Guide

HOW TO USE HUDL STATSBOMB + HUDL PHYSICAL DATA

How to Access and Use the Data

Within the resource folder, you will find the following files:

- Statsbomb_J1_League.json (the Hudl Statsbomb event data)
- matches.json (additional information relating to each match)
- Physical_Data.json (the aggregated Hudl Physical Data)
- players_mapping.csv (player ID mapping across Hudl Statsbomb and Wyscout)
- teams_mapping.csv (team ID mapping across Hudl Statsbomb and Wyscout)
- matches_mapping.csv (match ID mapping across Hudl Statsbomb and Wyscout)

Loading in the Statsbomb Event Data

Unlike previous free data releases from Hudl Statsbomb, this data is not accessible via the API. Instead, the two JSON files, *Statsbomb_J1_League* and *matches*, must be loaded into your preferred coding environment.

In R, the JSON files should be imported into dataframes named *events* and *matches*, respectively. First, move the files into your working directory and then run the following code:

```
library(jsonlite)

events = fromJSON("Statsbomb_J1_League.json")
events = as.data.frame(events)

matches = fromJSOn("matches.json")
matches = as.data.frame(matches)
```

Once loaded in, they function the same as the equivalent data pulled from the Hudl Statsbomb API, and you can follow any usage guide from that point onwards.

In Python, the JSON files should be imported into dataframes named *events_df* and *matches_df*, respectively:

```
import pandas as pd
import json

# Load the JSON event data, inserting relevant file path
with open('/insert_file_path/Statsbomb_J1_League.json') as f:
    e_data = json.load(f)

# Load the JSON match data, inserting relevant file path
with open('/insert_file_path/matches.json') as f:
    m_data = json.load(f)

# Convert the JSON data to a DataFrame
events_df = pd.json_normalize(e_data)

matches_df = pd.json_normalize(m_data)
```

You will then need to run the following code to adjust the column names to the standard naming convention for Statsbomb data in Python.

```
events_df.columns = events_df.columns.str.replace(".name", "", regex=True)
events_df.columns = events_df.columns.str.replace("[.]", "_", regex=True)

matches_df.columns = matches_df.columns.str.replace(".name", "", regex=True)
matches_df.columns = matches_df.columns.str.replace("[.]", "_", regex=True)
```

For more information about how to work the data, we have published the following guides.

Using Hudl Statsbomb Data in Python:

https://statsbomb.com/articles/soccer/using-statsbomb-free-data-in-python/

Using Hudl Statsbomb Data in R:

https://statsbomb.com/articles/soccer/using-statsbomb-free-data-in-r/

The data specification is another essential resource.

Hudl Physical Data

The Hudl Physical Data, *hudl_physical*, includes the following metrics:

- Total Distance (Metres; Total distance covered across all actions)
- Running Distance (Metres; Distance covered between 15 km/h and 20 km/h)
- High Intensity Distance (Metres; Distance covered above 20 km/h)
- High Speed Running Distance (Metres; Distance covered between 20 km/h and 25 km/h)
- Sprinting Distance (Metres; Distance covered above 25 km/h)
- M/Min (Metres/minute; Total distance covered across all actions, divided by minutes played)
- Max Speed (Kilometres/hour; The maximum speed recorded)
- Count HI (Count of high intensity actions, over 20 km/h)
- Count HSR (Count of high speed running actions, between 20 km/h and 25 km/h)
- Count Sprint (Count of sprinting actions, above 25 km/h)
- Count Medium Acceleration (Count of accelerations with a peak value between 1.5 and 3 metres/second^2)
- Count High Acceleration (Count of accelerations with a peak value greater than 3 metres/second^2)
- Count Medium Deceleration (Count of decelerations with a peak value between -1.5 and -3 metres/second^2)
- Count High Deceleration (Count of decelerations with a peak value of less than -3 metres/second^2)

These are aggregated at a player level on both a per-match basis and within given time periods (0-15 minutes, 16-30 minutes, etc...), as recorded in the *phase* column. The full match aggregation is named *Session*.

Combining the Two Datasets

The three ID mapping files, *players_mapping*, *teams_mapping* and *matches_mapping*, provide Statsbomb and Wyscout IDs (used for the Hudl Physical Data) for players, teams and matches, respectively.

How you choose to connect those to the Statsbomb Event Data and aggregated Hudl Physical Data will depend on user preference and use cases, but the easiest way to ensure they are ready to be connected when required would be to use the mapping files to add additional columns to each of those dataframes with the relevant IDs: Wyscout IDs for the Statsbomb Event Data; Statsbomb IDs for the Hudl Physical Data.

In R, the relevant command to achieve this would be `left_join`; in Python, it would be `pd.merge` with use of the `how="left"` option. The use of the `by` option in both cases will allow you to match the relevant IDs without renaming columns.

The addition of those columns to each dataframe will allow you to use the same method to connect them once you have aggregated the Statsbomb Event Data as desired to combine with the Hudl Physical Data.