---

### 0.0.1 Question 1c

Linear models are sensitive to multicollinearity among the columns of the design matrix. So, let's determine the extent of multicollinearity in the college rankings dataset.

In the following cell, we provide you with `arranged_cleaned_college_data`, which is a copy of `cleaned_college_data` containing just a subset of the columns arranged in a particular order.

Create a visualization that shows the pairwise correlation between each combination of columns in `arranged_cleaned_college_data`.

- For 2-D visualizations, consider the `sns.heatmap()` documentation.
- For full credit, title your plot, and set `annot=True`. This makes the plot easier to interpret.
- You may find your plot easier to read with a different color scale. For example, try including `cmap="coolwarm"` inside of `sns.heatmap()`.

**Hint**: Your plot should show $10 \times 10$ values corresponding to the pairwise correlations of the selected columns in `arranged_cleaned_college_data`:

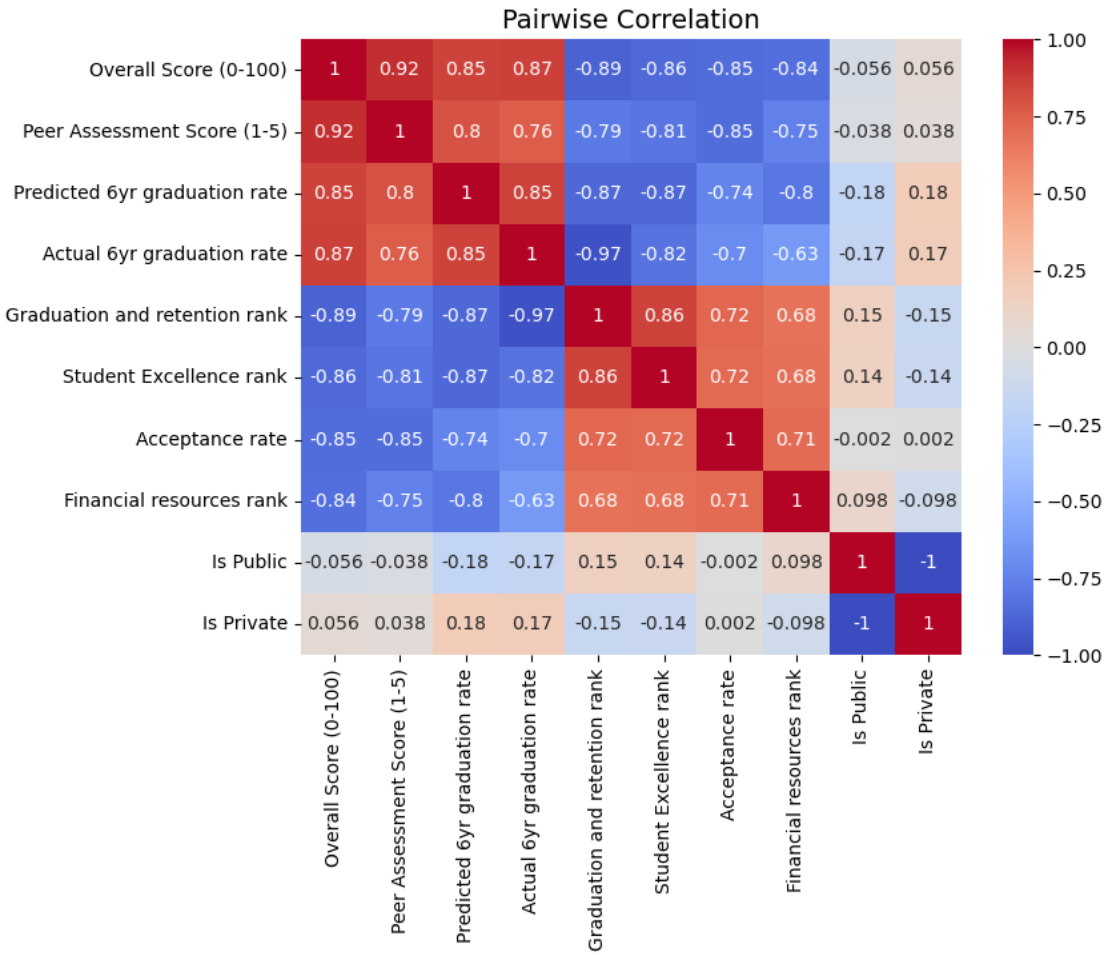
```
['Overall Score (0-100)',
'Peer Assessment Score (1-5)',
'Predicted 6yr graduation rate',
'Actual 6yr graduation rate',
'Graduation and retention rank',
'Student Excellence rank',
'Acceptance rate',
'Financial resources rank',
'Is Public',
'Is Private']
```

```
In [12]: # Running this cell helps you get a subset of cleaned_college_data with the above columns
         arranged_cleaned_college_data = cleaned_college_data[
             ['Overall Score (0-100)',
              'Peer Assessment Score (1-5)',
              'Predicted 6yr graduation rate',
              'Actual 6yr graduation rate',
              'Graduation and retention rank',
              'Student Excellence rank',
              'Acceptance rate',
              'Financial resources rank',
              'Is Public',
```

```
        'Is Private']
]
```

Your output figure should look similar to the following example:

```
In [13]: fig, ax = plt.subplots(figsize=(8, 6))
         sns.heatmap(
             arranged_cleaned_college_data.corr(),
             annot=True,
             cmap="coolwarm",
             ax=ax
         )
         ax.set_title("Pairwise Correlation", fontsize=14)
         plt.show()
```



---

### 0.0.2 Question 1d

Do you notice any patterns in the plot from part (c)? What might explain these patterns? Comment and hypothesize on at least two patterns you notice.

Here are some example questions to ponder:

1. Why do some feature-pairs have correlations of $\pm 1$? Is this a problem?
2. What does the correlation between pairs of features (i.e., graduation-related statistics) look like? Is the magnitude of any of the correlations problemtatically close to 1?
3. Are any features particularly strong predictors of the outcome? If so, why do you think this is the case?
4. Do any features seem potentially redundant? In other words, do you suspect that any features provide similar information about the outcome as other features?

One clear pattern in the heatmap is that "Is Public" and "Is Private" have a correlation of -1, which is expected since they are exact opposites — a college can't be both at once. However, this perfect negative correlation can cause problems in a linear model because it introduces perfect multicollinearity. Including both of them would be redundant, so one should be dropped.

Another noticeable pattern is that graduation-related features — such as "Predicted 6yr graduation rate," "Actual 6yr graduation rate," and "Graduation and retention rank" — are strongly positively correlated (close to 0.85–0.97). This shows that they're capturing very similar information, which makes sense since they're all measuring aspects of student success over time. However, this could lead to overemphasizing graduation in the model unless we select just one or use dimensionality reduction.

Lastly, I observed that "Acceptance rate" has strong negative correlations with features like "Overall Score" and "Peer Assessment Score." This means that colleges that are harder to get into often score higher on other metrics. While this reflects real-world patterns, it also signals multicollinearity that could confuse the model. If many variables are basically pointing in the same direction, the model might have trouble figuring out which one actually matters most.

---

### 0.0.3 Question 1e

If we tried to fit a linear regression model with an intercept term using all features in `cleaned_college_data`, we might run into some problems when fitting our model. The Data 100 staff suggests that we perform the following operation to the `DataFrame` before fitting a model:

```
In [14]: # You must run this cell to achieve pass the public tests for later questions.
         cleaned_college_data = cleaned_college_data.drop('Is Private', axis=1)
```

Describe the reasoning behind this operation. What problem(s) do we avoid by removing the `Is Private` column from the model fitting process?

We remove the "Is Private" column because it is perfectly negatively correlated with the "Is Public" column (correlation = -1). Including both in a linear regression model would create perfect multicollinearity, meaning one column can be exactly predicted by the other. This would confuse the model and make it impossible to uniquely estimate the coefficients. By dropping one of the two, we avoid this redundancy and allow the model to fit properly. This is a common step when dealing with dummy variables.

---

### 0.0.4 Question 2b

Let's visualize the model performance from part 2(a). Plot the following: 1. The observed values vs. the predicted values on the test set. 2. The residuals plot. Recall that for multiple linear regression, we plot the residuals against the predicted values.

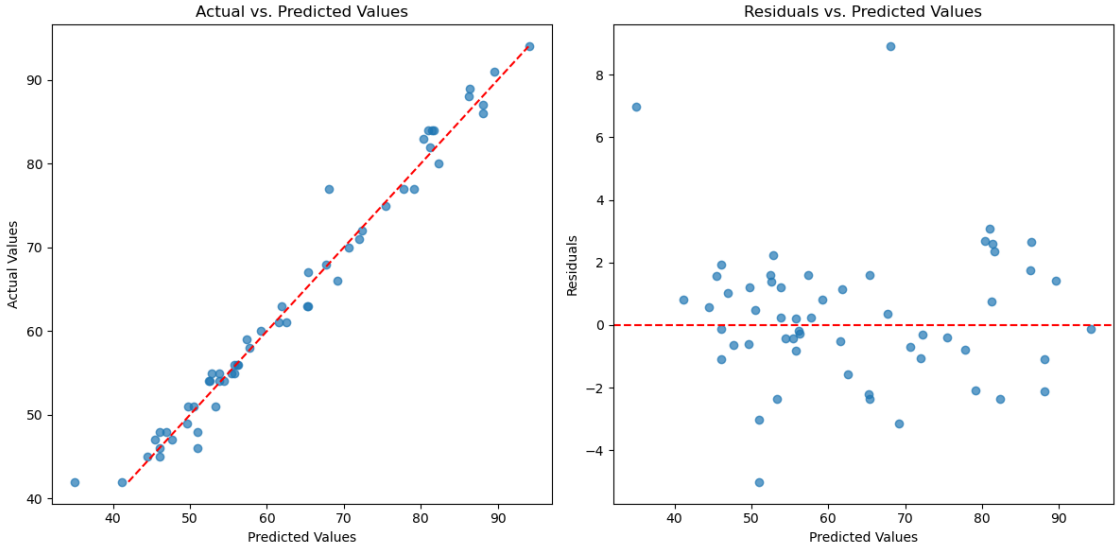
**In both plots, the predicted values should be on the x-axis.**

**Note:** * For a full-credit solution, you should use `plt.subplot()` (documentation) so that you can view both visualizations side-by-side. * The method `plt.subplot({# of rows}{# of cols}{index of plot})` sets the plottable area to the `index` of a `# rows` by `# cols` grid. * For example, `plt.subplot(121)` sets the plottable area to the first index of a 1x2 plot grid. Calling `Matplotlib` and `Seaborn` functions will plot on the first index. When you're ready to start plotting on the second index, run `plt.subplot(122)`. * **Remember to add a guiding line to both plots where $\hat{Y} = Y$, i.e., where the residual is 0**. * `plt.plot()`(documentation) and `plt.axhline()`(documentation) might be helpful here! * Make sure to add descriptive titles and axis labels. * To avoid distorted aspect ratios, ensure the limits of the x-axes for both plots are the same.

```
In [18]: plt.figure(figsize=(12,6))      # do not change this line
         plt.subplot(121)                # do not change this line
         # 1. plot observations vs. predictions
         plt.scatter(Y_test_pred, Y_test, alpha=0.7)
         plt.plot([Y_test.min(), Y_test.max()], [Y_test.min(), Y_test.max()], color='red', linestyle='-
         plt.xlabel('Predicted Values')
         plt.ylabel('Actual Values')
         plt.title('Actual vs. Predicted Values')

         plt.subplot(122)               # do not change this line
         # 2. plot residual plot
         plt.subplot(122)  # do not change this line
         residuals = Y_test - Y_test_pred
         plt.scatter(Y_test_pred, residuals, alpha=0.7)
         plt.axhline(y=0, color='red', linestyle='--')
         plt.xlabel('Predicted Values')
         plt.ylabel('Residuals')
         plt.title('Residuals vs. Predicted Values')

         plt.tight_layout()             # do not change this line
```


Actual vs. Predicted Values
Actual Values
Predicted Values
40
50
60
70
80
90
Residuals vs. Predicted Values
Residuals
Predicted Values
−4
−2
0
2
4
6
8

---

#### 0.0.5 Question 2c

Describe what the plots in part (b) indicate about this linear model. In particular, are the predictions good, and do the residuals appear uncorrelated with the predictions?

The plots show that our linear model is doing a pretty good job. In the first plot (actual vs. predicted), most of the dots are close to the red line, which means the predictions are pretty close to the real values. That's a good sign.

In the second plot (residuals vs. predicted), the dots are scattered kind of randomly around the zero line. There's no clear pattern, and the spread looks pretty even. That tells us that the errors don't depend on the predictions, which is what we want in a good linear model.

So overall, the predictions seem accurate, and the residuals don't show any weird behavior.

**Question 3d(i)** Let us first interpret **Model B**, the linear regression model that use a subset of features from our dataset.

```
In [31]: display(Markdown('#### Model B: Subset of Features'))
         print_confidence_intervals(partial_feature_models, partial_feature_cis)
```

**Model B: Subset of Features**

**Confidence Intervals:**

| parameter | feature name | lower | upper |
|---|---|---|---|
| $\theta_0$ | Intercept | -3.604 | 31.506 |
| $\theta_1$ | Peer Assessment Score (1-5) | 13.882 | 21.341 |
| $\theta_2$ | Acceptance rate | -0.245 | -0.061 |

Are $\theta_1$ and $\theta_2$ significantly different than 0? How do you know?

Does your answer imply that the relationship between `Overall Score (0-100)`, `Peer Assessment Score (1-5)`, and `Acceptance rate` are causal? Do you think the relationships are causal? Explain.

Yes, both and are significantly different from 0. I know this because the 95% confidence intervals for both coefficients do not contain 0. For (Peer Assessment Score), the interval is [13.882, 21.341], and for (Acceptance Rate), it's [-0.245, -0.061]. Since neither of these intervals includes 0, we can be reasonably confident that these features have a real relationship with the Overall Score.

However, this does not mean the relationships are causal. This is an observational dataset, so there could be other confounding variables affecting the results. For example, more selective schools (lower acceptance rate) may also have more resources or prestige, which could influence both the peer score and the overall score. So while the relationships are statistically significant, we cannot conclude that changing peer score or acceptance rate will directly cause a change in overall score.

**Question 3d(ii)** In what situation(s) would you prefer a more compact model with just key features, like Model B? On the other hand, in what situation(s) would you want to consider many features, like in Model A? Explain your answer to both of these questions.

In general, I would prefer a more compact model like Model B when simplicity, interpretability, and efficiency are important. For example, if I'm trying to communicate the main drivers of a college's overall score to a non-technical audience (like administrators or the general public), using just a couple of key features such as Peer Assessment Score and Acceptance Rate makes the model easier to explain and understand. A simpler model is also useful when I have limited data, since using fewer features helps reduce the risk of overfitting.

On the other hand, I would choose a model with many features like Model A when prediction accuracy is more important than interpretability, or when I want a more comprehensive understanding of the problem. Including more features can help capture more of the variation in the target variable and reduce the chance of missing important predictors. For instance, if I were developing a model to forecast future college rankings as accurately as possible, I would want to use all the features available—like graduation rates, financial resources, class size, and more—because each one could add useful information and improve model performance.

In summary, Model B is ideal for clear communication and quick insights, while Model A is better when the goal is strong predictive power and thorough analysis.

---

### 0.0.6 Question 4b

Using the `simulate` function from above, we can compute the model risk, model variance, and variance-to-risk ratio of **Model B**:

```
In [34]: x_last = X.iloc[X.shape[0] - 100]
         y_last = Y.iloc[X.shape[0] - 100]

         (
             partial_feature_model_risk,
             partial_feature_model_var,
             partial_feature_model_ratio
         ) = simulate(x_last[model_b_features], y_last, partial_feature_models)

         print('Model B risk:')
         print(partial_feature_model_risk)

         print('Model B variance:')
         print(partial_feature_model_var)

         print('Model B ratio:')
         print(partial_feature_model_ratio)
```

```
Model B risk:
2.492899391169856
Model B variance:
0.246481679240725
Model B ratio:
0.0988734965052309
```

Comment on the variance-to-risk ratio for Model B (`partial_feature_ratio`).

- Does the model variance appear to be the dominant term in the bias-variance decomposition? If not, what term(s) dominate the bias-variance decomposition?

Then, given your conclusion above, describe what operation(s) you might perform to reduce the model risk.

The variance-to-risk ratio for Model B is around 0.099, meaning only about 9.9% of the model's total error comes from variance. This tells me that variance is not the main issue — the bigger problem is bias. In other words, the model's predictions are not fluctuating a lot, but they are consistently off. This likely happens

because Model B only uses two features, which isn't enough to capture the full complexity of what drives a college's overall score.

Since most of the error is due to bias, the best way to improve the model is by reducing that bias. One approach is to include more relevant features to give the model a fuller picture — like we do in Model A. Another option is to use a more flexible model, such as decision trees, random forests, or ridge regression, which can better handle complex or subtle patterns in the data compared to basic linear regression.

In this case, using a model that can either see more features or better capture relationships between those features and the outcome would help lower the overall error and improve predictions.