5.1 - DATA WRANGLING with PANDAS

## **5.1.4 – Cleaning Data in Python**

Chapter-1: Exploring your data

|  |  |  |
| --- | --- | --- |
| Diagnose data for cleaning | | |
|  |  | Common data problems  ● Inconsistent column names  ● Missing data  ● Outliers  ● Duplicate rows  ● Untidy  ● Need to process columns  ● Column types can signal unexpected data values |
| Visually inspect | .head()  .tail()  .columns  .shape  .info() | Missing values  Capitalization  Spaces  Data type inconvenience |
| EDA | | |
| Data type inspection | .info() |  |
| Frequency counts | .column.value\_counts() | df.country.value\_counts(dropna=False).head()  ## “dropna=False” eksik verileri de saydigindan eklemekte fayda var  ## “NaN” for numeric // “missing” for strings |
| Summary statistics | .describe() | ## Outliers: Considerably higher or lower |
| Visual exploratory data analysis | | |
|  |  | ● Great way to spot outliers and obvious errors  ● More than just looking for pa!erns  ● Plan data cleaning steps |
| Look at frequencies | Histogram | Outliers |
|  | Box plot | df.boxplot(column='population', by='continent')  ### Outliers /// Min/max /// 25th, 50th, 75th percentiles |
|  | Scatter plot | ● Relationship between 2 numeric variables  ● Flag potentially bad data |

Chapter-2: Tidying data for analysis

|  |  |  |
| --- | --- | --- |
| Tidy data |  |  |
| Principles of tidy data |  | ● Columns represent separate variables  ● Rows represent individual observations  ● Observational units form tables |
| data problem: Columns containing values, instead of variables | pd.melt() | “id\_vars=” daki veriyi tutar, diger verileri bit sutunun icine eritir |
| Pivoting Data: un-melting data | | |
| Pivot |  | ● In melting, we turned columns into rows  ● Pivoting: turn unique values into separate columns |
| Pivot table |  | Has a parameter that specifies how to deal with duplicate  values |
| Beyond melt and pivot | | |
|  |  | Another common problem: Columns contain multiple bits of information ## bir sutunun birden fazla bilgi icermesi.  ## burada bitisik bilgilerin parse edilerek ayristirilmasi gerekiyor. |
| Melting and parsing | pd.melt() | pd.melt(frame=tb, id\_vars=['country', 'year']) |
|  | .str[] | tb\_melt['sex'] = tb\_melt.variable.str[0] |
|  | .str.split() | ebola\_melt['str\_split'] = ebola\_melt['type\_country'].str.split('\_') |
|  | .str.get() | ebola\_melt['type'] = ebola\_melt['str\_split'].str.get(0) |

Chapter-3: Combining data for analysis

|  |  |  |
| --- | --- | --- |
| Globbing |  |  |
|  | glob.glob() | import glob  csv\_files = glob.glob('\*.csv')  pd.concat(csv\_files) |
| Concatinating & Merging | | |

Chapter-4: Cleaning data for analysis

|  |  |  |
| --- | --- | --- |
| Converting data types | .astype() | df['treatment b'] = df['treatment b'].astype(str)  df['sex'] = df['sex'].astype('category') |
| Cleaning bad data | pd.to\_numeric() | df['treatment a'] = pd.to\_numeric(df['treatment a'], errors='coerce') |
| String manipulation | “re” | ● ‘re’ library for regular expressions  ● A formal way of specifying a pattern, Sequence of characters  ● Pattern matching, Similar to globbing |
| Using regular expressions | re.compile() & .match | In [1]: import re  In [2]: pattern = re.compile('\$\d\*\.\d{2}')  In [3]: result = pattern.match('$17.89')  In [4]: bool(result)  True |
| Find the numeric values | re.findall() | matches = re.findall('\d+', 'the recipe calls for 10 strawberries and 1 banana') |
| Using functions to clean data | | |
| Apply | .apply() | df.apply(np.mean, axis=0) |
|  | .replace() | import re  from numpy import NaN  pattern = re.compile('^\$\d\*\.\d{2}$')  def diff\_money(row, pattern):  icost = row['Initial Cost']  tef = row['Total Est. Fee']  if bool(pattern.match(icost)) and bool(pattern.match(tef)):  icost = icost.replace("$", "")  tef = tef.replace("$", "")  icost = float(icost)  tef = float(tef)  return icost - tef  else:  return(NaN)  df\_subset['diff'] = df\_subset.apply(diff\_money, axis=1, pattern=pattern) |
| Drop duplicates | .drop\_duplicates() | df = df.drop\_duplicates() ## tamami duplication ise siler |
| Drop missing values | .dropna() | tips\_dropped = tips\_nan.dropna() |
| Fill missing values | .fillna() | tips\_nan['sex'] = tips\_nan['sex'].fillna('missing')  tips\_nan[['total\_bill', 'size']] = tips\_nan[['total\_bill', 'size']].fillna(0) |
| Testing with asserts | assert | assert google.Close.notnull().all()  google\_0 = google.fillna(value=0)  assert google\_0.Close.notnull().all()  assert pd.notnull(ebola).all().all()  assert (ebola >= 0).all().all() |
|  | .apply(lambda) | ## tips['total\_dollar\_replace'] = tips.total\_dollar.apply(lambda x: x.replace('$', ''))  ## tips['total\_dollar\_re'] = tips.total\_dollar.apply(lambda x: re.findall('\d+\.\d+', x)[0]) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |