ps_4

February 28, 2019

1 Problem Set 4 (python stuff)

Formatting enviornment:

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        from pathlib import Path
        import requests
        import statsmodels.api as sm
        import statsmodels.formula.api as smf
        from stargazer.stargazer import Stargazer
        from math import isclose
        from scipy import stats
        from IPython.display import display
```

Loading Data:

```
In [2]: def fetch_and_cache(data_url, file, data_dir="data", force = False):
            """
            (Credit: John DeNero)
            Download and cache a url and return the file object.

            Dependent: from pathlib import Path
                       import requests

            data_url: the web address to download
            file: the file in which to save the results.
            data_dir: (default="data") the location to save the data
            force: if true the file is always re-downloaded

            return: The pathlib.Path to the file.
            """
            data_dir = Path(data_dir)
            data_dir.mkdir(exist_ok=True)
            file_path = data_dir/Path(file)
```

```
            if force and file_path.exists():
                file_path.unlink()
            if force or not file_path.exists():
                print('Downloading...', end = ' ')
                resp = requests.get(data_url)
                with file_path.open('wb') as f:
                    f.write(resp.content)
                print('Done!')
            else:
                import time
                created = time.ctime(file_path.stat().st_ctime)
                print("Using cached version downloaded at", created)
            return file_path

        data_url = 'https://bcourses.berkeley.edu/files/74496429/download?download_frd=1'
        file = 'ovb.csv'
        ovb_path = fetch_and_cache(data_url, file)

        ovb_df = pd.read_csv(ovb_path)
```

```
Using cached version downloaded at Wed Feb 27 12:57:56 2019
```

1.1 Question 2:

```
In [3]: models = ["logwage ~ 1 + educ", "educ ~ 1 + logwage"]

        f_df = ovb_df.loc[ovb_df['female'] == 1]

        rho = np.corrcoef(f_df['logwage'], f_df['educ'])[0,1]
        rho_squared = rho**2

        r_squared = []
        for mod in models:
            res = smf.ols(mod, data = f_df).fit()
            r_squared.append(res.rsquared)
        assert(isclose(r_squared[0], r_squared[1], abs_tol = 1e-6))
        assert(isclose(r_squared[0], rho_squared, abs_tol = 1e-6))

        print("correlation coefficient = {}".format(rho_squared))
        print("r^2 of logwage on educ = {}".format(r_squared[0]))
        print("r^2 of educ on logwage = {}".format(r_squared[1]))
```

```
correlation coefficient = 0.22388700299289657
r^2 of logwage on educ = 0.22388700299289477
r^2 of educ on logwage = 0.2238870029928961
```

1.2 Question 3:

1.2.1 a)

```
In [4]: q3_a = pd.DataFrame(
            [
                list((np.mean(s), np.std(s), len(s))) for s in
                    [f_df.loc[f_df['imm'] == i]['logwage'] for i in [0,1]]
                ]
            , columns = ['mean', 'std', "n"]
            )
        mean_diff = (q3_a['mean'][1] - q3_a['mean'][0])
        se = np.sqrt(sum(q3_a['std']**2 / q3_a['n']))
        t_test = mean_diff/se

        q3_a.rename({0: "non-immigrants", 1: "immigrants"} ,axis = 'index',
                    inplace = True)

        display(q3_a)
        print("the t-score for the difference in means test is {}".format(t_test))
        print("the difference in means is: {} \n with se {}".format(mean_diff,se))
```

	mean	std	n
non-immigrants	2.886378	0.651487	8616
immigrants	2.706393	0.715594	1985

```
the t-score for the difference in means test is -10.268432855577307
the difference in means is: -0.17998575423169516
 with se 0.017528064580364447
```

The t-test implies the means are different.

1.2.2 b)

```
In [5]: models = "logwage ~ 1 + imm"

        res = smf.ols(models, data = f_df).fit()
        beta = res.params
        assert(isclose(mean_diff, beta['imm'], abs_tol = 1e-6))
        print("the OLS estimate for logwage on imm is: {} \n is equal to difference in means:

        sigma = res.bse['imm']
        new_t = res.tvalues['imm']
        print("however, the new std error is {} \n which leads to t-score of: {}".format(sigma
```

```
the OLS estimate for logwage on imm is: -0.1799857542317315
 is equal to difference in means: -0.17998575423169516
```

```
however, the new std error is 0.016531953532059624
 which leads to t-score of: -10.887143729426397
```

This difference is because the original t-test was used heteroskedastic std error, but the regression model used a homoskedastic estimate.

1.2.3 c)

```
In [6]: #refitting data with White's heteroskedastic robust std. error
        het_res = smf.ols(models, data = f_df).fit(cov_type = 'HC0')

        het_beta = het_res.params
        assert(isclose(mean_diff, het_beta['imm'], abs_tol = 1e-6))
        print("the OLS estimate for logwage on imm is: {} \n is equal to difference in means: 

        het_sigma = het_res.bse['imm']
        het_new_t = het_res.tvalues['imm']
        print("the heteroskedastic robust std error is {} \n which leads to t-score of: {}".fo
```

```
the OLS estimate for logwage on imm is: -0.1799857542317315
 is equal to difference in means: -0.17998575423169516

the heteroskedastic robust std error is 0.01752806458036458
 which leads to t-score of: -10.268432855579304
```

The heteroskedastic robust std. error is equal to the original std. error which gives rise to the same t-score. The heteroskedastic robust std. error is slightly larger than the homoskedastic std. error.

```
In [ ]:
```