How to spot the Table of Death with R

Using basic R commands to analyze the performance of a competition BBQ team

Not to worry, no person or persons were harmed in the making of this tutorial. The Table of Death I am referring to is the hellish conglomeration of six particularly low scoring (dare I say mean) competition barbecue judges, together on a single table. A table which can and will destroy the dreams and aspirations of every BBQ team unlucky enough to land there.

OK, to be fair the judges probably aren't that mean…. well maybe a little… none the less finding the Table of Death (TOD) is crucial for a BBQ team to critically evaluate their performance.

If you haven't guessed I am a pitmaster of one of these competition BBQ teams. My team name is Good Smoke BBQ and we compete in Kansas City Barbecue Society (KCBS) sanctioned contests across the country. I am also an image scientist and all-around data enthusiast. When our sanctioning body, KCBS, decide to revamp the scoring reports a few years ago, my two passions were on a collision course.

Competition Barbecue

KCBS competitions are four category events, where teams prepare chicken, pork ribs, pork shoulder, and brisket. Teams turn in each category at a given time and will be placed on a table of judges. Each table has six team entries and six judges to score them. The event will have many tables, enough for all the teams. The total score of all four categories will determine the overall winner of the competition.

There is an inherent issue in how we are judged, and keen observers may have already noticed it, so let's me try to highlight it with an example. Let's say the I am competing heads up against Johnny Trigg (https://en.wikipedia.org/wiki/Johnny_Trigg) in ribs (my personal nightmare). In a perfect world to find out who has the best ribs we would want to be judged by every judge in the world, the largest population possible. The average would be very precise, in fact perfect. We call this the population mean or true mean. Clearly Trigg's average would be higher than mine and he would win.

Obviously, this isn't practical as there is no way to cook that much food. The great thing about statistics is that we can come up with a pretty good estimate (sample mean) of the population mean by taking a random sample (subset of judges) from the population. The mean of a random sample of a population will be equal to the population mean given enough samples (assuming the data is distributed normally). So, if we pick enough judges we should be able to approach the population mean for each table. If every table is close to the population mean than they would all be close to each other, meaning a fair evaluation of the teams in a competition across all the tables.

Statistical Limitations

The issue I alluded to earlier is that the KCBS rules are structured for choosing only six judges per team, which is far lower than what statistics say we should be using. We can view the seating of judges at a table as a random(ish) sample of the judging pool for a given category. With only six choices per table we are statistically limited and the odds that the collection of tables will be unbalanced are high. This leads to the dreaded TOD and by equal chance the TOA, the Table of Angels. Funny how we never complain about the TOA, but I digress.

I am not sure how KCBS came to the six-judge conclusion all those years ago. The driving factor was likely the practicality of turning in only six portions for a team. Clearly a statistician was not enjoying a beer around the fire when it was decided. In fact, this wasn't a problem until KCBS updated their scoring report to show each team which table they landed on.

To say this new scoring report was controversial is a gigantic understatement. The BBQ team community was up in arms about how variable the tables were. Hit the TOD in any of the categories and your chance of winning overall is over. Of course, this was always a problem, but the data to show it was hidden in the old reports. Being a lover a data, I was super excited to see the table listing! Did I tank because my food was bad or was it a statistical outlier table? Now we had the chance to try and understand.

Understanding your score

Many have suggested all sorts of ways to improve the variability and KCBS is working the problem. While doing analysis on the current system itself and possible changes sounds exciting (and daunting), I wanted to share the simpler analysis I do to examine my scores in each of the categories after a competition.

It is important for a team to understand as much as they can about their score. Knowing what type of table you land on can and should influence a team's response post competition. If you finished low and were on a good to decent table then maybe a recipe tweak or process change is in order. If you finished low but hit a bad table, maybe everything is OK and no changes are necessary at this point. Knowing is the key.

While none of the math is complex, using a tool like R can make this task simple and repeatable. If you don't have any experience with R, have no worries, it is a very user-friendly language, freely available for download (Language: https://www.r-project.org/ Editor: https://www.rstudio.com/) and you can even take a free course here: https://www.datacamp.com/courses/free-introduction-to-r

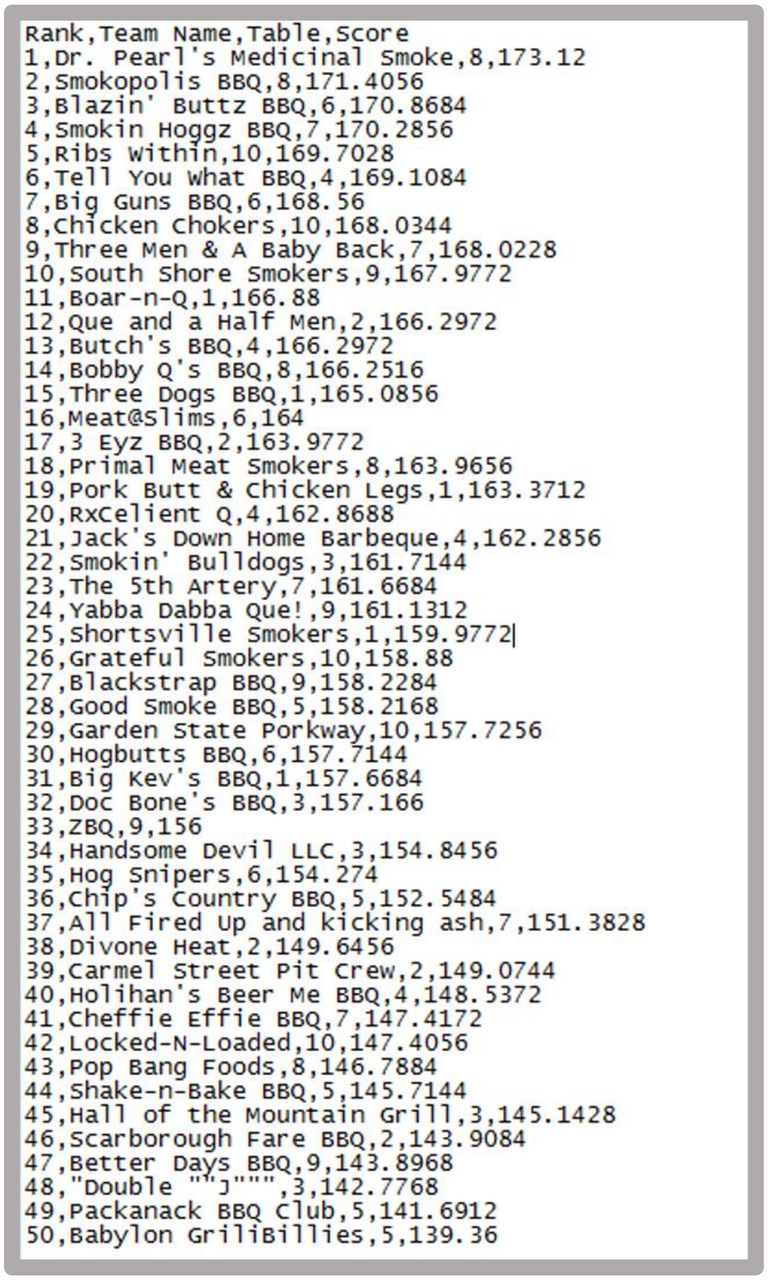
The example I am going to show is the KCBS chicken category I competed in from the 2013 Hudson Valley Ribfest (https://www.kcbs.us/event/4042/hudson-valley-ribfest). I thought our chicken turn-in was very good and was very surprised with how low it finished (28 out of 50).

Being confused about the finish and armed with the new data, I set off to find out if it was me or the table.

Importing the data

First thing to do is import our data. I took the data from our new scoring report and entered it in to a comma-separated values or CSV file. For those who want to play along at home the csv file along with the R script containing the commands can be found here: https://github.com/brianwemett/TOD

The csv file looked like this:



```
Rank,Team Name,Table,Score
1,Dr. Pearl's Medicinal Smoke,8,173.12
2,Smokopolis BBQ,8,171.4056
3,Blazin' Buttz BBQ,6,170.8684
4,Smokin Hoggz BBQ,7,170.2856
5,Ribs Within,10,169.7028
6,Tell You What BBQ,4,169.1084
7,Big Guns BBQ,6,168.56
8,Chicken Chokers,10,168.0344
9,Three Men & A Baby Back,7,168.0228
10,South Shore Smokers,9,167.9772
11,Boar-n-Q,1,166.88
12,Que and a Half Men,2,166.2972
13,Butch's BBQ,4,166.2972
14,Bobby Q's BBQ,8,166.2516
15,Three Dogs BBQ,1,165.0856
16,Meat@Slims,6,164
17,3 Eyz BBQ,2,163.9772
18,Primal Meat Smokers,8,163.9656
19,Pork Butt & Chicken Legs,1,163.3712
20,RxCelient Q,4,162.8688
21,Jack's Down Home Barbeque,4,162.2856
22,Smokin' Bulldogs,3,161.7144
23,The 5th Artery,7,161.6684
24,Yabba Dabba Que!,9,161.1312
25,Shortsville Smokers,1,159.9772
26,Grateful Smokers,10,158.88
27,Blackstrap BBQ,9,158.2284
28,Good Smoke BBQ,5,158.2168
29,Garden State Porkway,10,157.7256
30,Hogbutts BBQ,6,157.7144
31,Big Kev's BBQ,1,157.6684
32,Doc Bone's BBQ,3,157.166
33,ZBQ,9,156
34,Handsome Devil LLC,3,154.8456
35,Hog Snipers,6,154.274
36,Chip's Country BBQ,5,152.5484
37,All Fired Up and kicking ash,7,151.3828
38,Divone Heat,2,149.6456
39,Carmel Street Pit Crew,2,149.0744
40,Holihan's Beer Me BBQ,4,148.5372
41,Cheffie Effie BBQ,7,147.4172
42,Locked-N-Loaded,10,147.4056
43,Pop Bang Foods,8,146.7884
44,Shake-n-Bake BBQ,5,145.7144
45,Hall of the Mountain Grill,3,145.1428
46,Scarborough Fare BBQ,2,143.9084
47,Better Days BBQ,9,143.8968
48,"Double ""J""",3,142.7768
49,Packanack BBQ Club,5,141.6912
50,Babylon GriliBillies,5,139.36
```

The I used the `read.csv` command to store the data in a frame called `AllData`

```
#Read in CSV file
AllData = read.csv("ChixScores.csv")
```

Here is what the data frame `AllData` looked like after the import:

Rank	Team Name	Table	Score
1	Dr. Pearl's Medicinal Smoke	8	173.12
2	Smokopolis BBQ	8	171.4056
3	Blazin' Buttz BBQ	6	170.8684
4	Smokin Hoggz BBQ	7	170.2856
5	Ribs Within	10	169.7028
6	Tell You What BBQ	4	169.1084
7	Big Guns BBQ	6	168.56
8	Chicken Chokers	10	168.0344
9	Three Men & A Baby Back	7	168.0228
10	South Shore Smokers	9	167.9772
11	Boar-n-Q	1	166.88
12	Que and a Half Men	2	166.2972
13	Butch's BBQ	4	166.2972
14	Bobby Q's BBQ	8	166.2516
15	Three Dogs BBQ	1	165.0856
16	Meat@Slims	6	164
17	3 Eyz BBQ	2	163.9772
18	Primal Meat Smokers	8	163.9656
19	Pork Butt & Chicken Legs	1	163.3712
20	RxCelient Q	4	162.8688
21	Jack's Down Home Barbeque	4	162.2856
22	Smokin' Bulldogs	3	161.7144
23	The 5th Artery	7	161.6684
24	Yabba Dabba Que!	9	161.1312
25	Shortsville Smokers	1	159.9772
26	Grateful Smokers	10	158.88
27	Blackstrap BBQ	9	158.2284
28	Good Smoke BBQ	5	158.2168
29	Garden State Porkway	10	157.7256
30	Hogbutts BBQ	6	157.7144
31	Big Kev's BBQ	1	157.6684
32	Doc Bone's BBQ	3	157.166
33	ZBQ	9	156
34	Handsome Devil LLC	3	154.8456
35	Hog Snipers	6	154.274
36	Chip's Country BBQ	5	152.5484
37	All Fired Up and kicking ash	7	151.3828
38	Divone Heat	2	149.6456
39	Carmel Street Pit Crew	2	149.0744
40	Holihan's Beer Me BBQ	4	148.5372
41	Cheffie Effie BBQ	7	147.4172
42	Locked-N-Loaded	10	147.4056
43	Pop Bang Foods	8	146.7884
44	Shake-n-Bake BBQ	5	145.7144
45	Hall of the Mountain Grill	3	145.1428
46	Scarborough Fare BBQ	2	143.9084

47	Better Days BBQ	9	143.8968
48	Double "J"	3	142.7768
49	Packanack BBQ Club	5	141.6912
50	Babylon GriliBillies	5	139.36

The next step is to find the overall mean of the chicken scores, we accomplish that by simply using the mean function. To get the mean of the scores we need access or extract the appropriate column with the `$` operator.

```
#Calculate the overall chicken mean
chixMean <- mean(AllData$Score)

chixMean
[1] 158.3773
```

Awesome! Now we have the global mean for the chicken scores across all tables. This value is going to be the bar with which we compare each of the individual table means. We now need to get the means of each table. To start we do a compound command to find out how many tables we have. The process is extracting the `Table` column with the `$` operator, finding all the unique values with the `unique` function, and finally counting how many with the `length` command. This value is stored in `numTables`.

```
#Find the number of tables
numTables <- length(unique(AllData$Table))

numTables
[1] 10
```

Ok… looks like we have 10 different tables. Let's create a new vector to store the table means called `tableMeans`. The `numeric` function will create the vector of length `numTables`.

```
#Create a vector for table means
tableMeans <- numeric(numTables)

tableMeans
[1] 0 0 0 0 0 0 0 0 0 0
```

Now we need to populate the vector with the appropriate means which we will accomplish with a for loop. So, we subset the `AllData$Score` extraction with each table number and iterate across all tables, taking the mean.

```
#find the table means
for(i in 1:numTables)
{
  tableMeans[i] <- mean(AllData$Score[AllData$Table==i])
}

tableMeans
[1] 162.5965 154.5806 152.3291 161.8194 147.5062 163.0834 159.7554
164.3062 157.4467 160.3497
```

We can now start to make some conclusions on the scoring across the tables. Remember the chicken mean was 158.3773, so we see that 6 tables were above the mean and 4 were below. Pretty close to what statistics would predict, with more tables (samples) we would probably see the same number of tables above and below the mean.

Remember I was very happy with my chicken this day, can you see why I might have been right? My chicken hit table 5… the very lowest scoring table in the event. I was the highest scoring team on the table with a score of 158.2168 a score that is just below the overall chicken mean. So, it looks like I was just a bit unlucky to hit the lowest scoring table for this category.

We can after the fact "pretend" that the judges are from the same population and adjust the scores for individual table means. This is standard in the data analysis world, when we are compensating for some bias, particularly when dealing with humans. Not sure how I and other teams would respond to having their scores changed after the fact, but it does make for some interesting analysis.

So… let's do it! Let's find a bias factor for each table, adjust each team's score and re-rank the contest! My table was so bad, maybe I can even win the chicken category! We already have overall chicken mean and the mean for each of the tables. Now we can easily find our bias factor, which will be the overall chicken mean minus each table mean. This number will let us know how far each table is away from the overall chicken mean. The vector `biasFactors` is created by using the vector subtraction below:

```
#Create bias factor
biasFactors <- chixMean - tableMeans

biasFactors
[1] -4.219168  3.796752  6.048192 -3.442128 10.871152 -4.706048 -
1.378048 -5.928928  0.930592 -1.972368
```

Ok, looking good for table 5, with a whopping 10.871152 bias factor! Let's add the bias factor to our `AllData` frame, to keep all our data together and populate it with the `biasFactors` vector data. The third column of `AllData` is the table number for team `j` and we use it to index the `biasFactors` vector returning the bias factor for each team.

```
#Add Bias factor field to data frame
AllData$BiasFactor <- 0

#Loop over teams and add bias factor
for(j in 1:numTeams)
{
  AllData[[5]][j] <- biasFactors[AllData[[3]][j]]
}
```

Now let's add a team's new score to the data frame, which is their original score `AllData$Score` plus their bias factor `AllData$BiasFactor`.

```
#Add new score to data frame
AllData$NewScore <- AllData$Score + AllData$BiasFactor
```

Let's create a new data frame called `NewRankData` by re-ranking the event using the order function on the `AllData$NewScore` field. We use `AllData` column 6 as it is the new score, and we also apply the minus sign so the order returned is decreasing, leaving the highest score on top. Let's also add a new rank to the `NewRankData` so we can see a team's new finishing position and calculate how far a team changed in rank.

```
#Sort to find Find new rank
NewRankData <- AllData[order(-AllData[6]),]

#Add new rank
NewRankData$NewRank <- 1:numTeams

#Calculate change in rank
NewRankData$Change = NewRankData$Rank - NewRankData$NewRank)

NewRankData
```

Drum roll please… So how do you think we faired? Turns out we had the largest improvement, moving up 26 places. Yep that's right 2nd place, missed it by that much! The final data frame `NewRankData` is shown here:

Rank	Team Name	Table	Score	BiasFactor	NewScore	NewRank	Change
12	Que and a Half Men	2	166.2972	3.796752	170.094	1	11
28	Good Smoke BBQ	5	158.2168	10.87115	169.088	2	26
10	South Shore Smokers	9	167.9772	0.930592	168.9078	3	7
4	Smokin Hoggz BBQ	7	170.2856	-1.37805	168.9076	4	0
17	3 Eyz BBQ	2	163.9772	3.796752	167.774	5	12
22	Smokin' Bulldogs	3	161.7144	6.048192	167.7626	6	16
5	Ribs Within	10	169.7028	-1.97237	167.7304	7	-2
1	Dr. Pearl's Medicinal Smoke	8	173.12	-5.92893	167.1911	8	-7
9	Three Men & A Baby Back	7	168.0228	-1.37805	166.6448	9	0
3	Blazin' Buttz BBQ	6	170.8684	-4.70605	166.1624	10	-7
8	Chicken Chokers	10	168.0344	-1.97237	166.062	11	-3
6	Tell You What BBQ	4	169.1084	-3.44213	165.6663	12	-6
2	Smokopolis BBQ	8	171.4056	-5.92893	165.4767	13	-11
7	Big Guns BBQ	6	168.56	-4.70605	163.854	14	-7
36	Chip's Country BBQ	5	152.5484	10.87115	163.4196	15	21
32	Doc Bone's BBQ	3	157.166	6.048192	163.2142	16	16
13	Butch's BBQ	4	166.2972	-3.44213	162.8551	17	-4
11	Boar-n-Q	1	166.88	-4.21917	162.6608	18	-7
24	Yabba Dabba Que!	9	161.1312	0.930592	162.0618	19	5
34	Handsome Devil LLC	3	154.8456	6.048192	160.8938	20	14
15	Three Dogs BBQ	1	165.0856	-4.21917	160.8664	21	-6
14	Bobby Q's BBQ	8	166.2516	-5.92893	160.3227	22	-8
23	The 5th Artery	7	161.6684	-1.37805	160.2904	23	0
20	RxCelient Q	4	162.8688	-3.44213	159.4267	24	-4
16	Meat@Slims	6	164	-4.70605	159.294	25	-9
27	Blackstrap BBQ	9	158.2284	0.930592	159.159	26	1
19	Pork Butt & Chicken Legs	1	163.3712	-4.21917	159.152	27	-8
21	Jack's Down Home Barbeque	4	162.2856	-3.44213	158.8435	28	-7
18	Primal Meat Smokers	8	163.9656	-5.92893	158.0367	29	-11
33	ZBQ	9	156	0.930592	156.9306	30	3
26	Grateful Smokers	10	158.88	-1.97237	156.9076	31	-5

44	Shake-n-Bake BBQ	5	145.7144	10.87115	156.5856	32	12
25	Shortsville Smokers	1	159.9772	-4.21917	155.758	33	-8
29	Garden State Porkway	10	157.7256	-1.97237	155.7532	34	-5
31	Big Kev's BBQ	1	157.6684	-4.21917	153.4492	35	-4
38	Divone Heat	2	149.6456	3.796752	153.4424	36	2
30	Hogbutts BBQ	6	157.7144	-4.70605	153.0084	37	-7
39	Carmel Street Pit Crew	2	149.0744	3.796752	152.8712	38	1
49	Packanack BBQ Club	5	141.6912	10.87115	152.5624	39	10
45	Hall of the Mountain Grill	3	145.1428	6.048192	151.191	40	5
50	Babylon GriliBillies	5	139.36	10.87115	150.2312	41	9
37	All Fired Up and kicking ash	7	151.3828	-1.37805	150.0048	42	-5
35	Hog Snipers	6	154.274	-4.70605	149.568	43	-8
48	Double "J"	3	142.7768	6.048192	148.825	44	4
46	Scarborough Fare BBQ	2	143.9084	3.796752	147.7052	45	1
41	Cheffie Effie BBQ	7	147.4172	-1.37805	146.0392	46	-5
42	Locked-N-Loaded	10	147.4056	-1.97237	145.4332	47	-5
40	Holihan's Beer Me BBQ	4	148.5372	-3.44213	145.0951	48	-8
47	Better Days BBQ	9	143.8968	0.930592	144.8274	49	-2
43	Pop Bang Foods	8	146.7884	-5.92893	140.8595	50	-7

In conclusion, we hit a particularly bad table, and as such I didn't feel any large changes to our recipe and process were necessary. In fact, by the end of the year we took 10th place chicken at the world championship American Royal Invitational barbecue contest, out of 174 teams! I shudder to think what would have happened if I only had access to the old score sheets. I would not have been able to do this analysis and most likely assumed my food needed changes. This could have been disastrous for the rest of the season. A little data analysis can go a long way!