```python
import requests
import bs4
import math
import matplotlib.pyplot as plt
import pandas as pd
import re

# get the data from wikipedia using the wikipedia api
def get_data(start_year, end_year):
    results = {}
    for year in range(start_year, end_year + 1):
        results[year] = {"table": [], "top goals": 0}
        page = f"Fußball-Bundesliga_{year}/{(year + 1) % 100:02d}"
        if year == 1999:
            page = f"Fußball-Bundesliga_1999/2000"
        # get all the sections of the page with the wikipedia api
        url = f"https://de.wikipedia.org/w/api.php?action=parse&format=json&page={page}&prop=sections"
        try:
            sections = requests.get(url).json()["parse"]["sections"]
        except:
            print(url)
            exit(1)
        tabelle = []
        # get the section of "Abschlusstabelle"
        for section in sections:
            if section["line"] == "Abschlusstabelle":
                # get the html of the section
                html = requests.get(f"https://de.wikipedia.org/w/api.php?action=parse&format=json&page={page}&section={section['index']}").json()["parse"]["text"]["*"]
                # parse the html
                soup = bs4.BeautifulSoup(html, "html.parser")
                # get the table
                table = soup.find("table", {"class": "wikitable"})
                # get the rows of the table
                rows = table.find_all("tr")

                # get the first row of the table
                first_row = rows[0]
                # get the cells of the first row
                cells = first_row.find_all("th")
                # get the index of the cell that contains "Punkte"
                points_index = 0
                for cell in cells:
                    if "Punkte" in cell.text:
                        break
                    points_index += 1
                # get the index of the cell that contains "Sp."
                matches_index = 0
                for cell in cells:
                    if "Sp." in cell.text:
                        break
                    matches_index += 1
                # get the index of the cell that contains "Tore"
                goals_index = 0
                for cell in cells:
                    if "Tore" in cell.text:
                        break
                    goals_index += 1

                # get colors of the team in relegation or dropout
                relegation_color = ""
                dropout_color = ""
                legend = soup.findAll("li")
                for l in legend:
                    if "relegation" in str(l).lower():
                        expr = re.compile(r"#[0-9a-f]{6}")
                        relegation_color = expr.search(str(l).lower()).group(0)
                        break

                for l in legend:
                    if "abstieg" in str(l).lower():
                        expr = re.compile(r"#[0-9a-f]{6}")
                        dropout_color = expr.search(str(l).lower()).group(0)
                        break

                # get the data of every team
                for row in rows[1:]:
                    cells = row.find_all("td")
                    if not cells:
                        continue
                    points = cells[points_index].text.strip()
                    if ":" in points:
                        points = points.split(":")[0]
                    if ":" in cells[goals_index].text.strip():
                        goals = cells[goals_index].text.strip().split(":")[0]
                    relegation = False
                    abstieg = False


                    if dropout_color and dropout_color in str(row).lower():
                        abstieg = True
                    elif "ffcccc" in str(row).lower():
                        abstieg = True

                    if relegation_color and relegation_color in str(row).lower():
                        relegation = True
                    elif "fffccc" in str(row).lower() or "ffffcc" in str(row).lower():
                        relegation = True

                    team = {"position": int(cells[0].text.replace(".", "").strip()), "name": cells[1].text.strip(), "points": int(points), "abstieg": abstieg, "relegation": relegation,
                    ↪  "matches": int(cells[matches_index].text.strip()), "goals": int(goals)}
                    tabelle.append(team)


            # get Torschützenkönig
            if section["line"] == "Torschützenliste":
                # get the html of the section
                html = requests.get(f"https://de.wikipedia.org/w/api.php?action=parse&format=json&page={page}&section={section['index']}").json()["parse"]["text"]["*"]
                # parse the html
                soup = bs4.BeautifulSoup(html, "html.parser")
                # get the table
                table = soup.find("table", {"class": "wikitable"})
                # get the rows of the table
                rows = table.find_all("tr")
                # get the first row of the table
                first_row = rows[0]
                # get the cells of the first row
                cells = first_row.find_all("th")

                # get the data of the goal scorer
                for row in rows[1:]:
                    cells = row.find_all("td")
                    goals = cells[len(cells) - 1].text.strip()
                    results[year]["top goals"] = int(goals)
                    break

        # get average goals for season per match
        url = f"https://de.wikipedia.org/w/api.php?action=parse&format=json&page={page}&prop=text"
        html = requests.get(url).json()["parse"]["text"]["*"]
        soup = bs4.BeautifulSoup(html, "html.parser")
        # find table that contains "infotable" in the class
        table = soup.find("table", {"class": "infobox"})
        # get the rows of the table
        rows = table.find_all("tr")
        total_goals = 0
        total_matches = 0
        # get row that contains "Tore"
        for row in rows:
            cells = row.find_all("td")
            if "Tore" in cells[0].text:
                total_goals = cells[1].text
                total_goals = total_goals.replace(".", "")
                total_goals = total_goals.split(" ")[0]
                total_goals = int(total_goals.strip())

            if "Spiele" in cells[0].text:
                total_matches = cells[1].text
                total_matches = total_matches.replace(".", "")
                total_matches = total_matches.split("+")[0]
                total_matches = int(total_matches.strip())

        results[year]["average_goals"] = total_goals / total_matches

        results[year]["table"] = tabelle

    return results


def percentile(data, x: float):
    # get the x-th percentile of the data
    data = sorted(data)
    left = math.ceil(len(data) * x) # there have to be (at least) this amount of elements to the left of the percentile, including the percentile itself, which means, that left-1 is a valid
    ↪  index for the percentile

    return data[left - 1]

def median(data):
    data = sorted(data)
    if len(data) % 2 == 0:
        return (data[len(data) // 2] + data[len(data) // 2 - 1]) / 2
    else:
        return data[len(data) // 2]

def average(data):
    return sum(data) / len(data)


def boxPlot(data):
    minimum = min(data)
    maximum = max(data)
    q14 = percentile(data, 0.25)
    q34 = percentile(data, 0.75)
    dmedian = median(data)
    daverage = average(data)

    return minimum, maximum, q14, q34, dmedian, daverage


def extractData(data):
    points_of_winners = []
    points_of_dropouts = []
    #avg_goal_per_match = []
    avg_goal_per_season = []
    goals_of_top_scorer = []
    for year in data:
        points_of_winners.append(data[year]["table"][0]["points"])

        highest_points_of_dropouts = 0
        for team in data[year]["table"]:
            if team["abstieg"] or team["relegation"]:
                highest_points_of_dropouts = max(highest_points_of_dropouts, team["points"])

            # avg_goal_per_match.append(team["goals"] / team["matches"])

        points_of_dropouts.append(highest_points_of_dropouts)

        goals_of_top_scorer.append(data[year]["top goals"])

        avg_goal_per_season.append(data[year]["average_goals"])

    return points_of_winners, points_of_dropouts, avg_goal_per_season, goals_of_top_scorer


# main
if __name__ == "__main__":
    # two points for a win 1994
    data1 = get_data(1964, 1994)
    points_of_winners, points_of_dropouts, avg_goal_per_match1, goals_of_top_scorer1 = extractData(data1)

    # create a table of all the boxplot data with pandas
    df = pd.DataFrame({"points of winners": boxPlot(points_of_winners), "points of dropouts": boxPlot(points_of_dropouts)}, index=["minimum", "maximum", "q14", "q34", "median", "average"])

    # swap the columns and rows
    df = df.transpose()
    # print the table
    print("1964-1994")
    print(df.to_string())

    # three points for a win
    data2 = get_data(1995, 2021)
    points_of_winners, points_of_dropouts, avg_goal_per_match2, goals_of_top_scorer2 = extractData(data2)
    # create a table of all the boxplot data with pandas
    df = pd.DataFrame({"points of winners": boxPlot(points_of_winners), "points of dropouts": boxPlot(points_of_dropouts)}, index=["minimum", "maximum", "q14", "q34", "median", "average"])

    # swap the columns and rows
    df = df.transpose()
    # print the table
    print("\n1995-2021")
    print(df.to_string())

    print(f"\n{'-' * 20}\n")
    df = pd.DataFrame({"avg goals per match": boxPlot(avg_goal_per_match1 + avg_goal_per_match2), "goals of top scorer": boxPlot(goals_of_top_scorer1 + goals_of_top_scorer2)},
    ↪  index=["minimum", "maximum", "q14", "q34", "median", "average"])
    df = df.transpose()
    print(df.to_string())
```

```
1964-1994
                    minimum  maximum   q14   q34  median    average
points of winners      41.0     55.0  48.0  51.0    49.0  48.935484
points of dropouts     19.0     31.0  24.0  28.0    26.0  25.548387

1995-2021
                    minimum  maximum   q14   q34  median    average
points of winners      63.0     91.0  70.0  81.0    76.0  76.370370
points of dropouts     27.0     38.0  31.0  36.0    33.0  33.222222

--------------------

                       minimum    maximum        q14        q34     median    average
avg goals per match   2.581699   3.584967   2.859477   3.245098   2.973856   3.056476
goals of top scorer  17.000000  41.000000  22.000000  29.000000  24.000000  25.517241
```

Figure 1: Points of Winners 1964-1994

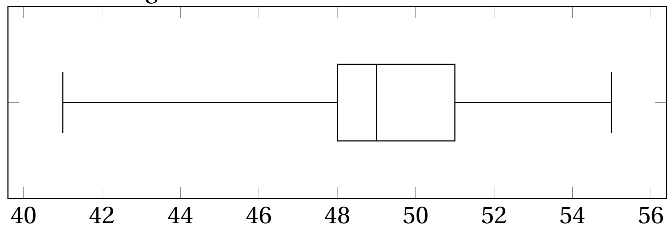


Figure 2: Points of Winners 1995-2021

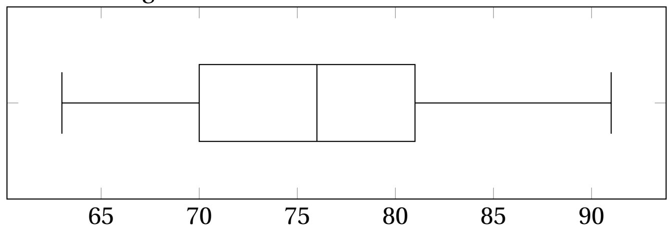


Figure 3: Points of Dropouts 1964-1994

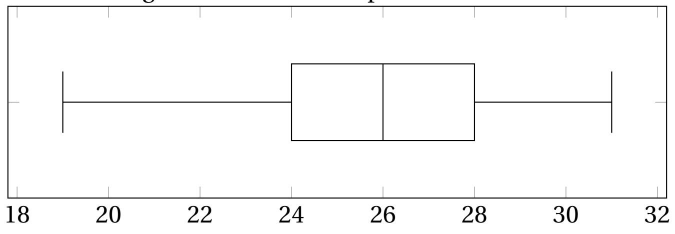


Figure 4: Points of Dropouts 1995-2021

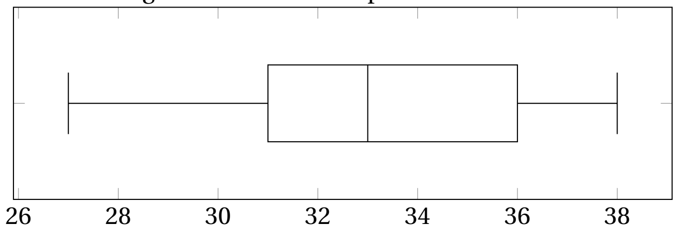


Figure 5: Goals per Match

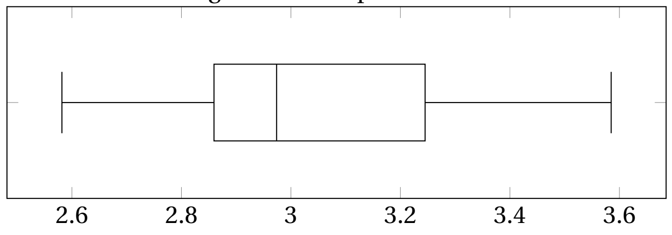
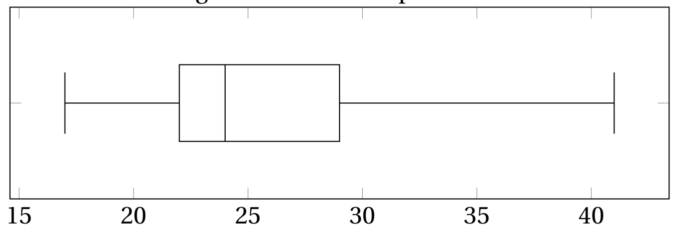


Figure 6: Goals of Top Scorer


15
20
25
30
35
40