assignment2

March 18, 2021

1 Assignment 2 - Continuous Visualization

Imagine you're a data scientist working for the University of Michigan teaching and learning team, and one of your job responsibilities is to offer instructional advice based on course performance metrics. As part of this activity you might be asked to analyze student's grade distribution in a range of undergraduate and graduate level courses to draw comparisons between courses and come up with insights regarding how to enhance residential education across different subjects.

1.1 Question 1 Draw probability density plot (30%)

Your first task is to compare 5 different distributions in three different ways. The five distributions are as follows:

- a t-distribution with 9, 99, 999, and 9999 degrees of freedom with mean 0 and standard deviation 2.
- a normal distribution with mean 0 and standard deviation 2.

First compare the five distributions using a **probability density plot** within a single figure so that each of the curves is in a different color and line type.

Next compare the five distributions using a **violin plot** within a single figure so that each of the curves is in a different color.

Next compare the five distributions using a **box and wiskers** within a single figure so that each of the distributions is in a different color.

If you need points for your a paticular plot type, take a sample of 500 points.

Please make a well-designed and well-annotated plots (e.g. visually appealing, titles, labels, etc).

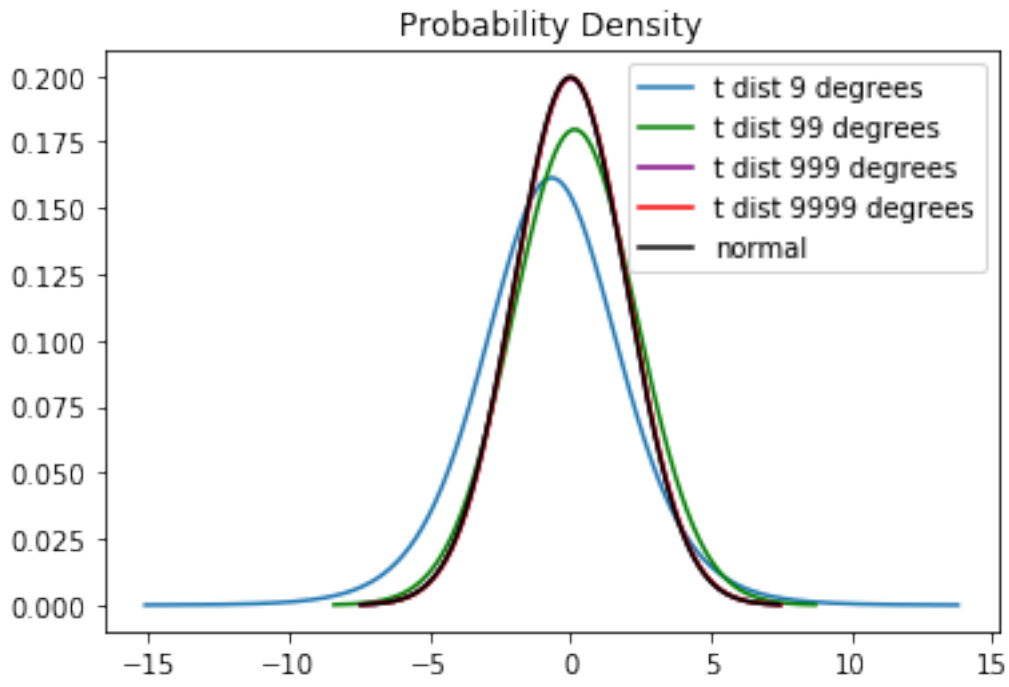
Hint: You can use the method ".ppf(·)" to get the density at a point of a scipy distribution. You might want to use scipy.stats.t, scipy.stats.norm, scipy.stats.norm.rvs, and/or stats.t.rvs.

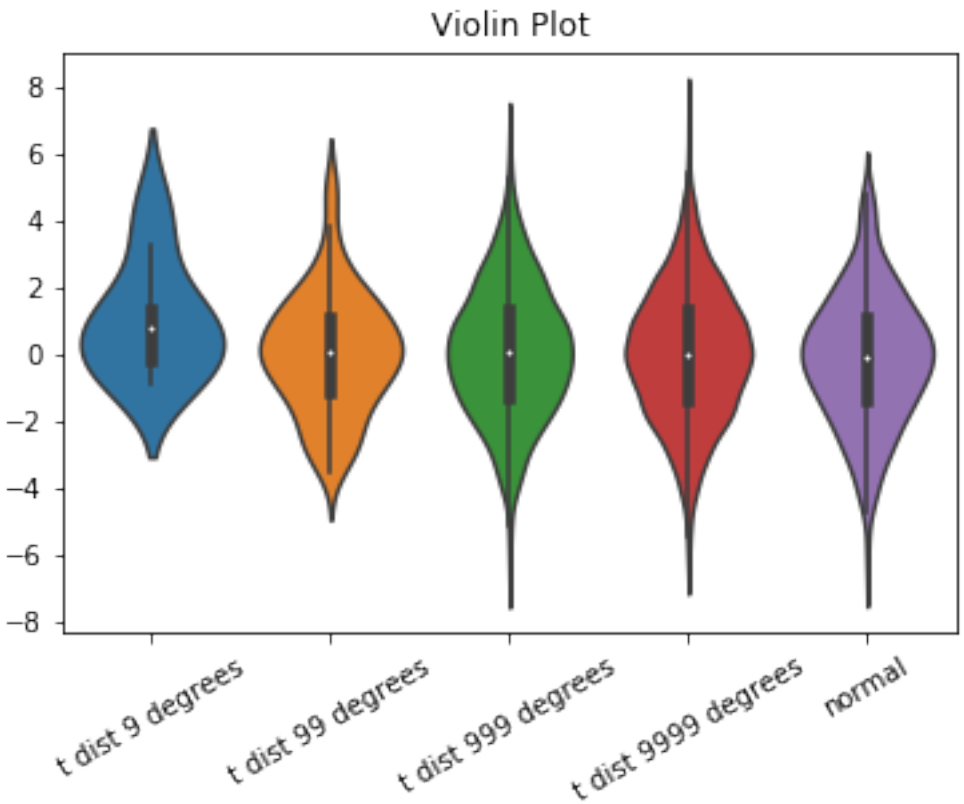
[1]:
```
import scipy.stats as stats
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import matplotlib.patches as mpatches
from itertools import repeat
```

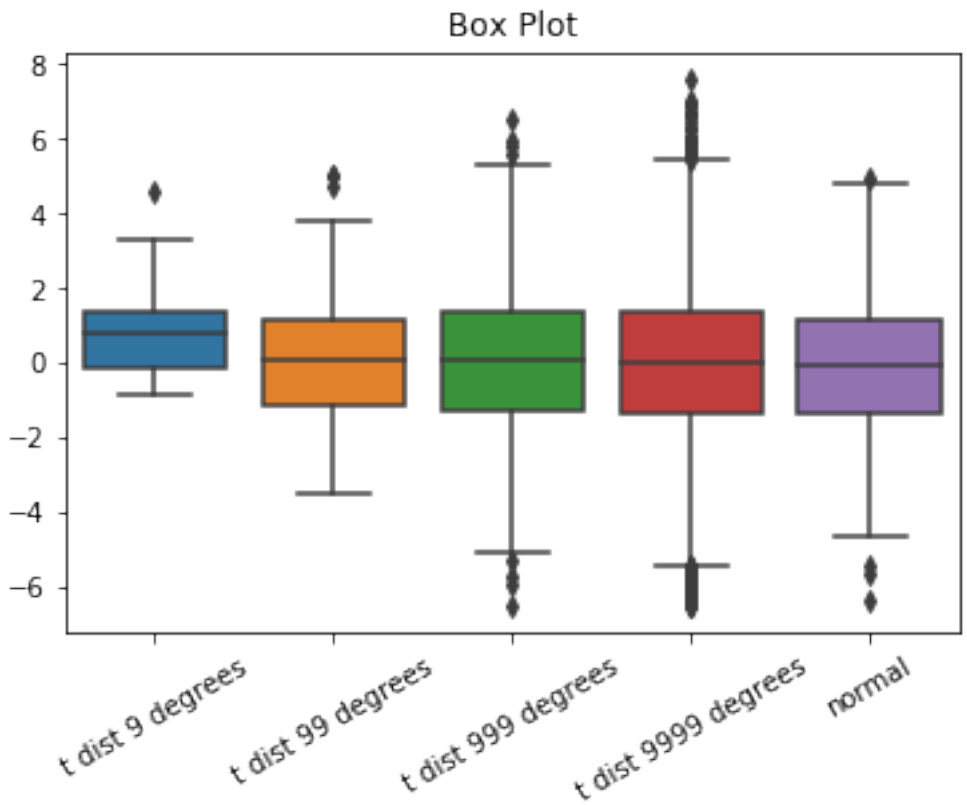
```
#stat.norm.rvs returns an array of random variables that are distributed␣
 ↪normally (a sample)
#stats.norm returns a distribution

def make_t_distribution(sample_size, mean, sd):
    t_sample = stats.t.rvs(sample_size - 1, mean, sd, sample_size) # Random␣
 ↪t-distribution sample
    sample_mean = np.mean(t_sample) # sample mean
    sample_std = np.std(t_sample) # sample standard deviation
    t_dist = stats.t(df = sample_size - 1, loc = sample_mean, scale =␣
 ↪sample_std) # make a t-distribution based on the sample
    x_axis = np.linspace(t_dist.ppf(0.0001), t_dist.ppf(0.9999), 500) #␣
 ↪Generate an x-axis based on t-quantile values

    return t_dist, x_axis


def make_prob_plot():
    plt.figure()
    d1,x1 = make_t_distribution(10,0,2) #t with 9 degrees of freedom
    d2,x2 = make_t_distribution(100,0,2)#t with 99 degrees of freedom
    d3,x3 = make_t_distribution(1000,0,2)#t with 999 degrees of freedom
    d4,x4 = make_t_distribution(10000,0,2)#t with 9999 degrees of freedom
    d5 = stats.norm(0,2) #normally distributed data with mean 0, std 2 and n 500
    x5 = np.linspace(d5.ppf(0.0001), d5.ppf(0.9999), 500)
    plt.plot(x1,d1.pdf(x1), label = 't dist 9 degrees')
    plt.plot(x2,d2.pdf(x2),c='green', label = 't dist 99 degrees')
    plt.plot(x3,d3.pdf(x3),c='purple', label = 't dist 999 degrees')
    plt.plot(x4,d4.pdf(x4),c='red', label = 't dist 9999 degrees')
    plt.plot(x5,d5.pdf(x5),c='black',label = 'normal')
    plt.title('Probability Density')
    plt.legend()



rv1 = stats.t.rvs(9, 0,2,10) #t dist with 9 degrees of freedom mean = 0 and sd␣
 ↪= 2
rv2 = stats.t.rvs(99,0,2,100)
rv3 = stats.t.rvs(999,0,2,1000)
rv4 = stats.t.rvs(9999,0,2,10000)
rv5=stats.norm.rvs(0,2,500)

def make_violin():
    plt.figure()
    distributions_v =[rv1,rv2,rv3,rv4,rv5]
    ax = sns.violinplot(data=distributions_v)
    ax.set_xticklabels(['t dist 9 degrees','t dist 99 degrees',
```

```
                        't dist 999 degrees','t dist 9999 degrees','normal'],␣
 ↪rotation=30)
    plt.title('Violin Plot')


def make_box():
    plt.figure()
    distributions_b=[rv1,rv2,rv3,rv4,rv5]
    ax = sns.boxplot(data=distributions_b)
    ax.set_xticklabels(['t dist 9 degrees','t dist 99 degrees',
                        't dist 999 degrees','t dist 9999 degrees','normal'],␣
 ↪rotation=30)
    plt.title('Box Plot')


make_prob_plot(), make_violin(),make_box()
```

[1]: (None, None, None)




Violin Plot
8
6
4
2
0
−2
−4
−6
−8
t dist 9 degrees
t dist 99 degrees
t dist 999 degrees
t dist 9999 degrees
normal



1.2 Question 2 Grade Distribution Comparison (40%)

Now you have impressed the management team, you have been given a sample data file `assets/class_grades.csv` for a number of courses, and you have been asked to consider the letter grades for STATS 250, DATASCI 306, MATH 217, ENGLISH 125, ECON 101, EECS 545 for the past records since 2015. The student grades are stored in 6 columns: * `STATS250_grade` stores the letter grades for those who took the STATS 250 course * `DATASCI306_grade` stores the letter grades for those who took the DATASCI 306 course * `MATH217_grade` stores the letter grades for those who took the MATH 217 course * `ENGLISH125_grade` stores the letter grades for those who took the ENGLISH 125 course * `ECON101_grade` stores the letter grades for those who took the ECON 101 course * `EECS545_grade` stores the letter grades for those who took the EECS 545 course

Prior to drawing plots for student grade distribution, it's useful to compute the total student enrollments for each course (of course, you need to ignore NAN values) and convert student's letter grades into standard grade points. Here's a nice table on the grade point systems available at the umich website:

Letter Grade	Grade Point
A+	4.3
A	4.0
A-	3.7
B+	3.3

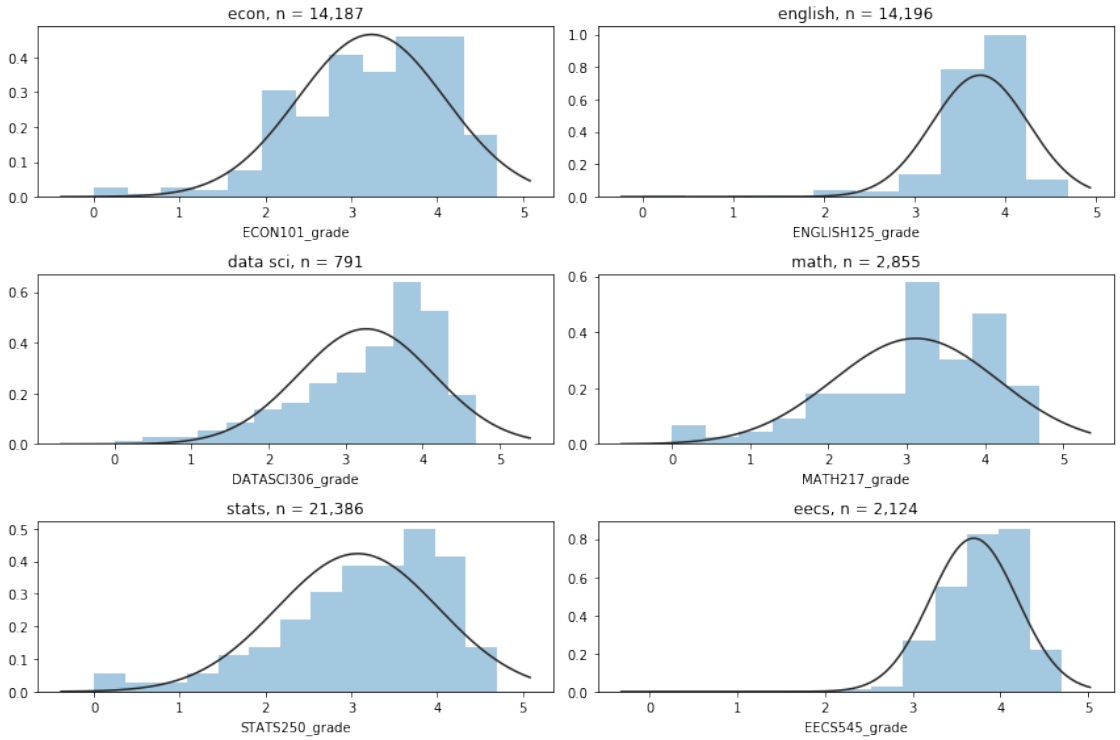
Letter Grade	Grade Point
B	3
B-	2.7
C+	2.3
C	2
C-	1.7
D+	1.3
D	1
D-	0.7
E	0

You are asked to: * Make a **3** * **2** figure (so 6 subplots) such that for each course you have a **histogram** using the student grade samples respectively * Remove the gaps between the bars in the histograms if any * For each probability plot, you should overlay a normal distribution with the same mean and standard deviation parameters as you see in the samples (you can calculate this!) * You should of course use a legend on each plot to specify the corresponding course name and number of students involved. For example, you can draw a legend and specify "STATS 250, n=5000" to indicate that you are analyzing STATS 250 course with 5000 enrolled students records being used for analysis

Hints: * To make subplots, one good way to start with is to use `fig, ax = plt.subplot()` * To remove the gaps that might show up in histograms, you can customize the `bins` parameter * If you want to make histograms using the distplot function in seaborn package, you need to specify the parameters `kde = False` and `norm_hist = True`

[2]:
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from scipy.stats import norm

grades = pd.read_csv('assets/class_grades.csv')
grades = grades.replace({'A+':4.7, 'A':4.0, 'A-':3.7,'B+':3.3,'B':3.0,'B-':2.
 ↪7,'C+':2.3,'C':2.0,'C-':1.7,'D+':1.3,'D':1.0,'D-':0.7,'E':0})
econ=grades['ECON101_grade'].dropna()
english=grades['ENGLISH125_grade'].dropna()
data=grades['DATASCI306_grade'].dropna()
math=grades['MATH217_grade'].dropna()
statistics=grades['STATS250_grade'].dropna()
eecs=grades['EECS545_grade'].dropna()

def grade_distribution():
    fig,ax = plt.subplots(3,2,figsize=(12,8))


    plt.subplot(3,2,1)
    plt.title('econ, n = 14,187')
    sns.distplot(econ,norm_hist =True, kde=False,fit=norm,bins = 12) #norm_hist␣
 ↪normailizes y axis
```

```
    plt.subplot(3,2,2)
    plt.title('english, n = 14,196')
    sns.distplot(english,norm_hist =True, kde=False,fit=norm,bins=10)

    plt.subplot(3,2,3)
    plt.title('data sci, n = 791')
    sns.distplot(data,norm_hist =True, kde=False,fit=norm,bins=13)

    plt.subplot(3,2,4)
    plt.title('math, n = 2,855')
    sns.distplot(math,norm_hist =True, kde=False,fit=norm,bins=11)

    plt.subplot(3,2,5)
    plt.title('stats, n = 21,386')
    sns.distplot(statistics,norm_hist =True, kde=False,fit=norm,bins=13)

    plt.subplot(3,2,6)
    plt.title('eecs, n = 2,124')
    sns.distplot(eecs,norm_hist =True, kde=False,fit=norm,bins=13)

    fig.tight_layout()


grade_distribution()
```



2 Question 3 Grade Distribution Normality Check (30%)

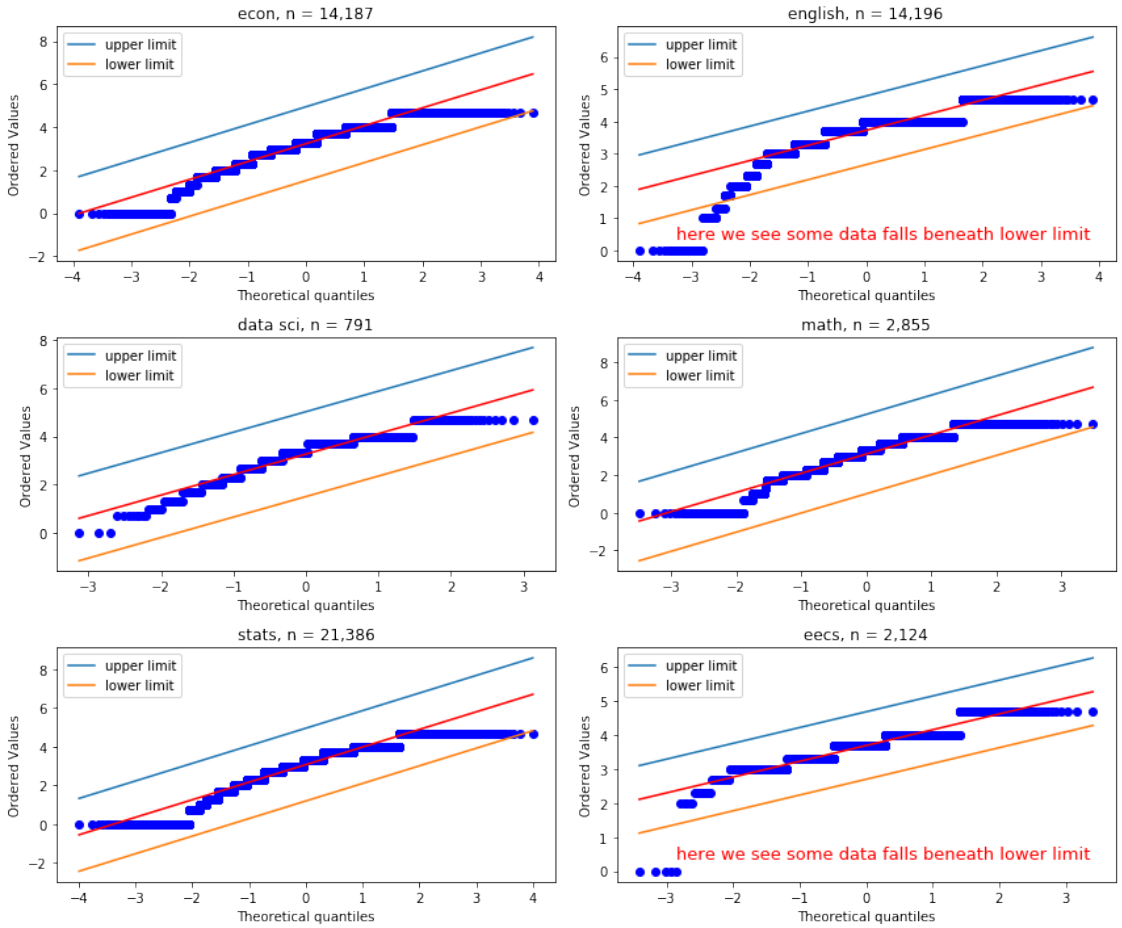
Seeing the student grade distributions of the 6 large residential courses, the team is tempted to draft recommendations for instructors and report to them what particular aspects could be addressed to improve students' academic learning outcome. However, before they launch statistical tests, they need to verify if the student grades data approximately follows normal distribution, a sufficient condition rendering the design of statistical models valid for those courses. You suggest that a QQ-plot is a great method to determine how similar a distribution is to another. Great idea! * Make a 3 * 2 figure (again, 6 subplots) so that for each course you have a QQ plot using the student grade samples versus the normal distribution with the same mean and standard deviation * You need to use a legend on each plot to specify the corresponding course name and number of students involved. For example, you can draw a legend and specify "STATS 250, n=5000" to indicate that you are analyzing STATS 250 course with 5000 enrolled students records being used for analysis * For each QQ-plot, mark observations which are 2 standard deviations outside from the QQ-line (a straight line showing the theoretical values for different quantiles under normal distribution). You may use the annotate tool inside the graph to circle each such instance or design some other manner to call out these points. * Write a couple of sentence about the figure discussing the courses and whether they seem to be normally distributed.

Hint: You may find using `fig = plt.figure()` and `fig.add_subplot()` functions helpful to create subplots. You don't have to use these functions though.

[3]:
```
import scipy.stats as stats
def grade_normality():
    fig,ax= plt.subplots(3,2,figsize=(12,10))

    ecn = plt.subplot(3,2,1)
    (osm1,osr1),(slope1,intercept1,_1)=stats.probplot(econ,dist=stats.
 ↪norm,plot=ecn)
    upper_lim1 = slope1*osm1+intercept1+2*econ.std()
    lower_lim1 = slope1*osm1+intercept1-2*econ.std()
    ecn.plot(osm1,upper_lim1,label='upper limit')
    ecn.plot(osm1,lower_lim1,label='lower limit')
    plt.title('econ, n = 14,187')
    plt.legend()

    eng = plt.subplot(3,2,2)
    (osm2,osr2),(slope2,intercept2,_2)=stats.probplot(english,dist=stats.
 ↪norm,plot=eng)
    upper_lim2 = slope2*osm2+intercept2+2*english.std()
    lower_lim2 = slope2*osm2+intercept2-2*english.std()
    eng.plot(osm2,upper_lim2,label='upper limit')
    eng.plot(osm2,lower_lim2,label='lower limit')
    eng.text(0.95, 0.08, 'here we see some data falls beneath lower limit',
        verticalalignment='bottom', horizontalalignment='right',
```

```
        transform=eng.transAxes,
        color='red', fontsize=13)
    plt.title('english, n = 14,196')
    plt.legend()

    dat = plt.subplot(3,2,3)
    (osm3,osr3),(slope3,intercept3,_3)=stats.probplot(data,dist=stats.
↪norm,plot=dat)
    upper_lim3 = slope3*osm3+intercept3+2*data.std()
    lower_lim3 = slope3*osm3+intercept3-2*data.std()
    dat.plot(osm3,upper_lim3,label='upper limit')
    dat.plot(osm3,lower_lim3,label='lower limit')
    plt.title('data sci, n = 791')
    plt.legend()

    mat=plt.subplot(3,2,4)
    (osm4,osr4),(slope4,intercept4,_4)=stats.probplot(math,dist=stats.
↪norm,plot=mat)
    upper_lim4 = slope4*osm4+intercept4+2*math.std()
    lower_lim4 = slope4*osm4+intercept4-2*math.std()
    mat.plot(osm4,upper_lim4,label='upper limit')
    mat.plot(osm4,lower_lim4,label='lower limit')
    plt.title('math, n = 2,855')
    plt.legend()

    stat= plt.subplot(3,2,5)
    (osm5,osr5),(slope5,intercept5,_5)=stats.probplot(statistics,dist=stats.
↪norm,plot=stat)
    upper_lim5 = slope5*osm5+intercept5+2*statistics.std()
    lower_lim5 = slope5*osm5+intercept5-2*statistics.std()
    stat.plot(osm5,upper_lim5,label='upper limit')
    stat.plot(osm5,lower_lim5,label='lower limit')
    plt.title('stats, n = 21,386')
    plt.legend()

    ecs=plt.subplot(3,2,6)
    (osm6,osr6),(slope6,intercept6,_6)=stats.probplot(eecs,dist=stats.
↪norm,plot=ecs)
    upper_lim6 = slope6*osm6+intercept6+2*eecs.std()
    lower_lim6 = slope6*osm6+intercept6-2*eecs.std()
    ecs.plot(osm6,upper_lim6,label='upper limit')
    ecs.plot(osm6,lower_lim6,label='lower limit')
    ecs.text(0.95, 0.08, 'here we see some data falls beneath lower limit',
        verticalalignment='bottom', horizontalalignment='right',
        transform=ecs.transAxes,
        color='red', fontsize=13)
    plt.title('eecs, n = 2,124')
```

```
    plt.legend()


    fig.tight_layout()


grade_normality()
```



For the most part, the data seems normally distributed. Only EECS and English class have data that is outside of 2 standard deviations of the mean

[ ]: