W3.summary

2020年3月26日 星期四 下午3:28

- Dataframe
  - creator: pd.Dataframe
  - value: [ ]
  - 3 rows: 3 dict { }
  - 2 cols: 2 items in 1 dict
    - key: col. name
    - value: col. value

1. Dataframe **merge**

- add new col: new col. identifier - df[new col. name]
  - value - a list of values
    - missing value + None
  - uniq index - series{dict: key - index : value}
    - align.
- join 2 large dataframe: pd.merge(df1, df2, method, left_index = True, right_index=True)
  - conflicts: location_x, location_y
  - left_on
  - right_on
  - left/right_on = [ ]

2. idiomatic pd -> Pandorable

- method chaining .rename(columns = {col.name: new col.name})
- apply: function - data - (a new row - pd.series{dict{key: func.name, value: func}})
  - row-col
  - add row - row[new col.name]
  - compare col value per row -> df.apply(func, axis=1) -> row
- apply + lambda.
  - df.apply(lambda x : function, axis=1)

3. **Groupby** - split up dataframe

- unique = for state in df['STNAME'].unique():
- groupby = for group, frame in df.groupby('STNAME')
  - np.average(df[df['STNAME']==state].dropna()['CENSUS2010POP'])
  - frame['CENSUS2010POP']
- timeit - speed different
- groupby{function} -> split
  - df.set_index()
  - df.groupby(func)
  - - groupby X original index - define func
  - define func
- groupby + func -> split_apply pattern
  - agg
  - df.grouping(index).agg({col.name = func})
- Apply
  - df.grouping(index).apply(lambda df: df.a . b : func, col.name)
- groupby object: single / list
  - df.setindex(index).groupby(level=0)[col.name].agg({name=func}).
  - a list of values

4. scale - data type

- col.data - categorical(nominal) . astype('category')
  - logical order - categories=[ ], ordered= T
- interval / ratio - categorical
  - pd.cut(col.value, bins)

5. pivot table: 2 data variables - 1 numeric variable - summary

df.pivot_table(value, index, columns, aggfunc)

- aggfunc -> single / list