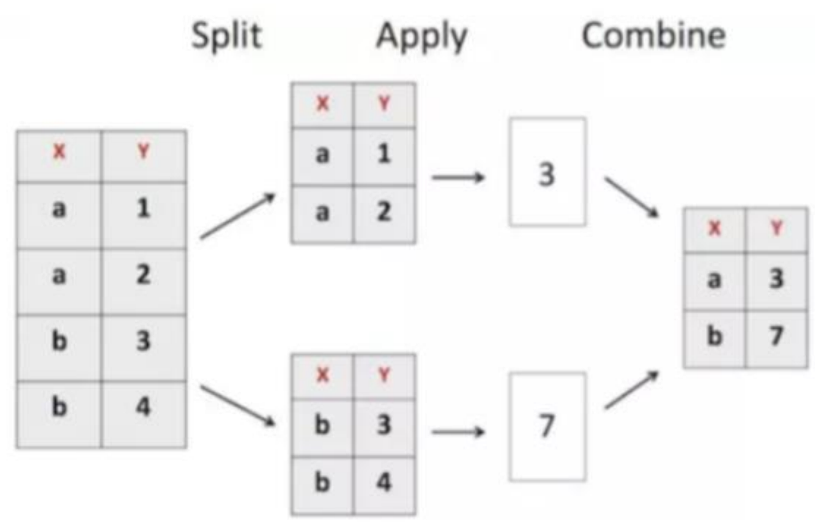
```
<function __main__.<lambda>>
```

Chapter 3: Aggregation

3.1 Split-Apply-Combine

- Recap on Aggregation

  - Grouped and Aggregated Calculations

    - 

df.groupby('x').sum()

le Select sum(y) as total_y from tb group by x

Chapter 3: Aggregation

3.1 Split-Apply-Combine

- Standard Aggregation Function

Function	Description
count	Number of non-NA observations
sum	Sum of values
mean	Mean of values
mad	Mean absolute deviation
median	Arithmetic median of values
min	Minimum
max	Maximum
mode	Mode
std	Bessel-corrected sample standard deviation
var	Unbiased variance
describe	Statistical summary

Chapter 3: Aggregation

3.1 Split-Apply-Combine

- Use of your own Aggregation function

```
df.groupby('a')['b','c'].agg(lambda x: sum(x)/len(x) - 10).reset_index()
```

	a	b	c
0	10	-8	0.5
1	20	-7	5.5
2	30	-6	10.5

- Applying multiple Functions simultaneously

```
def myfunc(x):
    return sum(x)/len(x) - 10
```

```
gdf = gap_df.groupby('year').lifeExp.agg([myfunc,np.count_nonzero, np.mean, np.std])

print(gdf.head())
```

```
         myfunc  count_nonzero       mean        std
year
1950  52.002568           39.0  62.002568   7.874083
1951  55.904167           24.0  65.904167   4.522637
1952  39.206867          143.0  49.206867  12.256571
1953  56.674563           24.0  66.674563   5.819556
1954  57.459817           24.0  67.459817   5.459547
```

Chapter 3: Aggregation

3.1 Split-Apply-Combine

- Use of your own Aggregation function

```
df.groupby('a')['b','c'].agg(lambda x: sum(x)/len(x) - 10).reset_index()
```

	a	b	c
0	10	-8	0.5
1	20	-7	5.5
2	30	-6	10.5

- Applying multiple Functions simultaneously

```
def myfunc(x):
    return sum(x)/len(x) - 10
```

```
gdf = gap_df.groupby('year').lifeExp.agg([myfunc,np.count_nonzero, np.mean, np.std])

print(gdf.head())
```

```
         myfunc  count_nonzero       mean        std
year
1950  52.002568           39.0  62.002568   7.874083
1951  55.904167           24.0  65.904167   4.522637
1952  39.206867          143.0  49.206867  12.256571
1953  56.674563           24.0  66.674563   5.819556
1954  57.459817           24.0  67.459817   5.459547
```

SUSS
SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

SUMMARY

Data Munging – LEARNING OBJECTIVES

1) Determine data types in Pandas and convert them between various forms
2) Handle and manipulate string and text data in Pandas
3) Understand how to apply built-in or self-defined functions on Pandas vector
4) Appreciate the use of techniques to group and combine Pandas data for calculation and analytical purposes

THANK YOU