



Data Analytics

Identifying cyclists' profiles based on their performances

Bulduk Eker

March, 2023

Table of content

TABLE OF CONTENT	2
INTRODUCTION.....	3
DATA AND DATA SOURCES	4
DATA COLLECTION	8
DATA CLEANING & DATA PREPARATION	10
EXPLORATORY DATA ANALYSIS	13
DATA BASE TYPE SELECTION	18
ENTITIES. ERD	19
CONCLUSION	24
ANNEXES.....	25

Introduction

There is an epic dimension about cycling competition. Watching those cyclists and their distorted faces by the efforts, while “dancing” on their pedals on the 26% gradient climb of the *Mur de Huy*... Watching them splitting a frenzy orange crowd that screams at the top of their lungs in the “*Dutch corner*” of the *Alpe d’Huez* climb... Watching their nervous faces stained with mud on the cobbled roads of the *Paris-Roubaix*, the “*hell of the North*”... Watching the last rider of the Tour de France battling against the broom wagon”, to avoid disqualification... Witnessing those athletes, it gives the feeling to watch modern gladiators on bikes in an open-world arena, surrounded by admiring and cheering plebeians.

You guessed it, I enjoyed watching cycling whenever I can, and I have been interested in cycling competition in general, and road cycling more particularly since a few years now.

Over time, beyond the spectacular aspect of road cycling, I was more and more interested by strategical aspects of the competition: the tactics and strategies of individuals and teams during races, the teamwork to lead/protect a team leader, the preparation of races, the usage of materials and their strategical selections of riders depending on races, tours and objectives for a race or for a season. Although the first elements are highly related to cycling tactics knowledges, live configuration of races, as well as live race information (including live weather condition for example), the latter is more related to knowledge of who the riders of a team are, i.e. what are their specialties, what are their strengths, weaknesses. Thus, it is crucial for teams to have a good understanding who are their members, to not only make selection of riders, but also in a recruitment perspective, may that be for building a balanced team filled with different profiles, or overcoming a weakness, or even strengthening more a specialty, or replacing a rider leaving the team.

Therefore, this is the goal of this project: identifying through data of past performances different profiles of riders, investigating who are the most prolific riders, look at the composition of teams based on this identification.

To achieve this goal, I will go through the “stages” that I had to face, from collection data via web scraping, to cleaning them and perform exploratory data analysis both with data visualization and SQL queries, in order to produce insights.

Data and data sources

The data are from a reliable source of cycling called ProCyclingStats (<https://www.procyclingstats.com/>), that I will refer by “PCS” throughout this paper. PCS is a website that is the statistical references in cycling competition, mainly road cycling and cyclo-cross. The website gathers a lot of information, among others:

- Race information, race by race, stage by stage, with detailed ranking, including amateur competition.
- Rider information, including year-per-year and race-per-race performances, statistics.
- Team information, statistics, including historical performances.
- Injury records.

For the purpose of this project, I decided to consider only races and performances after 2019. Indeed, 2019-2020 were the 2 years of the rise of numerous young riders. For example, when looking at the winners of 2018's monuments (5 most prestigious one-day races of the circuit), 2 winners are now retired, the 3 others have shown declining performances since then. Whereas among winners after 2019, 8 of them were not even in the professional circuit in 2018. It is thus important to keep only recent data, and not have historical performance data of riders once at their peak, that will just bring noises to performance analysis.

Here are the information gathered table per table, during the web scraping and before data cleaning and addition of new variables:

- **Race information**

Race code	Unique id per race and stage, that also allows to reach the page dedicated on the race easily on PCS. The code usually takes the generic name of the race and the year
Date	Day the race took place
Start time	Time the start of the race was given
Avg. speed winner	Average speed of the winner of the race
Race category	Category of the race (Elite or U23 or Junior)
Distance	Total distance of the race
Points scale	Point scale specific to PCS that awards points to riders depending on race / stage / ranks / importance of the race

UCI scale	Similar to the points scale but with the scale made by the <i>Union Cycliste Internationale</i> , the world governing body for sports cycling
Parcours type	Specify if the stage is flat, hilly or mountain, with flat finish or uphill finish
ProfileScore	Specific profile score of PCS that calculates how difficult is the race (the difficult climb closest to finish line, the higher the score). Link to detailed explanation can be found in annexes
Vert. Meters	Ascending meters climbed during the races
Departure	City of departure of the race
Arrival	City of arrival of the race
Race ranking	Ranking specific to PCS based on <i>startlist</i> quality in the current year (see below)
Startlist quality	Point system that shows how much highly ranked riders there was in the race
Won how	Type of finish: sprint (large group, small group, two persons), solo win, time trial win
Avg temperature	Average temperature during race

- **Race results**

Rnk	Rank achieved by a rider in a race
GC	General classification rank for stage race only
Timelag	General classification time delta to the first ranked in GC
BIB	Identification number specific to each race
Rider	Name of the rider
Age	Age of the rider
Team	Rider's team
UCI	UCI points awarded to rider
Pnt	PCS points awarded to rider
Time	Time performed by winning rider, and for other the time delta vs winner
Avg	Average speed of rider

Race_code	Race ID
-----------	---------

- **Riders' information**

Fullname	Full name of rider
Team	Rider's team name
Birthdate	Rider's birthdate
Country	Rider's country (country he represents for dual citizenship)
Height	Rider's height
Weight	Rider's weight
One_day_race_pts	Total career PCS points for one-day-race
gc_pts	Total career PCS points for general classification for stage race
tt_pts	Total career PCS points for time-trial
sprint_pts	Total career PCS points for flat race (profile score <=50)
climb_pts	Total career PCS points for climb race (profile score >50)
pcs_ranking	PCS points ranking for the current year (2023)
uci_ranking	UCI points ranking for the current year (2023)
number_wins	Total career wins including stage win and grand classification win
pts_pcs_2023	Total PCS points for 2023 (<i>column duplicated for the year 2022, 2021, 2020 and 2019</i>)
rank_pcs_2023	PCS points ranking for 2023 (<i>column duplicated for the year 2022, 2021, 2020 and 2019</i>)
Rider_code	Unique ID for each rider, that also allows to reach the page dedicated for the rider

- **Teams' information**

Team_url	Unique ID for each team per year, that also allows to reach the page dedicated for the rider. The name takes the name of the team and the year
Team-name	Team's name
Circuit	Division level the team competes in: World Tour (highest level) or ProSeries (second highest level)

Country	Team's country
Team_url	Team's dedicated page on PCS
More_gear_url	Team's gear & materials information on PCS
Bike	Team bike's provider
Groupset	Team group set's provider
<i>The previous two variables of providers are also available for the following materials: wheels, saddle, tyres, pedals, powermeter, sunglasses, helmets, shoes, bartape, kit, sport nutrition, cycling computer, hometrainer</i>	

- **Team historical performances**

Team_code_history	Unique ID for each team per year, that also allows to reach the page dedicated for the rider. The name is constituted by the name of the team and the year (one code per year for each team)
Team_url	Team's dedicated page on PCS for the specific year
Team_url_2023	Current team code of the team (in 2023)
Team Name	Team's name for the specific year
Wins	Number of wins for the specific year
Points	Total PCS points for the specific year
PCS ranking	Team's ranking on PCS points
UCI ranking	Team's ranking on UCI points

Data collection

Because PCS has no API or any other platform to collect data from their databases, I had to use web scrapping of the website. And if the web scrapping was a type of cycling races, that would certainly be a mountain race, with endless climbs.

1. First step was to scrap all the URLs of races, thanks to the calendar engine of PCS which records all races for a year and circuit. Teams' URLs have also been scrapped with a dedicated page.
2. Second step was to scrap the URLs of riders from the teams' URLs.
3. Third step was to scrap the actual data for races information, teams' information (both basic information and historical performance data), riders' information, and most importantly the race results. To make web scrapping more efficient, the races' URLs have been split year per year, with also a distinction for one-day-races and stage races, and grand tour (Giro, Tour, Vuelta). For each type of information I needed, I created a different function, one to scrap the data based on the URL of one page (as in the below image to scrap race results), and then a second one, that runs this process to a list of multiple URLs (as the second image below).

```
def create_dataframe_from_url(url):
    response = requests.get(url)
    if response.status_code != 200:
        return None

    soup = BeautifulSoup(response.content, "html.parser")

    rows = soup.find_all('tr')

    columns = [th.text.strip() for th in rows[0].find_all('th')]

    data = []

    for row in rows:
        values = [td.text.strip() for td in row.find_all('td')]

        # if the row contains a rider (i.e., it's not a header or footer row), add it to the data list

        # following line does not apply to one day race => to put in comment when scrapping for one day races and remove indenta
        if (values and values[2].startswith('+')) or (values and values[0] == 'DNF') or (values and values[0] == 'DNS'):
            data.append(values)

    df = pd.DataFrame(data, columns=columns)
    df['race_code'] = url.replace('https://www.procyclingstats.com/race/', '')

    return df

# function that runs the scrapping of races results for list of urls
def create_dataframe_from_urls(df, url_column):
    num_urls = len(df[url_column])
    dataframes = []
    for i, url in enumerate(df[url_column]):
        print(f"\rProcessing URL {i+1}/{num_urls}: {url}", end="")
        dataframes.append(create_dataframe_from_url(url))
        # random delay to avoid getting banned
        timer = 0.5 + 0.5 * random.random()
        time.sleep(timer)
    # concatenate all the dataframes in the list into a single dataframe
    result_df = pd.concat(dataframes, ignore_index=True)
    return result_df
```


We can note above the presence of a randomized time sleeper in the function between each URLs scraped. This was done to prevent any potential banishment. This was not necessarily done for all functions (such as teams' information which is "only" 36 teams).

In total, 839 pages have been scraped for race results, with at the end more than 119 thousand rows of results (before removal of duplicates which reduces it to 117.7k rows). The same number of pages have been scraped for race information generating 839 rows.

36 teams' information have been scraped also, and 139 historical performances for these teams have been scraped (4 years from 2019 to 2022, with some teams having made debut only after 2020, hence the total being 139, and not 144).

Finally, from the 36 teams, all their members' information has been scraped, which represents 937 riders.

Data cleaning & Data preparation

To continue the allegory with cycling, data cleaning looked like a time trial stage, with constant efforts and vigilances.

Data cleaning process, started by checking if columns are harmonized, removing every kind of formatting (spaces, trailing spaces, upper case) and dropping duplicates. A lot of columns created and scraped from the website were not in right tyoes, so some words had to be removed and types had to be changed to float or integer type (for example “240.3 km” instead of “240.3”). I also added other columns such as name of the race, the year, the stage number, the type of race (one-day race or stage race), if it is a time-trial race or not. Those columns would be helpful for next of data cleaning and exploratory data analysis. Finally, I “encoded” the “*won how*” columns, which was not really encoding as value were multiples (>150 different values). Therefore it was rather identifying if the column included the word “sprint” or “solo” or “time trial”. 2 others columns were created describing the type of sprint, and the distance for the solo win.

One major aspect of the cleaning, was to take into account general classification wins. Indeed, these are points awarded for final winners of general classification in stage but those information couldn’t be scraped easily which I eventually left for the cleaning and data preparation part. The first stage was to identify the final stages of a particular stage race using the “groupby” method on the *race_name* and year. A boolean type column was then created for final stages.

```
# Add a column to identify final stage for stage race
def identify_final_stage(df, race_name='race_name', year='year', stage_number='stage_number'):
    # Convert stage_number to integer type as it is currently object type
    df[stage_number] = df[stage_number].astype(int)

    max_stage = df.groupby([race_name, year])[stage_number].transform(max)

    df['final_stage'] = df[stage_number] == max_stage

    df['final_stage'] = df['final_stage'].map({True: 'True', False: 'False'})

    return df
```

The second step was to assign the points. After filtering the data on final stage only, and saving the points system in 3 lists, I ran the function below to assign the points to the first, second, etc. (order from the lowest to highest rank) based on the column ‘gc’ that is showing general classification rank. This was assigned in a newly created column.

```
def assign_points(group):
    if group['points_scale_x'].iloc[0] == 'gt.a':
        points_list = gt_a
    elif group['points_scale_x'].iloc[0] == 'gt.b':
        points_list = gt_b
    elif group['points_scale_x'].iloc[0] == '2.wt':
        points_list = two_wt
    else:
        raise ValueError('Invalid points_scale value')
    group = group.sort_values(by='gc')
    group['gc_pcs_point'] = points_list[:len(group)]
    return group

# Group by race_code and apply the function to each group
df_final_stage_finished = df_final_stage_finished.reset_index(drop=True)

df_final_stage_finished = df_final_stage_finished.reset_index(drop=True).groupby('race_code').apply(assign_points).reset_index(drop=True)
```

An other major challenge that I faced is that, in the results table, is that the name of the riders were in a different format than in the rider information tables. The website had a code for each rider specific to the PCS website, but the names in race results were in normal format, and in a reversed format (“*SURNAME Firstname*” instead of “*Firstname SURNAME*”). The issue was that many had middle names or particle name (especially Dutch and Belgium riders with “van” “van der” “de”), and others had middle names that appeared in the full name but not in the rider-code, and vice versa.

After many trial and long search, I found a library called “*fuzzywuzzy*”, a library that provides string matching functions, by comparing two strings and determine how similar they are, with a technique called the “*Levenshtein distance*”. In this library, the functions calculate a value that is between 0 and 100, 0 being a total mismatched, and 100 a complete match. Below, we keep a match only if the matching score is above 80.

```
def get_rider_code(name, df_rider_check):
    choices = df_rider_check['fullname'].tolist()
    match = process.extractOne(name, choices, scorer=fuzz.token_sort_ratio)
    if match[1] >= 80:
        index = choices.index(match[0])
        return df_rider_check.loc[index, 'rider_code']
    else:
        return None
```

Since there are more than 900 riders, and 117000 names to match, the process was incredibly long. However, the final results was really satisfying with no mismatch. Only riders that are not in the riders’ tables had no match (riders retired or out of professional circuits).

```
df_all_results['rider_code'] = None

for i in tqdm(range(len(df_all_results))):
    name = df_all_results.loc[i, 'rider']
    code = get_rider_code(name, df_rider_check)
    df_all_results.loc[i, 'rider_code'] = code

2%| | 1934/117770 [03:39<3:38:43, 8.83it/s]
```

Finally, to get summary metrics of riders and teams performance, a new dataset was created, based on race results information. Following the classification of PCS website on profilescore, for each flat stage (profile score= ≤ 50), climb stage (profile score ≥ 51), team trial stage, general classification, one-day race, both for riders and teams, I calculated :

- Number of races, wins, podiums, top 10s, total PCS points.
- Average rank, average PCS points.
- % of wins, podiums, top10s.

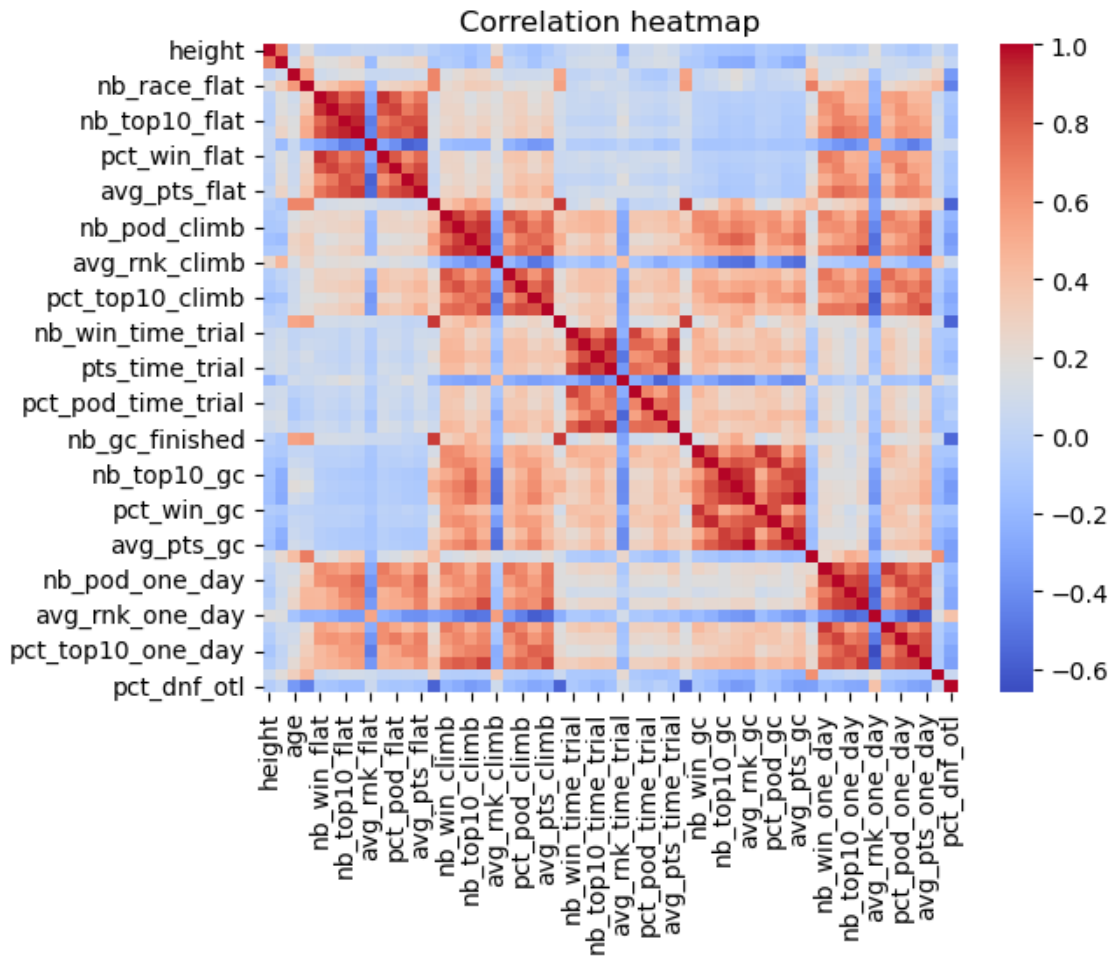
Of course, some of those metrics are likely to be highly correlated. So, in a perspective of running a modelling, it will most probably require to keep only few of those metrics. However, at this stage of exploratory data analysis, I didn't know which metrics will be more reliable than others.

Those datas were eventually merged to rider information table and team information table.

Once those data ready. I could perform the exploratory data analysis.

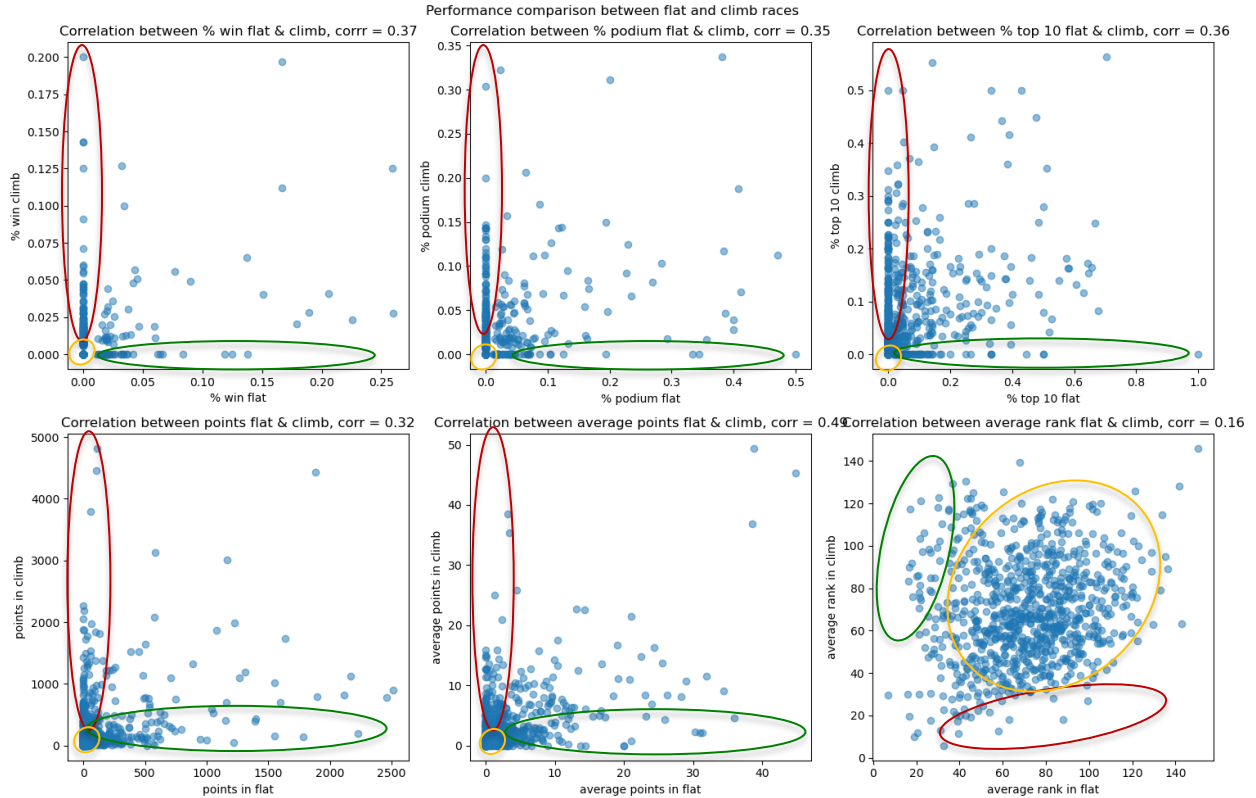
Exploratory data analysis

- Corellation heatmap:



As expected, number of wins, podium, top ten and points, etc. for each category of race created are highly correlated. Therefore, before doing any modelling on classification of riders, it will be important to keep only few composite indicators per type of races. The blue lines that we can see crossing each red squares are the average rank, and there are not correlated to other metrics from the same race category, so they can be a helpful metric. Finally, we don't see strong correlation between categories neither, which is a good sign for future modelling to identify type of riders.

- **Correlation between performances in flat races and climb races:**



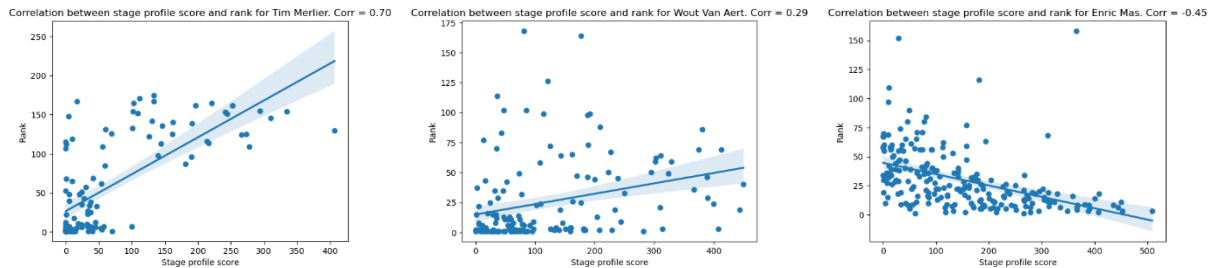
First, we see that there is little correlation between performance in flat stage and climb stage, which is a good sign for a potential future modelling, as I anticipated that there are specialists of flat stage and climb stage in the peloton.

The first row of charts which shows respectively percentage of wins, podium & top 10, we can see clearly that there is each time two “lines”: one vertical and one horizontal. This suggests that there are indeed at least 2 types of riders: cyclists that never won a climb stage but have some wins in flat stage (green circled), and vice and versa (red circled). There seems to be a third category, that has performances on both types of stages (in the scatter plot those who are located farther from the x and y axis), those are more likely to be “all-rounder” riders, meaning riders that are good in any kind of stages. Finally, the fourth one can be identified at the really bottom left (orange circled) of the scatter plots: they are riders getting no top 10, they are most likely to be domestiques, water carrier.

The second row shows for the first points and average points in each type of flat, which shows pretty much the same aspects: the top left (red circled) and bottom right (green circled) are riders that are specialists in respectively climb and flat stage. However, the interesting aspect here is that most of riders get little points (orange circled) in any

stage: most likely domestiques or lower performing riders. The last graphs, shows average rank, and indeed most of the riders are likely to be in the “peloton” meaning.

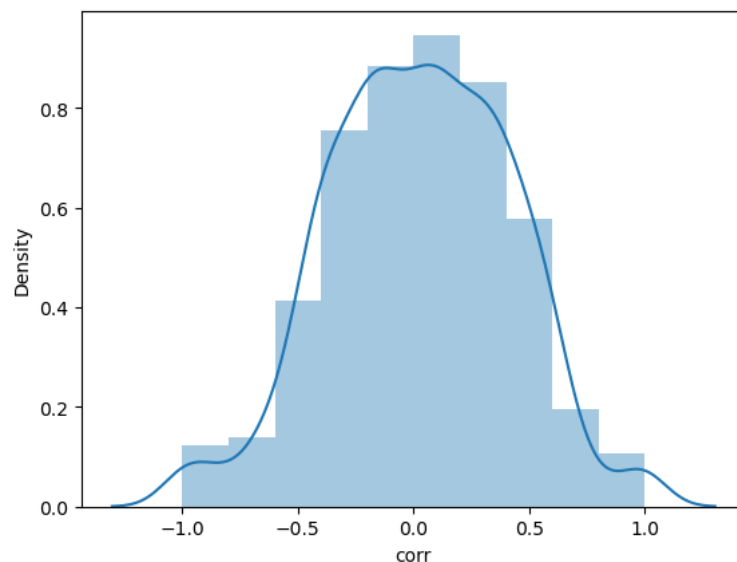
- Correlation between Profile Score and Rank, for selected riders:



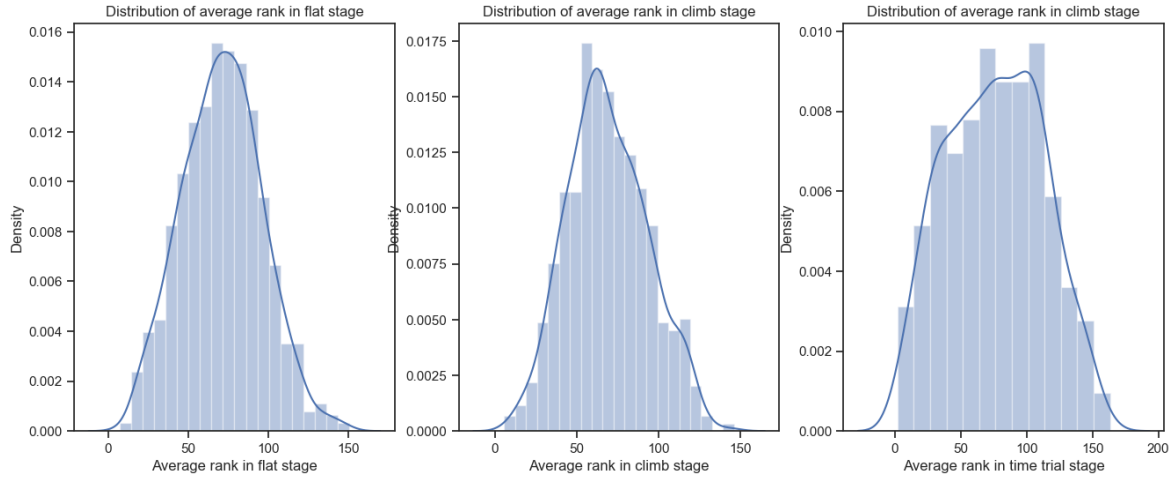
To confirm the existence of profile, we can also look for each riders the correlation between profile score and rank. I selected 3 riders, *Tim Merlier*, *Wout Van Aert* and *Enric Mas*, commonly categorised respectively as sprinter, all-rounder and climber.

We can clearly see that the first one, the rank is highly is correlated to the difficulty of the race. Oppositely for the last one, it is a negative correlation, meaning that the more difficult the race, the more likely this climber will finish in first place. As for the all-rounder in the middle, there is a lower correlation: this rider can in good or bad position in any race, although we can see that it gets a bit more comfortable with flat stage.

When this correlation is run for all riders, here is below the distribution of correlation between rank and profile score, with in both extremists pure specialist (left: climber, right: sprinters), and in the middle either all-rounder or domestiques.

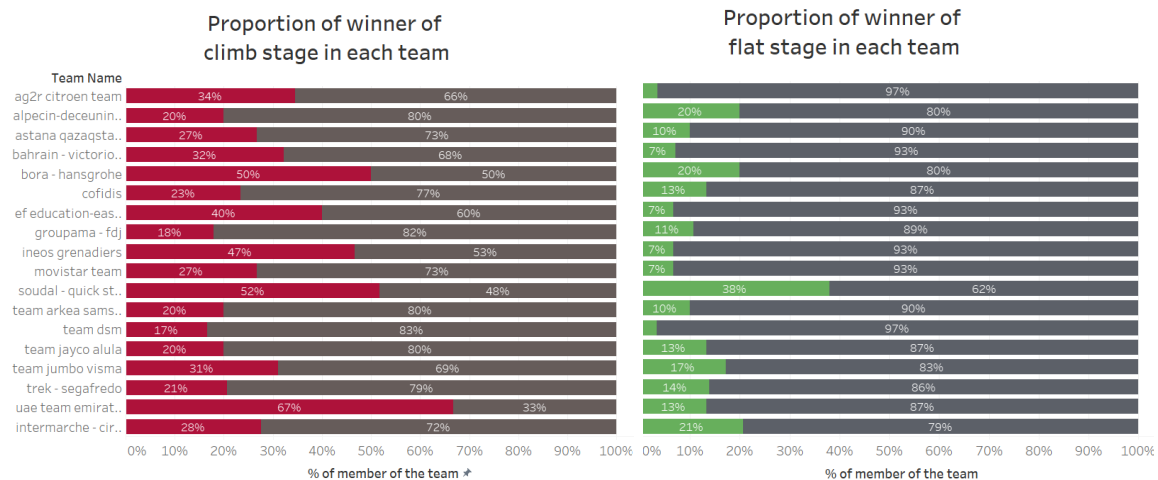


- Distribution of rank:



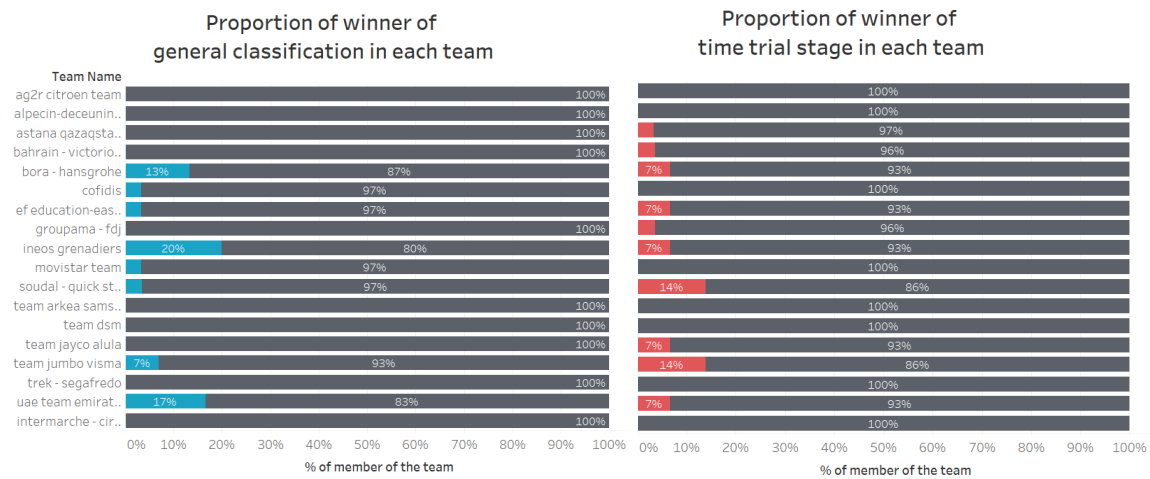
This normalized distribution of average ranking suggests that also there are riders that finishing more often at top positions depending on races. This supports again the idea that there are specialists of type of races. Also, we pointed earlier that average rank of race is an interesting metrics for future metrics as less likely to be correlated any others. We also see that the distribution is quite normalized for most type of races, which is a good aspect, as some modelling requires gaussian distribution, such as t-test or linear regressions.

- Proportion of specialists in each team:



Finally, I look for each team of the World Tour circuit at their riders' composition, more specifically if the teams had at least one winner of stage, to eventually identify teams that have strength and weaknesses in type of races. For example, *UAE Team Emirates* has 2/3 of its rider that won at least one climb stage, while the *Team DSM* and *AG2R*

Citroën Team are two teams that show weaknesses in flat stage. *Soudal Quick-Step* is good example of a balanced team, that have riders that can perform in flat and climb stage.



As for the general classification (in stage race), this seems to be a “private” competition where only few teams can realistically win (*Ineos Grenadier*, *Team Jumbo-Visma*, *UAE Team Emirates*). Same goes for time trial, which looks like even more niche specialisation where few riders win.

Data base type selection

After cleaning the data, the database needed to be created. The most obvious choice for the data I had, was to use SQL. Here are the reasons why:

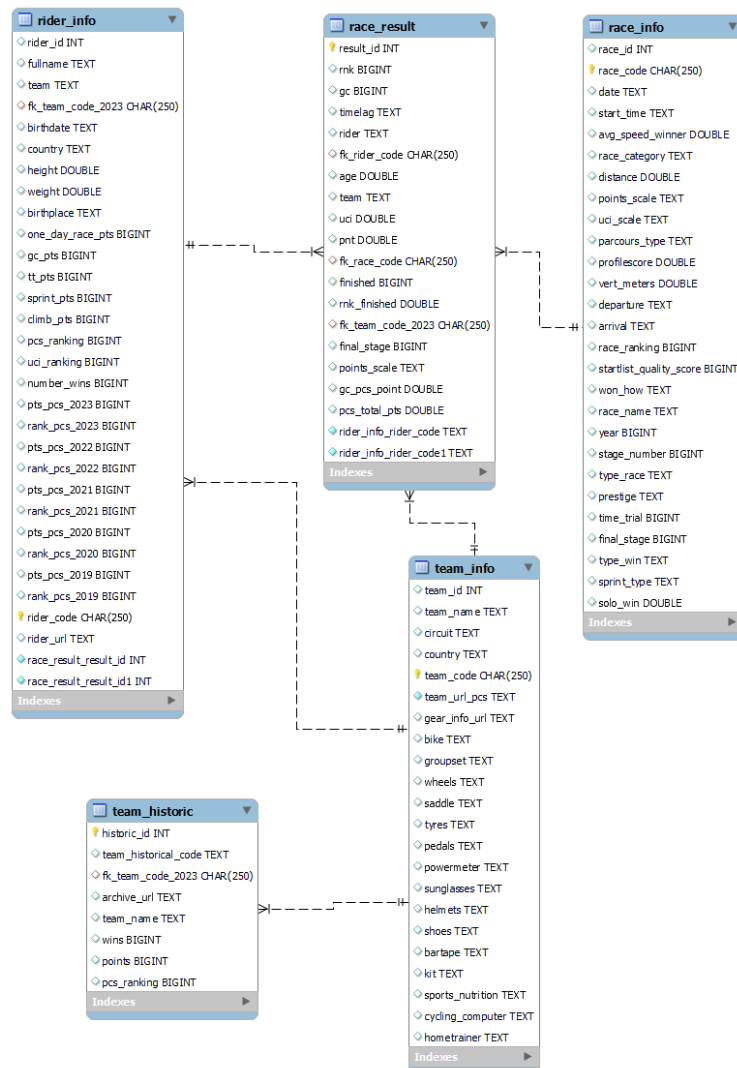
- SQL databases are relational, NoSQL databases are non-relational.
- SQL databases use structured query language and have a predefined schema. NoSQL databases have dynamic schemas for unstructured data.
- SQL databases are vertically scalable, while NoSQL databases are horizontally scalable.
- SQL databases are table-based, while NoSQL databases are document, key-value, graph, or wide-column stores.
- SQL databases are better for multi-row transactions, while NoSQL is better for unstructured data like documents or JSON.

In a configuration where I have riders, teams, race info and race results, the possibility to make relational databases between was a big asset, especially in one-to-many configuration.

Vertical scalability is also an important asset as races happen every week, and therefore, we expect more data to come. Same goes for riders' changing teams, teams changing names, etc.

Entities. ERD

For rider information, race information and team information, there was unique value scraped from the PCS website, with respectfully *rider_code*, *race_code* and *team_code*, and I used them as primary keys. However, for race result and team historic, there was not unique value, so the indexes, which had no name initially, were selected as primary keys and renamed *result_id* and *historic_id*. Note that the ERD was created prior to the data exploration analysis, so it does not include the composite metrics created and shown in exploratory data analysis.



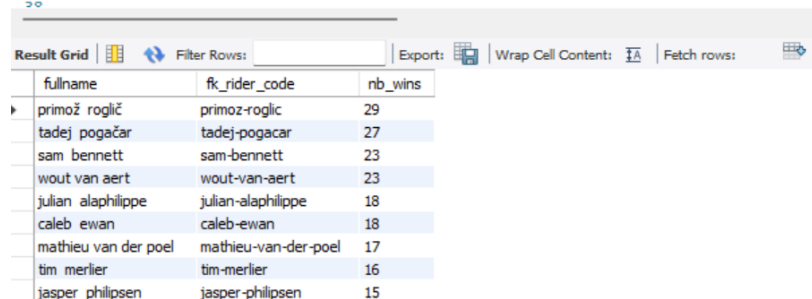
In order to create the table, I used sqlalchemy library on Python. Considering, the size of the final sets, the upload of the data would have been quicker using Python.

```
def convert_pd_df_tosql(fname, table_name, schema="cycling"):
    connection_string = 'mysql+pymysql://root:' + "lorem ipsum" + '@127.0.0.1:3306/'
    engine = create_engine(connection_string)
    engine.execute('CREATE SCHEMA IF NOT EXISTS cycling')
    df = pd.read_csv(fname)
    df.to_sql(table_name, engine, schema, index=False)
    return "Created table"
```

In data exploration, we confirmed the idea that there are individuals specialised in flat stage and other in climb stage. Now we will look at individuals and teams through SQL queries.

First, we can identify the most successful riders since 2019, among which we found cyclists commonly classified by connoisseurs, as pure sprinters (*Bennett, Ewan, Philipsen, Merlier*), one-day-race specialist (*Alaphilippe, van der Poel*) and all-rounder (*Pogačar, Roglič, Van Aert*).

```
31 • SELECT rider_info.fullname, race_result.fk_rider_code, COUNT(*) AS nb_wins
32 FROM race_result
33 JOIN rider_info ON rider_info.rider_code = race_result.fk_rider_code
34 WHERE race_result.rnk = 1
35 GROUP BY rider_info.fullname, race_result.fk_rider_code
36 ORDER BY nb_wins DESC
37 LIMIT 10;
38
```



The screenshot shows a Jupyter Notebook interface. At the top, a SQL query is displayed in a code cell. Below the code cell, the 'Result Grid' tab is active, showing a table with the results of the query. The table has three columns: 'fullname', 'fk_rider_code', and 'nb_wins'. The results are ordered by 'nb_wins' in descending order, with 10 rows displayed. The interface includes a 'Filter Rows' search bar, an 'Export' button, and a 'Fetch rows' button.

fullname	fk_rider_code	nb_wins
primoz roglic	primoz-roglic	29
tadej pogacar	tadej-pogacar	27
sam bennett	sam-bennett	23
wout van aert	wout-van-aert	23
julian alaphilippe	julian-alaphilippe	18
caleb ewan	caleb-ewan	18
mathieu van der poel	mathieu-van-der-poel	17
tim merlier	tim-merlier	16
jasper philipsen	jasper-philipsen	15

Some teams are much more successful such as *Soudal – Quick Step* and *Alpecin-Deceuninck* which we saw earlier in exploratory data analysis have more flat stage specialists, but also climb stage specialists for the first team. There is then a second tier of team, with *UAE Team Emirates*, *Team Jumbo-Visma*, both teams that have many different profiles in their teams as we saw earlier: climbers, grand classification specialists and time trialists.

```

39 • SELECT team_info.team_name, race_result.fk_team_code_2023, COUNT(*) AS nb_wins
40 FROM race_result
41 JOIN team_info ON team_info.team_code = race_result.fk_team_code_2023
42 JOIN race_info ON race_info.race_code = race_result.fk_race_code
43 WHERE (race_result.rnk = 1) AND (race_info.type_race LIKE 'one%')
44 GROUP BY team_info.team_name, race_result.fk_team_code_2023
45 ORDER BY nb_wins DESC
46 LIMIT 10;

```

team_name	fk_team_code_2023	nb_wins
soudal - quick step	soudal-quick-step-2023	44
alpecin-deceuninck	alpecin-deceuninck-2023	30
uae team emirates	uae-team-emirates-2023	24
team jumbo visma	team-jumbo-visma-2023	20
lotto dstny	lotto-dstny-2023	17
intermarche - circus - wanty	intermarche-circus-wanty-2023	16
cofidis	cofidis-2023	15
bora - hansgrohe	bora-hansgrohe-2023	13
ef education-easypost	ef-education-easypost-2023	11

In the query below, the *nb_race* column was added as a condition. Indeed, some young riders, with only one participation in relatively minor race, had better average position than real “kings of mountain” (*Roglič*, *Pogačar*, *Bernal*, *Yates*, *Mas*). This suggests that for future classification, to avoid riders with not enough data, the number of races should be considered somehow or alternatively the importance of the races with the PCS scale system which gives more points to important races.

```

58 • SELECT fullname, fk_rider_code, avg_position, nb_race
59 FROM (
60 SELECT rider_info.fullname, race_result.fk_rider_code, avg(race_result.rnk) AS avg_position, count(*) AS nb_race
61 FROM race_result
62 JOIN rider_info ON rider_info.rider_code = race_result.fk_rider_code
63 JOIN race_info ON race_info.race_code = race_result.fk_race_code
64 WHERE (race_info.parcours_type = 'hill_uphill_finish' OR race_info.parcours_type = 'mountain_flat_finish'
65 OR race_info.parcours_type = 'mountain_uphill_finish')
66 AND (race_result.finished = 1)
67 GROUP BY rider_info.fullname, race_result.fk_rider_code
68 HAVING nb_race > 5

```

```

69     ORDER BY avg_position ASC
70     LIMIT 10
71 ) subquery_position_climb;
72

```

fullname	fk_rider_code	avg_position	nb_race
primož rogljč	primoz-rogljic	11.376146788990825	109
tadej pogačar	tadej-pogacar	11.380952380952381	84
juan ayuso	juan-ayuso-pesquera	13.727272727272727	22
egan bernal	egan-bernal	15.316666666666666	60
joão almeida	joao-almeida	15.542857142857143	70
adam yates	adam-yates	16.46987951807229	83
aleksandr vlasov	aleksandr-vlasov	18.191176470588236	68
richard carapaz	richard-carapaz	19.326732673267326	101
enric mas	enric-mas	20.21551724137931	116
guillaume martin	guillaume-martin	21.410714285714285	112

Below, the PCS points system is good enough to include both grand classification specialist, all-rounder, and one-day-race specialist, however no sprinter (except Van Aert which can be counted as all-rounder). There are, however, more sprinter in the next 10 (second image).

```

73 • SELECT rider_info.fullname, race_result.fk_rider_code, SUM(race_result.pcs_total_pts) AS pcs_points
74 FROM race_result
75 JOIN rider_info ON rider_info.rider_code = race_result.fk_rider_code
76 GROUP BY rider_info.fullname, race_result.fk_rider_code
77 ORDER BY pcs_points DESC
78 LIMIT 10;

```

fullname	fk_rider_code	pcs_points
tadej pogačar	tadej-pogacar	8650
primož rogljč	primoz-rogljic	8446
wout van aert	wout-van-aert	7825
julian alaphilippe	julian-alaphilippe	5215
mathieu van der poel	mathieu-van-der-poel	4350
adam yates	adam-yates	4046
richard carapaz	richard-carapaz	4038
michael matthews	michael-matthews	3884
remco evenepoel	remco-evenepoel	3754
enric mas	enric-mas	3669
jasper philipsen	jasper-philipsen	3506
alexander kristoff	alexander-kristoff	3476
sam bennett	sam-bennett	3419
caleb ewan	caleb-ewan	3397
peter sagan	peter-sagan	3350
jakob fuglsang	jakob-fuglsang	3331
egan bernal	egan-bernal	3292
david gaudu	david-gaudu	3281
jonas vingegaard	jonas-vingegaard-ra...	3090
matteo trentin	matteo-trentin	3066

Finally, I looked at all non-finishers (DNF = did not finished; DNS = did not start; OTL = finished over time limit; DSQ = disqualified), with special on focus on the 'DNFs'. Some riders have a high amount of DNFs. In some races, such as classics (one-day-

races) or minor stage races, it is usual for some riders to stop the race earlier, because they produce important effort for their leaders, carry water and can quit the race when the job is done, or because they need to save energy for the next one-day-race (especially in spring season when they are multiple race in few days). Another reason is that they are too late in the race, and it does not worth it continuing for minor races. Last reason can be simply injuries, but it is unlikely that all the DNFs are results of injuries.

The queries allow us to see some names, which are indeed not the big names of the professional circuits.

```

80 • SELECT
81     rider_info.fullname, race_result.fk_rider_code,
82     COUNT( CASE WHEN rnk = 'dnf' THEN rider_code END) as dnf_count,
83     COUNT( CASE WHEN rnk = 'dns' THEN rider_code END) as dns_count,
84     COUNT( CASE WHEN rnk = 'otl' THEN rider_code END) as otl_count,
85     COUNT( CASE WHEN rnk = 'dsq' THEN rider_code END) as dsq_count,
86     COUNT(*) as nb_race
87 FROM race_result
88 JOIN rider_info ON rider_info.rider_code = race_result.fk_rider_code
89 GROUP BY rider_info.fullname, race_result.fk_rider_code
90 ORDER BY dnf_count DESC
91 LIMIT 10;
  
```

	fullname	fk_rider_code	dnf_count	dns_count	otl_count	dsq_count	nb_race
►	manuele boaro	manuele-boaro	34	1	0	0	176
	tom bohli	tom-bohli	31	0	0	0	174
	leonardo basso	leonardo-basso	31	0	0	0	101
	tom devriendt	tom-devriendt	31	1	0	0	96
	alexandr riabushenko	alexandr-riabushenko	30	1	2	0	145
	maciej bodnar	maciej-bodnar	29	2	1	0	183
	yevgeniy gidich	yevgeniy-gidich	29	0	0	0	100
	ivo oliveira	ivo-emanuel-alves	27	2	1	0	129
	guy sagiv	guy-sagiv	26	0	0	0	82
	martin laas	martin-laas	26	0	0	0	99

Conclusion

In conclusion, the analysis of professional cycling data using exploratory data analysis and SQL queries provided valuable insights into the existence of different types of riders. The analysis has led to the conclusion that there are indeed at least four distinct types of riders. Some excel at races that are flat (presumably sprinters), while some are better suited to roads with hill and mountains. This was confirmed by performing a correlation of profile score and rank for all riders. Some riders can be counted as all-rounders, as they can compete in variety of race types, no matter how flat or steep inclines they face. Finally, the fourth category, which is probably the biggest group are most of the time finishing in the “peloton”, with few honours.

A second important insight pulled from this analysis is that certain teams also have specialities. This might suggest that teams have strategies for selecting and training their riders based on precise objectives. We can then easily imagine a need to understand compatibility between teams and riders. When it comes to grand classifications and time trials, few riders and teams can really compete. So, there might be a fifth category there.

We identified some names among the most successful riders, but also among potential domestiques by watching the DNFs count.

This project could be improved further to gain deeper understanding of what makes riders successful. For example, we can look at how flat stage winners are winning, i.e., if they rather win in solo, or in a sprint, and if so, whether the sprint was a large or small group, which would help us to identify pure sprinter for those who wins in large group sprint configuration.

A further investigation can also involve a clustering modelling to confirm the existence of those groups and identify more easily the riders in each group. We can also build a tool that will find the best fit for a team, or the best replacement of a rider by another one.

Annexes

GitHub repository:

https://github.com/bulduk-e/Final_Project

ProCyclingStats:

- **Website:** <https://www.procyclingstats.com/>
- **Profile Score:** <https://www.procyclingstats.com/info/profile-score-explained>
- **PCS points scale:** <https://www.procyclingstats.com/info/point-scales>