Applied Econometrics Project

November 1, 2023

[4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import wooldridge as woo
import math
import seaborn as sns
import scipy.stats as stats
import statsmodels.formula.api as smf
from pandas_profiling import ProfileReport
import pandas_profiling
from fitter import Fitter
from scipy import stats
from boruta import BorutaPy
from sklearn.ensemble import RandomForestRegressor
from RegscorePy import *
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from BorutaShap import BorutaShap
import statsmodels.formula.api as sm
import statsmodels.stats.api as sms
```

[5]:
```
data = woo.data("attend")

data

data.drop(["missed", "atndrte"], axis=1, inplace=True)

# we drop these values because they don't have an impact on attendance,␣
 ↪"atndrte" is percent of "attend"
# and "missed" is minus attendance

# N=680, cross-sectional individual data on classes attended, bcuse attend
data
```

```
[5]:      attend  termGPA  priGPA  ACT  final  hwrte  frosh  soph   stndfnl
     0        27     3.19    2.64   23     28  100.0      0     1  0.472689
     1        22     2.73    3.52   25     26   87.5      0     0  0.052521
     2        30     3.00    2.46   24     30   87.5      0     0  0.892857
     3        31     2.04    2.61   20     27  100.0      0     1  0.262605
     4        32     3.68    3.32   23     34  100.0      0     1  1.733193
     ..      ...      ...     ...  ...    ...    ...    ...   ...       ...
     675      25     1.00    2.31   18     22  100.0      0     1 -0.787815
     676      32     2.69    2.38   19     19  100.0      0     1 -1.418067
     677      32     3.66    3.61   22     30  100.0      0     1  0.892857
     678      28     3.17    3.02   24     27  100.0      0     1  0.262605
     679      14     1.80    1.19   24     29   62.5      1     0  0.682773

     [680 rows x 9 columns]
```

Obs: 680

1. attend = classes attended out of 32
2. termgpa = GPA for term
3. priGPA = cumulative GPA prior to term
4. ACT = ACT score
5. final = final exam score
6. atndrte = percent classes attended
7. hwrte = percent homework turned in
8. frosh = 1 if freshman
9. soph = 1 if sophomore
10. skipped = number of classes skipped
11. stndfnl = (final - mean)/sd

```
[6]: description = data.describe()
     description
```

```
[6]:        attend     termGPA      priGPA         ACT       final       hwrte  \
     count  680.000000  680.000000  680.000000  680.000000  680.000000  674.000000
     mean    26.147059    2.601000    2.586775   22.510294   25.891176   87.908012
     std      5.455037    0.736586    0.544714    3.490768    4.709835   19.269258
     min      2.000000    0.000000    0.857000   13.000000   10.000000   12.500000
     25%     24.000000    2.137500    2.190000   20.000000   22.000000   87.500000
     50%     28.000000    2.670000    2.560000   22.000000   26.000000  100.000000
     75%     30.000000    3.120000    2.942500   25.000000   29.000000  100.000000
     max     32.000000    4.000000    3.930000   32.000000   39.000000  100.000000

                 frosh        soph     stndfnl
     count  680.000000  680.000000  680.000000
     mean     0.232353    0.576471    0.029659
     std      0.422644    0.494481    0.989461
     min      0.000000    0.000000   -3.308824
     25%      0.000000    0.000000   -0.787815
```

```
50%      0.000000    1.000000    0.052521
75%      0.000000    1.000000    0.682773
max      1.000000    1.000000    2.783613
```

[7]:
```
data.isnull().sum()
```

[7]:
```
attend     0
termGPA    0
priGPA     0
ACT        0
final      0
hwrte      6
frosh      0
soph       0
stndfnl    0
dtype: int64
```

[8]:
```
imputeVal_hwrte = description["hwrte"].iloc[5]
data['hwrte'].replace(np.nan, imputeVal_hwrte, inplace=True)
```

[9]:
```
data.isnull().sum()
```

[9]:
```
attend     0
termGPA    0
priGPA     0
ACT        0
final      0
hwrte      0
frosh      0
soph       0
stndfnl    0
dtype: int64
```

1 Descriptive Analysis: Perform a univariate analysis following the steps below

1.1 Begin by providing a descriptive analysis of your variables (include all predictors and response variable). This should include things like histograms, quantile plots, correlation plots, etc

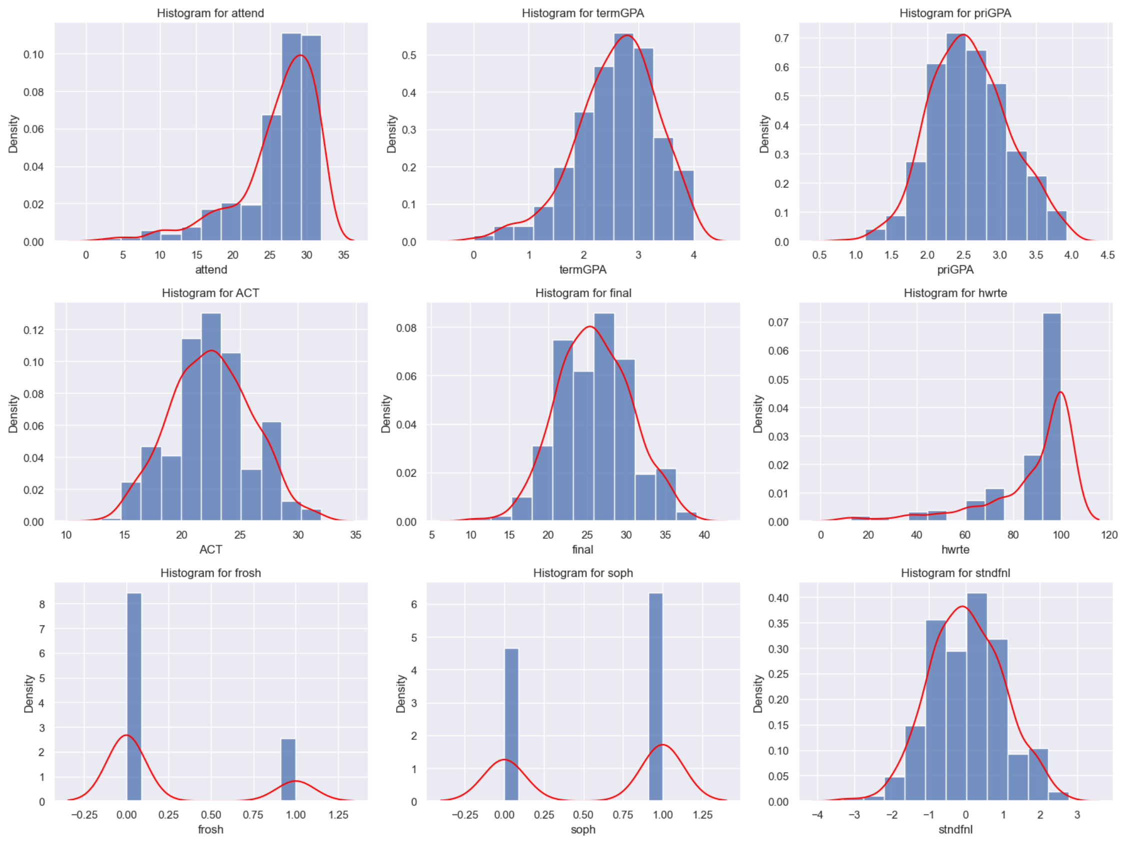
1.2 Histogram Analysis

- attend:
  - Outliers: No Outliers Observed from Histogram
  - Shape: Seems Leftly Skewed
  - Range: Doesn't seem large, between 2 and 32
  - Density: As we go more to the right side of the histogram, we see more data
- termGPA:

  - Outliers: No Outliers Observed from Histogram
  - Shape: Seems Normally Distributed
  - Range: Doesn't seem large, between 0 and 4
  - Density: Seems symmetric
- priGPA:
  - Outliers: No Outliers Observed from Histogram
  - Shape: Seems Normally Distributed
  - Range: Doesn't seem large, between 0 and 4
  - Density: Seems symmetric
- ACT:
  - Outliers: No Outliers Observed from Histogram
  - Shape: Seems Normally Distributed
  - Range: Doesn't seem large, between 15 and around 33
  - Density: Seems symmetric
- final:
  - Outliers: No Outliers Observed from Histogram
  - Shape: Seems Normally Distributed
  - Range: Doesn't seem large, between around 12 and around 38
  - Density: Seems symmetric
- hwrte:
  - Outliers: No Outliers Observed from Histogram
  - Shape: Seems Leftly Skewed
  - Range: The range of values seem relatively large to the dataset, between around 20 to around 100
  - Density: As we go more to the right side of the histogram, we see more data
- frosh:
  - Outliers: No Outliers Observed from Histogram
  - Shape: Categorical Variable, binary values
  - Range: Greater portion of non freshman than freshman
  - Density: Shows greater portion of non freshman
- soph:
  - Outliers: No Outliers Observed from Histogram
  - Shape: Categorical Variable, binary values
  - Range: Greater portion of sophmores than non sophmores
  - Density: Shows greater portion of sophmores
- stndfnl:
  - Outliers: No Outliers Observed from Histogram
  - Shape: Seems Normally Distributed
  - Range: Doesn't seem large, between -3 and 3
  - Density: Seems symmetric

[37]:
```
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(16, 12))


axes = axes.flatten()


```

```
for i, col in enumerate(data.columns):
    sns.histplot(data[col], stat="density", bins='sturges', ax=axes[i])
    sns.kdeplot(data[col], color="red", ax=axes[i], legend=False)
    axes[i].set_title("Histogram for " + col)
    axes[i].set_xlabel(col)


plt.tight_layout()


plt.show()
```

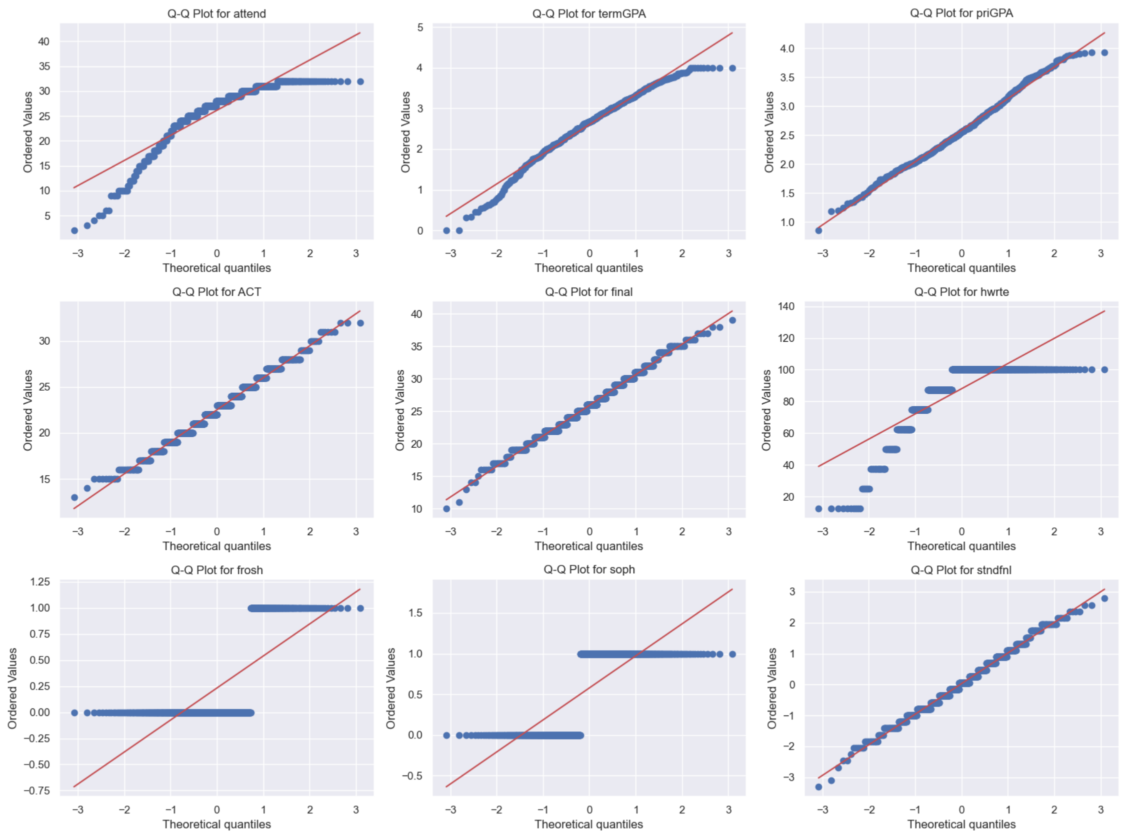


1.3 Quantile Plots

- attend: Does Not Appear Normally Distributed
- termGPA: Appears Normally Distributed
- priGPA: Appears Normally Distributed
- ACT: Appears Normally Distributed
- final: Appears Normally Distributed
- hwrte: Cannot Infer (Might have to do with NAN values)
- frosh: QQ Plots aren't appropriate for Categorical Variables

- soph: QQ Plots aren't appropriate for Categorical Variables
- stndfnl: Appears Normally Distributed

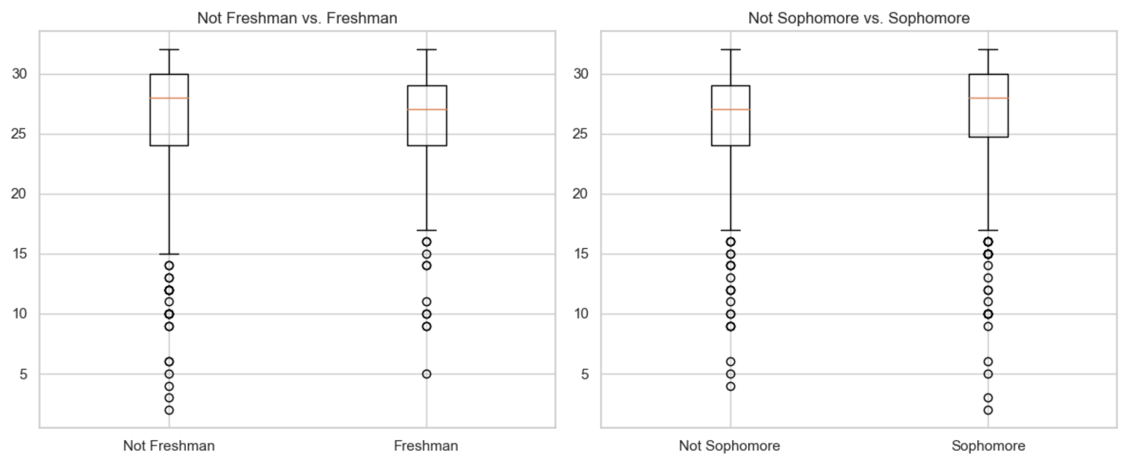
[38]:
```
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(16, 12))


axes = axes.flatten()


for i, col in enumerate(data.columns):
    stats.probplot(data[col], dist="norm", plot=axes[i])
    axes[i].set_title("Q-Q Plot for " + col)


plt.tight_layout()


plt.show()
```



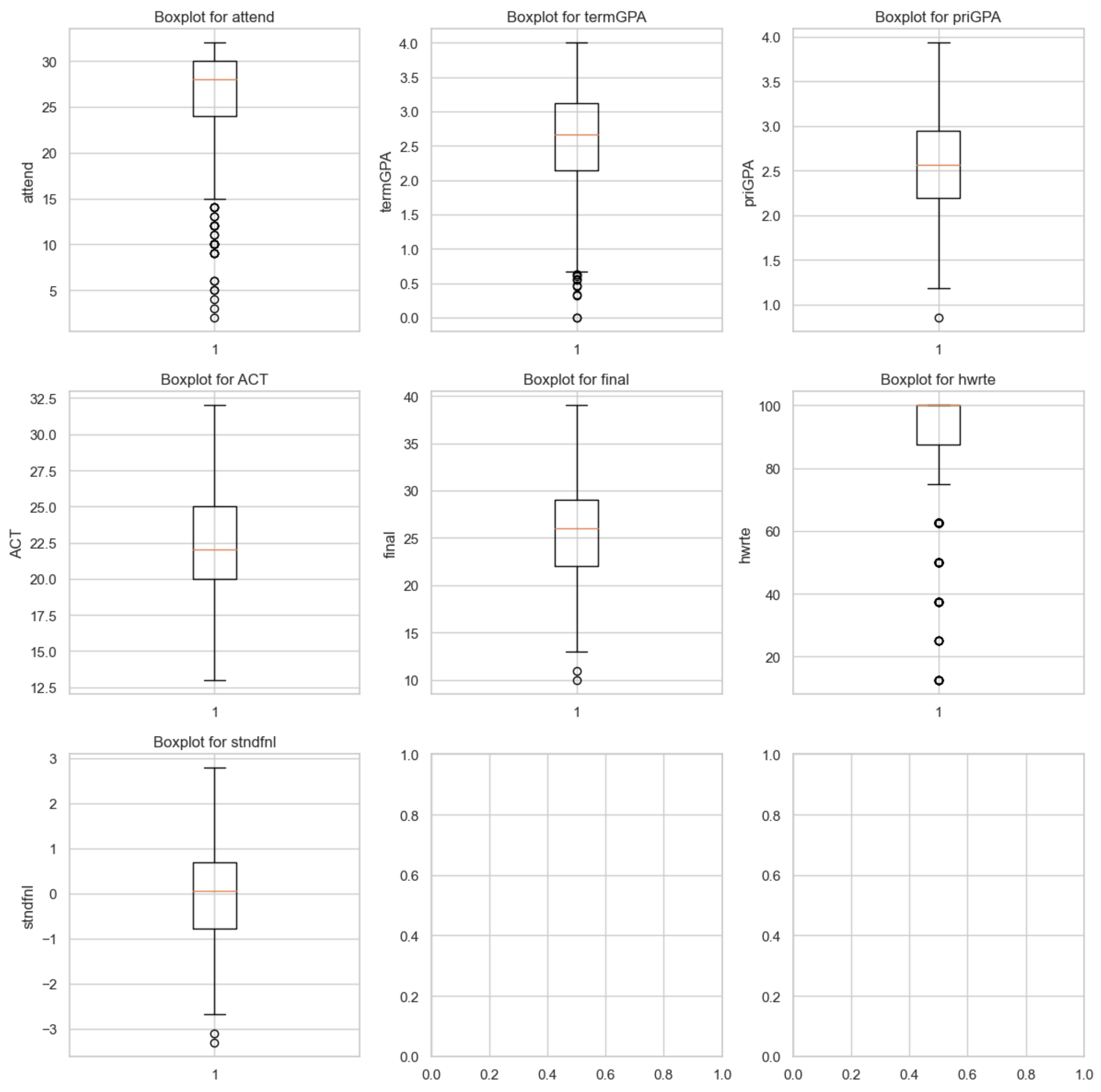
1.4 Box Plot

- Not Freshman vs Freshman:
  - Inference:
    - Not Freshman seem to have higher attendance based off the boxplot
    - The data from Not Freshman has more outliers than Freshman, representing the individuals that did not attend more than about 14 classes.
    - The average values are higher for classes attended for non Freshman.
    - The upper 25% of attendance values from not Freshman are higher than the upper 25% of attendance values of Freshman.
    - Freshman boxplot has higher minimum attendance values
- Not Sophmore vs Sophmore:
  - Inference:
    - Sophmores seem to have higher attendance based off the boxplot because they have a higher median and higher upper quartile 3.
    - Sophmores also have more outliers than non sophmores

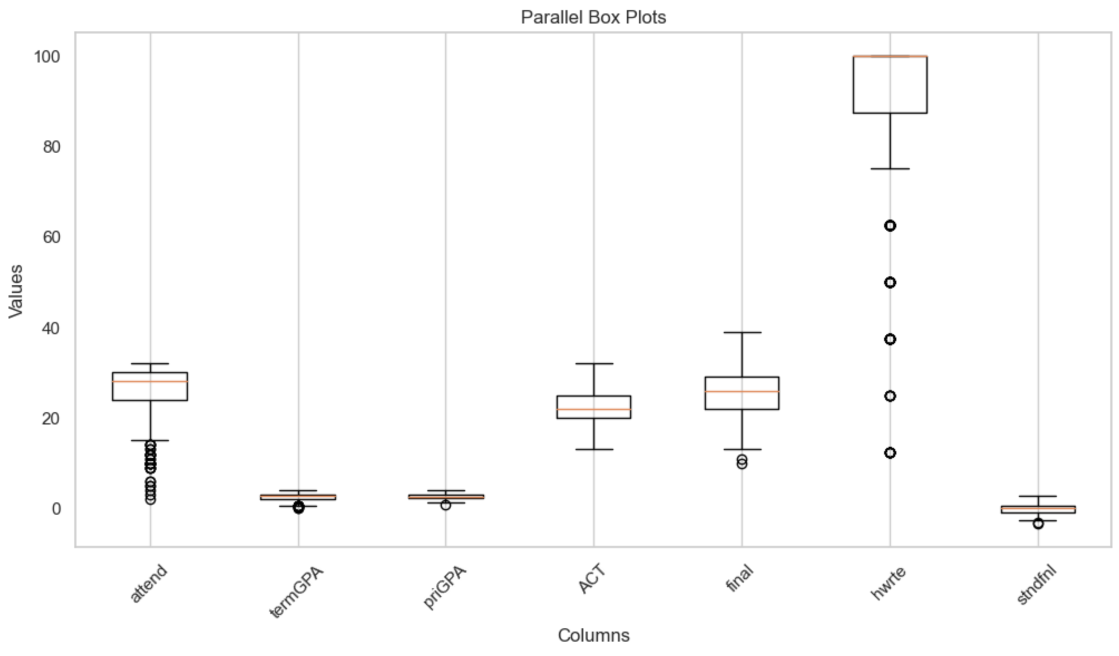
[24]:
```python
fig, axs = plt.subplots(1, 2, figsize=(12, 5))


data_set = [data[data.frosh == 0].attend, data[data.frosh != 0].attend]
axs[0].boxplot(data_set)
axs[0].set_xticks([1, 2])
axs[0].set_xticklabels(["Not Freshman", "Freshman"])
axs[0].set_title("Not Freshman vs. Freshman")


data_set = [data[data.soph == 0].attend, data[data.soph != 0].attend]
axs[1].boxplot(data_set)
axs[1].set_xticks([1, 2])
axs[1].set_xticklabels(["Not Sophomore", "Sophomore"])
axs[1].set_title("Not Sophomore vs. Sophomore")


plt.tight_layout()


plt.show()
```



[21]:
```
import matplotlib.pyplot as plt
import pandas as pd

# Define the number of rows and columns in the grid
num_rows, num_cols = 3, 3
num_plots = min(len(data.columns), num_rows * num_cols)

# Create a 3x3 grid of subplots and plot the boxplots for selected columns
fig, axes = plt.subplots(num_rows, num_cols, figsize=(12, 12))

# Specify the columns you want to include, excluding "frosh" and "soph"
columns_to_include = [col for col in data.columns if col not in ['frosh',␣
 ↪'soph']]

for i, column in enumerate(columns_to_include[:num_plots]):
    row, col = i // num_cols, i % num_cols
    ax = axes[row, col]
    ax.boxplot(data[column])
    ax.set_title(f'Boxplot for {column}')
    ax.set_ylabel(column)

# If you have unused subplots, remove the axis labels
for i in range(num_plots, num_rows * num_cols):
    fig.delaxes(axes.flatten()[i])

plt.tight_layout()
plt.show()
```



```
[170]: # Create a list of columns you want to include, excluding "frosh" and "soph"
columns_to_include = [col for col in data.columns if col not in ['frosh',␣
 ↪'soph']]

# Create a list of positions for the box plots
positions = range(1, len(columns_to_include) + 1)

# Create a list of data for the box plots
boxplot_data = [data[col] for col in columns_to_include]

# Create the parallel box plots
plt.figure(figsize=(12, 6))
plt.boxplot(boxplot_data, positions=positions, labels=columns_to_include)
plt.title("Parallel Box Plots")
plt.xlabel("Columns")
plt.ylabel("Values")
plt.xticks(positions, columns_to_include, rotation=45)
plt.grid(axis='y')

plt.show()
```



- Multiple box plots
    - Outliers present:
        * attend, termGPA, priGPA, final, hwrte, stndfnl
        * Attend has the most outliers
        * priGPA has the least outliers
    - Hwrte:
        * Range is very high compared to others

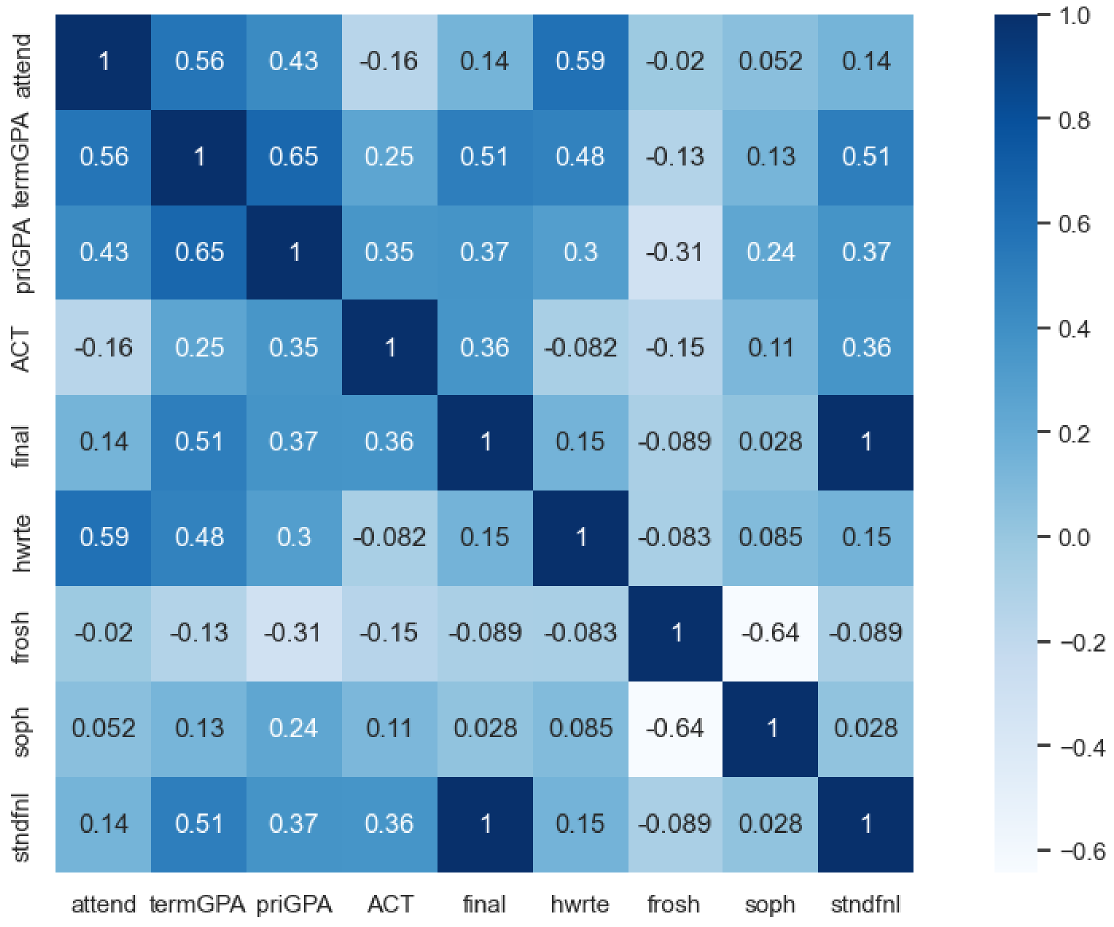
1.5 Correlation Plots

- “Attend” has strongest coorelation with “termGPA” and “hwrte” (homework turned in)
- “Attend” has weakest coorelation with my categorical variables, “frosh” and “soph”, but that is to be expected because they are categorical
- “attend” then has weakest coorelation with “final”

[42]:
```
plt.figure(figsize=(13,7))
data.corr()
c= data.corr()
```

```
sns.heatmap(c,cmap="Blues",square = True, annot = True)
```

```
[42]: <Axes: >
```



	attend	termGPA	priGPA	ACT	final	hwrte	frosh	soph	stndfnl
attend	1	0.56	0.43	-0.16	0.14	0.59	-0.02	0.052	0.14
termGPA	0.56	1	0.65	0.25	0.51	0.48	-0.13	0.13	0.51
priGPA	0.43	0.65	1	0.35	0.37	0.3	-0.31	0.24	0.37
ACT	-0.16	0.25	0.35	1	0.36	-0.082	-0.15	0.11	0.36
final	0.14	0.51	0.37	0.36	1	0.15	-0.089	0.028	1
hwrte	0.59	0.48	0.3	-0.082	0.15	1	-0.083	0.085	0.15
frosh	-0.02	-0.13	-0.31	-0.15	-0.089	-0.083	1	-0.64	-0.089
soph	0.052	0.13	0.24	0.11	0.028	0.085	-0.64	1	0.028
stndfnl	0.14	0.51	0.37	0.36	1	0.15	-0.089	0.028	1

1.6 Exploratory analysis using Pandas Profiling. Did you discover anything new?

- Variable Types:
  - My data has 2 types of variables, numeric and categorical.
- Dataset Statistics:
  - I have 9 variables 8 which are my predictors. I have 680 observations.
- Variable Skew:
  - By expanding upon each variables statistics, we can see their skewness values. A negative skew value, as seen in “attend” for example, is another way we can identify it as leftly skewed. A possitive skew value indicates right skew. Skew values near 0.something such as 0.2 indicate mild skewness.
- The quantile statistics:
  - gives us the numerical values of everything we saw for our variables in their visual

  boxplots.

- Interactions:
  - From interactions, we can see the relationship each predictor has with "attend". The variables did not seem like they had a relationship
- Correlation:
  - I was surprised to see my pandas profiling didn't have a "Correlaton" section, and I think that might have to do with the missing values. Once I impute my missing values, the "Correlation" appears, else it does not.
  - The coorelation is telling us everything my coorelation plot above tells us

[85]:
```
pandas_profiling.ProfileReport(data)
```

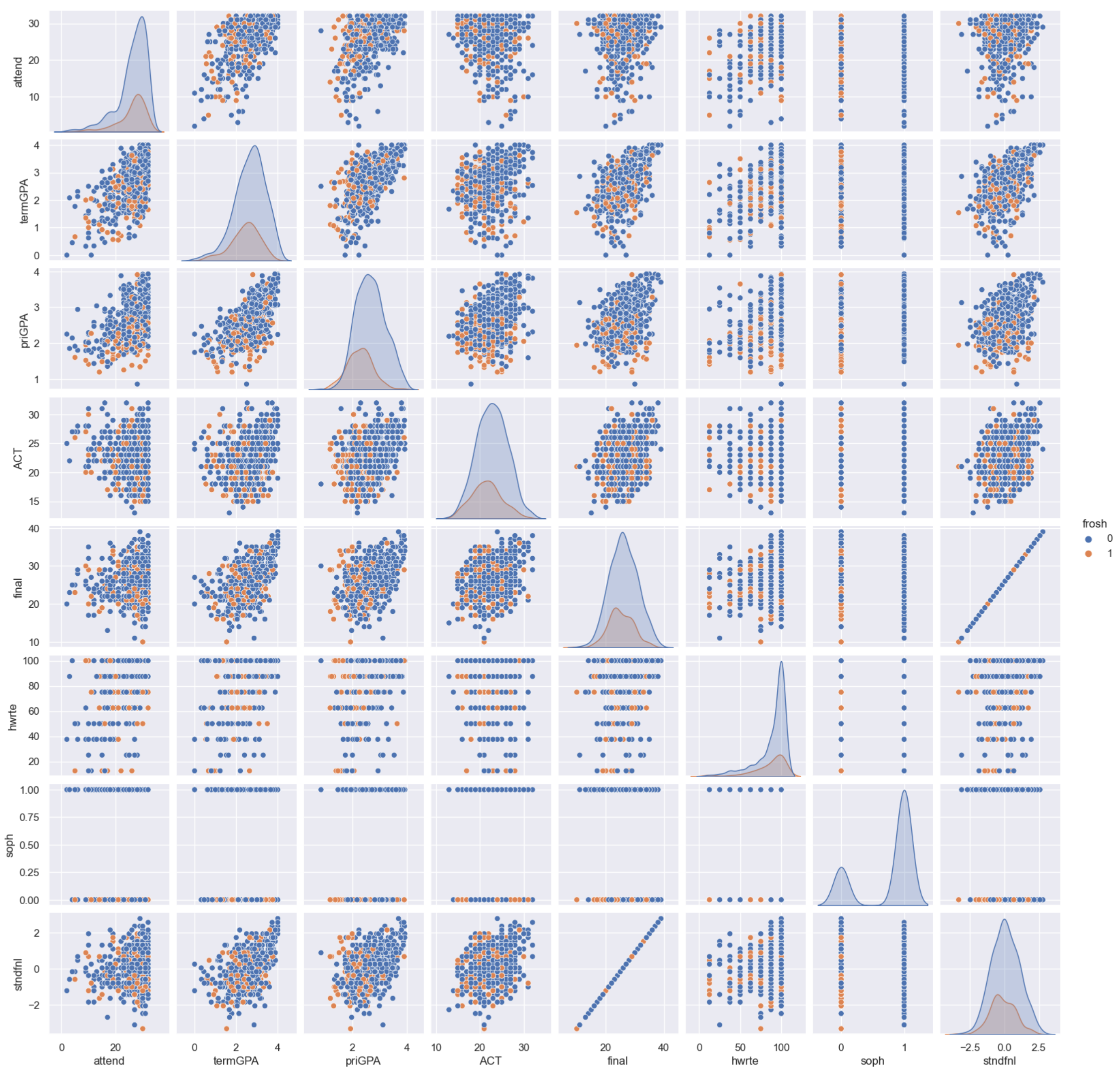
```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

<IPython.core.display.HTML object>
```

[85]:

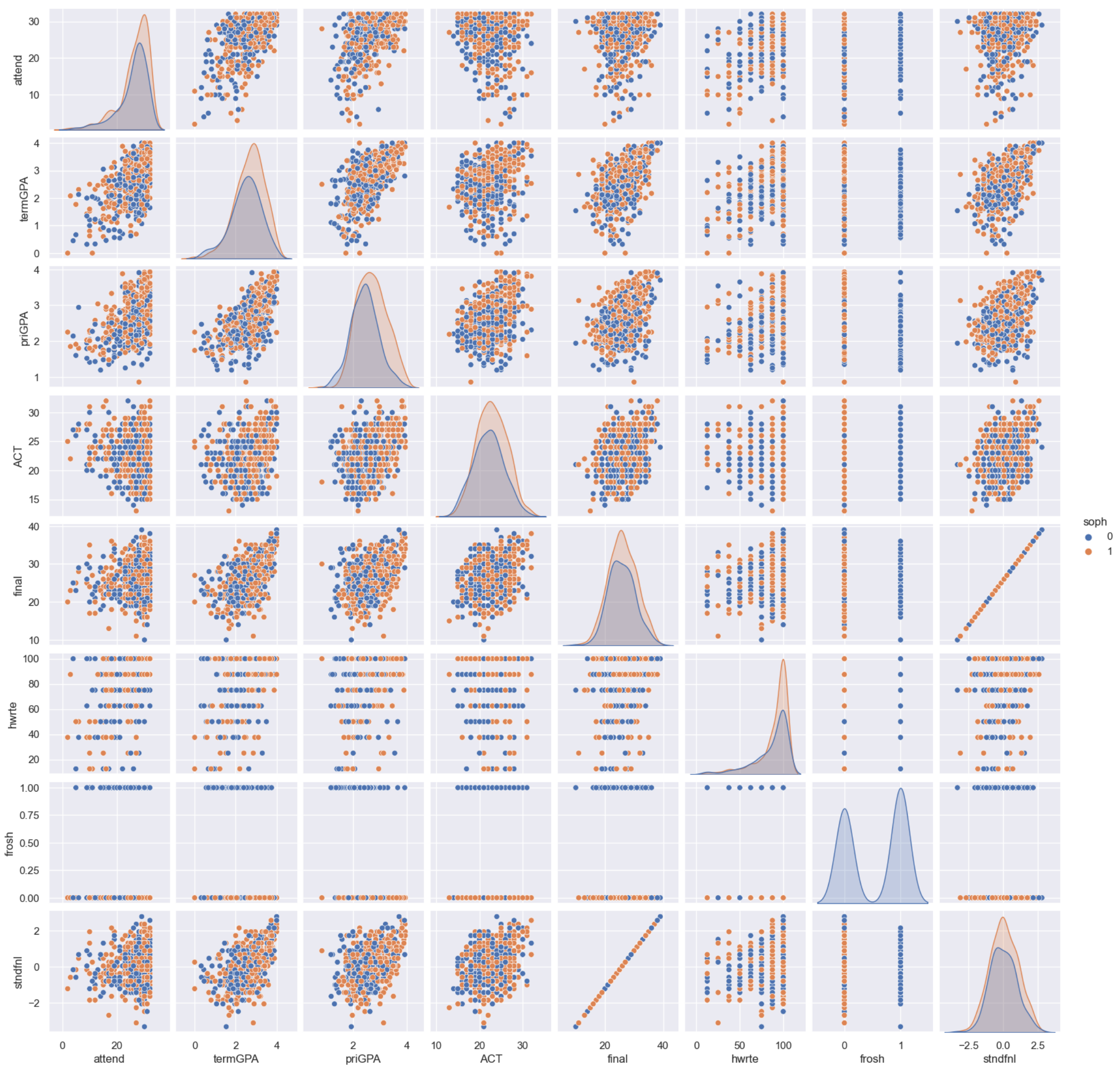
1.7 Estimate density distributions (e.g., Cullen & Frey) for all your variables, and show the plots with the respective fits.

1.8 Pairplot

[45]:
```
sns.set()
sns.pairplot(data, height = 2.0,hue ='frosh')
plt.show()
```



```
[46]: sns.set()
      sns.pairplot(data, height = 2.0,hue ='soph')
      plt.show()
```



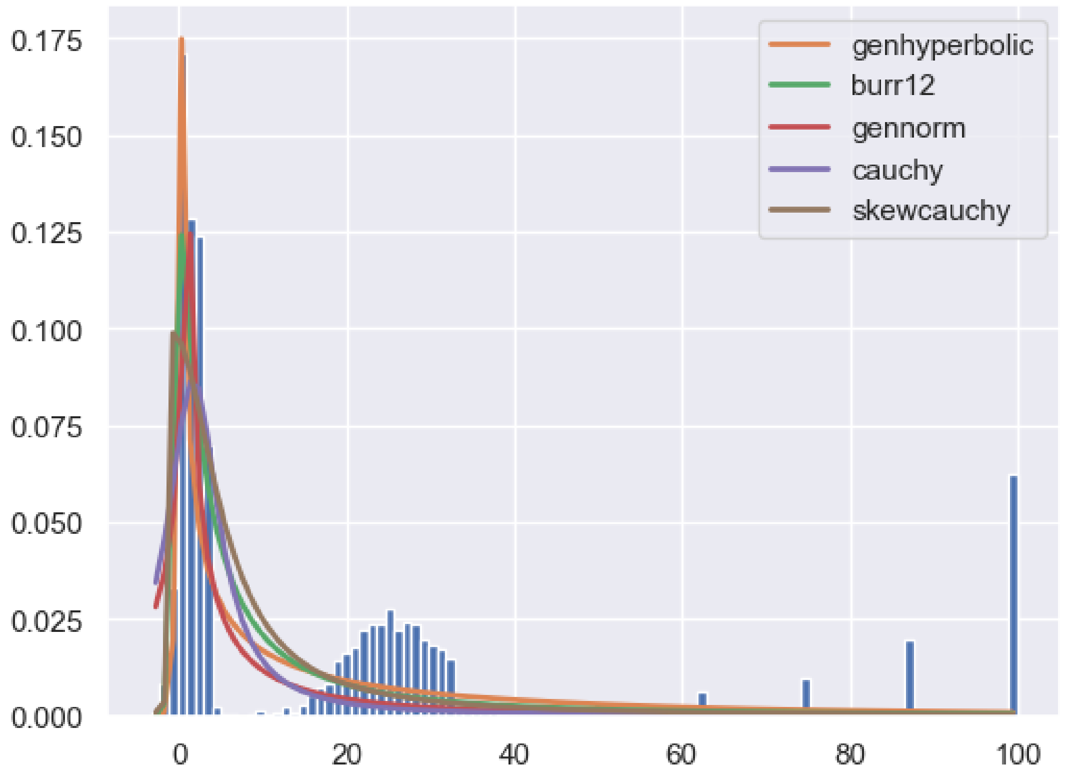
1.9 Cullen Frey

- "genhyperbolic" appears to be the distribution that provides the best fit to the data, with the lowest sumsquare error and the lowest AIC and BIC values.
- The high KS statistic and low p-value suggest a strong fit.

[47]:
```
f = Fitter(data)
f.fit()

f.summary()
```

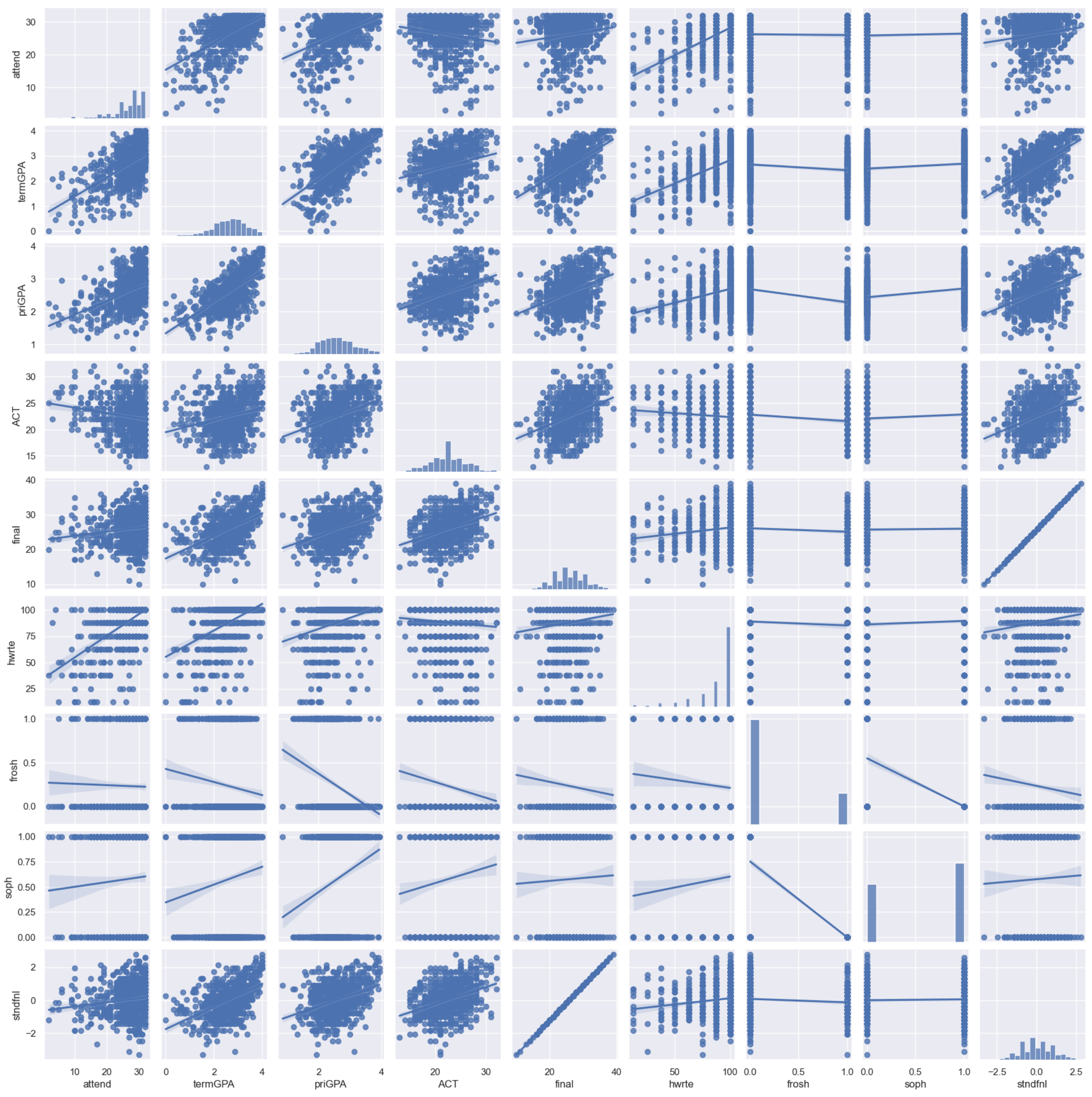
```
SKIPPED _fit distribution (taking more than 30 seconds)
SKIPPED kstwo distribution (taking more than 30 seconds)
SKIPPED johnsonsb distribution (taking more than 30 seconds)
SKIPPED johnsonsu distribution (taking more than 30 seconds)
SKIPPED loguniform distribution (taking more than 30 seconds)
SKIPPED kappa4 distribution (taking more than 30 seconds)
SKIPPED ksone distribution (taking more than 30 seconds)
SKIPPED levy_stable distribution (taking more than 30 seconds)
SKIPPED loggamma distribution (taking more than 30 seconds)
SKIPPED lognorm distribution (taking more than 30 seconds)
SKIPPED mielke distribution (taking more than 30 seconds)
SKIPPED nakagami distribution (taking more than 30 seconds)
SKIPPED ncf distribution (taking more than 30 seconds)
SKIPPED nct distribution (taking more than 30 seconds)
SKIPPED reciprocal distribution (taking more than 30 seconds)
SKIPPED ncx2 distribution (taking more than 30 seconds)
SKIPPED norminvgauss distribution (taking more than 30 seconds)
SKIPPED rv_continuous distribution (taking more than 30 seconds)
SKIPPED rv_histogram distribution (taking more than 30 seconds)
SKIPPED pearson3 distribution (taking more than 30 seconds)
SKIPPED powerlognorm distribution (taking more than 30 seconds)
SKIPPED powernorm distribution (taking more than 30 seconds)
SKIPPED recipinvgauss distribution (taking more than 30 seconds)
SKIPPED rel_breitwigner distribution (taking more than 30 seconds)
SKIPPED skewnorm distribution (taking more than 30 seconds)
SKIPPED studentized_range distribution (taking more than 30 seconds)
SKIPPED t distribution (taking more than 30 seconds)
SKIPPED vonmises_fisher distribution (taking more than 30 seconds)
SKIPPED triang distribution (taking more than 30 seconds)
SKIPPED truncnorm distribution (taking more than 30 seconds)
SKIPPED truncpareto distribution (taking more than 30 seconds)
SKIPPED truncweibull_min distribution (taking more than 30 seconds)
SKIPPED tukeylambda distribution (taking more than 30 seconds)
SKIPPED vonmises_line distribution (taking more than 30 seconds)
SKIPPED weibull_max distribution (taking more than 30 seconds)
SKIPPED weibull_min distribution (taking more than 30 seconds)
SKIPPED wrapcauchy distribution (taking more than 30 seconds)
```

[47]:
```
               sumsquare_error          aic          bic  kl_div  \
genhyperbolic         0.021481  1131.516768  1165.113355     inf
burr12                0.022680  1197.011205  1223.888475     inf
gennorm               0.024891  1321.578093  1341.736046     inf
cauchy                0.032206  1390.476735  1403.915370     inf
skewcauchy            0.032934  1216.903556  1237.061508     inf

               ks_statistic      ks_pvalue
genhyperbolic      0.805603   0.000000e+00
burr12             0.181498  6.919344e-177
gennorm            0.325489   0.000000e+00
cauchy             0.353942   0.000000e+00
skewcauchy         0.223303  1.446781e-268
```



1.10 Identify if there are any non-linearities within your variables. What transformations should you perform to make them linear? What would happen if you included non- linear variables in your regression models without transforming them first

```
[ ]: sns.set()
     sns.pairplot(data, height=2.0, kind="reg")
     plt.show()
```



- Non-linearities identified:
  - ACT
  - Final
  - Hwrte
  - stndfnl
- What transformations to perform:
  - The data has outliers or skewness, which is why Box Cox works best
  - Box Cox helps stablize variance
- What would happen if you included non - linear variables in your regression models without transforming them first:
  - May result in biased and inefficient parameter estimates
  - poor model fit

– Potential violations of key assumptions of linear regression

1.11 Comment on any outliers and/or unusual features of your variables

1.11.1

- Besides my “Act” column, all my boxplots for columns have outliers present.
- The outliers are low values compared to the data set.
- Having outliers on one side indicates skewness.

1.12 If you have any NAs, impute them using any of the methods discussed in class but make sure to justify your choice

We know hwrte is the one with problem, we’re missing 6 of its values. Since we’re only missing a few observations, we can keep the predictor and impute the values that are missing. We can effectively replace the NaN values with the median hwrte since hwrte is leftly skewed.

1.12.1 When to use Mean

We can use mean for imputing missing data when the distribution is symmetrical. If the data is skewed, outlier points will have an impact on mean, which is when we don’t want to use mean to impute missing values.

1.12.2 When to use Median

When data is skewed, it is a better idea to use median

2 Variable Selection:

2.1 Using the Boruta Algorithm identify the top 2 predictors

[27]:
```
x = data.iloc[:, 1:]
y = data['attend']

xCols = x.columns.tolist()


currentTrainX = x.to_numpy()
currentTrainY = y.to_numpy().ravel()
```

[28]:
```
forest = RandomForestRegressor(n_jobs=-1, max_depth = 5)
forest.fit(currentTrainX, currentTrainY)
```

[28]:
```
RandomForestRegressor(max_depth=5, n_jobs=-1)
```

[29]:
```
np.int = int
np.float = float
np.bool = bool
boruta = BorutaPy(forest, n_estimators='auto', verbose=2, random_state=1)
```

```
boruta.fit(currentTrainX, currentTrainY)
```

```
Iteration: 	1 / 100
Confirmed: 	0
Tentative: 	8
Rejected: 	0
Iteration: 	2 / 100
Confirmed: 	0
Tentative: 	8
Rejected: 	0
Iteration: 	3 / 100
Confirmed: 	0
Tentative: 	8
Rejected: 	0
Iteration: 	4 / 100
Confirmed: 	0
Tentative: 	8
Rejected: 	0
Iteration: 	5 / 100
Confirmed: 	0
Tentative: 	8
Rejected: 	0
Iteration: 	6 / 100
Confirmed: 	0
Tentative: 	8
Rejected: 	0
Iteration: 	7 / 100
Confirmed: 	0
Tentative: 	8
Rejected: 	0
Iteration: 	8 / 100
Confirmed: 	4
Tentative: 	0
Rejected: 	4


BorutaPy finished running.

Iteration: 	9 / 100
Confirmed: 	4
Tentative: 	0
Rejected: 	4
```

```
[29]: BorutaPy(estimator=RandomForestRegressor(max_depth=5, n_estimators=80,
                                              n_jobs=-1,
                                              random_state=RandomState(MT19937) at
       0x1400AFB40),
         n_estimators='auto', random_state=RandomState(MT19937) at 0x1400AFB40,
         verbose=2)
```

```
[30]: featureSupport = list((zip(xCols, boruta.support_)))
      featureSupport
```

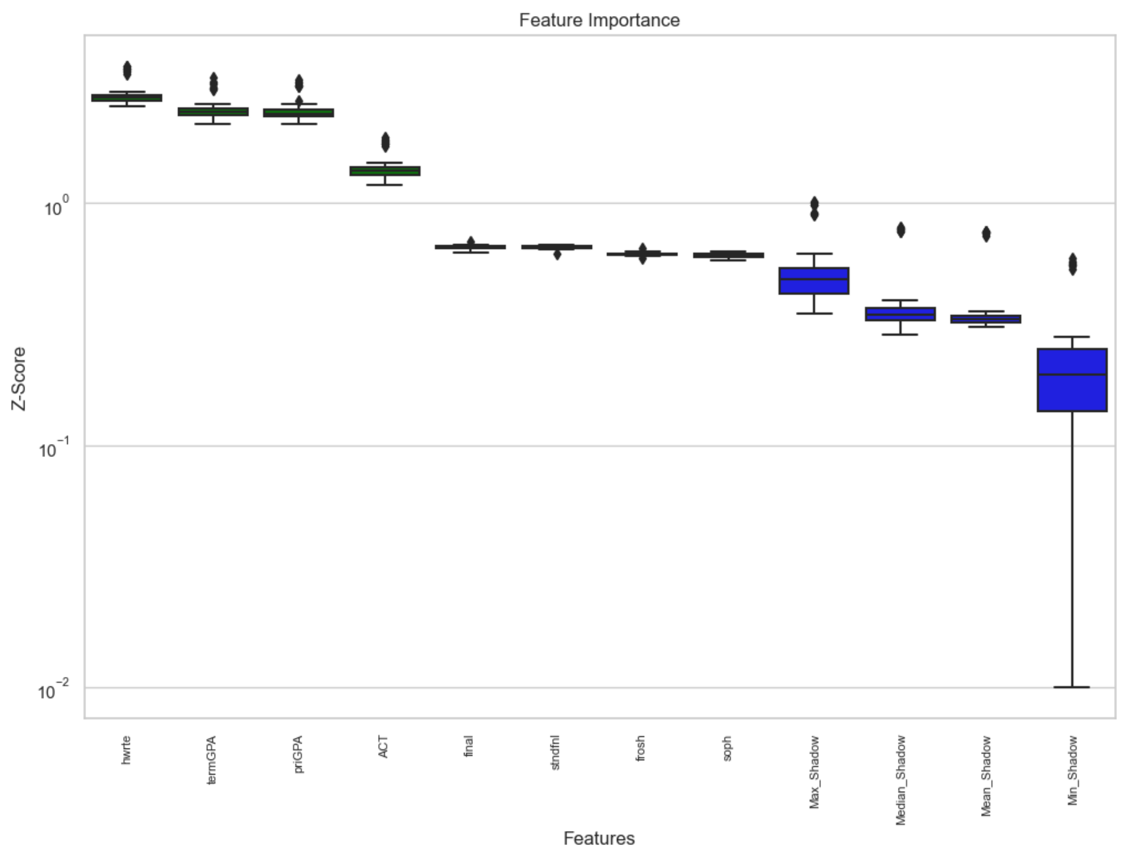
```
[30]: [('termGPA', True),
       ('priGPA', True),
       ('ACT', True),
       ('final', False),
       ('hwrte', True),
       ('frosh', False),
       ('soph', False),
       ('stndfnl', False)]
```

```
[31]: featureRanks = list(zip(xCols, boruta.ranking_))
      sorted(featureRanks, key=lambda x: x[1])
```

```
[31]: [('termGPA', 1),
       ('priGPA', 1),
       ('ACT', 1),
       ('hwrte', 1),
       ('stndfnl', 2),
       ('final', 3),
       ('soph', 4),
       ('frosh', 5)]
```

```
[32]: Feature_Selector = BorutaShap(importance_measure='shap', classification=False)
      Feature_Selector.fit(X=x, y=y, n_trials=50, random_state=0)
      Feature_Selector.plot(which_features='all')
```

```
  0%|          | 0/50 [00:00<?, ?it/s]

4 attributes confirmed important: ['priGPA', 'termGPA', 'hwrte', 'ACT']
4 attributes confirmed unimportant: ['frosh', 'soph', 'stndfnl', 'final']
0 tentative attributes remains: []
```



- Top 2 Predictors Boruta Algorithm:
  - Hwrte
  - TermGPA

2.2 Using Mallows Cp identify the top 2 predictors

[20]:
```
subdat = data[['attend', 'ACT', 'hwrte', 'final', 'termGPA', 'stndfnl',␣
    ↪'priGPA', 'soph', 'frosh']].copy()
```

[21]:
```
import itertools

# get the base model, y and its fitted values
model = smf.ols(formula='attend ~ ACT + hwrte + final + termGPA + stndfnl +␣
    ↪priGPA + soph + frosh', data=data)
results = model.fit()
y = data['attend']
y_pred=results.fittedvalues


storage_cp = pd.DataFrame(columns = ["Variables", "CP"])
```

```
k = 9 # number of parameters in orginal model (includes y-intercept)

for L in range(1, len(subdat.columns[1:]) + 1):
    for subset in itertools.combinations(subdat.columns[1:], L):

        # join the strings in the data together
        formula1 = 'attend~'+'+'.join(subset)

        # get the cp
        results = smf.ols(formula=formula1, data = data).fit()
        y_sub = results.fittedvalues
        p = len(subset)+1 # number of parameters in the subset model (includes␣
 ↪y-intercept)

        cp = mallow.mallow(y, y_pred,y_sub, k, p)

        # add to the dataframe
        #storage_cp = storage_cp.append({'Variables': subset, 'CP': cp},␣
 ↪ignore_index = True)
        storage_cp = pd.concat([storage_cp, pd.DataFrame({'Variables':␣
 ↪[subset], 'CP': [cp]})], ignore_index=True)
```

[35]:
```
storage_cp = storage_cp.sort_values(by = "CP")
storage_cp
```

[35]:
```
                                              Variables          CP
229      (ACT, hwrte, termGPA, stndfnl, priGPA, frosh)    5.213939
222        (ACT, hwrte, final, termGPA, priGPA, frosh)    5.213939
247  (ACT, hwrte, final, termGPA, stndfnl, priGPA, …    7.012357
251  (ACT, hwrte, termGPA, stndfnl, priGPA, soph, f…    7.201677
249  (ACT, hwrte, final, termGPA, priGPA, soph, frosh)    7.201678
..                                                 …         …
32                                  (stndfnl, frosh)  721.679701
78                           (final, stndfnl, frosh)  722.500027
6                                           (soph,)  743.819676
35                                     (soph, frosh)  745.380811
7                                          (frosh,)  747.111627

[255 rows x 2 columns]
```

[36]:
```
predictors_with_2_variables = storage_cp[storage_cp['Variables'].apply(len) ==␣
 ↪2]
print(predictors_with_2_variables)
```

```
             Variables          CP
16    (hwrte, termGPA)  117.356246
```

```
18     (hwrte, priGPA)  160.077518
10      (ACT, termGPA)  172.709558
8         (ACT, hwrte)  242.226420
17    (hwrte, stndfnl)  254.795688
15      (hwrte, final)  254.795688
20      (hwrte, frosh)  257.872785
19       (hwrte, soph)  259.040281
26  (termGPA, stndfnl)  262.637147
21    (final, termGPA)  262.637148
27   (termGPA, priGPA)  294.161087
29    (termGPA, frosh)  299.463709
28     (termGPA, soph)  302.870888
12       (ACT, priGPA)  335.983827
34     (priGPA, frosh)  470.197377
33      (priGPA, soph)  485.948205
30   (stndfnl, priGPA)  489.435884
23     (final, priGPA)  489.435884
9         (ACT, final)  651.620596
11      (ACT, stndfnl)  651.620596
13         (ACT, soph)  707.972944
14        (ACT, frosh)  712.128370
24       (final, soph)  718.451853
31     (stndfnl, soph)  718.451853
22    (final, stndfnl)  720.578780
25      (final, frosh)  721.679701
32    (stndfnl, frosh)  721.679701
35       (soph, frosh)  745.380811
```

- Top 2 Predictors Mallow_Cp:
  - Hwrte
  - TermGPA

3 Model Building: Explore several competing OLS models (based on part 2) and decide on one model only (with just one predictor). You will need to explain in detail how you arrived at your preferred model. Discuss the economic significance of your parameters, and overall findings. Make sure you discuss your main conclusions and recommendations

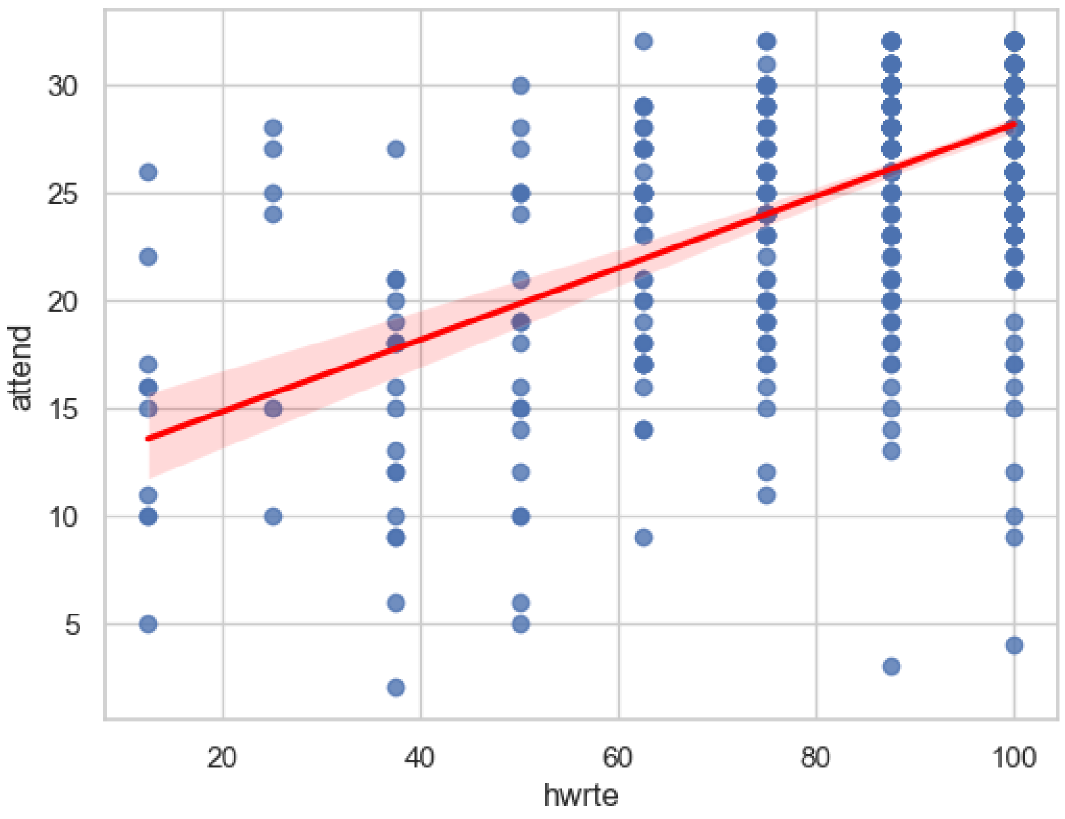
3.1 OLS Models

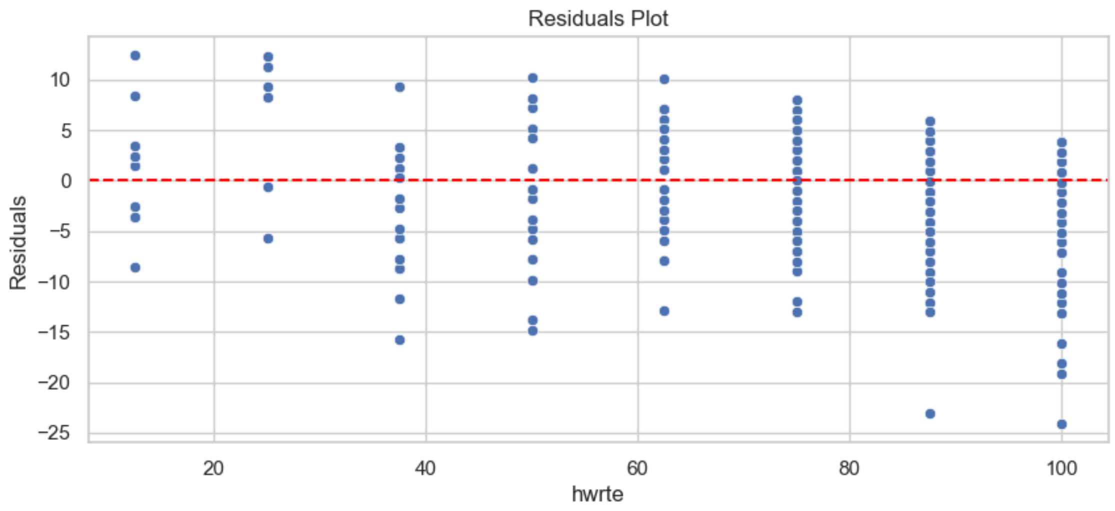
[67]:
```python
topPredictors = ["hwrte", "termGPA"]

for column in topPredictors:

    print("Attend and " + column)
```

```python
formula = f'attend ~ {column}'
ols_mod = smf.ols(formula=formula, data=data)


ols_fit = ols_mod.fit()


print("\n")
print(ols_fit.params)
print("\n")


print(ols_fit.summary())

# Linearity: Harvey-Collier --> Ho: model is linear
name = ["t-stat", "p-value"]
test = sms.linear_harvey_collier(ols_fit)
print("\nLinearity Test Results:",['bold'])
print(list(zip(name, test)))
print("\n")

# Normaility of the Residuals: Jarque-Bera --> Residuals ~ N(0,1)
name = ["Jarque-Bera", "Chi^2 two-tail prob.", "Skew", "Kurtosis"]
test = sms.jarque_bera(ols_fit.resid)
print("JB Results:",['bold'])
print(list(zip(name, test)))
print("\n")

# Heteroskedasticity: Breush-Pagan --> Ho: var = constant
name = ["Lagrange multiplier statistic", "p-value", "f-value", "f p-value"]
test = sms.het_breuschpagan(ols_fit.resid, ols_fit.model.exog)
print("BP Results:",['bold'])
print(list(zip(name, test)))

sns.regplot(x=column, y='attend', data=data, line_kws={'color': 'red'})

plt.figure(figsize = (10, 4))


sns.scatterplot(x = data[column], y = ols_fit.resid)
plt.axhline(0, linestyle = '--', color = "red")
plt.ylabel("Residuals")
plt.title("Residuals Plot")
plt.show()
```

```
Attend and hwrte
```

```
Intercept    11.480350
hwrte         0.166639
dtype: float64


                            OLS Regression Results
==============================================================================
Dep. Variable:                 attend   R-squared:                       0.345
Model:                            OLS   Adj. R-squared:                  0.344
Method:                 Least Squares   F-statistic:                     356.5
Date:                Wed, 25 Oct 2023   Prob (F-statistic):           3.21e-64
Time:                        14:02:37   Log-Likelihood:                -1974.4
No. Observations:                 680   AIC:                             3953.
Df Residuals:                     678   BIC:                             3962.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     11.4803      0.795     14.440      0.000       9.919      13.041
hwrte          0.1666      0.009     18.882      0.000       0.149       0.184
==============================================================================
Omnibus:                      175.999   Durbin-Watson:                   1.806
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              539.294
Skew:                          -1.242   Prob(JB):                   7.83e-118
Kurtosis:                       6.587   Cond. No.                         423.
==============================================================================

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

Linearity Test Results: ['bold']
[('t-stat', -2.206065036659753), ('p-value', 0.027715064135421446)]


JB Results: ['bold']
[('Jarque-Bera', 539.2944865503282), ('Chi^2 two-tail prob.',
7.828709629399452e-118), ('Skew', -1.2417037271995144), ('Kurtosis',
6.58700578061324)]


BP Results: ['bold']
[('Lagrange multiplier statistic', 28.374019495738874), ('p-value',
9.999834011712986e-08), ('f-value', 29.52243433146104), ('f p-value',
7.714229484970389e-08)]
```

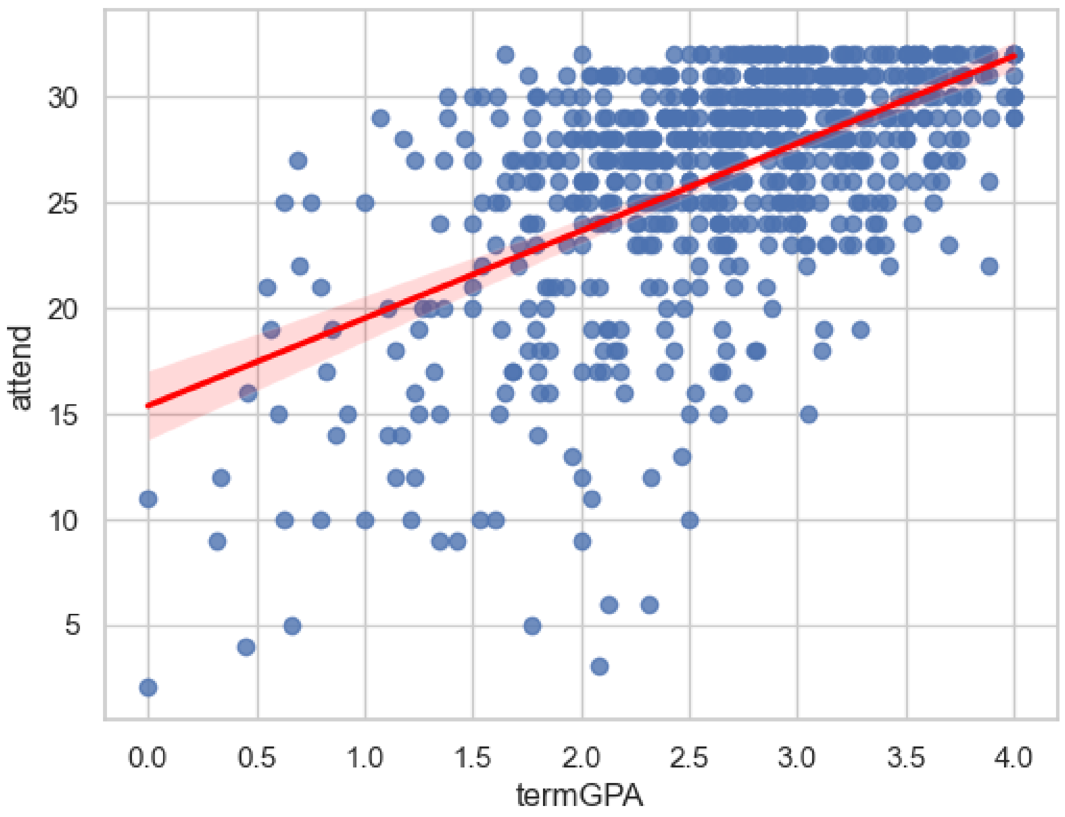


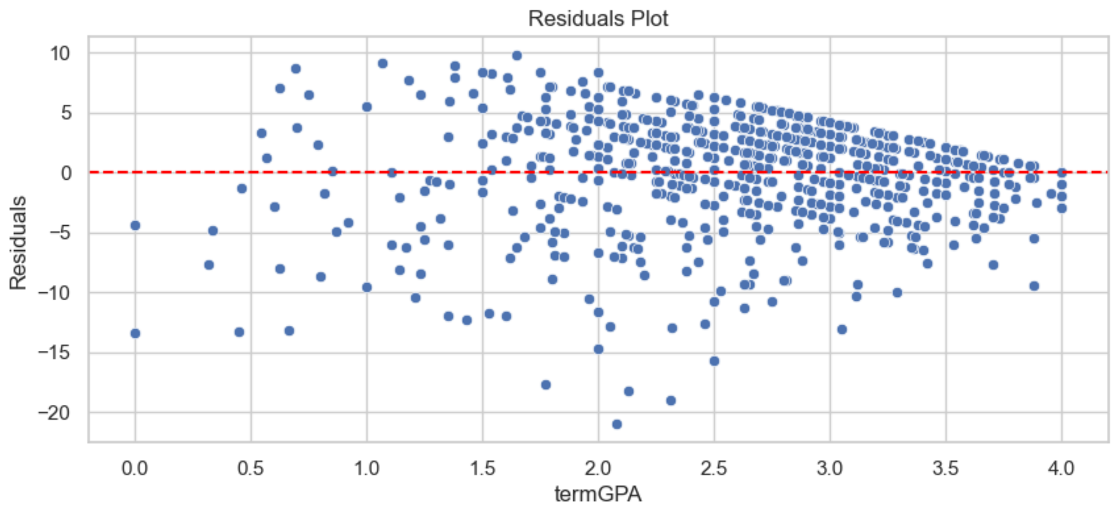


```
Attend and termGPA
```

```
Intercept    15.364338
```

```
termGPA      4.145606
dtype: float64


                            OLS Regression Results
==============================================================================
Dep. Variable:                 attend   R-squared:                       0.313
Model:                            OLS   Adj. R-squared:                  0.312
Method:                 Least Squares   F-statistic:                     309.4
Date:                Wed, 25 Oct 2023   Prob (F-statistic):           2.46e-57
Time:                        14:02:38   Log-Likelihood:                -1990.2
No. Observations:                 680   AIC:                             3984.
Df Residuals:                     678   BIC:                             3993.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     15.3643      0.637     24.117      0.000      14.113      16.615
termGPA        4.1456      0.236     17.590      0.000       3.683       4.608
==============================================================================
Omnibus:                      124.467   Durbin-Watson:                   1.899
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              230.917
Skew:                          -1.071   Prob(JB):                     7.19e-51
Kurtosis:                       4.888   Cond. No.                         11.2
==============================================================================

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

Linearity Test Results: ['bold']
[('t-stat', -2.1125043720154832), ('p-value', 0.035009291063449854)]


JB Results: ['bold']
[('Jarque-Bera', 230.91743580701535), ('Chi^2 two-tail prob.',
7.193097165882894e-51), ('Skew', -1.070602373001996), ('Kurtosis',
4.888192093463992)]


BP Results: ['bold']
[('Lagrange multiplier statistic', 57.05879707677701), ('p-value',
4.229751237887638e-14), ('f-value', 62.10195157507154), ('f p-value',
1.3018940510334511e-14)]
```





- Prefered Model: TermGPA
  - R- Squared:
    - A higher R - squared indicates a better fit for the model
    - Hwrte model has a higher R^2

    - JB Test:
        - The JB test assesses the normality of residuals.
        - Lower JB is preferred because that indicates residuals are closer to normal distribution
        - TermGPA model has a lower JB
    - Breusch-Pagan (BP) Test:
        - The BP test assesses heteroskedasticity
        - TemrmGPA has lower evidence of heteroskedasticity

We prefer TermGPA model with Attend because: - better normality of residuals - better heteroskedasticity

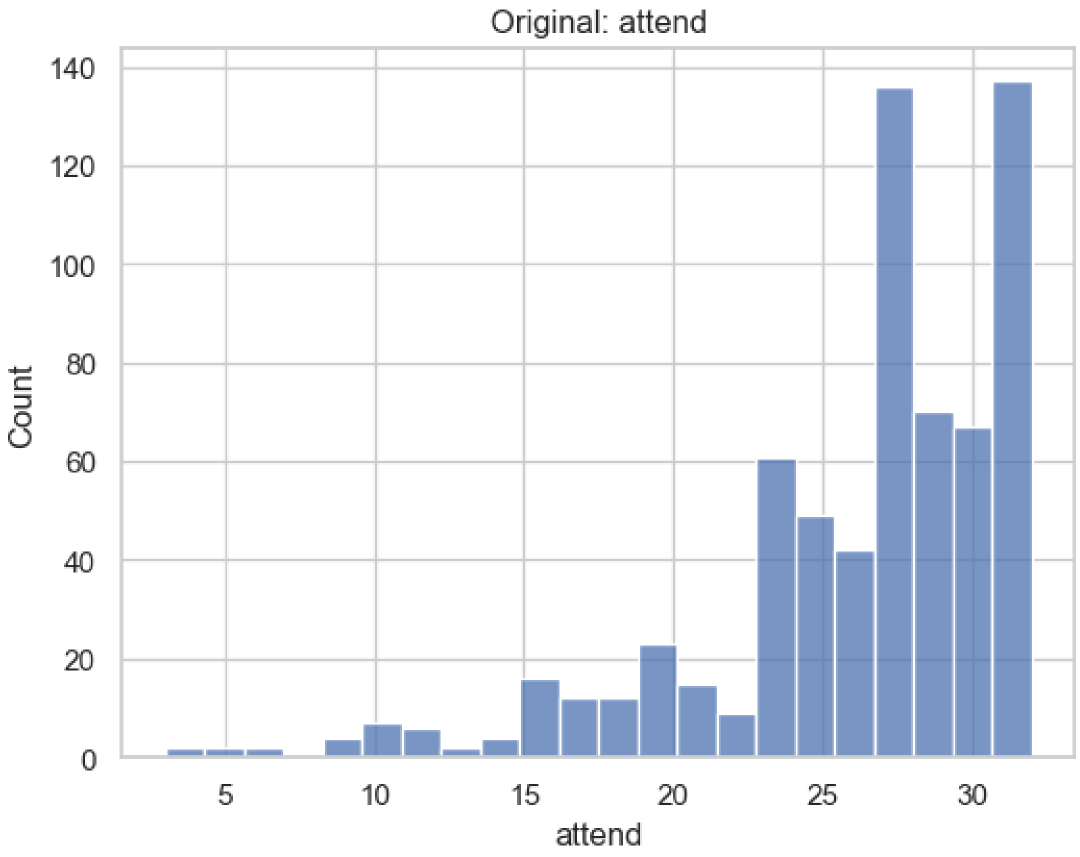
3.2 Evaluate transformations of variables

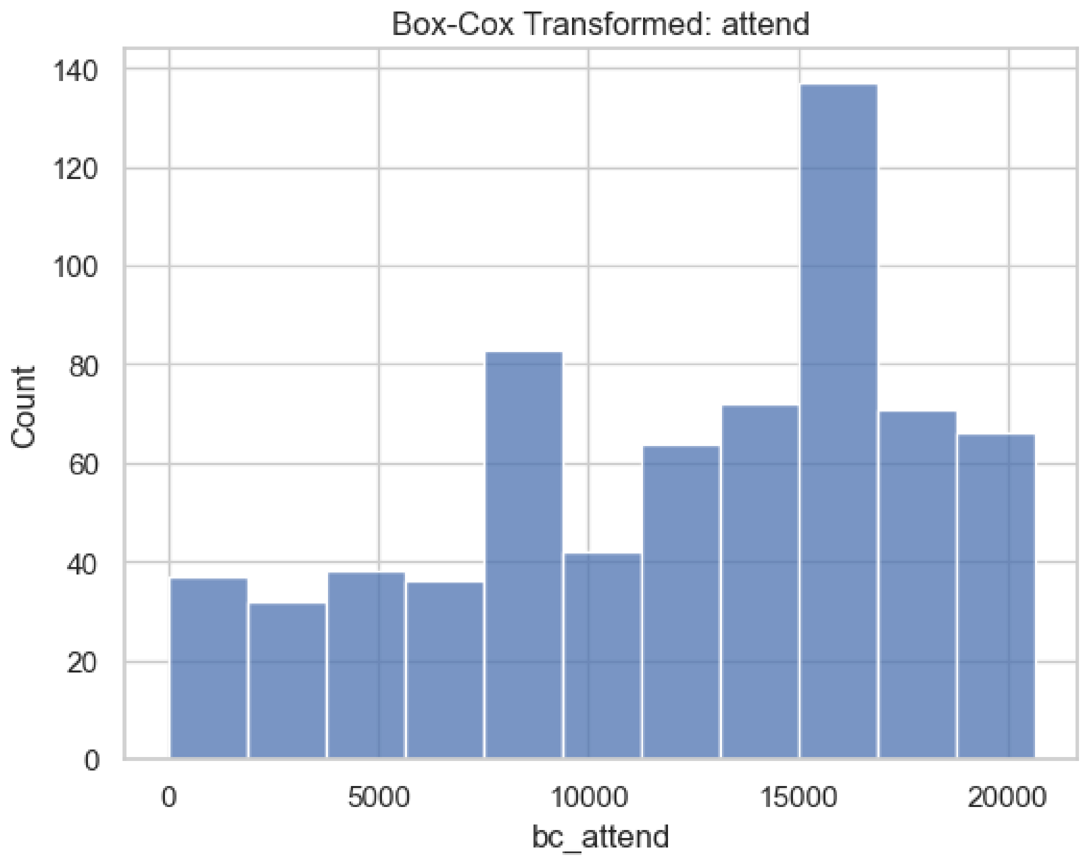
- Transformation Evaluation:
    - Attend:
        - Lambda value = 3.2
        - The data was initially right skewed
        - Attend was more rightly skewed than termGPA
        - box-cox transformed it into smaller values so it would fit better in the model
    - TermGPA:
        - Lambda value = 1.47
        - Made my values smaller.
        - The data was initially right skewed
        - box-cox transformed them into smaller values so it would fit better in the model

[171]:
```
data = data[data["termGPA"] != 0]
```

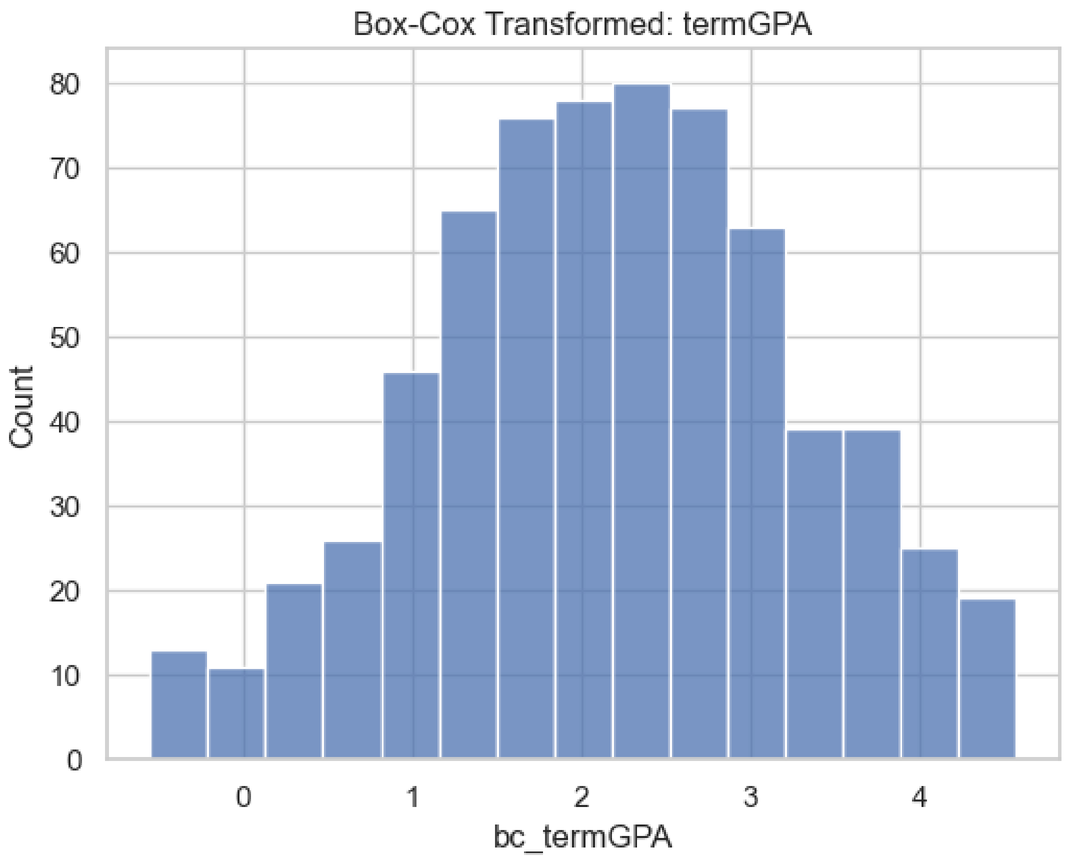
[172]:
```
import scipy
data["bc_attend"],data["lambda_attend"] = scipy.stats.boxcox(data["attend"])
print("lambda_attend: " + str(data['lambda_attend'].iloc[0]))

sns.histplot(data["attend"])
plt.title("Original: attend")
plt.show()

sns.histplot(data["bc_attend"])
plt.title("Box-Cox Transformed: attend")
plt.show()

data['bc_termGPA'],data['lambda_termGPA'] = scipy.stats.boxcox(data["termGPA"])
print("lambda_termGPA: " + str(data['lambda_termGPA'].iloc[0]))

sns.histplot(data['bc_termGPA'])
plt.title("Box-Cox Transformed: termGPA")
plt.show()
```

```
lambda_attend: 3.2022175172423966
```


Original: attend
Count
attend
0
20
40
60
80
100
120
140
5
10
15
20
25
30

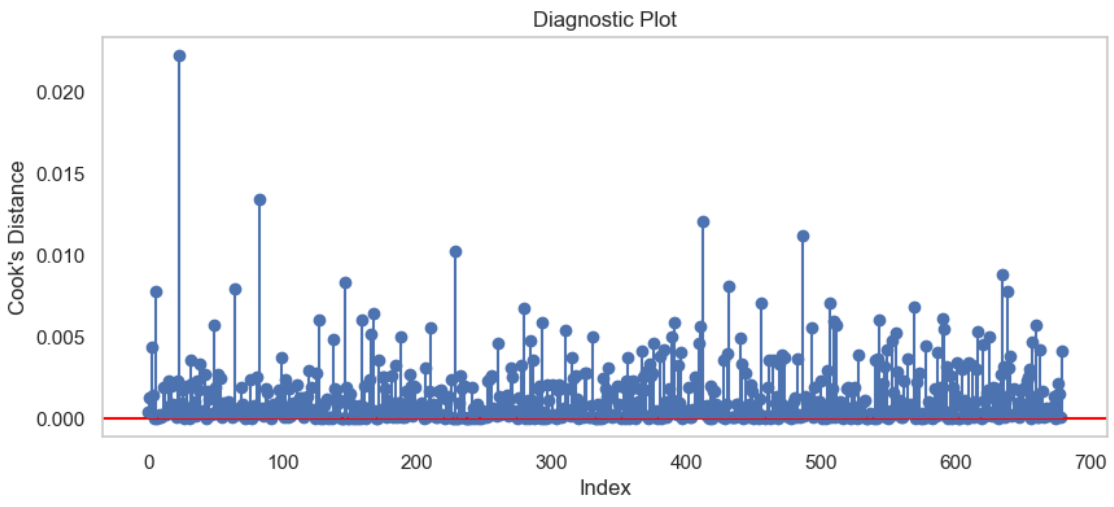


```
lambda_termGPA: 1.4782342808462594
```



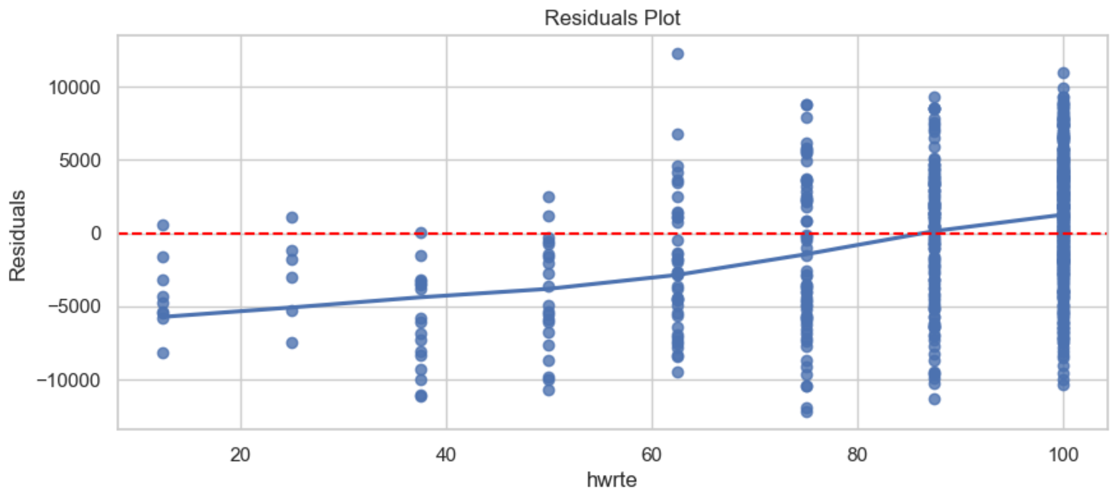
3.3 Look at Cook's distance Plot

[173]:
```
formula = f'bc_attend ~ bc_termGPA'
ols_mod = smf.ols(formula=formula, data=data)

ols_fit = ols_mod.fit()

cooks_distance = ols_fit.get_influence().cooks_distance

plt.figure(figsize = (10, 4))
plt.scatter(data.index, cooks_distance[0])
plt.axhline(0, color = 'red')
plt.vlines(x = data.index, ymin = 0, ymax = cooks_distance[0])
plt.xlabel('Index')
plt.ylabel('Cook\'s Distance')
plt.title("Diagnostic Plot")
plt.grid()
```



3.4 Residuals Plot

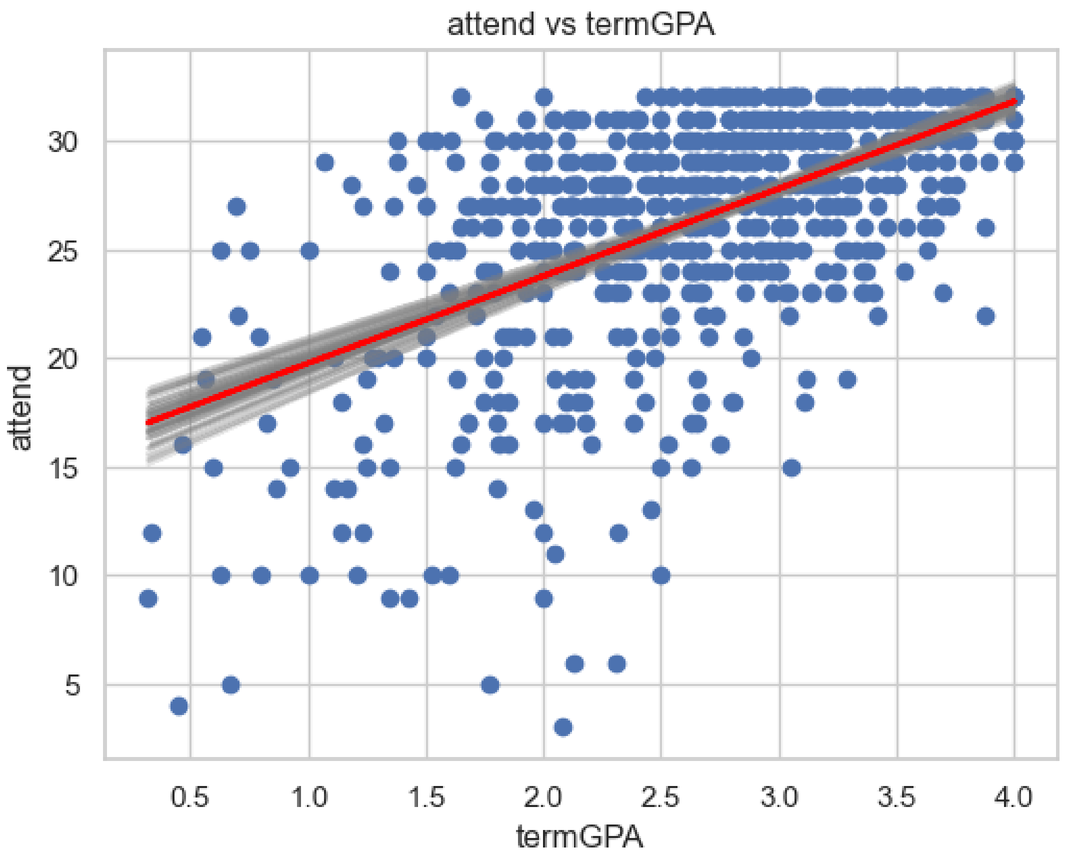
[174]:
```
formula = f'bc_attend ~ bc_termGPA'
ols_mod = smf.ols(formula=formula, data=data)

ols_fit = ols_mod.fit()


plt.figure(figsize = (10, 4))
sns.regplot(x = data.hwrte, y = ols_fit.resid, lowess = True)
plt.axhline(0, linestyle = '--', color = "red")
plt.ylabel("Residuals")
plt.title("Residuals Plot")
plt.show()
```



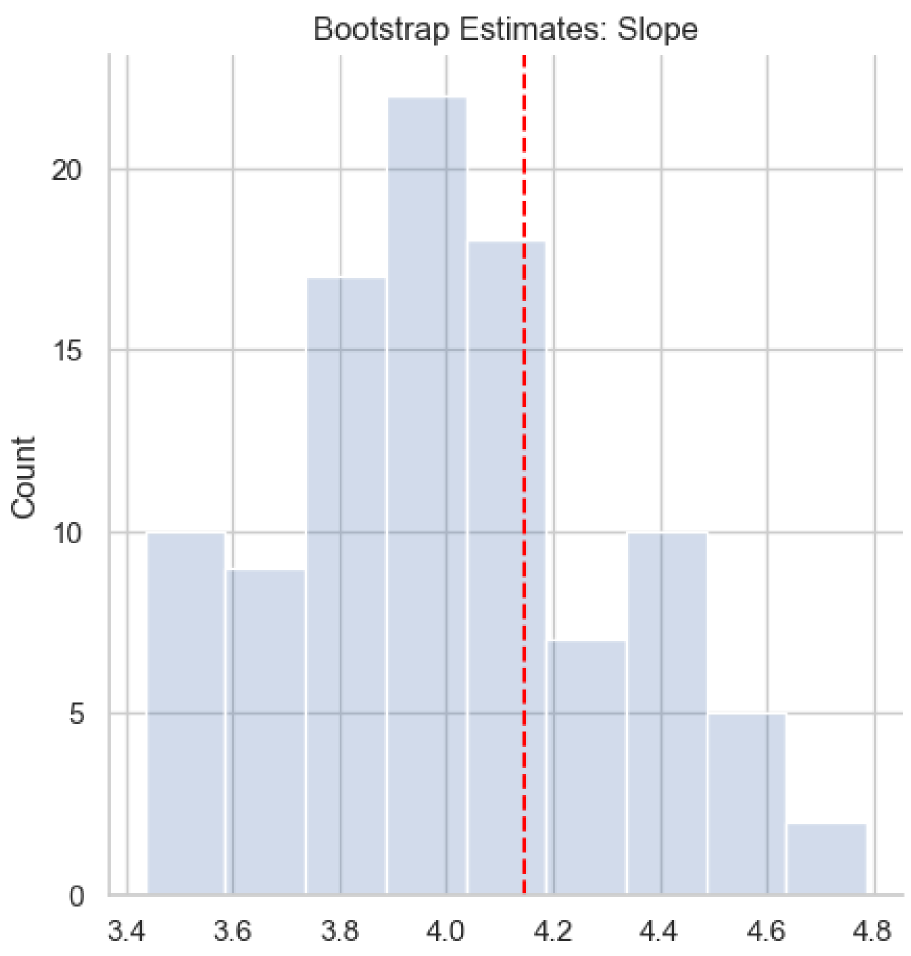
3.5 Evaluate the robustness of your coefficient estimates by bootstrapping your model. Provide a histogram of the bootstrapped estimates (including R2), and comment on the findings. In particular how do these estimates compare against your LS estimates?

3.6 Boostrapping

[205]:
```
ols_mod = sm.OLS(data['attend'], sm.add_constant(data['termGPA'])).fit()


boot_slopes = []
boot_interc = []
boot_adjR2 = []
n_boots = 100
n_points = data.shape[0]

plt.figure()

for _ in range(n_boots):

    sample_df = data.sample(n=n_points, replace=True)


    ols_model_temp = sm.OLS(sample_df['attend'], sm.
 ↪add_constant(sample_df['termGPA']))
    results_temp = ols_model_temp.fit()


    boot_interc.append(results_temp.params[0])
    boot_slopes.append(results_temp.params[1])
    boot_adjR2.append(results_temp.rsquared_adj)


    y_pred_temp = results_temp.predict(sm.add_constant(sample_df['termGPA']))
    plt.plot(sample_df['termGPA'], y_pred_temp, color='grey', alpha=0.2)


y_pred = ols_mod.predict(sm.add_constant(data['termGPA']))
plt.scatter(data['termGPA'], data['attend'])
plt.plot(data['termGPA'], y_pred, linewidth=2, color='red')
plt.grid(True)
plt.xlabel('termGPA')
plt.ylabel('attend')
plt.title('attend vs termGPA')
plt.show()
```

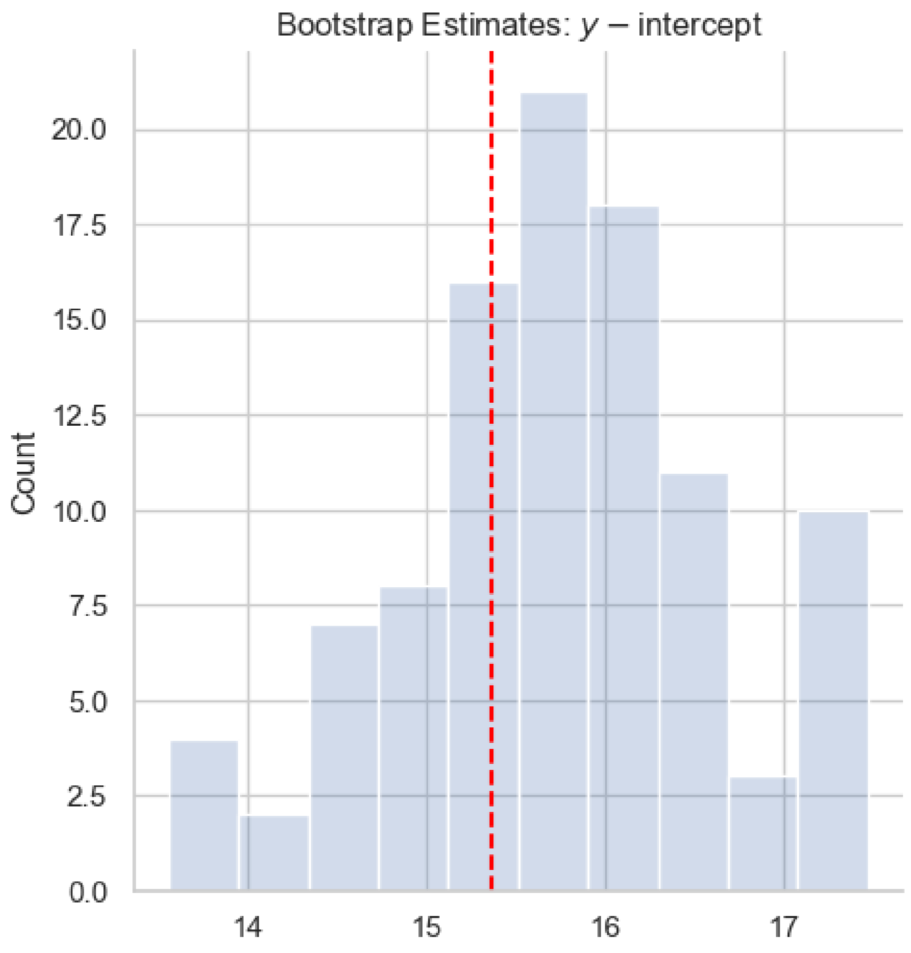


[209]:
```
sns.displot(boot_slopes, alpha = 0.25)
plt.axvline(x=4.1456,color='red', linestyle='--')
plt.title('Bootstrap Estimates: Slope')
plt.show()
```



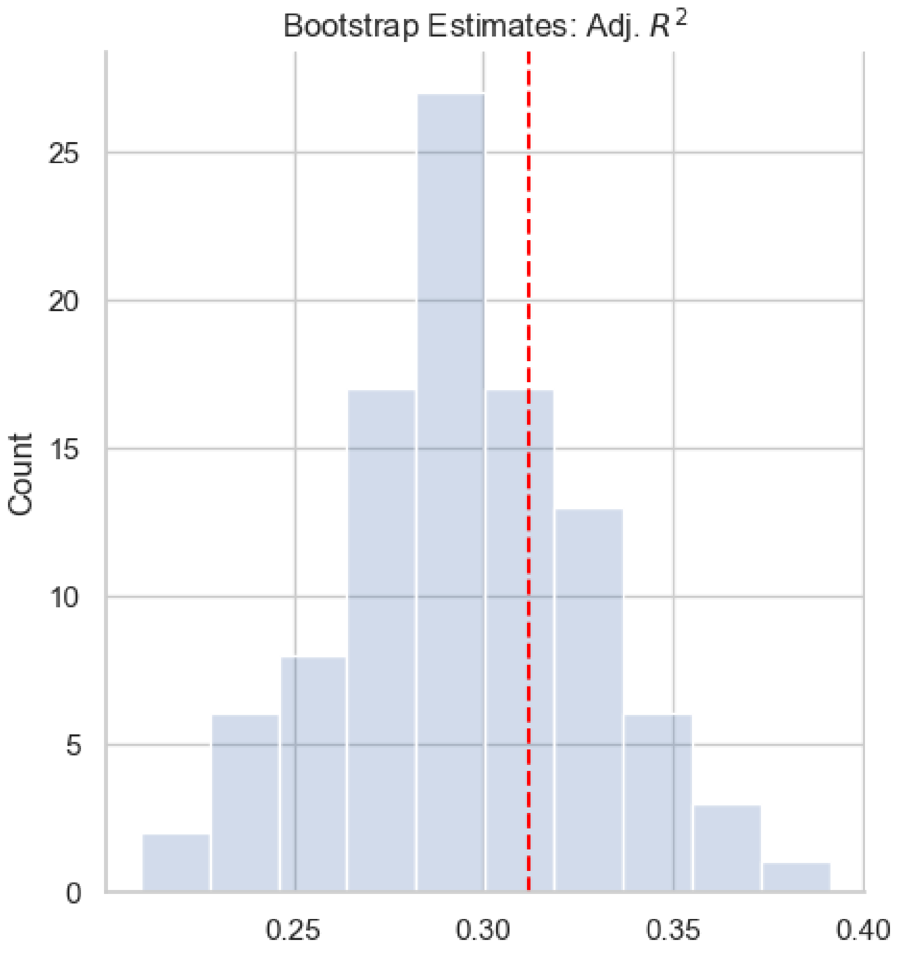
- Evaluate the robustness of your coefficient estimates by bootstrapping your model: Slope
  - Bootstrap Range: about 3.45 to about 4.75
  - LS line = 4.14
  - The model is robust because the range falls within the LS line

[208]:
```
sns.displot(boot_interc, alpha = 0.25)
plt.axvline(x=15.3643,color='red', linestyle='--')
plt.title('Bootstrap Estimates: $y-$intercept')
plt.show()
```



- Evaluate the robustness of your coefficient estimates by bootstrapping your model: Y intercept
  - Bootstrap Range: about 12 to about 19
  - LS line = 15.3643
  - The model is robust because the range falls within the LS line

[213]:
```
sns.displot(boot_adjR2, alpha = 0.25)
plt.axvline(x=0.312,color='red', linestyle='--')
plt.title('Bootstrap Estimates: Adj. $R^2$')
plt.show()
```



- Evaluate the robustness of your coefficient estimates by bootstrapping your model: Adj. Rsquared
  - Bootstrap Range: about 0.1 to about 0.35
  - LS line = 0.312
  - The model is robust because the range falls within the LS line

3.7 Use cross-validation to evaluate your model's performance

[212]:
```
x = data[['termGPA']]
y = data[['attend']]

regr = LinearRegression()
model = regr.fit(x,y)
regr.coef_
regr.intercept_
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,␣
 ↪random_state=0)


regr = LinearRegression()
regr.fit(x_train, y_train)


y_pred = regr.predict(x_test)



print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))



regr = linear_model.LinearRegression()
scores = cross_val_score(regr, x, y, cv=5,␣
 ↪scoring='neg_root_mean_squared_error')
print('5-Fold CV RMSE Scores:', scores)
```

```
MAE: 3.7309948840573575
MSE: 27.900479733846872
RMSE: 5.282090470055096
5-Fold CV RMSE Scores: [-3.8197452  -4.30131043 -4.77038373 -4.88245911
-4.73727784]
```

- Discuss the economic significance of your parameters, and overall findings.
  - According to the OLS model, every per unit increase in attendance is attributed by 4.15 units of term GPA
  - It has a positive relationship, an increase in attend leads to an increase in term GPA
- Make sure you discuss your main conclusions:
  - MAE = 3.74, model's predictions have an absolute error of 3.73
  - MSE = 27.9, the average squared error
  - RMSE = 5.28, model's predictions are off by 5.28 units
  - Model is overfitting
    - Because RMSE scores are all negative, which is unusual
    - Even though the cross-validated RMSE scores are negative, the fact that they are significantly lower than your training RMSE (5.28) indicates overfitting.
    - Overfit models tend to perform exceptionally well on the training data but poorly on unseen data, leading to a high divergence between training and validation performance.
    - Training RMSE: An RMSE of 5.28 suggests that your model's predictions have relatively low errors on the training data. However, this low training error, when compared to the unusual cross-validated RMSE scores, reinforces the overfitting

suspicion.

- Recommendations:
  - Recommend adding more variables and increasing observations, using a different algo for feature selection
  - Multiple regression can help ssolve the problems in our assessment
  - Look at regulaization techniques if model remains overfit to reduce model complexity