Staying in the Game: Ranking Competitiveness

Elissa Lerner Follow
Mar 20 · 9 min read

> ***Authors:*** *Jenny Chen, Jack Van Boening, Quang Nguyen*

> ***Contributors:*** *Jenny Xue, Luis Ulloa, Aimun Khan, Javier Banda, Beleicia Bullock, Gwendolyn Goins, Jose Eduardo Coronado, Alice Bian, Asim Hirji, Rabina Phuyel, Matt Huo*

They say the only person worth competing with is yourself. But anyone who watches college basketball knows that's not the whole story. Each game is a competition, and only one team can win. And while there are many factors that may help you in a game, "competitiveness" is a buzzword that pops up everywhere from coaches' huddles to media commentary. But what does it actually mean? Using the NCAA basketball data set, BigQuery, and Colab, our team of 14 college students set out to see if we could determine the competitiveness of each of the 353 D1 men's college basketball teams this season.

The Process

The tools were easy—especially with your Google Cloud based data analysis architecture, but the process? Not as much. Turning the qualitative into quantitative can be hard!

Take 1: Players Make the Team

Since a team is only as strong as the sum of its parts, we thought we'd try valuing competitiveness based on the makeup of the players on the court. Our first path relied on an algorithm that aggregated the overall efficiency score of the players on the court on a play-by-play basis. This dynamic metric could then be used to provide an average overall score for each team per game, which in turn could be used as a season-wide rating. A neat idea, but we realized we were more interested in team performance, rather than their specific lineups and individual players.

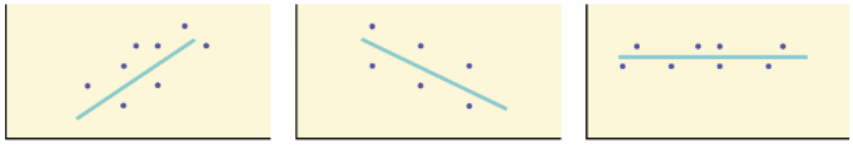
```
100   ARRAY (SELECT CASE WHEN x = bench_out THEN bench_in ELSE x END FROM unnest(starting_five_players) x) as current_players_
101
102   (time) as current_time
103
104   FROM pbp_first_players, pbp_home_agg
105   , UNNEST (H_AGGIN) as bench_in
106   , UNNEST (H_AGGOUT) as bench_out
107   )
108
109 SELECT *
110 FROM pbp_current_players
```

	starting_five_players	bench_in	bench_out	current_players_on_field	current_time
0	[1516058, 1635377, 1635380, 1964918, 1635378]	1635378	1516058	[1635378, 1635377, 1635380, 1964918, 1635378]	159
1	[1516058, 1635377, 1635380, 1964918, 1635378]	1856650	1635377	[1516058, 1856650, 1635380, 1964918, 1635378]	187
2	[1516058, 1635377, 1635380, 1964918, 1635378]	1964920	1635380	[1516058, 1635377, 1964920, 1964918, 1635378]	327

BigQuery Snippet Tracking the Current 5 Players on Court

Take 2: Fitting the Line

We started tackling a team-level metric using basic statistical regression models. We used the derivative of the score differential (read: the change in score margin) to see how teams were performing throughout the game. The derivatives had three generic possibilities: positive slope, negative slope, or approximately zero slope.



Positive, negative, zero slopes

A derivative close to zero meant a very close game while a constantly positive or negative slope could signify the possibility of a blowout. This metric would help measure the momentum of a team, and steep changes in the slope could help us identify runs. But we wanted to go further than simply matching a best-fit line.

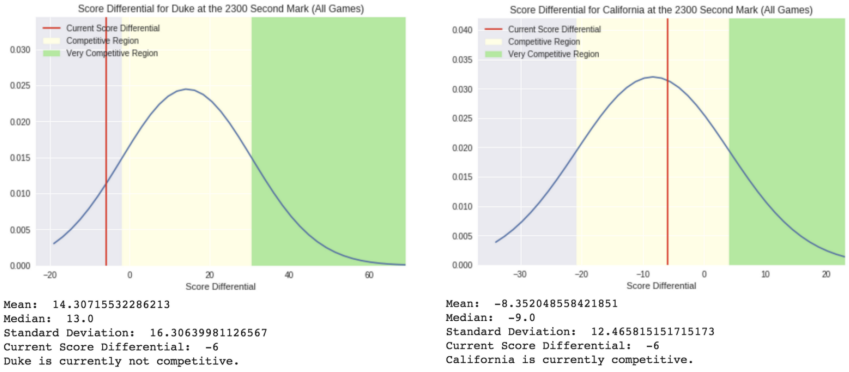
Take 3: Deviating from the Line

A best-fit line only informs us about the competitiveness of a game, not of a *team.* And even if we were to have ranked the teams based on the ones with the most ties, or score changes, or score margins closest to zero, we'd have been discarding strong teams along with weak teams (i.e., if Gonzaga is frequently up by 10+ points, we'd be losing them in the data). We needed a way to refine competitiveness for team performance throughout the game.

We sliced each game into 4-minute time segments, which is the approximate time between each TV timeout, and lends itself to game analysis. We then looked at how a team was performing at some point in a game and compared it against its own averages for that respective segment of time throughout the season, which could tell us whether or not the team was playing competitively by its standards. From there, we looked at the distribution, and decided to test out the following classifications for competitiveness:

- If the score differential is *below* one standard deviation of the average scores in that time segment, then the team is considered **uncompetitive**.
- If the score differential is *within* one standard deviation of the average scores in that time segment, then the team is considered **competitive**.
- If the score differential is *above* one standard deviation of the average scores in that time segment, then the team is considered to be **very competitive.**

As you might have guessed, college basketball wasn't so tidy in reality. Take a look at the hypothetical scenario below and you'll see why. Let's imagine Duke and Cal at the exact same point in two different games: 100 seconds left to play and both teams are down by 6.



Duke left, Cal right

On the left is Duke, on the right is Cal. You'd think that Duke being down by 6 would mean they'd still be competitive, and within reach of getting back in the game (two possessions with 1:40 left on the clock is plenty of time). But because we're testing this scenario against the team's own results from this season, the bar gets set much higher. Duke just hasn't been losing this season with about a minute and a half left to play—regardless of whether they've been up or down (which, if you think about it, sort of implies the team is *extremely* competitive). Meanwhile, Cal has struggled throughout the season, to the point that their average score margin is already in the negative at 100 seconds left to play. Which meant that Cal would still be competitive in this scenario, while Duke wouldn't.

Something was still off.

Take 4: A Working Definition

We continued to partition individual games into the same 4-minute segments as before. But this time, rather than attempting to create a definition of competitiveness that could stand on its own, we focused

on creating a ranking mechanism that would allow us instead to sort all 353 men's teams in relation to each other.

We decided to look at how much time a team plays "competitively" vis-a-vis their total season playing time, where "competitively" is a function of the change in score differential in each of those 4-minute time segments. We defined a competitive time segment as one in which the score differential for that team is positive, and that team is not down by 20 points or more at the end of that time segment. (Admittedly, 20 points was somewhat arbitrary, but generally speaking, a game with a 20-point lead is rarely considered competitive, exceptions this year notwithstanding. If a team starts to chip away and gets the score difference below 20, then competitiveness is back under consideration.)

When all of these terms are met, then the team has a competitive time segment. Here's a snippet of the function we used to find these segments:

Note: Function(s) are driven by DataFrames materialized from BigQuery views. We attempt to push down as much aggregation to SQL and then rely on pandas for the application of complex logic.

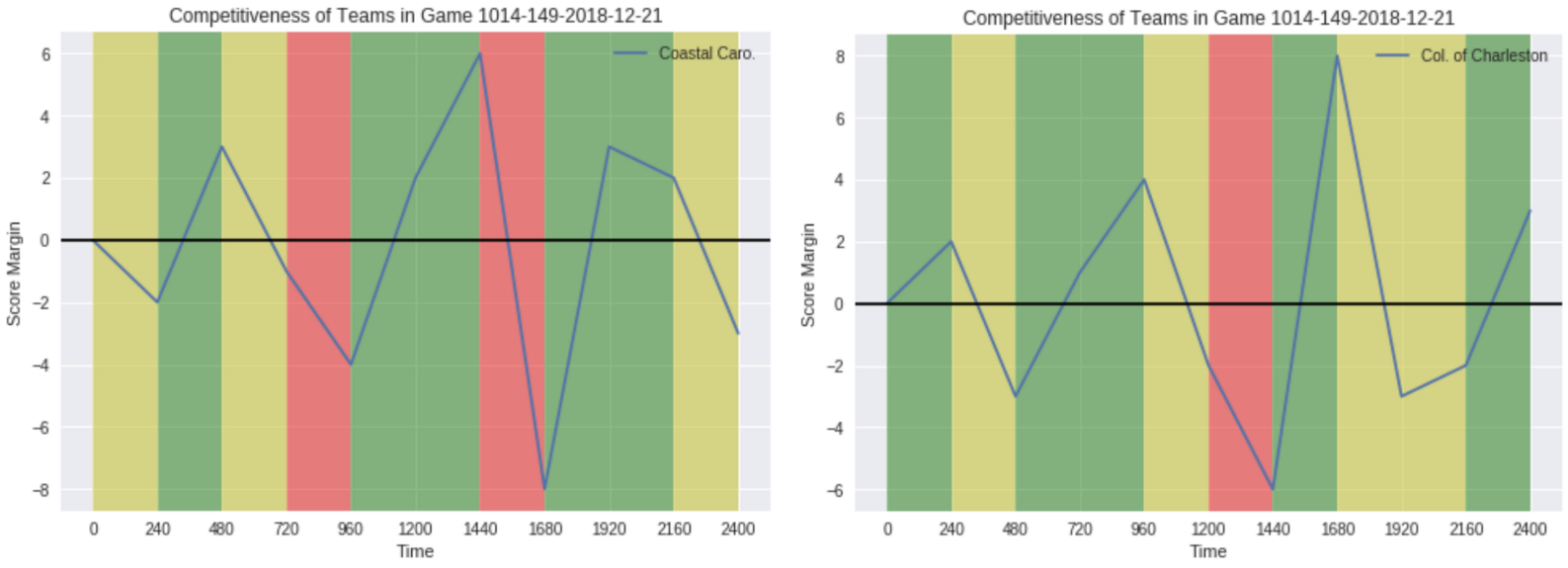
```
def get_metric_values(client, games_df):
  """
  Returns count of winning segments and count of total segments
  for the team specified in games_df.

  """
  specifiedCount = 0
  totalCount = 0
  game_ids = set(games_df["game_id"])
  for game_id in game_ids:
    game_segments_score_diff = [0]*10 # 10 segments, each 240
    game_total_score_diff = [0]*10 # 10 segments, each 240

    is_home = False
    for row, data in games_df[games_df["game_id"] == game_id].iterrows():
      time_segments = ((data["elapsed_time_sec"]) - 1) // 240
      if time_segments >= 10:
        continue

      points_scored = data["points_scored"]
      home_pts = data["home_pts"]
      away_pts = data["away_pts"]

      game_total_score_diff[time_segment] = home_pts - away_pts

      if points_scored != 0:
        if data["team_code"] == team_code:
          game_segments_score_diff[time_segments] += points_scored

        else:
          game_segments_score_diff[time_segments] -= points_scored

      # technically should only have to be done once
      if data["team_code"] == team_code and data["is_home"]:
        is_home = True


    if not is_home:
      for score in game_total_score_diff:
        score *= -1

    # competitiveSegment = score diff > AND game_score_diff > -20
    # Loop through each segment and determine if it is a 'comepteitiveSegment'
    gm_count = 0
    for i in range(10):
      # Check segment score
      if (game_segments_score_diff[i] > 0):
        # Check game score
        if game_total_score_diff[i] > -20:
          specifiedCount += 1
      totalCount += 1

  return specifiedCount, totalCount
```

Colab snippet, dark mode

From there, we expressed the ratio of a team's number of competitive time segments in a season to its total time segments in a season as follows, thereby creating a simple unified competitiveness score that allows us to compare and rank all teams from 1 to 353:

$$team\ competitiveness\ score\ =\ 100 * \left(\frac{\#\ of\ team\ \ competitive\ time\ segments}{Total\ \#\ of\ team\ time\ segments}\right)$$

Let's compare the differences between our third definition and the one we landed on by looking at the Coastal Carolina vs. College of Charleston matchup on December 21st. That game maintained a tight score margin throughout the entire game, with an ending score margin of just two points. Using our three competitiveness classifications from before, we graphed competitive time segments. The graphs below not only illustrate this frequent change of point leads throughout the entire game (the blue line), but also allows us to see the frequent change in competitiveness from each team's perspective.



Coastal Carolina left, College of Charleston right

The green bars illustrate positive score differentials, meaning the team was playing competitively (read: in a competitive time segment).

Meanwhile, the yellow and red bars illustrate negative score differentials, meaning uncompetitive time segments based on our current definition. But we took the absolute values of these negative score differentials to see how our standard deviations would look from before and to get a better sense of the overall game play. Yellow bars illustrate a score differential below one standard deviation from the team's season average in that time segment, and red bars illustrate

greater than one standard deviation from the team's season average in that time segment. Yellow shows uncompetitive time segments, and red shows *very* uncompetitive time segments.

When looked at this way, you can think of a team playing competitively if they are increasing the score margin over small intervals of time continuously throughout the game. This handles errors that would occur as a result of blow-out games (i.e., if a team was down by 25 but managed to outscore their opponents by 5 in the last 4 minutes, yielding a positive score margin for that time segment but still losing by 20), this model wouldn't count that segment as 'competitive').

Applied Competitiveness

With a solid definition in hand, we used ridge regressions to adjust our raw competitiveness ratings using schedule adjusted metrics of game-level data. This accounting for the difficulty of different opponent matchups helped put the final touch on our understanding of competitiveness.

Here's how competitiveness looks in the context of the current season using schedule-adjusted metrics, with our top ten results.

```
# Loop over raw and adjusted version of each metric
for metric_type in ['raw', 'adj']:
  # Translate each stat to 0-100 'rating' by normalizing vs season mean/sd
  tm_seasons_stats_ranks[metric_type + '_rtg'] = stats.norm.cdf(
    np.where(tm_seasons_stats_ranks['rank_asc'], -1, 1) *
    (tm_seasons_stats_ranks[metric_type + '_stat']
      - tm_seasons_stats_ranks['season_' + metric_type + '_stat_mean']) /
     tm_seasons_stats_ranks['season_' + metric_type + '_stat_sd']
    ) * 100

  # Rank on rtg field, in correct dir, since 0 = worse & 100 = better by des
  tm_seasons_stats_ranks[metric_type + '_rk'] = np.where(
    # No rankings for group of Non-D1 teams
    tm_seasons_stats_ranks['rk_group'] == 'NON-D1', np.nan,
    (tm_seasons_stats_ranks.
    # Group by season, stat name, & rank group (so Non-D1 teams don't 'mix'
    groupby(['season', 'stat_name', 'rk_group'])[metric_type + '_rtg'].
    rank(ascending = False)
    ))

tm_seasons_stats_ranks
```

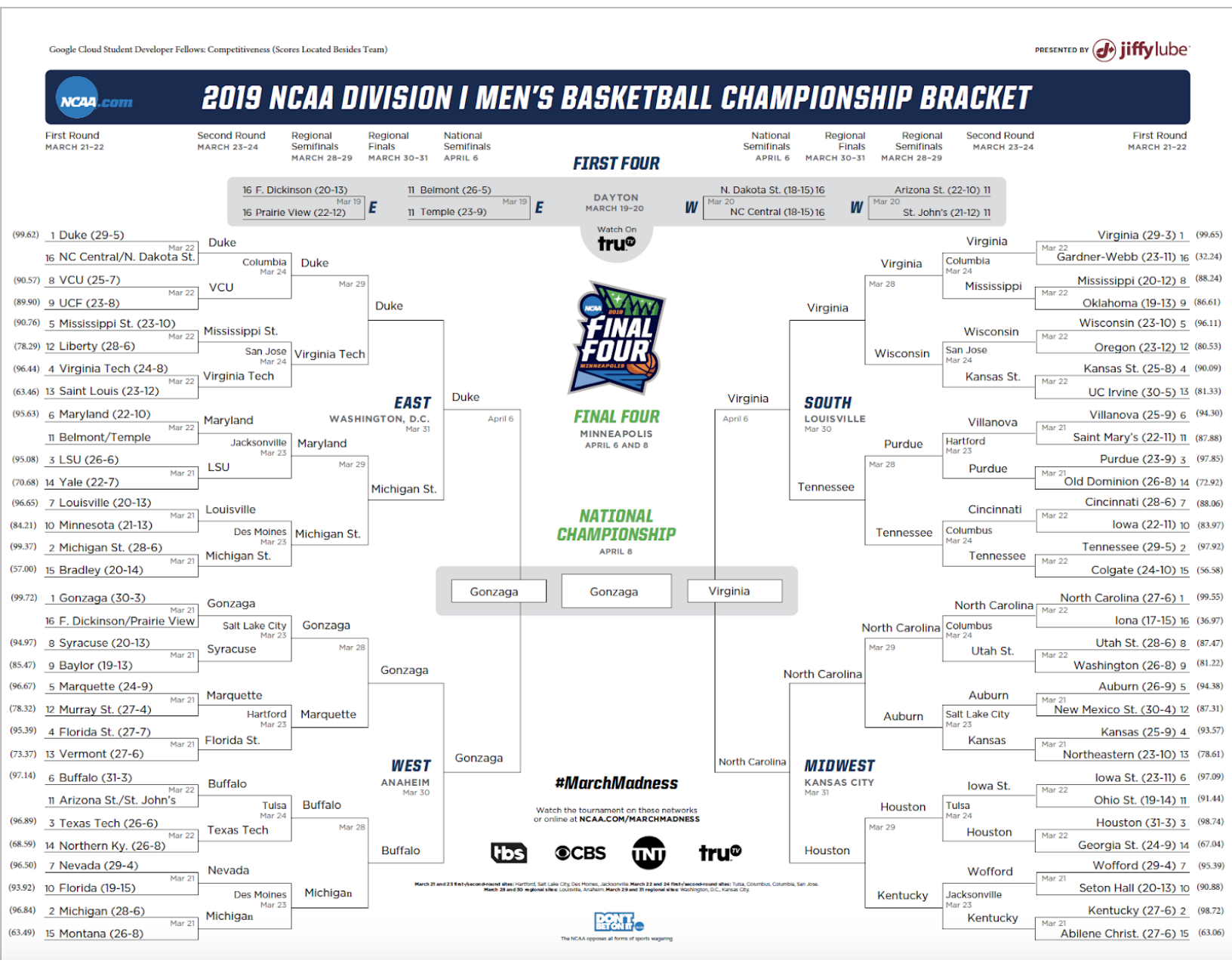
Colab snippet

School	Adj Competitive Rating	Raw Competitive Rating	Diff
1 Gonzaga	99.72	99.89	-0.17
2 Virginia	99.65	98.85	0.80
3 Duke	99.62	96.93	2.69
4 North Carolina	99.55	85.25	14.30
5 Michigan St.	99.37	97.08	2.29
6 Houston	98.74	99.91	-1.16
7 Kentucky	98.72	96.42	2.30
8 Tennessee	97.92	98.88	-0.95
9 Purdue	97.85	90.68	7.16
10 Buffalo	97.14	97.38	-0.24

'18-'19 Division 1 Basketball Rankings by Competitiveness Through 3/18

Lots of familiar names in this list, and the top nine are all 3-seeds or higher as of Selection Sunday. But Buffalo might give you pause—after all, they're only a 6-seed! However, the Bulls have performed well all season, going 31–3, with two of those losses getting decided by 4 points or fewer. They're also heading into the tournament on a hot streak after having won the MAC tournament and dominating their regular season. So it's less surprising to see Buffalo rank so highly in competitiveness, even after schedule adjustment.

Curious for more? We've created a bracket for you with our competitiveness scores next to the team names for every team (minus the First Four round):



A truly competitive bracket

If you're looking for potential upsets based on competitiveness, we suggest keeping an eye out for Nevada (7) vs Florida (10), Wofford (7) vs Seton Hall (10), Auburn (5) vs New Mexico St. (12), Cincinnati (7) vs Iowa (10), Villanova (6) vs Saint Mary's (11), and Kansas St. (4) vs UC Irvine (13). While we aren't predicting first-round upsets outright, these games notably pit two teams of similar competitiveness against one another.

Competitiveness: Beyond the definition

Creating a definition for competitiveness is only the beginning—now comes the challenge of using and exploring it in interesting ways. Next up, we'll be creating models exploring how competitiveness may factor into the winning odds of any given team, and potentially building a predictive model using this as a feature. And while we've been busy

with competitiveness, our counterpart team of student analysts have been investigating *explosiveness*, which measures a team's ability to go on scoring runs. Much like how adjusting for schedule affected the outcome of our definition, we'll be exploring the effects of competitiveness and explosiveness on each other.

Stay tuned, and let's start dancing!

If you'd like to learn how to use Google Cloud for data analysis and data science head over to the Google Cloud Training quest.

Additional thanks to Eric Schmidt, Alok Pattani, Elissa Lerner