HUANG

March 13, 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.

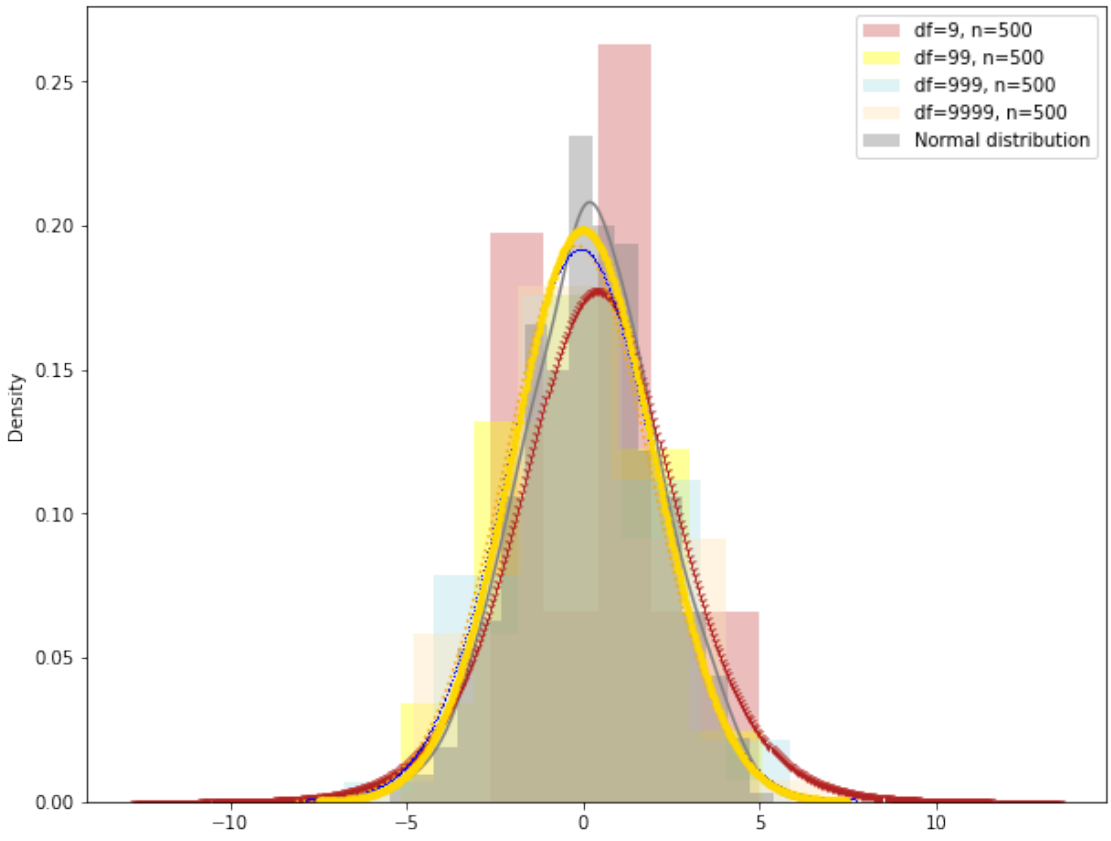
[1]:
```
from scipy.stats import norm
import scipy.stats as stats
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import statistics
```

[2]:
```
mean=0
sd=2

# df=9
sample_size1=10
t_sample1 = stats.t.rvs(sample_size1 - 1, mean, sd, sample_size1)
sample_mean1 = np.mean(t_sample1)
sample_std1 = np.std(t_sample1)
t_dist1 = stats.t(df = sample_size1 - 1, loc = sample_mean1, scale =␣
↪sample_std1)
x1 = np.linspace(t_dist1.ppf(0.0001), t_dist1.ppf(0.9999), 500)
y1 = t_dist1.pdf(x1)

# df=99
sample_size2=100
t_sample2 = stats.t.rvs(sample_size2 - 1, mean, sd, sample_size2)
sample_mean2 = np.mean(t_sample2)
sample_std2 = np.std(t_sample2)

t_dist2 = stats.t(df = sample_size2 - 1, loc = sample_mean2, scale =␣
↪sample_std2)
x2 = np.linspace(t_dist2.ppf(0.0001), t_dist2.ppf(0.9999), 500)
y2 = t_dist2.pdf(x2)

# df=999
sample_size3=1000
t_sample3 = stats.t.rvs(sample_size3 - 1, mean, sd, sample_size3)
sample_mean3 = np.mean(t_sample3)
sample_std3 = np.std(t_sample3)

t_dist3 = stats.t(df = sample_size3 - 1, loc = sample_mean3, scale =␣
↪sample_std3)
x3 = np.linspace(t_dist3.ppf(0.0001), t_dist3.ppf(0.9999), 500)
y3 = t_dist3.pdf(x3)

# df=9999
sample_size4=10000
t_sample4 = stats.t.rvs(sample_size4 - 1, 0, 2, sample_size4)
sample_mean4 = np.mean(t_sample4)
sample_std4 = np.std(t_sample4)

t_dist4 = stats.t(df = sample_size4 - 1, loc = sample_mean4, scale =␣
↪sample_std4)
x4 = np.linspace(t_dist4.ppf(0.0001), t_dist4.ppf(0.9999), 500)
y4 = t_dist4.pdf(x4)

# Normal distribution
```

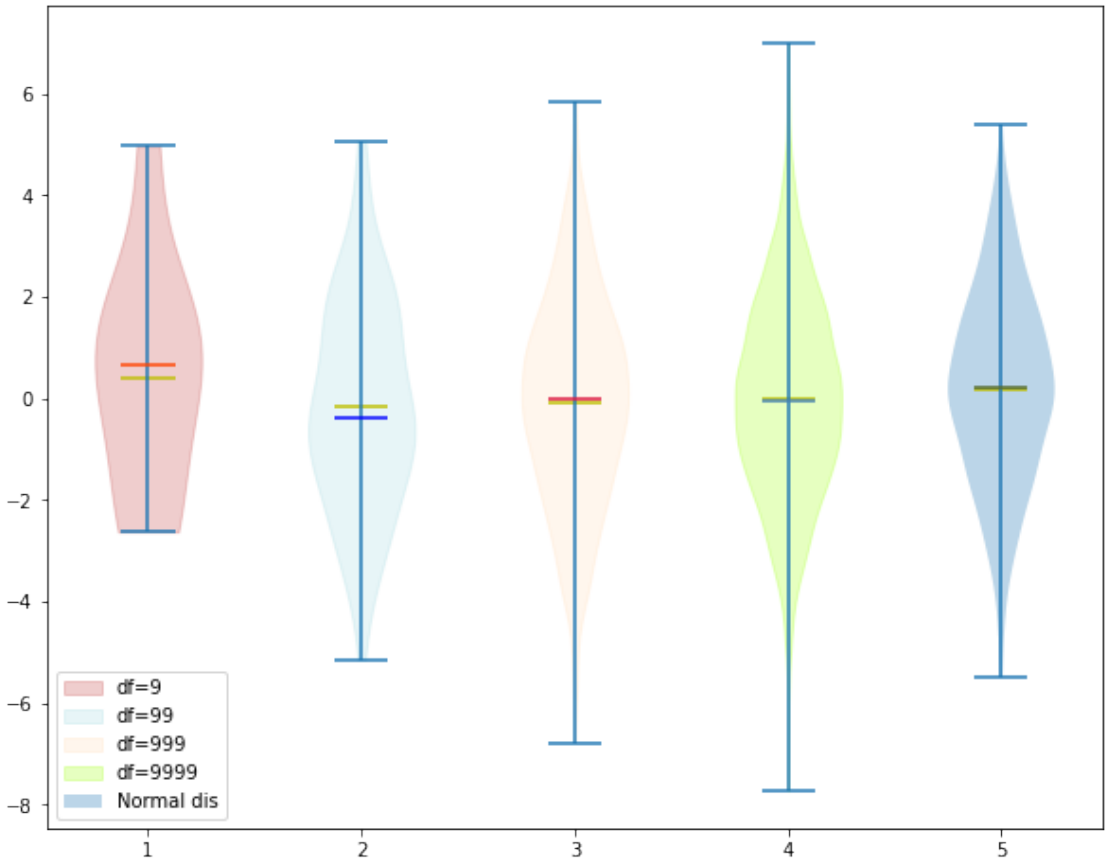
```
norm = norm.rvs(mean, sd, size=500)


# Create histograms
plt.figure(figsize=(10,8))
plot = sns.distplot(t_sample1, norm_hist=True, kde=False, bins=5, label="df=9,␣
↪n=500", color="indianred")
plot = sns.distplot(t_sample2, norm_hist=True, kde=False, bins=5, label="df=99,␣
↪n=500", color="yellow")
plot = sns.distplot(t_sample3, norm_hist=True, kde=False, bins=5,␣
↪label="df=999, n=500", color="powderblue")
plot = sns.distplot(t_sample4, norm_hist=True, kde=False, bins=5,␣
↪label="df=9999, n=500", color="moccasin")
plot = sns.distplot(norm, norm_hist=True, kde=True, label="Normal␣
↪distribution", color="grey")

# Create plot
plot.plot(x1,y1, '1', color='firebrick')
plot.plot(x2,y2, ':', color='orange')
plot.plot(x3,y3, ',', color='blue')
plot.plot(x4,y4, '.', color='gold')

# Add legend
plot.legend(loc="best")
```

```
C:\Users\huang\AppData\Local\Programs\Python\Python39\lib\site-
packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

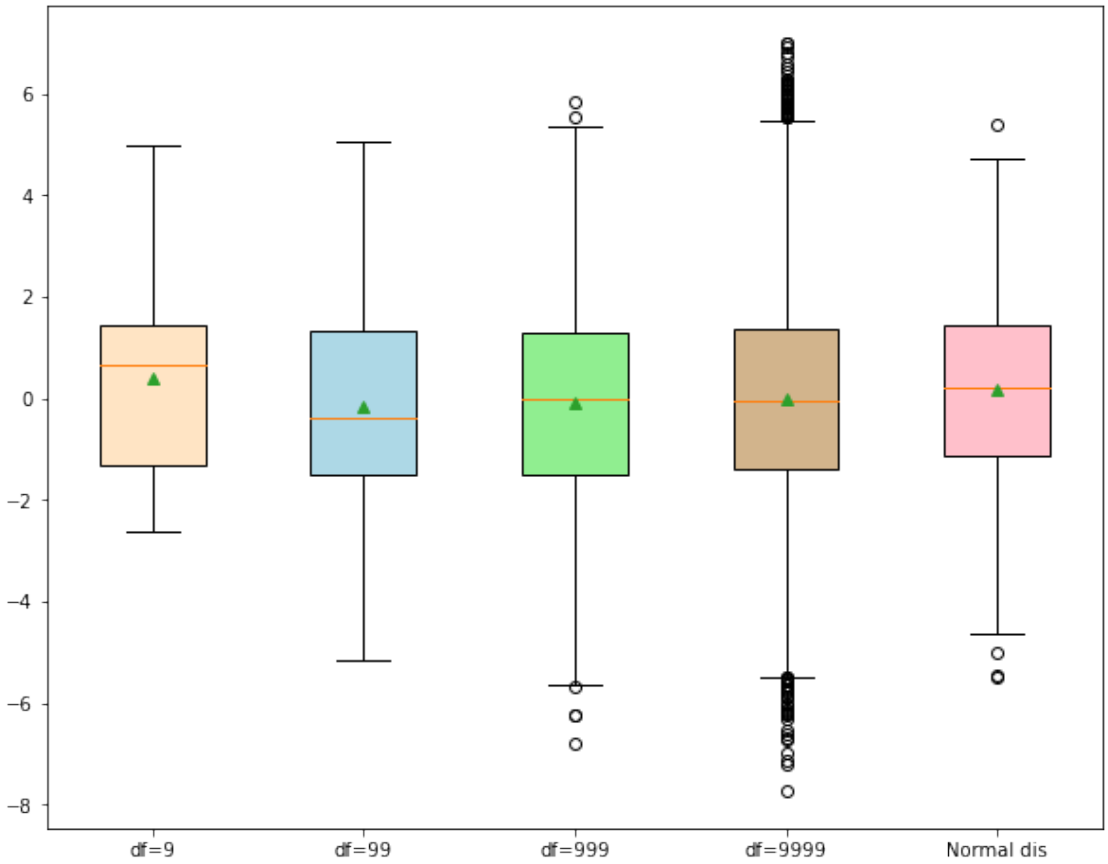
[2]: `<matplotlib.legend.Legend at 0x18a21b50760>`



[3]:
```
# nor_x = np.random.normal(loc=mean, scale=sd, size=500)

distri = (t_sample1,t_sample2,t_sample3,t_sample4,norm)
plt.figure(figsize=(10,8))
ax = plt.subplot(111)
violinplt = plt.violinplot(distri, showmeans=True, showmedians=True)

# Customize colors
violinplt['cmeans'].set_color('y')
violinplt['bodies'][0].set_color('indianred')
violinplt['bodies'][1].set_color('powderblue')
violinplt['bodies'][2].set_color('bisque')
violinplt['bodies'][3].set_color('greenyellow')

# Add labels and legend
label = ["df=9","df=99","df=999","df=9999","Normal dis"]
ax.legend(label, loc='lower left')

violinplt['cmedians'].
 ↪set_color(('orangered','b','crimson','steelblue','darkolivegreen'))
```



[4]:
```
plt.figure(figsize=(10,8))
box = plt.boxplot(distri, labels = ("df=9","df=99","df=999","df=9999","Normal␣
 ↪dis"), showmeans=True, patch_artist=True)

# Customize colors
colors = ['bisque','lightblue','lightgreen','tan','pink']
for patch, color in zip(box['boxes'], colors):
    patch.set_facecolor(color)
```



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

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

[5]:
```
grade = pd.read_csv("assets/class_grades.csv", index_col=0)
```

[6]:
```
grade.replace(["A+","A","A-","B+","B","B-","C+","C","C-","D+","D","D-","E"],[4.
↪3,4.0,3.7,3.3,3.0,2.7,2.3,2.0,1.7,1.3,1.0,0.7,0],inplace=True)
```

[13]:
```
eco101 = grade['ECON101_grade']
eco101.dropna(inplace=True)

eco101_n = np.asarray(eco101)
eco101_n = sorted(eco101_n)
fiteco101 = stats.norm.pdf(eco101_n, np.mean(grade['ECON101_grade']), np.
↪std(grade['ECON101_grade']))
```

[7]:
```
eng125 = grade['ENGLISH125_grade']
eng125.dropna(inplace=True)

eng125_n = np.asarray(eng125)
eng125_n = sorted(eng125_n)
```

```
fiteng125 = stats.norm.pdf(eng125_n, np.mean(grade['ENGLISH125_grade']), np.
 ↪std(grade['ENGLISH125_grade']))
```

[8]:
```
dat306 = grade['DATASCI306_grade']
dat306.dropna(inplace=True)

dat306_n = np.asarray(dat306)
dat306_n = sorted(dat306_n)
fitdat306 = stats.norm.pdf(dat306_n, np.mean(grade['DATASCI306_grade']), np.
 ↪std(grade['DATASCI306_grade']))
```

[9]:
```
mat217 = grade['MATH217_grade']
mat217.dropna(inplace=True)

mat217_n = np.asarray(mat217)
mat217_n = sorted(mat217_n)
fitmat217 = stats.norm.pdf(mat217_n, np.mean(grade['MATH217_grade']), np.
 ↪std(grade['MATH217_grade']))
```

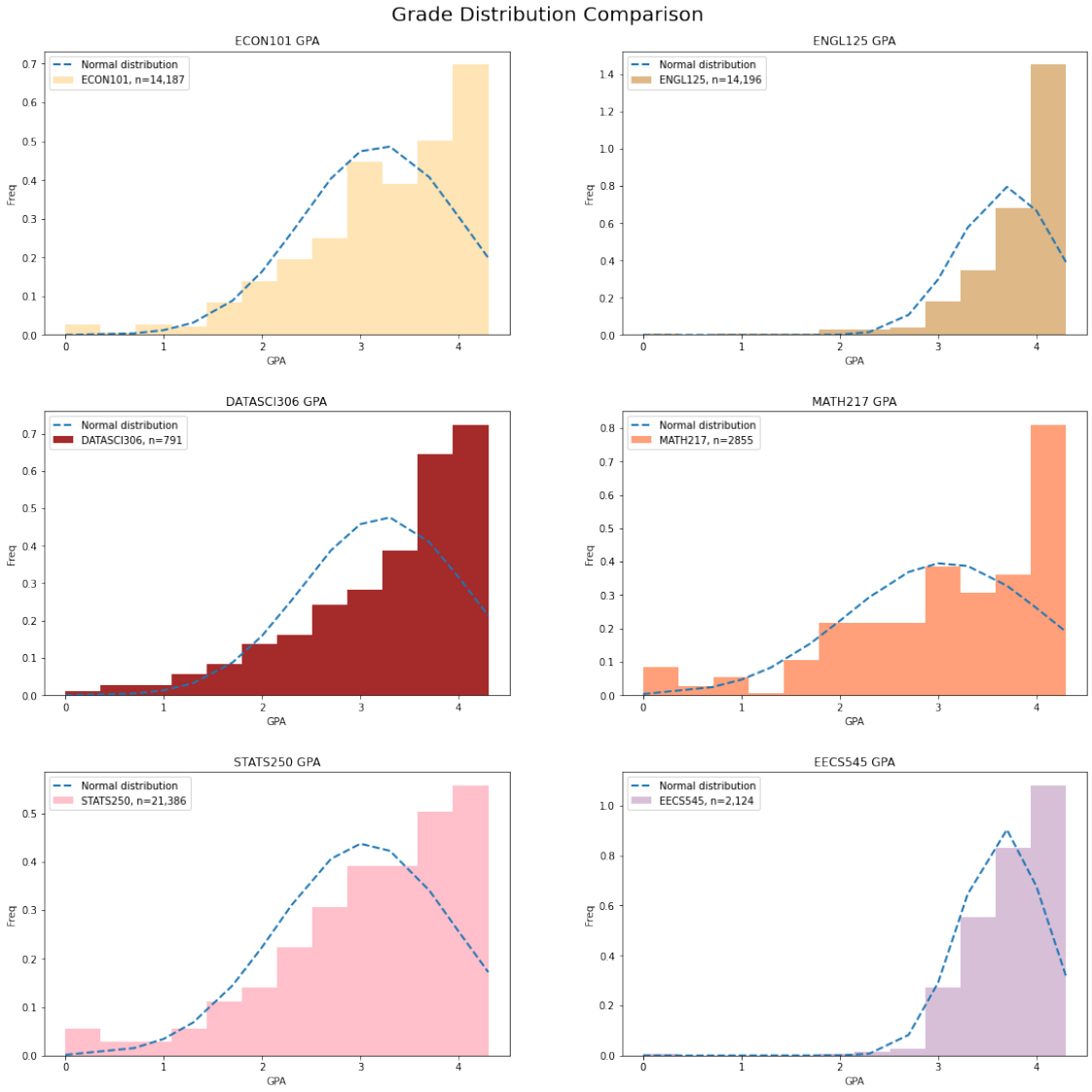
[10]:
```
sta250 = grade['STATS250_grade']
sta250.dropna(inplace=True)

sta250_n = np.asarray(sta250)
sta250_n = sorted(sta250_n)
fitsta250 = stats.norm.pdf(sta250_n, np.mean(grade['STATS250_grade']), np.
 ↪std(grade['STATS250_grade']))
```

[11]:
```
eec545 = grade['EECS545_grade']
eec545.dropna(inplace=True)

eec545_n = np.asarray(eec545)
eec545_n = sorted(eec545_n)
fiteec545 = stats.norm.pdf(eec545_n, np.mean(grade['EECS545_grade']), np.
 ↪std(grade['EECS545_grade']))
```

[14]:
```
# Create subplots
fig, ax = plt.subplots(3,2, figsize=(15,15), constrained_layout=True)

# ECON 101
plt.subplot(3,2,1)
plt.plot(eco101_n, fiteco101, '--', linewidth=2, label="Normal distribution")
plt.hist(eco101_n, density=True, bins=12, label="ECON101, n=14,187",␣
 ↪color="moccasin")
# Create title, labels and legend
plt.title("ECON101 GPA")
plt.xlabel("GPA")
plt.ylabel("Freq")
```

```
plt.legend()


# ENGL 125
plt.subplot(3,2,2)
plt.plot(eng125_n, fiteng125, '--', linewidth=2, label="Normal distribution")
plt.hist(eng125_n, density=True, bins=12, label="ENGL125, n=14,196",␣
 ↪color="burlywood")
# Create title, labels and legend
plt.title("ENGL125 GPA")
plt.xlabel("GPA")
plt.ylabel("Freq")
plt.legend()


# DATASCI 306
plt.subplot(3,2,3)
plt.plot(dat306_n, fitdat306, '--', linewidth=2, label="Normal distribution")
plt.hist(dat306_n, density=True, bins=12, label="DATASCI306, n=791",␣
 ↪color="brown")
# Create title, labels and legend
plt.title("DATASCI306 GPA")
plt.xlabel("GPA")
plt.ylabel("Freq")
plt.legend()


# MATH 217
plt.subplot(3,2,4)
plt.plot(mat217_n, fitmat217, '--', linewidth=2, label="Normal distribution")
plt.hist(mat217_n, density=True, bins=12, label="MATH217, n=2855",␣
 ↪color="lightsalmon")
# Create title, labels and legend
plt.title("MATH217 GPA")
plt.xlabel("GPA")
plt.ylabel("Freq")
plt.legend()


# STATS 250
plt.subplot(3,2,5)
plt.plot(sta250_n, fitsta250, '--', linewidth=2, label="Normal distribution")
plt.hist(sta250_n, density=True, bins=12, label="STATS250, n=21,386",␣
 ↪color='pink')
# Create title, labels and legend
plt.title("STATS250 GPA")
plt.xlabel("GPA")
```

```
plt.ylabel("Freq")
plt.legend()


# EECS 545
plt.subplot(3,2,6)
plt.plot(eec545_n, fiteec545, '--', linewidth=2, label="Normal distribution")
plt.hist(eec545_n, density=True, bins=12, label="EECS545, n=2,124",␣
 ↪color="thistle")
# Create title, labels and legend
plt.title("EECS545 GPA")
plt.xlabel("GPA")
plt.ylabel("Freq")
plt.legend()

# Add maintitle and padding
plt.suptitle("Grade Distribution Comparison", fontsize=20)
fig.set_constrained_layout_pads(w_pad=2/72, h_pad=2/72, hspace=0.2, wspace=0.2)
```



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

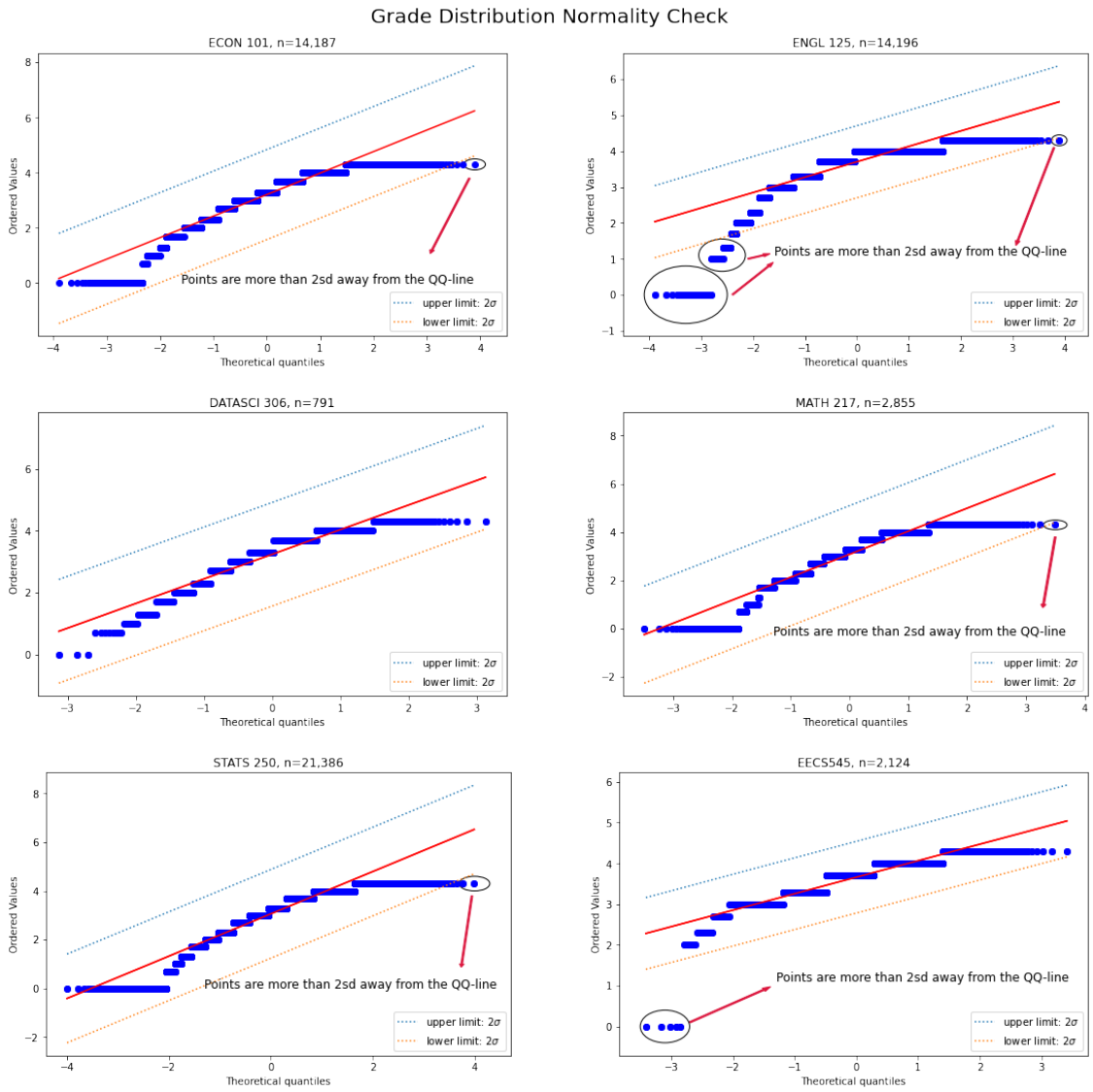
you are analyzing STATS 250 course with 5000 enrolled students records being used for analysis * For each QQ-plot, mark observations which are 2 standard deviations outside from the QQ-line (a straight line showing the theoretical values for different quantiles under normal distribution). You may use 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.

[15]:
```
import statistics

# Create a figure
fig = plt.figure(figsize=(15,15), constrained_layout=True)

# ECON 101
ax1 = fig.add_subplot(3,2,1)
stats.probplot(eco101, dist=stats.norm, plot=ax1)
# add limit  lines
(osm, osr), (slope, intercept, _) = stats.probplot(eco101, dist=stats.norm)
upper_limit = slope*osm + intercept + 2 * statistics.stdev(eco101)
lower_limit = slope*osm + intercept - 2 * statistics.stdev(eco101)
ax1.plot(osm, upper_limit, ':', label="upper limit: 2$\sigma$")
ax1.plot(osm, lower_limit, ':', label="lower limit: 2$\sigma$")
# add legend and title
ax1.legend(loc='lower right')
ax1.set_title("ECON 101, n=14,187")
# add annotation
plt.text(-1.6,0,'Points are more than 2sd away from the QQ-line',␣
 ↪fontsize='large') # add text
plt.arrow(3.8,3.8,-0.7,-2.6, width=0.03, color='crimson') # add arrow
circle1 = plt.Circle((3.9,4.3), 0.2, fill=None) # add circle
ax1.add_patch(circle1)

# ENGL 125
ax2 = fig.add_subplot(3,2,2)
stats.probplot(eng125, dist=stats.norm, plot=ax2)
# add limit  lines
(osm, osr), (slope, intercept, _) = stats.probplot(eng125, dist=stats.norm,␣
 ↪plot=ax2)
upper_limit = slope*osm + intercept + 2*statistics.stdev(eng125)
lower_limit = slope*osm + intercept - 2*statistics.stdev(eng125)
ax2.plot(osm, upper_limit, ':', label="upper limit: 2$\sigma$")
ax2.plot(osm, lower_limit, ':', label="lower limit: 2$\sigma$")
# add legend and title
ax2.legend(loc='lower right')
ax2.set_title("ENGL 125, n=14,196")
# add annotation
```

```
plt.text(-1.6,1.1,'Points are more than 2sd away from the QQ-line',␣
 ↪fontsize='large') # add text
plt.arrow(-2.1,1.0,0.3,0.1, width=0.03, color='crimson') # add arrow
plt.arrow(-2.4,0,0.7,0.8, width=0.03, color='crimson')
plt.arrow(3.8,4.1,-0.7,-2.6, width=0.03, color='crimson')
circle2 = plt.Circle((-3.3,0), 0.8, fill=None) # add circle
circle3 = plt.Circle((-2.6,1.1), 0.45, fill=None)
circle4 = plt.Circle((3.9,4.3), 0.15, fill=None)
ax2.add_patch(circle2)
ax2.add_patch(circle3)
ax2.add_patch(circle4)

# DATASCI 306
ax3 = fig.add_subplot(3,2,3)
stats.probplot(dat306, dist=stats.norm, plot=ax3)
# add limit  lines
(osm, osr), (slope, intercept, _) = stats.probplot(dat306, dist=stats.norm,␣
 ↪plot=ax3)
upper_limit = slope*osm + intercept + 2*statistics.stdev(dat306)
lower_limit = slope*osm + intercept - 2*statistics.stdev(dat306)
ax3.plot(osm, upper_limit, ':', label="upper limit: 2$\sigma$")
ax3.plot(osm, lower_limit, ':', label="lower limit: 2$\sigma$")
# add legend and title
ax3.legend(loc='lower right')
ax3.set_title("DATASCI 306, n=791")

# MATH 217
ax4 = fig.add_subplot(3,2,4)
stats.probplot(mat217, dist=stats.norm, plot=ax4)
# add limit  lines
(osm, osr), (slope, intercept, _) = stats.probplot(mat217, dist=stats.norm,␣
 ↪plot=ax4)
upper_limit = slope*osm + intercept + 2*statistics.stdev(mat217)
lower_limit = slope*osm + intercept - 2*statistics.stdev(mat217)
ax4.plot(osm, upper_limit, ':', label="upper limit: 2$\sigma$")
ax4.plot(osm, lower_limit, ':', label="lower limit: 2$\sigma$")
# add legend and title
ax4.legend(loc='lower right')
ax4.set_title("MATH 217, n=2,855")
# add annotation
plt.text(-1.3,-0.3,'Points are more than 2sd away from the QQ-line',␣
 ↪fontsize='large') # add text
plt.arrow(3.5,3.8,-0.2,-2.8, width=0.03, color='crimson') # add arrow
circle5 = plt.Circle((3.5,4.3), 0.2, fill=None) # add circle
ax4.add_patch(circle5)

# STATS 250
```

```
ax5 = fig.add_subplot(3,2,5)
stats.probplot(sta250, dist=stats.norm, plot=ax5)
# add limit  lines
(osm, osr), (slope, intercept, _) = stats.probplot(sta250, dist=stats.norm,␣
 ↪plot=ax5)
upper_limit = slope*osm + intercept + 2*statistics.stdev(sta250)
lower_limit = slope*osm + intercept - 2*statistics.stdev(sta250)
ax5.plot(osm, upper_limit, ':', label="upper limit: 2$\sigma$")
ax5.plot(osm, lower_limit, ':', label="lower limit: 2$\sigma$")
# add legend and title
ax5.legend(loc='lower right')
ax5.set_title("STATS 250, n=21,386")
# add annotation
plt.text(-1.3,0,'Points are more than 2sd away from the QQ-line',␣
 ↪fontsize='large') # add text
plt.arrow(3.95,3.8,-0.2,-2.8, width=0.03, color='crimson') # add arrow
circle6 = plt.Circle((4.0,4.3), 0.3, fill=None) # add circle
ax5.add_patch(circle6)

# EECS 545
ax6 = fig.add_subplot(3,2,6)
stats.probplot(eec545, dist=stats.norm, plot=ax6)
# add limit  lines
(osm, osr), (slope, intercept, _) = stats.probplot(eec545, dist=stats.norm,␣
 ↪plot=ax6)
upper_limit = slope*osm + intercept + 2*statistics.stdev(eec545)
lower_limit = slope*osm + intercept - 2*statistics.stdev(eec545)
ax6.plot(osm, upper_limit, ':', label="upper limit: 2$\sigma$")
ax6.plot(osm, lower_limit, ':', label="lower limit: 2$\sigma$")
# add legend and title
ax6.legend(loc='lower right')
ax6.set_title("EECS545, n=2,124")
# add annotation
plt.text(-1.3,1.1,'Points are more than 2sd away from the QQ-line',␣
 ↪fontsize='large') # add text
plt.arrow(-2.7,0.1,1.2,0.8, width=0.03, color='crimson') # add arrow
circle7 = plt.Circle((-3.1,0), 0.4, fill=None) # add circle
ax6.add_patch(circle7)

# add main title and set up the figure
plt.suptitle("Grade Distribution Normality Check", fontsize=20)
fig.set_constrained_layout_pads(w_pad=2/72, h_pad=2/72, hspace=0.2, wspace=0.2)
```



The figure of question 3 is likely a normal distribution, but not exact one. However, all of them are left skewed because the majority of students got A and A+ and only few of student got D and below on these courses and this causes extreme data. For all figures except DATASCI 306 course, there're ponts are more than two standard deviations away from the QQ-line. It might be caused by different reasons, like data entry error, unusual samples(for example, the student dropped the course during the semester, but the system doesn't remove the student) or a natural part of the population you are studying. For the first two reasons, my suggestion would be remove those points, but for the last reasoo, I will not remove those points.

[ ]: