Statistical Properties of Performance Indicators

Measuring football metrics

Author : Ashot Akopov

Master's Program in Econometrics and Statistics

Supervisor : Pierre Ribereau

Date : May 2025

Contents

1 Introduction

> *"People who run ball clubs, they think in terms of buying players. Your goal shouldn't be to buy players, your goal should be to buy wins.And in order to buy wins, you need to buy runs."*
>
> *– Moneyball (2011)*

The global football economy reached new heights in 2023–2024, with record revenues for the most powerful clubs. According to the Deloitte Football Money League report, the 20 richest clubs generated a total of €11.2 billion, representing a 6 percent increase compared to the previous season. Data is increasingly being used not only for economic development but also, and especially, for analyzing the performance of teams and athletes.

The use of data to measure performance was largely popularized in the United States, particularly in basketball, through sabermetrics. This approach, which inspired the movie Moneyball, was introduced by Bill James but was considered too theoretical by his contemporaries. Today, the biggest clubs have entire teams of data scientists who use the latest statistical methods to uncover insights.

As a result, we now have more and more metrics providing increasingly precise information on individual performances, and these metrics are becoming ever more sophisticated. Today, we will not focus on developing new metrics, but rather on comparing existing ones. We will use meta-metrics to study the statistics of the different metrics.

This work is largely based on the article Meta-Analytics: Tools for Understanding the Statistical Properties of Sports Metrics by Franks et al. (2016). The aim was to build on the authors' rigorous work, to understand it thoroughly, and to gain a practical command of it. At the end of this document, we will outline the next steps to be taken.

I would like to sincerely thank my supervisor, Pierre Ribereau, for his availability and for his detailed statistics courses, which were immensely helpful in the successful completion of this project.

2 Data description

The data was collected from the website fbref.com. Two methods were used for data extraction:

- Scraping via the worldfootballR package, used as long as access was not restricted.
- The Table Capture extension for Chrome, developed by georgemike.com, was used when the IP address was detected as a bot by the website.

We will therefore use two datasets.

- **allSeasonsData**: contains player and season-level statistics for the five major leagues: English, German, Spanish, French, and Italian. This data covers the 2017/2018 to 2023/2024 seasons.
- **gameLogs**: contains day-by-day and match-by-match data for the five major leagues for the 2023/2024 season.

All statistics have been normalized to 90 minutes of play, as follows: Statistic per 90 minutes = $\left(\frac{\text{Raw value}}{\text{Minutes played}}\right) \times 90$

You will find in the appendix the correspondence table and explanations of the various metric abbreviations.

3 Methodology

3.1 Sources of variation

The idea of this work is to study variance. We will distinguish three sources of variation:

- The player's intrinsic skill
- The context (for example, the influence of teammates)
- Randomness (i.e., sampling variability)

We consider that each player's metric is composed of a combination of these sources of variation. To distinguish these three sources of variation, we will use three criteria:

1. **Stability**: does the metric measure the same thing over time? This meta-metric quantifies the usefulness of a metric for distinguishing players within the same season.
2. **Discrimination**: does the metric allow players to be differentiated? This meta-metric measures the extent to which a metric varies from one season to another due to changes in context and player skill, after removing the effects of randomness.
3. **Independence**: does the metric provide new information? This meta-metric quantifies the extent to which the information contained in a metric is already captured by a set of other metrics.

Let us now see how to calculate the metrics.

3.1.1 Variance decomposition

1. Intra-season variance, between players

$$\sigma^2_{PM} + \sigma^2_{SPM} + \tau^2_M$$

This is the variation observed between different players during the same season.

- σ^2_{PM}: constant differences between players.
- σ^2_{SPM}: player-season interactions (form variations from one year to another).
- τ^2_M: sampling error (randomness).

These components explain why two players may have different values for the same metric, even during the same season.

2. Intra-player variance, between seasons

$$\sigma^2_{SM} + \sigma^2_{SPM} + \tau^2_M$$

This is the variation in the performance of the same player over different seasons.

- σ^2_{SM}: overall changes in the league between seasons.
- σ^2_{SPM}: player-specific variations depending on the season.
- τ^2_M: statistical noise due to sampling.

3. Total variance (all seasons and all players)

$$\sigma^2_{SM} + \sigma^2_{PM} + \sigma^2_{SPM} + \tau^2_M$$

This is the variance observed when considering all players across all seasons. All sources of variation are summed.

3.2 Mixed model

To model this variance analysis and the calculation of the meta-metrics, we will define a mixed-effects model.

3.2.1 Definition

We will work with three-dimensional vectors: $X_{spm} \in \mathbb{R}^3$, where:

- s represents the season,
- p represents the player,
- m represents the metric.

Thus, X_{spm} is the value of metric m for player p during season s.
The mixed-effects model is therefore defined as:

$$X_{spm} = \mu_m + Z_{sm} + Z_{pm} + Z_{spm} + \varepsilon_{spm}$$

where:

- μ_m: the overall mean of metric m across all seasons and players, it does not vary in the model, thus its variance is zero.
- $Z_{sm} \sim \mathcal{N}(0, \sigma^2_{SM})$: season-specific variation for metric m,
- $Z_{pm} \sim \mathcal{N}(0, \sigma^2_{PM})$: player-specific variation for metric m,
- $Z_{spm} \sim \mathcal{N}(0, \sigma^2_{SPM})$: player-season specific variation for metric m,
- $\varepsilon_{spm} \sim \mathcal{N}(0, \tau^2_M)$: sampling error associated with the season and the player. This represents statistical noise.

Note: For an infinitely long season, we would have $\tau^2_m \to 0$, thus $\varepsilon_{spm} = 0$. Indeed, if a season lasted forever, there would be so much data that the effects of randomness would cancel out. The observed average performance would be very precise, with little or no noise, meaning no more variation due to randomness $\Rightarrow \tau^2_m \approx 0$, and $\varepsilon_{spm} = 0$.

This is a **mixed-effects model**, composed of one fixed variable (μ) and three random variables (Z_{sm}, Z_{pm}, Z_{spm}). Finally, ε_{spm} represents the statistical error.

Note: We said there are three sources of variation, however, a fourth component appears in the model: Z_{sm}, with:

$$Z_{sm} \sim \mathcal{N}(0, \sigma^2_{SM})$$

- σ^2_{SM} measures the variation of league averages between seasons.
- Example: if the average points per match were 20 in 2020 and 24 in 2021, there is inter-season variation.
- This variation is generally small:
  - It is included in the model for reasons of theoretical completeness,
  - But it is not analyzed in detail in the results.

3.2.2 Limits of the mixed model

The linear mixed-effects model presented earlier can be useful to illustrate the concept of *meta-metrics*, as it is simple and well-known. However, it is not suitable for all sports metrics, because these metrics exhibit very different statistical characteristics.

For example:

- Percentages (such as shooting success rates) are bounded between 0 and 1.
- Per-game or per-season statistics (such as points or rebounds) are often strictly positive integers.
- Other advanced metrics (such as plus-minus or VORP in basketball) are real-valued and unbounded, and can be positive or negative.

These differences mean that the additive linear model (as used in mixed models) is not suitable for all metrics, as it often assumes a normal distribution of the data, which is unrealistic in several cases.

3.2.3 Solution

To overcome these limitations, we adopt a more flexible and general approach. We consider each observed value X_{spm} (metric m for player p during season s) as a random variable.

We then introduce a simplified notation to handle conditional expectations and variances:

- $\mathbb{E}_{spm}[X] = \mathbb{E}[X \mid S = s,\ P = p,\ M = m]$: expectation of X, conditioned on season s, player p, and metric m.
- $\mathrm{Var}_{spm}[X] = \mathrm{Var}[X \mid S = s,\ P = p,\ M = m]$: associated conditional variance.

For example, $E_{sm}[V_{spm}[X]]$ refers to the average of the variance across all players for a given season s and metric m.

This approach allows:

- adaptation to any data distribution (even non-normal),
- analysis of the variation of metrics from one player to another or from one season to another,
- construction of *meta-metrics* (stability, discrimination, independence) in a robust and generalizable way, without relying on an overly simplistic model.

3.2.4 Conclusion

Thus, we will not use the model itself directly; however, it allowed us to model the problem and to facilitate a decomposition of variance, which will help us in defining the meta-metrics.

3.3 Discrimination

We will now formally define discrimination. The discrimination meta-metric measures the ability of a metric (e.g., shooting percentage, number of rebounds...) to differentiate players from one another during a season. It answers the question:
What proportion of the observed variability between players is due to true differences in performance, and not to sampling randomness? To measure the discriminative power of a metric, we need to separate the variance due to randomness from the variance due to the player.

3.3.1 Total variance

The law of total variance applied to our context is written as:

$$V_{sm}[X] = \mathbb{E}_{spm}[V_{spm}[X]] + V_{sm}[\mathbb{E}_{spm}[X]]. \tag{1}$$

This decomposition allows us to interpret the variance observed between players in a given season ($V_{sm}[X]$) as the sum of two components:

- $\mathbb{E}_{spm}[V_{spm}[X]]$ represents the average variance observed for a given player, due to sampling effects or statistical noise. In other words, this quantity measures the uncertainty related to the fact that each player has only a finite number of observations (e.g., a limited number of shots or actions).
- $V_{sm}[\mathbb{E}_{spm}[X]]$ corresponds to the variance of the observed means between players. It captures the true differences in performance between players during the season—in other words, the signal.

This formulation makes explicit the share of variance that is attributable to noise (statistical randomness) versus the share that reflects actual performance differences between players.

3.3.2 Definition of discrimination

Now that we understand how the variance is distributed, we can define a meta-metric that measures what we truly care about: What proportion of the total variance between players is actually signal?
To do this, we calculate the ratio between the two components of the previous formula.

$$\text{Discrimination} = 1 - \frac{\text{variation due to randomness}}{\text{total variation between players}}$$

A value close to 1 indicates a strong ability to discriminate between players.
Thus, for a given season s and metric m, the **discrimination** is defined as:

$$D_{sm} = 1 - \frac{E_{sm}\left[\text{Var}_{spm}(X)\right]}{\text{Var}_{sm}(X)}$$

- X_{spm}: value of metric m for player p in season s
- $\text{Var}_{spm}(X)$: intra-player-season variability (due to randomness)
- $\text{Var}_{sm}(X)$: variability between players for a given season

3.3.3 Formulation using the mixed model

We can rewrite the previous formula using the variance terms defined earlier. Within the framework of the linear mixed-effects model defined previously, the discrimination meta-metric can be expressed explicitly.

In the context of the mixed-effects model, the variance component due to randomness is given by:

$$\mathbb{E}_{spm}[V_{spm}[X]] = \tau_M^2,$$

where τ_M^2 represents the random sampling variance (intra-player-season).

The total variance of the metric X, for a season s and a metric m, is then:

$$V_{sm}[X] = \sigma_{PM}^2 + \sigma_{SPM}^2 + \tau_M^2.$$

By plugging these expressions into the general discrimination formula, we obtain:

$$\mathcal{D}_m = 1 - \frac{\tau_M^2}{\sigma_{PM}^2 + \sigma_{SPM}^2 + \tau_M^2} = \frac{\sigma_{PM}^2 + \sigma_{SPM}^2}{\sigma_{PM}^2 + \sigma_{SPM}^2 + \tau_M^2}. \tag{2}$$

This expression allows us to interpret discrimination as the proportion of total variance between players that is actually attributable to performance differences (signal), rather than statistical noise (sampling randomness).

3.3.4 Bootstrap

3.3.5 Estimation

For expectation and variance, we will use empirical estimators. For the variance Var_{spm}, we will apply a bootstrap approach.

Bootstrap TO EXPLAIN OR NOT??

After transformations, we get:

- Var_{sm} becomes $\widehat{V}_{sm}[X] = \frac{1}{P}\sum_{p=1}^{P}\left(X_{spm} - \overline{X}_{s\cdot m}\right)^2$
- $\mathbb{E}_{sm}[V_{spm}[X]]$ becomes $\widehat{\mathbb{E}}_{sm}\left[V_{spm}[X]\right] = \frac{1}{P}\sum_{p=1}^{P}\text{BV}[X_{spm}]$

where:

- X_{spm} is the observed value for player p, in season s, for metric m,
- $\overline{X}_{s\cdot m}$ is the **average of** X_{spm} across all players in season s and metric m,
- $\text{BV}[X_{spm}]$ denotes the **bootstrap variance** for player p in season s, computed from the replicates,
- P is the total number of active players in that season.

We then obtain the estimator formula for discrimination:

$$\widehat{D}_{sm} = 1 - \frac{\frac{1}{P}\sum_{p=1}^{P}\text{BV}[X_{spm}]}{\frac{1}{P}\sum_{p=1}^{P}\left(X_{spm} - \overline{X}_{s\cdot m}\right)^2}$$

3.4 Stability

Stability measures how constant a metric is for a given player from one season to another. In other words: Does this statistic reflect a persistent trait of the player, or does it vary significantly due to context, teammates, luck, etc.?

For stability, we will use the same approach: we compute a variance ratio:

$$\text{Stability} = 1 - \frac{\text{average variation over time for a player (excluding randomness)}}{\text{overall variation across all players and seasons (excluding randomness)}}$$

Thus, the stability of a metric m is defined by the formula:

$$S_m = 1 - \frac{E_p\left[\text{Var}_{p,m}(X) - \text{Var}_{spm}(X)\right]}{\text{Var}_m(X) - E_{spm}[\text{Var}(X_{spm})]}$$

Where:

- X_{spm}: Value of metric m for player p in season s
- $\text{Var}_{p,m}(X)$: Variance of this metric for player p across their seasons
- $\text{Var}_{spm}(X)$: Intra-player-season variance (sampling variability, due to randomness)
- $\text{Var}_m(X)$: Total variance of the metric across all players and seasons

A value close to 1 indicates that performances are very consistent from one season to another. A metric with a stability score close to 1 can be considered a good indicator of a player's true value.

3.4.1 Formulation using the mixed model

Just like for discrimination, we can rewrite the previous formula using the variance components defined earlier.

$$S_m = 1 - \frac{\sigma^2_{\text{SM}} + \sigma^2_{\text{SPM}} + \tau^2_{\text{M}} - \tau^2_{\text{M}}}{\sigma^2_{\text{PM}} + \sigma^2_{\text{SM}} + \sigma^2_{\text{SPM}} + \tau^2_{\text{M}} - \tau^2_{\text{M}}} = \frac{\sigma^2_{\text{PM}}}{\sigma^2_{\text{PM}} + \sigma^2_{\text{SM}} + \sigma^2_{\text{SPM}}}.$$

3.4.2 Estimation

We now move on to the estimation. Once again, we will use empirical estimators for expectation and variance, as well as bootstrap.
We obtain the following formula:

$$\hat{S}_m = 1 - \frac{\frac{1}{P}\sum_{p=1}^{P}\frac{1}{S_p}\sum_{s=1}^{S_p}\left[\left(X_{spm} - \overline{X}_{\cdot pm}\right)^2 - \text{BV}[X_{spm}]\right]}{\frac{1}{P}\sum_{p=1}^{P}\frac{1}{S_p}\sum_{s=1}^{S_p}\left[\left(X_{spm} - \overline{X}_{\cdot\cdot m}\right)^2 - \text{BV}[X_{spm}]\right]}$$

where $\overline{X}_{\cdot pm}$ is the mean of metric m for player p over all seasons, $\overline{X}_{\cdot\cdot m}$ is the total mean over all player-seasons, and S_p is the number of seasons played by player p.
Where:

- $\overline{X}_{\cdot pm}$ is the mean of metric m for player p over all seasons,
- $\overline{X}_{\cdot\cdot m}$ is the total mean over all player-seasons, and
- S_p is the number of seasons played by player p.

Proof We start from the theoretical definition of stability:

$$S_m = 1 - \frac{\mathbb{E}_p\left[\mathrm{Var}_{p,m}(X) - \mathrm{Var}_{spm}(X)\right]}{\mathrm{Var}_m(X) - \mathbb{E}_{spm}\left[\mathrm{Var}(X_{spm})\right]}$$

Each term is approximated by its empirical estimator:

- $\mathbb{E}_p[\cdot]$ is estimated by averaging over the P players:

$$\mathbb{E}_p[\cdot] \approx \frac{1}{P}\sum_{p=1}^{P}(\cdot)$$

- $\mathrm{Var}_{p,m}(X)$ is estimated by the variance of X_{spm} around the player-specific mean:

$$\mathrm{Var}_{p,m}(X) \approx \frac{1}{S_p}\sum_{s=1}^{S_p}\left(X_{spm} - \overline{X}_{\cdot pm}\right)^2$$

- $\mathbb{E}_{spm}[\cdot]$ is estimated by averaging over all player-season pairs:

$$\mathbb{E}_{spm}[\cdot] \approx \frac{1}{P}\sum_{p=1}^{P}\frac{1}{S_p}\sum_{s=1}^{S_p}(\cdot)$$

- $\mathrm{Var}_m(X)$ is estimated by the overall variance around the total mean:

$$\mathrm{Var}_m(X) \approx \frac{1}{P}\sum_{p=1}^{P}\frac{1}{S_p}\sum_{s=1}^{S_p}\left(X_{spm} - \overline{X}_{\cdot\cdot m}\right)^2$$

- $\mathrm{Var}_{spm}(X)$ is estimated by the sampling variance (bootstrap), denoted $\mathrm{BV}[X_{spm}]$

By substituting expectations and variances with their estimates, we obtain:

$$\hat{S}_m = 1 - \frac{\frac{1}{P}\sum_{p=1}^{P}\frac{1}{S_p}\sum_{s=1}^{S_p}\left[\left(X_{spm} - \overline{X}_{\cdot pm}\right)^2 - \mathrm{BV}[X_{spm}]\right]}{\frac{1}{P}\sum_{p=1}^{P}\frac{1}{S_p}\sum_{s=1}^{S_p}\left[\left(X_{spm} - \overline{X}_{\cdot\cdot m}\right)^2 - \mathrm{BV}[X_{spm}]\right]}$$

This constitutes the empirical estimator of the stability S_m.

3.5 Independence

We now move on to the computation of independence. For this, we will use a different approach, as independence is not based on variances, but rather on the correlation of information between metrics.

3.5.1 Definition

Independence measures the extent to which a metric m provides new information compared to a set of metrics M.

In other words:

- Is this stat redundant, or does it offer complementary insight?
- Is it fully predicted by the others, or does it capture a unique dimension?

3.5.2 Formula for independence

For a metric m and a set of metrics M, independence is defined as:

$$I_{m|M} = 1 - R^2$$

Where R^2 is the coefficient of determination from a regression of m on the metrics M.

In other words: we are trying to determine how well m can be explained by the other metrics. Do you want to know whether metric m brings new information, or if it is fully predicted by other metrics M?

If the relationship between variables is linear, you can perform a standard regression and look at the R^2. But if the relationship is nonlinear or complex, standard regression fails to properly capture the dependence. This is where copulas come in.

3.5.3 Computation

Copulas Copulas allow us to describe the dependence between several random variables, independently of their marginal distributions.

Ranks To use the concept of copula, we need to work with cumulative distribution functions. However, we do not have them here.

To address this issue, we will empirically estimate these distribution functions using ranks.

We define:

$$F_X(x_i) \approx \frac{\text{rank}(x_i)}{n+1}$$

where:

- $\text{rank}(x_i)$ = the rank of x_i among the observations (1 = smallest, n = largest)
- $n + 1$ = we add 1 to the denominator to never exactly reach 1
- $\Rightarrow$ important when applying Φ^{-1} (normal quantile function) afterward

This rank divided by $n+1$ is an **empirical estimate** of $F_X(x_i)$.
It is used to ensure that:

$$u_i \in (0,1) \quad \text{(strictly between 0 and 1)}$$

Which helps to avoid infinite values when applying the normal quantile function Φ^{-1}:

$$\Phi^{-1}(1) = +\infty, \quad \Phi^{-1}(0) = -\infty$$

- It avoids the extremes 0 and 1 within $[0,1]$
- It ensures:
$$0 < \frac{\text{rank}(x_i)}{n+1} < 1$$
- $\Rightarrow$ essential when applying an inverse transformation of the normal distribution

We want to transform each variable into:

1. a uniform quantile $u \in [0,1]$
2. then into a standard normal variable $Z = \Phi^{-1}(u)$

Result: All variables are now:

- centered
- standardized
- comparable to one another

Normal transformation After empirically estimating the distribution functions, we normalize the metrics using a Gaussian copula.

Each metric X_{spm} (metric m for player p, season s) is transformed into a latent normal variable Z_{spm} via:

$$Z_{spm} = \Phi^{-1}(F_m(X_{spm}))$$

- F_m is the empirical distribution function of metric m
- Φ^{-1} is the quantile function of the standard normal distribution

Result: all variables Z_{spm} follow a standard normal distribution, which allows for consistent comparison of correlations between metrics.

Correlation matrix For a total of M metrics, we compute the correlation matrix between all latent variables Z: $C \in \mathbb{R}^{M \times M}$

Each element $C_{i,j}$ represents the correlation between metrics i and j in the normalized space.

Computing the conditional variance For a target metric m, we want to determine how much of its variance remains *after removing what can be predicted by the other metrics* M.

$$I_{mM} = \frac{\mathrm{Var}(Z_{spm} \mid \{Z_{spq} : q \in M\})}{\mathrm{Var}(Z_{spm})}$$

where:

- Z_{spm} is the specific component for player p in season s for metric m, extracted from the mixed-effects model (thus, noise-free);
- $\mathrm{Var}[Z_{spm}]$ is the total variance of this component in the population;
- $\mathrm{Var}[Z_{spm} \mid \{Z_{spq} : q \in \mathcal{M}\}]$ is the remaining variance of Z_{spm} after explaining what can be predicted from the metrics in set $\mathcal{M}$.

This formula measures the proportion of variance in metric m that is **independent** of what is explained by the other metrics $\mathcal{M}$. A value close to 1 indicates that the metric contains unique information, whereas a value close to 0 means it is redundant with those in $\mathcal{M}$. Note: A value of $\mathcal{I}_{m,\mathcal{M}} = 1$ indicates that metric m is completely independent of the set of metrics $\mathcal{M}$.

Since all variables Z are standardized, we have:

$$\mathrm{Var}(Z_{spm}) = 1 \quad \Rightarrow \quad I_{mM} = 1 - R^2$$

where R^2 is the coefficient of determination from a linear regression of Z_{spm} on the other Z_{spq}, $q \in M$.

Exact matrix computation using the correlation matrix

$$I_{mM} = C_{m,m} - C_{m,M} \cdot C_{M,M}^{-1} \cdot C_{M,m}$$

Where:

- $C_{m,m} = 1$ (variance of a standardized variable)
- $C_{m,M}$ is the vector of correlations between metric m and each metric in M
- $C_{M,M}$ is the submatrix of correlations between the metrics in M

The independence I_{mM} therefore measures the portion of variance in m that is not explained by the other metrics M. A value close to 1 indicates that the metric contains a large amount of unique information.

3.5.4 Approach through regression

We used copulas and Sklar's theorem to perform this calculation. However, we will now provide another explanation, based on regression, for a more intuitive understanding.

We perform a regression:

$$Z_{\mathrm{BPM}} = \alpha + \beta_1 Z_{\mathrm{PTS}} + \beta_2 Z_{\mathrm{AST}} + \varepsilon$$

4 Results

4.1 Analysis of Discrimination and Stability

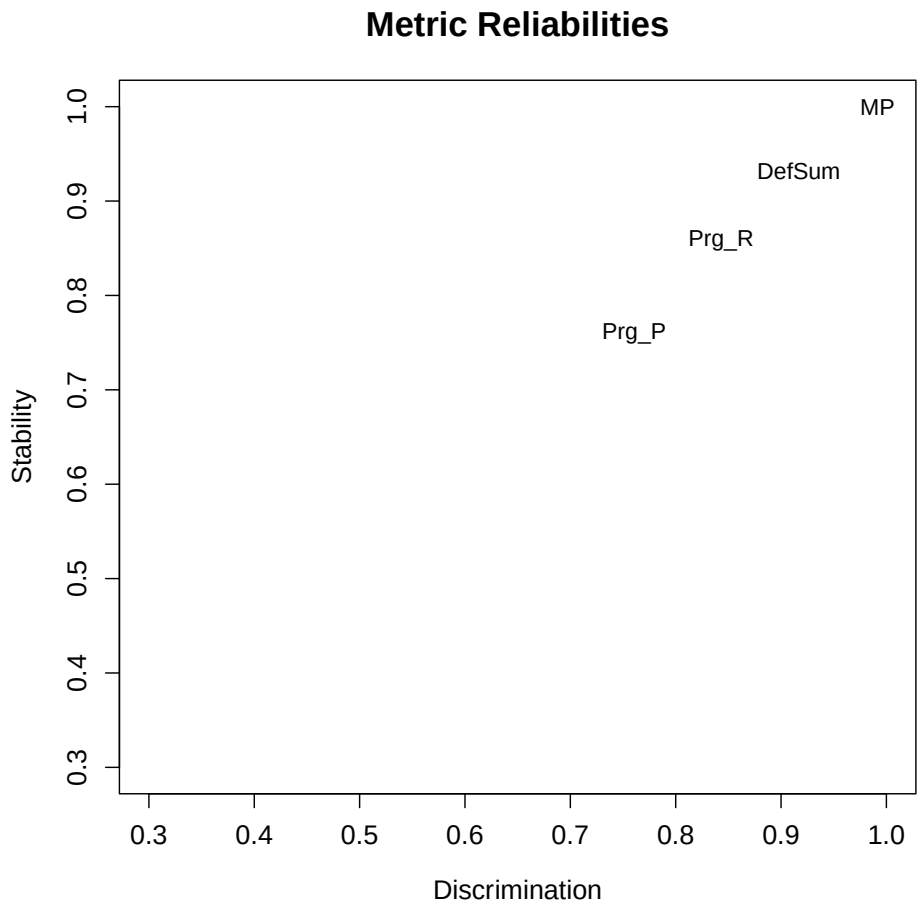

Figure 1: Metric Reliabilities

On figure 1, we see here a plot representing discrimination versus stability. We only observe four variables because all the others are close to zero.

In my opinion, this issue could stem from several causes: poor sampling, normalization problems, or a lack of proper filtering — all of which may lead to low variances. I tried adjusting these parameters, but was unable to obtain better results. Nevertheless, we can still say that three variables stand out (excluding minutes played):

- The total number of defensive actions is the most stable and most discriminating variable. This suggests that it is a very good indicator of individual player performance.
- Next, we have the number of passes and receptions that significantly advanced the ball toward the opponent's goal. This is because these metrics are highly dependent on the player's skill level, which explains the high discrimination score. Furthermore, since they are largely influenced by playing style, they tend to remain stable from one season to another, unless there is a major transfer or injury.

4.2 Analysis of Independance

4.2.1 Principal Component Analysis

We use Principal Component Analysis (PCA) to assess redundancy among the metrics and to identify independent information. The goal is to detect redundancies between metrics and determine whether some of them can be removed or combined without significant loss of information.

Method:

- Apply PCA to the correlation matrix C of the metrics.
- Perform spectral decomposition $C = U\Lambda U^\top$, where:
  - Λ contains the eigenvalues (variance explained by each component),
  - U contains the eigenvectors (principal components).
- Calculate the cumulative proportion of explained variance by the first k components:

$$F_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{M} \lambda_i}$$

Interpretation:

- If F_k is close to 1 for a small k, it indicates a strong redundancy among the metrics.
- Strong redundancy allows dimensionality reduction without significant information loss.
- The first components can be interpreted as synthetic axes (for example, "offensive efficiency", "defensive impact", etc.).

Usefulness:

- Reduce the complexity of analyses,
- Construct more robust and interpretable composite indicators,
- Optimize measurement systems by limiting information duplication.

Figure 5 shows the cumulative explained variance as a function of the number of principal components in a Principal Component Analysis (PCA) applied to the full set of football performance metrics. This curve helps quantify the extent to which the statistical structure of the data can be summarized using a reduced number of dimensions.

We observe that the first three to five components already account for between 70 percent and 85 percent of the total variance, which indicates a high degree of redundancy among the original variables. In other words, a large portion of the information captured by the different metrics (goals, passes, shots, xG, etc.) is correlated and can be effectively compressed without significant loss of signal.

We will not analyze the PCA directly, but instead move on to the independence curves.

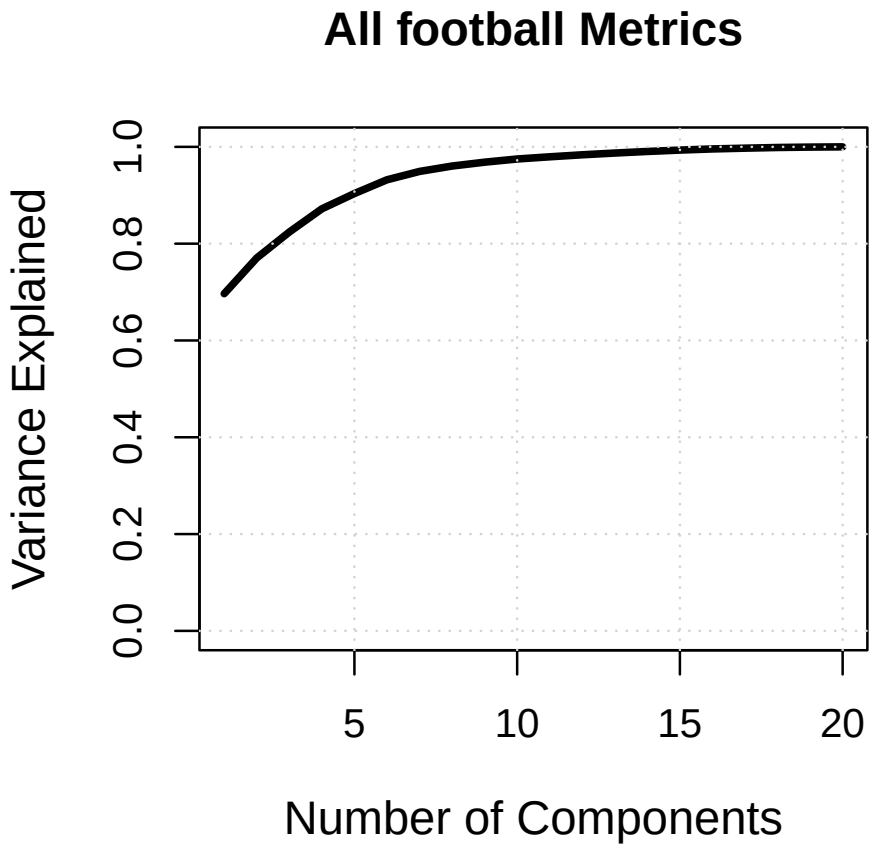

Figure 2: Explained variance

4.2.2 Independence Curves

We use the score I_{mM} to evaluate the independence of a metric m relative to a set M of other metrics. If I_{mM} is close to 0, it means that m is highly redundant with M. Conversely, if I_{mM} is close to 1, m contains independent information. It is important to note that this score directly depends on the choice of the set M.

To analyze this dependence dynamically, we construct **independence curves**. These curves show how I_{mM} evolves as the most correlated metrics with m are progressively removed from the set M.

The methodology followed is as follows: we start with the full set M, then at each step, we remove the metric whose removal maximizes the increase in the score I_{mM}. After each removal, the score is recalculated and recorded. The operation is repeated until the set M is exhausted.

In the resulting curves, the x-axis represents the number of removed metrics, while the y-axis represents the corresponding value of the score I_{mM}.

The interpretation of the shape of the curves provides valuable information:

- If the curve rises quickly after the first removals, it means that one or two metrics explain most of the redundancy of m.
- If the increase is slow and gradual, it suggests that m is structurally independent from the other metrics.
- A plateau close to 1 indicates that the metric has become effectively independent.

Thus, independence curves allow us to:

- Understand which metrics explain the observed redundancy,
- Identify the metrics most similar to a target metric,
- Reduce the dimensionality of datasets by removing redundant metrics,
- Select a subset of informative and uncorrelated metrics.

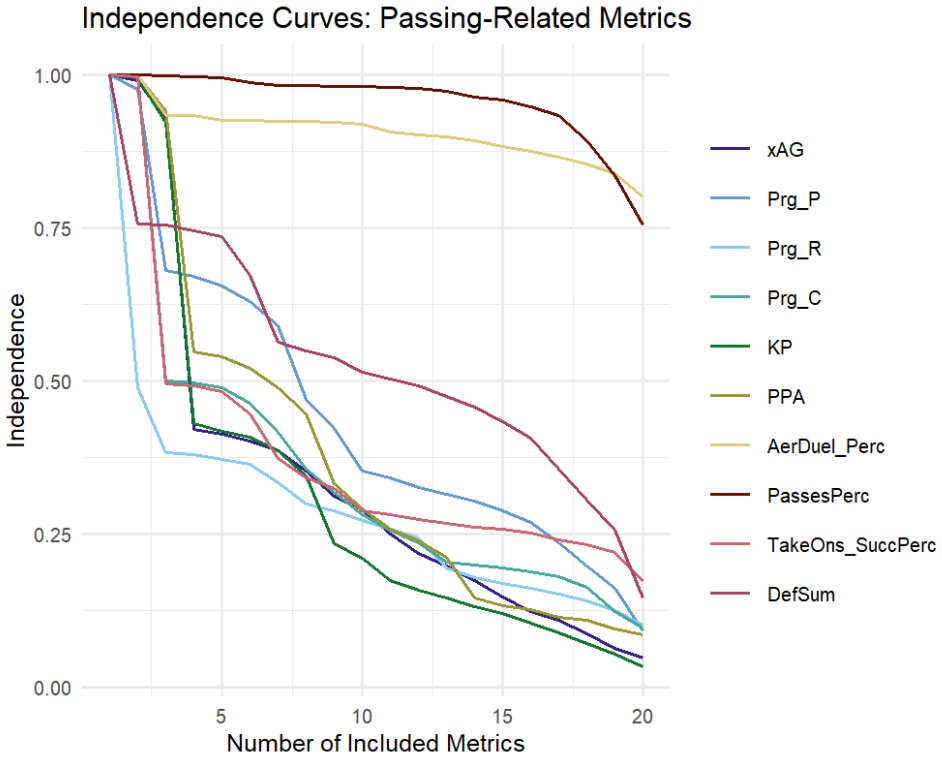

Figure 3: Independence Curves : Goal-Related Metrics

The figure 2 illustrates the independence curves for a set of metrics related to passing and offensive progression, aiming to assess the amount of unique information each metric provides relative to the others. Specifically, it shows how much a metric remains independent as an increasing number of other metrics are included in the conditional model.
We observe that certain variables, such as DefSum, maintain very high independence even when 20 other metrics are considered. This means they capture aspects of the game not reflected by the other statistics, making them particularly valuable indicators for multivariate analysis. Similarly, AerDuel_Perc follows a comparable trend: its curve remains high, suggesting it captures a physical or athletic dimension independent of traditional

offensive indicators.
In contrast, metrics centered on offensive creation, such as xAG, Prg_P, Prg_R, Prg_C, PPA, or KP, show curves that drop sharply, indicating strong informational redundancy. They all measure aspects of forward passing or chance creation and are thus highly correlated with each other. These results imply that, within a statistical model, including them together could cause issues (multicollinearity) or dilute interpretability.

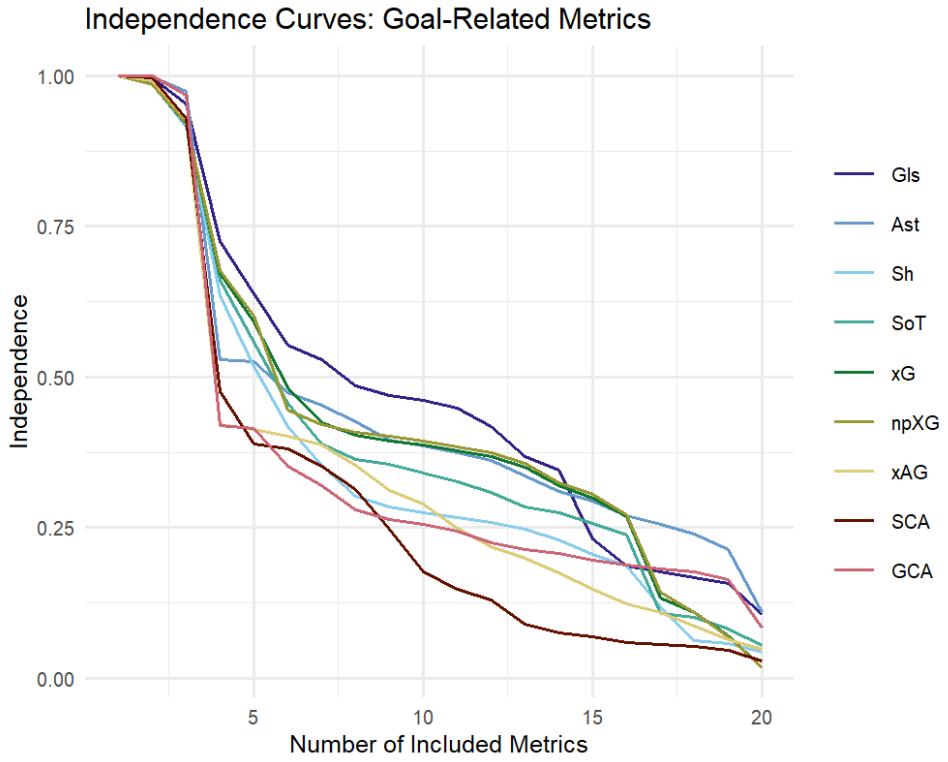

Figure 4: Independence Curves : Passing-Related Metrics

Figure 3 presents the independence curves for a set of metrics related to goal production, aiming to quantify the unique information each variable provides relative to the others.
We observe that certain metrics, such as the number of goals scored (Gls) or assists (Ast), maintain a relatively high level of independence, especially when few other variables are included in the conditional model. This suggests that these actions, although influenced by other offensive factors, remain partially unpredictable and depend on complex or even random contextual elements, such as positioning or finishing ability.
In contrast, other metrics such as SCA (Shot Creating Actions) and GCA (Goal Creating Actions) show a rapid drop in their independence score, indicating strong redundancy with other offensive statistics. These metrics, being composite aggregates, capture information that is already largely represented by variables such as shots (Sh), shots on target (SoT), or expected goals (xG).
Moreover, the curves of xG and npXG (non-penalty expected goals) almost completely overlap, indicating that their contributions are nearly identical and that it is often unnecessary to include both simultaneously in a model.

Figure 4 illustrates a hierarchical clustering analysis of the various metrics used to describe player performance in football. This method provides a visual representation of redundancy and complementarity among variables by grouping together those that are

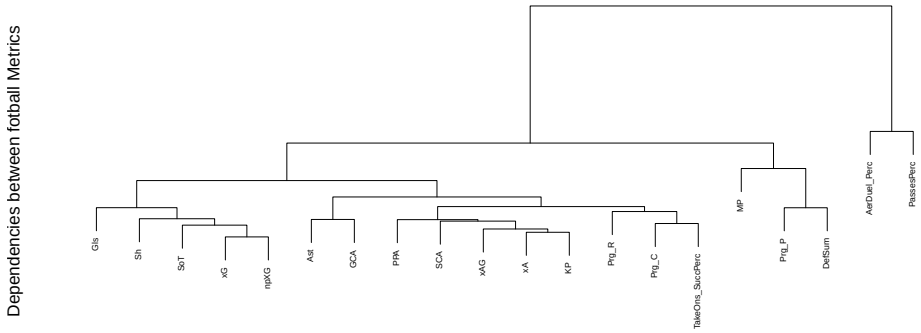

Figure 5: Hierarchical clustering

highly correlated. The dendrogram reveals a clear structure composed of three main clusters.

The first cluster, on the left side of the dendrogram, includes metrics related to goal scoring: Gls (Goals), Sh (Shots), SoT (Shots on Target), xG and npXG (Expected Goals, with and without penalties). These variables all capture some form of direct contribution to scoring or goal creation, and are therefore logically grouped together due to their strong intercorrelation.

The second cluster, located in the center, includes advanced offensive creation metrics such as Ast (Assists), GCA (Goal Creating Actions), PPA, SCA, xAG, xA, KP, and progressive passes (Prg_R, Prg_C). These variables primarily describe the ability to create chances or advance the play, and their close grouping confirms the strong interdependence previously observed in the independence curves. This block can be interpreted as a latent structure corresponding to offensive "build-up play."

Finally, a third, more heterogeneous group appears on the right side of the dendrogram. It includes MP (Minutes Played), Prg_P (Progressive Passes), DefSum (Defensive Contribution), as well as AerDuel_Perc (Aerial Duel Success Rate) and PassesPerc (Pass Accuracy). This cluster reflects a broader functional diversity, including metrics related to stamina, defensive involvement, and technical efficiency. Notably, DefSum and AerDuel_Perc form a distinct sub-cluster, suggesting they capture very specific dimensions that are largely uncorrelated with offensive variables.

Overall, this dendrogram supports the findings from previous figures: offensive metrics tend to form dense and highly interdependent clusters, while certain physical or defensive variables clearly stand apart, highlighting their unique and complementary informational value.

5 Conclusion and Outlook

The analyses conducted in this work, based on meta-analytic tools applied to football statistics, highlight a high degree of redundancy among certain player performance metrics. In particular, the independence curves (Figures 2 and 3) show that many offensive variables—such as progressive passes, expected goals (xG), and actions leading to a shot or a goal (GCA, SCA)—are highly correlated. This is largely due to the fact that these metrics often capture similar aspects of the game, primarily related to offensive build-up and finishing.

While this redundancy can be problematic in statistical models (especially due to multicollinearity), it also offers a clear advantage: the ability to simplify analysis by reducing the number of variables without losing the core informational content.

This project has been a highly enriching experience. It provided an opportunity to experiment with various statistical methods on football data and to develop a deeper understanding of the analytical techniques proposed in the literature. Applying these methods in a practical context helped reinforce both theoretical and technical knowledge.

That said, one of the limitations of this work was the amount of data available. We were not able to collect as much data as initially intended, which may have affected the scope and strength of our findings. As a result, a key next step will be to expand the dataset in order to enhance the robustness and reliability of the results. This will not only allow for more conclusive analyses but also open the door to more advanced explorations in future work.

6 References

- Alexander M. Franks, Alexander D'Amour, Daniel Cervone, and Luke Bornn. *Meta-analytics: Tools for understanding the statistical properties of sports metrics.* Journal of Quantitative Analysis in Sports, 2016, Vol. 12(4), pp. 151–165. DOI: https://doi.org/10.1515/jqas-2016-0098

- Frédéric Planchet, *Introduction à la théorie des copules*, Support de cours 2023-2024. Available online: https://www.ressources-actuarielles.net/C1256F13006585B2/0/0B9DF464E9543283C1256F130067B2F9/$FILE/Copules.pdf?OpenElement

- Efron, B., and C. Morris. 1975. "Data Analysis Using Stein's Estimator and Its Generalizations." *Journal of the American Statistical Association* 70:311–319.

- Efron, B., and R. Tibshirani. 1986. "Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy." *Statistical Science* 1:54–75.

- Fisher, R. A. 1925. *Statistical Methods for Research Workers.* Guildford: Genesis Publishing Pvt Ltd.

- Clot, Denis. *Analyse de données et clustering.* Lecture notes, Master 1 ES, Institut de Science Financière et d'Assurances, 2023–2024.

- Ribereau, Pierre. *Statistiques inférentielles.* Lecture notes, Master 1 ES, Institut de Science Financière et d'Assurances, 2023–2024.

Appendix

A Correspondence Table

Name	Description
MP	Minutes played
Gls	Goals scored
Ast	Assists
xG	Expected Goals
npxG	Non-penalty Expected Goals
SCA	Shot-Creating Actions
GCA	Goal-Creating Actions
PassesPerc	Pass completion percentage
Prg_P	Progressive passes
Prg_C	Progressive carries
xA	Expected Assists
xAG	Expected Assisted Goals
KP	Key Passes
PPA	Passes into penalty area
TakeOns_SuccPerc	Successful dribbles percentage
Prg_R	Progressive runs
AerDuel_Perc	Aerial duel success percentage
DefSum	Defensive actions total
Gls_Sh	Goals per shot
SoT_Sh	Shots on target per shot

Table 1: Mapping of the metrics present in `gameLogs`

B R Code for Bootstrap and Meta-Metrics

```
# 0. Libraries and Data
    ------------------------------------------------------------
library(dplyr)
library(sbgcop)
library(xtable)
library(readr)
library(plyr)
library(ggplot2)
library(Rgbp)


dat.dir <- 'Data'
fig.dir <- 'Figs'


# 1. Preliminary calculations for discrimination and stability
    -----------------

```

```
## 1.1. Bootstrap
   ------------------------------------------------------------------


### Aggregators

basic_aggr <- list(
# totals
MP      = function(df) sum(df$MP,      na.rm = TRUE),
Gls     = function(df) sum(df$Gls,     na.rm = TRUE),
Ast     = function(df) sum(df$Ast,     na.rm = TRUE),
xG      = function(df) sum(df$xG,      na.rm = TRUE),
npxG    = function(df) sum(df$npxG,    na.rm = TRUE),
SCA     = function(df) sum(df$SCA,     na.rm = TRUE),
GCA     = function(df) sum(df$GCA,     na.rm = TRUE),
Prg_P   = function(df) sum(df$Prg_P,   na.rm = TRUE),
Prg_C   = function(df) sum(df$Prg_C,   na.rm = TRUE),
xA      = function(df) sum(df$xA,      na.rm = TRUE),
xAG     = function(df) sum(df$xAG,     na.rm = TRUE),
KP      = function(df) sum(df$KP,      na.rm = TRUE),
PPA     = function(df) sum(df$PPA,     na.rm = TRUE),
Prg_R   = function(df) sum(df$Prg_R,   na.rm = TRUE),
DefSum = function(df) sum(df$DefSum, na.rm = TRUE),

# ratios / percentages
PassesPerc         = function(df) mean(df$PassesPerc,         na.
   rm = TRUE),
TakeOns_SuccPerc   = function(df) mean(df$TakeOns_SuccPerc,   na.
   rm = TRUE),
AerDuel_Perc       = function(df) mean(df$AerDuel_Perc,       na.
   rm = TRUE),
Gls_Sh             = function(df) mean(df$Gls_Sh,             na.
   rm = TRUE),
SoT_Sh             = function(df) mean(df$SoT_Sh,             na.
   rm = TRUE)
)


## 1.2.  Definition of the function
   ---------------------------------------------

bootstrap_foot_season <- function(gameLogs, identity = FALSE,
aggr_funs = basic_aggr) {

        # Extract teams
        teams <- unique(gameLogs$Tm)

        # Resample dates for each team
        resampledDates <- list()
        for (tm in teams) {
                dates <- unique(gameLogs$Date[gameLogs$Tm == tm
                   ])
                resampledDates[[tm]] <- if (identity) dates else
                    sample(dates, replace = TRUE)
        }

        # create the bootstrapped season
```

```
                resampled_season <- do.call(rbind, lapply(teams,
                    function(tm) {
                        team_data <- gameLogs[gameLogs$Tm == tm, ]
                        do.call(rbind, lapply(resampledDates[[tm]],
                        function(d) team_data[team_data$Date == d, ]))
                }))

                # Aggregate stats for each player
                player_stats <- ddply(resampled_season, .(Player),
                    function(df_player) {
                        sapply(aggr_funs, function(f) f(df_player))
                })

                return(player_stats)
        }

        ## 1.3 Verification
            ------------------------------------------------------------

        gameLogs_sub <- gameLogs[1:200, ] #200 rows to keep things
            lightweight

        # Run the function without bootstrap (identity = TRUE)
        reconstructed <- bootstrap_foot_season(gameLogs_sub, identity =
            TRUE)

        # Use the same results as the reference value
        season_totals <- reconstructed

        # Compare the stats
        common_vars <- setdiff(colnames(reconstructed), "Player")

        # Visualization loop (scatterplot for each metric)
        for (metric in common_vars) {
                plot(reconstructed[[metric]],
                season_totals[[metric]],
                xlab = paste("Reconstructed", metric),
                ylab = paste("Original", metric),
                main = metric)
                abline(a = 0, b = 1, col = "red")
                browser()
        }


        ## 1.4. Bootstrap with 20 iterations
            ------------------------------------------

        # Select players who have played at least 50 minutes
        minMP <- 50
        minPlayers <- allSeasonsData$Player[allSeasonsData$MP > minMP]
        totals <- allSeasonsData[allSeasonsData$Player %in% minPlayers,
            ]



        # Loop
        nboot <- 20
        gdfSubset <- gameLogs[gameLogs$Player %in% minPlayers, ]
```

```
115
116         repsList <- lapply(1:nboot, function(x) {
117                 print(sprintf("boostrapping␣%i␣of␣%i", x, nboot))
118                 bootstrap_foot_season(gdfSubset, identity = FALSE)
119         })
120         reps <- do.call(rbind, repsList)
121
122
123
124         # 2. Calculation of stability and discrimination
                ------------------------
125
126         ## 2.1. Calculation of variances
                ---------------------------------------------
127
128         getVariances <- function(totals,
129         reps,
130         seasons="23_24",
131         exclude=c(),
132         minSeason=4) {
133
134                 exclude <- c(exclude, "Player", "Tm")
135                 commonMetrics <- setdiff(intersect(colnames(totals),
                        colnames(reps)),
136                 exclude)
137
138                 ## Total variance
139                 TV <- apply(totals, 2, function(w) var(w, na.rm=TRUE))[
                        commonMetrics]
140
141                 ## Single season variance for calculating discrimination
142                 SV <- apply(totals[totals$Season %in% seasons, ], 2,
143                 function(w) var(w, na.rm=TRUE))[commonMetrics]
144
145                 bootstrapVars <- ddply(reps, .(Player), function(x) {
146                         if(nrow(x) < 10) {
147                                 vec <- c()
148                         } else{
149                                 vec <- apply(x, 2, function(w) var(w, na
                                        .rm=TRUE))
150                                 vec["Player"] <- x$Player[1]
151                         }
152                         vec
153                 })
154
155                 ## Average bootstrap vars within a season
156                 BV <- colMeans(data.matrix(bootstrapVars[, commonMetrics
                        ]), na.rm=TRUE)
157
158                 withinPlayerVar <- ddply(totals, .(Player), function(x)
                        {
159                         if(nrow(x) >= minSeason) {
160                                 vec <- apply(x, 2, function(w) var(w, na
                                        .rm=TRUE))
161                         } else{
162                                 vec <- rep(NA, ncol(x))
163                                 names(vec) <- colnames(x)
164                         }
```

```
                        vec["Player"] <- x$Player[1]
                        vec
                })
                WV <- colMeans(data.matrix(withinPlayerVar[,
                    commonMetrics]), na.rm=TRUE)

                intersectPlayers <- intersect(withinPlayerVar$Player,
                    bootstrapVars$Player)
                commonCols <- intersect(colnames(withinPlayerVar),
                    colnames(bootstrapVars))
                commonCols <- setdiff(commonCols, c("Player", "Year", "
                    Tm"))
                diffVars <-
                data.matrix(withinPlayerVar[withinPlayerVar$Player %in%
                    intersectPlayers,
                commonCols]) -
                data.matrix(bootstrapVars[bootstrapVars$Player %in%
                    intersectPlayers,
                commonCols])
                DV <- colMeans(diffVars, na.rm=TRUE)[commonMetrics]

                discriminationScores <- 1 - BV / SV

                stabilityScores <- 1 - (WV - BV) / (TV - BV)

                list(TV=TV, BV=BV, WV=BV, SV=SV,
                discrimination=discriminationScores,
                stability=stabilityScores)

        }

        varList <- getVariances(allSeasonsData, reps, minSeason = 1,
            seasons = "23_24")

        disc <- varList$discrimination
        stab <- varList$stability


        ## 2.2. Discrimination vs. Stability plot
            ----------------------------

        pdf(file.path(fig.dir, "reliability.pdf"))

        plot(disc, stab, cex=0,
        xlab="Discrimination", ylab="Stability",
        xlim=c(0.3, 1.0), ylim=c(0.3, 1.0),
        main="Metric␣Reliabilities", cex.axis=1.2, cex.main=1.5, cex.lab
            =1.2)

        text(disc, stab, labels=names(stab), cex=0)

        dev.off()




        # 3. Independance ---------------------------------------------
```

```
## 3.1. Data formatting ---------------------------------
row_names <- paste(allSeasonsData$League, allSeasonsData$Player,
seq_len(nrow(allSeasonsData)), sep = " - ")
allSeasonsData_clean <- allSeasonsData[, !(names(allSeasonsData)
    %in%
c("League", "Player", "Season"))]
rownames(allSeasonsData_clean) <- row_names
allSeasonsData <- allSeasonsData_clean
rm(allSeasonsData_clean)

## 3.2. Gaussian copulas -------------------------------
res <- sbgcop::sbgcop.mcmc(allSeasonsData) # Gaussian copula

C <- apply(res$C.psamp, c(1,2), mean) # Mean matrix

imputedTab <- res$Y.pmean


## 3.3. Dendrogram -------------------------------------------
pdf(file.path(fig.dir, "fballDendro.pdf"), width=14)
plot(hclust(dist(abs(C))), main="", xlab="", axes=F, sub="",
ylab="Dependencies between fotball Metrics", cex=0.75, cex.lab
    =1.5)
dev.off()


## 3.4. PCA
    ------------------------------------------------------------

### Eigen decomposition
eig <- eigen(C)
vals <- eig$values
vecs <- eig$vectors
head(cumsum(vals)/sum(vals), n=20)


### Variance explained figure
pdf(file.path(fig.dir, "fballVarExplained.pdf"), width=4,
    height=4)
plot(cumsum(vals)/sum(vals),ylim=c(0,1),pch=19,ylab="Variance
    Explained",
xlab="Number of Components", main="All football Metrics",
type="l",
lwd=3,
cex.lab=1.2)
grid()
dev.off()



### Return conditional variance
condVar <- function(cov, varCols, condCols=NULL) {
        if(is.null(condCols)) {
                condCols <- setdiff(colnames(cov), varCols)
        }
        allCols <- union(varCols, condCols)
        cov <- cov[allCols, allCols]
        cov[varCols, varCols] - cov[varCols, condCols] %*%
```

```
267                 solve(cov[condCols, condCols]) %*% cov[condCols, varCols
                        ]
268         }
269
270
271         ### PCA by hand, without any package
272         normTab <- apply(imputedTab, 2, function(col) {
273                 n <- length(col)
274                 F <- ecdf(col)
275                 qnorm(n/(n+1)*F(col))
276         })
277         rownames(normTab) <- rownames(imputedTab)
278
279         for(i in 1:3) {
280                 vi <- vecs[, i]
281                 names(vi) <- colnames(normTab)
282                 superstat <- as.matrix(normTab) %*% vi
283                 names(superstat) <- rownames(normTab)
284                 print(tail(sort(superstat), n=15))
285                 print(head(sort(superstat), n=15))
286                 print(head(sort(abs(vi), decreasing=TRUE), n=10))
287
288         }
289
290         v1 <- vecs[, 1]
291         v2 <- vecs[, 2]
292         v3 <- vecs[, 3]
293         v4 <- vecs[, 4]
294         v5 <- vecs[, 5]
295         names(v1) <- colnames(normTab)
296         superstat1 <- as.matrix(normTab) %*% v1
297         names(superstat1) <- rownames(normTab)
298         names(v2) <- colnames(normTab)
299         superstat2 <- -1 *as.matrix(normTab) %*% v2
300         names(v3) <- colnames(normTab)
301         superstat3 <- -1 *as.matrix(normTab) %*% v3
302         names(v4) <- colnames(normTab)
303         superstat4 <- -1 *as.matrix(normTab) %*% v4
304         names(v5) <- colnames(normTab)
305         superstat5 <- -1 *as.matrix(normTab) %*% v5
306         rnms <- rownames(normTab)
307         rnms <- sapply(rnms, function(nm) paste(unlist(strsplit(nm, "␣-␣
                ")))[1:2],
308         collapse="␣-␣"))
309         names(superstat1) <- names(superstat2) <- names(superstat3) <-
310         names(superstat4) <- names(superstat5) <- rnms
311
312         RsquaredMatrix <- MostSimilar <- matrix(1, nrow=ncol(C), ncol=
                ncol(C))
313         rownames(RsquaredMatrix) <- rownames(MostSimilar) <- colnames(C)
314         for(metric in colnames(C)) {
315                 print(metric)
316                 nextOut <- metric
317                 count <- ncol(C)
318                 removedSet <- c(metric)
319                 rsq <- condVar(C, metric, setdiff(colnames(C), c(
                        removedSet)))
320                 RsquaredMatrix[metric, count] <- rsq
```

```
                MostSimilar[metric, count] <- nextOut
                count <- count - 1

                while(count > 1) {

                        remaining <- setdiff(colnames(C), removedSet)
                        idx <- which.max(sapply(remaining, function(x)
                        condVar(C, metric, setdiff(colnames(C), c(
                           removedSet, x)))))
                        nextOut <- setdiff(colnames(C), removedSet)[idx]
                        removedSet <- c(removedSet, nextOut)
                        rsq <- condVar(C, metric, setdiff(colnames(C),
                           removedSet))
                        RsquaredMatrix[metric, count] <- rsq
                        MostSimilar[metric, count] <- nextOut
                        count <- count - 1
                }
                MostSimilar[metric, 1] <- setdiff(remaining, nextOut)
        }


        # 4. Plot
           ------------------------------------------------------------

        ## 4.1. Download function ----------------------------
        save_plot <- function(plot, filename) {
                png(filename = file.path(fig.dir, filename), width =
                   1000, height = 800)
                print(plot)
                dev.off()
        }

        ## 4.2. Goal metrics ----------------------------------------
        goal_metrics <- c("Gls", "Ast", "Sh", "SoT", "xG", "npXG", "xAG"
           , "SCA", "GCA")

        valid_metrics_goals <- goal_metrics[goal_metrics %in% rownames(
           RsquaredMatrix)]
        submat_goals <- RsquaredMatrix[valid_metrics_goals, , drop =
           FALSE]
        df_goals <- melt(submat_goals)
        colnames(df_goals) <- c("Metric", "Included_Metrics", "
           Independence")
        df_goals$Included_Metrics <- as.numeric(as.character(
           df_goals$Included_Metrics))

        colors_goals <- c("#332288", "#6699CC", "#88CCEE", "#44AA99",
        "#117733", "#999933", "#DDCC77", "#661100",
        "#CC6677")

        p_goals <- ggplot(df_goals, aes(x = Included_Metrics, y =
           Independence,
        color = Metric)) +
        geom_line(size = 1.1) +
        scale_color_manual(values = colors_goals) +
        labs(title = "Independence Curves: Goal-Related Metrics",
        x = "Number of Included Metrics",
        y = "Independence") +
```

```
        theme_minimal(base_size = 25) +
        theme(
        legend.position = "right",
        legend.title = element_blank(),
        legend.text = element_text(size = 20),
        legend.key.size = unit(1.8, "cm"),
        legend.spacing.y = unit(0.4, "cm")
        )

        print(p_goals)
        save_plot(p_goals, "independence_goals_ggplot.png")



        ## 4.2. Pass metrics -------------------------------------
        pass_metrics <- c("xAG", "Prg_P", "Prg_R", "Prg_C", "KP", "PPA",
        "AerDuel_Perc", "PassesPerc", "TakeOns_SuccPerc",
        "DefSum")

        valid_metrics_passes <- pass_metrics[pass_metrics %in% rownames(
           RsquaredMatrix)]
        submat_passes <- RsquaredMatrix[valid_metrics_passes, , drop =
           FALSE]
        df_passes <- melt(submat_passes)
        colnames(df_passes) <- c("Metric", "Included_Metrics", "
           Independence")

        df_passes$Included_Metrics <- as.numeric(as.character(
           df_passes$Included_Metrics))

        colors_passes <- c("#332288", "#6699CC", "#88CCEE", "#44AA99",
        "#117733", "#999933", "#DDCC77", "#661100",
        "#CC6677", "#AA4466")

        p_passes <- ggplot(df_passes, aes(x = Included_Metrics, y =
           Independence,
        color = Metric)) +
        geom_line(size = 1.1) +
        scale_color_manual(values = colors_passes) +
        labs(title = "Independence Curves: Passing-Related Metrics",
        x = "Number of Included Metrics",
        y = "Independence") +
        theme_minimal(base_size = 25) +
        theme(
        legend.position = "right",
        legend.title = element_blank(),
        legend.text = element_text(size = 20),
        legend.key.size = unit(1.8, "cm"),
        legend.spacing.y = unit(0.4, "cm")
        )
```