assignment 8 (own)

March 8, 2022

1 Experimenting with confidence interval on 2 numerical columns by changing confidence interval and amount of data

[1]:
```
import pandas as pd
```

[2]:
```
import seaborn as sns
```

[3]:
```
df = pd.read_csv('steam.csv', sep=',')
```

[4]:
```
df.pop('appid')
# Convert english too boolean
df['english'] = df['english'].astype('bool')
# set release date to datetime
df['release_date'] = pd.to_datetime(df['release_date'])
# create 3 seperate platform fields instead of 1
df['windows'], df['mac'], df['linux'] = df['platforms'].apply(lambda x:␣
 ↪'windows' in x),df['platforms'].apply(lambda x: 'mac' in x),df['platforms'].
 ↪apply(lambda x: 'linux' in x)
# Split owners categorical value in two numerical values
df['owners_low'] = df['owners'].apply(lambda x: x.split('-')[0]).astype('int')
df['owners_high'] = df['owners'].apply(lambda x: x.split('-')[1]).astype('int')
# Create int out of data column
df['release_year'] = df['release_date'].dt.year
genres = df['genres'].apply(lambda x: x.split(';')[0])
```

[5]:
```
medianPlaytimeFilter = df['median_playtime']> 0.5
ownersFilter = df['owners_low'] > 20000 #lowest range above 0
reviewFilter = df['positive_ratings'] > 5
noFreeGameFilter = df['price'] > 0.1
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 27075 entries, 0 to 27074
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
```

```
 0   name              27075 non-null  object
 1   release_date      27075 non-null  datetime64[ns]
 2   english           27075 non-null  bool
 3   developer         27075 non-null  object
 4   publisher         27075 non-null  object
 5   required_age      27075 non-null  int64
 6   categories        27075 non-null  object
 7   genres            27075 non-null  object
 8   steamspy_tags     27075 non-null  object
 9   achievements      27075 non-null  int64
 10  positive_ratings  27075 non-null  int64
 11  negative_ratings  27075 non-null  int64
 12  average_playtime  27075 non-null  int64
 13  median_playtime   27075 non-null  int64
 14  price             27075 non-null  float64
 15  windows           27075 non-null  bool
 16  mac               27075 non-null  bool
 17  linux             27075 non-null  bool
 18  owners_low        27075 non-null  int32
 19  owners_high       27075 non-null  int32
 20  release_year      27075 non-null  int64
dtypes: bool(4), datetime64[ns](1), float64(1), int32(2), int64(7), object(6)
memory usage: 3.4+ MB
```

2 Price

[7]:

```
import scipy.stats as st
confidence = 0.90
st.t.interval(confidence, len(df)-1, loc=df[medianPlaytimeFilter]['price'].
 ↪mean(), scale=st.sem(df[medianPlaytimeFilter]['price']))
```

[7]:

```
(7.294043583940756, 7.650176837453336)
```

2.1 Half of the rows and 90% confidence interval

[8]:

```
import scipy.stats as st
confidence = 0.90
sample = df[medianPlaytimeFilter].sample(frac = 0.5)
st.t.interval(confidence, len(df)-1, loc=sample['price'].mean(), scale=st.
 ↪sem(sample['price']))
```

[8]:

```
(7.103136242539671, 7.609097144819007)
```

2.2 95% confidence interval

[9]:
```
import scipy.stats as st
confidence = 0.95
st.t.interval(confidence, len(df)-1, loc=df[medianPlaytimeFilter]['price'].
 ↪mean(), scale=st.sem(df[medianPlaytimeFilter]['price']))
```

[9]:
```
(7.259928512324668, 7.684291909069424)
```

2.3 99% confidence interval

[10]:
```
import scipy.stats as st
confidence = 0.99
st.t.interval(confidence, len(df)-1, loc=df[medianPlaytimeFilter]['price'].
 ↪mean(), scale=st.sem(df[medianPlaytimeFilter]['price']))
```

[10]:
```
(7.193248997268224, 7.7509714241258685)
```

2.4 99.99% confidence interval

[11]:
```
import scipy.stats as st
confidence = 0.9999
st.t.interval(confidence, len(df)-1, loc=df[medianPlaytimeFilter]['price'].
 ↪mean(), scale=st.sem(df[medianPlaytimeFilter]['price']))
```

[11]:
```
(7.050878743061757, 7.893341678332336)
```

3 Release year

3.1 95% confidence interval

[12]:
```
import scipy.stats as st
confidence = 0.95
st.t.interval(confidence, len(df)-1,␣
 ↪loc=df[medianPlaytimeFilter]['release_year'].mean(), scale=st.
 ↪sem(df[medianPlaytimeFilter]['release_year' ]))
```

[12]:
```
(2014.9515874846857, 2015.083420619042)
```

3.2 Half of the rows and 90% confidence interval

[13]:
```
import scipy.stats as st
confidence = 0.90
sample = df[medianPlaytimeFilter].sample(frac = 0.5)
st.t.interval(confidence, len(df)-1, loc=sample['release_year'].mean(),␣
 ↪scale=st.sem(sample['release_year']))
```

[13]:
```
(2014.9186830728577, 2015.0761305414048)
```

3.3 99% confidence interval

```
[14]: import scipy.stats as st
      confidence = 0.99
      st.t.interval(confidence, len(df)-1,␣
       ↪loc=df[medianPlaytimeFilter]['release_year'].mean(), scale=st.
       ↪sem(df[medianPlaytimeFilter]['release_year']))
```

```
[14]: (2014.9308727634475, 2015.1041353402802)
```

3.4 99.99% confidence interval

```
[15]: import scipy.stats as st
      confidence = 0.9999
      st.t.interval(confidence, len(df)-1,␣
       ↪loc=df[medianPlaytimeFilter]['release_year'].mean(), scale=st.
       ↪sem(df[medianPlaytimeFilter]['release_year']))
```

```
[15]: (2014.886643885568, 2015.1483642181597)
```