DATA ANALYSIS OF THE 2021-22 @DOWNGOESBROWN NHL PREDICTION CONTEST

WILLIAM J TOWNSEND

BS Data Management & Data Analytics Capstone for

Western Governors University

Project Overview

A-1: Research Question/Organizational Need

Sean "@DownGoesBrown" McIndoe, a national hockey writer at The Athletic, runs an annual contest for readers (and other writers) to predict various outcomes and events of the forthcoming National Hockey League season. This project was designed to determine the contest winner and provide various descriptive statistics about the contest entries and outcomes by automating the process of scraping, parsing, standardizing, and scoring the nearly 1600 contest entries. This project also aimed to do this in a more efficient process than the laborious process of manual scoring that McIndoe had previously used.

A-2: Project Scope

This project's scope was to create a Python script which would automate the gathering of all entries, standardize them, and score them all. The final scored entries would be exported to a spreadsheet, to which some minor cosmetic changes would be made to enhance readability. In the process, the project would also generate a series of descriptive statistics detailing the answers for each question, which could be compiled in a slide deck. The slide deck and the final spreadsheet were then provided to McIndoe.

A-3: Solution Overview, Tools, Methodologies

The Python script requires as its sole input a saved .html copy of the 2021-22 prediction contest announcement, where contest entries were made in the comments. The script was built

using Jupyter Notebook in the Anaconda environment, and makes use of the Beautiful Soup, pandas, and NumPy libraries for Python. The resulting spreadsheet was modified and hosted in Google Sheets.

The input file is a hard requirement for this project, as it is the only source of contest entries for this year's prediction contest. While another programming language (and its associated libraries) could be used for the rest of the project, I am most comfortable with Python and was learning about Beautiful Soup at the time when the contest began. Beautiful Soup facilitates web scraping from HTML or XML tags, allowing for the gathering of both comment authors (contest entrants) and contents (contest entries), allowing me to easily pull this information programmatically. The pandas library allows for placing the scraped data into a dataframe, which is essentially a table in Python capable of sophisticated and efficient operations upon the data within. NumPy was used to populate Not-a-Number (NaN) values into the dataframe, along with providing some other mathematical functions. Google Sheets was used as a no-cost option for sharing the final data in a readable and sortable format.

Several methodologies were used in the entire course of this project, including in the project planning process (Agile), data collection (web scraping via Beautiful Soup), analytical (descriptive statistics), and statistical (inferential statistics and T-testing). These are explained in depth in the associated sections of this document.

Project Plan

B-1: Project Plan

This project was executed more quickly than anticipated in the project plan submitted in Task 2, the project proposal. This was due to the fact that the project was largely done, with the entirety of the data collection and standardization work being already done in Oct 2021, along with much of the descriptive statistics used in the final product. This made it very easy to demonstrate the practical significance of this project based upon work that has been subsequently produced by McIndoe using the data provided to him. As a result, only the automated scoring component of the script had yet to be completed, and this component was completed in a timeframe consistent with the estimate provided in the project plan. Generation of the slide deck and export of the data to a spreadsheet was also completed in a timeframe consistent with the budgeted estimates.

The only significant deviation that was not outlined as a part of the project plan was the approximately 5 hours spent on manually scoring contest entries to create a point of comparison between manual and programmatic scoring. The project scope was maintained without issue. The project had two deliverables:

- Spreadsheet containing all contest entries in standardized and scored form, cosmetically modified to enhance readability and usability by non-technical users
- Slide deck containing descriptive statistics regarding the contest entries (number of possible answers, answers actually provided, number of unique answers provided, etc.) and the contest outcomes (top scores overall, top scores without bonus question, distribution of correct/incorrect answers, etc.)

Both deliverables were provided to McIndoe with data up to date as of the submission of this capstone project on 23 Jun 2022. An updated version of each of these will be provided to McIndoe in mid/late July when the final NHL player transactions take place during the 2022

NHL draft and the start of NHL free agency for the 2022-23 season. This update will require only the addition of newly moved players to the lists of correct answers, and a subsequent execution of the script to reflect these final changes.

B-2: Project Planning Methodology

The modified Agile methodology (no standups or other collaborative elements) described in Task 2 was the methodology used in the course of completing the component steps of this project. This methodology called for a series of sprints to design each element of the final automation script:

- Scraping of contest entries from comment section of contest announcement
- Fixing of scraped comments as needed to allow parsing
- Generation of dataframe and import of entries, including granular answers
- Fixing of dataframe contents as needed to allow standardization
- Standardization of answers for all 10 questions
- Automated entry scoring system

The nature of the script necessitated a very straightforward approach in which each task had to be completed before the next could begin, because each task used the output of the previous task as its own input. Each task was completed in the course of its particular sprint, and a number of problematic contest entries were noted early on the process, which were checked throughout the process to make sure each was handled correctly by the parser and fixer functions, allowing the final standardization. The generation of the spreadsheet and the final slide deck were generated after the script was completed.

B-3 Project Timeline/Milestones

In Task 2 of this project, the following timeline for the project was created:

Project Task/Component	**Start Date**	**End Date**	**Projected Time Need**	**Milestone?**
Scraper	5/31/2022	6/1/2022	4 hours	
Comment Parsing/Fixing	6/1/2022	6/2/2022	12 hours	
Generating of Dataframe	6/3/2022	6/3/2022	2 hours	**X**
Dataframe Fixer	6/6/2022	6/10/2022	16 hours	**X**
Standardization Operations	6/13/2022	6/17/2022	24 hours	**X**
Automated Scoring System	6/20/2022	6/24/2022	16 hours	**X**
Slide Deck	6/27/2022	6/30/2022	6 hours	
Export Operations	6/27/2022	6/30/2022	2 hours	

The scheduled time estimates for each task were consistent with the time spent on each task. All tasks were completed well in advance of their Projected End Dates, as most of the project was completed in Oct 2021, and the remaining tasks were completed prior to the completion of Task 2 and the generation of this timeline. There were no significant overages, nor savings, on any of the tasks that were a part of the product timeline.

The one task that I failed to include in the timeline was the time that would need to be spent on generating estimates for how long manual scoring of the contest entries would take. The manual scoring of 160 entries (just over 10% of the entire 1582 contest entries) took nearly 5

hours, and then additional time was spent on using this information to generate a normal distribution based on this data for the purpose of determining the statistical significance of the programmatic solution's run time relative to this data. Altogether, this took approximately an extra 8 hours of work that was not planned for.

Project Methodology

A copy of the contest announcement webpage was made using Google Chrome (Right Click > Save As) at approximately 1930 ET on 12 Oct 2021, the time of the contest's closing to new entries. The Beautiful Soup library was used to parse the contents of the HTML page, pulling out the contents of the tags containing the id "parent-comment-container", which contains all elements of each comment made on the web page. The author of each comment was then extracts from the tag "comment-author-text" and placed into a list of authors, while the contents of each comment was extracted from the tag "comment-text-container" and placed into a list of comments.

Once scraped, a check is performed to verify the list of authors and comments are of equal length. As all contest entries are at least 8 lines long, comments will be checked for the number of new line characters they contain ("\n"), and any comment containing less than 8 lines has its index noted in a list. The noted indexes are then used to remove the comment and the associated author, in descending order to avoid moving the list's indexes as it is processed. Each author's name is checked for uniqueness within the list, and if a name is not unique within the dataset, the index of that author's position in the list of authors will be appended to the name.

An empty dataframe with column names is generated and parsing operations begin to fill the dataframe. For each comment, the comment is split by lines and has numerous checks performed against each line to verify the line does contain contest answers. If the line does not contain answers, the parser advances to the next line, until it identifies the first line to contain contest answers, at which point it begins a counter, which indicates which question (of 1 – 10) it is currently handling. The line identified as containing contest answers is then split into a list of the constituent answers. If the number of answers for that question is less than 5, then NaN is appended to the list until the length of the list is 5.

A new temporary list is created, beginning with the associated comment author's name, and each of the answers found is appended to that list. Once each of the answers for the question is appended, the question counter is incremented and the new question and associated answers are handled. Once the counter completes question 10, it stops parsing the comment, and appends the temporary list of author and answers to the dataframe. The parser then moves on to the next comment, until all contest entries are entered into the dataframe. At this point, the dataframe is populated with all of the collected data, and standardization operations may begin.

C-1: Advantages/Disadvantages of Data Set

For this project, there were not any alternative datasets that could be used, as this was the only one that would contain the contest entries for this year's contest. Scraping the data from The Athletic's website was simple due to their usage of clearly labelled containers for each comment and its component parts, making it easy for Beautiful Soup to extract the intended contents. As an avid hockey fan, the entirety of this dataset was largely common knowledge to me, making it

easy for me to understand the details of each entry, even where mistakes were made by the entrants themselves. This was especially useful in the process of standardizing the entries, whether entrants often made mistakes of content, rather than of formatting, and I was able to make a more generous interpretation of their intent based upon that knowledge.

In terms of limitations and disadvantages of this dataset, the biggest issue was the lengthy and often tedious data cleaning and wrangling process, which was mostly due to the unstructured and qualitative nature of the data. With all data consisting of user-submitted strings without no enforcement of standards or even format, this led to the majority of the project work being spent on addressing these issues. While there are potential improvements that could have been made to the code to reduce some of these issues, I do not believe there was any process that would've entirely eliminated these issues. Simply put, the breadth of possible human mistakes is wider than the possibility of programmatic solutions to handle those mistakes, without also mishandling other situations.

The other situation that complicated the development of this project was the contest's allowance for non-answers. For each question, each entrant was able to provide up to 5 answers for increasing numbers of points based on the number of correct answers, but they were not required to produce 5 answers for each question. In fact, for many questions where entrants felt less confident, this was a common tactic to get a larger chance at earning some number of points, rather than taking an unreasonable risk to get even more points. While this is conceptually simple and not an uncommon mechanic in any contest or game which involves a bid or wager of points, this created two complications for the programmatic handling of contest entries. First of all, it made the number of answers provided for each question into a variable rather than a constant, and this required evaluation and additional checks (both manual and automatic) to verify the

correct operation of the parser or to fill unused answers with NaN. The second issue created was evident in the autoscoring script, as each answer no longer had two binary possibilities (right or wrong) but instead had three possibilities (right, wrong, or not attempted). As a result of this, scoring of the contest entries became a more convoluted (and potentially less efficient) process than it was expected to be, requiring the generation of new columns from which the number of correct and incorrect answers for each question could be derived from, and then deleting the unneeded "intermediary" columns.

C-2: Data Selection/Collection Process Divergence from Plan

The process of selecting and collecting the data never changed from my intended plan for doing so. It did, however, require far more iteration than I'd ever anticipated when I begun working on the project, due to the obstacles that were encountered in the process, described below. This often involved making fixes to a comment which crashed the parser, and then re-running the parser and waiting several minutes to see where it crashed next, so that I could begin making fixes to that comment as well. This was a very slow process, as a result. Once I'd finished collecting the data and addressed these problems and observed which ones were most common, there were ideas which occurred to me to potentially address those problems without needing the specific intervention that was performed, but re-writing the scraping/parsing process to fix issues that were already fixed seemed a poor use of time for limited upside. If this scraper/parser were applied to another year's contest data, I would probably build in these changes, rather than going through a similar process of manually writing in programmatic intervention for these comments. Even with such a change, however, there would still be a number of comments that would need such direct intervention.

C-3: Obstacles Encountered in Data Collection

In the process of scraping and parsing the contest entries from The Athletic, a number of entries were not able to be parsed automatically, mostly because of their formatting. The parser depends on certain patterns to correctly handle a comment, and when these expected patterns were not followed, a crash of the parser would usually result. There were two main problems that the parser struggled with.

First, the parser depended upon delimiters between the provided answers to identify a line as containing answers and to divide those answers up correctly. These delimiters were most commonly commas, as in "Colorado, Tampa Bay, Las Vegas, St. Louis", though "/" was often used by entrants as well, and yet more entrants would insert the word "and", sometimes with or without an Oxford comma ("…, Las Vegas, and St. Louis" or "…, Las Vegas and St. Louis"). These were all handled correctly by the parser through various replacement functions, but the biggest problems arose when entrants used no delimiter, or the entrant used a period as a delimiter. In the case of a period as a delimiter, the string "Colorado. Tampa Bay. Las Vegas. St. Louis" would be read as five different answers ("St." and "Louis" becoming separated). In the case of no delimiter whatsoever, "Colorado Tampa Bay Las Vegas St. Louis" would result in 7 different answers: "Colorado", "Tampa", "Bay", "Las", "Vegas", "St.", and "Louis". These cases were difficult to handle safely through replacement functions of the period or the space throughout all entries, so it was decided to just handle these issues with a specific intervention to fix the offending comment. This was sometimes done through replacement of the problematic punctuation, but many situations required the comment to be rewritten in a parsable format.

The other common issue that arose was entrants prefixing their contest entry with an amount of commentary. The parser searched for the first line within the comment containing a recognized delimiter (comma, slash, etc.), using this to begin parsing operations and separating out answers. If a comment started with a line that did not contain such delimiters, such as "WJT contest entry", the parser could move past that and correctly find the first submitted answer in that comment. However, if the comment began with a lengthy string that used such delimiters, such as "This is William T's contest entry, I'm so excited for this contest, I think I'm going to win", that string would be accepted as the answer for the first question, and the subsequent lines would be processed as the second question, third, and so on. There was no way to easily account for all the permutations of such comments, so each of these required a specific intervention to fix the offending comment as well. This was generally done by using string slicing to update the comment to exclude the offending portion, but some comments required full rewriting if such commentary appeared throughout the entry or if other issues cropped up in addition to the commentary preceding the contest entry.

In total, 67 entries required such intervention to avoid being handled incorrectly or crashing the parser. The comment fixer function programmatically addresses these entries, rewriting problematic elements into a format capable of handling by the parser. This took care of the majority of parsing issues, though several other "one-off" issues were also encountered and handled in this function as well.

After parsing was complete and the data was placed into a dataframe, there were further issues of a smaller nature which needed to be corrected as well. Overwhelmingly, this involved issues of inconsistent delimiters, such as inserting additional delimiters into an entry ("Colorado,, Tampa Bay,…"), ceasing to use delimiters in the entry ("Colorado, Tampa Bay, Las Vegas St.

Louis"), or most commonly, inserting extra periods into the entry instead of commas ("Colorado. Tampa Bay, Las Vegas,…"). Another 117 entries required this sort of intervention, which was performed within the dataframe fixer function.

In total, 184 contest entries required special intervention to ensure their answers were correctly tabulated, before any standardization operations were performed. This was the most frustrating and time-intensive element of building this program, and it is almost entirely a consequence of the format involving people typing unstructured contest data into the comments field, especially when many users do so on the mobile The Athletic app. The use of mobile introduces issues such as auto correct of player names, inserting periods where a sentence is believed to have stopped (such as making two spaces), or the common typo of one's right thumb hitting the period rather than the comma (or both).

C-4: Unplanned Data Governance Issues

There were no data governance issues that arose during the project. As previously established in Task 1 and Task 2 of the project, all of the contest entries were entered into the public by the contest entrants, and any subscriber to The Athletic could view or copy these comments. Comments are pseudo-anonymized by The Athletic to the form of first name and last initial (for example, William T) and pseudonyms are even used by some subscribers. As a result, there is no sensitive personal data here that would require special consideration.

D-1: Data Extraction & Preparation Processes

With all of the contest data loaded into a dataframe and problematic entries given specific intervention to correct them, the process of preparing the data for analysis could begin. This consisted of building a series of dictionaries for each question which could take an existing contest entry answer (the dictionary key) and be translated into a standardized form of that answer (the dictionary value) by using the dictionary's key/value pairing. While all entries were converted to lower case in the course of their handling, every single encountered permutation of a name had to be handled, whether that stemmed from a misspelling, the inclusion of additional detail, the use of a city name instead of a team name, or even an autocorrect error. For example, all of the following contest answers were standardized into "tbl": "tampa bay lightning", "tampa bay", "tb lightning", "lightining", "lightening", "lightnings", "tgl", "tamp bay", as well as several other similar permutations.

This process got considerably longer for player and staff names, especially those with non-English names (of which there are many in the NHL), those with particularly difficult names to spell, those whose names use punctuation, or any combination of those factors. For example, Rod Brind'Amour, head coach of the Carolina Hurricanes and a very famous player from his own playing career, had 62 different permutations of his name written in the contest data (again, without considering upper/lower case), all of which were standardized to a single "brind'amour" outcome. Tampa Bay Lightning General Manager Julien BriseBois had 69 different permutations of his name, not counting the number of people who added a hyphen to his name, which required specific intervention in the dataframe or comment fixing functions prior to this standardization. Taking the cake in this area was Tampa Bay Lightning goalie Andrei Vasilevskiy, with an astonishing 86 different permutations of his name that had to be standardized to a single "vasilevskiy" outcome.

Once the dictionaries were built, the standardization operation would loop through each of the 5 columns for a question, applying a replace function to find the existing key in the dictionary that matched the submitted answer, and replacing it with the associated value in that dictionary. This was done for all 10 questions, and the entire standardization operation utilized approximately 3800 lines of code, almost all of which consisted of these dictionaries.

Once standardized, the autoscoring function could begin its work. Correct answers were found for each of the 10 questions, and a list was created for each question containing those correct answers. A "number of correct answers", "number of incorrect answers", and "question score" columns were generated for each question and inserted to the dataframe, as well as a "score" column for questions 1 – 9, and a "total score" question which included the bonus question (question 10) score, for which an incorrect answer could set the entrant's total score to 0 regardless of their score to that point.

The autoscorer then generates a new column (e.g., 'q1a1score') for each column it operates on (e.g. 'q1a1'), where it uses a nested NumPy where() function to take one of three actions based upon the value found in 'q1a1'. If the value found is NaN, then the new column is populated with NaN as well. If the value found is also found within the list of correct answers for that question, then the new column's value is 1. If neither is the case (meaning the submitted answer is neither NaN nor correct – thus, it is an incorrect answer), the new column's value is 100. This process repeats for all 5 columns that are a part of that question (e.g. 'q1a1', 'q1a2', 'q1a3', 'q1a4', 'q1a5'), creating 5 new columns that have a numeric value (or NaN) based upon the originally submitted answer. The sum of these 5 new columns is calculated into another new column. For example, an entry with 2 correct answers and 2 incorrect answers (and only submitted 4 answers) will have a resulting sum of 202.

The number of correct/incorrect answers for a question are then reverse engineered from this sum, and that information is used to create a score for the question. The count of correct answers is derived by finding the modulus of the sum when divided by 100 (e.g. 202 % 100 = 2, so 2 correct answers). Similarly, the count of incorrect answers is derived by using integer division of the sum divided by 100 (e.g. 202 // 100 = 2, so 2 incorrect answers). This information is placed into the appropriate columns that were already inserted into the dataframe previously, and a score is calculated, using a series of nested NumPy where() functions. If the number of incorrect answers is greater than 0, then the score for that question is set to 0. If that is not the case, then the function tries to match the number of correct answers to 0, 1, 2, 3, 4, or 5, and awards 0, 1, 3, 6, 10, or 15 points respectively based on the number of correct answers, placing that point value into the question score column.

This process is repeated for each of questions 1 – 9, as each has 5 possible answers. Question 10, the bonus question, is entirely optional and accepts only a single answer. If answered correctly, this question provides 15 bonus points, but if answered incorrectly, this question sets an entrant's score to 0. This was also handled using another nested NumPy where() function, setting the 'q10score' to NaN if the answer for question 10 was NaN (not attempted). If the provided answer was found in the list of correct answers for question 10, then the score is instead set to 15, but in any other case besides a correct answer or a non-answer, the score is set to 0.

Totals are then created for questions 1-9, summing up the number of correct answers across these 9 questions, the number of incorrect answers, and the total score. A final 'total_score' is then created using another nested NumPy where() function. If the score for question 10 is NaN (not attempted), then the total score is set to equal the total score for

questions 1-9. If the score for question 10 is 15, then the total score is set to the sum of the scores for questions 1-9 and question 10. In any other case (question 10 was attempted and the score received was 0 due to an incorrect answer), then the total score is set to 0. With all scores generated, the dataframe then drops the intermediary columns that were used to generate the final counts and scores for questions 1 – 9.

This process struck me as not being the most Pythonic solution, and the repeated use of nested NumPy where() functions is somewhat convoluted to follow. However, the function executes very quickly across all results (%%timeit in Jupyter Notebook indicates that this particular element of the code takes approximately 100 - 180 ms to execute for all entries, scoring all answers) and works as intended. The complexity of this process is a result of the non-binary nature of the contest answers, in which an answer can be not just correct or incorrect, but also may simply not exist and should do so without penalty to the entrant. There may be a more elegant solution, but the solution was clearly effective enough that it was maintained and commented with commensurate detail to explain the complexity of the code used.

E-1: Data Analysis Methods

The primary analytical method used in this project was descriptive data analysis. This was most appropriate for this project because the interests with this project were in standardizing and analyzing historical data to see what happened. While the contest itself was geared towards predicting the future, the contest as a device is used by McIndoe to discuss what has happened both within the NHL season and how that compared to what his readership (as well as himself and some of his peers) thought would happen during the 2021-22 NHL season. Those are past-

tense questions which can be answered by using descriptive analysis of what has previously happened, and this project aimed to complete that analysis so that McIndoe could use it to tell stories about the contest. Much of this was done through data aggregation, such as aggregating all provided contest answers for a given question to reach a conclusion, such as "Of 1580 entries, 1505 chose the Las Vegas Golden Knights to make the playoffs". This observation, among many others, was then provided to McIndoe for the purpose of deciding how (and if) to use it in his work relating to the contest.

E-2: Advantages/Disadvantages of Analysis Tools & Techniques

This project was put together almost entirely in Python, chiefly using the Beautiful Soup, NumPy, and pandas libraries. This is the programming language in which I am most proficient, and I was familiar enough with NumPy and Pandas that I had very little issue building the program's functionality within those two libraries. This project initially started out as a way to learn to use Beautiful Soup's web-scraping functionality, and in that regard, it was very successful. The use of Jupyter Notebooks also made it very easy to iterate through and update my code, by breaking it into distinct cells that could be rerun, rather than having to execute the entire script over and over. Combined with using the MatPlotLib library, I was able to build dynamic slide decks for McIndoe and visualizations that were useful for both him and this capstone project. With regard to limitations of these tools, the main issue that I ran into was the maintenance of the stand-alone .py script against the Jupyter Notebook. The Notebook facilitated more rapid iteration and exploration of the data, but the singular Python script was necessary to provide a single-file solution to the entire project. I ended up doing all of my development in Jupyter Notebook and only transferring code into the Python script at the very end of the project,

where each would deviate slightly – the Notebook would generate the completed spreadsheet, while Notebook would generate the slide deck.

This project almost entirely revolved around qualitative content analysis. The distribution data for answers provided by entrants in the prediction contest and the eventual outcomes for those answers. This allowed me to look for patterns in the content, such as the frequency of ideas ("Nathan MacKinnon will score 100 points this season") or outcomes ("Almost all entrants got question 3 wrong"). Providing these observations to McIndoe, along with the raw data for him to explore and make his own observations, is key to allowing him as an author to craft a story around those observations. The limitation of quantitative content analysis is that it is very time consuming. While the script is able to scrape, record, standardize, and score entries very quickly, the unstructured quantitative data that I was exploring was so low-quality that I had to spend an inordinate amount of time on data cleaning & wrangling operations to facilitate this sort of analysis. Once completed, the nature of that data required a very broad and time-consuming analysis. This was further underlined by the fact that this data was being provided to someone I respect and whose work I enjoy, because in using my data analysis in his writing, McIndoe was to some extent trusting his reputation to the accuracy and completeness of my work in performing this analysis.

E-3: Application of Methods to Data

The analysis of the data was performed in three distinct phases. The first phase was analysis of the contest entries themselves, performed shortly after the contest closed to new entries in Oct 2021. This process included determining the total number of contest entries

(dataframe length) and aggregation of the answers for each question to generate descriptive statistics. The value_counts() for each of the 5 columns for a single question would be aggregated into a single sum of all value counts for that question, printing out a very clean table listing the frequency with which each unique answer was picked for a particular question. The number of unique answers is provided by finding the length of the series of aggregated value counts. The number of NaN values (unattempted answers) is also counted, and compared to the total number of possible answers (length of dataframe x number of possible answers – 5 for questions 1-9, 1 for question 10). These statistics were placed into a slide deck, along with any other observations or clarifications about the data that I found interesting, and this slide deck was provided to McIndoe on 21 Oct 2021.

The second phase of analysis concerned the contest results. After the entries were scored in the process outlined above, a new series of descriptive statistics were generated. For each question, value_counts() was used to generate three tables for each question: the distribution of scores, distribution of correct answers, and distribution of incorrect answers. The top 10 contest entries by total number of correct answers on questions 1 – 9 were also displayed, sorting by this value and using head() to show the desired entries. The same process was used to generate a top 10 contest entries by number of incorrect answers on questions 1 – 9, as well as top 10 by score on those questions. Finally, a top 10 overall contest entries by total score, including the bonus question, was generated from the same process. A histogram was also generated, showing the distribution of all overall contest scores. Again, any observations that I found interesting or clarifications about the data were also generated, and this was placed into a slide deck and provided to McIndoe.

The final phase of data analysis for the product was the comparison of time spent scoring the entries with the new automated process versus the manual process. The time spent on the automated process was handled by using the time module in Python to create timestamps at the beginning of the code (prior to importing the necessary libraries, besides 'time') and at the end of the code. The difference between the timestamps was the run-time and was printed out to the console. To ensure that the result for the automated time was not somehow an outlier, the script was executed in its entirety (from scraping the data all the way to scoring it and generating the descriptive statistics) five times, with each result being recorded and the mean of the 5 attempts used as the time spent for the automated process.

For calculation of the manual scoring process, I created a series of spreadsheets on which 10 entries would be recorded and scored from the web page containing the original contest entries. This process would not involve entering every single answer provided by a contest entrant (up to 46 total answers), but instead would only involve recording the entrant's name, their number of correct answers, their number of incorrect answers, and their score for each of the 10 questions. This would be followed by a total score, which would be the sum of the scores for all 10 questions, unless question 10 was answered incorrectly, in which case the total score would be zero. For each set of 10 entries, I would start a stopwatch on my phone and begin scoring 10 consecutive entries on the contest web page, without stopping or interrupting my work. After the final entry was scored, I stopped the stopwatch and recorded my name and the time taken on the spreadsheet.

I had originally planned to have several different people score 10 entries on a spreadsheet, but I realized that a scorer who was less intensely familiar with the data as myself would take considerably longer to score 10 contest entries. As this process was intended to

simulate the time taken if McIndoe manually scored every single contest entry, I decided that the most equitable comparison would specifically require me to handle each of these manual scoring attempts based upon my close familiarity with the dataset and the correct answers. With just under 1600 entries in the contest, 16 different manual scoring attempts of 10 entries each were performed, all using unique comments and thus comprising over 10% of the entire population of contest entries.

All of the data for the manual and programmatic contest evaluation was compiled in a Google Sheet, visible here.

Project Results

F-1: Statistical Significance

One of the goals of this project was for this software-based approach to work more efficiently at evaluating the contest entries than the manual process that McIndoe had previously used for evaluating the contest entries. To this end, a null hypothesis was formed which stated that the manual solution would be equally as fast or faster than the programmatic solution, and the alternative hypothesis was that the programmatic solution would be faster instead.

$$\mathbf{H_0: time_{(program)} - time_{(manual)} \geq 0}$$

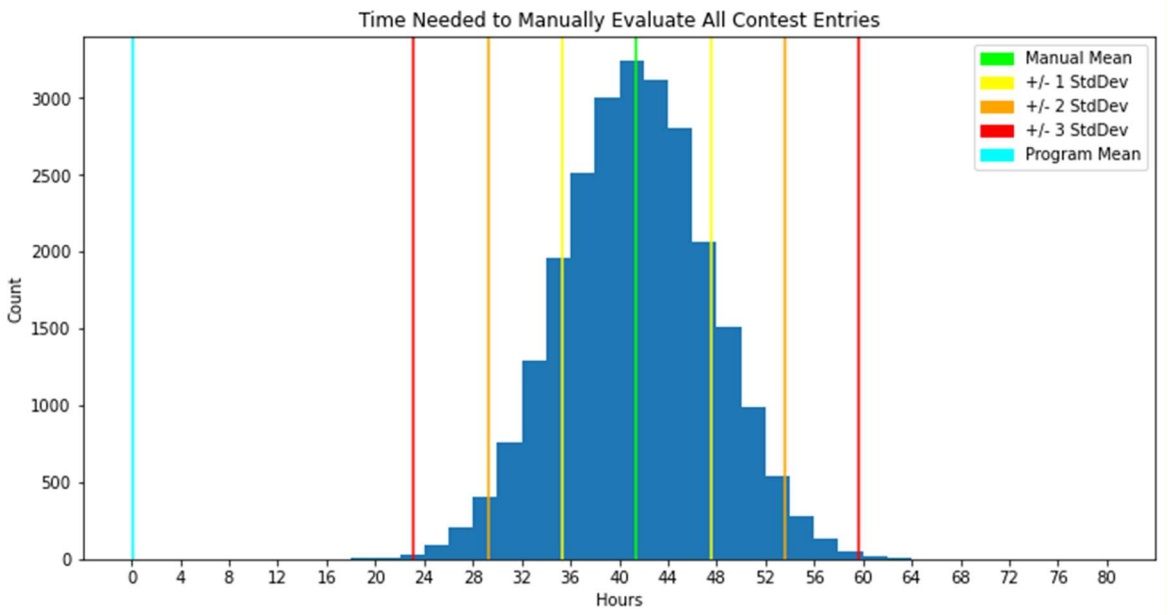
$$\mathbf{H_1: time_{(program)} - time_{(manual)} < 0}$$

As described above, a mean was derived for the manual evaluation process. 10 entries were evaluated by hand into a spreadsheet, tracking the number of correct and incorrect answers for each question, the score for each question, and then an overall score. This process was

repeated 16 times, and each process used 10 unique entries, so that 160 entries of the full 1580 dataset were manually evaluated. The time needed for each attempt was multiplied by 158 to calculate the time needed for the entire dataset's manual evaluation, and a mean was calculated from the resulting 16 values of approximately 41 hours, 20 minutes, and 25 seconds, or 41.34 hours. As it relates to the hypotheses described above, this value would be $time_{(manual)}$. The standard deviation for this data was 6 hours, 6 minutes, and 46 seconds, or 6.11 hours.

To determine the value of time(program), I used the Python time library to determine the total runtime of the script from start to finish – scraping the entries, fixing problematic entries, standardizing all of the answers, and evaluating them. This process was repeated 5 times, and the mean of these runtimes was found to be about 5 minutes, 45 seconds, or 0.095 hours, with a standard deviation of less than 0.001 hours.

Using the manual mean and the manual standard deviation, a normal distribution was created with the NumPy library, simulating 25,000 manual evaluations of the entire contest data of 1580 entries. The mean and standard deviation of this simulation were verified to be very close to the values previously derived by the manual scoring process. This data was plotted and a series of lines were plotted atop it:



The green line highlights the mean of this distribution, approximately 41 hours and 20 minutes. The yellow, orange, and red lines signify distances of 1, 2, and 3 standard deviations from that mean. The light blue line signifies the mean time taken for the Python script to programmatically scrape, standardize and evaluate all 1580 contest entries. Using the mean program and mean manual data, it is apparent that the alternative hypothesis is correct: substituting 0.2 hours for the program time and 41.34 hours for the manual time, this evaluates to less than 0, supporting the alternative hypothesis. The above plot supports this, and the distance with which the programmatic solution is outside of any expected result for the distribution of time taken for manual evaluation suggests that this result is statistically significant.

However, these two conclusions are only indications of a conclusion. To thoroughly disprove the null hypothesis, it is necessary to go further and calculate a z-score to determine the number of standard deviations the observed datapoint is from the established mean. A p-value is then determined, indicating the probability of finding such a result within the distribution. That

p-value will then be compared to the confidence level (α) of 0.01 that was declared in the project proposal. A p-value less than α indicates that the result is statistically significant.

The z-score was calculated by evaluating (0.095 - 41.34)/6.113. This results in a z-score of -6.73. Using the p-value calculator at socscistatistics.com for a one-tailed hypothesis and α of 0.01, a p-value is returned of < 0.00001. This is smaller than 0.01, and as a result, the result is deemed to be statistically significant. As a result, the null hypothesis is rejected and the programmatic approach to evaluating the contest entries is determined to be more efficient than the manual approach.

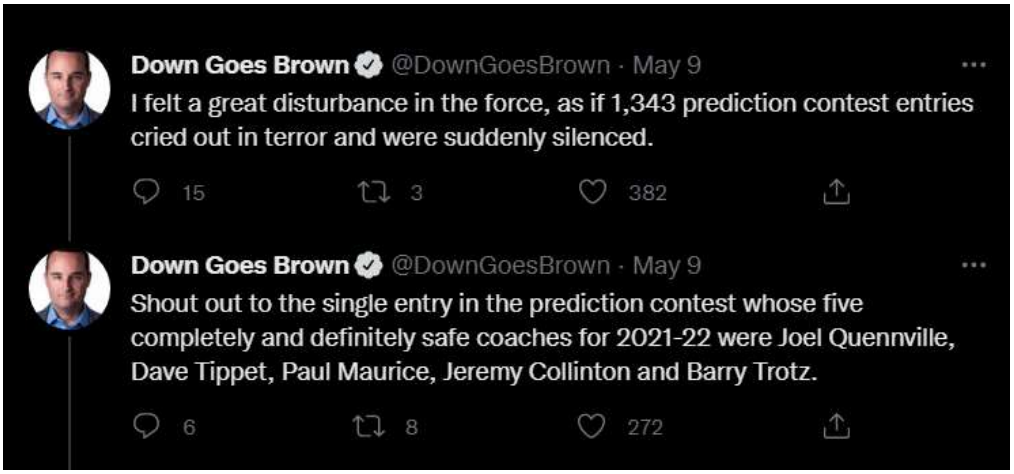
F-2: Practical Significance

The most obvious practical impact of this project and the software-based approach to evaluating the contest entries is that it saved McIndoe approximately 41 hours that would have been required to score every contest entry. In actual practice, the time spent manually evaluating contest entries could have been reduced further by simply grading question 10 first and cancelling out the 509 entrants (nearly one third of the total) that answered it incorrectly. This would require that the contest entries not be examined until after the conclusion of the NHL regular season, making the data unusable as a prompt for content until the playoffs, when such prompts are already abundant. This approach would've also involved deliberately taking a less detailed approach to the evaluation of the contest entries. For example, an entry that scored a perfect 135 points on questions 1 – 9 but failed on question 10 to earn a final score of 0 would be unobserved under this approach. Searching for the "near miss" is a part of the fun of this sort of

contest, and that is evident in the time that McIndoe spent on covering such near misses in last year's contest (McIndoe, 2021a).

Perhaps an even larger practical impact than the actual time saved is the amount of detail that the script was able to provide regarding the contest, and how quickly it was able to do so at various points in the season. The standardization of contest results allowed for the generation of descriptive statistics about the contest entries themselves (most common answers, number of entrants, questions which had more unattempted answers, etc.), and this was provided to McIndoe within two weeks of the start of the NHL season. This information was only possible to generate because of the standardization operations, and an attempt to replicate this manually to build a spreadsheet similar to the one created by this script would've taken much longer than the 41 hours that evaluation of the entry was projected to take.

Part of the point of this contest was the creation of content, and the extra analysis and detail provided by this script made the contest even more useful in that regard. Using this analysis, McIndoe was able to write on multiple occasions throughout the season about the contest. This showed up as "throw ins" added to existing articles (McIndoe, 2021c), and this data was also used to generate an entire article talking about the contest and how it was going, heading into the last month of the season (McIndoe, 2022). This also provided prompts for engagement on social media throughout the season. For example, when New York Islanders head coach Barry Trotz was abruptly fired in one of the most surprising moves of the 2021-22 NHL season, McIndoe tweeted:



The ability to count how many entries contained Trotz was only possible with the standardized data, because there were 17 different forms of "Trotz" in the raw contest entries. The provision of the complete spreadsheet of all standardized data also allowed for the finding of the only entry which had managed to get all 5 answers wrong on that question, as each of those 5 coaches were replaced. The usage of this data throughout McIndoe's work in the 2021-22 season demonstrates that this script and the analysis it provides was very practically significant.

F-3: Overall Success & Effectiveness

In the project proposal (Task 2), four criteria were chosen to determine if this project was a success:

- ✓ All contest entries must be scored correctly
- ✓ Delivery to McIndoe of a spreadsheet containing all contest data
- ✓ Delivery to McIndoe of slide deck(s) containing descriptive statistics defined in the project scope

- ✓ Statistical determination of the software-based entry handler being faster and more efficient than a manual scoring process

Each of these criteria was completed successfully. Scoring was performed correctly, as several tests of different edge cases were performed to make sure that these would be handled correctly. This was further verified by checking a number of the entries to ensure scoring was performed correctly, while I was exploring the finished dataset and results. The deliverables were provided to McIndoe, and a final update will be provided in mid/late July when the new NHL league year begins for 2022-23. Testing of the software-based process showed that it was far more efficient than the manual process, while also providing much more detail and information, enabling more content creation. The solution was undeniably statistically and practically significant, and it did everything that McIndoe needed it to do. Truly, there's not much more that it could've possibly done.

This entire project is ultimately a tremendous success, in my judgment. This is especially exciting for me on a personal level because it started out as an experiment of sorts, before growing into a freelance project that ended up becoming the capstone project for my BS in Data Management & Data Analysis. Right before McIndoe's announcement of this year's prediction contest, I had just finished a class on web scraping in Python, which had introduced me to the Beautiful Soup library. When I saw the contest, I immediately wondered if I could use Beautiful Soup to pull the data, and once I started making progress with doing so, I quickly recognized the necessary steps to make this into something useful. From there, the project organically grew into what it has become, and while it was far more work than I would have anticipated at the start, I am proud of the results.

G-1: Conclusions

The breadth of the data within this project, and the detail with which it has broken down to the individual question and entry levels, allows for numerous conclusions to be drawn, far more than could be discussed here. Additionally, the contest data is not completely finalized, as player transactions at the 2022 NHL draft and the start of 2022-23 NHL free agency will still impact the scoring question 9, and consequently, the overall total scores for the contest. However, at the big picture level, we can draw several conclusions from this project, even without that last update of the contest data:

The programmatic evaluation is superior to the manual evaluation: One of the criteria for project success was the superiority of the software-based process, and as outlined in the statistical and practical significance portions of this report, the programmatic evaluation is much faster than the manual one, while also providing tremendous detail. This extra detail makes the contest even more useful as a prompt for content generation, which is borne out by McIndoe's increased use of the contest data this year relative to the previous year's contest.

The python script correctly scores contest entries: This was the other significant criteria for project success, aside from the deliverables (spreadsheet and slide decks) to McIndoe. The script correctly scores entries in accordance with the rules of the contest. More impressively, the script correctly handles dozens of edge cases with entries without issue, including but not limited to:

- Exclusion of comments which were not contest entries
- Irrelevant line breaks or commentary within entries
- Inconsistent/non-existent delimiters among submitted answers

- Selection of ineligible or even non-existent submitted answers
- Handling of non-English characters in submitted answers
- Skipped answers or even entire questions
- Misspelled names

On every occasion in which contest entries were checked for appropriate handling and evaluation throughout the iterative development process, these entries were found to be handled correctly – or if they were not, changes were immediately made to correct the issue. In performing the deeply involved process of analyzing the provided contest answers and the contest outcomes, no errors were found or evident in the results of the scripting, and all products of the script are behaving as expected.

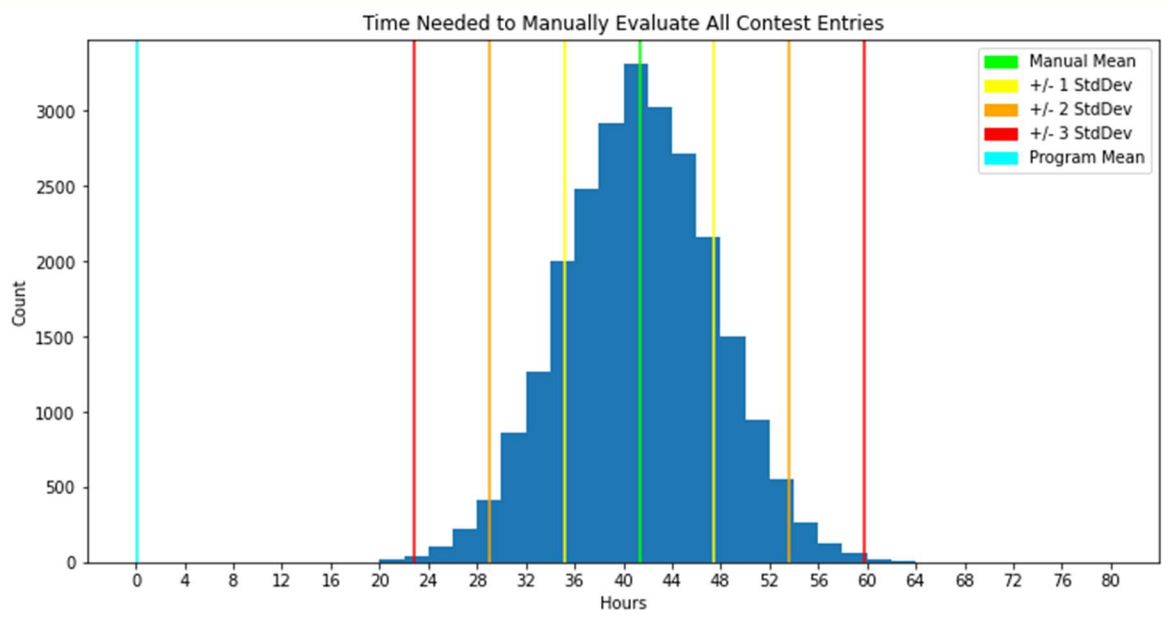
The CSV and slide deck(s) were provided to McIndoe: The remaining two criteria for project success was the provision of the deliverables to McIndoe. This included the export of the complete contest data (entrants, all answers, and all scoring) to a spreadsheet for his own use and potential sharing with readers (at his discretion), as well as the generation of slide decks which provided the qualitative data analysis of the contest data, which were prepared in Jupyter Notebook.

The generation of various descriptive statistics about the contest, including the winner: In handling the standardization and scoring of the prediction contest, the script also easily sorts through columns to present the winning entry. This functionality, among others, was actually built into the slide deck as well to operate dynamically, such that the slide deck will automatically determine the winning score (or, in the event of a tie, scores) and return the data relevant to that entry, without further modification to the slide deck from myself. Other high-level information gained from this analysis includes, but is not limited to:

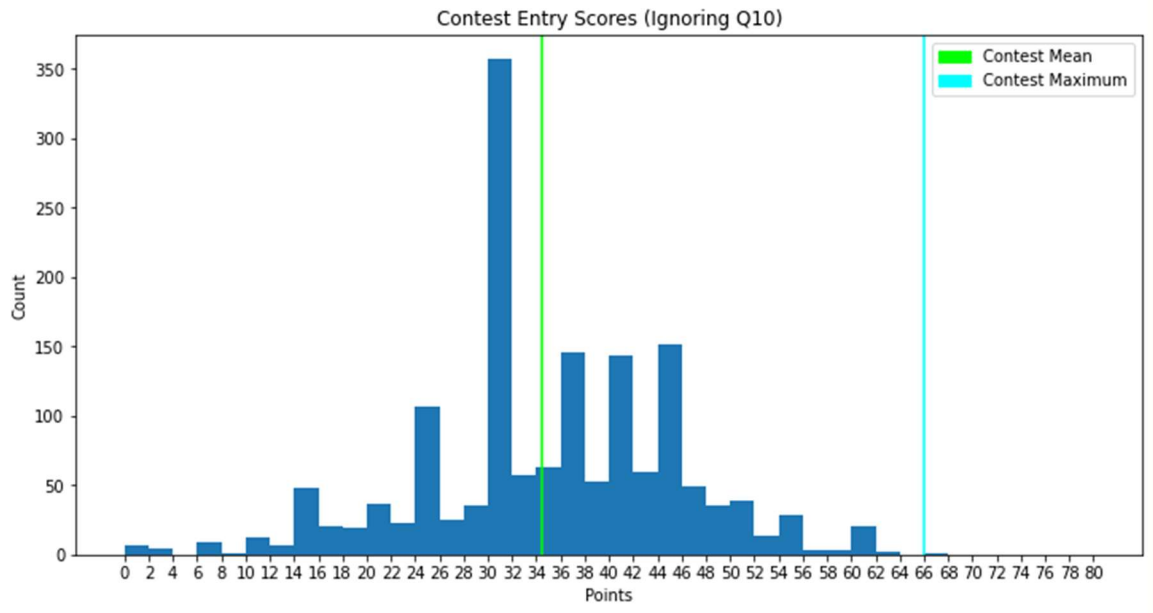
- The number of contest entries: 1580 entries
- The mean of contest entries: 30.18 points
- The standard deviation of contest entry scores: 23.39 points
- The high score: 76 points, reached by 3 entries
- The most correct answers provided: 36, accomplished by 1 entry
- The fewest correct answers provided: 6, accomplished by 1 entry
- The most incorrect answers provided: 22, accomplished by 1 entry
- The fewest incorrect answers: 2, accomplished by 6 entries

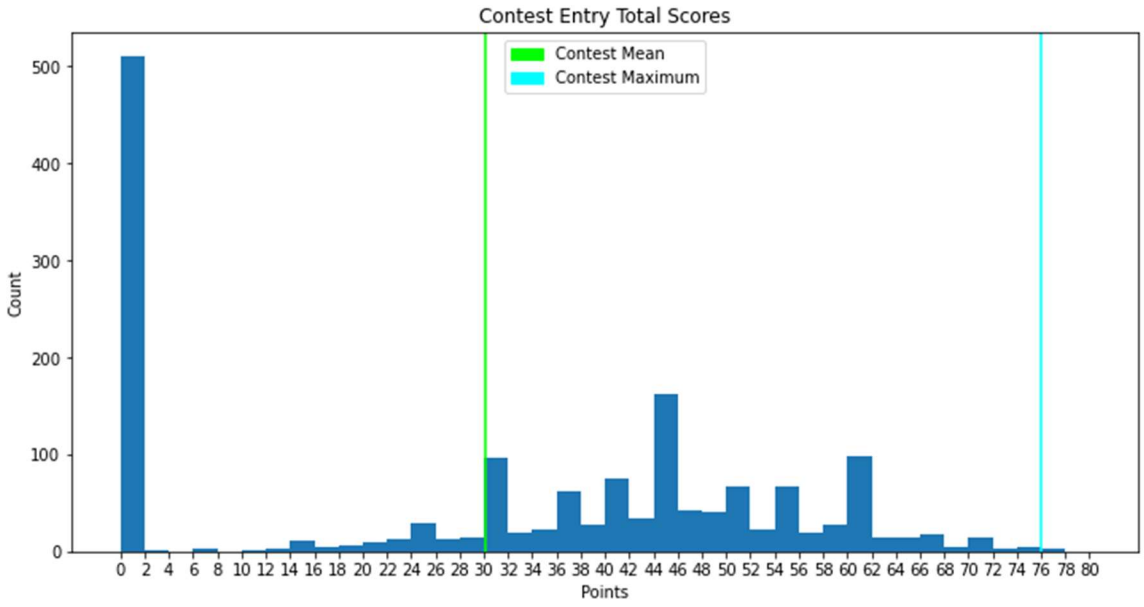
G-2: Visualization & Storytelling

The primary data visualizations within this project are the graphs regarding the time comparison (manual vs software) and the overall contest scores, as well as the tables of data regarding the distribution of answers on each question.



This graph of the time comparison between the manual and automated approaches to evaluating contest entries is abundantly clear – the automated approach takes very, very little time when compared to the potential amount of time spent on a manual approach. While the estimate of approximately 41.33 hours to perform manual scoring for 1580 entries could be quibbled with in some ways, this graph makes it emphatically clear that the difference between the two processes is so vast that the manual process simply cannot be remotely competitive with the automated process.





The graphs of the contest scores demonstrate the overall trends within the contest. These two graphs in conjunction show the impact of Question 10, which entrants could choose to answer for 15 bonus points, at the risk of zeroing out their entire entry. While nearly a quarter of entrants refused to try this question, over 40% of those who did attempt it got the question wrong, resulting in a total score of 0. This questions' existence changes the distribution of scores significantly, as nearly a third of all scores get moved into the "0" bucket while two-thirds of the remaining entries were advanced 15 points along the x-axis. This has some value for McIndoe, as the inclusion of Q10 and its "all-or-nothing" nature was a new inclusion for this year's prediction contest.

The total score graph is also informative to both McIndoe and his readers. Readers can compare their own score (or what their score would have been, had they not been zeroed out on question 10) by comparing it to the overall distribution of scores for the entire contest. This can also provide some context for future contest participants, as it can provide some context of what score is likely to be necessary to win, informing decisions on aggressively taking risks vs being

more conservative in their approach. These scores and the contest's basis as being "easy" based on it being about picking the "sure thing(s)" for each category also provide a prompt for McIndoe regarding the unpredictable nature of the NHL season.

The tables of data provided throughout the slide decks are not always necessarily thought of as a data visualization element, but they do also clearly communicate important elements of the story. The data in those tables speaks to the overall perceptions of the McIndoe readership with regard to what they think is a "sure thing" (high frequency of picks in the contest) and what they think is specifically not (low frequency of picks). While some of this data could have potentially been placed into graphs instead, the table format felt like the best way to present information in a clear and succinct way for McIndoe to choose how to weave them into a story. McIndoe's articles (McIndoe, 2021c, 2022) providing contest updates throughout this season have directly demonstrated the story telling value of these visualizations by using that data throughout each article.

G-3: Recommendations

This project was largely built around the nature of the contest itself, especially the unstructured nature of the entries being collected as comments on the article without any particular format. Some portions of the script that were created as a result of this could be reused for future years. For example, the standardization of team names would remain the same, as would many of the coaches, general managers, and players (Andrei Vasilevskiy will still be a high-end goalie next year and having already found 86 different permutations of his name this year, relatively few new permutations could be created next year that aren't already accounted

for). However, the lengthy process of fixing poorly or inconsistently formatted comments and entries is still a very lengthy and complex one that consumes a lot of time and requires lots of specialized code operations. All of this was complicated somewhat by my commitment to not remove any entry and to make all of them work, even if they failed to follow the contest directions or made unreasonable errors in their entry.

Additionally, it could be useful to create a means by which McIndoe's audience could easily view the contest results, whether looking for their own result or looking for interesting results themselves. This first portion of this was addressed by McIndoe's suggestion to entrants that they "tag" their entry with their initials or something, allowing them to CTRL+F to find their entry, but this created the issue of adding more unstandardized and unwanted data to the contest entries. This was also unnecessary – The Athletic displays any comment made by that user as coming from "You", rather than your name, so every entrant could instead just CTRL+F for "You" and find their entry. The second portion of that issue could be solved by sharing the spreadsheet I provided him, but that simply is not the most user-friendly solution. In addressing each of these issues, I have two primary recommendations for future years of the prediction contest:

Standardized form or format for contest entries: Eliminating (or at least reducing) the unstructured nature of the contest entries goes a long way towards making any handling or analysis of the entries much easier. An external form, such as through Google Forms or another service, could work to restrict entrant inputs, especially if an auto-fill mechanic were used to help users find the player whose name they were trying to spell. This would probably be the most effective approach, but the requirement to leave The Athletic's comment section (especially on mobile) likely reduces click-through and the number of entrants as a result.

An alternative that might compromise the two would be to provide a specific format for entrants to use by copy/pasting it into the comments, accompanied with a rule that any entry not using this format would be eliminated. This could standardize prompts for each line of answers, allowing the parser to easily identify each line in a comment. The provision of a "pass" answer of some sort would also standardize the length of each line, simplifying the parser's operations, while a requirement for a specific delimiter (a comma or slash, ideally) would mitigate numerous formatting problems while also facilitating easy parsing of the entry. This would force all entries into an easily parsable format, and entries which disobeyed the rules would be eliminated, rather than requiring intervention to fix them. Standardization of answers would still be necessary, but the ability to use existing dictionaries from this year's script would reduce the negative impact of this.

Web-hosted contest results: Publication of the contest results to a very simple website built for the purpose would allow a new level of engagement with the audience to examine those results. There is a reasonable argument that maybe it would be best to keep the entirety of the results inaccessible so that the engagement goes through McIndoe with his audience, rather than amongst the audience members themselves, but as I am not a content creator, I am not sure what the best approach would be in that regard. Nonetheless, this could be used to easily provide a contest entrant with their own results, and even if the entire dataset were kept publicly inaccessible, it could be made accessible to the contest operator – McIndoe, in this case. It would also be possible to build a dashboard that would allow for "power searching" of results in addition to the basic descriptive statistics that could be automatically generated. This entire enterprise could also go hand in hand with using a web-based form for contest entry, if desired. Once built, this could be used to operate other similar contests inspired by the McIndoe

prediction contest, such as this one for the 2021-22 NFL season run by The Athletic's Bo Wulf, an NFL writer primarily covering the Philadelphia Eagles. Building such a website could even prove to be a worthwhile commercial venture, because a fee could be charged for running a contest on this site in exchange for the time savings and statistics that such a product would provide to the contest operator.

H-1: Panopto Recording

A recording of my presentation of this capstone can be found here:

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=fafdfc9a-87c5-4a2f-abe5-aebf00fb4dca

https://drive.google.com/file/d/1KpU15NSBgPr_TeSPebIqlRv7hRxPRw2f/view?usp=sharing

Appendices

I-1: Evidence of Project Completion

Attached to this report for my capstone submission are the following items:

- The python script which handles all scraping, standardization, evaluation, and reporting operations of the contest data (DGB2021entries.txt)
- A spreadsheet containing all data regarding the time taken for manual and programmatic evaluation of the contest entries (Contest Time Data.xlsx)

- The spreadsheet deliverable provided to McIndoe containing all standardized and evaluated contest data, cosmetically modified to enable easier navigation (contest_entries_scored.xslx)
- The two slide deck deliverables provided to McIndoe, containing various descriptive statistics regarding the contest entries and outcomes. These slide decks, generated in Jupyter Notebook, open in a web browser and are navigated using arrow keys. (contest_summary1.slides.html & contest_summary2.slides.html)
- Copies of the McIndoe articles discussed in this report, due to their publication being behind a subscription-based paywall:
    - 202021 Contest 03: Conclusion & results of the 2020-21 contest
    - 202122 Contest 01: Announcement & entries for the 2021-22 contest
    - 202122 Contest 02: Updates on contest, using analysis from this project
    - 202122 Contest 03: Updates on contest, using analysis from this project

I-2: Web Sources for Code Solutions

The development of my code for this project used the following sources for code, or approaches which could be applied to my own code:

- Timing of the python script using the time module: https://datatofish.com/measure-time-to-run-python-script/
- Standard deviation of time elapsed: https://www.mrexcel.com/board/threads/standard-deviation-of-time-values.930794/

- Use of variables inside of markdown: https://discuss.dizzycoding.com/print-variable-in-jupyter-notebook-markdown-cell-python/
- P-value from Z-score, one tailed test: https://www.socscistatistics.com/pvalues/normaldistribution.aspx

Sources

McIndoe, S. (2021a, August 3). *Down Goes Brown prediction contest results: Did anyone have a perfect entry? Would that be enough to win?* The Athletic. https://theathletic.com/2747031/2021/08/03/down-goes-brown-prediction-contest-results-did-anyone-have-a-perfect-entry-would-that-be-enough-to-win/ (Attached as 202021 Contest 03 due to paywall)

McIndoe, S. (2021b, October 10). *Down Goes Brown: Predict the NHL season with the return of the contest that's so easy it's almost impossible*. The Athletic. https://theathletic.com/2869497/2021/10/10/down-goes-brown-predict-the-nhl-season-with-the-return-of-the-contest-thats-so-easy-its-almost-impossible/ (Attached as 202122 Contest 01 due to paywall)

McIndoe, S. (2021c, December 13). *The Canucks turn it around, don't trust the Rangers just yet, a contest update, and a Tortorella rant*. The Athletic. https://theathletic.com/3011764/2021/12/13/the-canucks-turn-it-around-dont-trust-the-rangers-just-yet-a-contest-update-and-a-tortorella-rant/ (Attached as 202122 Contest 02 due to paywall)

McIndoe, S. (2022, March 4). *Down Goes Brown: Checking in on that super-easy NHL prediction contest, in which many of you are already out*. The Athletic.

https://theathletic.com/3157320/2022/03/04/down-goes-brown-checking-in-on-that-super-easy-nhl-prediction-contest-in-which-many-of-you-are-already-out/

(Attached as 202122 Contest 03 due to paywall)

McIndoe, S. [@DownGoesBrown]. (2022a, May 9). *Down Goes Brown on Twitter* [Tweet]. Twitter. https://twitter.com/DownGoesBrown/status/1523674768439922688

McIndoe, S. [@DownGoesBrown]. (2022b, May 9). *Down Goes Brown on Twitter* [Tweet]. Twitter. https://twitter.com/DownGoesBrown/status/1523812031748612098

Wulf, B. (2021, September 7). *Think you know the NFL? Enter The Athletic's 2021 season prediction contest*. The Athletic. https://theathletic.com/2808061/2021/09/07/think-you-know-the-nfl-enter-the-athletics-2021-season-prediction-contest/