University of Auckland

Faculty of Science

Improving Analytics in Golf

Master of Data Science Dissertation

Author:
Jaskirat Atwal
736729947

Supervisors:
Dr. Beatrix Jones
Dr. Oliver Stevenson

Golf is a growing sport with many new and existing players interested in improving their play each year with data-driven analytics driving players to new heights. This project improves on the analytics and insight available to golfers through three major areas of work: 1) computing novel performance summaries, 2) grouping and recommending of players based on play-style, and 3) adjusting for course difficulty when comparing player performance.

Department of Statistics

November, 2021

Acknowledgements

I would like to thank my supervisors, Ollie Stevenson and Beatrix Jones for their wisdom and guidance throughout the course of the project. Thanks also to the team at Luma Analytics for providing me with a welcoming environment during my time on this project.

Contents

List of Figures

List of Tables

1 Introduction

The modern version of golf dates back to 15th century Scotland [1] and in 2020 had a player-ship of 37 million in America alone [2]. While traditionally associated with an older demographic, young adults (18-34 years old) are one of the sports largest customer age segments. The sport continues to grow steadily—according to the National Golf Foundation, 17 million Americans who didn't play golf in 2020 are "very interested" in playing on a golf course [2]. After a long hiatus, golf has now returned to the Olympics, being played by over 40 countries at the 2016 Rio and 2020 Tokyo Olympic games [3, 4]. As of 2019, the USA was home to 43% of the worlds golf courses, followed by Japan and Canada with 8% and 7% respectively [5].

Aside from its many golf courses and large player-base, the USA is home to some of the most important and well-known tournaments in golf. With global recognition and winners receiving many millions of dollars, the PGA TOUR is likely the most important tournament in golf. It is the goal of many golfers to be invited to play on the PGA TOUR, and for young players obtaining a golf scholarship to an American college is a highly promising start to a career in golf.

Like many sports, the game of golf has been subject to a data-driven revolution in recent years. Detailed analytics and metrics that were once only accessible by professionals are now becoming available for amateurs to hone their game. Platforms now exist where any golfer can input their shot data to receive information on areas of opportunity and targeted training. Given the parity of data collection between professional and amateur players, golfers of all abilities are able to benefit from shared analytics. With one such online sports analytics platform as a client, the broad problem statement of this project was to: "*Use new and existing data to improve analytics and insight on the platform and beyond by providing novel visualisations, recommendations, and benchmarks*".

This was broken down into three major areas of analysis:

- *PCA*: improving the depth and visualisation of player metrics and summaries by incorporating a large number of variables with Principal Component Analysis.
- *Clustering*: adjusting for player ability before grouping golfers by play style, and recommending professional players to amateurs with similar play styles.
- *Course Difficulty*: adjusting for a course's difficulty for unbiased comparisons and to develop a new US college golfer benchmark for players to compare their performance against.

This report introduces relevant background concepts, before separating into three chapters for each of the major areas of analysis. These chapters follow a structure of data and pre-processing, methodology, results, and discussion. This project was completed over one semester at the University of Auckland, in collaboration with Luma Analytics and as part of the Master of Data Science programme. All analysis was performed in `R` on standard laptop hardware.

2 Background

This chapter introduces relevant concepts and terminology in golf before covering the technical background of this report and briefly commenting on related work.

2.1 Background on Golf

Golf is a sport in which players compete at hitting a ball into a series of holes, forming a course. Each hole begins with a teeing ground, ends with a putting green, and contains various terrain such as long grass and sand traps, as well as hazards such as water and rocks. The layout and arrangement of each hole on a course are unique but has a set *par* number of strokes of which a skilled golfer would need to complete the hole. The typical game of golf consists of 18 holes, and the winner is determined by whom completes the course in as few strokes as possible. This results in the counter-intuitive notion of a low score denoting a positive performance.

Golf involves various types of strokes, and a golfer carries clubs for each circumstance. Typically the first stroke is intended for hitting the ball a large distance and is taken with a long-shafted and large-headed *driver* club. Once the ball is on the putting green, shorter and lighter clubs are used for hitting the ball the remaining short distance. Typically, a golfer's game is broken into the four aspects: Driving, Long game, Short game, and Putting. Long-game shots are those taken from a distance over 100 yards (including the initial shot), and Short game includes any shot under 100 yards (including putts).

The winner of a round of golf is determined by the lowest total number of strokes (or equivalently the difference between the final number of strokes and the par for the course, *score-to-par*). However, less clear are the factors that contributed to the victory. For example, there was significant debate around Tiger Woods during his prime; whether his low scores were due to "*superior putting, wedge play around the greens, driving, or some other factor or combination of factors*" [6]. Furthermore, scoring at a certain level relative to par scores can have a wildly different meaning depending on the difficulties of the courses being played.

The *Strokes Gained* (SG) metric [7] has become widely adopted as a meaningful and interpretable way of assessing a golfer's performance. Strokes gained works by estimating a function for the number of strokes a PGA TOUR golfer would take to complete a hole given the distance of the ball from the hole and the condition of the current terrain (fairway, rough, green, sand, or recovery). The difference between the value of this function before and after the golfer takes the shot thereby quantifies how good the shot was relative to other PGA TOUR golfers. For example, from a distance of 16 feet on the putting green, a PGA TOUR golfer will sink the ball in one putt 20% of the time and in two putts 80% of the time [6]. The expected number of strokes needed to complete the hole from 16 feet is therefore 1.8. In practice, a player making this putt who sinks the ball in one putt has gained 0.8 strokes, and a player who sinks it in two strokes has lost 0.2 strokes.

By calculating the strokes gained metric over each of the categories of stroke (Driving, Long game, Short game, Putting), a golfer's performance in each of these categories as well as on a per-shot basis can be directly compared. To return to the earlier Tiger Woods example, Broadie [6] showed that his scoring advantage between 2003-2010 was 3.20 strokes per round better than average. Of this 3.20, Tiger's Long game shots accounted for 2.08 strokes. While he still excelled at putting, most of his score advantage came from shots beyond 100 yards from the hole.

Strokes gained is also meaningful on a per-shot basis. Consider two golfers A and B on a par-3 course:

- *Golfer A* hits a phenomenal drive leaving the ball within a few yards of the hole, and putts it in for a total of two strokes. The strokes gained for the drive will be large and positive, while the strokes gained for the putt will be close to zero as it was an easy shot to make.

- *Golfer B* hits an awful drive but manages to then sink the ball with a lucky shot for a total of two strokes. The strokes gained for the drive will be negative, while the strokes gained for the second shot will be large and positive.

The golfers have both scored one under par (known in golf as a birdie), but the strokes gained metric provides information on *where* the gains were made.

The strokes gained metric attempts to address a long-standing problem in golf: because every course is different, it is difficult to compare a player's performance between different courses. Popular courses usually come with a *course rating* which is an estimate of the average number of strokes a "scratch" golfer (one that shoots at or better than par) would require to complete the course [8]. This is distinct from the par score for the course–for example, a par-72 course with a course rating of 74 would be considered difficult, while a course rating of 68 would indicate the course to be easy. In cases where a strokes gained metric is not available, the course rating can help when comparing scores-to-par.

2.2 Technical Background

This section discusses the concepts and background of the technical aspects of this report.

Principal Component Analysis

Especially at the elite level, golf is rich in data. Datasets such as the PGA TOUR have hundreds of variables available, making *Principal Component Analysis* (PCA) a good starting point. PCA is often useful in exploratory data analysis or predictive modelling but is worth consideration whenever one has a large number of variables and wishes to understand the relationships between them.

PCA is the process of computing a series of vector "components", where each component: 1) minimises the squared distance from the data points to the line and 2) is orthogonal to every previous component. By construction, the components represent independent dimensions in the data that retain as much information (variance) as possible. Highly correlated variables tend to collapse or "load" onto the same principal component, with unrelated variables loading onto different components. The result is that an analyst can choose to consider only the first few components, and in doing so, can achieve a significant dimensionality reduction. That is, in a dataset with many correlated variables, PCA allows one to represent as much of the original variance in the data as possible by using a reduced number of "summarising" variables. In most cases, using only the first one or two PCs (referred to as PC1 and PC2 respectively) is sufficient to explain a large amount of variation while remaining within the realm of what is possible for humans to visualise and conceive.

Consider an observation vector x, where each element x_i represents the measured value of the ith variable. Having computed a PCA and stored the loading coefficients c_{ij} for the jth principal component, the output or "score" PC_j of the jth principal component upon applying the PCA to x is computed as:

$$\mathrm{PC}_j(x) = \sum_i x_i c_{ij}$$

In this way, PCA acts as both a summary and a data compression tool. A dataset of N observations requires storing $N \times M$ elements where $i = 1, ..., M$. In contrast, by storing a $P \times M$ matrix of loading coefficients where $j = 1, ..., P$ and $P << M$, the data can instead be summarised with just an additional $N \times P$ matrix. Storing the results of the PCA means that the same method of summarisation and compression can be used on new data directly.

In practice, PCA involves computing the eigenvectors of the data's covariance matrix, which can be done efficiently using singular value decomposition. It is often beneficial to normalise the input variables, for example, to have a mean of 0 and a standard deviation of 1. This maps every variable onto a similar range and allows linear combination without biasing towards numerically larger variables. For example, without normalisation, a variable like driving distance in yards could bias the results when other variables exist on different scales such as percentages or proportions.

Linear Regression

Linear regression is an approach for linearly modelling the relationship between a scalar response variable and one or many explanatory variables. For a dataset with $i = 1, ..., n$ observations of a response variable y and $j = 1, ..., p$ explanatory variables x, the formula for a linear regression is:

$$y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \epsilon_i$$

Where ϵ_i is random noise. Removing ϵ_i gives an equation for $\bar{y}$, the fitted value of y given the explanatory variables. Accurate modelling of y depends on several assumptions such as a linear relationship between y and x and constant variance. However, even without rigorous satisfaction of these assumptions, some basic inferences can still be made.

Linear models can be fitted through various means, most commonly using least squares or maximum likelihood estimation. Once a model has been fitted (and if the purpose of the model is prediction), the most pertinent question is often "how well does the model fit?". The coefficient of determination, or R^2, is the proportion of variation in the response variable explained by the explanatory variables. With $\bar{y}$ as the mean of y and the fitted values from the model $\hat{y}$, the R^2 is defined from the following:

$$SS_{\text{resid}} = \sum_i (y_i - \hat{y}_i)^2$$
$$SS_{\text{total}} = \sum_i (y_i - \bar{y}_i)^2$$
$$R^2 = 1 - \frac{SS_{\text{resid}}}{SS_{\text{total}}}$$

With $0 \leq R^2 \leq 1$, a large R^2 means that relative to the total variation in the data SS_{total}, the variation of the fitted values about the observed values of y is small and the model fits well.

K-Means Clustering

K-Means clustering [9] is a classical and widely-used method for partitioning n observations into K clusters. This is done in such a way that the within-cluster variation is minimised, based on a user-specified similarity metric (such as Euclidean distance). The naïve algorithm is simple: start by randomly assigning each observation to a cluster and compute the cluster centroids as the mean of every observation within that cluster. Then: iteratively compute the distance between each observation and each cluster, assign observations to the closest cluster, and recompute the cluster centroids until convergence. Given the cluster centroids, unseen data can be classified into a cluster by choosing the nearest centroid. If the number of clusters $K = 1$, this is equivalent to the also popular and similarly named k-Nearest-Neighbour algorithm [10].

When determining the distance from one point to the next, different metrics are possible that result in different clustering arrangements. Consider two points a and b in an n-dimensional space. Some common distance metrics are:

$$\text{Taxicab Distance } (L_1\text{-norm}) = \sum_i |a_i - b_i|$$
$$\text{Euclidean Distance } (L_2\text{-norm}) = \sqrt{\sum_i (a_i - b_i)^2}$$
$$L_\infty\text{-norm} = \max_i |a_i - b_i|$$

The Euclidean distance is a popular choice as it progressively penalises more distant values. However, the best choice of metric can depend on the specific application.

Because of its simplicity and efficacy, K-Means is popular for exploratory data analysis and gaining an intuition of the data structure. Other uses include image compression or as part of a pipeline for "pre-classifying" observations before more complex methods exploit the differences in characteristics between the clusters.

There are times when "K" is known in advance. However, often this is not the case, and the analyst may need to inspect the results of applying different Ks to determine an appropriate value. *Silhouettes* [11] are a formal mathematical measure of how good a clustering is and can be used to determine a good value for K. For a distance metric $d(i, j)$ measuring the distance between two points i and j, the silhouette width $s(i)$ of point i in cluster C_i is computed from:

- cohesion(i): the mean distance (using from distance metric) from i to every other point in C_i. Formally:

$$\text{cohesion}(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

  The cohesion can be interpreted as how well the point i belongs to its own cluster.

- separation(i): the minimum of the mean distances from i to every point in every other cluster $C_k \neq C_i$. The C_k that minimises is termed the *neighbouring cluster* and is the next best fitting cluster for the point i. Formally:

$$\text{separation}(i) = \min_{k \neq i} \frac{1}{C_k} \sum_{j \in C_k} d(i, j)$$

  The separation can be interpreted as how well the point i belongs to the neighbouring cluster.

The silhouette width $s(i)$ is now defined as:

$$s(i) = \begin{cases} 1 - \text{cohesion(i)/separation(i)}, & \text{if cohesion(i)} < \text{separation(i)} \\ 0, & \text{if cohesion(i)} = \text{separation(i)} \\ \text{separation(i)/cohesion(i)} - 1, & \text{if cohesion(i)} > \text{separation(i)} \end{cases}$$

The silhouette width then lies between $-1 \leq s(i) \leq 1$, with values close to 1 for a point i indicating high cohesion and good fit to the current cluster C_i, and values close to -1 indicating point i would be better suited to the neighbouring cluster. A value near 0 means it it is near the boundary between the two choices of cluster.

The mean of $s(i)$ over all points in a cluster measures how tightly grouped the cluster is. Thus the mean $s(i)$ over the entire dataset measures how appropriately the data have been clustered and can be used to compare the clustering for different values of K. A poor clustering configuration with too few or too many clusters can be identified by many low or negative value points.

Web Scraping

The web is home to a wealth of information unprecedented in all of human existence. *Web scraping* refers to the automated extraction of information from web pages. This can include text, images, and metadata about what is displayed on page.

Visiting a page on the web involves the page's HTML source code being rendered by a web browser (e.g. Chrome, Firefox, Safari). HTML code contains all the information needed to display the page, including text and hyperlinks to images and videos. Elements in an HTML document are nested in a tree structure and have broad classes such as heading, table, link, image. This means that tools such as *xpath* and *xquery* can exploit and query the structure of the HTML document to extract desired information.

A *static* web page is one where all content on the page is loaded together, and no changes are made to the HTML source code. The general workflow for scraping a static web page is to:

1. Define the target URL of a web page.
2. Retrieve the page's source code (this can be done programmatically or within a web browser).
3. Inspect the source code to gain an intuition around how the information of interest is structured.
4. Based on the document structure, develop a query to extract the information of interest from the source code.

However, this procedure may fail for *dynamic* web pages: e.g. a page that dynamically retrieves and displays search results without overtly exposing any URL endpoint for accessing the search results directly. This type of web page is designed to be interacted with by a human user rather than a machine. For these cases, tools have been developed to simulate human interaction. The RSelenium [12] package offers programmatic control over a web browser–allowing things to be clicked and keystrokes to be registered in the same way as if it were a human user interacting with the browser. This is frequently paired with a headless version of a chosen web browser, as rendering the graphical display is unnecessary for the machine's interaction. Docker [13] containers are a popular method of running such programs in an isolated environment.

The HTML source can still be retrieved at any time to guide the program in its interaction or to extract the desired information directly. Some websites, however, require the user to complete a captcha upon submitting multiple requests from the same IP address within a short time-frame. This is explicitly designed to stop machines overloading the server, and scraping information from pages with such security measures is outside this project's scope.

2.3 Related Work

Academic literature on golf comes predominantly from areas such as sports psychology ([14], [15]), physiology ([16], [17]), sociology ([18], [19]), or economics and the environment ([20], [21], [22]). Journals such as the Journal of Sports Analytics [23] and Journal of Quantitative Analysis in Sports [24] are examples of relevant academic publications to this work. However, overall, the kind of data and analysis conducted in this work is far more prevalent in an industry than an academic context and this work remains highly novel.

A literature survey returns some tangentially related analyses in the field of golf analytics. For example, Yousefi and Swartz [25] develop a new metric to assess putting performance. Chan et al. [26] investigate how to better allocate player handicap scores. Drappi and Co Ting Keh [27] predict golf scores at the per-shot level. Regardless, Broadie 2012's [6] strokes gained concept was the main direct influence on this work.

3 Principal Component Analysis

The strokes gained (SG) metric offers a powerful scalar summary of a player's performance in each golfing aspect, but unfortunately, data is not always available or is difficult to calculate. Principal Component Analysis (PCA) offers the ability to condense multiple variables into a single metric that potentially contains the same or more explanatory power than the SG metric. This chapter covers the technical details and methodology of the Principal Component Analysis, from data acquisition to analysis to results.

3.1 Dataset and Pre-processing

One of the main sources of data for this project came from the PGA TOUR. A small amount of data on amateur players was also available with many of the same variables.

PGA TOUR Dataset

The PGA TOUR is the main professional golf tour played by men in the United States of America. A similar tour exists for women (LPGA) but unfortunately has less data and variables available, thus this project was limited to men's PGA TOUR data. Prior to the beginning of this project, a dataset of player statistics had been scraped directly from the PGA TOUR website over the 2019 season (October 2018 – August 2019). This was available as a ∼900MB CSV file.

	Raw Data	Processed Data
% Missing	84	0
# of Variables	1497	56
# of Golfers	1451	218

Table 3.1: PGA Tour dataset statistics.

A wide range of metrics was available for each player, from standard ones like strokes gained and scores-to-par to less useful ones like tournament winnings. A full list of variables with explanations is available in Appendix 8.1. **Table 3.1** summarises the number of players, variables (excluding the player name), and missing values before and after pre-processing. The large data file size was due to the scraper having been run periodically and continuously appending player statistics to the file. Pre-processing was necessary to extract the most recently updated value for each player and each of that player's available metrics. The dataset was in "long" format: each row had one column for the metric and one for the value. This needed conversion into "wide" format where each row represented a singular observation (in this case, player) with each collected metric having its own column.

There were also a significant amount of missing values and the phenomenon of players who were only occasionally invited to play on the PGA TOUR for whom data had not been so thoroughly collected. This was a problem as the PCA was unable to handle missing values. A complete case analysis was taken, ensuring a high-quality dataset of professional players.

While the PGA TOUR data offered metrics tracking the average performance of players, the outcome of a single round of golf can still be highly variable. The Official World Golf Rankings (OWGR) [28] were used to augment the dataset with the real-world performance of the player at the end of 2020. Rankings at the end of 2019 would have likely been more consistent with the PGA TOUR dataset but, unfortunately, were not easily available. **Figure 3.1** displays the distribution of the OWGR rank of players that were dropped from the dataset due to missing data. The assumption was made in taking the complete-case analysis that no important players were dropped, but this shows that most dropped players were at a lower rank, and many were not on the OWGR rankings at all (coded as rank 2000).

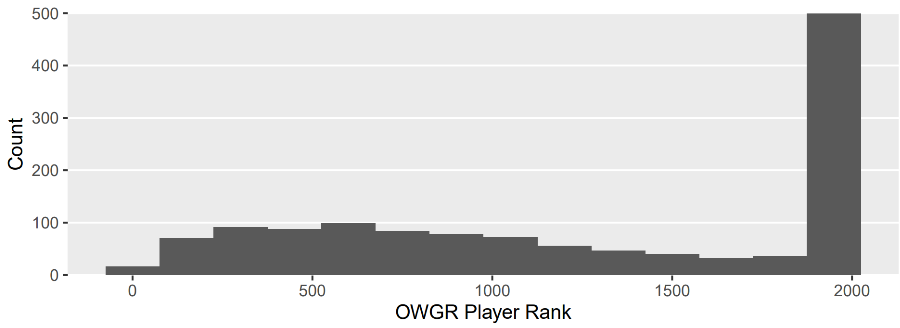


Figure 3.1: Distribution of rank of players dropped from analysis.
Note: Players with no OWGR ranking coded as 2000.

The dataset was further augmented with player headshots scraped from the PGA TOUR website [29] to be displayed on the player visualisations. The website provided a single static page containing a link to the personal profile of every player on the tour. Each player's ID was determined by retrieving the page's HTML source and extracting every link element with the class of "player-link". When generating a visualisation for a certain player of interest, their names could be matched against the links dataset to retrieve their player ID. From there, a standard URL was formatted with the player's ID and the resulting image file downloaded and rendered directly on the plot.

Amateur Dataset

A golf instruction company in New Zealand maintains an online platform, which allows amateur and professional golfers alike to record their stats and receive tailored feedback and training plans to help improve their play. Unfortunately, due to Covid-19 [30], amateur data was not able to be collected from this platform at a large scale. Data for a handful of amateurs were manually collected from the website and used to offer a proof of concept of what might be possible with a larger cohort of amateur data.

While this dataset resembled the PGA TOUR data in that many of the same variables were present in both, some manual work was still involved in connecting the two. A JSON file was created to store a mapping of column names between the two datasets and the corresponding golfing aspect of each column. The columns needed only to be renamed and assigned a category for the amateur data to be integrated into the existing pipelines built for the PGA TOUR data. The columns all measured the same qualities as in the PGA TOUR data, except for the strokes gained (SG) variables; where the PGA TOUR SG variables were relative to other PGA TOUR players, the SG variables in the amateur dataset were relative to "scratch" players.

3.2 Methodology

Traditionally the game of golf is broken down into four key aspects: *Driving*, *Long-game*, *Short-game*, and *Putting*. With a large number of variables in the PGA TOUR dataset, it was desirable to condense these into summaries of a golfer's overall performance in each of these aspects. Principle Component Analysis (PCA) using `R`'s inbuilt prcomp function provided a powerful means of reducing the dimensionality of the dataset and obtaining said summaries. This function took a dataset as input and returned a sequence of principal components (PCs) as output, decreasing in order of explanatory power. The first principal component (PC1) could then be extracted and used as a scalar summary, or the first two taken to generate a plot.

Variable Selection and Categorisation

Variables first had to be assigned to one of the categories of golf. This was done based on a variable list from a previous project on developing benchmark statistics. Variable names were matched against the ones in the PGA TOUR dataset and associated with one of the different aspects of golf. A similar process was undertaken with the amateur dataset using the variables in the processed PGA TOUR dataset. Some variables were useful for describing a golfer's overall game but did not apply to a particular aspect of golf and were saved under a "General" category. The final set of variables is described in **Table 3.2** with examples of variables in each category. A comprehensive list is available in Appendix 8.1 with descriptions and explanations of the variables.

	# of Variables	Examples
General	10	Avg. Stroke Differential, Avg. Score on Par 3/4/5
Driving	4	Avg. Driving Dist., Driving Accuracy (%)
Long-game	12	Proximity to Hole after Stroke from 125-150 yards
Short-game	12	Sand Save (%), Proximity to Hole from 50-75 yards
Putting	18	One-putt (%), Avg. Dist. of Putts
Total	56	

Table 3.2: Variable categories with examples.

Due to the small number of variables in Driving and their semantic similarity, combining the Driving and Long-game aspects was considered. However, the resulting PCA had differences in variable loading that were sufficient to warrant the separation of the two aspects. Golf is typically separated into the four distinct aspects and for the sake of consistency and familiarity, they were left separate.

Importance of Strokes Gained

Because the SG variables will not always be available and is often difficult to calculate for amateur players, an investigation was performed into what can be done even in the absence of SG data. This consisted of:

- Fitting a linear regression to the non-SG variables to predict the SG variable and measuring the R^2 value. The R^2 measures the proportion of variation in the SG variable explained by the rest of the variables. If the R^2 is high, the non-SG variables can accurately predict the SG variable. A convenient side-effect is that the coefficients of the linear model can also be used in place of the loading coefficients of the PCA to generate an alternate summary.

- Performing the PCA with and without the SG variables included and comparing the percentage of variation explained by the first principal component.

- Measuring the Pearson correlation of the first two principal components of a PCA with their respective SG variable. This measured the extent to which PC1 or PC2 was measuring the same thing as the SG variable. For example, a large correlation would imply that a good golfer would be deemed good by both the SG and the PC variable.

Output Summaries

After executing the PCA to summarise each golfing aspect, the summaries were normalised to each have mean 0 and standard deviation 1. This allowed for direct comparison (in terms of a player's deviation from the mean) between the summaries. PCA was also applied to all variables simultaneously (including ones in the "General" category) for an overall summary of a player's performance. As each aspect had a different number of variables assigned, there was a risk of bias towards players who were stronger in the aspects with more variables. By overriding the default column-by-column scaling behaviour of `prcomp`, a custom aspect-by-aspect scaling was implemented. The variables were normalised as usual but then multiplied by the reciprocal of the number of variables in that aspect. Aspects with fewer variables were therefore allowed to vary more, resulting in variables that were more valuable in the overall PCA.

Visualisation

The results of the PCA were visualised with interactive linked plots using the plotly [31] and ggplot [32] libraries. For each of the golfing aspects and the overall summary with every variable included, the first principal component was plotted against the second. With too many points to label individually, hovering over a point in the plot displayed an annotation with the player's name instead. The plots were linked together such that when the user clicked on the player in one of the plots, that player was automatically highlighted in every other plot. Alternatively, the user could query the player's name in the search box to the same effect. This allowed the user to very quickly see where that player sat amongst the cohort of PGA golfers, both overall and broken down by golfing aspect.

The sign of a PC can be arbitrary, and a rigorous interpretation requires understanding how the underlying variables have contributed to the output of the PCA. To aid in interpreting the summary plots, supplementary plots were generated that visualised the loading of each variable onto PC1 and PC2 of each aspect. By hovering over each bar, the loading coefficient and name of the variable was displayed, allowing the user to interactively explore how each variable was associated with PC1 and PC2. These plots also aided in detecting if a sign change was necessary; PC1 of the Driving aspect had to be negated in the summary plots to give its axis the direction of conventional interpretation.

For visualising the overall skill profile of a player, *radar charts* from the fmsb package were used. These produced plots where the multiple aspects of a player's game were visualised on the same plot along with their headshot from the PGA website. The axis limits were the entire range of values for the cohort of PGA TOUR players to visualise how the player is performing relative to their cohort.

Amateur Data

Once amateur data was available, the PCAs trained on the PGA TOUR dataset were used to generate predictions for the amateur players. The amateur players were then included on the same plots as the professional players for comparison. Note that five columns were missing from the amateur data and these were filled with zeros. Zero was used as a replacement value to minimise the bias of the missingness on the PCA.

3.3 Results

The results of the PCA analysis examine the viability of the PC summaries as an alternative to the SG variables. An interactive HTML document containing the full range of plots in this section is hosted on GitHub[1].

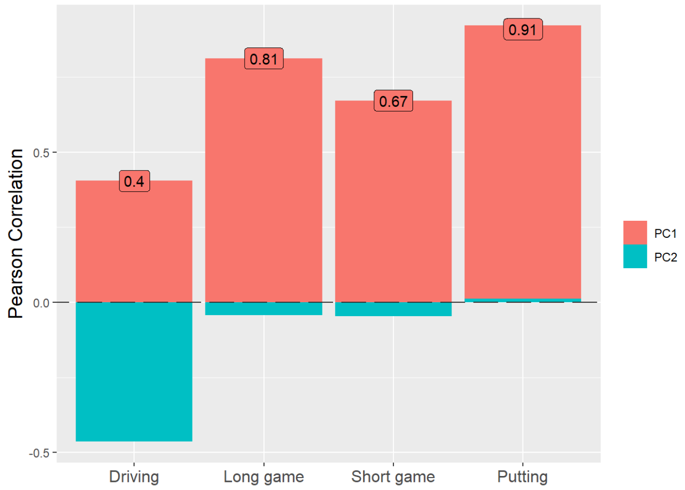
PCA as a replacement for SG

Computing Strokes Gained (SG) data requires the player to know the distance from the hole and condition of the course at each stroke. This data is not always available and can be difficult to calculate. PCA summaries computed on the non-SG variables were used to try to match the explanatory power of the SG variables and obtain a similarly powerful summary metric of performance.

Figure 3.2a shows the Pearson correlation of PC1 and PC2 with each of the SG variables. In general, each PC1 is reasonably correlated with its respective SG variable, confirming that the PC1s can be interpreted similarly to how a SG variable would be interpreted with respect to the player's ability. For example, a high correlation in this case means that a good putter with a large and positive strokes gained in putting is very likely to have a large and positive value in PC1 for Putting. However, the one variable that does not follow this trend is Driving. The correlation of PC1 with the Driving strokes gained is lower than the rest of the variables, and PC2 is also negatively correlated with the strokes gained for Driving. Ideally, and what is seen with the other aspects, is that PC1 is able to combine all the useful variables to create a summary that is correlated with the SG variable. With Driving, the variables contained distinct information—thus no meaningful combination was possible.

Table 3.2b displays the R^2 values obtained when fitting a linear regression model with the SG variable as the response and the non-SG variables as the p explanatory variables for each golfing aspect. The R^2 values measure how much of the variation in the SG variable can be explained by the non-SG variables, and shows that Putting can be predicted with very high accuracy, Driving and Long game with reasonable accuracy, and Short game with poor accuracy. Despite only having three explanatory variables, the SG for Driving can be predicted surprisingly well ($R^2 = 0.72$). When this is interpreted together with **Figure 3.2a**, the conclusion is that despite being capable of reasonably reconstructing the SG variable, the PCA has found that more variance was able to explained through the construction of something different.

[1] `https://github.com/OptimusPrinceps/Golf-Masters/blob/main/output/PCA.html`



(a) Pearson Correlation of PC1 and PC2 with the corresponding SG variable.

	p	R^2
Driving	4	0.72
Long game	12	0.70
Short game	12	0.53
Putting	18	0.95

(b) Number of explanatory variables p and R^2 values for linear models fitted to each SG variable

Figure 3.2: Investigation of PC variables as alternatives to SG.

Figure 8.1 in Appendix 8.2 shows the amount of variation explained in the data by each PC1 with and without the SG variable included. The results are consistent with **Figure 3.2** in that PC1 in each aspect performs as an adequate summary of the data. The PC1 for Driving in particular explains a large amount of the variance in the data. The following section further examines the Driving PCA through an example.

Case Study: Rory McIlroy and Cameron Champ

Figure 3.3 visually compares the PC against the SG variables for two PGA TOUR players Rory McIlroy and Cameron Champ. For each golfing aspect, the PCA was run on all available non-SG variables, and PC1 plotted as the summarising variable. For each summarising variable type (SG or PC), the axes are scaled based on the range of values in the data. For example, Cameron Champ's Driving statistic was the highest in the dataset when using the PC variables as a summary. Using the SG variables as a summary, his Driving statistic was still relatively good, but not the best. Rory was the opposite: he had the best Driving statistic in SG terms and was close to the best in PC terms. Only two player's plots are given here. However, many more were inspected, and all were largely similar between the SG and PC variables, with the Driving variable being the primary source of discrepancy.

Inspecting **Table 3.3** gives an even closer look into the discrepancy between Cameron and Rory's Driving statistics. Note that the values for the variables given (e.g. an average Driving Distance for Cameron Champ of 317.9 Yards) were the raw values from the dataset prior to centering and scaling to remove the effect of having different units and scales. After centering and scaling, the PC output column is the result of multiplying each variable by its loading coefficient and taking the sum. Driving distance was most strongly loaded against PC1, followed by the accuracy of placing the ball on the putting green (GIR).

However, the variable for accuracy of hitting the ball onto the fairway was given a negative coefficient. This meant that while Rory was unequivocally a better player (both in terms of real-world performance and the SG measure), a player like Cameron that was able to hit the ball further and was more accurate with getting it on to the putting green—even despite being less accurate with getting the ball on the fairway—was given a larger PC output for Driving.

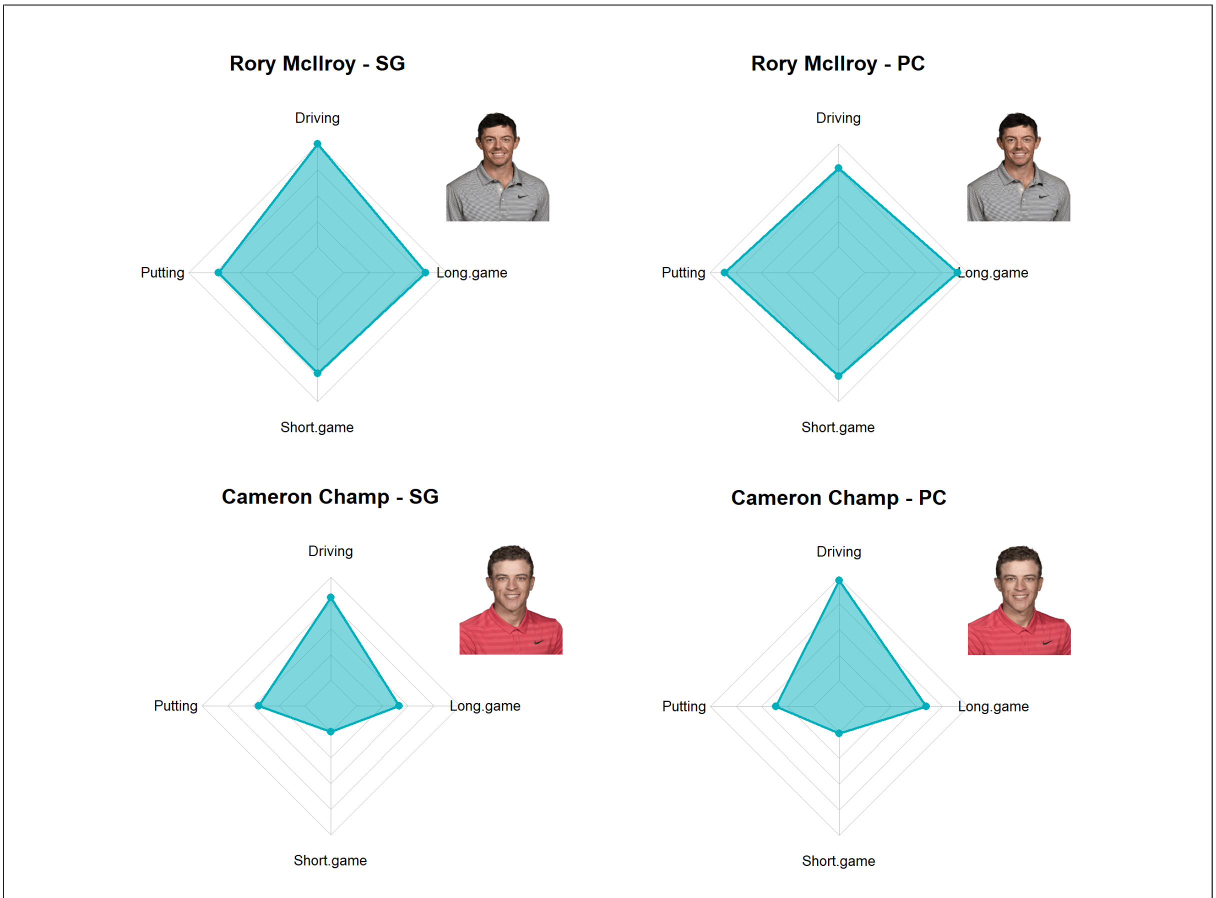


Figure 3.3: Strokes Gained vs Principal Component variables comparison.

	Driving Dist. (Yards)		Fairways Hit (%)		GIR (%)		PC1	SG
	Pre	Post	Pre	Post	Pre	Post		
Cameron Champ	317.9	2.74	55.3	-1.35	81.2	1.52	3.34	0.66
Rory McIlroy	313.5	2.24	61.8	-0.09	78.3	0.54	1.83	1.2
Loading Coef.	0.67		-0.53		0.52			

Table 3.3: The raw input Driving variables (prior to centering and scaling) to the PCA, and the raw output of the Driving PCA. The loading coefficient of each variable along PC1 and the SG in Driving is also given.

PC1 for the Driving aspect can therefore be interpreted as being high for players that "hit the ball large distances, especially onto the green, but not on to the fairway". In a practical sense this may mean a player can hit the ball a large distance and not worry so much about accuracy, as long as they can land it on the green in time to meet the par score. This corresponds with the current prevailing wisdom in golf that such a strategy is one of the best ways to achieve an overall lower score. While this summary may help to explain much of the variation in the dataset, **Figure 3.2a** still casts doubt on how effective this is as a performance metric. Additionally, Cameron, who exemplifies this strategy, clearly has a worse strokes gained than Rory. Given that the R^2 value for Driving was reasonable, a better use of the data could be to directly predict the Driving SG variable (through a linear model or otherwise) rather than using PCA to try and obtain a summary. This is left as future work.

PCA Summaries

Figure 3.4 displays the interactive linked plots that summarise each player's performance in each of the golfing aspects, as well as overall. Each subplot has PC1 and PC2 along the vertical and horizontal axes, respectively, and the values on the axes are interpretable as standard deviations from the mean. For example, Rory McIlroy's Short game was one standard deviation above the average player along PC1 and two standard deviations away from the average player along PC2. PC1's positive direction is easily interpreted as being a better player. However, PC2 is less black-and-white with its interpretation and requires an understanding of the variable loadings to interpret meaningfully.

The player's name is displayed when hovering over a point in the plot, and shown in the figure is the result of querying "Rory McIlroy" in the search bar above the plots (with additional players labelled in place of interactivity). At the time of writing, Rory was one of the best golfers around and was especially renowned for his driving ability. The summary plot and PC1 of the Driving subplot have correctly identified Rory as a significant outlier. Other than Rory, the upper hemisphere of the plot has been populated by golfers considered to be good (large positive value in PC1). In every PC1 vs PC2 plot, there is no apparent correlation between PC1 and PC2. PC2 is constructed to be independent of PC1 and serves more the purpose of visual separation than interpretation.

Figure 8.2 in Appendix 8.3 shows the result of running the existing PCA (trained on professional players) on an amateur player (pseudonymised as Amateur 1). Amateur 1 was reasonable at golf, on average scoring a few strokes above par. However, when compared to professional PGA TOUR players, Amateur 1 was a complete outlier. It is no surprise that the amateur has been placed at the very bottom of PC1 in every plot. A comparison is possible, but it is certain that it would be misguided due to the characteristics of the variables varying between the amateur and professional cohorts of players.

These PCA summary plots served as a user-friendly and intuitive means of comparing and visualising the performance of PGA TOUR players. While the individual PC plots provided a great deal of detail, the overall plot summarised things nicely.

Principal Component Loading

Figure 3.5 shows one of the visualisations produced for aiding in the interpretation of PC1 and PC2. Some of the information that can be gleaned from this plot:

- The variable most strongly associated with PC1 was the % on green (Green in regulation) variable for 150–175 yards.
- All the accuracy based variables (% on green variables) were loaded positively along PC1. All the distance-based variables (proximity to hole) were negatively loaded along PC1. This means that PC1 mostly measures a combination of the strokes

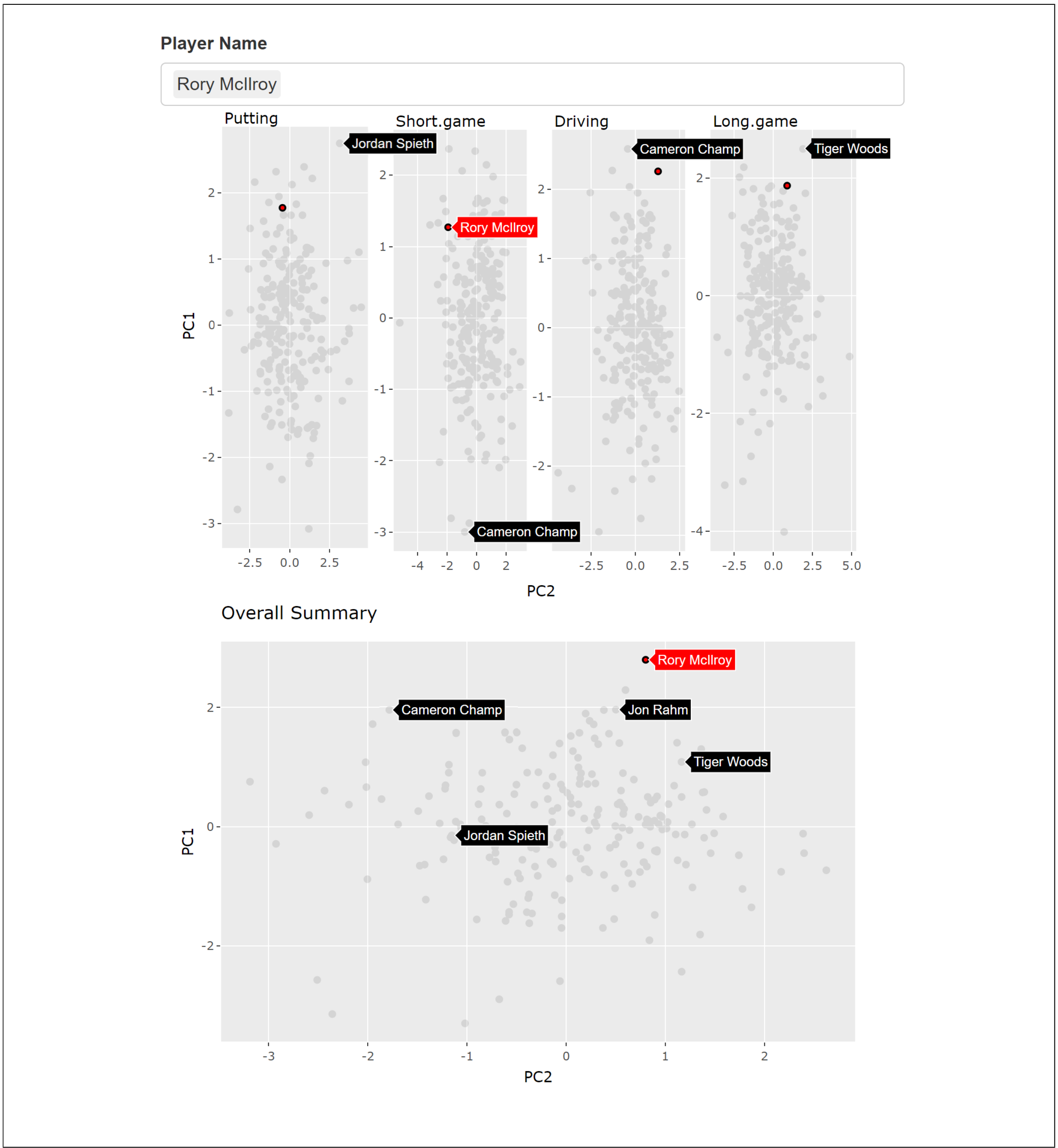


Figure 3.4: PCA summaries for each aspect, and the overall summary. The axes have been normalised and so are interpretable as standard deviations from the mean. Note: Additional players have been manually labelled in black in lieu of an interactivity.

gained and the accuracy of a player's Long game.

- By construction, PC2 has ended up associating itself with what is left unexplained by PC1; in this case, the distance-based variables.

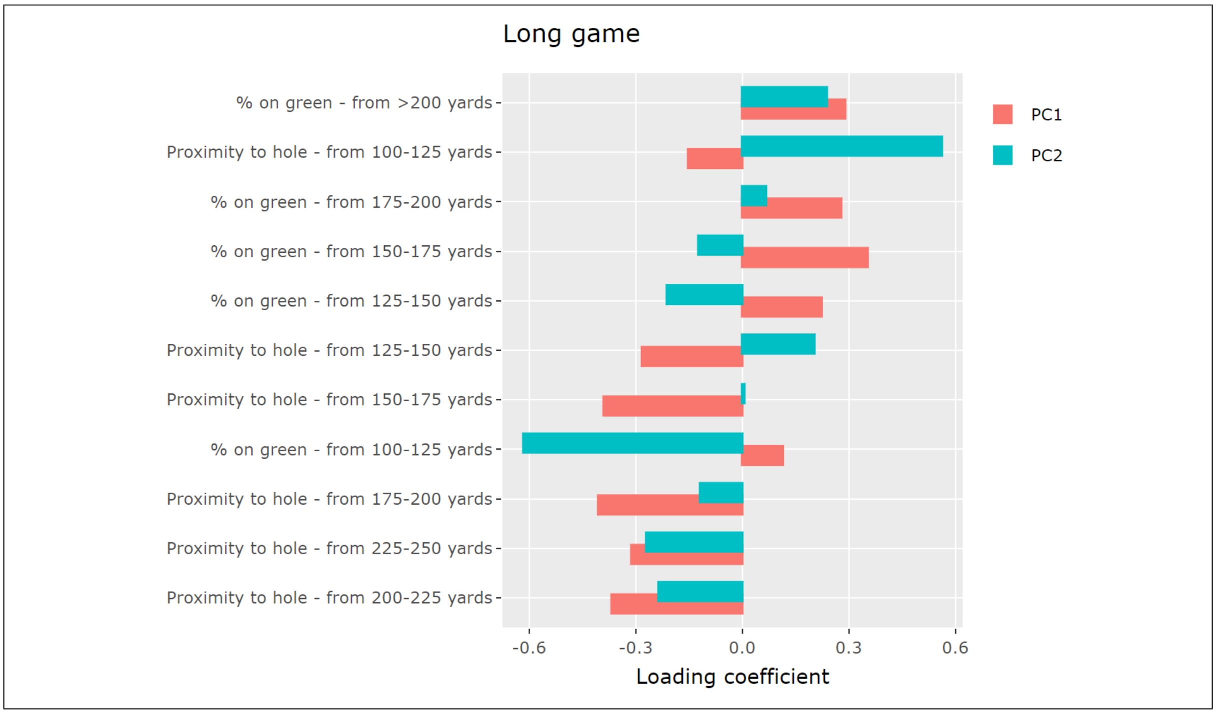


Figure 3.5: PC1 and PC2 loading visualisation for the Long game aspect.

These interactive and visual plots were far more effective for understanding the PCs than a table of text and numbers.

Key Findings

The key findings of the PCA are summarised as follows:

- Where available, the SG metric is the preferred method of summarising a player's performance. However, in cases where SG data is unavailable, PCA can still offer a compelling alternative summary.
- The PCA for the Driving aspect was flawed due to distinct information in the variables. Linear regression is one viable alternative for generating the coefficients to use in a summary.
- It is possible but fraught to apply the PCA generated on professional players to amateur players. Between the cohorts of players the variables are likely to have different characteristics.

3.4 Discussion

The main goal of the PCA was to compute a summary of a golfer's performance in a particular aspect of their game or overall. Because the strokes gained metric is already well understood, the PCA mainly has its use in the cases where SG data is not available. For these situations, the PC summaries serve as new metrics of their own. By training the

PCA on a cohort of players, the variable loadings could construct a summarising metric without needing strokes gained. This means that even without SG data, an amateur player or coach would still be able to construct a simple summary of their performance without needing to get lost in the details of each metric. It is still a question of how close these summaries would be to optimal in the sense of maximising real-world golfing performance, but a summary still has some inherent value.

The results of this work showed that, overall, the resulting PCA summaries would still be reasonable. PC1 of each aspect was reasonably correlated with its SG variable, and plotting the performance of each player in terms of SG and PC showed minor variation between the two sets of variables. However, the main limitation of this approach is its interpretability. Strokes gained is measured in strokes—a fundamental concept any golfer can understand. In contrast, even reporting the PCA scores as a ”standard deviation from the mean” has little intuitive interpretation by non-statisticians. Even worse, imagine trying to explain to a layman that their golfing ability as calculated by a PCA involves summing the result of multiplying variable X by seemingly arbitrary coefficient Y for every measured variable. Much of the “advanced analytics” in other sports are often better than traditional statistics for rating player ability, but their uptake by the spectatorship is limited by their interpretability. The success of the strokes gained metric is not solely due to its explanatory power, but largely also to its interpretability. In situations where interpretability is not an issue (for example, outside of outside of spectatorship and more towards elite-level coaching) it would be easy to include the SG variables within the PCA and construct a variable with even more explanatory power than the SG variables alone. This could then be given an approachable name like “Driving Index” and used for further advanced analytics.

Another limitation of the PCA is its specificity to the cohort of players it was calculated over. Computing a PCA over a cohort of professional players and using the same loading coefficients for amateur players assumes a linear relationship between ability and the value of the metric. The actual relationship between ability and metric is unlikely to be linear when extrapolating outside of the range of the data. For example, seeing negative values for metrics measured as a percentage does not make sense: there is some range of values that ”make sense” to be observed, but this information is not coded into the PCA. For example, the utility of going from driving the ball 300 to 305 yards is far greater than the 1.7% improvement that this appears to be on a linear scale. The range and variance characteristics of variables will also vary between cohorts: for example, the driving distance in the professional dataset varies between 270–320. Meanwhile, the driving distance of a cohort of amateurs is likely to have a much wider spread and be centered at a lower value. A major limitation of this project was that no cohort of amateur player data was available. Access to such would have enabled further interesting analysis comparing the cohorts: seeing which variables were important in both and how their loading varied between the cohorts.

The Driving aspect was even one where the PCA results were outright unreliable due to distinct information in the variables of the aspect. The PCA is only concerned with maximising the amount of variation explained which is not necessarily the same thing as computing a useful performance metric. As driving is a relatively controlled activity compared to taking shots from elsewhere on the golf course, it may be challenging to remedy this problem by merely finding more variables to include. The linear model approach of predicting the SG variable, given the other variables, worked surprisingly well for the Driving category and could be a viable alternative method of generating a summary. Even still, the linear model was as simple as possible and could have added complexity like interaction terms or data transformations to improve its predictive power, or completely different methods of prediction could be trialled. Asides from Driving, even the other categories with more variables must be questioned because the same quantity is often stratified by distance to the hole. For example, "% on green" was measured five times: from shots taken between 100–125, 125–150, 150–175, 175–200, and 200+ yards from the hole. This was the case for many of the variables in the dataset, and this level of granularity seems somewhat difficult to justify to compute a summary.

From a data engineering perspective, the PCA analysis offered an interesting challenge: while the data had to be converted from long to wide format, the long format did prove efficient for the task of collecting the most up-to-date statistics. By performing a grouping operation on both the player name and the metric's name, the operation of filtering for the most recent metrics was far more efficient than the equivalent computation on the wide-format dataset. After only keeping the most recent statistics and then converting to wide format, the file size was decreased from ∼900MB to just ∼10MB. Part of this increase in memory efficiency was due to only keeping the most recent value for each statistic. However, the wide data format also eliminated much redundant information that was needed in the long format.

To summarise, further work in the realm of PCA would address:

- Exploring more variables than the ones given in the handpicked variable list. These variables were previously known to have already been useful, but other variables in the dataset could have also been useful. At the same time, including more variables may make it onerous on the amateur player to collect all the additional information. Some analysis could be done into how many of the included variables were actually necessary for similarly good summary statistics.

- Similarly, new derived variables could be computed given the right data. For example, "Putts per Green in Regulation" could be derived given shot-by-shot data to compute how many putts were made per regulation landing on the green.

- Finding alternatives to PCA for the Driving aspect, which had unreliable results. For example, using the linear model coefficients as an alternative. Support Vector Machines [33] or Random Forests [34] may be effective alternatives.

- Collecting a cohort of amateur players and comparing the results of the PCA between the cohorts.

- The player rankings could be scraped periodically to have them be most consistent with the statistics reported in the PGA TOUR data. The PCA summary plots (**Figures 3.4** and **8.2**) could also have been improved by incorporating player rank information.

4 Cluster Analysis

Having two options of summary statistics (SG and PC), the next step was to group players into different play styles (e.g., driving-dominant, expert putter). Additionally, this four-dimensional mapping could be used with a distance function for an amateur player to identify which professional golfers most match their play style. This would inform the amateur which golfer's games and training they could receive the most benefit from following and improve engagement with the sport.

4.1 Methodology

The ClusterR package [35] was used to perform K-means clustering on the dataset of PGA TOUR players. As only successful and noteworthy golfers were desired, the dataset was filtered to keep only the top 250 players according to OWGR ranking. The number of clusters K was given different values $K \in \{2, 3, 4, 5, 6\}$ and for each value of K, two experiments were run. Clustering was performed using 1) the four strokes-gained variables and 2) with the four PCA-summarised variables. The goal of the clustering was, therefore, to find groups of similar players in the four-dimensional space mapped by either the strokes gained or PC variables.

Prior to clustering, each player's metrics were adjusted for their overall ability by normalising the player's metrics to have mean 0 and standard deviation 1. Adjustment was necessary to avoid the clustering being more about overall ability than play style.

The clustering results were highly variable, and in general, K-means does not find unique solutions. For each experiment, clustering was performed $N = 5000$ times, and the arrangement with the highest average silhouette width was used as the definitive clustering. The large number of replications made it highly probable that the global optimum was found.

Visualisation

Visualising the clustering of players again used the ggplot and plotly libraries. Identifying the different play styles that each cluster represented was done using box and whisker plots. A box plot was produced for each cluster to show how the play style of players varied between the clusters.

The radar charts in the PCA chapter were used for inter-player comparisons of a player relative to their cohort. Radar charts were also used in the cluster analysis, but were instead focused intra-player comparisons or a "player profile". While the axis limits remained the entire range of values for all players, the data used in the clustering analysis was adjusted for player ability. An extreme value on these axes just meant a player had the strongest preference for that particular golfing aspect relative to the other aspects.

Amateur Recommendation

For an amateur player, it may be interesting to identify a successful professional golfer with a similar play style and closely follow their training and games. This could also lead to improved engagement with the game through an identification with a professional player.

The amateur data was adjusted for player ability in the same way as the professional data: normalising the metrics to have mean 0 and standard deviation 1. Following this the data could be fed through the same clustering pipeline to generate cluster predictions for amateur players. More importantly, professionals with similar play styles could be identified by choosing a suitable distance metric. One example amateur was picked from the dataset, and an investigation was conducted into the best choice of distance metric:

- *L_1-norm*: Poor results, too many players had too similar distances from the amateur.

- *L_2-norm*: Better than the L_1-norm with clearer distinctions between the players. However, the mathematically short distances failed to translate into visually similar player profile plots. This was an issue because it was ultimately a visually similar profile plot that would convince the amateur of the recommendation.

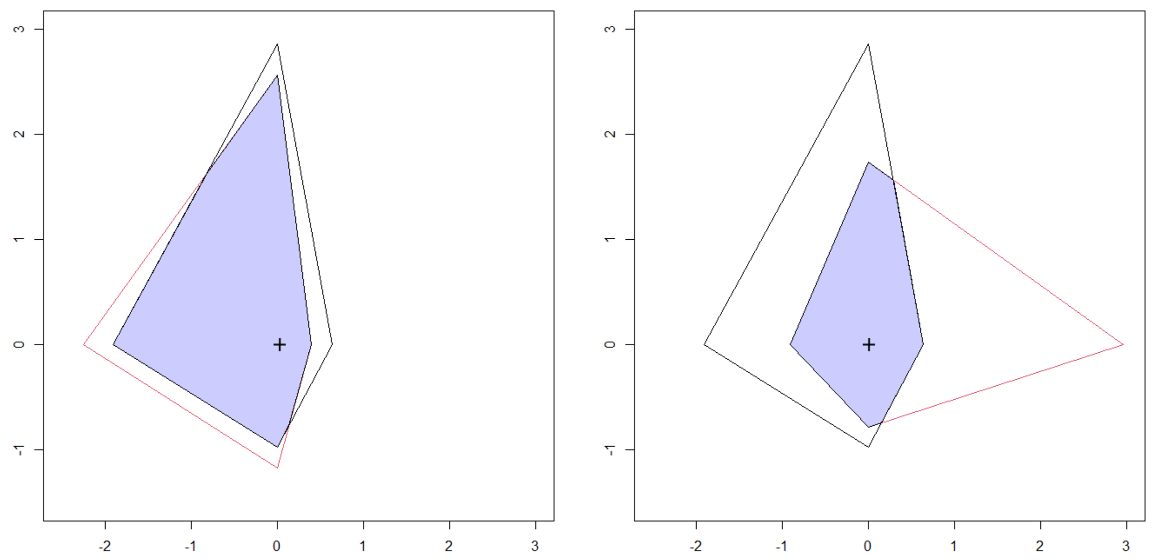
- *Ordinal Matching*: This worked in two steps. First, find all professional players who had the same ordering of metrics. For the given amateur, this meant finding all professional players whose Driving > Putting > Short-game > Long-game. The second step was to arrange the professionals by their OWGR ranking and recommend the highest-ranked players. This worked well at enforcing the idea that recommended players must have similar strengths and weaknesses.

- *Polygon Intersection*: A polygon in 2-D space was constructed out of a player's metrics that was a direct translation of the player's profile plot. By translating the points completely into the visual dimension, the problem of the visual perception of similarity was addressed directly. Given the amateur's polygon and the polygon of a professional player, the area of intersection A_{int} was computed. A_{int} as a proportion p_{int} of the area of the amateur's polygon A_{am} was then used to generate the polygon intersection metric d:

$$p_{\text{int}} = \frac{A_{\text{int}}}{A_{\text{am}}}$$
$$d = \frac{1}{p_{\text{int}}} - 1$$

  Taking the reciprocal of p_{int} and subtracting 1 was necessary to give d the typical interpretation of a distance metric: starting at 0 for a perfect match and increasing the worse the fit (smaller area of intersection). **Figures 4.1a** and **4.1b** illustrate this concept and serve as an example of a good and bad match respectively. Since the player metrics have already been adjusted to remove the effect of player ability, there was no need to adjust for different sized polygons when matching. Experimentation verified that doing so did not provide any noticeable benefit to the matching, confirming that the adjustment for ability was sufficient.

In each case, the recommended results were filtered to only suggest players in the top 50, ensuring relevancy of the results. The polygon intersection was used as a "gold standard" for evaluating the performance of the other distance metrics.



(a) Amateur with Dustin Johnson. (Closest match, $d = 0.12$)

(b) Amateur with Brooks Koepka. (30th closest match, $d = 1.5$)

Figure 4.1: Polygon intersection method.
Key: Black outline = Amateur, Red outline = Professional, Purple shaded = Intersection

4.2 Results

The results of the cluster analysis examine the choice of PC vs SG variables, the choice of K, the resulting cluster configuration, and recommendation to amateurs. Interactive HTML documents containing the full range of plots in this section are hosted on GitHub for both the SG[1] and PC[2] variable analyses.

SG vs PC

With both SG and PC variables available to use as summaries, the first step was to decide which set of variables to use. **Table 4.1** shows the average silhouette width of clusterings performed using the SG or PC variables. Each clustering configuration was performed with 5000 replications and the average silhouette width over each point in the clustering was computed to determine the optimal clustering. The value reported is then the best average silhouette width over every replication. The remainder of this section reports results using PC variables and $K = 4$. While this configuration did have the worst average silhouette width, the differences are minor. The PC variables were preferred due to their accessibility, and $K = 4$ was chosen for interpretability as a compromise between too few and too many clusters. For comparison, Appendix 8.4 contains results of a clustering using $K = 6$.

[1] https://github.com/OptimusPrinceps/Golf-Masters/blob/main/output/cluster_SG.html
[2] https://github.com/OptimusPrinceps/Golf-Masters/blob/main/output/cluster_PC.html

It is important to note for this section that the clustering was done on the data post-adjustment for player ability. That is, a player's "phenotype" was determined by the relative comparison of their abilities in each aspect.

K	2	3	4	5	6
SG	0.37	0.36	0.34	0.36	0.36
PC	0.39	0.35	0.33	0.36	0.36

Table 4.1: Average silhouette width of clustering using SG or PC variables.

Clustering Visualisation

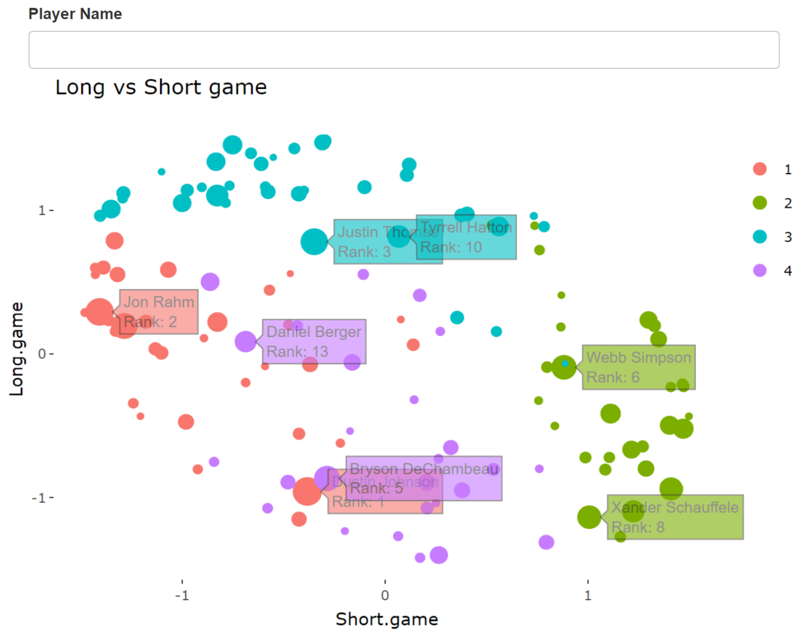
Figure 4.2a shows a visualisation of the clustering using the SG variables and plotted along the Long game and Short game axes. **Figure 4.2b** contains a pairs-style plot for every combination of axes.

This plot also featured the same interactivity of the search box and hovering over points to view the player name and rank. It provided a high-level overview of the clustering, allowing players to be located on the plot and compared to each other, as well as a visual heuristic for judging the quality of the clustering. The top two players by OWGR rank in each cluster are annotated.

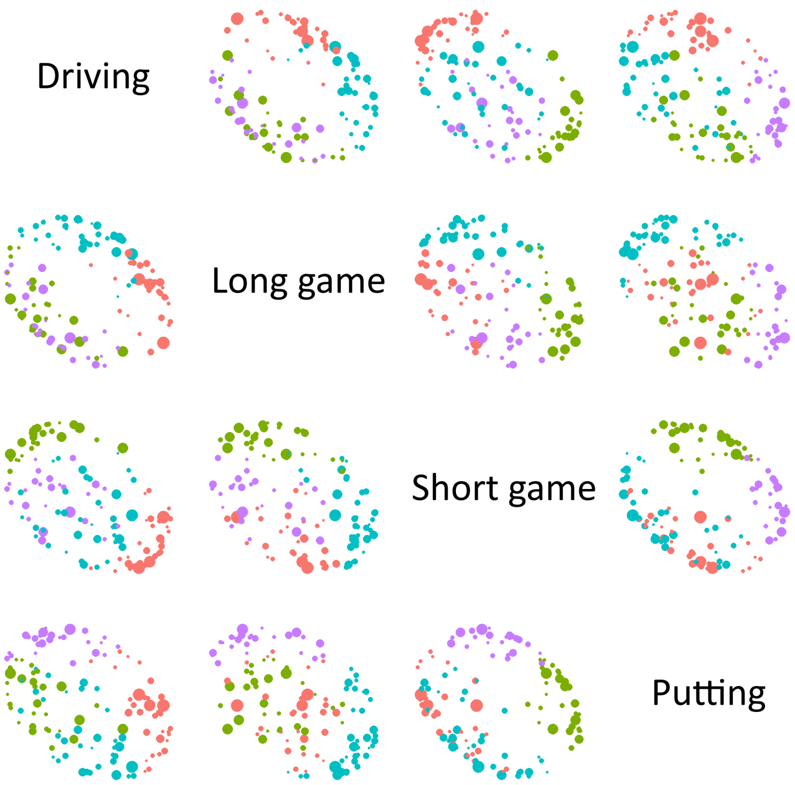
Figure 4.3a shows a box-plot of each cluster. Each cluster in **Figure 4.3a** is described in **Table 4.3b.** Cluster 3 has a much better median rank than the other clusters. Because the data had already been adjusted for player ability, it serves as evidence that players who are strong in Long game are more successful on the tour. This is somewhat consistent with the current prevailing wisdom in golf that hitting the ball further on the initial drive is one of the best ways to achieve a better overall score. Bryson DeChambeau is a player well known for this strategy, however he has actually been assigned to Cluster 4. This indicates that his putting may be much better than perceived and illustrates the insights available by using SG or the PC summary metrics.

Clusters 2 and 4 feature a preference away from Driving and Long game and have the worst median ranks. The silhouette width measures how well a point is suited to its assigned cluster, and the average silhouette width across a cluster measures how well-defined the cluster itself is. Cluster 3 had the "loosest" cluster but also had the largest cluster size.

The health of each cluster is also visualised in **Figure 4.3c**. Cluster 3 shows one player who may have been better suited to a different cluster (negative silhouette width) and the overall low silhouette width of each point in the cluster. Note that while it is possible to reassign this point with the negative silhouette width to its neighbouring cluster, this does not necessarily result in a better clustering configuration. After reassignment, the point could still be ill-suited to the new cluster. This reassignment procedure was implemented but found to have made no difference to the optimal clustering configuration. For this sample size of players, 5000 replications seemed enough to find the optimal configuration.

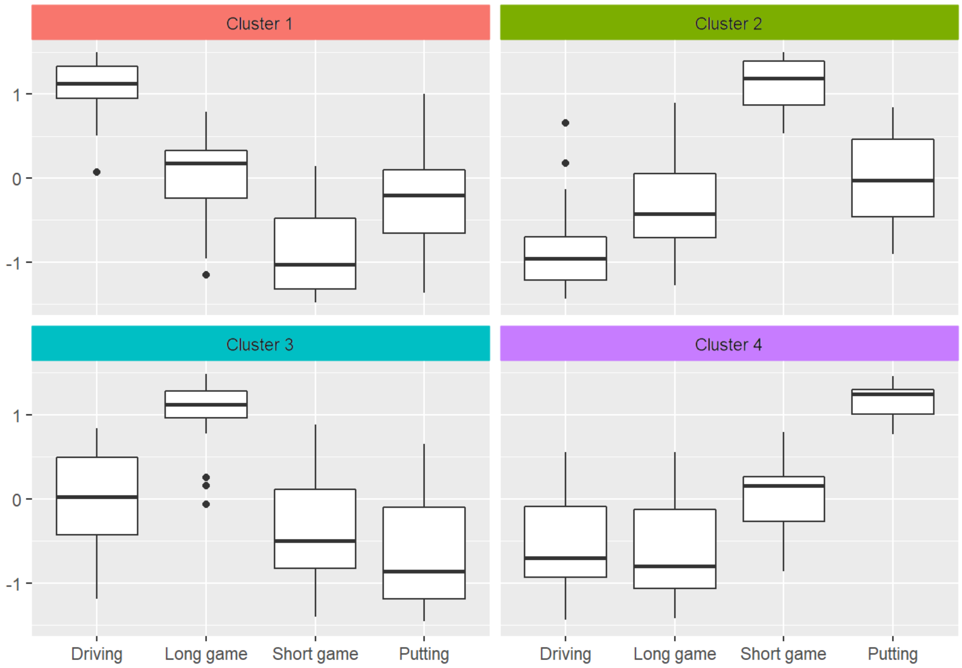


(a) Overall cluster visualisation with top two ranked players labelled in each cluster.



(b) Pairs-plot style clusters.
E.g. the first column has Driving on the x-axis and the first row has Driving on the y-axis.

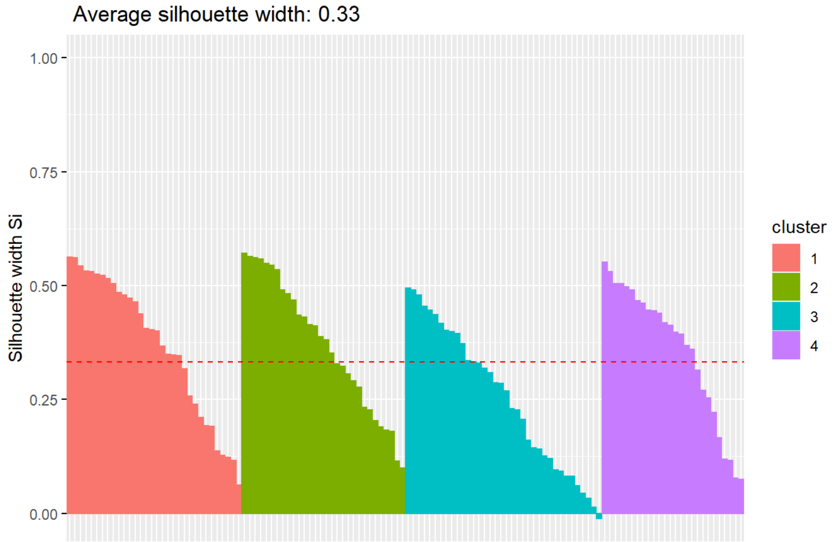
Figure 4.2: Clustering Visualisation for $K = 4$ and PC variables. Larger sized points represent higher ranked players.



(a) Box-plots.

	Player Phenotype	Cluster Size	Median Rank	Avg. Silhouette Width
Cluster 1	Strong Driving, weak Short game	32	97	0.37
Cluster 2	Strong Short game, weak Driving	20	100	0.37
Cluster 3	Strong Long game, weak Putting	36	67.5	0.25
Cluster 4	Strong Putting, weak Long game	26	100	0.36

(b) Summary of each cluster.



(c) Silhouette plot.

Figure 4.3: Cluster configuration for $K = 4$ and PC variables.

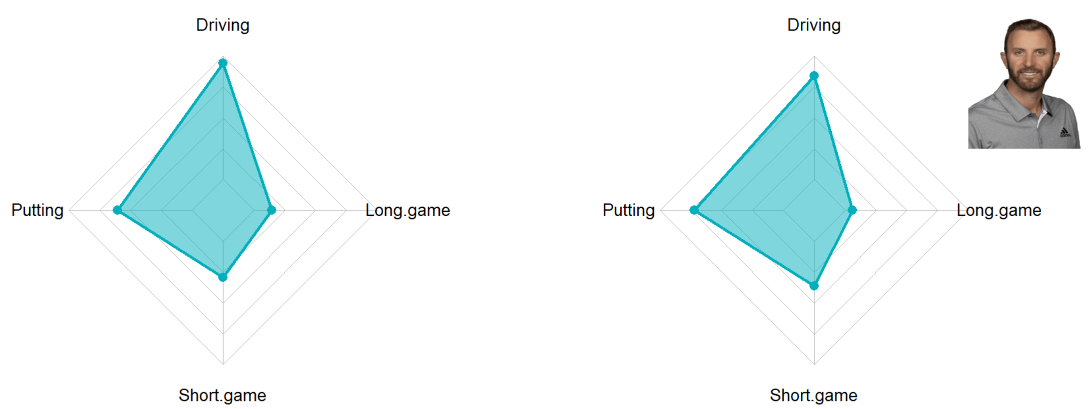
Amateur Recommendation

Using the *polygon intersection* distance metric, **Figures 4.4a** and **4.4b** show the player profiles of Amateur 1 and the closest matching professional. Dustin Johnson was also the top match using the ordinal matching and L2-norm metrics. However, the ordinal matching relied on Dustin's high rank to push him to the top of the list, and the L2-norm's subsequent recommendations were less convincing than those generated by polygon intersection. Using the polygon intersection as a ground truth, **Table 4.2** reports the top five recommended players for each of the distance metrics. The polygon intersection score is used as a "gold standard" to evaluate the performance of the other distance metrics, with a lower value corresponding to a better recommendation. The L2-norm is the next best alternative to polygon intersection, matching the top three recommendations of the polygon intersection exactly. The ordinal matching approach did not work so well on its own but could be used in conjunction with the L2-norm. This would enforce the ordinal constraint while retaining a good measure of distance, However, for these data, this still results in an average top-5 polygon intersection score of 0.47 (i.e. no improvement over using the L2-norm alone).

	L1-Norm		Ordinal Matching		L2-Norm		Polygon Intersection	
1	Xander Schauffele	0.58	Dustin Johnson	0.12	Dustin Johnson	0.12	Dustin Johnson	0.12
2	Rory McIlroy	0.51	Jon Rahm	0.50	Jason Day	0.22	Jason Day	0.22
3	Daniel Berger	0.87	Justin Thomas	2.16	Harris English	0.35	Harris English	0.35
4	Jon Rahm	0.50	Rory McIlroy	0.51	Xander Schauffele	0.58	Bryson DeChambeau	0.37
5	Dustin Johnson	0.12	Bryson DeChambeau	0.37	Bubba Watson	1.08	Jon Rahm	0.50
Avg.	**0.52**		**0.73**		**0.47**		**0.31**	

Table 4.2: Top five recommended players for each distance metric. The polygon intersection score is given as a "gold standard" to evaluate the metric's performance. Lower values correspond to better recommendations using the polygon intersection method.

Given this recommendation, Amateur 1 would recognise that Dustin has a similar play style, and the amateur could follow Dustin's training and games to see how to maximise their game given their strengths and weaknesses. At worst, it still adds a personal connection to a kindred player and deepens their engagement with the sport.



(a) Amateur 1's player profile. **(b)** Dustin Johnson's player profile.

Figure 4.4: Amateur play style based recommendation.

Key Findings

The key findings of the cluster analysis are summarised as follows:

- After adjusting for player ability, it was possible to use clustering to group players by their play style.
- The silhouette width varied little with the choice of SG or PC for summarising variable and of K. This made the decision of which variable set and which K to use somewhat arbitrary. The PC variables and $K = 4$ were chosen for accessibility and interpretability.
- The cluster arrangement maximising silhouette width with $K = 4$ grouped players into four distinct and interpretable categories.
- There was a marked preference for highly ranked players towards Cluster 3 (characterised by being Long game dominant).
- Providing recommendations of similar players is feasible and very effective due to the polygon intersection distance metric.

4.3 Discussion

The objective of the cluster analysis was to adjust for player ability and group players by their play styles. The main application area of this is in those interested in and following the sport of golf: grouping players by similar play styles provides an additional layer of analytic value. Trends could be deduced not just on individual players, but on groups of players with similar play styles and this information conflated with the types of courses that are being played in tournaments and what kind of players they favour.

Adjustment for the player's ability was also required in order to avoid their ability becoming a confounder. The results showed that the adjustment was largely successful and provided meaningful insight into the preferences of different players. However, because the adjustment enforced constant variance of the metrics, well-rounded players ended up with exaggerated differences in their profile plots. This could be rectified by exploring alternative adjustment transformations.

The choice of K and whether to use the SG or the PC summarising variables was difficult to make. Each decision had multiple viable options, though in the end it came down to accessibility and interpretability: the PC variables are broadly available, and $K = 4$ provided distinct and sensibly defined clusters. Rather than only having hard clusters (where a player is in one cluster only), the clustering could be easily extended to allow players to exist on a continuum towards each cluster. For example, rather than classifying a player as being “Putting dominant”, a second closest cluster could further describe the player, such as having a major preference for the Putting-dominant cluster, with a minor preference for the Long game-dominant cluster. This problem would also be solved by having more clusters that naturally separate into groups like this, but the small number of players available made it unreasonable to have K too large. Incorporating player data from other years would help to alleviate this problem and allow for very interesting visualisations and a whole new longitudinal class of analysis. A plot like **Figure 3.4** or **4.2a** animated to show how the performance/play style of players evolved over multiple years of playing golf would be highly interesting. Performing the clustering a large number of times and taking the best configuration helped to at least improve the results' reliability.

The player recommendation was highly successful due to the power of the polygon intersection method. However, the same question arises of whether to use SG or PC variables when computing the distance metric. Interpretability is still a concern, but less so given that the user will be visually confronted with the player profile charts reinforcing the recommendation. In this case, it may be preferable to use the PC variables given that they do not require SG data and the primary audience for the recommendations will be amateurs. For analytics platforms that natively compute SG metrics, an experiment could be to roll out to different groups of users recommendations based on the SG and the PC variables separately and collect user feedback to judge which variable set is more effective. The ideal scenario is a user that, upon receiving a recommendation of a professional player with a similar play style, follows that player's games and training and then applies what they learn to improve their own game. Unfortunately, as only very little amateur data was available, it was difficult to assess the efficacy of the recommendations over multiple players.

One limitation of the data available for this project was that the OWGR ranking was only available for the end of 2020, while the PGA TOUR dataset was collected over 2019. The OWGR keeps an archive of previous rankings, but the link to the rankings for 2019 were broken. The OWGR ranking data would need to be matched with up-to-date PGA TOUR statistics for the most relevant visualisations. Currently, the visualisations and inference based on the player rank may be misleading. Players were also filtered out by rank before the clustering, and some players were also dropped due to missing data in the PGA TOUR dataset. The impact of dropping these players could be investigated further, and techniques like imputation used to address missing data.

To summarise, further work in the realm of the cluster analysis would address:

- Alternative methods of adjusting for player ability that retain relative differences in metrics.
- Incorporating more player data from multiple years and extending the analysis to be longitudinal.
- Collecting a cohort of amateur data to assess the efficacy of the recommendations. Furthermore, an assessment on whether SG or PC based player recommendations are preferred by end-users.
- Matching of timeframes of PGA TOUR data and OWGR rankings.
- Analysis on the rank cut-off for players to be included in the cluster analysis, and the effect of dropping players with missing data.

5 Course Difficulty Analysis

The PCA chapter used a range of player metrics to create an alternative summary statistic for players in cases where strokes gained data is not available. The Clustering chapter then looked at using summary statistics to group players by play style. This chapter examines using course ratings to adjust for course difficulty and make player comparisons when neither detailed metrics nor summary statistics are available. Creating new comparisons between players provides more visibility and allows players to better rate their ability even when playing on completely different courses. For example, a player in New Zealand can currently only guess how good they are compared to a college golfer in the US.

Following the background on data pre-processing, this chapter is broken into two sections: 1) comparing the college players to scratch golfers and 2) adjusting player scores for the difficulty of the course.

5.1 Dataset and Pre-processing

The PGA TOUR dataset used in the previous two chapters contained statistics on players averaged over many of their games. In contrast, this chapter considers data on a hole-by-hole basis.

The National Collegiate Athletic Association (NCAA) runs college golf tournaments in the United States. A dataset of two CSV files (∼150MB each for males and females) had previously been scraped from their website. The NCAA dataset contained the following columns: player name, tournament name, venue, the round, the hole number, the total distance of the hole, the par score for the hole, and the player's score on the given hole. Players in this dataset play across three divisions: I, II, and III.

One of the main issues with comparing golf scores is in accounting for course difficulty. The strokes gained metric manages to address this but requires a formula and hole-by-hole or even shot-by-shot distances. The course rating can be used instead to account for difficulty when SG cannot be calculated. The NCAA dataset did not come with course ratings. Instead, the venue names it contained were used to search the US Golf Association's National Course Rating Database (NCRDB) [36] for the course rating. However, unfortunately, the search results were retrieved dynamically. A significant amount of time could easily have been invested into reverse-engineering the requests and headers necessary to retrieve the results directly without any guarantee of success. Instead, a Docker [13] container was set up to run a headless version of the Firefox web browser and RSelenium used to simulate human interaction with the browser.

The RSelenium [12] package was then used to provide programmatic control over the browser. Once the page had loaded, the page's source was used to locate the text field element for search query input. The venue name was injected into the element, and the search button clicked by the program. At this point, `R` was instructed to wait 5 seconds for the results to be retrieved and dynamically loaded onto the webpage before the page source could then again be inspected for the links to the course rating pages returned by the result of the query.

The code was able to run without any human intervention. However, due to a combination of the overhead involved in running the Firefox browser and time spent waiting for the search query results, it required 4-5 hours to run to completion. Fortunately, no captchas were encountered that could have slowed the process down even further. Results were stored at each step and re-used if necessary to avoid repetition of tasks. Error handling was also put into place that restarted the browser process in the event of a crash. Unfortunately, some of the course names from the dataset provided did not match the names on the course rating database website and failed to return any results. In many cases, this was due to the search string being too specific. For these cases, a complex regular expression was developed to format the string before submitting the search query, removing many words and symbols that did not help distinguish the unique course name. Further details on this are available in Appendix 8.5.

The course rating pages themselves were simple static web pages. The page source contained an HTML table of course ratings that could be read directly into `R`, and the appropriate values extracted. Hence, a separate script was written to load the output of the dynamic web scraping and populate a dataset of course ratings for males and females in under 10 minutes. This code was kept independent of the code for retrieving the course IDs to avoid running 4-5 hours worth of code before executing something much more straightforward.

Due to the mismatch of the course names in the NCAA and the online dataset, of the 496 distinct courses in the NCAA datasets, 337 (68%) were matched with course ratings. As the purpose of this analysis was to adjust for course difficulty, players who had only played on one course had to be dropped. **Figure 5.1** shows the distribution of the number of unique venues played. After excluding ∼4,500 players, the final NCAA dataset contained 3,083 males and 2,691 females (5,774 total) who had played more than one course. In total, there were 975,036 holes of data.

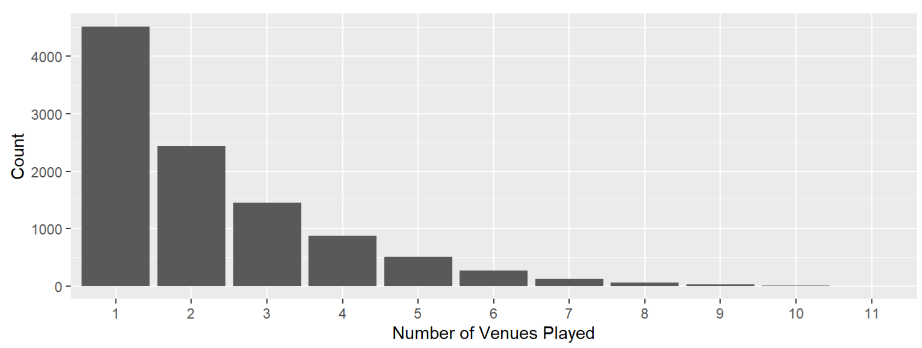


Figure 5.1: Distribution of the number of unique venues played by each player in the NCAA dataset.

5.2 Comparison to Scratch

The first area of investigation with the NCAA dataset was comparing college golfers to scratch players. The total score for the round was calculated, with **Figure 5.2** visualising the total score against the course rating. In cases where a player may have played on the same course multiple times, the tournament ID uniquely identifies games. Few courses have ratings below 70 and above 80, but compared to the typical par of around 72 this means most courses in the NCAA dataset are more difficult than average. As the course ratings are defined as the number of strokes, a scratch player would require to complete a round on the course, the red dashed line (gradient of 1) shows the expected total score of a scratch player. The green smoothed line shows how the NCAA amateurs actually performed. On easier courses the amateurs are scoring worse (higher score) but start to approach scratch level for the harder courses. This may suggest that some courses have a reputation of being easier or harder and are particularly played by lower or higher skill players respectively. Alternatively, it may even suggest that the course rating is not actually informative of difficulty for these players. A mixed model could be used to formally test whether the course rating has an effect on the to-par scores of players. For example, with the course rating as a fixed effect and the player as a random effect, a non-zero coefficient for course rating would provide evidence that it does inform player scores. This is left as future work.

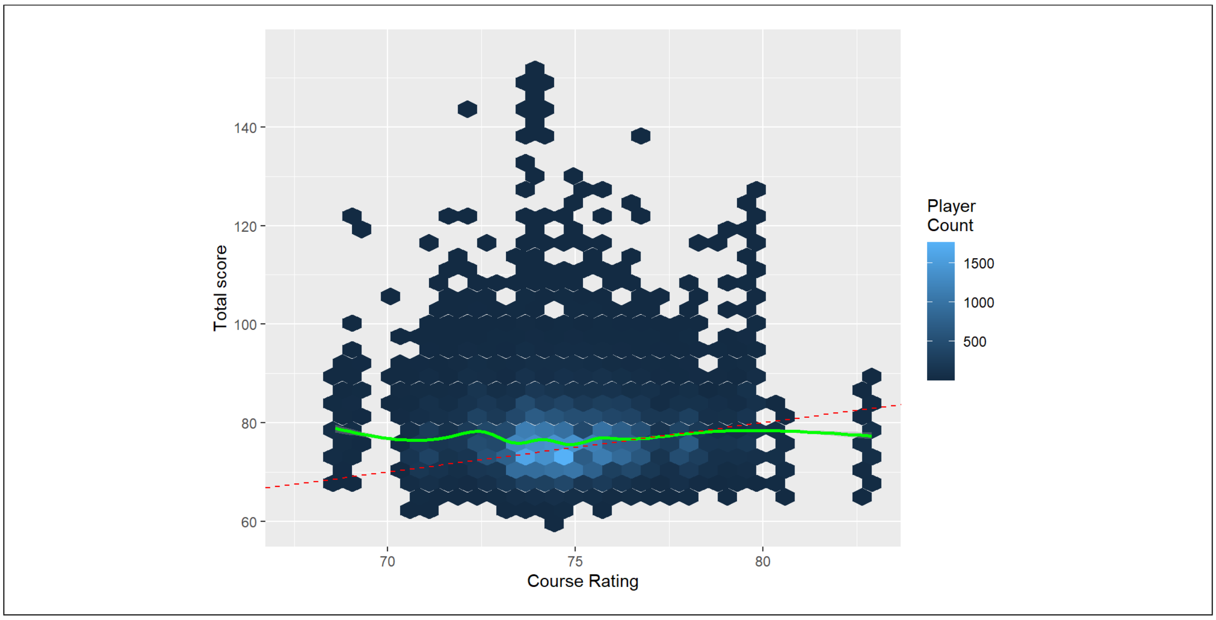


Figure 5.2: Player score vs course rating. GAM smoother fitted in green. Red dashed line is expected total score for a scratch player.

The difference between the course rating and total score was taken to compare the college golfers to scratch players. **Figure 5.3** shows the distribution of strokes per round relative to scratch for college golfers. A round of golf typically has a par of 72 and 18 holes, so a strokes relative to scratch of 18 would mean the college player is, on average, taking one additional shot compared to the scratch player per hole. **Table 5.1** gives additional quantile values and shows that the top 31.6% of college players in the dataset are at or better than scratch. There is a right-tail of some poorer players needing more shots to complete the round. The worst seen was 104 shots above scratch, but is a major outlier.

Top	10%	25%	31.6%	50%	75%	90%	
Strokes above scratch	-2.38	-0.63	0		1.95	5.65	11.72

Table 5.1: Quantile table of number of strokes above scratch per round for college golfers.

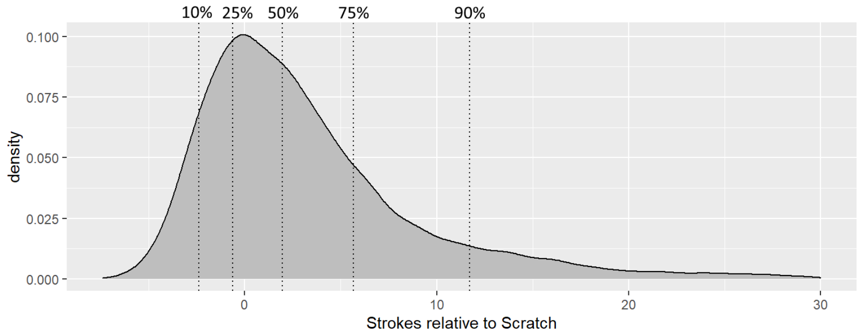


Figure 5.3: Number of strokes per round of college golfers relative to scratch golfers with labelled quantiles. *Note:* the X-axis has been truncated at 30.

5.3 Course Difficulty Adjustment

The second area of investigation with the NCAA data was to adjust player score-to-par performance for the difficulty of the course. In addition to the course ratings, the expected number of strokes required to complete the hole (from the tee) was computed using the strokes gained formula and the distance of each hole. This served as another adjusted par score and provided a comparison against the adjustments from the course rating. The analysis is defined formally as follows:

Let P be the set of all players, T the set of college golf tournaments, C the set of all courses, R the set of rounds of golf, and H the set of holes played. Now define $s_{p,t,c,r,h}$ to be the score of player $p \in P$ in tournament t on course $c \in C$ in round $r \in R$ on hole $h \in H$. Let $x_{t,c,r,h}$ denote the par score for hole h in round r of course c in tournament t.

For round r of a course c with rating CR_c, the course rating adjustment factor $\text{CRadj}_{c,r}$ was the course rating divided by the par for the round for any $t_1 \in T$:

$$\text{CRadj}_{c,r} = \frac{\text{CR}_c}{\sum_h x_{t_1,c,r,h}}$$

As the course rating is defined as the expected number of strokes required for a scratch player to complete the round r of course c, $\text{CRadj}_{c,r} > 1$ if c is more difficult than usual and $\text{CRadj}_{c,r} < 1$ if c is easier than usual.

For hole h of round r on course c, let $SG(h)$ return the expected number of strokes needed for a scratch player to complete hole h of known total distance via the SG metric.

Now compute the average scores-to-par across a course for each player and each course using the mean μ, where $\text{CourseRating}_{p,c}$ and $\text{SGscratch}_{p,c}$ are the scores-to-par adjusted by course rating and course distance respectively..

$$\begin{aligned}
\text{Raw}_{p,c} &= \underset{t,r,h}{\mu} \left[s_{p,c,r,h} - x_{c,r,h} \right] \\
\text{CourseRating}_{p,c} &= \underset{t,r,h}{\mu} \left[(s_{p,c,r,h} \times \text{CRadj}_{c,r}) - x_{c,r,h} \right] \\
\text{SGscratch}_{p,c} &= \underset{t,r,h}{\mu} \left[s_{p,c,r,h} - SG(h) \right]
\end{aligned}$$

Figure 5.4 shows the mean to-par scores over every player $\text{Raw}_{p,c}$, $\text{CourseRating}_{p,c}$, and $\text{SGscratch}_{p,c}$. Both adjustments have decreased the average score on the course, indicating that most courses are indeed harder than par. At this point a mixed model could be used to formally test the effect of the course rating on the to-par scores.

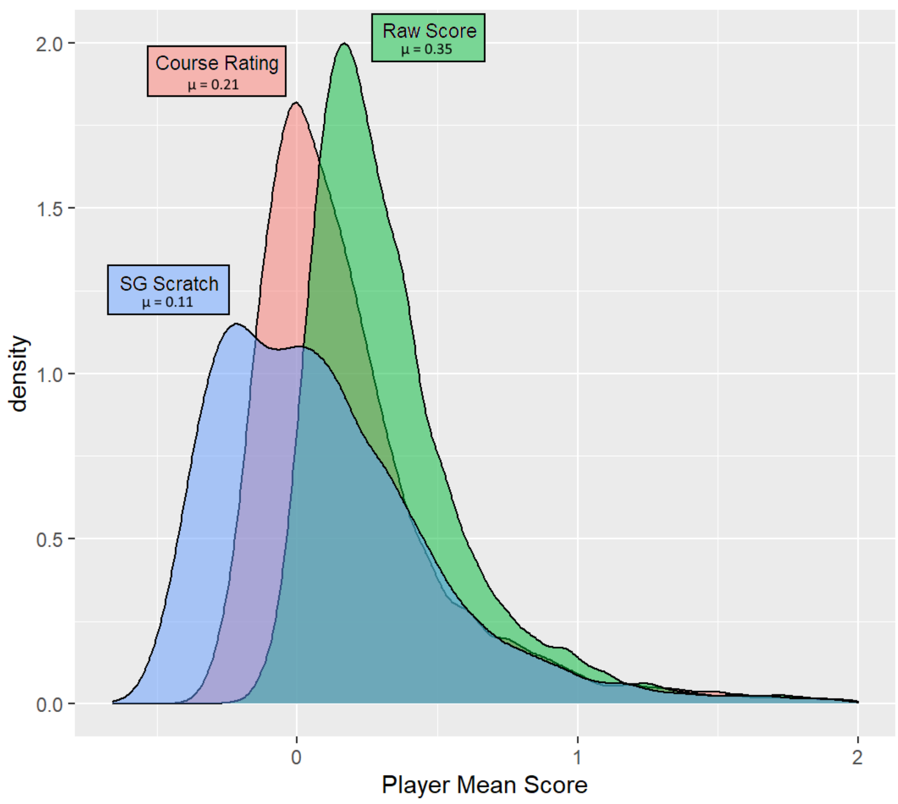


Figure 5.4: Mean to-par score over every player including adjustments. The mean of each distribution is annotated. *Note*: the X-axis has been truncated at 2.

The difference in standard deviations between the adjusted scores-to-par and the raw score-to-par for each player is then denoted by Δ. This measures how the within-player score variability has changed relative to using the unadjusted score-to-par. At this point, a mixed model could again be used with a random player effect to formally test whether there is any significant effect due to the adjustment. This is left as future work.

$$\Delta_p^{CR} = \sigma_c(\text{CourseRating}_{p,c}) - \sigma_c(\text{Raw}_{p,c})$$
$$\Delta_p^{SG} = \sigma_c(\text{SGscratch}_{p,c}) - \sigma_c(\text{Raw}_{p,c})$$

$\sigma_c(\text{CourseRating}_{p,c})$ and $\sigma_c(\text{SGscratch}_{p,c})$ are then measures of the within-player score variability after adjustment. **Figure 5.5** visualises each Δ to show how the within-player score variability has changed from the raw score-to-par baseline. Ideally, after adjustment, the within-player score variability would have decreased as the scoring bias due to the difficulty of the course was removed. Unfortunately, both methods of adjustment seem to increase the within-player score variability with means above 0. The SG adjustment seems to have made little difference, while the course rating adjustment is quite varied in that it can make a player's score variation better or much worse.

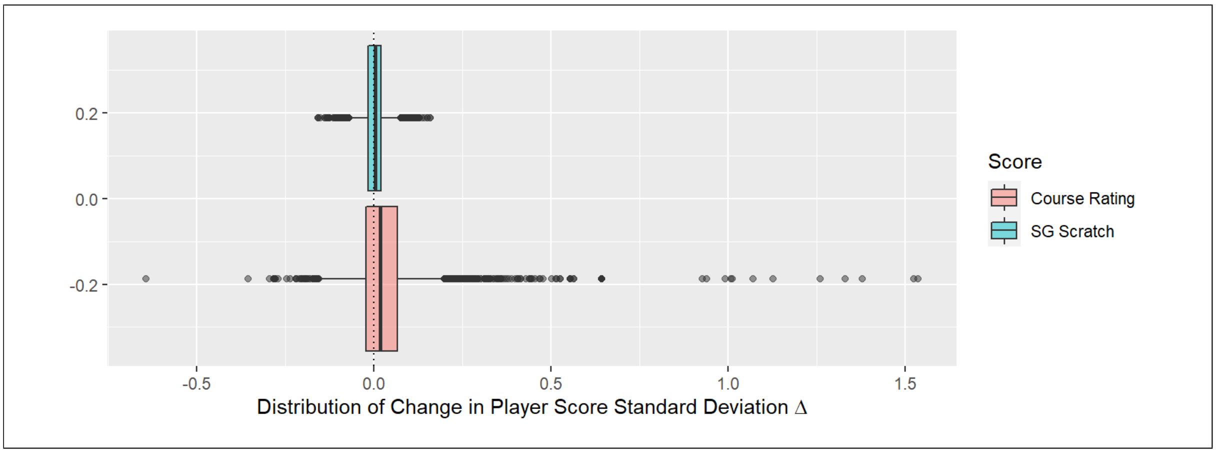


Figure 5.5: Distribution of within player change in scoring standard deviation Δ_p^{CR} and Δ_p^{SG}.

5.4 Discussion

The objectives of the course difficulty analysis were to compare college golfers against scratch golfers and against each other, having removed the bias caused by the difficulty of the course. Comparing college golfers against scratch uses an established benchmark to provide a new level of comparison for performance in golfing. For example, a young player in New Zealand playing at a scratch level would immediately know they were in something like the top 30% of college golfers and know that a golfing scholarship to an American university could be within reach. Similarly, adjusting for the difficulty of the courses allows for comparisons of games and players on courses of varying difficulty. For example, a player scoring a par of 72 at their local golf course may not necessarily score par at a course played by the PGA TOUR.

The NCAA dataset contained players of a range of ability that somewhat limits the value of a benchmark developed on this dataset. Removing players that have not played multiple courses has addressed this by only keeping players with some consistent showing in tournaments, however, the scope of the analysis could be further reduced to only NCAA players in division I. Data on players in each division is available and doing so would yield a more pragmatic benchmark. Alternatively, using the current dataset, only the top-scoring players in each tournament could be kept.

Unfortunately, adjusting for the difficulty of the course did not have the desired effect. Adjusting for course difficulty is a high-yield, but difficult problem and even course ratings designed specifically for this purpose have not managed to solve it. Strokes gained only considers the distance of the hole but incorporating additional information reduces its efficacy as an interpretable metric. Still, something like a "course-adjusted strokes gained" would be a highly valuable metric for comparison. In relation to the PCA analysis, having variables that factored in the course difficulty could help to improve the results and accuracy of the metrics of comparison. Even without a variable adjusted explicitly for course difficulty, something like the average course rating of courses a player has played on would be insightful. This would not work with the existing PGA TOUR dataset for this project but could conceivably be collected and incorporated in the future.

To summarise, further work in the realm of the course difficulty analysis would address:

- A significant amount of effort was put into matching the course names with those in the online database, but this could still be improved with some manual inspection and better accounting for every edge case.
- Mixed models could be used to formally test the effects of course rating and the adjustments on scores and scoring variation respectively.
- The scope of the NCAA dataset could be restricted to only high-performing college golfers by computing tournament rankings of players or filtering based on NCAA division.
- Course ratings could be collected for PGA tour players and incorporated into the PCA analysis.
- Ongoing work into how to effectively adjust for the difficulty of the course.

6 Summary and Conclusion

Golf is a growing sport with many new and existing players interested in improving their play each year. Analytics platforms exist that allow amateur and professional golfers alike to log their data and receive personalised feedback and training plans. Over the course of a semester, this project has improved on the analytics and insight available to players of golf through three major areas of work.

The Principal Component Analysis showed that it was possible to reconstruct summarising metrics with similar explanatory power to the strokes gained metric. This means that in cases where strokes gained data is not available, players can still receive powerful summaries of their performance in each aspect of their game and overall. While the PCA was less effective for the Driving aspect, a viable alternative was identified.

The cluster analysis successfully adjusted for player ability and grouped players based on their play style. An interpretable set of clusters was identified and amateurs were also able to use their data to be matched with a professional golfer of similar play style, to suggest how following that professional may provide insight into improving their play and deepen their engagement with the sport.

The course difficulty analysis aimed to remove the bias of differing course difficulty. In the process, a new benchmark was established comparing college golfers to scratch players. Adjusting for the difficulty of a course is a valuable prospect but remains a difficult problem with further investigation required.

As a part of the process, this work has also identified challenges, limitations, and opportunities for further work. By integrating the novel metrics, visualisations, recommendations, and benchmarks produced in this work into analytics platforms, golf players around the world can benefit from additional engagement and insight into their game.

7 References

[1] Scottish Golf History. *Oldest Golf Courses*. 2021. URL: https://www.scottishgolfhistory.org/oldest-golf-courses/ (visited on 09/15/2021).

[2] National Golf Foundation. *Golf Industry Facts*. 2021. URL: https://www.ngf.org/golf-industry-research/ (visited on 09/15/2021).

[3] Olympics - Tokyo 2020. *Golf — Olympic Sport*. 2021. URL: https://olympics.com/tokyo-2020/en/sports/golf/ (visited on 09/15/2021).

[4] Rio 2016. *Olympic Golf*. 2021. URL: https://web.archive.org/web/20160921082103/https://www.rio2016.com/en/golf (visited on 09/15/2021).

[5] Golf Monthly. *How Many Golf Courses Are There In The World?* 2021. URL: https://www.golfmonthly.com/features/the-game/how-many-golf-courses-are-there-in-the-world-182153 (visited on 09/15/2021).

[6] "Assessing Golfer Performance on the PGA Tour". In: *Interfaces* 42 (Feb. 2011). DOI: 10.2307/41472743.

[7] M. Broadie. "Assessing Golfer Performance Using Golfmetrics". In: 2008.

[8] United States Golf Association. *USGA*. 2021. URL: https://www.usga.org/content/usga/home-page.html (visited on 08/27/2021).

[9] J. A. Hartigan and M. A. Wong. "Algorithm AS 136: A K-Means Clustering Algorithm". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1 (1979), pp. 100–108. ISSN: 00359254, 14679876. URL: http://www.jstor.org/stable/2346830.

[10] Evelyn Fix and J. L. Hodges. "Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties". In: *International Statistical Review / Revue Internationale de Statistique* 57.3 (1989), pp. 238–247. ISSN: 03067734, 17515823. URL: http://www.jstor.org/stable/1403797.

[11] Peter J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. URL: https://www.sciencedirect.com/science/article/pii/0377042787901257.

[12] John Harrison. *RSelenium: R Bindings for 'Selenium WebDriver'*. R package version 1.7.7. 2020. URL: https://CRAN.R-project.org/package=RSelenium.

[13] Docker. *Docker*. 2021. URL: https://www.docker.com/ (visited on 08/27/2021).

[14] Alexander B. Cohen, Gershon Tenenbaum, and R. William English. "Emotions and Golf Performance: An IZOF-Based Applied Sport Psychology Case Study". In: *Behavior Modification* 30.3 (2006). PMID: 16574814, pp. 259–280. DOI: 10.1177/0145445503261174. eprint: https://doi.org/10.1177/0145445503261174. URL: https://doi.org/10.1177/0145445503261174.

[15] Terrence P Clark, Ian R Tofler, and Michael T Lardon. "The sport psychiatrist and golf". In: *Clinics in sports medicine* 24.4 (2005), pp. 959–971.

[16] Michael M Reinold et al. “Interval sport programs: guidelines for baseball, tennis, and golf”. In: *Journal of Orthopaedic & Sports Physical Therapy* 32.6 (2002), pp. 293–298.

[17] Hassan Ghasemzadeh et al. “Sport training using body sensor networks: A statistical approach to measure wrist rotation for golf swing”. In: *Proceedings of the Fourth International Conference on Body Area Networks*. 2009, pp. 1–8.

[18] Popi Sotiriadou. “Sport development planning: The sunny golf club”. In: *Sport Management Review* 16.4 (2013), pp. 514–523.

[19] Brian Stoddart. “Sport, television, interpretation, and practice reconsidered: Televised golf and analytical orthodoxies”. In: *Journal of Sport and Social Issues* 18.1 (1994), pp. 76–88.

[20] Brad Millington and Brian Wilson. *The greening of golf: Sport, globalization and the environment*. Manchester University Press, 2016.

[21] Kit Wheeler and John Nauright. “A global perspective on the environmental impact of golf”. In: *Sport in society* 9.3 (2006), pp. 427–443.

[22] Charles C Lim and Ian Patterson. “Sport tourism on the islands: The impact of an international mega golf event”. In: *Journal of Sport & Tourism* 13.2 (2008), pp. 115–133.

[23] Journal of Sports Analytics. *Journal of Sports Analytics*. 2021. URL: `http://journalofsportsanalytics.com/` (visited on 11/12/2021).

[24] Journal of Quantitative Analysis in Sports. *Journal of Quantitative Analysis in Sports*. 2021. URL: `https://www.degruyter.com/journal/key/jqas/html` (visited on 11/12/2021).

[25] Kasra Yousefi and Tim B. Swartz. “Advanced putting metrics in golf”. In: *Journal of Quantitative Analysis in Sports* 9.3 (2013), pp. 239–248. DOI: `doi:10.1515/jqas-2013-0010`. URL: `https://doi.org/10.1515/jqas-2013-0010`.

[26] Timothy CY Chan, David Madras, and Martin L Puterman. “Improving fairness in match play golf through enhanced handicap allocation”. In: *Journal of Sports Analytics* 4.4 (2018), pp. 251–262.

[27] Christian Drappi and Lance Co Ting Keh. “Predicting golf scores at the shot level”. In: *Journal of Sports Analytics* 5.2 (2019), pp. 65–73.

[28] OFFICIAL WORLD GOLF RANKING. *Official World Golf Ranking*. 2021. URL: `owgr.com/ranking` (visited on 08/09/2021).

[29] PGA TOUR. *Statistics*. 2021. URL: `https://www.pgatour.com/stats.html` (visited on 08/09/2021).

[30] World Health Organisation. *Coronavirus disease (COVID-19) pandemic*. 2021. URL: `https://www.who.int/emergencies/diseases/novel-coronavirus-2019` (visited on 09/27/2021).

[31] Carson Sievert. *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC, 2020. ISBN: 9781138331457. URL: `https://plotly-r.com`.

[32] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN: 978-3-319-24277-4. URL: `https://ggplot2.tidyverse.org`.

[33] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine Learning* 20 (1995), pp. 273–297. DOI: `https://doi.org/10.1007/BF00994018`.

[34] Leo Breiman. “Random Forests”. In: *Machine Learning* 45 (2001), pp. 5–32. DOI: https://doi.org/10.1023/A:1010933404324.

[35] Lampros Mouselimis. *ClusterR: Gaussian Mixture Models, K-Means, Mini-Batch-Kmeans, K-Medoids and Affinity Propagation Clustering*. R package version 1.2.5. 2021. URL: https://CRAN.R-project.org/package=ClusterR.

[36] United States Golf Association. *Course Rating and Slope Database*. 2021. URL: https://ncrdb.usga.org/ (visited on 08/27/2021).

8 Appendices

8.1 Complete List of PGA TOUR Variables

Table 8.1 contains the name, golf aspect, and short description of every variable used in the analyses on the PGA TOUR dataset. All variables are averages over all of the games played by a golfer in the 2019 season.

Abbreviations and terms used in the table are defined:

- *GIR* (Green in Regulation): ball hit onto the green with at least two strokes left to score par.
- *Scrambling*: Scoring a par or better after landing in a greenside bunker.
- *Sandsave*: Rate at which the player sinks the ball within two shots from a greenside bunker (regardless of final score).
- *One-putt*: Sinking the ball in a single putt from the green.
- *Three-putt avoidance*: Sinking the ball in two or less putts.

Aspect	Name	Description
General	Score vs par	Number of strokes taken minus hole par
	SG: Total	Overall strokes gained over the entire hole
	Birdie or better	Rate of scoring 1 or more better than par
	Par 3 - Birdie or better	Birdie rate on par 3 holes
	Par 4 - Birdie or better	
	Par 5 - Birdie or better	
	Bogey or worse	Rate of scoring 1 or more or worse than par
	Par 3 Average score	Average score on par 3 holes
	Par 4 Average score	
	Par 5 Average score	
Driving	SG: Driving	Strokes gained on drives
	Driving distance	Driving distance in yards
	Fairways hit	Rate of balls hit onto fairway from the tee
	GIR	(See description above)
Long game	SG: Long game	Strokes gained on shots >100 yards from hole
	GIR: 100-125 yds	Rate of GIRs from shots taken 100-125 yards from hole
	GIR: 125-150 yds	
	GIR: 150-175 yds	
	GIR: 175-200 yds	
	GIR: >200 yds	
	Proximity to hole: 100-125 yds	Remaining dist. to hole of shots taken 100-125 yds from hole
	Proximity to hole: 125-150 yds	
	Proximity to hole: 150-175 yds	
	Proximity to hole: 175-200 yds	
	Proximity to hole: 200-225 yards	
	Proximity to hole: 225-250 yards	
Short game	SG: Short game	Strokes gained on shots <100 yards from hole
	Scrambling	(See description above)
	Scrambling: <10 yds	
	Scrambling: 10-20 yds	
	Scrambling: 20-30 yds	
	Scrambling: >30 yds	
	Sandsaves	(See description above)
	GIR: <75 yds	
	GIR: 75-100 yds	
	Proximity to hole: 50-75 yds	
	Proximity to hole: 75-100 yds	
	Proximity to hole: bunkers	Remaining dist. to hole of shots taken from bunkers/sand
Putting	SG: Putting	Strokes gained on shots taken on the green
	One-putt: all distances	(See description above)
	One-putt: <5 yds	
	One-putt: 5-10 yds	
	One-putt: 10-15 yds	
	One-putt: 15-20 yds	
	One-putt: 20-25 yds	
	One-putt: >25 yds	
	Three-putt avoidance	(See description above)
	Three-putt avoidance: <5 yds	
	Three-putt avoidance: 5-10 yds	
	Three-putt avoidance: 10-15 yds	
	Three-putt avoidance: 15-20 yds	
	Three-putt avoidance: 20-25 yds	
	Three-putt avoidance: >25 yds	
	Birdie conversion	Rate at which a player successfully putts a birdie or better
	Putting average	Average number of putts a player will make on the green
	Total distance of all putts	Total distance of all putts made per round in inches

Table 8.1: Complete list of variables and descriptions

8.2 Percentage of Variance Explained

For each aspect, a PCA was conducted with every variable included and compared against a PCA without the SG variable included. **Figure 8.1** shows the percentage of variance in the data explained by PC1 with and without the SG variable included.

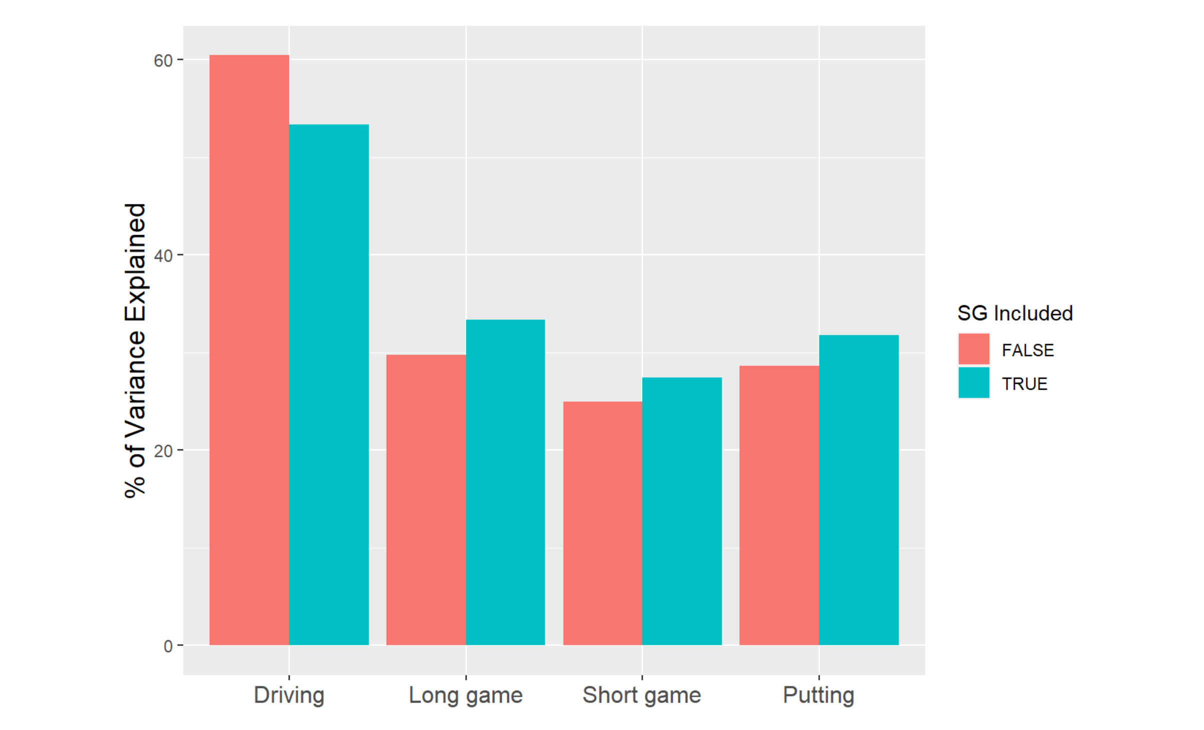


Figure 8.1: Percentage of variance in the data explained by PC1 with and without including the SG variables.

Interpreting the change in variance explained when including the SG variable would be bad practice as the underlying data used to train the two sets of PCAs has changed.

In every aspect, the percentage of variance explained by PC1 is reasonable enough for it to be a worthwhile summary. The Driving aspect in particular had a high amount of variance explained.

8.3 PCA applied to Amateur Data

Figure 8.2 shows the result of running the existing PCA (trained on professional players) on an amateur player (pseudonymised as Amateur 1). Amateur 1 was reasonable at golf, on average scoring a few strokes above par. However, when compared to professional PGA TOUR players, Amateur 1 was a complete outlier. It is no surprise that the amateur has been placed at the very bottom of PC1 in every plot. This is even despite the strokes gained variable for the amateur being relative to scratch golfers (those who play at par or better), while the strokes gained variable for the players in the PGA TOUR dataset is relative to PGA TOUR professionals. By incorporating other variables into the PCA summaries, a comparison can still be made despite the different definitions of strokes gained. This comparison would be far more reasonable for a player of ability closer to a PGA TOUR professional.

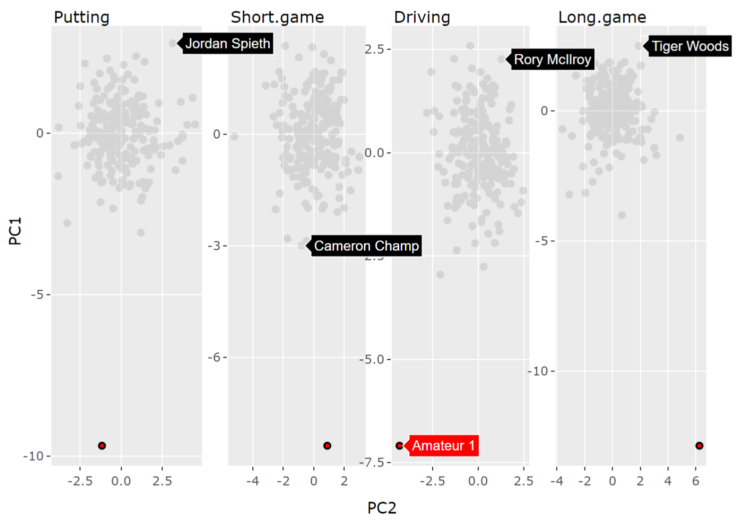


Figure 8.2: PCA Summary with amateur data included. The axes have been normalised and so are interpretable as "standard deviations from the mean"
Note: Additional players have been manually labelled in black in lieu of an interactivity.

8.4 Clustering with $K = 6$

Figures 8.3 and **8.4** show the boxplot and silhouette plot of a clustering using the PC variables and $K = 6$. **Table 8.2** summarises the properties of each of the clusters.

	Cluster Size	Median Rank	Avg. Silhouette Width
Cluster 1	22	55.5	0.42
Cluster 2	15	97	0.27
Cluster 3	19	136	0.46
Cluster 4	28	104	0.36
Cluster 5	22	68.5	0.36
Cluster 6	18	124	0.24

Table 8.2: Summary of each cluster

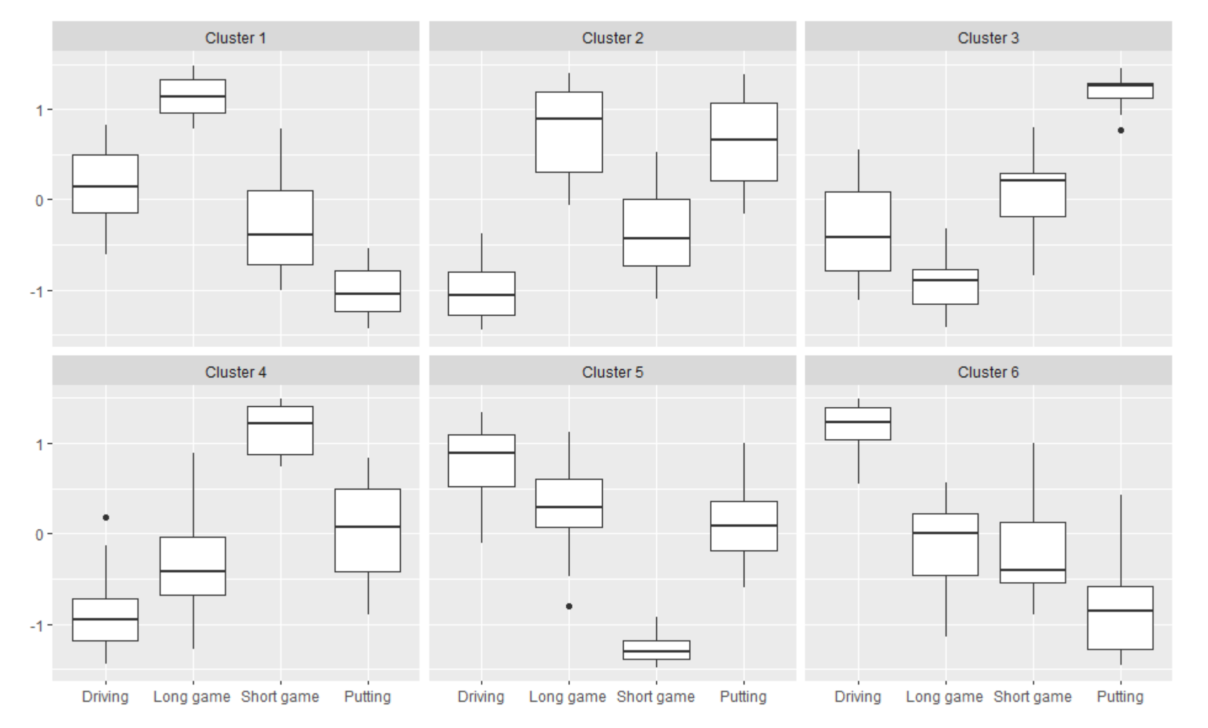


Figure 8.3: Cluster box-plots using PC variables and $K = 6$.

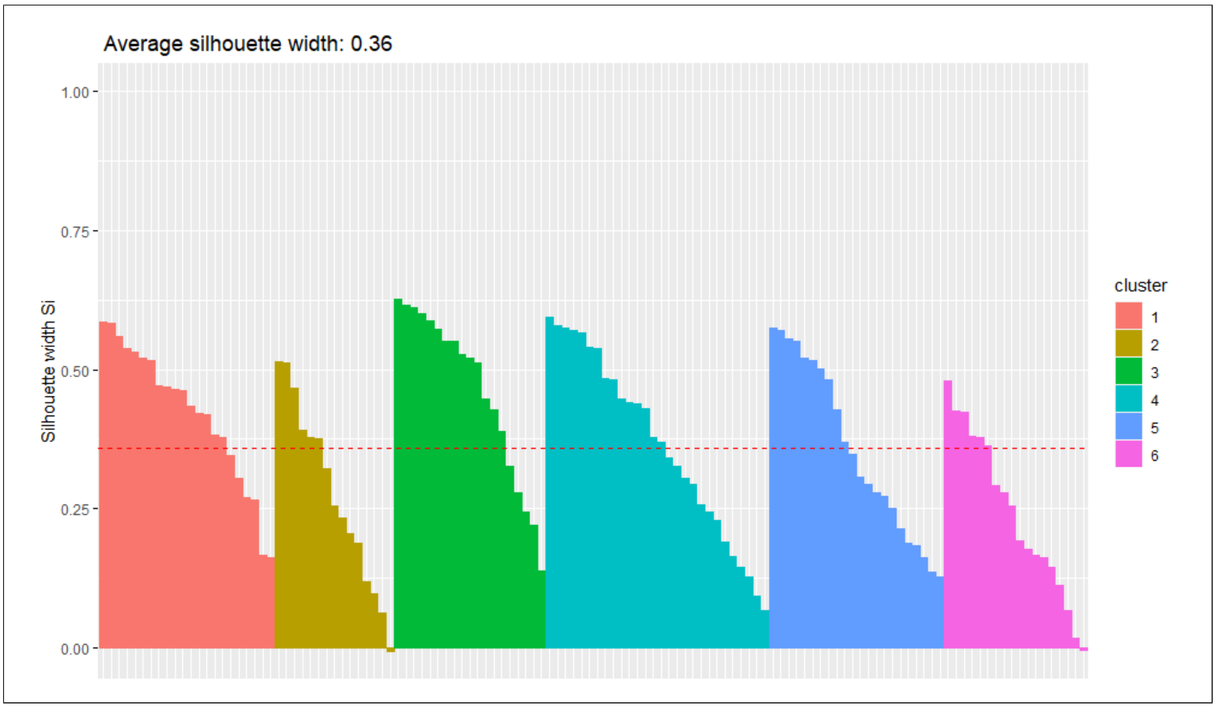


Figure 8.4: Cluster silhouette plot with $K = 6$.

8.5 Regular Expression for Course Name processing

The following R code creates a string `rexpr` by joining each of the patterns together with a vertical bar character. This is then used with `gsub` to remove all occurrences of the matching patterns. **Table 8.3** gives examples of course names before and after processing with this regular expression. Note that course names were only processed if they did not return any results as a search query on NCRDB.

```
rexpr <-
  paste('The',
        'College',
        'University',
        'Golf',
        'Club',
        'Course',
        'Country',
        'Beach',
        '.*@', # Everything before an @ symbol
        '\\(.*', # Everything after an open bracket
        '\\bat\\b', # at (whole word matching)
        '\\bof\\b', # of (whole word matching)
        '\\band\\b', # and (whole word matching)
        '\\b(G|C)\\.?C\\.?\\b', # matches GC or G.C. or CC or C.C (whole word matching)
        # Using negative lookbehind, matches a - followed by anything as long as it is not preceded by a two-letter word at the start of the line.
        '(?<!^\\w{2})-.*',
        sep='|') # Join everything together with a |

# Applies rexpr to each coursename in a vector
clean_course_name <- function(x) {
  # Input:
  # x: vector of coursenames
  # Returns:
  # A vector of shortened coursenames

  x %>%
    gsub(rexpr, '', .,
         ignore.case = TRUE,
         perl=TRUE) %>%
    # Remove extra whitespace
    gsub(' {2,}?|^ +| +$', '', .)
}
```

Listing 8.1: Regular expression course name processing.

Before	After
The University Club at Arlington	Arlington
En-Joie Golf Club	En-Joie
Watertown Golf Club - Inside - Outside Course	Watertown
Royal Oaks Country Club (Dallas)	Royal Oaks
Wildhorse Golf Club @ Robson Ranch - West/North	Wildhorse

Table 8.3: Before and after course name processing

8.6 Complete Code Listings

This section contains all the code used in this project. The directory structure is as follows:

- *utils*
  - data_vis.R
  - rating_utils.R
  - read_data.R
- cluster.Rmd
- course_ID_scrape.R
- course_rating.Rmd
- course_rating_scrape.R
- data_aggregation.R
- PCA.Rmd

```r
################################################################################
# This file contains functions used in producing data visualisations   #
################################################################################
library(pacman)
# May need to run the line below if loading ggbiplot directly does not work
# install_github("vqv/ggbiplot")
p_load(devtools, ggbiplot, fmsb, rvest, RCurl, fields, png)

# URL for retrieving player headshots
baseURL <- 'https://www.pgatour.com'

# Constant list of links to players' profile pages
# Page contains <a> elements with class "player-link" and href pointing to the player's profile page
tryCatch({
  player.links <- read_html(paste0(baseURL, "/players.html")) %>%
    html_nodes(xpath='//a[@class="player-link"]') %>%
    html_attr('href')
}, error=function(e) message('Could not connect to internet'))


# Function for producing biplots
ggbiplot.func <- function(DIM.pca) {
  ###############
  # Inputs:
  # - DIM.pca: an object created by calling run_pca in PCA.rmd
  #
  # Outputs:
  # Prints the biplot
  ###############

  g <- ggbiplot(DIM.pca$pca,
                obs.scale = 1,
                var.scale = 1,
                # groups =
```

```
                   ellipse = TRUE,
                   circle = TRUE,
                   ellipse.prob = 0.68)
  g <- g + scale_color_discrete(name = '')
  g <- g + theme(legend.direction = 'horizontal',
                 legend.position = 'top')

  # Note that arrows close to each other in this plot indicates high
      correlation.
  print(g)

}

# Search list of URLS for player name and return the player ID
# This can be appended to the baseURL to get the player's profile page URL
get_headshot_url <- function(poi) {
  ##############
  # Inputs:
  # - poi: the name of the player of interest
  #
  # Returns:
  # URL of the headshot image for the player of interest
  ##############

  player.url <- gsub('( |\\.)', '-', poi) %>%
    grep(x = player.links, ignore.case = TRUE, value=TRUE)

  # Indicates multiple matches
  # TODO: error handling here
  stopifnot(length(player.url)==1)

  # Remove any non-digit character to just leave behind the player ID
  player.id <- player.url %>%
    gsub('(\\D)', replacement = '', x = .)

  # Return the headshot url
  # TODO: This is a bit dangerous as opposed to finding the link via the
      html of the player profile page
  sprintf('https://pga-tour-res.cloudinary.com/image/upload/c_fill,d_
      headshots_default.png,f_auto,g_face:center,h_350,q_auto,w_280/
      headshots_%s.png', player.id)

}

# Produces the radar chart/player profiles
golf_chart <- function(df, poi,
                       colour = "#00AFBB", #TODO could get the dominant
                           colour of the jersey using k-means
                       vlabels=colnames(df),
                       vlcex = 0.8, # Variable label character expansion
                       caxislabels = NULL,
                       title=NULL, ...) {

  ##############
  # Inputs:
  # - df: the dataframe of pga tour player stats
  # - poi: the name of the player of interest
  # - colour: the colour of the radarchart
```

```r
  #
  # For documentation on the rest of the arguments, see ?radarchart
  #
  # Outputs:
  # The player profile radarchart including headshot if available
  ##############

  # Indexes the player in the dataframe by name
  player.indices <- which(df$`Player Name` == poi)
  stopifnot(length(player.indices)!=0) # indicates player name did not match

  # Get the axes limits
  min.row <- df %>% select(-'Player Name') %>% apply(2, min) %>% t
  max.row <- df %>% select(-'Player Name') %>% apply(2, max) %>% t

  # Remove all rows except that of the player of interest
  df %<>%  filter(`Player Name`==poi) %>% select(-`Player Name`) %>% head(1)

  rbind(max.row,
        min.row,
        df) %>% ### TODO: choose player by year too, not just name
      radarchart(# Polygon customisation
        pcol = colour,
        pfcol = scales::alpha(colour, 0.5),
        plwd = 2,
        plty = 1,
        # Grid customisation
        cglcol = 'grey',
        cglty = 1,
        cglwd = 0.8,
        # Axis customisation
        #axislabcol = 'grey',
        # Variable labels
        vlcex = vlcex, vlabels = vlabels,
        caxislabels = caxislabels, title = title, ...
      )

  # Attempt to download the headshot image of the player
  tryCatch({
    headshot.img <- get_headshot_url(poi) %>% getURLContent %>% readPNG
    # Coordinates of the headshot
    x0 <- 1
    y0 <- 0.4
    x1 <- x0 + 0.8
    y1 <- y0+0.9

    # Render the headshot
    rasterImage(headshot.img, xleft = x0, ybottom = y0, xright = x1, ytop = y1 )
  }, error= function(e) message('Could not retrieve resource from the internet'))

}

# Plotting function for producing PC1 v PC2 plots
pc.plot.func <- function(p, dim) {
```

```

  ##############
  # Inputs:
  # - p: a ggplot plot specifying the axes of the plot
  # - dim: the name of the golfing dimension the PCA was conducted on
  #
  # Returns:
  # An interactive plotly visualisation of the input ggplot
  ##############

  ggplotly(p + geom_point(aes(text=sprintf('%s', `Player Name`))),
           tooltip = c('text')) %>% # Set tooltip to only display the 'text
              ' aesthetic
    add_annotations(
      text = dim,
      x = 0,
      y = 1,
      yref = "paper",
      xref = "paper",
      xanchor = "left",
      yanchor = "top",
      yshift = 20,
      showarrow = FALSE,
      font = list(size = 15)
    )
}

polygon.plot.func <- function(p1,p2,p3) {
  x11()
  par(mar=c(3,3,1,1))
  plot(1,1,ylim=c(-1.5,3),xlim=c(-2.5,3), t="n", xlab="", ylab="")
  polygon(p1$X, p1$Y, border=2)
  polygon(p2$X, p2$Y)
  polygon(p3$X, p3$Y, col=rgb(0,0,1,0.2))
}
```

src/utils/data_vis.R

```
############################################
# This file contains functions and objects #
# used in the course  rating analysis.     #
############################################
library(pacman)

# Regular expression for improving search query
rexpr <-
  paste('The',
        'College',
        'University',
        'Golf',
        'Club',
        'Course',
        'Country',
        'Beach',
        '.*@', # Everything before an @ symbol
        '\\(.*', # Everything after an open bracket
        '\\bat\\b', # at (whole word matching)
        '\\bof\\b', # of (whole word matching)
```

```
          '\\band\\b', # and (whole word matching)
          '\\b(G|C)\\.?C\\.?\\b', # matches GC or G.C. or CC or C.C (whole
             word matching)
          # This little beauty uses negative lookbehind:
          # Will match a - followed by anything as long as it is not preceded
              by a two-letter word at the start of the line
          '(?<!^\\w{2})-.*',
          sep='|')

# Applies rexpr to each coursename in a vector
clean_course_name <- function(x) {
  ###############
  # Inputs:
  # - x: vector of coursenames
  #
  # Returns:
  # A vector of shortened coursenames
  ###############
  x %>%
    gsub(rexpr, '', .,
         ignore.case = TRUE,
         perl=TRUE) %>%
    # Remove extra whitespace
    gsub(' {2,}?|^ +| +$', '', .)
}

# Filenames of the college master data
course.filenames <- c('master - NCAA Men.csv', 'master - NCAA Women.csv')

# Function for reading the college golfer data
read.college <- function(fname, ...) {
  ###############
  # Inputs:
  # - fname: name of the college master file
  # - ...: additional args passed to fread
  #
  # Returns:
  # A cleaned dataframe of college golfer tournament stats
  ###############

  # Read file
  fread(file=paste0('../data/course_rating/', fname), ...) %>%
    # Clean up
    mutate(venue=gsub('^.*</?span> ', '', venue),
           venue=gsub('&(amp)?;?', '', venue))
}
```

src/utils/rating_utils.R

```
######################################################################
# This file contains functions and constants used for reading in   #
# the data and pre-processing it using different variables        #
######################################################################
library(pacman)
p_load(data.table, rjson)

# A column selection function that takes all the (averaged) metrics
    possible from the dataset
```

```
all_averages <- function(df) {

  ##############
  # Input:
  # - df: the dataframe of pga tour player stats
  #
  # Returns:
  #  the dataframe with all averaged metrics included
  ##############

  cat('Using all columns with %, Shots Gained, and lots of other terms\n')
  df %>%
    select(`Player Name`, Date,
      # Select only numeric columns
      where(is.numeric) &
        # That contain certain terms in their name (default for contains()
            is case insensitive)
        (contains('%') |
           (contains('SG') & contains('AVERAGE')) |
           (contains(c('Driv', 'Fairway', 'GIR', 'Sandsaves', 'Birdie', '
              Par', 'Bogey', 'Putt', 'Green', 'Approach')) & contains('AVG'
              ))
        ))
}


# This function reads the json file containing the columns to be used in
    the analysis
# and subsets the dataframe to include only these columns
benchmark_vars <- function(df, json.filepath='../data/variable_name_dict.
    json') {

  ##############
  # Inputs:
  # - df: the dataframe of pga tour player stats
  # - json.filepath: filepath to the json file of variable names
  #
  # Returns:
  #  the dataframe with all available columns from the json file included
  #
  ##############

  # Check if the json variables have already been read from file
  if (!exists('json.vars')) {
    json.vars <<- fromJSON(file=json.filepath) %>% lapply(unlist) # Double
        arrow writes to global scope
  }
  # Extract columns and only keep those that are also present in the
      dataset
  cols <- c('Player Name', unlist(json.vars, use.names = FALSE))
  cols <- subset(cols, cols %in% colnames(df))
  # Subset the dataframe with the specified columns
  df %>% select(all_of(cols))

}

# This function subsets the dataframe by a golfing dimension (or
    combination of dimensions)
```

```r
subset_by_dim <- function(df,
                          dimensions=c('General', 'Driving', 'Putting', 'Long.game', 'Short.game'),
                          sg.vars=TRUE,
                          amateur=FALSE){

  ##############
  # Inputs:
  # - df: the dataframe of pga tour player stats
  # - dimensions: the golfing dimension(s) to subset. can choose multiple
  # - sg.vars: set to FALSE if strokes gained variables should be dropped
  # - amateur: set to TRUE if reading the amateur IoG data, FALSE for the PGA TOUR data
  #
  # Returns:
  #  the dataframe containing only variables from the specified golfing dimension(s)
  #
  ##############

  dimensions <- match.arg(dimensions, several.ok = TRUE)

  # Dataframe contains amateur data and columns
  if (amateur) {
    # Changes made to the original .csv file: rename player column and GIR column, remove the duplicate 3 putt rate columns
    # Renamed SG: Approaches to SG: Approaches (from >100 yards)
    # Renamed the putting and short game columns too (they were per shot)

    # Retrieve variables from the json file
    player.cols <- c('Player Name'='Player Name', unlist(json.vars[dimensions])) %>%
                   subset(., . != '') %>% # Remove empty ones
                   names %>% # Get the keys rather than the values
                   # Remove the pre-pended category names
                   gsub('^(General|Driving|Putting|Long.game|Short.game).', '', ., ignore.case = TRUE)

    intersecting.cols <- player.cols %in% colnames(df) # Columns that are actually in the data frame
    dropped.cols <- player.cols[!intersecting.cols] # Columns that do not appear in dataframe (but should)
    # Warn about dropped columns
    if (length(dropped.cols)>0) {
      paste0(dropped.cols, collapse = '\n') %>%
        sprintf('Warning: amateur data contained %i missing columns:\n%s\n\n', length(dropped.cols), .) %>%
        message()
    }

    # Only keep columns that are inside the dataframe
    player.cols <- subset(player.cols, intersecting.cols)

    # Construct columns full of zeros to replace the ones that have been dropped
    spare.cols <- matrix(0, nrow=nrow(df),ncol=length(dropped.cols)) %>%
                    as.data.frame()
    colnames(spare.cols) <- dropped.cols
```

```

    # Subset the columns and append any missing ones
    df <- df %>% select(all_of(player.cols)) %>%
                cbind(spare.cols)


  } else { # Not using amateur data
    # Get columns from json file
    cols <- c('Player Name', unlist(json.vars[dimensions]))
    # Get only the cols that are contained in the dataframe
    cols <- subset(cols, cols %in% colnames(df))
    names(cols) <-  names(cols) %>%
                          # Remove pre-pended golfing dimension name
                          gsub('^(General|Driving|Putting|Long.game|Short.
                              game).', '', ., ignore.case = TRUE)

    # Select only the matched columns
    df <- df %>% select(all_of(cols))
  }

  if (!sg.vars) {
    df <- df %>% select(!contains('SG'))
  }

  return (df)

}
```

src/utils/read_data.R

```
---
title: "Cluster Analysis"
author: "josh atwal"
output: html_document
date: "`r format(Sys.time(), '%d %B, %Y, %H:%M')`"
knit: (function(inputFile, encoding) {
      rmarkdown::render(inputFile,
                        encoding=encoding,
                        output_file=file.path(dirname(inputFile), '..', '
                            output', 'cluster.html')) })
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE, warning = FALSE, message = FALSE)
library(pacman)
p_load(tidyverse, magrittr, ClusterR, plotly, PBSmapping, cluster,
    factoextra, gridExtra)
source('utils/data_vis.R')
set.seed(0)

# SET THIS VARIABLE TO SG TO EVALUATE CLUSTERING ON SG RATHER THAN PC
    VARIABLES
var.set <- 'PC'
cat(sprintf('Using %s variables\n', var.set))

# Shorthand variable subset function
select_vars <- function(df,
                        var.set = c('SG', 'PC'),
```
```

```
                          include.rank=TRUE) {
  ###############
  # Input:
  # - df: the dataframe of pga tour player stats
  # - var.set: subset for the SG or the PC variables
  # - include.rank: set to FALSE if rank column should be dropped
  #
  # Returns:
  # A dataframe containing the desired variables
  ###############

  var.set <- match.arg(var.set)
  if (var.set == 'SG') {
    df <- df %>%
      select('Player Name', rank, contains('SG')) %>%
      rename_with(~gsub('SG: ', '', .x)) # Rename the variables
  } else {
    df <- df %>%
      select(!contains('SG'))
  }

  if (!include.rank) {
    df <- df %>% select(-rank)
  }

  return(df)
}

# Rescales numeric values to start globally from 0, and for each player divide by the total value
rescale_player <- function(df) {

  ###############
  # Input:
  # - df: the dataframe of pga tour player stats
  #
  # Returns:
  # A dataframe where every player has been scaled such that their metrics have mean 0 and std. 1
  ###############

  # Store the non-numeric columns
  extra.cols <- df %>% select(!where(is.double))

  # Drop the non-numeric columns
  df <- df %>% select(where(is.double))

  # Normalise each row to have mean 0 and std. 1
  df %>%
      apply(1, function(x) {
        x <- x - mean(x)
        x/sqrt(var(x))

      }) %>%
    t() %>%
    cbind(extra.cols, .)

}
```

````


```


Only players with rank < 250 are kept for the clustering analysis

```{r}
# Read output of PCA.rmd
fpath <- '../data/pc_data.rds'
pc.df <- readRDS(fpath) %>% select(!contains('2'))

# Read official golf rankings
fpath <- '../data/owgr.csv'
tryCatch({
  owgr <- read.csv(fpath)
  }, error = function(e) {
     'Could not find the data file at %s. Please download the data from http://www.owgr.com/ranking.\n' %>%
       sprintf(fpath) %>% stop
     },
   finally = {
     pc.df <- owgr %>% mutate('Player Name'=sprintf('%s %s', First.Name, Last.Name),
                              rank=End.2020) %>% # Using rankings at the end of 2020
                                                 # TODO: get the rankings dynamically instead of using the previously downloaded file
                             select('Player Name', rank) %>%
                             right_join(pc.df, by='Player Name')
  })


# REMOVE PLAYERS WITH RANK >= 200
pc.df %<>% filter(rank < 250)


pc.dist <- pc.df %>% select_vars(var.set) %>%
                     rescale_player() %>%
                     select(where(is.double)) %>%
                     dist()

```

## Normalised player values
```{r}

pc.df %>% select_vars(var.set, include.rank=FALSE) %>%
  filter(`Player Name`=='Bryson DeChambeau') %>%
  rescale_player()

```

# K-Means Clustering

Note that box plots use the transformed (normalise each players metrics) data
````
````

````
```{r}
# K-Means Clustering
run_kMeans <- function(df, K, N=1, print.output=TRUE, title=NULL) {

  ##############
  # Input:
  # - df: a dataframe of players containing either the SG or PC variables for each golfing dimension
  # - K: the number of clusters to fit
  # - N: the number of replicates to perform
  # - print.output: set to FALSE to disable printing of output tables and plots
  # - title: sets the title of the generated plot
  #
  # Outputs:
  # - The top ranked players in each cluster
  # - A boxplot displaying the stat distribution of players in each cluster
  #
  # Returns:
  # A list with two elements:
  # - df: the original dataframe with a new column specifying which cluster each player was assigned to
  # - clusters: the object returned by kmeans(), can be used to extract the cluster centroids
  ##############

  # Compute clusters N number of times and aggregate the results
  cluster.matrix <- lapply(1:N, function(i){
    clustering <- df %>% select(where(is.double)) %>% kmeans(K, nstart = 200)
    clustering$cluster
  }) %>% do.call(cbind, .)

  # Choose clustering with largest average silhouette value
  best.index <- cluster.matrix  %>%
    apply(2, function(clustering) {

      # First column of s is the clustering
      # Second columns of s is the neighbour
      # Third column of s is the silhouette width
      silhouette(clustering, pc.dist)[,3] %>% mean()
    }) %>%
    which.max()

  best.cluster <- cluster.matrix[,best.index]

  df %<>% mutate(cluster=best.cluster) %>% relocate('cluster') %>% relocate('Player Name')

  # Display the top ranked players in each cluster
  if (print.output){
    for (i in 1:K){
      df %>%
        filter(cluster==i) %>%
        arrange(rank) %>%
        select('Player Name', 'cluster', 'rank') %>%
        head() %>%
````
````

````
            show()
        }

        df %>%
          select(-`Player Name`, -rank) %>%
          gather('Metric', 'Value', -cluster) %>%
          mutate(cluster=sprintf('Cluster %i', cluster),
                 Metric=gsub('\\.', ' ', Metric)) %>%
          ggplot(aes(factor(Metric,
                            level=c('Driving', 'Long game', 'Short game', 'Putting')),
                     Value)) +
          geom_boxplot() +
          facet_wrap(~cluster) +
          ggtitle(title) +
          theme(axis.title.x =  element_blank()) -> p
        print(p)
    }


    return(df)
}


```

## K=6

```{r, fig.width=10}

pc.df %>%
    select_vars(var.set) %>%
        rescale_player() %>%
        run_kMeans(6, N=5000, title=sprintf('%s Vars', var.set)) -> k6

k6 %>% group_by(cluster) %>% dplyr::summarise(median(rank))

silhouette(k6$cluster, pc.dist) %>%
           fviz_silhouette()
```

## K=5

```{r}

pc.df %>%
    select_vars(var.set) %>%
        rescale_player() %>%
        run_kMeans(5, N=5000, title=sprintf('%s Vars', var.set)) -> k5

k5 %>% group_by(cluster) %>% dplyr::summarise(median(rank))

silhouette(k5$cluster, pc.dist) %>%
           fviz_silhouette()
```

## K=4

````
````

```
```{r}

pc.df %>%
  select_vars(var.set) %>%
      rescale_player() %>%
      run_kMeans(4, N=5000, title=sprintf('%s Vars', var.set)) -> k4

k4 %>% group_by(cluster) %>% dplyr::summarise(median(rank))

silhouette(k4$cluster, pc.dist) %>%
          fviz_silhouette()
```


K=3

```{r, fig.width=10}
pc.df %>%
  select_vars(var.set) %>%
      rescale_player() %>%
      run_kMeans(3, N=5000,  title=sprintf('%s Vars', var.set)) -> k3

silhouette(k3$cluster, pc.dist) %>%
          fviz_silhouette()
```

K=2


```{r}
pc.df %>%
  select_vars(var.set) %>%
      rescale_player() %>%
      run_kMeans(2, N=5000, title=sprintf('%s Vars', var.set)) -> k2

silhouette(k2$cluster, pc.dist) %>%
          fviz_silhouette()
```


Player Profiles

```{r}

for (poi in c('Dustin Johnson', 'Cameron Champ', 'Tiger Woods', 'Rory
    McIlroy', 'Bryson DeChambeau')) {
  # Raw unscaled SG variables for comparison
  pc.df %>%
      select_vars('SG', include.rank=FALSE) %>%
      golf_chart(poi, title=sprintf('Unscaled SG vars - %s', poi))

  # Normalised SG or PC variables to adjust for player ability
  pc.df %>%
    select_vars(var.set, include.rank=FALSE) %>%
    rescale_player() %>%
```
```

````
    golf_chart(poi,
               title=sprintf('%s %s Player Profile', var.set, poi))

}

```


## Summary

```{r}
# Plot of the players and their cluster using driving and putting
k4 %>% mutate(cluster=as.factor(cluster)) %>%
            rescale_player() %>%
            highlight_key(key= ~ `Player Name`, "Player Name") %>%
            ggplot(aes(Putting, Driving, col=cluster, size=1/(rank+10))) -> p
ggplotly(p +
           geom_point(aes(text=sprintf('%s\nRank: %i', `Player Name`, rank))) +
           scale_colour_discrete()+
           labs(title='Driving vs Putting') +
           theme(panel.background = element_blank()),
                   tooltip = c('text')) %>%
  highlight(on='plotly_click', color='red', opacityDim = 0.1, selectize = TRUE)


# Plot of the players and their cluster using long game and short game
k4 %>% mutate(cluster=as.factor(cluster)) %>%
            rescale_player() %>%
            highlight_key(key= ~ `Player Name`, "Player Name") %>%
            ggplot(aes(Short.game, Long.game, col=cluster, size=1/(rank+10))) -> p
ggplotly(p +
           geom_point(aes(text=sprintf('%s\nRank: %i', `Player Name`, rank))) +
           scale_colour_discrete()+
           labs(title='Long vs Short game') +
           theme(panel.background = element_blank()),
                   tooltip = c('text')) %>%
  highlight(on='plotly_click', color='red', opacityDim = 0.1, selectize = TRUE)

```

## Pairs plot

```{r, fig.width=12, fig.height=12}
k4.data <- k4 %>% mutate(cluster=as.factor(cluster),
                         size=1/(rank+10)) %>%
            rescale_player()


plotfn <- function(axes) {
    sp <- strsplit(axes, ' ')[[1]]
    xdim <- sp[1]
    ydim <- sp[2]
````

```

    p <- ggplot(k4.data,
        aes_string(xdim, ydim,
            col='cluster',
            size='size')) +
         geom_point(show.legend = FALSE) +
          scale_colour_discrete()+
          labs(title='', x='', y='') +
          theme(panel.background = element_blank(),
                axis.ticks = element_blank(),
                axis.text = element_blank())

    return(p)
}


c('Driving Driving',
  'Long.game Driving',
  'Short.game Driving',
  'Putting Driving',
  'Driving Long.game',
  'Long.game Long.game',
  'Short.game Long.game',
  'Putting Long.game',
  'Driving Short.game',
  'Long.game Short.game',
  'Short.game Short.game',
  'Putting Short.game',
  'Driving Putting',
  'Long.game Putting',
  'Short.game Putting',
  'Putting Putting') %>%
  lapply(function(x) plotfn(x)) -> myGrobs


grid.arrange(grobs=myGrobs, nrow=4, ncol=4)




```


New Data

```{r}
# Read processed amateur data file from PCA.rmd output
fpath <- '../data/amateur_pc.RDS'

pc.amateur <- readRDS(fpath) %>%
                  select(!contains('2'))

if (var.set=='SG') {
  pc.amateur <- pc.amateur %>% select('Player Name', contains('SG')) %>%
                                          relocate('Player Name', 'SG: Driving', '
                                             SG: Putting', 'SG: Short.game') # Re
                                             order columns
```

```
} else{
  pc.amateur <- pc.amateur %>% select(!contains('SG'))
}

pc.amateur <- pc.amateur %>% rescale_player()

# Join amateur data on to pro dataset
pc.amateur.pro <- pc.df %>% full_join(pc.amateur) %>%
                              select_vars(var.set) %>%
                              rescale_player()
```
```

### Recommendation


```{r}
# Stats specifically for first player in amateur dataset
aaron <- pc.amateur %>% select(!contains('2'), -'Player Name') %>% head(1)
    %>% as.numeric()

# Used to adjust all values to be non-negative when calculating polygons
data.min <- pc.amateur.pro %>% select(where(is.double)) %>% min(na.rm=TRUE)
data.min <- data.min - 0.1 # Add 0.1 just to avoid having 0 in dataset
aaron <- aaron - data.min

###### Distance functions ######
L1_norm <- function(x) sum(abs((x-aaron)))
L2_norm <- function(x) sum((x-aaron)^2)
# This function finds players who have the same relative ordering of stats
same_order <- function(x) abs(sum(order(x)-order(aaron)))

# Find pro players with the same relative ordering of stats
order.dist <- pc.amateur.pro %>% filter(`Player Name`!='Aaron Small') %>%
                        # TODO: distance only calculated on PC variables at the
                             moment
                        select_vars(var.set) %>%
                        select(where(is.double)) %>%
                        as.matrix() %>%
                        apply(1, same_order)

# This function returns a distance metric based on the area of the two
    drawn polygons
polygon_overlap <- function(x, plot.poly=FALSE) {

  ##############
  # Input:
  # - x: a vector of length 4 containing a professional player's metrics,
      with mean 0 and std. 1
  #      they MUST be in a specific order: Driving, Putting, Short.game,
      Long.game
  # - plot.poly: set to TRUE if it is desired to plot the polygons
  #
  # Computes:
  # - intersection.prop: polygons of the visual representations of the
      player profile radar charts
  #                       are constructed for aaron and the pro player.
  #                       the proportion of the intersection that overlaps
      with aaron's polyon is then computed
```
```

```r
  #
  # Returns:
  # A distance metric based on intersections. It is the reciprocal of the sum of the proportions
  # The reciprocal is taken so that a large intersection corresponds to a small distance
  ##############

  # Begin by adjusting the input by the minimum value in the dataframe, to make every value positive
  x <- x - data.min

  # This function defines a polygon that represents the visual representation of the player profile in the radar charts
  construct_poly <- function(x, PID) {
    data.frame(PID=rep(PID,4),
               POS=1:4,
               X=c(0, -x[2], 0, x[4]),
               Y=c(x[1], 0, -x[3], 0))
  }

  # Pro-player's polygon
  p1 <- construct_poly(x, 1)
  # Aaron's scaled polygon
  p2 <- construct_poly(aaron, 2)
  # The intersection between the two polygons
  p3 <- joinPolys(p1,p2)

  # Optional polygon intersection plot
  if (plot.poly) polygon.plot.func(p1,p2,p3)

  # Compute the area of the intersection polygon / aaron's polygon
  intersection.prop <- calcArea(p3)$area / calcArea(p2)$area

  # Return the reciprocal so that a large intersection corresponds to a small computed distance metric
  # subtract 1 so that it begins at 0
  (1 / intersection.prop) - 1

}

# Compute distance from player of interest to every pro player
amateur.dist <- pc.amateur.pro %>% filter(`Player Name`!='Aaron Small') %>%
                select(where(is.double)) %>%
                as.matrix() %>%
                apply(1, polygon_overlap)


# Append distance column to dataset and sort by distance
pc.df %>%  mutate('amateur.dist'=amateur.dist, 'order.dist'=order.dist) %>%
        select('Player Name', 'rank', 'amateur.dist') %>%
        filter(rank<49) %>%
        arrange(amateur.dist) -> similar.players

# Display top most similar players
similar.players %>% head(5)
```

````
```

## Evaluating other metrics using polygon intersection
```{r}
# L1 Norm

amateur.dist.L1 <- pc.amateur.pro %>% filter(`Player Name`!='Aaron Small') %>%
                           select(where(is.double)) %>%
                           as.matrix() %>%
                           apply(1, L1_norm)

amateur.dist.L2 <- pc.amateur.pro %>% filter(`Player Name`!='Aaron Small') %>%
                           select(where(is.double)) %>%
                           as.matrix() %>%
                           apply(1, L2_norm)

amateur.dist.ord <- pc.df %>% mutate(order.dist=order.dist) %>%
                           select(where(is.double)) %>%
                           as.matrix() %>%
                           apply(1, L1_norm)



other.recs <- pc.df %>% select('Player Name', rank) %>%
                              mutate('amateur.dist.L1'=amateur.dist.L1,
                                   'amateur.dist.L2'=amateur.dist.L2,
                                   'order.dist'=order.dist,
                                   'amateur.dist'=amateur.dist) %>%
                              filter(rank<49)





other.recs %>% arrange(amateur.dist.L1) %>% head(5) #%>% pull(amateur.dist) %>% mean
other.recs %>% arrange(amateur.dist.L2) %>% head(5) #%>% pull(amateur.dist) %>% mean
other.recs %>% filter(order.dist==0) %>% arrange(rank) %>% head(5) %>% pull(amateur.dist) %>% mean

other.recs %>% filter(order.dist==0) %>% arrange(amateur.dist.L2) %>% head(5) %>% pull(amateur.dist) %>% mean

```


```{r}
# Draw player chart of Aaron Small's stats (normalised)
poi <- 'Aaron Small'
# TODO aaron only has PC variables at the moment
pc.amateur.pro %>%
  select(-rank) %>%
  rescale_player() %>%
  golf_chart(poi,
             title='Amateur Player Profile')
````

```

# Draw charts for the top 3 most similar players to Aaron
for (poi in similar.players$`Player Name`[1:5]) {
  pc.df %>%
    select_vars(var.set, include.rank = FALSE) %>%
    rescale_player() %>%
    golf_chart(poi,
               title=sprintf('%s Player Profile', poi))

}

#predClusters <- predict_KMeans(predict(pca, newGolfers)[,1:N], clusters$centers, threads = 1)

```
```

src/cluster.Rmd

```
##
This file uses the college golf master files to get a unique list of course names
and runs a headless browser within a docker container with RSelenium to
dynamically retrieve the course ratings for each of the unique courses in the file.
##

References:
https://docs.docker.com/get-started/
https://www.lambdatest.com/blog/run-selenium-tests-in-docker/
https://cran.r-project.org/web/packages/RSelenium/vignettes/basics.html
https://docs.ropensci.org/RSelenium/articles/docker.html

library(pacman)
p_load(RSelenium, seleniumPipes, data.table, tidyverse)
source('utils/rating_utils.R')

# Filepath to write output
course_ids.fpath <- '../data/course_rating/course_ids.csv'

# Get machine ip address
ip.addr <- system('ipconfig', intern=TRUE) %>%
  grep("IPv4", ., value = TRUE) %>%
  gsub('.*? ', '', .)

port <- 4445

Install docker before running this
sprintf('docker run -d -p %i:4444 -p 5901:5900 selenium/standalone-firefox-debug:2.53.0', port) %>%
  system(ignore.stdout = TRUE)

# Initiate browser object
remDr <- remoteDriver(remoteServerAddr=ip.addr[2],
```
```

```r
                       port=port,
                       browser='firefox')

Sys.sleep(1)
remDr$open(silent = TRUE)

# Function for getting the course ID of a course, given its name
getCourseID <- function(query) {

  ##############
  # Inputs:
  # - query: name of the course to be searched
  #
  # Returns:
  # A dataframe of all courses and their IDs that matched the query
  ##############

  result.courses <- character()
  count <- 0
  while (length(result.courses) == 0){
    # Direct browser to the NCRDB search page
    remDr$navigate("http://ncrdb.usga.org")

    # Find the text field for specifying the club name
    e <- remDr$findElement(using = 'xpath',
                           '//input[@name="txtClubName"]')

    # Search for the club name
    e$sendKeysToElement(list(query, key = "enter"))
    # Wait for results to be returned
    Sys.sleep(5)

    # Process results
    result.courses <- remDr$getPageSource()[[1]] %>%
      read_html() %>%
      xml_find_all('//table//a')

    if (count==0){
      # Try shortened version of query
      query <- query %>% clean_course_name(query)

      Sys.sleep(5)
    } else if (count==1) return(NULL) # Skip to next one if still no result

    count <- count + 1
  }

  # Get course names
  course.names <- result.courses %>%
    xml_find_all('text()') %>%
    as.character()

  # Get course IDs
  course.ids <- result.courses %>%
    xml_find_all('@href') %>%
    as.character() %>%
    regmatches(., regexpr('\\d+', .))

```

```
  # Return dataframe of course names with IDs that matched the query
  data.frame(course=course.names, id=course.ids)
}


# Read master files to get list of course names to be searched
course.filenames <- c('master - NCAA Men.csv', 'master - NCAA Women.csv')
courses <- lapply(course.filenames,
                  # Function reads file and extracts the vector of venues
                  function(fname) {
                    # Read file
                    read.college(fname, select=c('venue')) %>%
                    # Get unique names of golf courses
                    unique() %>%
                    as.data.frame() %>%
                    deframe() %>%
                    unique()
                  }) %>%
  do.call(c, .) %>% # bind them together
  unique()

# Progress bar
pb = txtProgressBar(min = 0,
                    max = length(courses),
                    initial = 0)

# Check if file already exists and re-use if it does
if (file.exists(course_ids.fpath)) {
  scraped.courses <- read.csv(course_ids.fpath,
                              colClasses = rep('character', 2))
} else scraped.courses <- data.frame(course=character(), id=character())

# Loop over course names and retrieve IDs
for (i in 1:length(courses)) {
  # If the course already has an ID, skip it
  loc <- which(courses[i] == scraped.courses$course)
  if (length(loc)!=0 && !is.na(scraped.courses$id[loc])) next

  # Get the dataframe of courses and IDs returned from search results
  tryCatch(df <- getCourseID(courses[i]),
           # If an error is thrown for some reason, try just restarting the browser and doing the same thing
           error = function(e) {
             message(e$message)
             remDr$close
             Sys.sleep(5)
             remDr$open(silent = TRUE)

             # Retry
             df <- getCourseID(courses[i])
           })

  # If no ID was available, just produce an empty result for the course rating
  if (is.null(df)) {
    df <- data.frame(course=courses[i], id=NA)
  }

```

```
  scraped.courses <- full_join(scraped.courses, df,
                               by=c('course', 'id'))

  setTxtProgressBar(pb,i) # update progress bar
}
close(pb)

# Save courses with IDs
write.csv(scraped.courses,
          file=course_ids.fpath,
          row.names = FALSE)
```

src/course_ID_scrape.R

```
---
title: "Course Rating Analysis"
author: "Josh Atwal"
output: html_document
date: "`r format(Sys.time(), '%d %B, %Y, %H:%M')`"
knit: (function(inputFile, encoding) {
      rmarkdown::render(inputFile,
                        encoding=encoding,
                        output_file=file.path(dirname(inputFile), '..', '
                            output', 'course_rating.html')) })

---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
library(pacman)
p_load(tidyverse, data.table)

source('utils/rating_utils.R')
```

```{r, echo=FALSE}
data.path <- '../data/course_rating/'

# Read in result of course_rating_scrape
rated.courses <-
  read.csv(paste0(data.path, 'rated_courses.csv')) %>%
  drop_na() %>%
  mutate(course.short=clean_course_name(course)) %>%
  select(-c(id, course))

# Read SG benchmark data (computed via formula)
sg_scratch_benchmark <- read.csv(paste0(data.path,"average_shots_scratch.
    csv"), fileEncoding = "UTF-8-BOM")
sg_pro_benchmark <- read.csv(paste0(data.path,"average_shots_pro.csv"),
    fileEncoding = "UTF-8-BOM")


```

```{r}
# Distribution of course ratings
rated.courses %>%
    pivot_longer(c(`Male.Rating`, `Female.Rating`),
```
```

```
                   names_to = 'Gender',
                   values_to = 'Rating') %>%
    ggplot(aes(Rating, group=Gender, fill=Gender)) +
    geom_density(alpha=0.5)


```

```{r, cache=TRUE}
# Read master files
college <- lapply(course.filenames,
                  # Function reads file and extracts the vector of venues
                  function(fname) {
                    read.college(fname) %>%
                      select(-c(event_start_date, school, school_seed,
                          player_id, tournament_name)) %>%
                      mutate(gender=ifelse(grepl('women',
                                                 fname,
                                                 ignore.case = TRUE),
                                           'F',
                                           'M'))
                  }) %>%
  do.call(rbind, .) # Join M and F dataframes together

# Calculate the total par of each course
total_par <- college %>%
  distinct(venue, round, hole, .keep_all = TRUE) %>%
  group_by(venue, round) %>%
  summarise(totalpar = sum(par))

```

Join ratings

```{r}
# Join ratings onto NCAA data
college.rated <-
  inner_join(college, total_par) %>%
  mutate(course.short=clean_course_name(venue)) %>%
  inner_join(rated.courses, by='course.short') %>%
  mutate(rating=ifelse(gender=='M', Male.Rating, Female.Rating),
         player_name = as.factor(player_name),
         course.short = as.factor(course.short),
         gender=as.factor(gender)) %>%
  select(-c(Male.Rating, Female.Rating, venue)) %>%
  # Remove rows where score to par is too large
  filter(to_par < 20)

# Average ratings where rows have been duplicated
college.rated <-
  college.rated %>%
  group_by(course.short) %>%
  summarise(rating=mean(rating)) %>%
  inner_join(select(college.rated, -rating)) %>%
  distinct()

# Clean up large objects
rm(college)
```

```
rm(courses)
gc(reset=TRUE, verbose = FALSE)

```

Dataset statistics

```{r, fig.height=3, fig.width=8}

# Number of male players
college.rated %>% filter(gender=='M') %>% select(player_name) %>% unique
    %>% nrow
# Number of male players
college.rated %>% filter(gender=='F') %>% select(player_name) %>% unique
    %>% nrow


# Calculate number of venues played by each player
venues.played <-
  college.rated %>%
  select(player_name, course.short, tournament_id) %>%
  distinct() %>%
  group_by(player_name) %>%
  summarise(n_venues = n())

# Plot venues played distribution
venues.played %>%
  pull(n_venues) %>%
  table %>%
  as.data.frame() %>%
  rename(`Number of Venues Played`='.', Count='Freq') %>%
  ggplot(aes(`Number of Venues Played`, Count)) +
  geom_bar(stat='identity')

```


Relationship between course rating and score

```{r}
# Join number of venues and filter players that havent played at more than
    1
college.rated <-
  inner_join(college.rated, venues.played) %>%
  filter(n_venues > 1) %>%
  select(-n_venues)

# Compute scores of players relative to scratch
college.rel_to_scratch <-
  college.rated %>%
  group_by(player_name, course.short, round, tournament_id) %>%
  summarise(total_score=sum(score)) %>%
  inner_join(college.rated) %>%
  mutate(rel_to_scratch = (total_score-rating)/totalpar*72)

# Plot hex plot
college.rel_to_scratch %>%
  distinct(player_name, course.short, round, tournament_id, .keep_all =
```

````
      TRUE) %>%
  #filter(total_score < 100) %>%
  ggplot(aes(rating, total_score)) +
  #geom_jitter(alpha=0.05) +
  geom_hex() +
  xlim(67.5,NA) +
  geom_smooth(col='green', method = 'gam')+
  labs(x='Course Rating', y='Total score', fill='Player\nCount') + ylim(NA
      ,155) +
  geom_abline(slope = 1, intercept = 0, linetype='dashed', col='red')


```

## Distribution of players relative to scratch with quantiles plotted
```{r, fig.height=3, fig.width=8}

college.rel_to_scratch <-
  college.rel_to_scratch %>%
  group_by(player_name) %>%
  summarise(rel_to_scratch=mean(rel_to_scratch))

# Compute quantiles
quantiles <- college.rel_to_scratch %>%
  pull(rel_to_scratch) %>%
  quantile(c(0.1,0.25,0.5, 0.75, 0.9))

quantiles

# Plot distribution
college.rel_to_scratch %>%
  ggplot(aes(rel_to_scratch)) +
  geom_density(fill='grey') +
  xlim(NA,30) +
  geom_vline(xintercept = quantiles, linetype='dotted') +
  xlab('Strokes relative to Scratch')


```


```{r, fig.height=3, fig.width=8}
# Compute adjusted to-par scores
college.rated <-
  college.rated %>%
  mutate(sg_to_par_pro = score - sg_pro_benchmark$pro_tee[yardage],
         sg_to_par_scratch = score - sg_scratch_benchmark$sc_tee[yardage],
         cr_to_par = score - (par * rating/totalpar)) %>%
  group_by(player_name, course.short) %>%
  summarise(to_par = mean(to_par),
            sg_to_par_pro = mean(sg_to_par_pro),
            sg_to_par_scratch = mean(sg_to_par_scratch),
            cr_to_par = mean(cr_to_par)) %>%
  ungroup()

# Compute standard deviations of within-player scores
college.rated.std <-
  college.rated %>%
````

````
  group_by(player_name) %>%
  summarise(`Raw Score` = sqrt(var(to_par)),
            `SG Pro` = sqrt(var(sg_to_par_pro)),
            `SG Scratch` = sqrt(var(sg_to_par_scratch)),
            `Course Rating` = sqrt(var(cr_to_par))) %>%
  replace_na(list(rep(0,4)))

mean(college.rated.std$`Raw Score`)
mean(college.rated.std$`Course Rating`)
mean(college.rated.std$`SG Scratch`)

# Kolmogorov smirnov tests
ks.test(college.rated.std$`Raw Score`, college.rated.std$`Course Rating`)
ks.test(college.rated.std$`Raw Score`, college.rated.std$`SG Scratch`)

# Plot standard deviation distributions
college.rated.std %>%
  pivot_longer(c(`Raw Score`, `SG Scratch`, `Course Rating`),
               names_to = 'Score',
               values_to = 'Standard Deviation') %>%
  ggplot(aes(`Standard Deviation`, group = `Score`, fill = `Score`)) +
  geom_density(alpha=0.5) +
  #xlim(0.4, 1.7) +
  xlab('Player Score Standard Deviation')

# Plot difference due to adjustment
college.rated.std %>%
  mutate(`SG Pro` = `SG Pro` - `Raw Score`,
         `SG Scratch` = `SG Scratch` - `Raw Score`,
         `Course Rating` = `Course Rating` - `Raw Score`) %>%
  pivot_longer(c(`SG Scratch`, `Course Rating`), #, `SG Pro`),
               names_to = 'Score',
               values_to = 'Change in Standard Deviation') %>%
  ggplot(aes(`Change in Standard Deviation`, group = `Score`, fill = `Score
    `)) +
  geom_boxplot(alpha=0.5) +
  geom_vline(xintercept = 0, linetype = 'dotted') +
  labs(x=expression(paste('Distribution of Change in Player Score Standard
    Deviation ', Delta)))

```


```{r}
# Plot mean player score distribution
college.rated %>%
  group_by(player_name) %>%
  summarise(`Raw Score` = mean(to_par),
            `SG Pro` = mean(sg_to_par_pro),
            `SG Scratch` = mean(sg_to_par_scratch),
            `Course Rating` = mean(cr_to_par)) %>%
  pivot_longer(c(`Raw Score`, `SG Scratch`, `Course Rating`),
               names_to = 'Score',
               values_to = 'Mean') %>%
  ggplot(aes(`Mean`, group = `Score`, fill = `Score`)) +
  geom_density(alpha=0.5) +
  xlim(NA, 2) +
  xlab('Player Mean Score')
````

```
```

Average score on a course
```{r}
college.rated %>%
  group_by(course.short) %>%
  summarise(`Raw Score` = mean(to_par),
            `SG Scratch` = mean(sg_to_par_scratch),
            `Course Rating` = mean(cr_to_par)) %>%
  pivot_longer(c(`Raw Score`, `SG Scratch`, `Course Rating`),
               names_to = 'Score',
               values_to = 'Mean') %>%
  ggplot(aes(`Mean`, group = `Score`, fill = `Score`)) +
  geom_density(alpha=0.5) +
  xlab('Course Mean Score')

```
```

src/course_rating.rmd

```
# ##########################################################################################
# This file uses the file containing the generated list of course ids to retrieve        #
# the table of course ratings for males and females and save them all into one dataset   #
# ##########################################################################################

library(pacman)
p_load(rvest, tidyverse)

baseURL <- 'https://ncrdb.usga.org/courseTeeInfo.aspx?CourseID='

# Filepath to read input from course_IDs.R
course_ids.fpath <- '../data/course_rating/course_ids.csv'

# Function that retrieves the course rating from the static webpage, given the course ID
getCourseRating <- function(id){

  ##############
  # Input:
  # - id: the course id
  #
  # Returns:
  # The male and female course rating
  ##############

  # Extract the table from the webpage
  t <- read_html(paste0(baseURL, id)) %>%
    html_nodes(xpath='//table[@id="gvTee"]') %>%
    html_table()

  # If no table was found, return NULL
  if (length(t) == 0) return(NULL)
```
```

```
  # Filter the table to only include the rows with highest course rating
     for men and women
  t <- t %>% as.data.frame() %>%
    select(Gender, contains('Course')) %>%
    rename('Course Rating'=2) %>%
    group_by(Gender) %>%
    filter(`Course Rating` == max(`Course Rating`)) %>%
    slice(1) %>%
    ungroup()

  # Format into output vector
  sapply(c('M', 'F'),
         function(g) filter(t, Gender==g) %>% pull('Course Rating'))

}

# Read saved course IDs from output of course_IDs.R
scraped.courses <- read.csv(course_ids.fpath,
                            colClasses = rep('character', 2))

# Init empty vectors
male <- female <- numeric(nrow(scraped.courses))

pb <- txtProgressBar(0, nrow(scraped.courses), initial=0)

# Loop over each course ID and retrieve the top male and female rating
for (i in 1:nrow(scraped.courses)) {
  if (is.na(scraped.courses$id[i])) {
    ratings <- c(NA, NA)
  } else{
    # Get ratings
    ratings <- getCourseRating(scraped.courses$id[i])
    # If there is no rating
    if (is.null(ratings)) {
      ratings <- c(NA, NA)
    }
  }

  # Append ratings to vectors
  male[i] <- ratings[1]
  female[i] <- ratings[2]
  setTxtProgressBar(pb,i)
}
close(pb)

# Create one dataframe with courses and ratings and save it
rated.courses <-
  scraped.courses %>%
  mutate('Male Rating'=as.numeric(male),
         'Female Rating'=as.numeric(female))

# Write the rated courses to csv
write.csv(rated.courses,
          file=course_ids.fpath,
          row.names = FALSE)
```

src/course_rating_scrape.R

```r
##############################################################
# This file reads in and processes the raw scraped data  #
##############################################################
library(pacman)
p_load(data.table, tidyverse, stringr, magrittr)

# In and output paths for data
in_data_path <- '../data/Raw data by season/'
out_data_path <- '../data/Processed data by season/'

files <- list.files(in_data_path) # All files in folder
first <- TRUE # TRUE for the first time the loop is entered

for (fname in files){ # Iterate over each file in the folder
  df <- paste0(in_data_path, fname) %>% # Construct file name
          fread() %>% # Read csv file
          select(-Statistic) %>% # Drop the statistic column
          mutate(Date=as.Date(Date)) %>%
          # Group by player name AND variable and keep only rows for which the variable is most recent
          group_by(`Player Name`, Variable) %>%
          filter(Date == max(Date)) %>%
          ungroup()

  # Convert the foot, inch measurements into decimals
  footinch <- str_match(df$Value, '(\\d+)\' (\\d+)"')[,2:3] # Matches the foot inches pattern of measurement
  footinch <- as.numeric(footinch[,1]) + as.numeric(footinch[,2])/12

  df %<>%
    # String formatting
    mutate(Value=str_replace_all(Value, c(# Remove commas, $, +, unnecessary double quote
                                          '[,\\+$]|"""?' = "",
                                          '^$' = NA, # Replace empty strings with NA
                                          "'" = '' # Remove apostrophe
                                        ))) %>%
    # Replace the foot inch measurements with the decimal values
    mutate(Value=ifelse(is.na(footinch), Value, footinch)) %>%
    # Spread the variable column over multiple columns to make a wide dataframe
    pivot_wider(names_from = 'Variable', values_from = 'Value')

  # For each player, take the row with the least number of missing values
  df %<>%
    mutate(n_missing = apply(df, 1, function(x) sum(is.na(x)))) %>%
    group_by(`Player Name`) %>%
    filter(n_missing == min(n_missing)) %>%
    ungroup() %>%
    select(-n_missing)


  # Join processed files into one big file containing all data
  if (first) {
    all_data <- df
    first <- FALSE
```

```
  } else all_data <- full_join(all_data, df)

  # Write to file
  df %>%
    write.csv(paste0(out_data_path, fname), row.names = FALSE)

  gc(reset = TRUE) # garbage collection
}

# Write all data to file
all_data %>%
  write.csv(paste0(out_data_path, 'all_data.csv'), row.names = FALSE)
```

src/data_aggregation.R

```
---
title: "Principal Component Analysis"
author: "Josh Atwal"
output: html_document
date: "`r format(Sys.time(), '%d %B, %Y, %H:%M')`"
knit: (function(inputFile, encoding) {
      rmarkdown::render(inputFile,
                        encoding=encoding,
                        output_file=file.path(dirname(inputFile), '..', '
                          output', 'PCA.html')) })

---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
library(pacman)
p_load(tidyverse, magrittr)

source('utils/read_data.R')
source('utils/data_vis.R')
# Rescales columns to have mean 0 std 1
normalise.col <- function(x) {
                          x <- x - mean(x)
                          x / sqrt(var(x))
                        }

```

```{r}
include.sg.vars = FALSE # SET THIS TO FALSE TO DROP THE SG VARIABLES FROM
   THE PCA
```

Data input

```{r}
# Read a file from the ../data/Processed data by season/ folder, using the
# "benchmark_vars" column selection function and the variable name
    dictionary specified.
# See read_data.R for more details

in_data_path <- '../data/Processed data by season/'
fname <- '2019_data.csv'
```
```

~~~~
json.filepath <- '../data/variable_name_dict.json'

# Read file and pass it to function for pre_processing
pga.df <- fread(paste0(in_data_path, fname)) %>%
              # Keep only players with less than 50% missing values (ie the pros)
              filter(rowSums(is.na(across(!c(`Player Name`, Date))))/ncol(.) < 0.5) %>%
              # Select columns of interest. Can also use all_averages() if desired
              benchmark_vars(json.filepath) %>%
              # And that contain a maximum of 20% missing values
              select(where(function(x) mean(is.na(x)) < 0.2)) %>%
              # Drop cases with any missing values still remaining
              drop_na()

dim(pga.df)
```

We have `r ncol(pga.df)-1` variables recorded for `r nrow(pga.df)` golfers.

# R^2 analysis

```{r}
dim.names <- c('Driving', 'Long.game', 'Short.game', 'Putting')


golf.fits <- lapply(dim.names, function(dim) {
    df <- pga.df %>%
          subset_by_dim(dim) %>% # Subset by golfing dimension
          rename_with(~gsub('SG:.*', 'SG', .x)) # Rename column

    # Fit linear model to try predict strokes gained variable
    lm(SG ~ . , data = select(df, -`Player Name`))

})

golf.fits %>% sapply(function(golf.fit)summary(golf.fit)$r.squared) %>% `names<-`(dim.names)


```

# Principle Component Analysis

```{r echo=FALSE}
run_pca <- function(df,
                    golf.dim = c('General', 'Driving', 'Putting', 'Long.game', 'Short.game'),
                    pre.scaled=FALSE,
                    print.output=FALSE) {

  ##############
  # Input:
  # - df: the dataframe of pga tour player stats
  # - golf.dim: one or many of the golfing dimensions
  # - pre.scaled: has the data been pre-scaled (no need for prcomp to center and scale the input)
~~~~

```
# - print.output: whether to print diagnostic output
#
# Returns:
# A list with the following elements:
# - pca: the raw object returned by calling prcomp.
# - N: the number of significant Principal Components
# - pca.summary: a summary of the variation explained by each of the PCs
# - var.mat: a matrix with the variable loadings for each of the PCs
# - pc1.vars: the variables most strongly associated with PC1
# - pc2.vars: the variables most strongly associated with PC2
##############

 golf.dim <- match.arg(golf.dim, several.ok = TRUE)

### Run PCA
 df %>% select(where(is.numeric)) %>%
        # Prcomp preforms the PCA. centering and scaling is done unless
           manual scaling has been specified
        prcomp(center = TRUE,
               scale = !pre.scaled) -> pca

### Calculate number of useful components
N = which.max(pca$sdev <= 1) - 1
# Always at least one PC
 if (N==0) N <- 1

### Computing variable summary
comp.std <- pca$sd[1:N]

# Proportion of total variance accounted for by each component
 var.prop <- (comp.std^2)/sum(pca$sd^2)

# Cumulative proportion of variance explained by each component
cum.prop <- cumsum(var.prop)

# Summary of the variation explained
 pca.summary <- rbind(comp.std, var.prop, cum.prop)
 row.names(pca.summary) <- c('Standard deviation', 'Proportion of Variance
    ', 'Cumulative Proportion')
 colnames(pca.summary) <- sprintf('PC%i', 1:N)


### Variable importance

# Variable loading stored here
 var.mat <- pca$rotation[,1:N]


 if (N==1){
   # Variables contributing the most to PC1
   data.frame(varName=names(var.mat),
             PC1=var.mat) %>%
              arrange(desc(abs(PC1))) -> pc1.vars

 } else {
   # Variables contributing the most to PC1
   var.mat %>%
     as.data.frame() %>%
```

```
      mutate(varName = row.names(var.mat)) %>%
      select(varName, PC1, PC2) %>%
      arrange(desc(abs(PC1))) -> pc1.vars

    # Variables contributing the most to PC2
    var.mat %>%
      as.data.frame() %>%
      mutate(varName = row.names(var.mat)) %>%
      select(varName, PC1, PC2) %>%
      arrange(desc(abs(PC2))) -> pc2.vars

  }

  if (print.output) {
    # % of variation explained by PC1
    #cat(sprintf('PC1 of the %s variables explains %.2f%% of the variation\
       n', golf.dim, 100*pca.summary[2,1]))
    ### Screeplot
    plot(pca$sd^2, xlab='component', ylab='variance', main=golf.dim)
    abline(h=1)
    # Variables contributing most to PC1
    cat('Top variables contributing to PC1\n')
    if (N>1){
      pc1.vars %>% as_tibble() %>% head(5) %>% print()
    }
  }

  if (N>1){
    list(pca=pca,
        N=N,
        pca.summary=pca.summary,
        var.mat=var.mat,
        pc1.vars=pc1.vars,
        pc2.vars=pc2.vars)
  } else {
    list(pca=pca,
        N=N,
        pca.summary=pca.summary,
        var.mat=var.mat)
  }
}


```

Note that according to the 1-SD rule, we should only consider the first N principle components which have a standard deviation > 1 for a meaningful dimensionality reduction. PCs with an SD less than 1 explain less than a single explanatory variable would.

Putting

```{r}
# Note that documentation on the output of the run_pca function is given at
    the bottom of the function definition
golf.dim <- 'Putting'
pga.df %>% subset_by_dim(golf.dim, sg.vars = include.sg.vars) %>%
            run_pca(golf.dim, print.output = TRUE) -> putt.pca
```

```
#ggbiplot.func(putt.pca)
```

Driving

```{r}
golf.dim <- 'Driving'
pga.df %>% subset_by_dim(golf.dim, sg.vars = include.sg.vars) %>%
              run_pca(golf.dim, print.output = TRUE) -> drive.pca

ggbiplot.func(drive.pca)

```

```{r, eval=FALSE, echo=FALSE}
# Combining Driving and Long game into a single dimension
golf.dim <- c('Driving', 'Long.game')
pga.df %>% subset_by_dim(golf.dim, sg.vars = include.sg.vars) %>%
              run_pca(golf.dim, print.output = TRUE) -> drive.long.pca

# Top variables when combining long game and driving
drive.long.pca$var.mat %>%
  as.data.frame() %>%
  mutate(varName=row.names(drive.long.pca$var.mat)) %>%
  relocate(varName) %>%
  arrange(desc(abs(PC1))) #%>%select(-varName)

```

Long-game

```{r}
golf.dim <- 'Long.game'
pga.df %>% subset_by_dim(golf.dim, sg.vars = include.sg.vars) %>%
              run_pca(golf.dim, print.output = TRUE) -> long.pca

ggbiplot.func(long.pca)
```

Short-game

```{r}
golf.dim <- 'Short.game'
pga.df %>% subset_by_dim(golf.dim, sg.vars = include.sg.vars) %>%
              run_pca(golf.dim, print.output = TRUE) ->  short.pca

ggbiplot.func(short.pca)
```

Correlation Analysis

```{r echo=FALSE}
# Extract the PC1 columns from each of the output objects
pc.cols <- list(drive.pca, putt.pca, short.pca, long.pca) %>%
                lapply(function(pc) pc$pca$x[,1:2]) %>%
                as.data.frame() %>%
                `colnames<-`(paste0(rep(c('Driving', 'Putting', 'Short.game',
```

```
                        'Long.game'),
                                    each=2),
                               c('','2'))) %>%
                   # Negate driving because the PCA sign is arbitrary
                   mutate(Driving=-Driving)

# Normalise each PC variable to have mean 0 std. 1
pc.cols.scaled <- pc.cols %>% apply(2, normalise.col) %>% as.data.frame()

# Combine the PC columns with the shots gained columns
pc.df <- pga.df %>%
                select(`Player Name`, contains('SG') & contains(c('Putt', 'Off-the-tee', 'Approach', 'Around the Green'))) %>%
                rename_with(~gsub(' - \\(AVERAGE)', '', .x)) %>% # Rename the variables
                dplyr::rename('SG: Driving'='SG: Off-the-Tee',
                              'SG: Short.game'='SG: Around the Green',
                              'SG: Long.game'='SG: Approach the Green'
                              ) %>% # Rename the variables
                # Just reordering columns
                select('Player Name', 'SG: Driving', 'SG: Putting', 'SG: Short.game', 'SG: Long.game') %>%
                cbind(select(pc.cols.scaled, !contains('2')))

# Save data to file
saveRDS(pc.df, '../data/pc_data.rds')

```
```

```{r}
sg.cols.names <- rep(c('SG: Off-the-Tee - (AVERAGE)',
       'SG: Putting - (AVERAGE)',
       'SG: Around the Green - (AVERAGE)',
       'SG: Approach the Green - (AVERAGE)'
       ),
     each=2)

p.correlations <- sapply(1:8, function(i){
  cor.test(pull(select(pga.df, sg.cols.names[i])),
              pc.cols.scaled[,i], method='pearson')$estimate
}) %>% `names<-`(gsub('\\.', ' ', colnames(pc.cols.scaled)))

level.order <- c('Driving', 'Long game', 'Short game', 'Putting')

p.correlations %>%
  as.data.frame() %>%
  mutate(vname = gsub('2', '', names(p.correlations)),
         PC=sprintf('PC%i', rep(1:2, 4))) %>%
  pivot_longer(cols = '.', values_to = 'Pearson Correlation') %>%
  mutate(label=round(ifelse(PC=='PC1', `Pearson Correlation`, NA), 2)) %>%
  select(-name) %>%
  ggplot(aes(factor(vname, level=level.order), `Pearson Correlation`, fill=PC, label=label)) +
  geom_col() +
  geom_label() +
  theme(axis.title.x = element_blank(),
        axis.text.x = element_text(size=12),
```
```

```
        axis.title.y = element_text(size=14))





```

Visualisation

Radar Charts

```{r}

for (poi in c('Tiger Woods', 'Cameron Champ', 'Rory McIlroy', 'Dustin Johnson', 'Brooks Koepka', 'Bryson DeChambeau')){
  # Player Profile charts with the SG variables
  pc.df %>%
    select(`Player Name`, contains('SG:')) %>%
    rename_with(~gsub('SG: ', '', .x)) %>%
    golf_chart(poi, title=sprintf('%s - SG', poi))

  # Player Profile charts with the PC variables
  pc.df %>%
    select(`Player Name`, !contains('SG:')) %>%
    golf_chart(poi, title=sprintf('%s - PC', poi))

}

```


PC Plots

Note the axes of these plots have been normalised to have mean 0 and std. 1

```{r, echo=FALSE}
p_load(plotly)

# Big summary PCA using all variables across all categories
golf.dims <- c('General', 'Long.game', 'Driving', 'Putting', 'Short.game')
lapply(golf.dims,
       function(dim) {
          df <- subset_by_dim(pga.df, dim)
          # This scales to have mean 0 and std dev. 1, before adding a constant such that the matrix adds up to (an arbitrary) 10
          df %>% mutate_if(is.numeric, function(x) {
            # Normalises each variable to have mean 0 and std. 1, before multiplying by a constant
            # that accounts for the different number of variables in each dimension
            x <- x - mean(x)
            x <- x / sqrt(var(x))
            x * (ncol(pga.df))/(ncol(df)-1)
          })
        }) %>%
  # Join all the dataframes together
  Reduce(function(x, y) full_join(x, y, by='Player Name'), .) %>%
```

```r
  run_pca(golf.dims, pre.scaled=TRUE, print.output = FALSE) -> all.pca

# Extract the PC1 and PC2 of the overall summary PCA
all.pca.df <- all.pca$pca$x[,1:2] %>%
                apply(2, normalise.col) %>%
                as.data.frame() %>%
                mutate_at(c('PC1', 'PC2'), as.numeric)


# Add columns for PC2
pca.df.vis <- pc.df %>% select('Player Name', !contains('SG')) %>%
                        cbind(select(pc.cols, contains('2')), all.pca.df)

# Plotly dataset
shared <- pca.df.vis %>%
              highlight_key(key= ~ `Player Name`, "Player Name")

```


```{r, echo=FALSE}
# PC1 vs PC2 plots. Note the Axes have been normalised to have mean 0 and std. 1

g1 <- ggplot(shared, aes(x=Putting2,
                          y=Putting,
                          group=`Player Name`)) %>% pc.plot.func('Putting')

g2 <- ggplot(shared, aes(x=Driving2,
                          y=Driving,
                          group=`Player Name`)) %>% pc.plot.func('Driving')

g3 <- ggplot(shared, aes(x=Short.game2,
                          y=Short.game,
                          group=`Player Name`)) %>% pc.plot.func('Short.game')

g4 <- ggplot(shared, aes(x=Long.game2,
                          y=Long.game,
                          group=`Player Name`)) %>% pc.plot.func('Long.game')


# Join individual plots side by side
subplot(g1, g3, g2, g4) %>%
          highlight(on='plotly_click', color='red', opacityDim = 0.1, selectize = TRUE)

# Construct the overall summary plot
ggplotly(ggplot(shared,
                aes(x=PC2,
                    y=PC1,
                    group=`Player Name`)) + labs(title='Overall Summary') +
            geom_point(aes(text=sprintf('%s', `Player Name`))), tooltip=c('text'))  %>%
          highlight(on='plotly_click', color='red', opacityDim = 0.1, selectize = TRUE)

```

~~~~
```

## New Data

Missing variables were set to columns of zeros

```{r, message=TRUE, echo=FALSE}
# Read amateur data
amateurs <- read.csv('../data/amateur_data.csv', check.names = FALSE)


# Replace missing values with 0
if (sum(is.na(amateurs)) > 0) {
  message(sprintf('%i missing values being replaced with 0...', sum(is.na(amateurs))))
  amateurs <- amateurs %>% replace(is.na(.), 0)
}

pca.list <- list(drive.pca, putt.pca, short.pca, long.pca, all.pca)
dim.names <- c('Driving', 'Putting', 'Short.game', 'Long.game')

# Perform predictions of PC1 and PC2 for the amateur data
pc.amateur <- lapply(1:4, function(i) {
                     pc.vars <- amateurs %>%
                                subset_by_dim(dim.names[i], amateur = TRUE) %>%
                                select(where(is.numeric)) %>%
                                predict(pca.list[[i]]$pca, .)  # Get predictions
                     pc.vars[,1:2] %>% t()

                   }) %>%
                     do.call(cbind, .) %>% # Join columns together
                     as.data.frame() %>%
                     `colnames<-`(c(rbind(dim.names, paste0(dim.names, '2')))) %>%
                     # Negate driving column because it was negated for professionals too
                     mutate(Driving=-Driving)


# Normalise the PC columns according to the same means and standard deviation from the pro dataset
# Note: dont have to subtract mean because already comes with mean 0, so just dividing by variance
pc.amateur <- (pc.amateur / sqrt(apply(pc.cols, 2, var))) %>%
                   cbind('Player Name'=amateurs$`Player Name`, .)  # Add player name column back



# Save amateur PC predictions to file along with the SG variables (renaming first)
amateurs %>% dplyr::rename('SG: Driving'='SG: Drives',
                           'SG: Long.game'='SG: Approaches (from >100 yards)',
                           'SG: Short.game'='SG: Short Game (from <100 yards)') %>%
                   select(contains('SG:')) %>%
~~~~

````
                 cbind(pc.amateur, .) %>%
                 saveRDS(file='../data/amateur_pc.RDS')

```

```{r, echo=FALSE}

# Generate the PC1 vs PC2 plots including the amateur data.
# Still normalising the axes to have mean 0 and std. 1

pc.amateur[1,1] <- 'Amateur 1' # Anonymise name before plotting
shared2 <- pca.df.vis %>%
          select('Player Name', !contains('SG') & !contains('PC')) %>%
          rbind(pc.amateur) %>%
          highlight_key(key= ~ `Player Name`, "Player Name")


g1 <- ggplot(shared2, aes(x=Putting2,
                          y=Putting,
                          group=`Player Name`)) %>% pc.plot.func('Putting')


g2 <- ggplot(shared2, aes(x=Driving2,
                          y=Driving,
                          group=`Player Name`)) %>% pc.plot.func('Driving')

g3 <- ggplot(shared2, aes(x=Short.game2,
                          y=Short.game,
                          group=`Player Name`)) %>% pc.plot.func('Short.game')

g4 <- ggplot(shared2, aes(x=Long.game2,
                          y=Long.game,
                          group=`Player Name`)) %>% pc.plot.func('Long.game')


# Individual plots
subplot(g1, g3, g2, g4) %>%
          highlight(on='plotly_click', color='red', opacityDim = 0.1, selectize = TRUE)


```

## Explaining the PCA
```{r, echo=FALSE}
pc.list <- list(drive.pca, putt.pca, short.pca, long.pca, all.pca)
dim.names <- c('Driving', 'Putting', 'Short.game', 'Long.game', 'All Variables')

i<-4
pc.list[[i]]$pc1.vars %>%
            pivot_longer(-varName, names_to = 'PC Dimension', values_to = 'Loading coefficient') %>%
            ggplot(aes(y=reorder(varName, `Loading coefficient`),
                       x=`Loading coefficient`,
                       group=`PC Dimension`,
````

```
                              col=`PC Dimension`,
                              fill=`PC Dimension`)) -> g

     ggplotly(g +
                geom_col(aes(text=sprintf('%s', varName)),
                            position = position_dodge(0.5),
                            width=.75) +
                labs(y='', title=gsub('\\.', ' ', dim.names[i])),
          tooltip = c('text', 'Loading coefficient'))




```


Plot of with and without SG variables
```{r, echo=FALSE}
pc.list <- list(drive.pca, putt.pca, short.pca, long.pca, all.pca)
dim.names <- c('Driving', 'Putting', 'Short.game', 'Long.game')


level.order <- c('Driving', 'Long game', 'Short game', 'Putting')
# Get % of variance explained by PC1 for every dimension
pct.var.1 <- sapply(1:4, function(i){
                   pc.list[[i]]$pca.summary[3,1]*100
                 }) %>% data.frame('Dimension'=gsub('\\.', ' ', dim.names),
                                   '% of Variance Explained'=.,
                                   'SG Included'=include.sg.vars,
                                   check.names = FALSE)

# Get % of variance explained by PC1 for every dimension but with/without SG variables depending on option is set
pct.var.2 <- sapply(1:4, function(i){

  pca <- pga.df %>% subset_by_dim(dim.names[i], sg.vars = !include.sg.vars) %>%
              run_pca(dim.names[i], print.output = FALSE)
  pca$pca.summary[3,1]*100


}) %>% data.frame('Dimension'=gsub('\\.', ' ', dim.names),
                '% of Variance Explained'=.,
                'SG Included'=!include.sg.vars,
                check.names = FALSE
                )


rbind(pct.var.1, pct.var.2) %>%
  ggplot(aes(factor(Dimension, level=level.order), `% of Variance Explained`, fill=`SG Included`)) +
  geom_col(position = 'dodge') +
  theme(axis.text.x = element_text(size=12),
        axis.title.x = element_blank(),
        axis.title.y = element_text(size=14))

```

```
```
```

src/PCA.Rmd
```