xD

(Expected dribbling)

Evaluación de la probabilidad inicial de completar un regate

Jaime Barrio Jimenez
Lucas Bracamonte

Junio 2025

Sport Data Campus | Sevilla FC | UCAM

Universidad Católica San Antonio de Murcia (UCAM)
Master en Big Data aplicado al Scouting en Fútbol
Estudiante: Jaime Barrio Jimenez
Tutor: Lucas Bracamonte

Nueva métrica para el futbol: Regate esperado (xD)

Evaluación de la probabilidad inicial de completar un regate

Índice

Tablas

Imágenes

Abstract

El regate es una acción fundamental en el fútbol desde sus orígenes. A pesar de su importancia táctica, ha recibido menos atención analítica que otras métricas como los tiros o las posesiones. Este trabajo presenta xD (Expected Dribble), un modelo probabilístico que predice el éxito de un regate considerando factores contextuales espaciales, temporales y tácticos. Utilizando datos de StatsBomb, el modelo integra la posición en el campo mediante una matriz de 162 zonas, factores del atacante como velocidad y apoyo de compañeros, presión defensiva incluyendo número y posición de rivales, y contexto del partido como fatiga y marcador. La métrica resultante proporciona probabilidades entre 0.1% y 65%, coherentes con las tasas de éxito observadas en el fútbol profesional, ofreciendo una evaluación granular y contextual que permite análisis táctico avanzado y evaluación objetiva del rendimiento individual en una de las acciones más espectaculares del juego.

Dribbling has been a fundamental action in football since its origins. Despite its tactical importance, it has received less analytical attention than other metrics such as shots or possessions. This work presents xD (Expected Dribble), a probabilistic model that predicts dribbling success by considering spatial, temporal, and tactical contextual factors. Using StatsBomb data, the model integrates field position through a 162-zone matrix, attacker factors such as velocity and teammate support, defensive pressure including number and position of opponents, and match context such as fatigue and score. The resulting metric provides probabilities ranging from 0.1% to 65%, consistent with success rates observed in professional football, offering a granular and contextual assessment that enables advanced tactical analysis and objective evaluation of individual performance in one of the game's most spectacular actions

Agradecimientos:

A mi familia, que a pesar de que en casa no les gusta el fútbol, han sido capaces de aguantar mi intensidad antes de empezar este máster, durante, y lo que nos quedará. Y si, **Fernando**, hijo mio, esto es un agradecimiento para ti. siempre constantes, curiosos, y sin limites. Y para ti **Irene**, hija mía, en los pequeños detalles esta la grandeza de lo que se hace, y se puede hacer de dos maneras: con una sonrisa como tu, o sin ella, elijamos siempre la correcta. Y a mi mujer, **María José**, que tiene alas para dar y regalar a cada uno de los que estamos cerca.

A los compañeros de viaje durante este año:

Daniel Suárez y Xavi Torres, quien me diría que estaría rodeado de tanta calidad. Apoyo constante y comprensión del juego, mas allá de los datos. Gracias infinitas.

Jaume, Daniel Carreño, Cristian Dieguez, Enric Blanch, Javier Sansus Ferri. Parece una tonteria pero con cada pregunta en las clases, con cada publicación de LinkedIn, habéis sumado un grano de arena tras otro a este trabajo. Inspiración y ganas de superación cada vez que encendía el ordenador.

A **SportDataCampus** y su gente. Desde la primera conversación con **Miguel**, hasta la última con **Lúcas** (tutor de este trabajo), pasando por **Lupe**, **David (los dos)**, la clases de **José Rodriguez**, el **Sport Data Forum de Sevilla**, y a tantos que me dejo…ha sido un año espectacular, ¡se me ha pasado volado!

Un hueco especial para **Pablo Sanzol**, que nos ha dado tantas clases, de tal calidad humana y de fútbol que es imposible de cuantificar. Gracias.

Y al gran **Yeremay Hernández**, inspiración absoluta de este trabajo, verlo me hizo pensar en cómo podría cuantificar la calidad de los regates que hace.

Y por último, y no por ello menos importante, a **Carlos**. Por su ayuda por teléfono y en papel. Diseñador de la portada de este trabajo con la imagen de (Fernández, 2024), que me enseña cada día que no hay que tener ningún miedo a reinventarse por enésima vez. Amigo:

FOOTBALL IS LIFE

1 Introducción

1.1 La Revolución Analítica del Fútbol

El fútbol ha experimentado una transformación profunda en las últimas dos décadas, evolucionando desde un deporte analizado principalmente a través de la intuición hacia uno donde los datos complementan la comprensión táctica. Esta revolución comenzó con métricas simples como kilómetros recorridos y porcentaje de posesión, pero rápidamente evolucionó hacia modelos más sofisticados. La introducción de Expected Goals (xG) en 2012 marcó un punto de inflexión, demostrando que era posible cuantificar la calidad de las oportunidades de gol más allá del simple conteo de tiros.

Desde entonces, el ecosistema analítico del fútbol se ha expandido exponencialmente. Métricas como Expected Assists (xA), Expected Threat (xT), y VAEP (Valuing Actions by Estimating Probabilities) han enriquecido nuestra comprensión del juego. Los clubes de élite emplean departamentos enteros dedicados al análisis de datos, mientras que plataformas como StatsBomb, Opta y Wyscout proporcionan datos cada vez más granulares. Esta infraestructura ha permitido avances significativos en el scouting, la preparación táctica y la evaluación del rendimiento, transformando decisiones que antes se basaban en la intuición en procesos respaldados por evidencia empírica.

Sin embargo, esta revolución ha sido desigual. Mientras que las acciones relacionadas con la finalización han sido exhaustivamente modeladas, otras facetas fundamentales del juego permanecen relativamente inexploradas. El pase ha recibido atención considerable, pero principalmente en términos de su contribución a las oportunidades de gol. Las acciones defensivas como tackles e intercepciones se cuentan pero raramente se contextualizan. Y el regate, a pesar de ser una de las acciones más decisivas y espectaculares del fútbol, carece de un framework analítico robusto que capture su complejidad inherente y su contexto.

1.2 El Regate: Arte, Ciencia y Paradoja Táctica

El regate ocupa un lugar único en la taxonomía futbolística. Es simultáneamente la expresión más pura de la habilidad individual y un elemento crucial del juego colectivo. Históricamente, los

grandes regateadores han sido venerados: Garrincha desequilibrando defensas enteras, Maradona saliendo victorioso entre rivales, Ronaldinho convirtiendo el regate en arte, Messi redefiniendo lo posible con balón en los pies. Estos jugadores no solo entretenían; transformaban la dinámica táctica de sus equipos y forzaban a los rivales a adaptar sus sistemas defensivos. Los equipos contrarios les tenian respeto cuando tenian el balon en los pies, por no decir miedo. Esto condiciona muchísimo las decisiones defensivas, y a estos jugadores su reputación les precede. Entonces, ¿se podría medir la imposibilidad de realizar un regate antes de que ocurra?

Desde una perspectiva táctica, el regate cumple funciones múltiples y complejas. En zonas profundas, permite la salida limpia del balón bajo presión, iniciando transiciones ofensivas. En el mediocampo, puede romper líneas de presión y crear superioridad numérica. En el último tercio, desequilibra defensas organizadas y genera espacios para compañeros. Cada contexto requiere no solo habilidades técnicas diferentes, sino también decisiones tácticas distintas sobre cuándo intentar el regate versus otras opciones como el pase o la retención.

La paradoja del regate moderno radica en su relación riesgo-recompensa. Cada vez es mas complicado ver a grandes regateadores precisamente por el riesgo que conlleva. Un regate exitoso puede ser devastador para la defensa rival, pero su fracaso puede resultar en transiciones peligrosas en contra. Esta dualidad ha llevado a filosofías divergentes: mientras algunos entrenadores como Pep Guardiola permiten regates en zonas específicas del campo a sus jugadores, prefiere el pase como primera opción de desequilibrio, otros entrenadores dejan un libre alvedrio mas propio de otros tiempos. La ausencia de métricas sofisticadas para evaluar esta relación riesgo-recompensa ha perpetuado debates basados más en preferencias estilísticas que en análisis objetivos, ¿Podriamos ser objetivos midiendo un regate?

1.3 Hacia una Cuantificación Contextual del Regate

La necesidad de una métrica comprensiva para el regate se vuelve evidente cuando consideramos las limitaciones de los enfoques actuales. La tasa de éxito simple (regates completados/intentados) ignora diferencias fundamentales en dificultad: un regate en el centro del campo sin presión no

es comparable a uno en el área rival rodeado de defensores. Métricas más avanzadas como xPG (Expected Possession Goals) de (Ho, 2018) y VAEP consideran el valor de las acciones pero no la dificultad específica de ejecución. Esta brecha deja a entrenadores y analistas sin herramientas adecuadas para responder preguntas cruciales: ¿Cuándo debe un jugador intentar regatear? ¿Qué factores contextuales aumentan o disminuyen la probabilidad de éxito? ¿Cómo evaluamos objetivamente la efectividad de un regateador más allá de estadísticas superficiales?

Ser capaces de evaluar la osadía de un jugador para intentar un regate de complejidad alta y a la vez ser capaces de evaluar si su osadía se transforma en habilidad o en problemas para el equipo, pueden ayudar a los jugadores a tomar mejores decisiones cuando intenten un regate.

El desarrollo de xD (Expected Dribble) busca abordar estas preguntas fundamentales. Mi enfoque reconoce que el éxito del regate está determinado por una constelación de factores interrelacionados: la ubicación en el campo, la velocidad y dirección del movimiento, la presión defensiva, el apoyo de compañeros, la fatiga acumulada y el contexto del partido. Al integrar estos elementos en un modelo unificado, xD proporciona no solo una predicción de probabilidad, sino también insights sobre los factores que más influyen en cada situación específica.

Este trabajo representa un paso hacia la democratización del análisis avanzado del regate. Al proporcionar una metodología transparente y replicable, espero invitar a futuras investigaciones y refinamientos. Más allá de su utilidad práctica para clubes y analistas, xD contribuye a una comprensión más profunda de una de las acciones más fascinantes del fútbol, intentando acercar el arte intuitivo del regate y la ciencia emergente del análisis futbolístico.

1.4 Objetivos

Este trabajo presenta **xD (Expected Dribble)**, una métrica diseñada específicamente para:

1. Predecir la probabilidad de éxito de un regate antes de su ejecución
2. Considerar el contexto espacial, temporal y táctico
3. Proporcionar una herramienta objetiva para la evaluación del rendimiento
4. Facilitar el análisis táctico y la toma de decisiones

2 Modelo xD: Formula

2.1 Conceptualización del Modelo

El modelo xD se fundamenta en la premisa de que el éxito de un regate no es un evento aleatorio, sino el resultado de la interacción entre tres dimensiones fundamentales: las capacidades y circunstancias del atacante, la organización y efectividad defensiva, y el contexto temporal y situacional del partido. Esta conceptualización tripartita refleja la complejidad inherente del regate, donde múltiples factores convergen en fracciones de segundo para determinar el resultado.

La arquitectura del modelo se expresa mediante la fórmula:

$$xD = A \times (1 - D) \times X$$

Donde:

- A -> representa el potencial ofensivo agregado del atacante en el momento del regate
- D -> cuantifica la efectividad defensiva que reduce la probabilidad de éxito
- X -> captura los factores contextuales externos que modulan la dificultad

Esta estructura multiplicativa fue elegida deliberadamente sobre alternativas aditivas por varias razones:

- Primero, refleja la naturaleza interdependiente de los factores: incluso en la mejor zona del campo (con un alto A), una presión defensiva extrema (con un alto D) debe reducir drásticamente la probabilidad de éxito.
- Segundo, asegura que el modelo produzca probabilidades válidas en el rango [0,1].
- Tercero, permite que cada componente actúe como un modificador proporcional, manteniendo la interpretabilidad del modelo.

2.2 Componente de Ataque (A)

El componente de ataque agrega todos los factores que favorecen al jugador con balón:

$$A = Z_{campo} \times V \times C \times M_d$$

2.2.1 Factor Espacial (Z_{campo})

El corazón del componente ofensivo es Z_{campo}, una matriz de probabilidades base que divide el campo en 162 zonas (18 columnas × 9 filas), como ya hizo (Moore, 2018) en su Expected Possession Goals. Esta granularidad captura las variaciones sutiles en la dificultad del regate según la ubicación. La matriz fue calibrada empíricamente para reflejar realidades tácticas fundamentales:

- **Zonas centrales del medio campo** presentan las probabilidades más altas (hasta 0.875), donde los jugadores tienen máximo espacio y opciones
- **Áreas rival** muestran valores mínimos (0.003-0.020), reflejando la alta densidad defensiva y la preferencia por disparar
- **Bandas** consistentemente penalizadas respecto al centro, capturando la limitación espacial, el jugador está mas arrinconado
- **Asimetría campo propio vs rival**, reconociendo que la presión defensiva intensifica cerca de la portería que defienden

La matriz base se ajusta dinámicamente mediante un factor de distancia euclidiana:

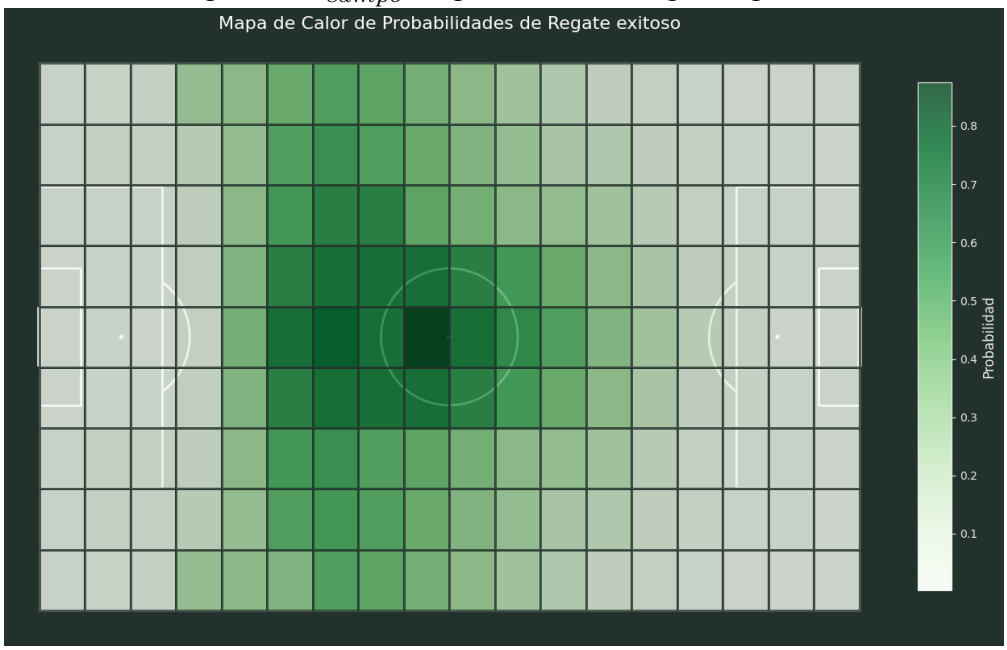
$$F_{distancia} = 0.75 + 0.25 \times \left(1 - \frac{\sqrt{(120 - x)^2 + (40 - y)^2}}{134}\right)$$

Este ajuste sutil pero importante reconoce que incluso dentro de la misma zona, la proximidad a la portería rival incrementa la intensidad defensiva. Los valores de 120 en la X representa el largo del campo, y el 40 en la Y la mitad del ancho, que es donde se ubica la portería rival. El 134, son los metros de distancia máxima posible desde una de las esquinas opuestas del campo.

2.2.2 Factor de Velocidad (V)

La velocidad del atacante influye significativamente en el éxito del regate, modelado como:

Figure 1: Z_{campo} Mapa de calor de regate esperado



$$V = 0.5 + 0.3 \times \min\left(\frac{v_{atacante}}{36}, 1\right)$$

Esta formulación captura insights clave:

- Un jugador estático ($v = 0$) tiene desventaja significativa ($V = 0.5$), siendo más predecible
- La ventaja aumenta linealmente hasta velocidades de sprint ($v = 36$ km/h tomada como velocidad maxima, puede ser otra y ser ajustable)
- No se penalizan velocidades extremas, reconociendo que el control a alta velocidad es una habilidad, no una limitación

2.2.3 Factor de Compañeros (C)

La presencia y posición de compañeros crea un efecto complejo:

$$C = C_{espacio} \times C_{atraccin} \times C_{opciones}$$

- $C_{espacio}$ penaliza la congestión de compañeros muy cercanos que reducen el espacio de

maniobra

- $C_{atraccin}$ bonifica ligeramente a compañeros a media distancia que pueden atraer defensores
- $C_{opciones}$ recompensa tener líneas de pase disponibles post-regate

2.2.4 Dirección del Movimiento (M_d)

La dirección del regate relativa a la portería rival afecta la respuesta defensiva:

$$M_d = 0.725 - 0.125 \times \cos(\theta)$$

- Avanzar hacia portería ($\theta = 0°$) genera máxima oposición ($M_d = 0.60$)
- Movimientos laterales encuentran resistencia moderada ($M_d = 0.725$)
- Retroceder permite más libertad pero sigue siendo arriesgado ($M_d = 0.85$)

2.3 Componente Defensivo (D)

La presión defensiva se modela mediante:

$$D = D_{presin} \times D_{cobertura}$$

2.3.1 Presión Individual (D_{presin})

La proximidad del defensor más cercano es crucial:

$$D_{presin} = 0.3 + 0.4 \times \left(1 - \left(\frac{d_{cercano}}{10}\right)^2\right)$$

Esta función cuadrática refleja que la presión aumenta dramáticamente en distancias cortas pero se estabiliza más allá de 5 metros.

2.3.2 Cobertura Colectiva ($D_{cobertura}$)

Los defensores adicionales multiplican la dificultad:

$$D_{cobertura} = \min(1.0, 0.7 + 0.15 \times n_{<2m} + 0.10 \times n_{2-5m})$$

La formulación reconoce que cada defensor adicional reduce las opciones del atacante, con impacto decreciente según la distancia.

2.4 Factores Contextuales (X)

Los elementos externos modulan sutilmente la probabilidad:

$$X = F_t \times M_{partido} \times T_{momento}$$

Como la principal fuente de probabilidad son los factores del atacante y la defensa, en esta variable (X) lo que pretendo es darle valor al contexto de cuando se realiza un regate, que otros factores, por mínimos que sean, pueden afectar al éxito del mismo.

2.4.1 Fatiga Relativa (F_t)

La fatiga afecta tanto al atacante como al defensor:

$$F_t = \frac{0.95 + 0.05 \times (1 - (min_{atacante}/90)^2)}{0.95 + 0.05 \times (1 - (min_{defensor}/90)^2)}$$

Con esto quiero reflejar la pequeña ventaja que tendría un atacante recién salido al campo, respecto a un defensor que lleva, por ejemplo, 80 minutos jugados. En teoría, el jugador fresco tiene una capacidad defensiva/ofensiva mayor en el regate.

2.4.2 Estado del Marcador ($M_{partido}$)

El estado del marcador influye en la intensidad defensiva de manera contraintuitiva pero tácticamente coherente. Nuestra formulación captura estas dinámicas:

$$M_{partido} = \begin{cases} 0.97 & \text{si gana por } \geq 3 \\ 0.98 & \text{si gana por } 1-2 \\ 1.00 & \text{si empate} \\ 0.99 & \text{si pierde por } 1 \\ 0.98 & \text{si pierde por } \geq 2 \end{cases}$$

La lógica subyacente refleja patrones observados en el fútbol profesional:

- **Victoria amplia (≥3 goles)**: La defensa del equipo que gana se vuelve ligeramente más permisiva, priorizando la gestión del partido sobre la intensidad. Los defensores evitan entradas arriesgadas que podrían resultar en tarjetas o lesiones innecesarias.
- **Victoria ajustada (1-2 goles)**: Paradójicamente, estos escenarios generan la mayor intensidad defensiva del equipo que gana. La necesidad de proteger una ventaja frágil resulta en presión más agresiva y organizada, reduciendo espacios y tiempo para el atacante.
- **Empate**: Representa el estado neutro donde ambos equipos mantienen su intensidad táctica estándar sin ajustes motivacionales.
- **Derrota por 1 gol**: El equipo que pierde necesita recuperar el balón para empatar, pero no puede permitirse una presión desorganizada que exponga aún más su defensa. La intensidad aumenta marginalmente pero de manera controlada.
- **Derrota amplia (≥2 goles)**: La defensa del equipo perdedor frecuentemente abandona la estructura compacta en busca del gol, creando espacios que facilitan ligeramente los regates. La desesperación táctica prevalece sobre la disciplina posicional.

2.4.3 Momento del Partido ($T_{momento}$)

El factor temporal reconoce que los minutos finales del partido alteran fundamentalmente la dinámica del juego:

$$T_{momento} = \begin{cases} 1.00 & \text{si } t < 80 \\ 0.98 & \text{si } t \geq 80 \end{cases}$$

Esta modelación aparentemente simple encapsula múltiples fenómenos que convergen en los minutos finales:

- **Fatiga acumulada generalizada**: Después de 80 minutos, tanto atacantes como defensores operan con capacidades físicas disminuidas. Sin embargo, la fatiga afecta más a las acciones explosivas como el regate que a la organización defensiva posicional.
- **Gestión del riesgo**: En minutos finales, los jugadores se vuelven más conservadores con acciones individuales arriesgadas. Un regate fallido que resulte en contraataque puede ser decisivo, llevando a una selección más cuidadosa de cuándo intentarlo.
- **Ritmo de juego reducido**: El tempo general del partido disminuye, con más interrupciones, substituciones tácticas y gestión del tiempo. Este contexto fragmentado reduce las oportunidades de regates en transición rápida donde las probabilidades de éxito son mayores.
- **Concentración defensiva aumentada**: Paradójicamente, aunque físicamente fatigados, los defensores a menudo muestran mayor concentración en minutos finales, conscientes de que un error puede ser irreversible.

La reducción del 2% captura el efecto neto de estos factores sin sobreponderar el impacto temporal, reconociendo que jugadores excepcionales pueden mantener su efectividad incluso en estados de fatiga avanzada.

2.5 Calibración y Validación

El modelo produce probabilidades en el rango [0.1%, 65%], deliberadamente calibrado para reflejar la realidad del fútbol profesional donde incluso los mejores regateadores en condiciones óptimas raramente superan el 70% de efectividad. La validación se realiza comparando las predicciones

xD con los resultados reales registrados en los datos de StatsBomb, permitiendo ajustes iterativos de los parámetros para optimizar la precisión predictiva.

Basándome en datos y análisis de fuentes como Opta ((Analyst, 2023), (Football, 2023)), CIES Football Observatory ((Observatory, 2019) (Observatory, 2018) (Observatory, 2024)) y StatMuse (enlaces dinámicos por temporada (StatMuse, 2025a) (StatMuse, 2025b) (StatMuse, 2024)) , se puede establecer una media general aproximada:

- La mayoría de los regateadores profesionales en las ligas top (Big 5 de Europa) tienen una tasa de éxito de regates que oscila aproximadamente entre el 45% y el 60%.

Algunos puntos clave y ejemplos que apoyan esto:

- Grandes regateadores: Jugadores como Lionel Messi o Eden Hazard históricamente han mantenido tasas de éxito notablemente altas, a menudo por encima del 55-60%, e incluso algunos registros antiguos de Hazard alcanzan el 75% cuando intentaba más de 100 regates. Otros grandes regateadores como Wilfried Zaha o Jeremy Doku suelen estar en el rango de 55-65%.
- Jugadores de otras posiciones: Un centrocampista como Sergio Busquets en un Mundial, por ejemplo, ha tenido una tasa de éxito altísima (91.67% en una muestra pequeña) porque sus regates son más conservadores y en zonas menos arriesgadas. Esto eleva la media si se incluyen todos los tipos de jugadores.
- Datos recientes: Las estadísticas de la temporada actual (2024/25) en ligas como la Premier League muestran a muchos de los mejores regateadores con porcentajes de éxito entre el 50% y el 65% (por ejemplo, Jeremy Doku con 64.07%, Lamine Yamal con 55.71%, o Cole Palmer con 52.58%).

3 Aplicación Práctica: De la Teoría a la Implementación

3.1 Introducción: Transformando Conceptos en Código

La verdadera prueba de cualquier modelo analítico en el fútbol no reside en su elegancia teórica, sino en su capacidad para generar valores accionables a partir de datos reales. En esta sección, transitamos desde la formulación matemática de xD hacia su implementación práctica, demostrando cómo los conceptos abstractos se traducen en herramientas concretas para el análisis del rendimiento.

Este proceso de implementación enfrenta desafíos que demostrarán tanto las fortalezas como las limitaciones del modelo. Los datos del mundo real son imperfectos per se: eventos faltantes, precisión variable en las coordenadas, y la necesidad de inferir información no directamente observable como la velocidad a partir de posiciones discretas. Además, la naturaleza de los datos de StatsBomb –eventos discretos en lugar de tracking continuo– requiere aproximaciones inteligentes para variables que puede que no tengamos.

Para facilitar la adopción y experimentación con xD, hemos desarrollado implementaciones en dos de los lenguajes más populares en el análisis deportivo: R y Python. Estas librerías no solo calculan el valor xD para cada intento de regate, sino que también proporcionan funciones auxiliares para la visualización, validación y análisis comparativo. La arquitectura modular permite a los usuarios ajustar parámetros, experimentar con formulaciones alternativas, y adaptar el modelo a contextos específicos o filosofías tácticas particulares. Cada ejemplo ha sido seleccionado para demostrar aspectos específicos del modelo: cómo diferentes contextos producen valores xD dramáticamente diferentes para acciones aparentemente similares, cómo el modelo captura la intuición táctica de entrenadores experimentados, y cómo puede revelar ideas no evidentes en análisis tradicionales.

Más allá de los cálculos individuales, exploramos cómo xD puede integrarse en flujos de trabajo analíticos más amplios. También soy consciente de que esta idea tendrá sus ajustes y fallos, por ello expronde elementos de mejora del modelo para iniciativas posteriores que refinen mejor la

formula, o los datos necesarios para su cálculo.

La implementación práctica de xD representa más que un ejercicio técnico; es una demostración de cómo el rigor analítico puede coexistir con la complejidad y belleza del fútbol. Al proporcionar herramientas accesibles y transparentes, espero no solo facilitar el análisis avanzado del regate, sino también inspirar futuras innovaciones en la cuantificación de acciones técnicas complejas. Los ejemplos que siguen son, tanto una validación del modelo como una invitación a la comunidad analítica para construir sobre estos fundamentos.

3.2 Obtención de datos y filtrado

Vamos a obtener los datos gratuitos de las 5 grandes ligas del año 2015/2016 liberados por StatsBomb con motivo de su décimo aniversario. Como 5 ligas son muchísimos eventos empezaré por analizar una única liga, la española:

```
devtools::install_github("statsbomb/StatsBombR")
library(StatsBombR)
```

Una vez instalada e inicializada la librería cargo los datos:

```
Comp <- FreeCompetitions() %>% filter(season_id==27 & competition_id == 11)
```

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be
```

```
Matches <- FreeMatches(Comp)
```

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be
```

```
allEvents <- free_allevents(MatchesDF = Matches, Parallel = T)
```

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be
```

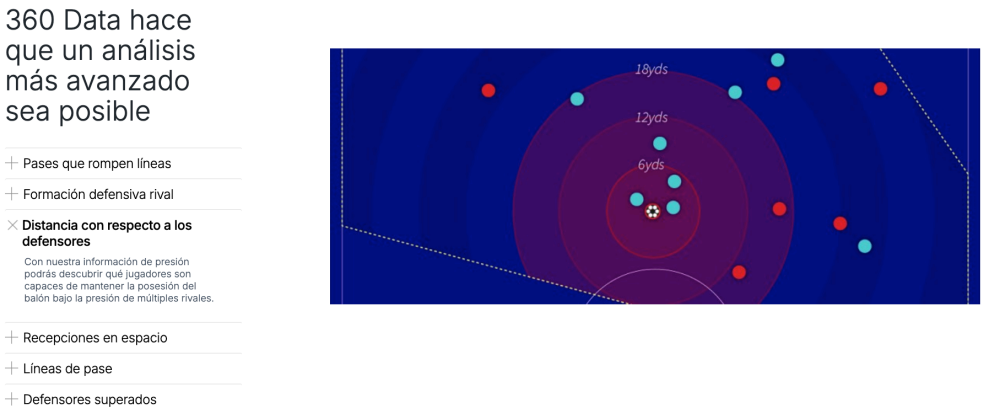
```
# Limpio los eventos
allEventsClean <- allclean(allEvents)
# Filtro por los eventos de tipo regate
dribbles <- allEventsClean %>% filter(type.name == "Dribble")
unique(dribbles$shot.freeze_frame)
```

```
## [[1]]
## NULL
```

Con este filtro vemos que no tenemos las posiciones de los jugadores alrededor del jugador que realiza el regate.

Para resolver este problema, he investigado si StatsBomb podría tener estos datos en algún otro repositorio ampliado, y con StatsBomb360 parece que tendríamos mas posicionamientos de jugadores cercanos a los eventos que queremos analizar. Segun se explica en esta la imagen 2 tenemos las distancias en yardas, pero puedo encontrar estos datos y usarlos para el cálculo de la nueva métrica.

Figure 2: Datos 360 StatsBomb, contexto regate



Una vez que he resuelto el problema del posicionamiento de los jugadores cercanos, tengo que encontrar algún repositorio gratuito de StatsBomb 360. Y utilizar sus propias funciones del repositorio de R (StatsBomb, 2024) para encontrarlo. Si analizamos la variable *Matches*, podemos

ver que existe una columna llamada *match_status_360*, la cual indica si el partido puede o no cargar los datos 360.

Gracias a esta columna podemos encontrar que partidos tienen los datos 360 abiertos:

```
# Paso 1: Obtener todas las competiciones gratuitas
Comp <- FreeCompetitions()
```

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be
```

```
# Paso 2: Obtener todos los partidos de esas competiciones gratuitas
Matches <- FreeMatches(Comp)
```

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be
```

```
# Paso 3: Identificar partidos con datos 360
matches_with_360_data <- Matches %>% filter(match_status_360 == "available")
# Paso 4: Carga del primer partido con 360
selected_match_df <- matches_with_360_data[1,]
paste(selected_match_df$home_team.home_team_name,
      " VS ",
      selected_match_df$away_team.away_team_name)
```

```
## [1] "Bayer Leverkusen  VS  Werder Bremen"
```

```
# Tambien podemos ver de que competiciones tenemos tambien datos 360
unique(matches_with_360_data$competition.competition_name)
```

```
## [1] "1. Bundesliga"       "FIFA World Cup"      "La Liga"
## [4] "Ligue 1"             "Major League Soccer" "UEFA Euro"
## [7] "UEFA Women's Euro"   "Women's World Cup"
```

Con estos datos ya tengo, al menos un partido con los datos 360 para poner a prueba mi formula del calculo del xD. Pero ahora cuando utilizaba la función de la librería de R para la carga de los eventos 360, esta no me funcionaba, por lo que he tenido que modificarla, creando mi propia función de carga, basada en la función de carga **StatsBombFree360Events**. Mi función solo puede cargar un partido porque es lo que estoy utilizando para validar la fórmula:

```
jaimeStatsBombFree360Events <- function (MatchesDF = "ALL", Parallel = T)
{
  events.df <- tibble()
  if (Parallel == T) {
    if (MatchesDF$match_id == "ALL") {
      Matches2 <- FreeMatches(FreeCompetitions())
      cl <- makeCluster(detectCores())
      registerDoParallel(cl)
      events.df <- foreach(i = 1:dim(Matches2)[1], .combine = bind_rows,
        .multicombine = TRUE, .errorhandling = "remove",
        .export = c("get.360matchFree"), .packages = c("httr",
          "jsonlite", "dplyr")) %dopar% {
        get.matchFree(Matches2[i, ])
      }
      stopCluster(cl)
    }
    else {
      cl <- makeCluster(detectCores())
      registerDoParallel(cl)
      events.df <- foreach(i = 1:dim(MatchesDF)[1], .combine = bind_rows,
        .multicombine = TRUE, .errorhandling = "remove",
        .export = c("get.360matchFree"), .packages = c("httr",
```

```
          "jsonlite", "dplyr")) %dopar% {
        get.360matchFree(MatchesDF[i, ])
      }
      stopCluster(cl)
    }
  }
  else {
    if (MatchesDF$match_id == "ALL") {
      Matches2 <- FreeMatches(FreeCompetitions())
      for (i in 1:length(Matches2$match_id)) {
        events <- get.360matchFree(Matches2[i, ])
        events.df <- bind_rows(events.df, events)
      }
    }
    else {
      for (i in 1:length(MatchesDF$match_id)) {
        events <- get.360matchFree(MatchesDF[i, ])
        events.df <- bind_rows(events.df, events)
      }
    }
  }
  return(events.df)
}



jaimeAllMatchesStatsBombFree360Events <- function(MatchesDF = "ALL", Parallel = T){
  events.df <- tibble()
```

```
# Verificar si MatchesDF es "ALL" (string) o un data frame
is_all_matches <- is.character(MatchesDF) && length(MatchesDF) == 1 && MatchesDF == "
if(Parallel == T){
  if(is_all_matches){
    Matches2 <- FreeMatches(FreeCompetitions())
    cl <- makeCluster(detectCores())
    registerDoParallel(cl)
    events.df <- foreach(i = 1:dim(Matches2)[1], .combine=bind_rows, .multicombine = T
                         .errorhandling = 'remove', .export = c("get.360matchFree"),
                         .packages = c("httr", "jsonlite", "dplyr")) %dopar%
      {get.360matchFree(Matches2[i,])}
    stopCluster(cl)
  } else {
    cl <- makeCluster(detectCores())
    registerDoParallel(cl)
    events.df <- foreach(i = 1:dim(MatchesDF)[1], .combine=bind_rows, .multicombine =
                         .errorhandling = 'remove', .export = c("get.360matchFree"),
                         .packages = c("httr", "jsonlite", "dplyr")) %dopar%
      {get.360matchFree(MatchesDF[i,])}
    stopCluster(cl)
  }
} else { #Begin Else, parallel == F
  if(is_all_matches){
    Matches2 <- FreeMatches(FreeCompetitions())
    for(i in 1:length(Matches2$match_id)){
      events <- get.360matchFree(Matches2[i,])
      events.df <- bind_rows(events.df, events)
```

```
      }
    } else {
      for(i in 1:length(MatchesDF$match_id)){
        events <- get.360matchFree(MatchesDF[i,])
        events.df <- bind_rows(events.df, events)
      }
    }
  }
  return(events.df)
}
```

Vamos a cargar los datos 360 del partido seleccionado:

```
match360 <- jaimeStatsBombFree360Events(selected_match_df)
```

Al tener ya los datos 360 del partido podemos filtrar los datos que sean regates y unirlos con los datos 360 para ver de que regates tenemos el frame del regate.

```
# Carga de eventos de ese partido
matchAllEvents <- free_allevents(MatchesDF = selected_match_df, Parallel = T)
```

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be
```

```
# Limpio los eventos
matchAllEventsClean <- allclean(matchAllEvents)
# Filtro por los eventos de tipo regate
matchDribbles <- matchAllEventsClean %>% filter(type.name == "Dribble")
# Creacion de df con las columnas necesarias del match360
events360_seleccionado <- match360 %>%
```

```
  select(
    event_uuid,                    # La columna clave para la unión
    freeze_frame,                  # freeze_frame con los jugadores
    visible_area                   # area visible, no creo que la necesite
  )
# Unión de los frames en los eventos de tipo regate
df_combinado <- left_join(
  matchDribbles,
  events360_seleccionado,
  by = c("id" = "event_uuid"))
# Limpieza de columnas que son vacías o nulas (por ejemplo, location del tiro)
df_combinado <- Filter(function(x) !all(sapply(x, is.null)), df_combinado)
df_combinado <- Filter(function(col) !all(is.na(col)), df_combinado)
```

3.3 xD automatizado con StatsBomb360

Al llegar a este apartado, he desarrollado la librería en R y en Python que se encargaran de calcular la formula del xD. Se pueden consultar en detalle en el Apéndice A.2 (librería de R) y el Apéndice A.3 (librería de Python), la implementación del cálculo. En este apartado voy a mostrar como usar esta librería para ver el resultado de aplicarlas.

Esto nos permite:

1. **Validación estadística robusta**: Comparar predicciones con resultados reales
2. **Análisis por zonas y contextos**: Identificar patrones y anomalías
3. **Calibración del modelo**: Ajustar parámetros basándose en datos masivos

Cargando la libreria *xDCalculator* podemos añadir la colimna xD automaticamente al dataframe de regates.

```
source("xDCalculator.R")
library(ggplot2)
library(dplyr)
library(tidyr)
library(kableExtra)
library(tibble)


# Elimino las filas que tienes freeze_frame nulo
df_combinado_limpio <- df_combinado[
                      !sapply(df_combinado$freeze_frame, is.null), ]
misXDcalculados <- c()
for (i in 1:nrow(df_combinado_limpio)) {
  evento_actual <- df_combinado_limpio$freeze_frame[[i]]
  if (!is.null(evento_actual )){
  # Extraccion de coordenadas "c(x, y)"
  if ("location" %in% names(evento_actual)) {
      evento_actual$location.x <- sapply(evento_actual$location,
                                         function(loc) loc[1])
      evento_actual$location.y <- sapply(evento_actual$location,
                                         function(loc) loc[2])
  }
  #Tomamos valor por defecto:
  # - La velocidad
  # - El angulo a portería
  # - Que ambos jugadores llevan todo el partido jugando
  # - No hay diferencia de goles
  xdResultado = calculate_xd(freeze_frame =  evento_actual,
```

```
                              minute = df_combinado_limpio$minute[[i]],
                attacker_minutes_played = df_combinado_limpio$minute[[i]],
                defender_minutes_played = df_combinado_limpio$minute[[i]],
                 goal_difference = 0)
  misXDcalculados <- append(misXDcalculados, xdResultado$xD)
  }
}
df_combinado_limpio$xD <- misXDcalculados
df_combinado_limpio <- df_combinado_limpio %>%
                 mutate(
                   post.xD = if_else(dribble.outcome.name == "Complete", 1, 0))


df_combinado_limpio %>%
  select(
    Equipo = possession_team.name,
    Jugador = player.name,
    Minuto = minute,
    xD,
    post.xD
  ) %>%
  kable("pipe",
        caption = "xD de cada regate del partido (del que tengo datos)") %>%
  kable_styling(bootstrap_options = "striped", full_width = FALSE)
```

Table 1: xD de cada regate del partido (del que tengo datos)

Equipo	Jugador	Minuto	xD	post.xD
Werder Bremen	Romano Schmid	2	0.0750470	0

Bayer Leverkusen	Robert Andrich	18	0.0206902	1
Werder Bremen	Nick Woltemade	25	0.0708494	1
Werder Bremen	Nick Woltemade	25	0.0762477	0
Bayer Leverkusen	Amine Adli	34	0.2086100	1
Bayer Leverkusen	Jonas Hofmann	34	0.1992615	1
Werder Bremen	Amine Adli	45	0.2291935	0
Werder Bremen	Christian Groß	45	0.2096051	1
Bayer Leverkusen	Nathan Tella	47	0.0396148	0
Werder Bremen	Nick Woltemade	47	0.0263643	0
Bayer Leverkusen	Nathan Tella	55	0.0957961	0
Bayer Leverkusen	Victor Okoh Boniface	59	0.1975413	1
Bayer Leverkusen	Florian Wirtz	64	0.1795493	1
Bayer Leverkusen	Jeremie Frimpong	66	0.0187943	0
Werder Bremen	Jeremie Frimpong	71	0.2177355	1
Bayer Leverkusen	Patrik Schick	72	0.0409529	0
Werder Bremen	Florian Wirtz	74	0.1707851	0
Werder Bremen	Isak Hansen-Aarøen	78	0.0222886	0
Bayer Leverkusen	Florian Wirtz	79	0.0669445	0
Werder Bremen	Romano Schmid	85	0.0795560	0
Bayer Leverkusen	Exequiel Alejandro Palacios	86	0.0985239	1
Werder Bremen	Romano Schmid	87	0.0600018	1
Werder Bremen	Mitchell Weiser	89	0.2362763	1

Ahora voy a centrarme en un único regate, que genera el segundo gol del Bayer Leverkusen, en el minuto 59 por Victor Okoh Boniface.

Este es el frozen frame que hemos obtenido, y podemos ver que coincide con el del vídeo:

3.4 xD manual sobre video

Complementamos con análisis detallado de regates memorables:

Ejemplo: Messi vs Boateng (Barcelona vs Bayern, 2015)

Creamos manualmente el DataFrame de posiciones y realizamos el cálculo en Python:

```
# Datos extraídos del video frame a frame
boateng_dribble_df = pd.DataFrame({
    'teammate': [False, False, False, True, False, True],
    'actor': [False, False, False, False, False, False],
    'keeper': [False, False, False, False, False, False],
    'location': [
```

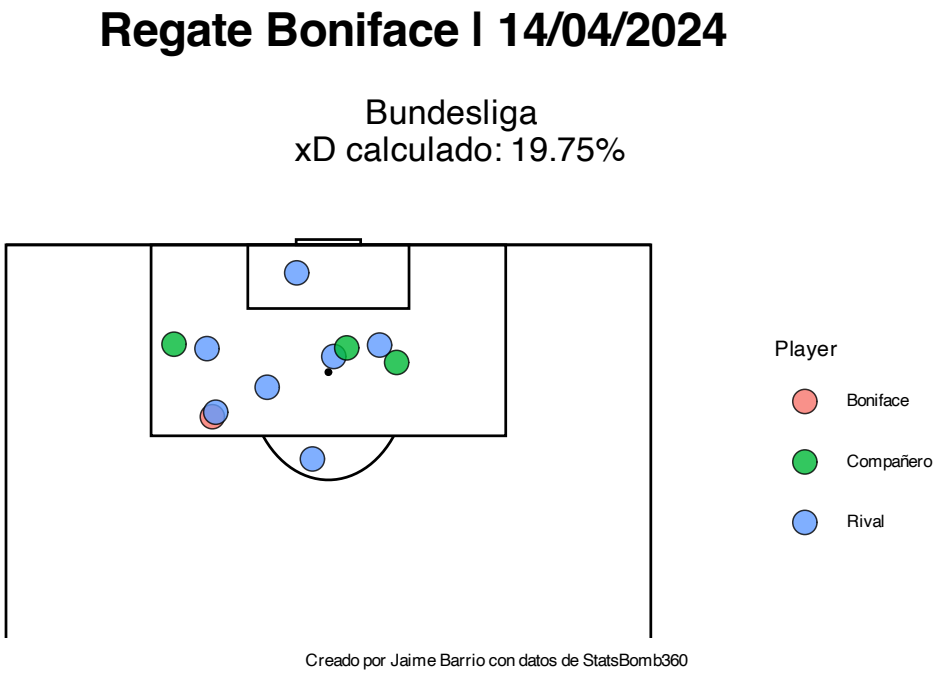


Figure 3: Regate Boniface | 14/04/2024

Figure 4: Regate Boniface | 14/04/2024 | Video Real

Figure 5: Messi-Boateng (Barca Vs Bayern) - 1

Figure 6: Messi-Boateng (Barca Vs Bayern) - 2

```
            [103, 26.5],     # Boateng (defensor principal)
            [99, 34],      # Benatia (cobertura)
            [95, 24],      # Juan Bernat(segunda línea)
            [94, 37],      # Luis Suarez  (apoyo)
            [98.5, 46.2],  # Rafinha (tapando pase)
            [98.4, 49.7],  # Neymar (opción de pase)
        ]
    })

    # Ubicación de Messi
    messi_location = [102, 24]

    # Procesar para xD

    # Minuto 77, Barcelona 0-0 Bayern (ida de semifinales Champions)
    result = calculator.calculate_xd(
       x=102,
       y=24,
       velocity=22,  # Messi en velocidad alta según el video
       angle=-15,    # Hacia portería)
       teammates_close=xd_inputs['teammates_close'],
       teammates_medium=xd_inputs['teammates_medium'],
       teammates_far=xd_inputs['teammates_far'],
       closest_defender_distance=xd_inputs['closest_defender_distance'],
       defenders_very_close=xd_inputs['defenders_very_close'],
       defenders_close=xd_inputs['defenders_close'],
       minute=77,
       attacker_minutes_played=77,
       defender_minutes_played=77,  # Boateng jugó todo el partido
       goal_difference=0
    )

Resultado real: Messi supera a Boateng (que cae) y marca
xD calculado: 28.3%
```

Visualizacion de las posiciones utilizadas para el cálculo:

Utilizando el DataFrame de posiciones de pintado y realizamos el cálculo en R:

```r
source("xDCalculator.R")

result = calculate_xd(velocity=22, angle=-15, freeze_frame=regateMessi, minute = 77,
                      attacker_minutes_played = 77, defender_minutes_played = 77,
                      goal_difference = 0)
```

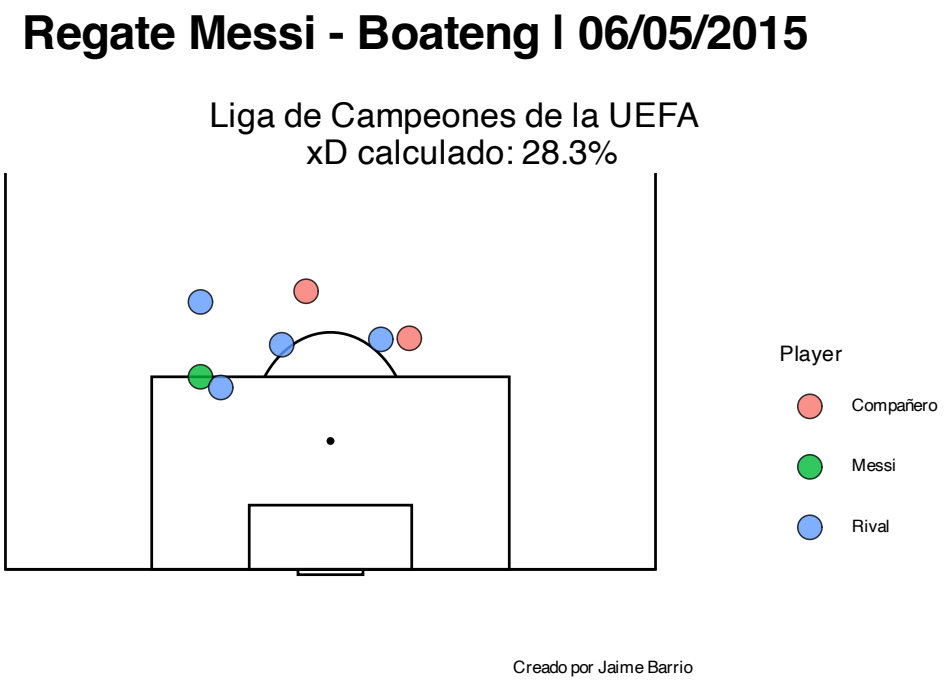


Figure 7: Regate Messi - Boateng | 06/05/2015

Table 2: Resultados Detallados del Cálculo de xD

Métrica	Valor
xD	0.2827442
xD_percentage	28.3%
location.x	102
location.y	24
components.attack	0.4661413
components.defense	0.4108747
components.context	1.0296
factors.z_campo	1.102481
factors.velocity	0.6833333
factors.teammates	1
factors.direction	0.6187482
factors.pressure	0.5135934
factors.coverage	0.8
factors.physical	1
distances.teammates_close	0
distances.teammates_medium	0
distances.teammates_far	0
distances.closest_defender_distance	2.692582
distances.defenders_very_close	0

distances.defenders_close	1

3.5 xD automatizado en el Mudial 2022 con StatsBomb360

Ahora voy a dar un paso mas en la automatización del cálculo. Voy a calcular el xD de una competición completa, para posteriormente encontrar al jugador que completó mas regates exitosos con un xD menor del 10%. También buscaré al jugador que intentó mas regates con menos del 10% de xD, sin éxito, para identificar un jugador que tomó, a priori, malas decisiones. Y por último al jugador que con un xD mayor de 15% falló mas regates.

3.5.1 Obtención y procesamiento de datos

```
worldCup22Matches <- matches_with_360_data %>% filter(competition.competition_id == 43)
WCallEvents <- free_allevents(MatchesDF = worldCup22Matches, Parallel = T)
```

```
## [1] "Whilst we are keen to share data and facilitate research, we also urge you to be
```

```
# Limpio los eventos
WCallEventsClean <- allclean(WCallEvents)
# Filtro por los eventos de tipo regate
WCdribbles <- WCallEventsClean %>% filter(type.name == "Dribble")
# Carga datos 360 del mundial
WCMatches360Events <- jaimeAllMatchesStatsBombFree360Events(worldCup22Matches)
# Creacion de df con las columnas necesarias del match360
WCevents360_seleccionado <- WCMatches360Events %>%
  select(
    event_uuid,                 # La columna clave para la unión
    freeze_frame,               # freeze_frame con los jugadores
    visible_area                # area visible, no creo que la necesite
```

```
  )
# Unión de los frames en los eventos de tipo regate
df_wc_combinado <- left_join(
  WCdribbles,
  WCevents360_seleccionado,
  by = c("id" = "event_uuid"))
# Limpieza de columnas que son vacías o nulas (por ejemplo, location del tiro)
df_wc_combinado <- Filter(function(x) !all(sapply(x, is.null)), df_wc_combinado)
df_wc_combinado <- Filter(function(col) !all(is.na(col)), df_wc_combinado)
# Elimino las filas que tienes freeze_frame nulo
df_wc_combinado <- df_wc_combinado[
                      !sapply(df_wc_combinado$freeze_frame, is.null), ]
```

3.5.2 Cálculo de xD en Mundial 2022

Ahora con los datos ya preparados, aplico la función del cálculo del xD a todos los regates que se produjeron en el mundial.

```
wcXDcalculados <- c()
for (i in 1:nrow(df_wc_combinado)) {
  evento_actual <- df_wc_combinado$freeze_frame[[i]]
  if (!is.null(evento_actual)){
  # Extraccion de coordenadas "c(x, y)"
  if ("location" %in% names(evento_actual)) {
      evento_actual$location.x <- sapply(evento_actual$location,
                                         function(loc) loc[1])
      evento_actual$location.y <- sapply(evento_actual$location,
                                         function(loc) loc[2])
```

```
  }
  # Tomamos valor por defecto, porque no los tengo:
  # - La velocidad
  # - El angulo a portería
  # - Que ambos jugadores llevan todo el partido jugando
  # - No hay diferencia de goles
  xdResultado = calculate_xd(freeze_frame =  evento_actual,
                             minute = df_wc_combinado$minute[[i]],
                attacker_minutes_played = df_wc_combinado$minute[[i]],
                defender_minutes_played = df_wc_combinado$minute[[i]],
                 goal_difference = 0)
  wcXDcalculados <- append(wcXDcalculados, xdResultado$xD)
  }
}
df_wc_combinado$xD <- wcXDcalculados
df_wc_combinado <- df_wc_combinado %>%
                 mutate(
                   post.xD = if_else(dribble.outcome.name == "Complete", 1, 0))


tablaWC <- df_wc_combinado %>%
  select(
    Equipo = possession_team.name,
    Jugador = player.name,
    Minuto = minute,
    xD,
    post.xD
  )
```

```
head(tablaWC,10) %>%
  kable("pipe",
        caption = "xD de cada regate del mundial") %>%
  kable_styling(bootstrap_options = "striped", full_width = FALSE)
```

Table 3: xD de cada regate del mundial

Equipo	Jugador	Minuto	xD	post.xD
Denmark	Christian Dannemann Eriksen	15	0.0609751	0
Tunisia	Aïssa Bilal Laïdouni	16	0.1115340	1
Tunisia	Mohamed Dräger	17	0.1600219	0
Denmark	Christian Dannemann Eriksen	20	0.2790358	1
Denmark	Issam Jebali	20	0.1175574	0
Tunisia	Youssef Msakni	31	0.0611944	1
Tunisia	Mohamed Dräger	31	0.0271200	0
Denmark	Joakim Mæhle	35	0.0236014	0
Tunisia	Aïssa Bilal Laïdouni	43	0.0724980	0
Tunisia	Youssef Msakni	45	0.0653920	1

Voy a crear ahora una tabla para porcentajes de regates difíciles sobre el total de regates de cada jugador:

```
tablaWCPorcentajes <- tablaWC %>%
  group_by(Jugador) %>%  # Cambia por el nombre real de tu columna de jugador
  summarise(
    # Total de regates del jugador
    regatesTotales = n(),
    # Cuántas veces xD > 0.15 (15%)
    xD_mayor_15 = sum(xD > 0.15, na.rm = TRUE),
    # De los xD > 15%, cuántos fueron fallidos (post.xD == 0)
```

```
    xD_mayor_15_fallidos = sum(xD > 0.15 & post.xD == 0, na.rm = TRUE),
    # Cuántas veces xD < 0.10 (10%)
    xD_menor_10 = sum(xD < 0.10, na.rm = TRUE),
    # De los xD < 10%, cuántos fueron exitosos (post.xD == 1)
    xD_menor_10_exitosos = sum(xD < 0.10 & post.xD == 1, na.rm = TRUE),
    # De los xD < 10%, cuántos fueron fallidos (post.xD == 0)
    xD_menor_10_fallidos = sum(xD < 0.10 & post.xD == 0, na.rm = TRUE),
    # Columnas adicionales útiles
    porcentaje_xD_menor_10pct = round((xD_menor_10 / regatesTotales) * 100, 2),
    porcentaje_exito_xD_menor_10pct =
      ifelse(xD_menor_10 == 0,
             0,
             round((xD_menor_10_exitosos / xD_menor_10) * 100, 2)),
     porcentaje_xD_mayor_15_fallidospct =
      ifelse(xD_mayor_15 == 0,
             0,
             round((xD_mayor_15_fallidos / xD_mayor_15) * 100, 2)),
    .groups = 'drop'
  )
head(tablaWCPorcentajes,10) %>%
  arrange(desc(regatesTotales)) %>%
  kable("latex",
        caption = "Porcentaje de regates difíciles") %>%
  kable_styling(bootstrap_options = "striped", full_width = FALSE,
                latex_options = "scale_down")
```

Table 4: Porcentaje de regates difíciles

Jugador	regatesTotales	xD_mayor_15	xD_mayor_15_fallidos	xD_menor_10	xD_menor_10_exitosos	xD_menor_10_fallidos	porcentaje_xD_menor_10pct	porcentaje_exito_xD_menor_10pct	porcentaje_xD_mayor_15_fallidospct
Aaron Ramsey	9	2	0	7	1	6	77.78	14.29	0
Aaron Mooy	5	1	1	4	2	2	80.00	50.00	100
Abdul Rahman Baba	4	1	0	2	0	2	50.00	0.00	0
Abdelkarim Hassan Al Haj Fadlalla	3	0	0	3	2	1	100.00	66.67	0
Abdulrahman Al-Obood	2	0	0	1	0	1	50.00	0.00	0
Abdelhamid Sabiri	1	0	0	1	0	1	100.00	0.00	0
Abderrazak Hamdallah	1	1	0	0	0	0	0.00	0.00	0
Abdessamad Ezzalzouli	1	1	0	0	0	0	0.00	0.00	0
Abdul Fatawu Issahaku	1	0	0	1	0	1	100.00	0.00	0
Abdulaziz Hatem Mohammed Abdullah	1	0	0	1	1	0	100.00	100.00	0

Table 5: Jugadores más habilidosos - Ordenado por cantidad de regates difíciles intentados

Jugador	Regates Totales	Regates Difíciles	Regates con xD < 10% Éxito	%Regates con xD < 10%
Mateo Kovačić	14	9	75 %	85.71 %
Azzedine Ounahi	11	8	88.89 %	81.82 %
Lionel Andrés Messi Cuccittini	30	8	47.06 %	56.67 %
Alphonso Davies	19	7	53.85 %	68.42 %
Jamal Musiala	30	5	35.71 %	46.67 %
Mohammed Kudus	11	5	83.33 %	54.55 %
Sofiane Boufal	19	5	38.46 %	68.42 %
Francisco Javier Calvo Quesada	4	4	100 %	100 %
Gonzalo Jordy Plata Jiménez	8	4	66.67 %	75 %
Luka Modrić	10	4	50 %	80 %

3.5.3 Presentación de resultados - Mundial 2022

Ahora vamos a mostrar el ranking de los jugadores que mencionamos al principio de este apartado:

1. Jugador con más regates exitosos con un xD menor del 10%
2. Jugador con más regates fallados con un xD menor del 10%

```
jugadoresMasPresionados <- tablaWC %>%
  filter(
    xD <= 0.10,
    post.xD == 0
  ) %>%
  group_by(Jugador) %>%
  summarise(
    total_xD = sum(xD, na.rm = TRUE),
    total_post.xD = sum(post.xD, na.rm = TRUE),
    intentos = n(),
```

```
    .groups = 'drop'
  )



jugadoresMasPresionados <- left_join(jugadoresMasPresionados,
                                     tablaWCPorcentajes,
                                     by="Jugador") %>%
                           arrange(desc(xD_menor_10))
jugadoresMasPresionados <- jugadoresMasPresionados %>%
                          mutate(
                            total_xD = round(total_xD,3),
                            porcentaje_xD_menor_10pct =
                              paste0(round(porcentaje_xD_menor_10pct,2), " %"),
                            porcentaje_exito_xD_menor_10pct =
                              paste0(100-round(porcentaje_exito_xD_menor_10pct,2), " %"),
                          ) %>%
                          rename(
                             "Regates Totales" = regatesTotales,
                             "Regates Difíciles" = xD_menor_10,
                             "%Regates con xD < 10%" = porcentaje_xD_menor_10pct,
                             "Regates con xD < 10% Fallidos" =
                               porcentaje_exito_xD_menor_10pct
                           )



head(jugadoresMasPresionados,10) %>%
  select(
```

Table 6: Jugadores con mas regates difíciles fallidos - Ordenados por cantidad de regates difíciles intentados

Jugador	Regates Totales	Regates Difíciles	%Regates con xD < 10%	Regates con xD < 10% Fallidos
Kylian Mbappé Lottin	40	17	42.5 %	82.35 %
Lionel Andrés Messi Cuccittini	30	17	56.67 %	52.94 %
Jamal Musiala	30	14	46.67 %	64.29 %
Alphonso Davies	19	13	68.42 %	46.15 %
Neymar da Silva Santos Junior	19	13	68.42 %	69.23 %
Sofiane Boufal	19	13	68.42 %	61.54 %
Mateo Kovačić	14	12	85.71 %	25 %
Ivan Perišić	16	11	68.75 %	81.82 %
Daniel Olmo Carvajal	12	10	83.33 %	90 %
Mathew Leckie	13	10	76.92 %	100 %

```
    Jugador,
    `Regates Totales`,
    `Regates Difíciles`,
    `%Regates con xD < 10%`,
    `Regates con xD < 10% Fallidos`,
) %>%
kable("latex",
      caption = "Jugadores con mas regates difíciles fallidos
      - Ordenados por cantidad de regates difíciles intentados") %>%
kable_styling(bootstrap_options = "striped", full_width = FALSE,
              latex_options = "scale_down")
```

3. Jugador con más regates fallados con un xD mayor del 15%

Table 7: Jugadores con mas intentos fáciles fallidos

Jugador	% Regates con xD > 15% fallados	Regates con xD > 15%	Regates Totales
Gareth Frank Bale	100.00	2	3
Leandro Trossard	100.00	2	3
Neco Williams	100.00	2	3
Sang-Ho Na	100.00	2	4
Mehdi Taremi	66.67	3	5
Rodrygo Silva de Goes	66.67	3	7
Ali Abdi	50.00	2	2
Ali Gholizadeh	50.00	2	5
Alphonso Davies	50.00	2	19
Antoine Griezmann	50.00	2	8

4 Lineas futuras

4.1 Velocidad del portador

Al no tener el dato real en el eventing de la velocidad actual del regateador se estima basándose en eventos discretos, o utilizando la velocidad por defecto. Una mejora significativa sería:

- Tracking data: Utilizar datos de seguimiento para obtener velocidad instantánea real
- Aceleración: Incluir la aceleración/desaceleración en el momento del regate
- Velocidad relativa: Considerar la velocidad del defensor más cercano para calcular velocidades relativas

4.2 Contexto táctico

- Transición ofensiva: Mayor espacio disponible → xD más alto
- Ataque posicional: Menor espacio → xD más bajo
- Contraataque: Velocidad como factor crucial

4.3 Posición del Balón

Si conociéramos la posición del balón, podríamos tener en cuenta si el esférico está expuesto al rival. Si el regate se está intentando de espaldas, o cara a cara. Podriamos clasificar el regate y ajustar cuales son mas exitosos:

- Regate frontal: Mayor control visual, xD más alto
- Regate de espaldas: Mayor dificultad técnica, xD más bajo
- Protección del balón: Si el balón está expuesto o protegido
- Ángulo de aproximación: Dirección desde la que llega el defensor

4.4 Matriz de posiciones

Una mejora sería generar una matriz de regates, con todos los regates completados en todas las competiciones, comparando los completados y los no completados para así tener un mapa de calor de donde se realizan mas regates con éxito. Calibrar la matriz automáticamente con modelos de aprendizaje automático o machine learning, que activamente vayan nutriendo con retroalimentación el algoritmo.

4.5 Diferencia Goles

Automatizar el cálculo del resultado en el minuto de juego en el que se realiza el regate. Con el eventing de StatsBomb tenemos los eventos de goles, por lo tanto podría implementar otra función que se encargue de comparar el minuto del regate con el resultado actual del partido.

4.6 ***Evaluar defensas***

Si un defensa ante una situación desfavorable es superado o viceversa, también se puede utilizar para obtener otra métrica de evaluación de jugadores. Por ejemplo, un xDD (Expected Defensive Dribbling)

5 Conclusiones

El modelo Expected Dribbling (xD) representa un potencial avance en la analítica del fútbol al proporcionar una métrica objetiva y contextual para evaluar la dificultad y probabilidad de éxito de los regates.

5.1 Principales aportes

Cuantificación objetiva Transforma una habilidad tradicionalmente subjetiva en una métrica contextualizada, medible y comparable.

Contexto integral: Considera múltiples factores simultáneamente:

- Posición en el campo
- Velocidad del jugador
- Presión defensiva
- Apoyo de compañeros
- Fatiga acumulada
- Contexto del partido

Aplicabilidad práctica:

- Entrenadores: Identificar situaciones favorables para intentar regates
- Jugador: Evaluación de las decisiones tomadas, y su resultado e impacto en el equipo.
- Scouts: Evaluar jugadores de manera mas objetiva
- Analistas: Comprender patrones de juego

Base para futuras investigaciones: Formula extensible para incorporar nuevas variables y metodologías. Esta métrica puede ser solo la primera piedra de un calculo mas preciso con el que descubrir jugadores que hacen del regate su manera de juego.

5.2 Limitaciones actuales

- **Calidad de datos:** Dependencia de la precisión de los datos de eventos. Esto es algo que en cualquier métrica que utilizamos nos va a influir.
- **Factores psicológicos:** No considera presión psicológica, confianza o momentum. Intenta contextualizar el regate respecto al partido, pero no respecto al histórico de regates. Si un jugador ha fallado los 3 últimos regates, tal vez el cuarto tampoco salga por un factor psicológico y de confianza.
- **Generalización:** Modelo entrenado con datos específicos puede no generalizar perfectamente.

6 Copywright

6.1 Términos de uso

6.2 Cita recomendada

Para citar este trabajo en publicaciones académicas o profesionales:

Barrio, J. (2025). "Expected Dribbling (xD): Un modelo probabilístico para evaluar regates en el fútbol". [UCAM/Sports Data Campus]. Disponible en: GitHub

6.3 Declaración de originalidad

Este modelo ha sido desarrollado de manera independiente y representa una contribución original al campo de la analítica deportiva. Cualquier similitud con trabajos existentes es coincidencia.

A Apendices

A.1 Repositorio GitHub

https://github.com/jbarriojim/xD_Expected-Dribbling

A.2 LIBRERIA R

Listing 1: xD en R

aboveskip

```r
#' Calculate expected dribbling success (xD)
#'
#' @param velocity Attacker velocity in km/h
#' @param angle Movement direction relative to goal (degrees)
#' @param freeze_frame DataFrame with player positions (teammate, actor,
    keeper, location)
#' @param minute Current match minute
#' @param attacker_minutes_played Minutes played by attacker
#' @param defender_minutes_played Minutes played by defender
#' @param goal_difference Current goal difference
#' @return List with xD value and component breakdowns
calculate_xd <- function(velocity=15, angle=15, freeze_frame,
                         minute = 45,
                         attacker_minutes_played = 45,
                         defender_minutes_played = 45,
                         goal_difference = 0) {

  # Process freeze frame to get distance variables and dribbler location
  distance_vars <- process_freeze_frame(freeze_frame)

  # Extract coordinates
  x <- distance_vars$dribbler_location[1]
  y <- distance_vars$dribbler_location[2]

  # Extract processed variables
  teammates_close <- distance_vars$teammates_close
  teammates_medium <- distance_vars$teammates_medium
  teammates_far <- distance_vars$teammates_far
  closest_defender_distance <- distance_vars$closest_defender_distance
  defenders_very_close <- distance_vars$defenders_very_close
  defenders_close <- distance_vars$defenders_close

  # Calculate Attack Component (A)

  # Z_campo: Field value
  if (x < 60) {
    z_campo <- 0.09 * exp(0.027 * x)
  } else {
    z_campo <- 0.09 * exp(0.021 * x) + 0.008 * (x - 60)
  }
```

```r

  # V: Velocity factor
  v_norm <- velocity / 36
  V <- 0.5 + 0.3 * v_norm

  # C: Teammate support
  c_close <- 1 + 0.06 * teammates_close
  c_medium <- 1 + 0.03 * teammates_medium
  c_far <- 1 + 0.015 * teammates_far
  C <- c_close * c_medium * c_far

  # M_d: Movement direction
  angle_rad <- angle * pi / 180
  center_y <- 40
  y_factor <- 1 - (abs(y - center_y) / 40) * 0.3
  M_d <- 0.725 - 0.125 * cos(angle_rad) * y_factor

  # Total Attack
  A <- z_campo * V * C * M_d

  # Calculate Defense Component (D)

  # D_presion: Defensive pressure
  d_dist_norm <- closest_defender_distance / 10
  D_presion <- 0.3 + 0.4 * (1 - d_dist_norm)^2

  # D_cobertura: Defensive coverage
  D_cobertura <- 0.7 + 0.15 * defenders_very_close + 0.10 * defenders_close

  # M: Physical mismatch
  fatigue_att <- 1 - (attacker_minutes_played / 90) * 0.15
  fatigue_def <- 1 - (defender_minutes_played / 90) * 0.15
  M <- 0.33 + 0.67 * (fatigue_att / fatigue_def)

  # Total Defense
  D <- D_presion * D_cobertura * M

  # Calculate Context Component (X)

  # Time pressure
  if (minute <= 75) {
    time_factor <- 1.0
  } else {
    time_factor <- 1.0 + 0.02 * (minute - 75)
  }

  # Score pressure
  if (goal_difference == 0) {
    score_factor <- 1.0
  } else if (goal_difference > 0) {
    score_factor <- 0.95
  } else {
    score_factor <- 1.05
  }

  X <- 0.99 * time_factor * score_factor

```

```r
  # Final xD calculation
  xD <- A * (1 - D) * X

  # Ensure xD is between 0 and 1
  xD <- max(0, min(1, xD))

  # Return results
  list(
    xD = xD,
    xD_percentage = paste0(round(xD * 100, 1), "%"),
    location = list(x = x, y = y),
    components = list(
      attack = A,
      defense = D,
      context = X
    ),
    factors = list(
      z_campo = z_campo,
      velocity = V,
      teammates = C,
      direction = M_d,
      pressure = D_presion,
      coverage = D_cobertura,
      physical = M
    ),
    distances = list(
      teammates_close = teammates_close,
      teammates_medium = teammates_medium,
      teammates_far = teammates_far,
      closest_defender_distance = closest_defender_distance,
      defenders_very_close = defenders_very_close,
      defenders_close = defenders_close
    )
  )
}

#' Process freeze frame to extract distance-based variables
#'
#' @param freeze_frame DataFrame with columns teammate, actor, keeper, location
#' @return List with all calculated distances and counts, including dribbler location
process_freeze_frame <- function(freeze_frame) {

  # Find the actor (dribbler)
  actor_row <- freeze_frame[freeze_frame$actor == TRUE, ]
  if (nrow(actor_row) == 0) {
    stop("No actor found in freeze frame")
  }

  dribbler_x <- actor_row$location.x[1]
  dribbler_y <- actor_row$location.y[1]
  dribbler_location <- c(dribbler_x, dribbler_y)

  # Filter out only the dribbler
  players <- freeze_frame[!freeze_frame$actor, ]

```

```
  # Calculate distances
  players$distance <- sqrt(
    (players$location.x - dribbler_x)^2 +
      (players$location.y - dribbler_y)^2
  )

  # Separate teammates and opponents
  teammates <- players[players$teammate == TRUE, ]
  opponents <- players[players$teammate == FALSE, ]

  # Calculate teammate counts
  teammates_close <- sum(teammates$distance < 3)
  teammates_medium <- sum(teammates$distance >= 3 & teammates$distance < 5)
  teammates_far <- sum(teammates$distance >= 5 & teammates$distance < 10)

  # Calculate defender counts
  defenders_very_close <- sum(opponents$distance < 2)
  defenders_close <- sum(opponents$distance >= 2 & opponents$distance < 5)

  # Find closest defender distance
  if (nrow(opponents) > 0) {
    closest_defender_distance <- min(opponents$distance)
  } else {
    closest_defender_distance <- 10
  }

  list(
    dribbler_location = dribbler_location,
    teammates_close = teammates_close,
    teammates_medium = teammates_medium,
    teammates_far = teammates_far,
    closest_defender_distance = closest_defender_distance,
    defenders_very_close = defenders_very_close,
    defenders_close = defenders_close
  )
}
```

A.3 LIBRERIA Python

Listing 2: xD en Python

aboveskip

```
import numpy as np
import pandas as pd

# Definir la clase xDCalculator
class xDCalculator:
  def __init__(self):
    pass

  def _process_freeze_frame(self, freeze_frame):
    # Separate coordinates if location is a list/array
    if 'location' in freeze_frame.columns:
```

```
    freeze_frame = freeze_frame.copy()
    freeze_frame['x'] = freeze_frame['location'].apply(
      lambda loc: loc[0] if isinstance(loc, (list, np.ndarray)) else loc
    )
    freeze_frame['y'] = freeze_frame['location'].apply(
      lambda loc: loc[1] if isinstance(loc, (list, np.ndarray)) else loc
    )

  # Find the actor (dribbler)
  actor_row = freeze_frame[freeze_frame['actor'] == True]
  if len(actor_row) == 0:
    raise ValueError("No actor found in freeze frame")

  dribbler_x = actor_row.iloc[0]['x']
  dribbler_y = actor_row.iloc[0]['y']
  dribbler_location = (dribbler_x, dribbler_y)

  # Filter out only the dribbler himself
  players = freeze_frame[~freeze_frame['actor']].copy()

  # Calculate distances
  players['distance'] = players.apply(
    lambda row: np.sqrt(
      (row['x'] - dribbler_x)**2 +
      (row['y'] - dribbler_y)**2
    ),
    axis=1
  )

  # Separate teammates and opponents
  teammates = players[players['teammate'] == True]
  opponents = players[players['teammate'] == False]

  # Calculate teammate counts by distance
  teammates_close = len(teammates[teammates['distance'] < 3])
  teammates_medium = len(teammates[(teammates['distance'] >= 3) &
                                   (teammates['distance'] < 5)])
  teammates_far = len(teammates[(teammates['distance'] >= 5) &
                                (teammates['distance'] < 10)])

  # Calculate defender counts by distance
  defenders_very_close = len(opponents[opponents['distance'] < 2])
  defenders_close = len(opponents[(opponents['distance'] >= 2) &
                                  (opponents['distance'] < 5)])

  # Find closest defender
  if len(opponents) > 0:
    closest_defender_distance = opponents['distance'].min()
  else:
    closest_defender_distance = 10  # Default if no defenders

  return {
    'dribbler_location': dribbler_location,
    'teammates_close': teammates_close,
    'teammates_medium': teammates_medium,
    'teammates_far': teammates_far,
    'closest_defender_distance': closest_defender_distance,
```

```
      'defenders_very_close': defenders_very_close,
      'defenders_close': defenders_close
    }

  def calculate_xd(self, velocity, angle, freeze_frame, minute=45,
                   attacker_minutes_played=45, defender_minutes_played=45,
                   goal_difference=0):
    # Process freeze frame to get distance variables and dribbler location
    distance_vars = self._process_freeze_frame(freeze_frame)
    x, y = distance_vars['dribbler_location']

    # Calculate Attack Component (A)

    # Z_campo: Field value
    if x < 60:  # Own half
      z_campo = 0.09 * np.exp(0.027 * x)
    else:  # Opponent half
      z_campo = 0.09 * np.exp(0.021 * x) + 0.008 * (x - 60)

    # V: Velocity factor
    v_norm = velocity / 36  # Max speed ~36 km/h
    V = 0.5 + 0.3 * v_norm

    # C: Teammate support
    c_close = 1 + 0.06 * distance_vars['teammates_close']
    c_medium = 1 + 0.03 * distance_vars['teammates_medium']
    c_far = 1 + 0.015 * distance_vars['teammates_far']
    C = c_close * c_medium * c_far

    # M_d: Movement direction
    angle_rad = np.radians(angle)
    center_y = 40  # Field center
    y_factor = 1 - (abs(y - center_y) / 40) * 0.3
    M_d = 0.725 - 0.125 * np.cos(angle_rad) * y_factor

    # Total Attack
    A = z_campo * V * C * M_d

    # Calculate Defense Component (D)

    # D_presion: Defensive pressure
    d_dist_norm = distance_vars['closest_defender_distance'] / 10
    D_presion = 0.3 + 0.4 * (1 - d_dist_norm)**2

    # D_cobertura: Defensive coverage
    D_cobertura = (0.7 +
                   0.15 * distance_vars['defenders_very_close'] +
                   0.10 * distance_vars['defenders_close'])

    # M: Physical mismatch
    fatigue_att = 1 - (attacker_minutes_played / 90) * 0.15
    fatigue_def = 1 - (defender_minutes_played / 90) * 0.15
    M = 0.33 + 0.67 * (fatigue_att / fatigue_def)

    # Total Defense
    D = D_presion * D_cobertura * M

```

```
    # Calculate Context Component (X)

    # Time pressure
    if minute <= 75:
      time_factor = 1.0
    else:
      time_factor = 1.0 + 0.02 * (minute - 75)

    # Score pressure
    if goal_difference == 0:
      score_factor = 1.0
    elif goal_difference > 0:
      score_factor = 0.95
    else:
      score_factor = 1.05

    X = 0.99 * time_factor * score_factor

    # Final xD calculation
    xD = A * (1 - D) * X

    # Ensure xD is between 0 and 1
    xD = max(0, min(1, xD))

    return {
      'xD': xD,
      'xD_percentage': f"{xD * 100:.1f}%",
      'location': {'x': x, 'y': y},
      'components': {
        'attack': A,
        'defense': D,
        'context': X
      },
      'factors': {
        'z_campo': z_campo,
        'velocity': V,
        'teammates': C,
        'direction': M_d,
        'pressure': D_presion,
        'coverage': D_cobertura,
        'physical': M
      },
      'distances': {k: v for k, v in distance_vars.items() if k != '
          dribbler_location'}
    }
```

Bibliografía

Analyst, O. (2023). *Top 5 league dribbles interactive*. https://dataviz.theanalyst.com/ad-hoc/rd-dribble-charts-2023-10-02/?viz=1

Fernández, F. (2024). *Pedro alcalá, central del cartagena, trata de agarrar a yeremay durante el partido disputado en cartagonova*. Retrieved November 3, 2024, from https://www.dxtcampeon.com/images/showid2/7023559?w=900

Football, A. (2023). *Opta stats reveal most clinical forwards in europe's top-five leagues so far*. Retrieved May 2, 2025, from https://m.allfootballapp.com/news/EPL/Opta-stats-reveal-most-clinical-forwards-in-Europes-top-five-leagues-so-far/2310838

Ho, C. H. (2018). *Quantifying successful possession (xpg)*. Retrieved July 11, 2018, from https://www.americansocceranalysis.com/home/2018/7/10/ra32uzf18ma2viefm74yjsph8ywywk

Moore, J. (2018). *Expected possession goals: The value of a possession and comparing xpg to other metrics*. Retrieved July 11, 2018, from https://www.americansocceranalysis.com/home/2018/7/10/expected-possession-goals-the-value-of-a-possession-and-comparing-xpg-to-other-metrics

Observatory, C. F. (2018). *Best dribblers: Hazard ahead of neymar and messi*. Retrieved May 2, 2025, from https://football-observatory.com/Best-dribblers-Hazard-ahead-of-Neymar-and-Messi-1996

Observatory, C. F. (2019). *Best dribblers in the big-5: Messi ahead of saint-maximin*. Retrieved May 2, 2025, from https://football-observatory.com/Best-dribblers-in-the-big-5-Messi-ahead-of-Saint

Observatory, C. F. (2024). *Take on index: World rankings*. Retrieved May 2, 2025, from https://football-observatory.com/WeeklyPost476

StatMuse. (2024). *Best dribble completion rate in the la liga 2023-24*. Retrieved May 10, 2025, from https://www.statmuse.com/fc/ask?q=best+dribble+completion+rate+in+the+la+liga+2023-24

StatMuse. (2025a). *Best dribble completion rate in the premier league this season*. Retrieved May 10, 2025, from https://www.statmuse.com/fc/ask/best-dribble-completion-rate-in-the-premier-league-this-season

StatMuse. (2025b). *Most successful dribbles completed this season europe top 5 league*. Retrieved May 10, 2025, from https://www.statmuse.com/fc/ask/most-dribbles-completed-this-season-europe-top-5-league

StatsBomb. (2024). *Libreria gratuita de como usar los datos de statsbomb con r*. Retrieved May 30, 2025, from https://github.com/statsbomb/StatsBombR/tree/master/R