# Cleaning Data - due thu 09/30 | *by Charlotte*

## Exercise 1

For our data cleaning exercises, we will return one last time to our ACS data here. Download and import the 10percent ACS sample.

In [ ]:
```
import pandas as pd
acs = pd.read_stata('US_ACS_2017_10pct_sample.dta')
```

## Exercise 2

For our exercises today, we'll focus on age, gender, sex, and inctot. Subset your data to those variables, and quickly look at a sample of 10 rows.

In [ ]:
```
sub_acs = acs.loc[:, (acs.columns == 'age') | (acs.columns == 'sex') | (acs
sub_acs
import numpy.random as npr
npr.seed(42)
sub_acs.sample(10)
```

Out[ ]:

| | sex | age | educ | inctot |
|---|---|---|---|---|
| **166590** | male | 62 | 1 year of college | 170000 |
| **207895** | female | 6 | nursery school to grade 4 | 9999999 |
| **214500** | male | 18 | grade 12 | 0 |
| **28863** | female | less than 1 year old | n/a or no schooling | 9999999 |
| **18280** | female | 11 | grade 5, 6, 7, or 8 | 9999999 |
| **40280** | male | 56 | 4 years of college | 60000 |
| **131910** | male | 9 | nursery school to grade 4 | 9999999 |
| **207176** | male | 28 | grade 12 | 0 |
| **50852** | female | 10 | nursery school to grade 4 | 9999999 |
| **42735** | male | 65 | 2 years of college | 350200 |

## Exercise 3

First, replace all the values of inctot that are 9999999 with np.nan

In [ ]:

```
import numpy as np
sub_acs['inctot'].replace(9999999, np.nan)
sub_acs['inctot']
```

Out[ ]:

```
0         9999999
1            6000
2            6150
3           14000
4         9999999
           ...
318999      22130
319000    9999999
319001       5000
319002     240000
319003      48000
Name: inctot, Length: 319004, dtype: int32
```

## Exercise 4

Calculate the average age of people in our data. What do you get?

In [ ]:

```
f'Mean age is ${small_acs["age"].mean()}'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/h9/g02cmrsn6y571zb1v96gs56c0000gn/T/ipykernel_10859/1239298680.py in <module>
----> 1 f'Mean age is ${small_acs["age"].mean()}'

~/miniforge3/lib/python3.9/site-packages/pandas/core/generic.py in mean(self, axis, skipna, level, numeric_only, **kwargs)
  10749             )
  10750         def mean(self, axis=None, skipna=None, level=None, numeric_only=None, **kwargs):
> 10751             return NDFrame.mean(self, axis, skipna, level, numeric_only, **kwargs)
  10752 
  10753         setattr(cls, "mean", mean)

~/miniforge3/lib/python3.9/site-packages/pandas/core/generic.py in mean(self, axis, skipna, level, numeric_only, **kwargs)
  10367 
  10368     def mean(self, axis=None, skipna=None, level=None, numeric_only=None, **kwargs):
> 10369         return self._stat_function(
  10370             "mean", nanops.nanmean, axis, skipna, level, numeric_only, **kwargs
  10371         )

~/miniforge3/lib/python3.9/site-packages/pandas/core/generic.py in _stat_function(self, name, func, axis, skipna, level, numeric_only, **kwargs)
  10352                 name, axis=axis, level=level, skipna=skipna, numeric_only=numeric_only
  10353             )
> 10354         return self._reduce(
  10355             func, name=name, axis=axis, skipna=skipna, numeric_only=numeric_only
  10356         )

~/miniforge3/lib/python3.9/site-packages/pandas/core/series.py in _reduce(self, op, name, axis, skipna, numeric_only, filter_type, **kwds)
   4381         if isinstance(delegate, ExtensionArray):
   4382             # dispatch to ExtensionArray interface
-> 4383             return delegate._reduce(name, skipna=skipna, **kwds)
   4384 
   4385         else:

~/miniforge3/lib/python3.9/site-packages/pandas/core/arrays/_mixins.py in _reduce(self, name, skipna, **kwargs)
    258         else:
    259             msg = f"'{type(self).__name__}' does not implement reduction '{name}'"
--> 260             raise TypeError(msg)
    261 
    262     def _wrap_reduction_result(self, axis: int | None, result):

TypeError: 'Categorical' does not implement reduction 'mean'
```

*Problematic.*

## Exercise 5

We want to be able to calculate things using age, so we need it to be a numeric type. Check all the values of age to figure out why it's categorical and not numeric. You should find two problematic categories.

In [ ]:

```
age_list = sub_acs['age'].tolist()
print(age_list)
```

- The two problematic values I find are: "less than 1 year old" and "90 (90+ in 1980 and 1990)"
- Should be strings, i.e. not numeric

## Exercise 6

In order to convert age into a numeric variable, we need to replace those problematic entries with values that pandas can later convert into numbers. Pick appropriate substitutions for the existing values and replace the current values. Hint 1: Categorical variables act like strings, so you might want to use string methods! Hint 2: Remember that characters like parentheses, pluses, asterices, etc. are special in Python strings, and you have to escape them if you want them to be interpreted literally!

In [ ]:

```
sub_acs["age"] = sub_acs["age"].replace("90 (90+ in 1980 and 1990)" , "90")
sub_acs["age"] = sub_acs["age"].replace("less than 1 year old", "0").copy()
sub_acs = sub_acs.copy()
```

```
/var/folders/h9/g02cmrsn6y571zb1v96gs56c0000gn/T/ipykernel_10859/1377460654
.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  sub_acs["age"] = sub_acs["age"].replace("90 (90+ in 1980 and 1990)" , "90
").copy()
/var/folders/h9/g02cmrsn6y571zb1v96gs56c0000gn/T/ipykernel_10859/1377460654
.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  sub_acs["age"] = sub_acs["age"].replace("less than 1 year old", "0").copy
()
```

## Exercise 7

Now convert age from a categorical to numeric.

In [ ]:
```
sub_acs["age"] = pd.to_numeric(sub_acs["age"])
sub_acs.dtypes
```

Out[ ]:
```
sex        category
age           int64
educ       category
inctot        int32
dtype: object
```

## Exercise 8

Let's now filter out anyone in our data whose age is less than than 18. Note that before made age a numeric variable, we couldn't do this!

In [ ]:
```
print("Subsetting the dataset for the people who are above 18 years old:")
adult = sub_acs.loc[sub_acs["age"] > 18]
adult
```

```
Subsetting the dataset for the people who are above 18 years old:
```

Out[ ]:

| | sex | age | educ | inctot |
|---|---|---|---|---|
| **2** | male | 63 | 4 years of college | 6150 |
| **3** | female | 66 | grade 12 | 14000 |
| **5** | male | 50 | grade 12 | 50000 |
| **6** | male | 82 | 1 year of college | 27100 |
| **10** | male | 47 | n/a or no schooling | 18000 |
| **...** | ... | ... | ... | ... |
| **318998** | female | 31 | 5+ years of college | 0 |
| **318999** | female | 33 | 4 years of college | 22130 |
| **319001** | male | 20 | grade 12 | 5000 |
| **319002** | male | 47 | 5+ years of college | 240000 |
| **319003** | male | 33 | 5+ years of college | 48000 |

248538 rows × 4 columns

## Exercise 9

Create an indicator variable for whether each person has at least a college degree called college_degree

In [ ]:

```
print("Check values again for education variable")
adult['educ'].value_counts()
```

Out[ ]:

```
Check values again for education variable
grade 12                      89743
4 years of college            47212
1 year of college             38384
5+ years of college           29801
2 years of college            20731
grade 5, 6, 7, or 8            5942
grade 11                       4763
grade 10                       3942
n/a or no schooling            3627
grade 9                        3105
nursery school to grade 4      1288
Name: educ, dtype: int64
```

In [ ]:

```
print("Create Boolean indicator")
adult['college_degree'] = (adult['educ'] == '4 years of college') | (adult[
adult['college_degree']
adult
```

```
Create Boolean indicator
/var/folders/h9/g02cmrsn6y571zb1v96gs56c0000gn/T/ipykernel_10859/1651839132
.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs
/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  adult['college_degree'] = (adult['educ'] == '4 years of college') | (adul
t['educ'] =='5+ years of college')
```

Out[ ]:

| | sex | age | educ | inctot | college_degree |
|---|---|---|---|---|---|
| **2** | male | 63 | 4 years of college | 6150 | True |
| **3** | female | 66 | grade 12 | 14000 | False |
| **5** | male | 50 | grade 12 | 50000 | False |
| **6** | male | 82 | 1 year of college | 27100 | False |
| **10** | male | 47 | n/a or no schooling | 18000 | False |
| **...** | ... | ... | ... | ... | ... |
| **318998** | female | 31 | 5+ years of college | 0 | True |
| **318999** | female | 33 | 4 years of college | 22130 | True |
| **319001** | male | 20 | grade 12 | 5000 | False |
| **319002** | male | 47 | 5+ years of college | 240000 | True |
| **319003** | male | 33 | 5+ years of college | 48000 | True |

248538 rows × 5 columns

## Exercise 10

Let's examine how the educational gender gap. Use pd.crosstab to create a cross-tabulation of sex and college_degree. pd.crosstab will give you the number of people who have each combination of sex and college_degree (so in this case, it will give us a 2x2 table with Male and Female as rows, and college_degree True and False as columns, or vice versa.

In [ ]:

```
pd.crosstab(adult['sex'], adult['college_degree'])
```

Out[ ]:

| **college_degree** | **False** | **True** |
|---|---|---|
| **sex** | | |
| **male** | 83581 | 36181 |
| **female** | 87944 | 40832 |

## Exercise 11

Counts are kind of hard to interpret. pd.crosstab can also normalize values to give percentages. Look at the pd.crosstab help file to figure out how to normalize the values in the table. Normalize them so that you get the share of men with and without college degree, and the share of women with and without college degrees.

In [ ]:

```
pd.crosstab(adult['sex'], adult['college_degree'], normalize='columns')
```

Out[ ]:

| college_degree | False | True |
|---|---|---|
| sex | | |
| male | 0.487282 | 0.469804 |
| female | 0.512718 | 0.530196 |

## Exercise 12

Now, let's recreate that table for people over 40 and people under 40. Has the difference between men and women in terms of getting a college degree impoved, stayed the same, or worsened?

In [ ]:

```
below_acs = adult.loc[adult['age'] < 40]
pd.crosstab(below_acs['sex'], below_acs['college_degree'], normalize='colur
```

Out[ ]:

| college_degree | False | True |
|---|---|---|
| sex | | |
| male | 0.532658 | 0.437058 |
| female | 0.467342 | 0.562942 |

Comparing the overall sample to the subset of those below 40, on the first look the gap has overall stayed pretty much the same. However, marginally, the majority attribution for those without a degree have changed:

- We find that in the full sample, 51% (i.e. the majority) of those who did not have a degree were women, while in the subset the majority without a degree are men (53%). When looking at those with a degree the majority attribution has stayed the same but distribution has changed:
- In the full model, 53% of those with a degree were female, in the subset that number increases to 56%.

In [ ]:

```
above_acs = adult.loc[adult['age'] > 40]
pd.crosstab(above_acs['sex'], above_acs['college_degree'], normalize='colur
```

Out[ ]:

| college_degree | False | True |
|---|---|---|
| **sex** | | |
| **male** | 0.464684 | 0.487035 |
| **female** | 0.535316 | 0.512965 |

Comparing the overall sample to the subset of those above 40, on the first look the gap has overall stayed pretty much the same. THE majority attribution has stayed the same but distribution has changed:

- We find that in the full sample, 51% (i.e. the majority) of those who did not have a degree were women, while in the subset that number increases to 53%.
- In the full model, 53% of those with a degree were female, in the subset that number decreases to 51%.

## Conclusion

As we see the age gap is generally not very strong but it is interesting to see how the age gap behaves across generations. We find differences in the majority attribution for those above and below 40. On a first glance we can state that women are more likely than men to have a degree in the age group below 40. This would require further investigation.