Assignment #2

MACS 30000, Dr. Evans

Sanittawan Tan (sanittawan)

In [82]:
```
# import packages
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
# plt.style.use('seaboarn')
```

1. Imputing age and and gender

I begin with data preparation before answering questions

In [83]:
```
# import BestIncome.txt and add headers

headers = ["lab_inc", "cap_inc", "hgt", "wgt"]

best_income = pd.read_table("BestIncome.txt", sep=",", names = headers,
index_col=False)

best_income.head()
```

Out[83]:

	lab_inc	cap_inc	hgt	wgt
0	52655.605507	9279.509829	64.568138	152.920634
1	70586.979225	9451.016902	65.727648	159.534414
2	53738.008339	8078.132315	66.268796	152.502405
3	55128.180903	12692.670403	62.910559	149.218189
4	44482.794867	9812.975746	68.678295	152.726358

In [84]:
```
best_income.tail()
```

Out[84]:

	lab_inc	cap_inc	hgt	wgt
9995	51502.225233	14786.050723	66.781187	154.645212
9996	52624.117104	11048.811747	64.499036	165.868002
9997	50725.310645	13195.218100	64.508873	154.657639
9998	56392.824076	8470.592718	62.161556	145.498194
9999	44274.098164	12765.748454	64.974145	135.936862

In [85]:
```
# import SurvIncome.txt and add headers

headers_survey = ["tot_inc", "wgt", "age", "female"]

survey_income = pd.read_table("SurvIncome.txt", sep=",", names = headers
_survey, index_col=False)

survey_income.head()
```

Out[85]:

	tot_inc	wgt	age	female
0	63642.513655	134.998269	46.610021	1.0
1	49177.380692	134.392957	48.791349	1.0
2	67833.339128	126.482992	48.429894	1.0
3	62962.266217	128.038121	41.543926	1.0
4	58716.952597	126.211980	41.201245	1.0

In [86]:
```
survey_income.tail()
```

Out[86]:

	tot_inc	wgt	age	female
995	61270.538697	184.930002	46.356881	0.0
996	59039.159876	180.482304	50.986966	0.0
997	67967.188804	156.816883	40.965268	0.0
998	79726.914251	158.935050	41.190371	0.0
999	71005.223603	169.067695	48.480007	0.0

(a) Imputing age (age_i) and gender ($female_i$) variables

In order to impute age (age_i) variable into the `BestIncome.txt` data by using information from `SurvIncome.txt`, I propose that we build linear regression model.

1. conduct a regression analysis by predicting age from total income and weight from data in `SurvIncome.txt`. We will obtain coefficients of total income and weight which can be used in the next step. The proposed equation is as follows:

$$\widehat{\text{age}_i} = \hat{\beta}_0 + \hat{\beta}_1 \text{tot\_inc}_i + \hat{\beta}_2 \text{weight}_i$$

1. we then turn to `BestIncome.txt` and create a new variable called total income (tot_inc_i) which is the sum of capital income and labor income.

$$\text{tot\_inc}_i = \text{lab\_inc}_i + \text{cap\_inc}_i$$

1. After total income (tot_inc_i) was created, we will use coefficients obtained from part 1 to impute age (age_i) for each observation by using data on total income and weight in the `BestIncome.txt` dataset. The equation is similar to the one we have in (1):

$$\widehat{\text{age}_i} = \hat{\beta}_0 + \hat{\beta}_1 \text{tot\_inc}_i + \hat{\beta}_2 \text{weight}_i$$

In order to impute gender ($female_i$) variable into the `BestIncome.txt` data by using information from `SurvIncome.txt`, I propose that we build a logistic regression model since this becomes a categorization problem.

1. conduct a logistic regression by predicting gender ($female_i$) from total income and weight from data in `SurvIncome.txt`. We will obtain coefficients of total income and weight which can be used in the next step. The equation is as follows:

$$p(\text{gender} = \text{female}) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 \text{tot\_inc}_i + \hat{\beta}_2 \text{weight}_i)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 \text{tot\_inc}_i + \hat{\beta}_2 \text{weight}_i)}$$

1. As total income in `BestIncome.txt` has been created, we can conduct logistic regression analysis to predict gender from total income and weight data in `BestIncome.txt` and obtain its coefficients for use in the next part.

1. We will use coefficients from (2) to calculate the probability of being female for each observation in `BestIncome.txt` using similar equation from (1):

$$p(\text{gender} = \text{female}) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 \text{tot\_inc}_i + \hat{\beta}_2 \text{weight}_i)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 \text{tot\_inc}_i + \hat{\beta}_2 \text{weight}_i)}$$

We establish a threshold of $p > 0.5$, meaning that if the $p(\text{gender} = \text{female}) > 0.5$, that observation will be categorized as female.

(b) Imputing variables using the proposed methods

Imputing age (age_i)

In [87]:
```
# Define dependent and independent variables
age = 'age'
features = ['tot_inc', 'wgt']
X, y = survey_income[features], survey_income[age]
```

In [88]:
```
X.head()
```

Out[88]:

	tot_inc	wgt
0	63642.513655	134.998269
1	49177.380692	134.392957
2	67833.339128	126.482992
3	62962.266217	128.038121
4	58716.952597	126.211980

In [89]:
```
X.shape
```

Out[89]:
```
(1000, 2)
```

In [90]:
```
y.head()
```

Out[90]:
```
0    46.610021
1    48.791349
2    48.429894
3    41.543926
4    41.201245
Name: age, dtype: float64
```

In [91]:
```
# Fit and summarize model

X = sm.add_constant(X, prepend=False)
regressed_age = sm.OLS(y, X)
res_age = regressed_age.fit()
print(res_age.summary())
```

OLS Regression Results

Dep. Variable:	age	R-squared:	0.001
Model:	OLS	Adj. R-squared:	-0.001
Method:	Least Squares	F-statistic:	0.6326
Date:	Wed, 17 Oct 2018	Prob (F-statistic):	0.531
Time:	08:35:04	Log-Likelihood:	-3199.4
No. Observations:	1000	AIC:	6405.
Df Residuals:	997	BIC:	6419.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
tot_inc	2.52e-05	2.26e-05	1.114	0.266	-1.92e-05	6.96e-05
wgt	-0.0067	0.010	-0.686	0.493	-0.026	0.013
const	44.2097	1.490	29.666	0.000	41.285	47.134

Omnibus:	2.460	Durbin-Watson:	1.921
Prob(Omnibus):	0.292	Jarque-Bera (JB):	2.322
Skew:	-0.109	Prob(JB):	0.313
Kurtosis:	3.092	Cond. No.	5.20e+05

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.2e+05. This might indicate that there are
strong multicollinearity or other numerical problems.

In [92]:
```
ols_survey_income = pd.concat([y, X], axis=1)
ols_survey_income.head()
```

Out[92]:

	age	tot_inc	wgt	const
0	46.610021	63642.513655	134.998269	1.0
1	48.791349	49177.380692	134.392957	1.0
2	48.429894	67833.339128	126.482992	1.0
3	41.543926	62962.266217	128.038121	1.0
4	41.201245	58716.952597	126.211980	1.0

In [93]:
```
ols_survey_income['predicted_age'] = res_age.predict(X)
ols_survey_income.head()
```

Out[93]:

	age	tot_inc	wgt	const	predicted_age
0	46.610021	63642.513655	134.998269	1.0	44.906121
1	48.791349	49177.380692	134.392957	1.0	44.545636
2	48.429894	67833.339128	126.482992	1.0	45.068980
3	41.543926	62962.266217	128.038121	1.0	44.935764
4	41.201245	58716.952597	126.211980	1.0	44.841048

In [94]:
```
ols_survey_income.tail()
```

Out[94]:

	age	tot_inc	wgt	const	predicted_age
995	46.356881	61270.538697	184.930002	1.0	44.510693
996	50.986966	59039.159876	180.482304	1.0	44.484356
997	40.965268	67967.188804	156.816883	1.0	44.868444
998	41.190371	79726.914251	158.935050	1.0	45.150577
999	48.480007	71005.223603	169.067695	1.0	44.862658

In order to impute age variable in `BestIncome.txt` , we plug in the coefficients, obtain the following formular and wrote a Python function to calculate age for each observation using data from `BestIncome.txt`

$$\widehat{\text{age}_i} = 44.2097 + 0.0000252\text{tot\_inc}_i - 0.0067\text{weight}_i$$

However, we have to create a new column called "total income" for `BestIncome.txt` by summing labor income and capital income:

$$\text{tot\_inc}_i = \text{lab\_inc}_i + \text{cap\_inc}_i$$

In [95]:
```
# create a new column called tot_inc in best_income dataframe

best_income['tot_inc'] = best_income['lab_inc'] + best_income['cap_inc']
best_income.head()
```

Out[95]:

	lab_inc	cap_inc	hgt	wgt	tot_inc
0	52655.605507	9279.509829	64.568138	152.920634	61935.115336
1	70586.979225	9451.016902	65.727648	159.534414	80037.996127
2	53738.008339	8078.132315	66.268796	152.502405	61816.140654
3	55128.180903	12692.670403	62.910559	149.218189	67820.851305
4	44482.794867	9812.975746	68.678295	152.726358	54295.770612

In [96]:
```
best_income.shape
```

Out[96]:
```
(10000, 5)
```

In [97]:
```
# create a function to impute age

def get_age(row):

    tot_inc = row[0]
    weight = row[1]

    age = 44.2097 + (0.0000252 * tot_inc) - (0.0067 * weight)

    return age
```

In [98]:
```
# impute age and add it as a column to best_income dataframe

best_income['imputed_age'] = best_income[['tot_inc', 'wgt']].apply(get_a
ge, axis=1)
best_income.head()
```

Out[98]:

	lab_inc	cap_inc	hgt	wgt	tot_inc	imputed_age
0	52655.605507	9279.509829	64.568138	152.920634	61935.115336	44.745897
1	70586.979225	9451.016902	65.727648	159.534414	80037.996127	45.157777
2	53738.008339	8078.132315	66.268796	152.502405	61816.140654	44.745701
3	55128.180903	12692.670403	62.910559	149.218189	67820.851305	44.919024
4	44482.794867	9812.975746	68.678295	152.726358	54295.770612	44.554687

In [99]:
```
best_income.tail()
```

Out[99]:

	lab_inc	cap_inc	hgt	wgt	tot_inc	imputed_age
9995	51502.225233	14786.050723	66.781187	154.645212	66288.275956	44.844042
9996	52624.117104	11048.811747	64.499036	165.868002	63672.928851	44.702942
9997	50725.310645	13195.218100	64.508873	154.657639	63920.528745	44.784291
9998	56392.824076	8470.592718	62.161556	145.498194	64863.416794	44.869420
9999	44274.098164	12765.748454	64.974145	135.936862	57039.846618	44.736327

Imputing gender ($female_i$)

In [100]:
```
# Define dependent and independent variables
y1 = 'female'
x1 = ['tot_inc', 'wgt']
x1, y1 = survey_income[x1], survey_income[y1]
```

In [101]:
```
x1.head()
```

Out[101]:

	tot_inc	wgt
0	63642.513655	134.998269
1	49177.380692	134.392957
2	67833.339128	126.482992
3	62962.266217	128.038121
4	58716.952597	126.211980

In [102]:
```
x1.shape
```

Out[102]:
```
(1000, 2)
```

In [103]:
```
y1.head()
```

Out[103]:
```
0    1.0
1    1.0
2    1.0
3    1.0
4    1.0
Name: female, dtype: float64
```

In [104]:
```
# Fit and summarize model

x1 = sm.add_constant(x1, prepend=False)
regressed_gender = sm.Logit(y1, x1)
res_gender = regressed_gender.fit()
print(res_gender.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.036050
         Iterations 11
                           Logit Regression Results

==============================================================================
=======
Dep. Variable:                  female   No. Observations:
   1000
Model:                           Logit   Df Residuals:
    997
Method:                            MLE   Df Model:
      2
Date:                Wed, 17 Oct 2018   Pseudo R-squ.:
0.9480
Time:                        08:35:04   Log-Likelihood:
-36.050
converged:                       True   LL-Null:
-693.15
                                        LLR p-value:                4.2
32e-286
==============================================================================
=======
                 coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
-------
tot_inc       -0.0002   4.25e-05     -3.660      0.000      -0.000   -
7.22e-05
wgt           -0.4460      0.062     -7.219      0.000      -0.567
-0.325
const         76.7929     10.569      7.266      0.000      56.078
97.508
==============================================================================
=======

Possibly complete quasi-separation: A fraction 0.55 of observations can
be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

In [105]:
```
logit_survey_income = pd.concat([y1, x1], axis=1)
logit_survey_income.head()
```

Out[105]:

	female	tot_inc	wgt	const
0	1.0	63642.513655	134.998269	1.0
1	1.0	49177.380692	134.392957	1.0
2	1.0	67833.339128	126.482992	1.0
3	1.0	62962.266217	128.038121	1.0
4	1.0	58716.952597	126.211980	1.0

In [106]:
```
# Add predicted gender
logit_survey_income['predicted_gender'] = res_gender.predict(x1)
logit_survey_income.head()
```

Out[106]:

	female	tot_inc	wgt	const	predicted_gender
0	1.0	63642.513655	134.998269	1.0	0.998746
1	1.0	49177.380692	134.392957	1.0	0.999899
2	1.0	67833.339128	126.482992	1.0	0.999946
3	1.0	62962.266217	128.038121	1.0	0.999949
4	1.0	58716.952597	126.211980	1.0	0.999988

In [107]:
```
logit_survey_income.tail()
```

Out[107]:

	female	tot_inc	wgt	const	predicted_gender
995	0.0	61270.538697	184.930002	1.0	2.446782e-07
996	0.0	59039.159876	180.482304	1.0	2.517052e-06
997	0.0	67967.188804	156.816883	1.0	2.354661e-02
998	0.0	79726.914251	158.935050	1.0	1.503330e-03
999	0.0	71005.223603	169.067695	1.0	6.366193e-05

In order to impute gender variable in `BestIncome.txt` , we plug in the coefficients, obtain the following formular and wrote a Python function to calculate probability of being female for each observation using data from `BestIncome.txt`

$$p(\text{gender} = \text{female}) = \frac{\exp(76.7929 - 0.0002\text{tot\_inc}_i - 0.446\text{weight}_i)}{1+\exp(76.7929 - 0.0002\text{tot\_inc}_i - 0.446\text{weight}_i)}$$

If probability is more than 0.5, an observation associated with that probability is categorized as female

In [108]:
```
best_income['constant'] = 1
```

In [109]:
```
best_income.head()
```

Out[109]:

	lab_inc	cap_inc	hgt	wgt	tot_inc	imputed_age	constant
0	52655.605507	9279.509829	64.568138	152.920634	61935.115336	44.745897	1
1	70586.979225	9451.016902	65.727648	159.534414	80037.996127	45.157777	1
2	53738.008339	8078.132315	66.268796	152.502405	61816.140654	44.745701	1
3	55128.180903	12692.670403	62.910559	149.218189	67820.851305	44.919024	1
4	44482.794867	9812.975746	68.678295	152.726358	54295.770612	44.554687	1

In [110]:
```
best_income['prob_female'] = res_gender.predict(best_income[['tot_inc',
'wgt', 'constant']])
```

In [111]:
```
best_income.head()
```

Out[111]:

	lab_inc	cap_inc	hgt	wgt	tot_inc	imputed_age	constant	pro
0	52655.605507	9279.509829	64.568138	152.920634	61935.115336	44.745897	1	
1	70586.979225	9451.016902	65.727648	159.534414	80037.996127	45.157777	1	
2	53738.008339	8078.132315	66.268796	152.502405	61816.140654	44.745701	1	
3	55128.180903	12692.670403	62.910559	149.218189	67820.851305	44.919024	1	
4	44482.794867	9812.975746	68.678295	152.726358	54295.770612	44.554687	1	

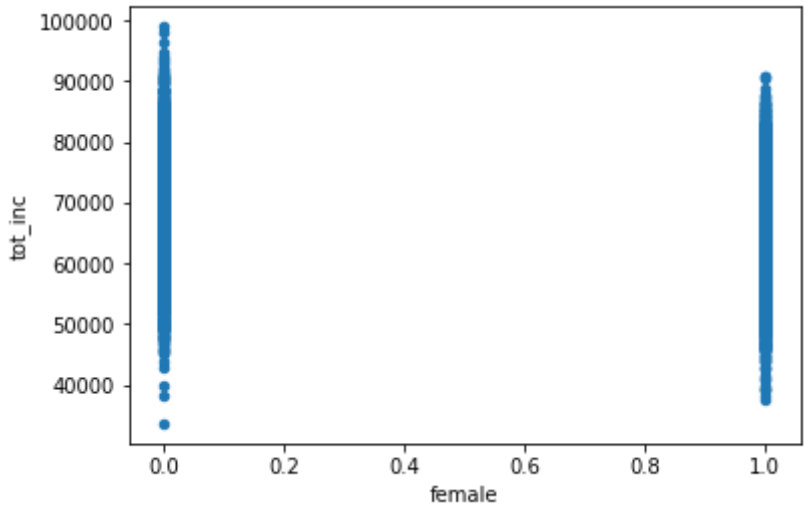
In [112]:
```
# replace prob female with 0 or 1 if p >= 0.5
best_income.loc[best_income.prob_female >= 0.5,'female'] = 1
best_income.loc[best_income.prob_female < 0.5,'female'] = 0
```

In [113]:
```
best_income.head()
```

Out[113]:

	lab_inc	cap_inc	hgt	wgt	tot_inc	imputed_age	constant	pro
0	52655.605507	9279.509829	64.568138	152.920634	61935.115336	44.745897	1	
1	70586.979225	9451.016902	65.727648	159.534414	80037.996127	45.157777	1	
2	53738.008339	8078.132315	66.268796	152.502405	61816.140654	44.745701	1	
3	55128.180903	12692.670403	62.910559	149.218189	67820.851305	44.919024	1	
4	44482.794867	9812.975746	68.678295	152.726358	54295.770612	44.554687	1	

In [114]:
```
# Make a scatter plot to check probability calculation
female = best_income['female']
tot_inc = best_income['tot_inc']
best_income.plot(x='female', y='tot_inc', kind='scatter')
plt.show()
```



(c) Report mean, standard deviation, minimum, maximum and number of observations of imputed age and gender

In [115]:
```
best_income['imputed_age'].describe()
```

Out[115]:
```
count    10000.000000
mean        44.894036
std          0.219066
min         43.980016
25%         44.747065
50%         44.890281
75%         45.042239
max         45.706849
Name: imputed_age, dtype: float64
```

In [116]:
```
best_income['female'].describe()
```

Out[116]:
```
count    10000.000000
mean         0.454600
std          0.497959
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          1.000000
Name: female, dtype: float64
```

As for imputed age,

- mean = 44.89 years old
- standard deviation = 0.2191
- minimum = 43.98 years old
- maximum = 45.71 years old
- number of observations = 10,000

As for imputed gender (female),

- mean = 0.4546 (~45% are female)
- standard deviation = 0.498
- minimum = 0
- maximum = 1
- number of observations = 10,0000

(d) report the correlation matrix for six variables

In [117]:
```
best_income.head()
```

Out[117]:

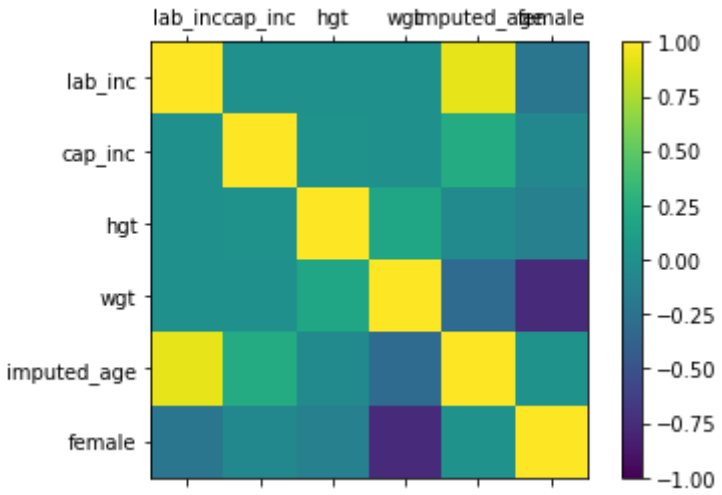
	lab_inc	cap_inc	hgt	wgt	tot_inc	imputed_age	constant	pro
0	52655.605507	9279.509829	64.568138	152.920634	61935.115336	44.745897	1	
1	70586.979225	9451.016902	65.727648	159.534414	80037.996127	45.157777	1	
2	53738.008339	8078.132315	66.268796	152.502405	61816.140654	44.745701	1	
3	55128.180903	12692.670403	62.910559	149.218189	67820.851305	44.919024	1	
4	44482.794867	9812.975746	68.678295	152.726358	54295.770612	44.554687	1	

In [118]:
```
# Eliminate tot_inc and predicted_fem from the dataframe

cor_best_income = best_income.drop(columns=['tot_inc', 'prob_female', 'c
onstant'])
cor_best_income.head()
```

Out[118]:

	lab_inc	cap_inc	hgt	wgt	imputed_age	female
0	52655.605507	9279.509829	64.568138	152.920634	44.745897	0.0
1	70586.979225	9451.016902	65.727648	159.534414	45.157777	0.0
2	53738.008339	8078.132315	66.268796	152.502405	44.745701	0.0
3	55128.180903	12692.670403	62.910559	149.218189	44.919024	0.0
4	44482.794867	9812.975746	68.678295	152.726358	44.554687	1.0

In [119]:
```
def corr_plot(df):
    import matplotlib.pyplot as plt
    import numpy as np
    import pandas as pd

    names = df.columns
    N = len(names)

    correlations = df.corr()
    fig = plt.figure()
    ax = fig.add_subplot(111)
    cax = ax.matshow(correlations, vmin=-1, vmax=1)
    fig.colorbar(cax)
    ticks = np.arange(0,N,1)
    ax.set_xticks(ticks)
    ax.set_yticks(ticks)
    ax.set_xticklabels(names)
    ax.set_yticklabels(names)
    plt.show()

corr_plot(cor_best_income)
```



In [120]:
```
corr = cor_best_income.corr()
corr.style.background_gradient()
```

Out[120]:

	lab_inc	cap_inc	hgt	wgt	imputed_age	female
lab_inc	1	0.0053253	0.00278978	0.00450691	0.924329	-0.215469
cap_inc	0.0053253	1	0.0215716	0.00629868	0.234234	-0.0625691
hgt	0.00278978	0.0215716	1	0.172103	-0.0449272	-0.127416
wgt	0.00450691	0.00629868	0.172103	1	-0.299395	-0.763821
imputed_age	0.924329	0.234234	-0.0449272	-0.299395	1	0.0193158
female	-0.215469	-0.0625691	-0.127416	-0.763821	0.0193158	1

2. Stationarity and data drift

In [121]:
```
# Data import and preparation

headers = ["grad_year", "gre_qnt", "salary_p4"]
income_intel = pd.read_table("IncomeIntel.txt", sep=",", names = headers
, index_col=False)
income_intel
```

Out[121]:

	grad_year	gre_qnt	salary_p4
0	2001.0	739.737072	67400.475185
1	2001.0	721.811673	67600.584142
2	2001.0	736.277908	58704.880589
3	2001.0	770.498485	64707.290345
4	2001.0	735.002861	51737.324165
5	2001.0	763.876037	64010.822579
6	2001.0	738.758659	60080.107481
7	2001.0	706.407471	56263.309815
8	2001.0	705.886037	62109.859243
9	2001.0	700.971986	50189.704747
10	2001.0	709.754522	58721.753127
11	2001.0	734.854582	65380.594586
12	2001.0	753.384151	52857.212365
13	2001.0	690.312090	63572.217765
14	2001.0	774.154371	65892.177035
15	2001.0	726.377225	67454.545201
16	2001.0	702.735945	59346.670232
17	2001.0	723.806542	70031.012603
18	2001.0	758.051159	53441.672888
19	2001.0	711.063082	61008.652046
20	2001.0	702.975969	50065.932451
21	2001.0	733.877837	75612.225369
22	2001.0	735.918767	59580.620375
23	2001.0	749.069115	57825.611782
24	2001.0	732.581793	52809.225854
25	2001.0	728.050446	57492.084316
26	2001.0	690.265988	64686.224351
27	2001.0	732.448836	53067.021394
28	2001.0	724.755887	58902.707320
29	2001.0	721.739038	62094.061567
...	...	...	...
970	2013.0	158.578197	79263.470892
971	2013.0	147.667305	104782.627567
972	2013.0	160.086274	94013.946074

	grad_year	gre_qnt	salary_p4
973	2013.0	156.289493	74032.543183
974	2013.0	150.340044	84220.290724
975	2013.0	163.054596	74940.546965
976	2013.0	157.624151	83293.343135
977	2013.0	150.927266	78340.908128
978	2013.0	157.393763	91066.889575
979	2013.0	154.449630	87169.012509
980	2013.0	153.756644	90033.601423
981	2013.0	150.796371	98650.768576
982	2013.0	150.691700	70455.885421
983	2013.0	153.639896	91133.301177
984	2013.0	150.374470	91796.617819
985	2013.0	162.350725	73780.832249
986	2013.0	155.803279	96927.925237
987	2013.0	159.111662	71875.246552
988	2013.0	158.338350	103357.966587
989	2013.0	162.308518	73780.472319
990	2013.0	156.651125	79055.571295
991	2013.0	153.836045	91529.313046
992	2013.0	149.542467	75940.200168
993	2013.0	155.349020	97688.397380
994	2013.0	161.767399	75260.194609
995	2013.0	160.441025	100430.166532
996	2013.0	160.431891	82198.200872
997	2013.0	154.254526	84340.214218
998	2013.0	162.036321	87600.881985
999	2013.0	156.946735	82854.576903

1000 rows × 3 columns

(a) Estimate coefficients in the regression of $\text{salary\_p4}_i = \beta_0 + \beta_1 \text{gre\_qnt}_i + \varepsilon_i$

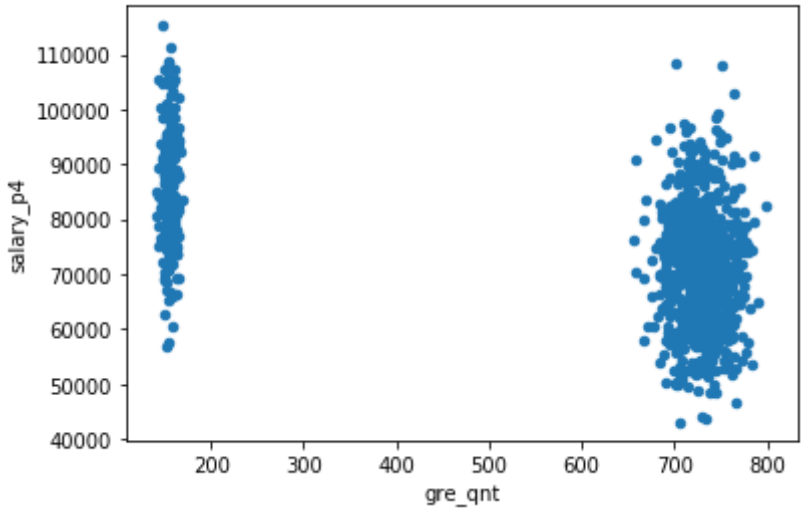
In [122]:
```
# Define dependent and independent variables

X_quant, y_salary = income_intel['gre_qnt'], income_intel['salary_p4']
```

In [123]:
```
# Fit and summarize model

X_quant = sm.add_constant(X_quant, prepend=False)
regressed_salary = sm.OLS(y_salary, X_quant)
res_salary = regressed_salary.fit()
print(res_salary.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:              salary_p4   R-squared:
  0.263
Model:                            OLS   Adj. R-squared:
  0.262
Method:                 Least Squares   F-statistic:
  356.3
Date:                Wed, 17 Oct 2018   Prob (F-statistic):
3.43e-68
Time:                        08:35:05   Log-Likelihood:
-10673.
No. Observations:                1000   AIC:                             2.
135e+04
Df Residuals:                     998   BIC:                             2.
136e+04
Df Model:                           1

Covariance Type:            nonrobust

==============================================================================
=======
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-------
gre_qnt      -25.7632      1.365    -18.875      0.000     -28.442
-23.085
const       8.954e+04    878.764    101.895      0.000    8.78e+04
9.13e+04
==============================================================================
=======
Omnibus:                        9.118   Durbin-Watson:
  1.424
Prob(Omnibus):                  0.010   Jarque-Bera (JB):
  9.100
Skew:                           0.230   Prob(JB):
0.0106
Kurtosis:                       3.077   Cond. No.
1.71e+03
==============================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.71e+03. This might indicate that t
here are
strong multicollinearity or other numerical problems.
```

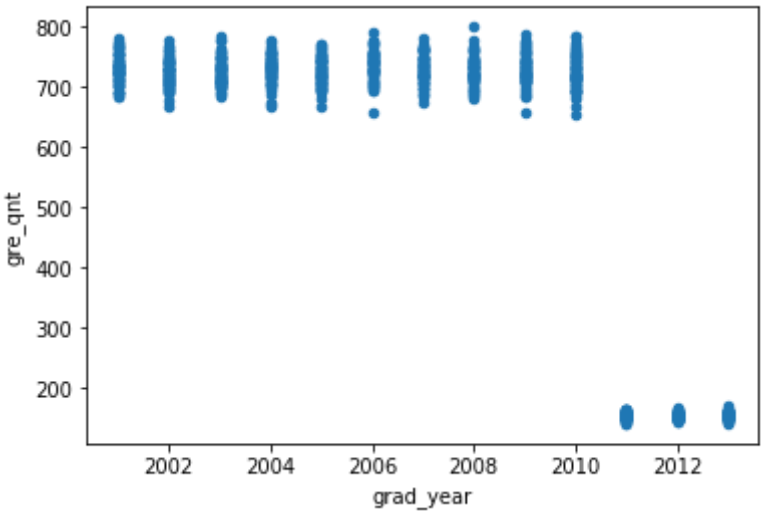
- β_0 is equal to 89540 with standard error of 878.764
- β_1 is equal to -25.7632 with standard error of 1.365 (noted that p-value is less than 0.05)

In [124]:
```
# Make a diagnostic scatter plot
salary_p4 = income_intel['salary_p4']
gre_quant = income_intel['gre_qnt']
income_intel.plot(x='gre_qnt', y='salary_p4', kind='scatter')
plt.show()
```



(b) Create a scatter plot of GRE quantitative score and graduation year

In [125]:
```
# Make a scatter plot
grad_year = income_intel['grad_year']
gre_quant = income_intel['gre_qnt']
income_intel.plot(x='grad_year', y='gre_qnt', kind='scatter')
plt.show()
```



The scatter plot shows that the GRE quantitative scores before and after 2011 are not on the same scale. This is problematic for our regression model because the GRE changed the scoring scale in 2011. The old scoring scale ranges from 200 to 800 while the new scoring scale ranges from 130 to 170.

I propose that we convert the old GRE quant scores to the new scale by:

1. rounding up both old and new GRE scores in the data set because old GRE score is a multiple of ten ranging from 200 to 800 and new GRE is an integer between 130 and 170
2. convert old GRE scores to new GRE scores using [GRE Concordance table (https://www.ets.org/s/gre/pdf/concordance_information.pdf)]

It is noted that the old GRE score is a multiple of ten, but our data set include scores which are not multiples of ten. I resolved this problem by creating a range of score conversion. For example, a score of 719 of our first observation falls between 710 (converted to 155) and 720 (converted to 156). Since 719 is more than 710 but less than 720, I convert it to 156. Thus, I treated lower bound as a cutoff point rather than looking at the distance of the score from the lower and upperbound.

In [126]:
```
income_intel = income_intel.round({'gre_qnt': 0})
```

In [127]:
```
income_intel = income_intel.astype(dtype='int64')
```

In [128]:
```
income_intel.head()
```

Out[128]:

	grad_year	gre_qnt	salary_p4
0	2001	740	67400
1	2001	722	67600
2	2001	736	58704
3	2001	770	64707
4	2001	735	51737

In [129]:
```
income_intel.tail()
```

Out[129]:

	grad_year	gre_qnt	salary_p4
995	2013	160	100430
996	2013	160	82198
997	2013	154	84340
998	2013	162	87600
999	2013	157	82854

In [130]:
```
income_intel2 = income_intel.copy()
```

In [131]:
```
income_intel2.head()
```

Out[131]:

	grad_year	gre_qnt	salary_p4
0	2001	740	67400
1	2001	722	67600
2	2001	736	58704
3	2001	770	64707
4	2001	735	51737

In [132]:
```
gre_qnt_array = income_intel2['gre_qnt'].values
```

In [133]:
```
def gre_conversion(array):
    n = 0
    array_mutated = array.copy()
    for vals in np.nditer(array):
        if vals <= 800 and vals > 790:
            array_mutated[n] = 166
        elif vals <= 790 and vals > 780:
            array_mutated[n] = 164
        elif vals <= 780 and vals > 770:
            array_mutated[n] = 163
        elif vals <= 770 and vals > 760:
            array_mutated[n] = 161
        elif vals <= 760 and vals > 750:
            array_mutated[n] = 160
        elif vals <= 750 and vals > 740:
            array_mutated[n] = 159
        elif vals <= 740 and vals > 730:
            array_mutated[n] = 158
        elif vals <= 730 and vals > 720:
            array_mutated[n] = 157
        elif vals <= 720 and vals > 710:
            array_mutated[n] = 156
        elif vals <= 710 and vals > 690:
            array_mutated[n] = 155
        elif vals <= 690 and vals > 680:
            array_mutated[n] = 154
        elif vals <= 680 and vals > 670:
            array_mutated[n] = 153
        elif vals <= 670 and vals > 650:
            array_mutated[n] = 152
        elif vals <= 650 and vals > 630:
            array_mutated[n] = 151
        elif vals <= 630 and vals > 620:
            array_mutated[n] = 150
        elif vals <= 620 and vals > 600:
            array_mutated[n] = 149
        elif vals <= 600 and vals > 580:
            array_mutated[n] = 148
        n = n + 1
    return array_mutated
```

In [134]:
```
adj_gre_qnt = np.apply_along_axis(gre_conversion, 0, gre_qnt_array)
```

In [135]:
```
income_intel['adj_gre_qnt'] = adj_gre_qnt
```

In [136]:
```
income_intel
```

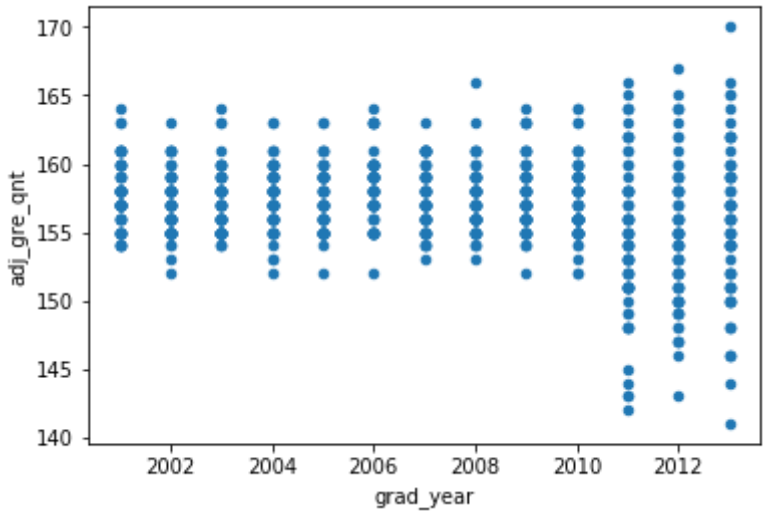
Out[136]:

	grad_year	gre_qnt	salary_p4	adj_gre_qnt
0	2001	740	67400	158
1	2001	722	67600	157
2	2001	736	58704	158
3	2001	770	64707	161
4	2001	735	51737	158
5	2001	764	64010	161
6	2001	739	60080	158
7	2001	706	56263	155
8	2001	706	62109	155
9	2001	701	50189	155
10	2001	710	58721	155
11	2001	735	65380	158
12	2001	753	52857	160
13	2001	690	63572	154
14	2001	774	65892	163
15	2001	726	67454	157
16	2001	703	59346	155
17	2001	724	70031	157
18	2001	758	53441	160
19	2001	711	61008	156
20	2001	703	50065	155
21	2001	734	75612	158
22	2001	736	59580	158
23	2001	749	57825	159
24	2001	733	52809	158
25	2001	728	57492	157
26	2001	690	64686	154
27	2001	732	53067	158
28	2001	725	58902	157
29	2001	722	62094	157
...	...	...	...	...
970	2013	159	79263	159
971	2013	148	104782	148
972	2013	160	94013	160

	grad_year	gre_qnt	salary_p4	adj_gre_qnt
973	2013	156	74032	156
974	2013	150	84220	150
975	2013	163	74940	163
976	2013	158	83293	158
977	2013	151	78340	151
978	2013	157	91066	157
979	2013	154	87169	154
980	2013	154	90033	154
981	2013	151	98650	151
982	2013	151	70455	151
983	2013	154	91133	154
984	2013	150	91796	150
985	2013	162	73780	162
986	2013	156	96927	156
987	2013	159	71875	159
988	2013	158	103357	158
989	2013	162	73780	162
990	2013	157	79055	157
991	2013	154	91529	154
992	2013	150	75940	150
993	2013	155	97688	155
994	2013	162	75260	162
995	2013	160	100430	160
996	2013	160	82198	160
997	2013	154	84340	154
998	2013	162	87600	162
999	2013	157	82854	157

1000 rows × 4 columns

In [137]:
```
# Make a scatter plot to check the data after adjustment
grad_year = income_intel['grad_year']
adj_gre_quant = income_intel['adj_gre_qnt']
income_intel.plot(x='grad_year', y='adj_gre_qnt', kind='scatter')
plt.show()
```



According to this scatter plot, the variance of GRE scores after 2011 is much higher than the adjusted GRE scores before 2011. This is because the raw GRE scores before 2011 are not lower than 580; therefore, no scores are lower than 150 on the new scoring range. The scatter plot which shows heteroskedasticity problem suggests that we should adopt a different method to standardize these scores. One way is to use z-scores. The implementation is as follows:

In [138]:
```
#zscore = lambda x: (x - x.mean()) / x.std()
df = income_intel.copy()
gre2 = df.groupby('grad_year').transform(lambda x : (x - x.mean()) / x.s
td())
gre2.head()
```

Out[138]:

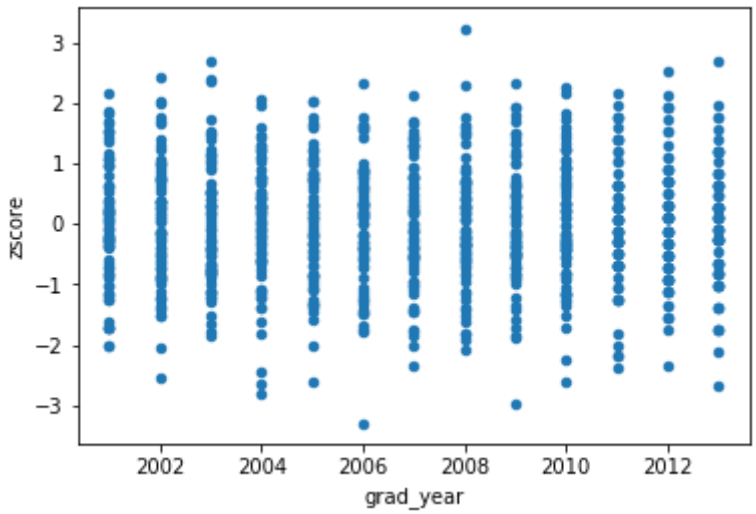
	gre_qnt	salary_p4	adj_gre_qnt
0	0.417950	1.051621	0.152934
1	-0.347280	1.083060	-0.283211
2	0.247899	-0.315369	0.152934
3	1.693334	0.628288	1.461368
4	0.205386	-1.410564	0.152934

In [139]:
```
income_intel['zscore'] = gre2['gre_qnt']
income_intel.head()
```

Out[139]:

	grad_year	gre_qnt	salary_p4	adj_gre_qnt	zscore
0	2001	740	67400	158	0.417950
1	2001	722	67600	157	-0.347280
2	2001	736	58704	158	0.247899
3	2001	770	64707	161	1.693334
4	2001	735	51737	158	0.205386

In [140]:
```
# create a scatter plot
grad_year = income_intel['grad_year']
zscore = income_intel['zscore']
income_intel.plot(x='grad_year', y='zscore', kind='scatter')
plt.show()
```



As we can see from the scatter plot of z-scores against graduation year, normalizing the GRE score using z-scores handles the variance in the data better than normal conversion using the GRE table. We will use z-scores to run a regression.

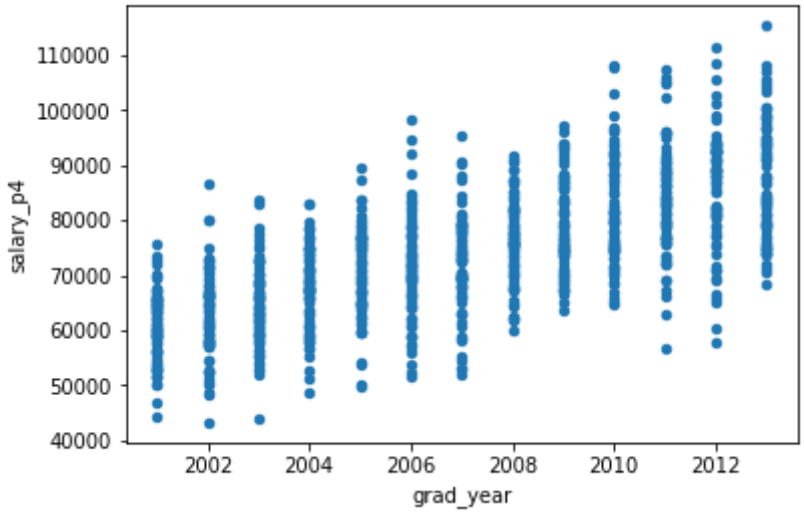
In [141]:
```
income_intel.head()
```

Out[141]:

	grad_year	gre_qnt	salary_p4	adj_gre_qnt	zscore
0	2001	740	67400	158	0.417950
1	2001	722	67600	157	-0.347280
2	2001	736	58704	158	0.247899
3	2001	770	64707	161	1.693334
4	2001	735	51737	158	0.205386

(c) Create a scatter plot of income 4 years after graduation and graduation year

In [142]:
```
# Make a scatter plot
grad_year = income_intel['grad_year']
salary = income_intel['salary_p4']
income_intel.plot(x='grad_year', y='salary_p4', kind='scatter')
plt.show()
```



The scatter plot shows a detectable upward trend in salary which is a data drift problem. The solution is to detrend the data by choosing 2001 as the base year, calculate the average growth rate of salary, and adjust every data point (i.e. discount by average growth rate) to 2001 dollars term. The implementation of this solution is as follows:

In [143]:
```
avg_inc_by_year = income_intel['salary_p4'].groupby(income_intel['grad_year']).mean().values
avg_inc_by_year
```

Out[143]:
```
array([60710.19480519, 63033.93506494, 64518.28571429, 67772.96103896,
       70492.05194805, 71677.68831169, 72133.19480519, 76432.06493506,
       79030.18181818, 81740.79220779, 83563.36363636, 86012.03896104,
       87299.97368421])
```

In [144]:
```
#avg_growth_rate = ((avg_inc_by_year[1:] - avg_inc_by_year[:-1]) / avg_inc_by_year[:-1]).mean()
avg_growth_rate_vec = (avg_inc_by_year[1:] - avg_inc_by_year[:-1]) / avg_inc_by_year[:-1]
avg_growth_rate = avg_growth_rate_vec.mean()
```

In [145]:
```
income_intel['adj_salary'] = income_intel['salary_p4'] / (1 + avg_growth_rate)**(income_intel['grad_year'] - 2001)
```

In [146]:
```
income_intel.head()
```

Out[146]:

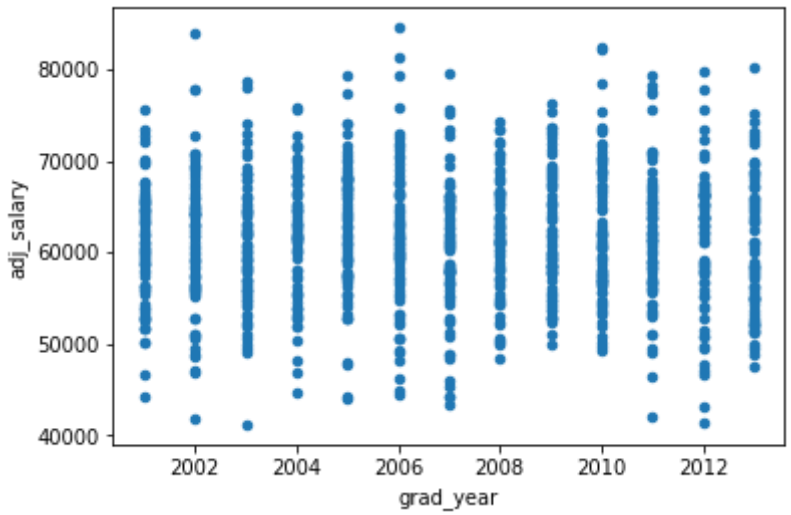
	grad_year	gre_qnt	salary_p4	adj_gre_qnt	zscore	adj_salary
0	2001	740	67400	158	0.417950	67400.0
1	2001	722	67600	157	-0.347280	67600.0
2	2001	736	58704	158	0.247899	58704.0
3	2001	770	64707	161	1.693334	64707.0
4	2001	735	51737	158	0.205386	51737.0

In [147]:
```
income_intel.tail()
```

Out[147]:

	grad_year	gre_qnt	salary_p4	adj_gre_qnt	zscore	adj_salary
995	2013	160	100430	160	0.838600	69757.499599
996	2013	160	82198	160	0.838600	57093.766325
997	2013	154	84340	154	-0.273033	58581.574392
998	2013	162	87600	162	1.209145	60845.932141
999	2013	157	82854	157	0.282784	57549.416228

In [148]:
```
# Make a scatter plot to see the change
grad_year = income_intel['grad_year']
salary = income_intel['adj_salary']
income_intel.plot(x='grad_year', y='adj_salary', kind='scatter')
plt.show()
```



The salary data is now detrended. We can proceed to regression using adjusted salary and z-scores.

(d) Rerun the regression using changes in part (b) and (c). Report new coefficients. How do these coefficients differ from part (a)? Interpret why changes in part (b) and (c) resulted in changes in coefficient values. Does the result provide evidence against or for the hypothesis?

In [149]:
```
# Using z-scores
# Defining independent and dependent variables
X_zscore, y_new_salary = income_intel['zscore'], income_intel['adj_salar
y']

# Run regression and fit the model
X_zscore = sm.add_constant(X_zscore, prepend=False)
regressed_new_salary = sm.OLS(y_new_salary, X_zscore)
res_new_salary = regressed_new_salary.fit()
print(res_new_salary.summary())
```

OLS Regression Results

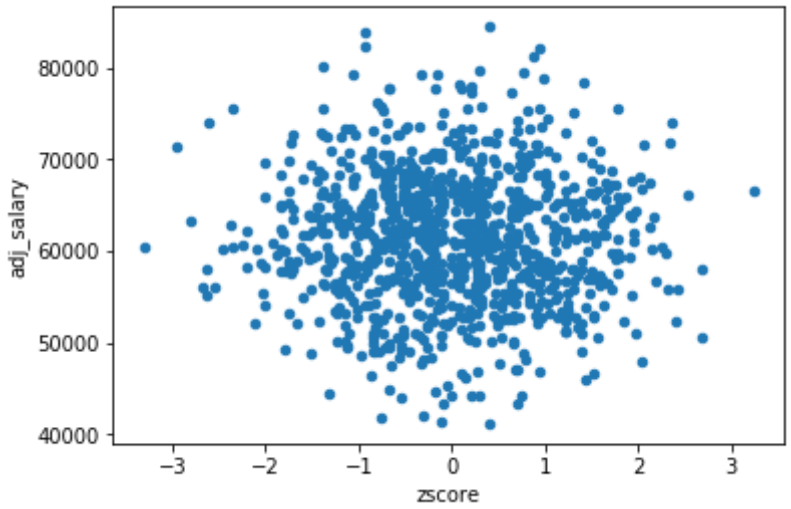
Dep. Variable:	adj_salary	R-squared:	0.000
Model:	OLS	Adj. R-squared:	-0.001
Method:	Least Squares	F-statistic:	0.3947
Date:	Wed, 17 Oct 2018	Prob (F-statistic):	0.530
Time:	08:35:06	Log-Likelihood:	-10291.
No. Observations:	1000	AIC:	2.059e+04
Df Residuals:	998	BIC:	2.060e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
zscore	-142.7355	227.198	-0.628	0.530	-588.575	303.104
const	6.142e+04	225.716	272.109	0.000	6.1e+04	6.19e+04

Omnibus:	0.774	Durbin-Watson:	2.025
Prob(Omnibus):	0.679	Jarque-Bera (JB):	0.686
Skew:	0.059	Prob(JB):	0.710
Kurtosis:	3.049	Cond. No.	1.01

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [150]:
```
# Make a diagnostic scatter plot
adj_salary = income_intel['adj_salary']
zscore = income_intel['zscore']
income_intel.plot(x='zscore', y='adj_salary', kind='scatter')
plt.show()
```



Re-estimated result from the OLS model using adjusted GRE z-scores and adjusted salary data shows that

- β_0 is equal to 69080 with standard error of 225.72
- β_1 is equal to -142.7355 with standard error of 227.20 (noted that p-value is more than 0.05)

The coefficients are different from the result from part (a). In part (a), β_0 is equal to 89540 with standard error of 878.764 and β_1 is equal to -25.7632 with standard error of 1.365. The coefficients changed because we adjusted our data which underlies the independent variable and dependent variable. In the new result, the coefficient of the GRE z-score can be interpreted as follows: as z-score increases by one unit, the salary decreases by 142.74 dollars. In contrast, according to the result from the regression model prior to any adjustment, the salary decreases by 25.76 when GRE quantitative score increases by one unit.

Regarding hypothesis testing, we are curious to know if higher intelligence is associated with higher income. Looking at p-value from the regression result before adjustment, we found that GRE scores actually negatively associate with income and this relationship is significant since p-value is less than 0.05. However, we know that GRE scores should be normalized and salary should be detrended. Thus, we should focus on the result of the linear regression after data adjustment. Looking at the p-value from the latter case, we found that it is 0.53 which is much higher than the significance level of 0.05. This suggests that GRE scores do not have any linear relationship with salary. The scatter plot of z scores and adjusted salary also reiterates this conclusion. Therefore, we changed our conclusion about the relationship between GRE scores, as evidence of intelligence, and higher income.

P-value and our conclusion changed because we normalized the GRE score and salary due to systematic and time drift problems in our data. Prior to normalization, plotting GRE scores against salary seems to show a downward trend (as shown in part (a)) which is reflected in the statistically significant result in part (a). However, after normalization, we can see from the diagnostic scatter plot in part (d) that there is no linear relationship between income and GRE quantitative scores. As a result, p-value after the normalization of data is higher than the significance level.

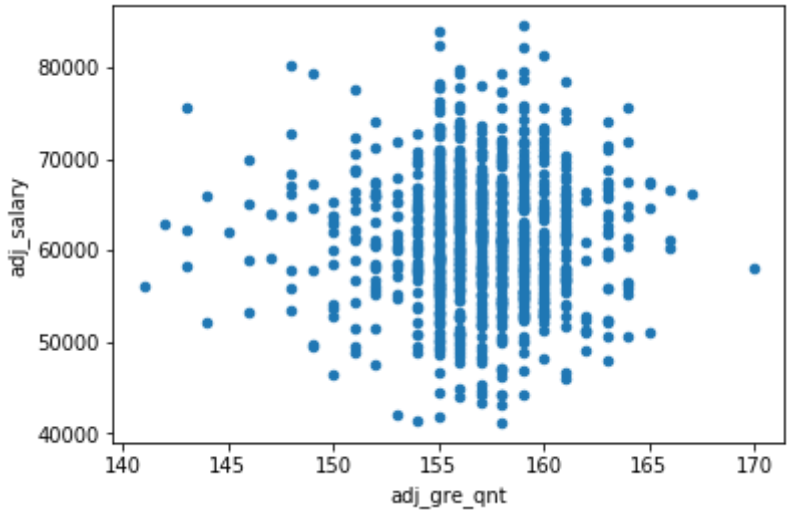
I also ran a regression on adjusted GRE scores that were done by conversion. The result is in line with the result we obtained from using z scores. (Please see below)

In [151]:
```
# Using adjusted GRE scores
# Defining independent and dependent variables
X_adj_gre_qnt, y_new_salary2 = income_intel['adj_gre_qnt'], income_intel
['adj_salary']

# Run regression and fit the model
X_adj_gre_qnt = sm.add_constant(X_adj_gre_qnt, prepend=False)
regressed_new_salary2 = sm.OLS(y_new_salary2, X_adj_gre_qnt)
res_new_salary2 = regressed_new_salary2.fit()
print(res_new_salary2.summary())
```

OLS Regression Results

```
==============================================================================
=======
Dep. Variable:             adj_salary   R-squared:
  0.001
Model:                            OLS   Adj. R-squared:
-0.000
Method:                 Least Squares   F-statistic:
0.5305
Date:                Wed, 17 Oct 2018   Prob (F-statistic):
  0.467
Time:                        08:35:06   Log-Likelihood:
-10291.
No. Observations:                1000   AIC:                            2.
059e+04
Df Residuals:                     998   BIC:                            2.
060e+04
Df Model:                           1

Covariance Type:            nonrobust

==============================================================================
========
                  coef    std err          t      P>|t|      [0.025
  0.975]
------------------------------------------------------------------------------
--------
adj_gre_qnt   -48.8426     67.061     -0.728      0.467    -180.440
  82.755
const        6.908e+04   1.05e+04      6.566      0.000    4.84e+04
8.97e+04
==============================================================================
=======
Omnibus:                        0.748   Durbin-Watson:
  2.025
Prob(Omnibus):                  0.688   Jarque-Bera (JB):
  0.658
Skew:                           0.058   Prob(JB):
  0.720
Kurtosis:                       3.050   Cond. No.
7.31e+03
==============================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 7.31e+03. This might indicate that t
here are
strong multicollinearity or other numerical problems.
```

In [152]:
```
# Make a diagnostic scatter plot
adj_salary = income_intel['adj_salary']
adj_gre_qnt = income_intel['adj_gre_qnt']
income_intel.plot(x='adj_gre_qnt', y='adj_salary', kind='scatter')
plt.show()
```



3. Asssessment of Kossinets and Watts

In their article *Origins of Homophily in an Evolving Social Network*, Kossinets and Watts examined the question of how homophily, the tendency of "like to associate with like," originates. More specifically, they set out to answer the question of how and on what grounds individuals choose to make or break certain ties over others and how the former choices might be able to explain why similar people tend to become acquainted than dissimilar counterparts. Historically, two distinct theoretical approaches explain why similar people form ties with the like, choice homophily (i.e. individual preferences) vis-à-vis induced homophily (i.e. structural proximity). To see which school of thought is correct, the authors set out to answer their research question by using a network data set from a particular research university. Their analysis shows that both choice and induced homophily play an important, but partial, role in originating homophily in their population of interests. In other words, both individual preferences and structural proximity reinforce each other in how the network of like people forms.

Kossinets and Watts constructed their data set by merging information obtained from three different databases which are (1) logs of email interactions of undergraduate and graduate students, faculty and staff within a university over one academic year, (2) individual attributes such as status, gender, age etc. and (3) records of course registration. The population of the study consists of 30,396 observations which include undergraduate students (21%), graduate and professional students (27%), faculty members (13%), administrators and staff (13.4%) and affiliates (which includes postdoctoral researchers, visiting scholars, exchange students and recent alumni; 25%) within a large U.S. research university. The data spans over a one year period, but the authors note that their email logs include 7,156,162 messages during 270 days of observation (p. 411), which corresponds to the length of the two academic semesters. The description and definitionof all variables can be found in Appendix A of the paper (p. 439-442).

Regarding their choice of data cleaning, there are three potential problems. Firstly, Kossinets and Watts included only messages that were sent to a single recipient. Although this category makes up 82 percent of thier data, cleaning data in this fashion does not take into account emails that were sent to multiple recipients which can shed light on the origins of homophily. For instance, some users may send emails to a group of acquaintances without sending them to a particular person because they rely on other mode of communication. This also leads us to question on to what extent email logs can explain origins of homophily which will be discussed in the next section. Secondly, the authors noted that email accounts provided by university departments were excluded from the data set because they cannot be matched with employee records. Exclusion of departmental emails poses a problem to the analysis because it is questionable if the data set accurately reflects how individuals with both departmental and general email accounts choose to make or break ties, especially communication among users from the same department. If departmental emails were to be included, it may increase the role of structural proximity and change the authors' conclusion. Finally, Kossinets and Watts mentioned that a set of heuristics were used to determine conflicting values such as gender. This approach also casts doubt on how accurate the authors capture "similarity" among observations, especially when the authors did not report the percentage of conflicting and missing observations.

One weakness of the match between explaining social relationships and email logs is that social relationships have more aspects than what can be captured by one means of communication. For example, younger users in the study may use other means of communication such as private messengers and telephone to communicate rather than emails. Some users may even have more than one email accounts for use in the personal and professional contexts. In addition, due to privacy concerns, email logs are limiting because the authors were not able to analyze the content of email conversation. Since the content was censored, the authors adopted the sliding window filter method to track how network of individuals evolve over time, determining how ties were established and lapsed. However, it is difficult to capture the depth of these relationships. For instance, some users may frequently communicate with each other through university emails, but they may not choose to spend

time with each other outside of work or study context. As Kossinets and Watts noted, electronic communication through emails in certain organizations may systematically differ from one another and from normal everyday interaction. The authors addressed this limitation by proposing that future studies could conduct comparative analyses of network evolution of various environments such as businesses and government agencies. Finally, the authors admitted that their email logs data lack some attributes such as race and socioeconomic status. Due to limited available information, they were not able to address this concern in the research design. The authors may have missed an important clue to the origin of homophily because subjects in the study who are not considered similar based on available categories may be bonded by races or socioeconomic status. Overall, the use of university email logs to explain origins of homophily casts doubt on how much the result of the study can be generalized because of the intrinsic complexity of social relationships.