Cross Country Course Standardization

Arianna DeBoer and Annie Wicker

December 2024

1 Introduction

Cross country courses can vary significantly in difficulty based on factors like terrain (flat or hilly), surface (grass or dirt), weather, and altitude. Comparing runners' times between different courses can lead to misleading conclusions about capabilities and future race results. While it is impossible to predict how runners will do on a given course and day, it is possible to retrospectively quantify how hard a given race was based on how runners performed compared to past performances. The Track and Field Results Reporting System (TFRRS) is a reputable platform that publishes race results for cross country races across the country. However, it does not have a way to compare results from different courses.

The goal of this project is to create an automated workflow that collects data from TFRRS to standardize runners' times across cross country courses. The goal is to be able to compare race times on courses that may have differed in terrain, elevation, and other factors. Though our analysis will not directly measure the effect of weather fluctuations or specific terrains, it will compare runners' times across different courses and different days to predict how they will do on a future course. This will prove a useful tool for collegiate coaches to allow for more accurate comparison of runners' performances.

2 Data Description

The Track and Field Results Reporting System (TFRRS) is a platform that compiles data from college cross country and track and field competitions. This website obtains data from individual timing companies and uniformly organizes the results. Each meet has its own webpage which contains results from every race run at that meet. For example, the page for 2024 Panorama Farms Invitational has results from both the women's six kilometer race and the men's eight kilometer race. The sample page described can be found through this link.

Each race result page on TFRRS contains the name of the race, the date it was run, and the distance of both the men's and women's courses. The page also contains information about each person who ran in the race.

Our workflow scrapes TFRRS data given a race results page URL. A user can specify that they want to scrape men's or women's results, and the scraped data is compiled into pandas dataframes. The dataframe contains the data described above: the place each runner finished (in that specific race), their name (First Last), year, college, average mile pace, total race time, score (the runner's place as a point value), the name of the meet, and the date this meet was run. These data are recorded for every runner who ran the race whose URL is entered.

Scraped data is loaded into a relational database. All column names are the same as in a TFRRS results table, with the exception of "time" and "raw time". The raw time column formats the time column to be in MM:SS.MS, while the time column lists the time in total seconds. The data is stored in a sqlite database, and the accompanying tables are shown in the below entity relationship diagram.

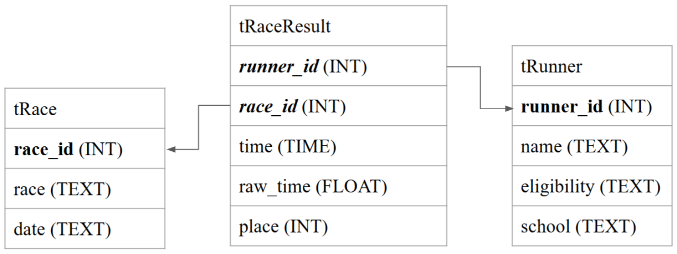


Figure 1: Database Entity Relationship Diagram

3 Methods

For our project, we created a database that contains webscraped race results and various querying functions. A majority of the querying functions are incorporated into a dashboard which was created using the Python library Dash.

The dashboard code contains two sections: layout, which creates the headings, instructions, tabs, and dropdown menus on the dashboard, and the callbacks, which are automatically called when a user interacts with the dashboard. The code for the Dash app calls the database file, and all of the callback functions incorporate the database querying functions. Below, we justify our methods for the course comparison and prediction functions.

The compare_two_courses() function calculates the average differences in time between two courses. It has the following workflow and mathematics:

- Two courses are entered by their ID's. The first is called the primary course and is used as a reference point. The second is called the secondary course.
- All runners who competed in both of the specified races are located. These are the only times used to compare the two races in order to prevent selection bias. The number of runners used in the comparison is outputed as 'NumCompared'.
- The times from these runners are averaged for the primary course as well as the secondary course. This number is then saved.
- The average difference in seconds is found by subtracting the average time for the secondary course from the average time for the primary course. This is outputted as 'Difference'.
- A positive value for 'Difference' means that the secondary course was slower than the primary course.
- The ratio of the two courses is found by dividing the secondary course's average time by the first course's. This is outputted as 'Ratio'.
- A value of 'Ratio' that is greater than 1 means that the secondary course was slower than the primary course.

The predict_times function predicts a runner's expected time based on their past race performances and the average difficulty of other courses they have run.

- The function first calculates the average time for the target course.
- A difficulty ratio is calculated for each course in the database, dividing each course's average time by the average time run on the target course. A difficulty ratio greater than one indicates that the course is harder than the target course, a difficulty ratio less than one means the course is easier.
- An adjusted time is calculated for each runner in the database by multiplying their true time by the course's difficulty ratio.

The conversions() function calculates how fast each loaded race was in comparison to an inputted primary course. These calculations are made both as time differences and as ratios. The function has the following workflow and mathematics:

- One course is entered by it's ID. This course is set as the primary course
- A minimum number of comparisons is entered as an integer, set to min_comparisons = 15 if not specified. There must be at least that many runners in common for a race to be converted. For clarity, for the rest of the explanation, we refer to this number as 15.

- First, all races with at least 15 runners in common with the primary course are designated secondary courses. The compare_two_courses() function is run with the primary course and each secondary course. Results for ratio and time difference are entered into a dataframe as ratio_conversion and time_conversion.

- Next, all non-secondary races with at least 15 runners in common with either the primary or secondary courses are designated tertiary courses. The compare_two_courses() function is run with each tertiary course and each primary or secondary course it shares runners with. In order to find the ratio conversion for each tertiary course, the results from each course comparison are weighted based on how many runners were compared. The calculation for each looks something like this:

  tertiary ratio $= 4/15 * (1 \rightarrow 2) * (2 \rightarrow 4) + 9/15 * (1 \rightarrow 3) * (3 \rightarrow 4) + 2/15 * (1 \rightarrow 4)$

  where the ratio of course 1 to course 4 is being calculated, $(1 \rightarrow 2)$ is the ratio conversion from course 1 to course 2, and 4 runners are in common between these two courses out of 15 total comparisons available for course 4. A similar calculation is completed for the time difference in seconds:

  tertiary difference $= 4/15 * ((1 \rightarrow 2) + (2 \rightarrow 4)) + 9/15 * ((1 \rightarrow 3) + (3 \rightarrow 4)) + 2/15 * (1 \rightarrow 4)$

  where the difference in seconds from course 1 to course 4 is being calculated, $(1 \rightarrow 2)$ is the time conversion from course 1 to course 2, and 4 runners are in common between these two courses out of 15 total comparisons available for course 4.

- This same process is repeated for all other courses that have not yet been converted. These courses are called quaternary courses if they share at least 15 runners in common with any combination of primary, secondary, or tertiary courses.

- At this point, any courses that have not been classified as primary, secondary, tertiary, or quaternary are left out the analysis. The user is alerted of this. All other results are returned in a dataframe.

The virtual_race() function takes a list of schools and runs a virtual race between them based on the times of runners in these schools. The function has the following workflow:

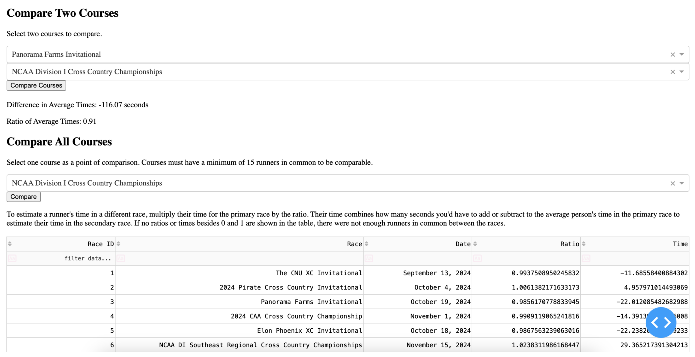
- The conversions() function is called to calculate standardized ratios for all meets.

- These conversions are applied to all runners from the inputted schools.

- The converted times are averaged for each runner.
- The runners are ordered by average time.

4 Output

Below, some screenshots of possible outputs from each tab of our Dashboard are included.



Compare Two Courses

Select two courses to compare.

Panorama Farms Invitational

NCAA Division I Cross Country Championships

Compare Courses

Difference in Average Times: -116.07 seconds

Ratio of Average Times: 0.91

Compare All Courses

Select one course as a point of comparison. Courses must have a minimum of 15 runners in common to be comparable.

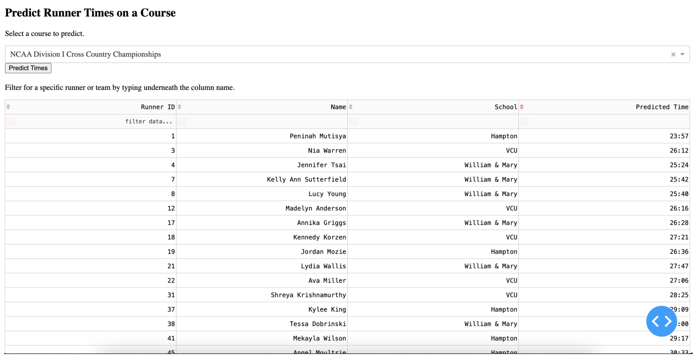
NCAA Division I Cross Country Championships

Compare

To estimate a runner's time in a different race, multiply their time for the primary race by the ratio. Their time combines how many seconds you'd have to add or subtract to the average person's time in the primary race to estimate their time in the secondary race. If no ratios or times besides 0 and 1 are shown in the table, there were not enough runners in common between the races.

Race ID	Race	Date	Ratio	Time
filter data...				
1	The CNU XC Invitational	September 13, 2024	0.9937508950245832	-11.68558400884302
2	2024 Pirate Cross Country Invitational	October 4, 2024	1.0061382171633173	4.957971014493069
3	Panorama Farms Invitational	October 19, 2024	0.9856170778833945	-22.012085482682988
4	2024 CAA Cross Country Championship	November 1, 2024	0.9909119065241816	-14.3913[illegible]5008
5	Elon Phoenix XC Invitational	October 18, 2024	0.9867563239063016	-22.2382[illegible]9233
6	NCAA DI Southeast Regional Cross Country Championships	November 15, 2024	1.0238311986168447	29.365217391304213

Figure 2: Filled dropdown menus comparing two courses and comparing the preloaded courses to the NCAA cross country championships



Predict Runner Times on a Course

Select a course to predict.

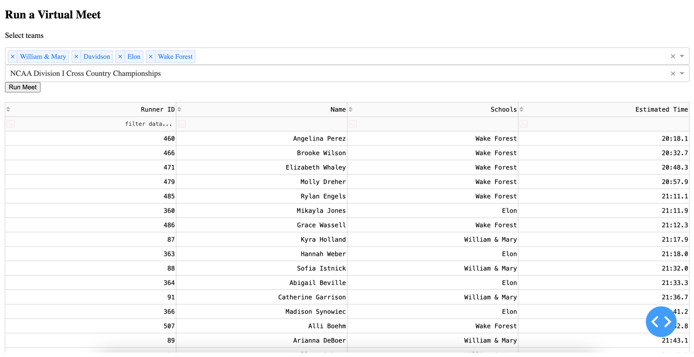
NCAA Division I Cross Country Championships

Predict Times

Filter for a specific runner or team by typing underneath the column name.

Runner ID	Name	School	Predicted Time
filter data...			
1	Peninah Mutisya	Hampton	23:57
3	Nia Warren	VCU	26:12
4	Jennifer Tsai	William & Mary	25:24
7	Kelly Ann Sutterfield	William & Mary	25:42
8	Lucy Young	William & Mary	25:40
12	Madelyn Anderson	VCU	26:16
17	Annika Griggs	William & Mary	26:28
18	Kennedy Korzen	VCU	27:21
19	Jordan Mozie	Hampton	26:36
21	Lydia Wallis	William & Mary	27:47
22	Ava Miller	VCU	27:06
31	Shreya Krishnamurthy	VCU	28:25
37	Kylee King	Hampton	[illegible]9:09
38	Tessa Dobrinski	William & Mary	[illegible]:00
41	Mekayla Wilson	Hampton	29:17
45	Angel Moultrie	Hampton	30:33

Figure 3: Predicting the times of each runner in the database would at the NCAA cross country championships.



Run a Virtual Meet

Select teams

× William & Mary × Davidson × Elon × Wake Forest

NCAA Division I Cross Country Championships

Run Meet

Runner ID	Name	Schools	Estimated Time
filter data...			
460	Angelina Perez	Wake Forest	20:18.1
466	Brooke Wilson	Wake Forest	20:32.7
471	Elizabeth Whaley	Wake Forest	20:48.3
479	Molly Dreher	Wake Forest	20:57.9
485	Rylan Engels	Wake Forest	21:11.1
360	Mikayla Jones	Elon	21:11.9
486	Grace Wassell	Wake Forest	21:12.3
87	Kyra Holland	William & Mary	21:17.9
363	Hannah Weber	Elon	21:18.0
88	Sofia Istnick	William & Mary	21:32.0
364	Abigail Beville	Elon	21:33.3
91	Catherine Garrison	William & Mary	21:36.7
366	Madison Synowiec	Elon	41.2
507	Alli Boehm	Wake Forest	2.8
89	Arianna DeBoer	William & Mary	21:43.1

Figure 4: Running a virtual meet between WM, Davidson, Elon, and Wake Forest at the NCAA cross country championships.

5 Limitations

The compare_two_courses() function course works best when there are many points of comparison. Courses with few runners in common will have less meaningful results. Additionally, all runners in common must be running the course as a race for best results. If a team ran one meet as a workout, for instance, a comparison using this race will be less accurate. This function also assumes that runners have relatively little gain in fitness over the course of the season. If an entire team became much faster between two meets the comparison would be less accurate.

The conversions() function works best when the primary course has many points of comparison with other courses. Because of this, the primary course should be chosen carefully, and the function should be re-run with a different primary course if not enough courses are able to be compared.

The virtual_race() function works best when comparing teams who race in meets that can be connected. Additionally, if at any point one of the teams purposefully runs a meet without giving their full effort, for instance, as a workout, the results will be less accurate.

6 Future Goals

As we continue this project, we plan to create a function that chooses which course should be inputted as the primary course in the conversions() function in order to maximize connections and therefore accuracy. We also plan to create a function that scores the virtual meet and lets the user know which team won.

7 Group Work

Annie wrote the webscraping script to get the data from TFRRS. Arianna created the database and loaded the scraped data into it. Both of us wrote functions to query the database, and Arianna wrote the major function to map courses by relative difficulty. Annie created the dashboard and incorporated the database functionality into it.