Module 3. Data visualization

Unit 3.1 Data visualization

Data visualization represents data and information through visual elements like charts, graphs, maps, and diagrams. It helps transform complex datasets into easily understandable visual formats that reveal patterns, trends, relationships, and insights that might be difficult to spot in raw numbers or text.

Data visualization serves as a bridge between complex data and human understanding. In today's data-driven world, effectively communicating insights through visuals has become increasingly crucial. When data is presented visually, patterns, trends, and outliers become immediately apparent, enabling faster and more informed decision-making.

Unit 3.2 Why data visualization matters

1. Cognitive processing: humans process visual information more efficiently than text or numbers. The human brain can process images in as little as 13 milliseconds, making visualization a powerful tool for quick understanding.

2. Pattern recognition: visual representations help identify trends, correlations, and anomalies that might be invisible in raw data. This pattern recognition capability is essential for both exploratory data analysis and communicating findings.

3. Communication power: visualizations transcend language barriers and technical expertise levels, making complex data accessible to diverse audiences, from technical experts to business stakeholders.

4. Memory retention: people remember visual information better than textual data. Well-designed visualizations can make key insights memorable and actionable.

Unit 3.3 Key aspects of an exemplary data visualization

Good data visualization is built on several key principles that enhance understanding and engagement. Here are the essential aspects.

Clarity and simplicity

- The visualization should communicate its message without unnecessary complexity or decoration.
- Every visual element should serve a purpose - remove chart junk and decorative elements that do not add meaning.

Accurate data representation

- The visualization must accurately reflect the underlying data without distortion.
- Scales and axes should start at zero when appropriate and use consistent intervals.
- The size of visual elements should be proportional to the data they represent.

Effective use of visual elements

- Colors should be chosen purposefully - using appropriate contrasts and considering color blindness.
- Typography should be readable and appropriately sized for hierarchy.
- Spacing and layout should guide the viewer's attention logically through the information.

Proper context

- Include clear titles, labels, and legends that explain what is being shown.
- Provide source information and any necessary data notes.
- Add appropriate annotations to highlight key insights or explain unusual patterns.

Audience consideration

- The visualization should be appropriate for its intended audience's technical expertise and needs.
- Complex concepts should be broken down into digestible components when needed.
- The level of detail and interactivity should match user requirements.

Data-to-ink ratio

- Maximize the ratio of data-carrying ink to total ink used in the visualization.
- Remove redundant elements that don't add new information.
- Keep the focus on the data rather than decorative elements.

Storytelling

- The visualization should help tell a straightforward story about the data.
- Important patterns or insights should be immediately apparent.
- The design should guide viewers toward key takeaways.

Unit 3.4 Common types of data visualizations

1. Distribution visualizations

- Histograms: show the distribution of a single numerical variable.
- Box plots: display the distribution of data through quartiles, helping identify outliers.
- Violin plots: combine Box plot and density plot features to show full distribution shape.

2. Relationship visualizations

- Scatter plots: display relationships between two numerical variables.
- Bubble charts: add a third dimension to scatter plots through point size.
- Heatmaps: show relationships between variables using color intensity.

3. Comparison visualizations

- Bar charts: compare quantities across different categories.
- Line charts: show trends over time or ordered categories.
- Radar charts: compare multiple variables in a radial format.

4. Composition visualizations

1. Pie charts: show parts of a whole for a single category.
2. Stacked bar charts: display parts of a whole over multiple categories.

3. Tree maps: represent hierarchical data through nested rectangles.

5. Spatial visualizations

- Choropleth maps: display variations across geographic regions.
- Point maps: show specific locations or events.
- Connection maps: illustrate relationships between locations.

Unit 3.5 Matching visualizations to data types and questions

In this section, we summarize the optimal visualizations in multiple ways.

1. By data type.
2. By the question type.

3.5.1 By data type

Table 1. By data type

Data type	Visualization type	Example
Categorical data	**Bar charts** for comparing categories	-Sales performance by product category -Shot totals by shot types in soccer
	Pie charts for showing the composition	-Distribution of goals conceded by chance type in soccer
	Heatmaps for showing categorical relationships	-Locations of 3-point shots in an NBA match
Numerical data	**Histograms** for distribution analysis	-Distribution of employee salaries -Distribution of goals per 90 minutes of all the players in the Premier League
	Scatter plots for relationship analysis	-Correlation between house size and price

		-Correlation between #completed passes in the final 3rd and points per game in soccer
	Line charts for trend analysis	-Daily website traffic patterns on www.espn.com -Points per game of a FC Barcelona in La Liga during the 24-25 season
Temporal data	**Area charts** for cumulative trends	-Revenue growth by product category over the years
	Candlestick charts for financial data	-Daily stock price showing open, close, high and low values
Hierarchical data	**Treemaps** for nested categories	-Company structure showing departments, teams, and team sizes
	Network diagrams for complex relationships	-social network connections between users -Pass network diagram between players of the soccer team in a match
Geographical data	**Choropleth maps** for regional comparisons	-Sales density by state or country -Population distribution across regions
	Point maps for location-specific data	-Store locations with performance indicators -Customer incident reports by location -Shot locations of a forward in a soccer match
	Flow maps for movement patterns	-Migration patterns between cities -Supply chain routes and volumes
	Bubble charts for 3-variable analysis	-Countries plotted by GDP (x), Life Expectancy (y), and Population (size)

Multi-dimensional data		-Countries plotted by GDP (x), Life Expectancy (y), and Population (size)
	Radar or spider charts for multi-metric comparison	-Footballer level of skill across multiple metrics -product feature comparison across brands

Source: own elaboration.

3.5.2 By question type

Table 2. By question type

Data type	Questions	Best visualizations
Comparison questions	How do values compare across categories?	-Bar charts -Dot plots -Bullet charts
Relationship questions	How do variables relate to each other?	-Scatter plots -Bubble charts -Heatmaps
Composition questions	What are the parts of the whole? What is the makeup of this item?	-Pie charts -Stacked bar charts -Treemaps
Trend questions	How do values change over time?	-Line charts -Area charts -Sparklines
Distribution questions	What is the spread of my data?	-Histograms -Box plots -Violin plots

Source: own elaboration.

These are quick guidelines to help you choose the most appropriate visualization for the problem you are trying to solve. However, these are not strict rules. Multiple visualization types apply across questions and issues. Sometimes, it also comes down to the audience and the type of visuals they respond to best.

Unit 3.6 Sports-specific visualizations

Sports is a unique domain where questions and data across sports are fundamentally similar in what they capture, such as events on the ball, geographic locations on the pitch, etc.

Here are some standard chart types used heavily in sports.

1. **Shot charts**
    - Basketball: player shooting percentages from different court locations.
    - Soccer: goal attempts and conversion rates from different pitch positions.
    - Hockey: shot frequency and scoring percentage by ice location.

    All these have the soccer pitch, basketball court or hockey rink as the background, with locations mapped to the coordinates on the surface.

2. **Time series analysis (line charts)**
    - Player performance trends across a season.
    - Team win-loss records over multiple seasons.
    - Athlete's speed/stamina metrics during a single game.

3. **Player comparison (radar charts)**
    - Basketball: player stats across categories (points, assists, rebounds, steals, blocks).
    - Soccer: player attributes (pace, shooting, passing, dribbling, defense).
    - Baseball: player metrics (batting average, home runs, RBIs, stolen bases, fielding percentage).

4. **Statistical distributions (Box plots)**

- Team salary distributions in a league.
- Player heights and weights by position.
- Game scores distribution across a season.

5. **Rankings and comparisons (bar charts)**
    - Team standings in a league.
    - Top scorers in a tournament.
    - Win percentages by team.
6. **Player movement (path maps)**
    - Tennis: player court coverage during a match.
    - Soccer: heatmaps of player positions during a game.
    - Basketball: player movement patterns in specific plays.
7. **Game strategy analysis (network diagrams)**
    - Football: pass completion networks between players.
    - Basketball: assist networks among team members.
    - Soccer: pass patterns between positions.
8. **Performance over time (area charts)**
    - Cumulative points scored in a season.
    - Team possession percentage throughout a game.
    - Distance covered by players during a match.
9. **Squad analysis (treemaps)**
    - Team salary distribution by position.
    - Playing time allocation across the roster.
    - Injury time lost by team section.
10. **Match analysis (stacked bar charts)**

- Possession statistics by time.
- Points scored by quarter/period.
- Shot attempts by player/position.

Unit 3.7 Advanced visualization considerations

Interactive visualizations

Interactive visualizations add another dimension to data exploration by allowing users to:

- filter data,
- drill down into details,
- change variables or perspectives,
- highlight specific data points.

Dashboards

Dashboards combine multiple visualizations and link them through filters and parameters to create rich visuals that tell a story and add additional context. However, it is easy to get carried away by adding too much information and interactivity to a dashboard.

When combining multiple visualizations:

- ensure logical layout and flow,
- maintain consistent styling,
- consider user interaction patterns,
- include the appropriate level of detail.

Unit 3.8 Basic visualization examples

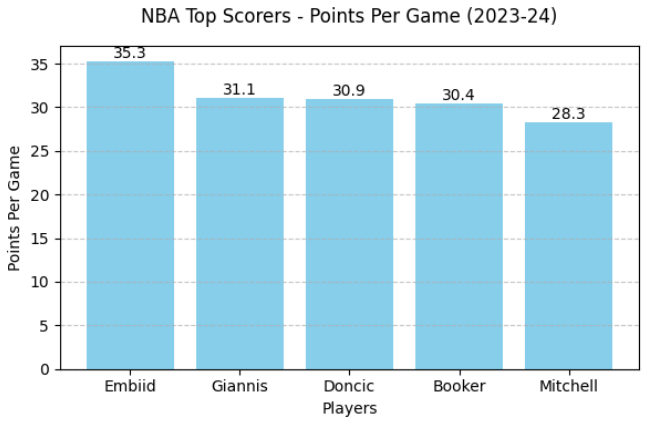
This section will examine how to create some of the most commonly used chart types and visuals in Tableau and Python.

Bar chart

This chart shows the NBA's top scorers from the 23-24 season and their average points per game.

Figure 1. NBA's top scorers

```python
import matplotlib.pyplot as plt

# NBA Top Scorers data (2023-24 season)
players = ['Embiid', 'Giannis', 'Doncic', 'Booker', 'Mitchell']
points_per_game = [35.3, 31.1, 30.9, 30.4, 28.3]

# Create the bar chart
plt.figure(figsize=(10, 6))
bars = plt.bar(players, points_per_game, color='skyblue')

# Customize the chart
plt.title('NBA Top Scorers - Points Per Game (2023-24)', pad=15)
plt.xlabel('Players')
plt.ylabel('Points Per Game')

# Add value labels on top of each bar
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height,
             f'{height:.1f}',
             ha='center', va='bottom')

# Add grid lines for better readability
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Display the chart
plt.tight_layout()
plt.show()
```



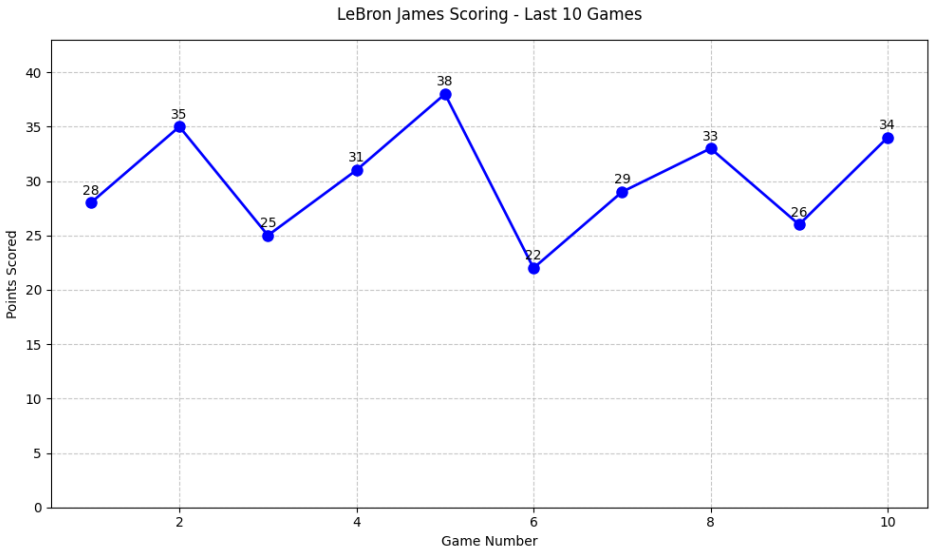
Source: own elaboration.

Line chart

This line chart plots the points scored by LeBron James in his last 10 matches.

Figure 2. Points scored by LeBron James in his last 10 matches

```python
import matplotlib.pyplot as plt

# Data: LeBron James scoring in 10 consecutive games
games = list(range(1, 11))  # Games 1 through 10
points = [28, 35, 25, 31, 38, 22, 29, 33, 26, 34]  # Points scored in each game

# Create the line chart
plt.figure(figsize=(10, 6))
plt.plot(games, points, marker='o', linewidth=2, markersize=8, color='blue')

# Customize the chart
plt.title("LeBron James Scoring - Last 10 Games", pad=15)
plt.xlabel("Game Number")
plt.ylabel("Points Scored")

# Add grid lines
plt.grid(True, linestyle='--', alpha=0.7)

# Add value labels for each point
for i, v in enumerate(points):
    plt.text(games[i], v + 0.5, str(v),
             ha='center', va='bottom')

# Set y-axis range to start from 0
plt.ylim(0, max(points) + 5)

# Display the chart
plt.tight_layout()
plt.show()
```



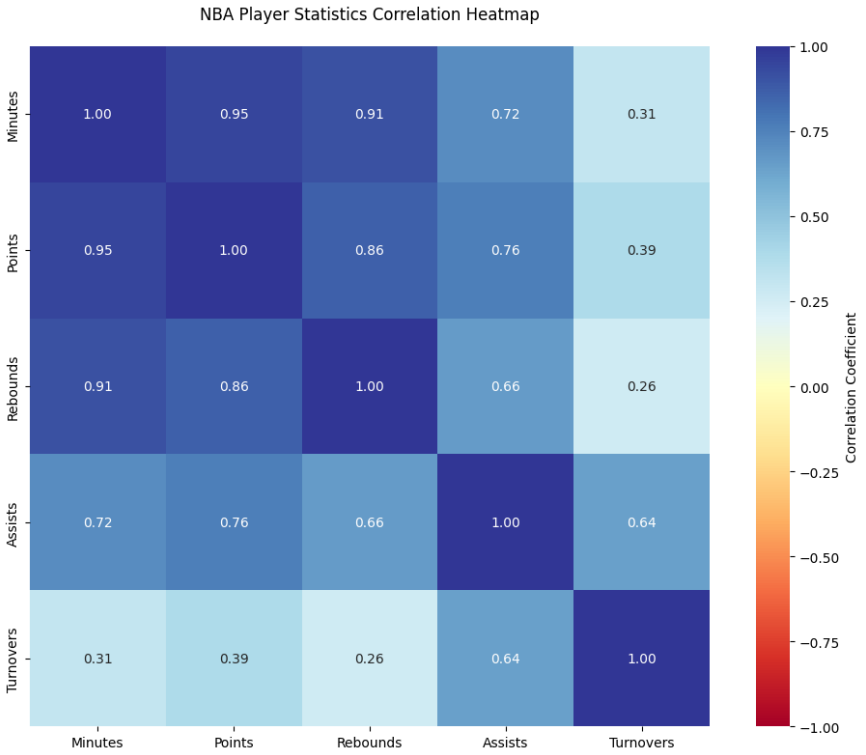
Source: own elaboration.

Heatmap

This chart shows correlations among the top NBA metrics, such as minutes, points, rebounds, assists and turnovers. Please note that this is dummy data, not data from actual matches.

Figure 3. Correlations among the top NBA metrics

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Create sample basketball data with intentional correlations
np.random.seed(42)
n_players = 100

# Base minutes data
minutes = np.random.normal(25, 8, n_players)

# Create correlated stats
points = minutes * 0.8 + np.random.normal(5, 2, n_players)  # Points-minutes
rebounds = minutes * 0.3 + np.random.normal(2, 1, n_players) # Rebounds-minutes
assists = points * 0.2 + np.random.normal(2, 1, n_players)  # Assists-points
turnovers = assists * 0.3 + np.random.normal(1, 0.5, n_players)#Turnoversassists

# Create DataFrame
data = {
    'Minutes': minutes,
    'Points': points,
    'Rebounds': rebounds,
    'Assists': assists,
    'Turnovers': turnovers
}

df = pd.DataFrame(data)

# Calculate correlation matrix
correlation_matrix = df.corr()

# Create heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix,
            annot=True,  # Show correlation values
            cmap='RdYlBu',  # Red-Yellow-Blue colormap
            vmin=-1, vmax=1,  # Set correlation range
            center=0,  # Center the colormap at 0
            fmt='.2f',  # Format correlation values to 2 decimal places
            square=True,  # Make cells square
            cbar_kws={'label': 'Correlation Coefficient'})

# Customize the chart
plt.title('NBA Player Statistics Correlation Heatmap', pad=20)

# Adjust layout
plt.tight_layout()

# Show the plot
plt.show()
```

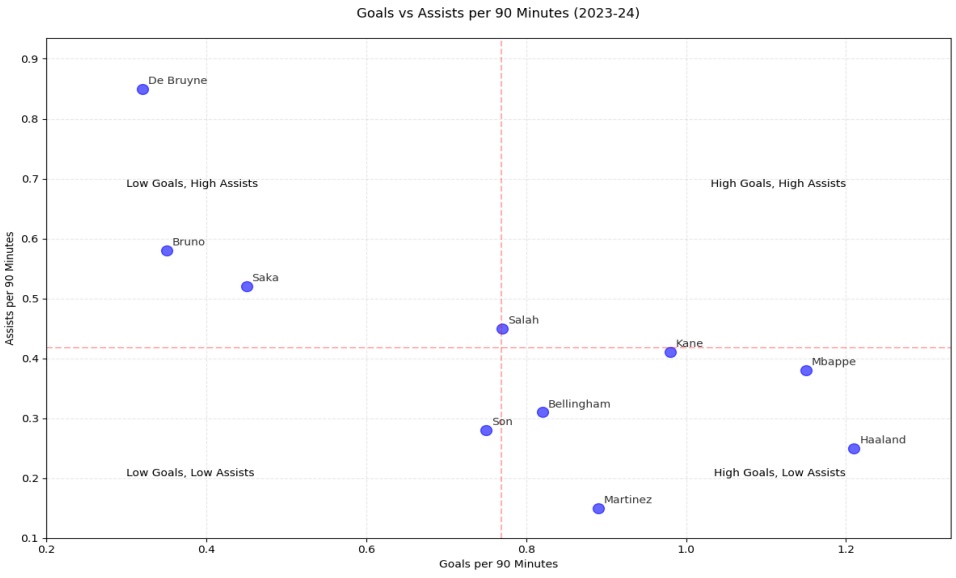


Source: own elaboration.

Scatter plot

This is an example of a scatter plot showing the goals and assists per 90 minutes of some of the top players in the Premier League.

Figure 4. Goals and assists per 90 minutes of some of the top players in the Premier League



Source: own elaboration.

Scatter plot code

Figure 5. Scatter plot code

```
import matplotlib.pyplot as plt
import numpy as np

# Sample data for top players (2023-24 season stats)
players = [
    'Haaland', 'Kane', 'Mbappe', 'Bellingham', 'Salah',
    'Martinez', 'Son', 'Saka', 'De Bruyne', 'Bruno'
]
# Goals and assists per 90 minutes
goals_per_90 = [1.21, 0.98, 1.15, 0.82, 0.77, 0.89, 0.75, 0.45, 0.32, 0.35]
assists_per_90 = [0.25, 0.41, 0.38, 0.31, 0.45, 0.15, 0.28, 0.52, 0.85, 0.58]

# Create the scatter plot
plt.figure(figsize=(12,8))

# Create scatter plot with custom styling
plt.scatter(goals_per_90, assists_per_90,
           c='blue',
           alpha=0.6,
           s=100)  # Size of points

# Add labels for each point
for i, player in enumerate(players):
    plt.annotate(player,
                 (goals_per_90[i], assists_per_90[i]),
                 xytext=(5, 5),  # 5 points offset
                 textcoords='offset points',
                 fontsize=10,
                 alpha=0.8)
# Customize the chart
plt.title('Goals vs Assists per 90 Minutes (2023-24)', pad=20)
plt.xlabel('Goals per 90 Minutes')
plt.ylabel('Assists per 90 Minutes')
# Add grid for better readability
plt.grid(True, linestyle='--', alpha=0.3)
# Set axis ranges with some padding
plt.xlim(0.2, max(goals_per_90) * 1.1)
plt.ylim(0.1, max(assists_per_90) * 1.1)
# Add average lines
plt.axvline(x=np.mean(goals_per_90), color='r', linestyle='--', alpha=0.3)
plt.axhline(y=np.mean(assists_per_90), color='r', linestyle='--', alpha=0.3)

# Add text for quadrant labels
plt.text(1.2, 0.7, 'High Goals, High Assists', horizontalalignment='right',
         verticalalignment='top')
plt.text(0.3, 0.7, 'Low Goals, High Assists', horizontalalignment='left',
         verticalalignment='top')
plt.text(1.2, 0.2, 'High Goals, Low Assists', horizontalalignment='right',
         verticalalignment='bottom')
plt.text(0.3, 0.2, 'Low Goals, Low Assists',horizontalalignment='left',
         verticalalignment='bottom')

# Display the plot
plt.tight_layout()
plt.show()
```

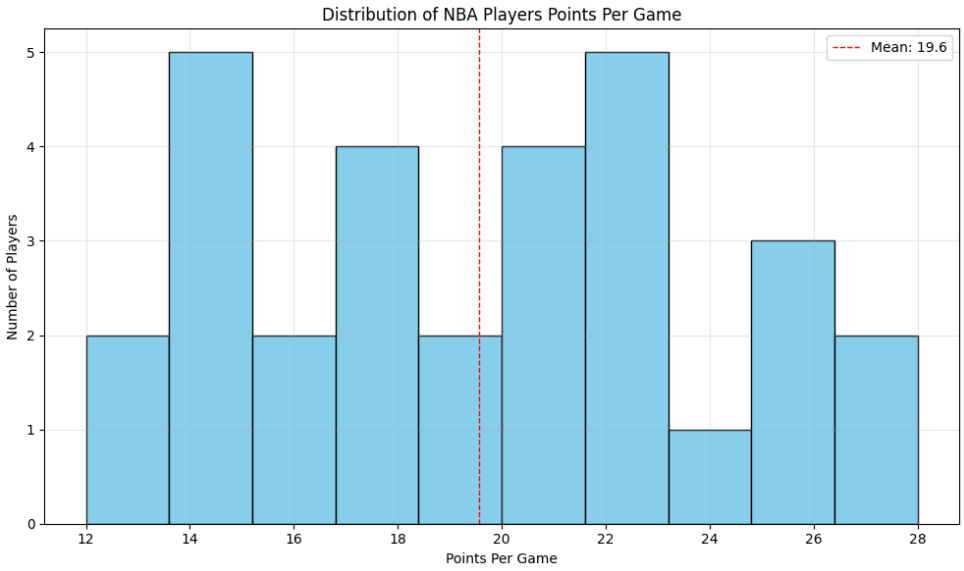
Source: own elaboration.

Histogram

A histogram showing the distribution of players' scoring averages. Redline the mean/average.

Figure 6. Histogram showing the distribution of players' scoring averages

```
import matplotlib.pyplot as plt
import numpy as np

# Create sample data - NBA player points per game
points_per_game = [
    18, 22, 15, 28, 12, 25, 19, 21, 16, 23,
    17, 20, 14, 26, 13, 24, 18, 22, 15, 27,
    16, 21, 19, 23, 15, 25, 17, 20, 14, 22
]

# Create histogram
plt.figure(figsize=(10, 6))
plt.hist(points_per_game,
         bins=10,  # Number of bins
         color='skyblue',
         edgecolor='black')

# Add labels and title
plt.title('Distribution of NBA Players Points Per Game')
plt.xlabel('Points Per Game')
plt.ylabel('Number of Players')

# Add grid for better readability
plt.grid(True, alpha=0.3)

# Display mean line
plt.axvline(np.mean(points_per_game),
            color='red',
            linestyle='dashed',
            linewidth=1,
            label=f'Mean: {np.mean(points_per_game):.1f}')
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()
```



Source: own elaboration.

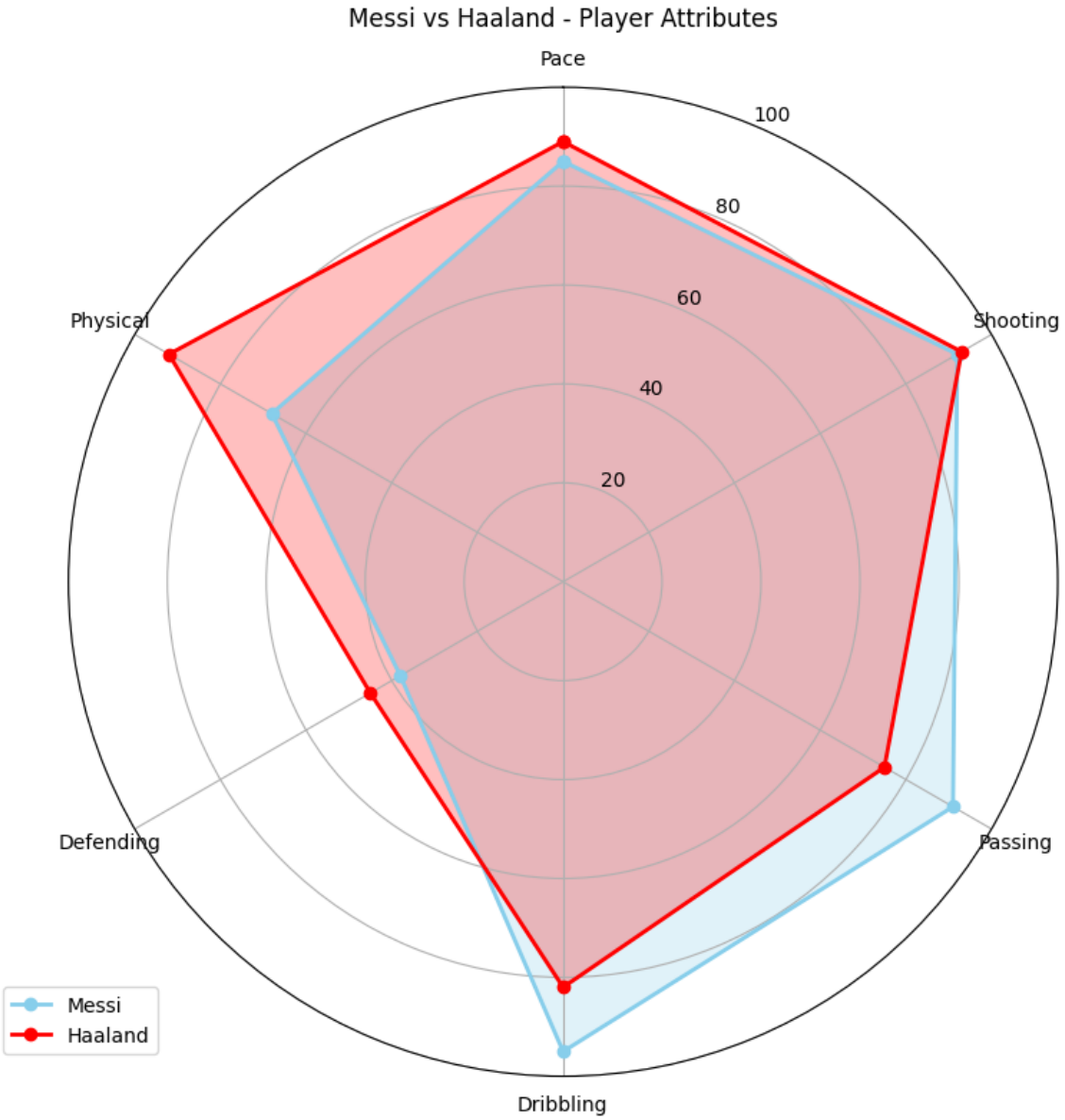
Unit 3.9 Sports-specific visualizations

While we use basic visualizations extensively in sports, some are very specific to sports and are used heavily in player and team analyses.

Radar chart

Here is a basic radar chart comparing two players across various metrics. This is an illustrative example between Messi and Haaland.

Figure 7. Basic radar chart comparing two players across various metrics



Source: own elaboration.

Here is the code to produce this visual. You can compare more than two or have one category(players in this case). You can also play with the colors and the number of metrics.

Figure 8. Code to produce the radar chart

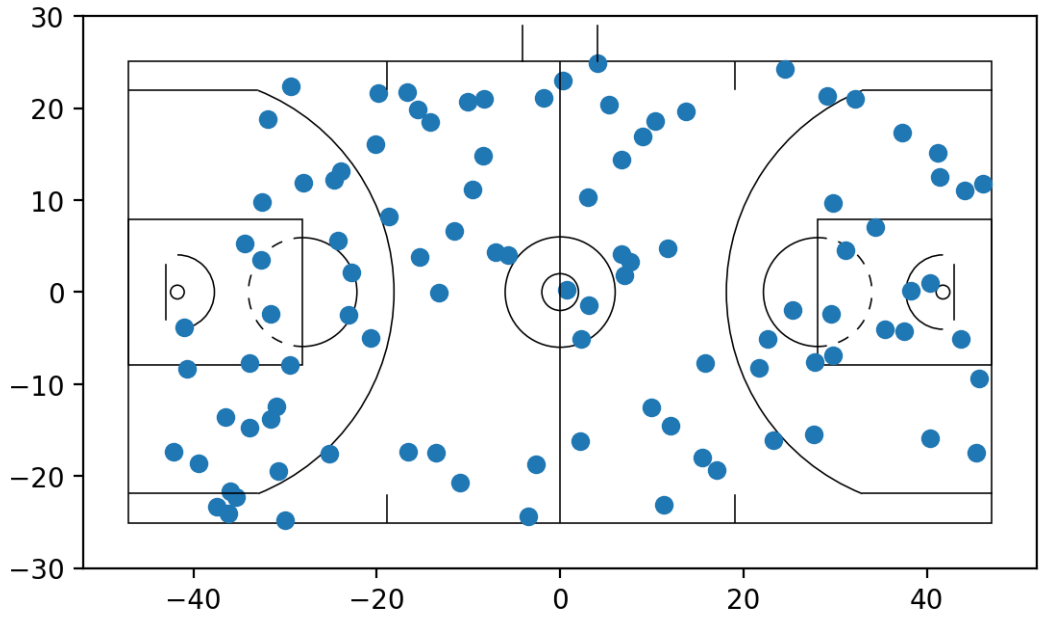
```
import numpy as np
import matplotlib.pyplot as plt

# Categories for comparison
categories = ['Pace', 'Shooting', 'Passing',
              'Dribbling', 'Defending', 'Physical']

# Player stats (on a scale of 0-100)
messi = [85, 92, 91, 95, 38, 68]
haaland = [89, 93, 75, 82, 45, 92]

# Number of categories
num_cats = len(categories)

# Compute angle for each axis
angles = [n / float(num_cats) * 2 * np.pi for n in range(num_cats)]
angles += angles[:1]

# Initialize the figure
fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(projection='polar'))

# Add the first player's stats
stats1 = messi + [messi[0]]
ax.plot(angles, stats1, 'o-', linewidth=2, label='Messi', color='skyblue')
ax.fill(angles, stats1, alpha=0.25, color='skyblue')

# Add the second player's stats
stats2 = haaland + [haaland[0]]
ax.plot(angles, stats2, 'o-', linewidth=2, label='Haaland', color='red')
ax.fill(angles, stats2, alpha=0.25, color='red')

# Fix axis to go in the right order and start at 12 o'clock
ax.set_theta_offset(np.pi / 2)
ax.set_theta_direction(-1)

# Draw axis lines for each angle and label
ax.set_xticks(angles[:-1])
ax.set_xticklabels(categories)

# Add legend
plt.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1))

# Add a title
plt.title("Messi vs Haaland - Player Attributes", y=1.05)

# Add gridlines and set scale
ax.set_ylim(0, 100)
ax.grid(True)

# Display the plot
plt.tight_layout()
plt.show()
```

Source: own elaboration.

Shot chart

Here is an example of an NBA shot chart. This is using random data.

Figure 9. Example of an NBA shot chart



Source: own elaboration.

Figure 10. Code for the NBA shot chart

```
import matplotlib.pyplot as plt
from mplbasketball.court import Court
import numpy as np

# Create figure and axis with white background
#plt.style.use('white')
fig = plt.figure(figsize=(12, 6))

# Create and draw the court
# Pass 'nba' as the first argument to specify the court type
court = Court(court_type="nba", origin="center", units="ft")
fig, ax = court.draw(showaxis=True)

n_points = 100
x = np.random.uniform(-47, 47, size=n_points)
y = np.random.uniform(-25, 25, size=n_points)

ax.scatter(x, y)
```

Source: own elaboration.

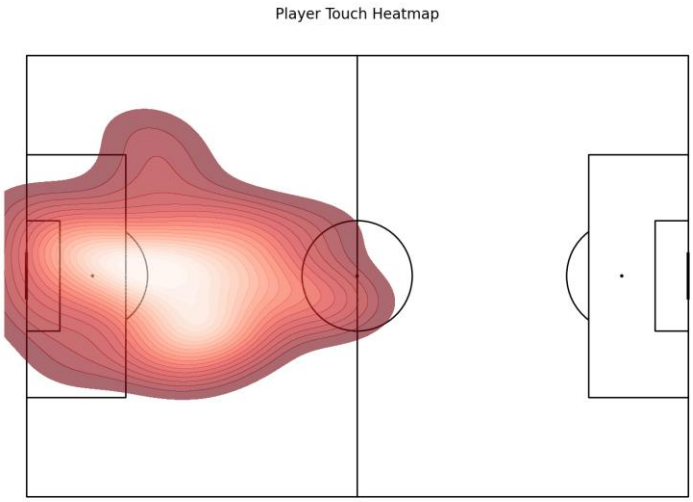
We are using the package **MPLBasketball** to draw the above map. The package is compelling, and you can learn more about it at https://pypi.org/project/mplbasketball/

Heatmap

MPLSoccer is a very popular soccer package that lets you create custom visuals specific to soccer. You can find more information and examples on how to use it at the link below: https://mplsoccer.readthedocs.io/en/latest/index.html.

We will use the MPLSoccer package to generate a player touch heatmap (also known as a density map).

Figure 11. Soccer heatmap



```python
import numpy as np
import matplotlib.pyplot as plt
from mplsoccer import Pitch
import seaborn as sns

def create_heatmap(touches_data):
    """
    Create a heatmap of player touches on a soccer pitch using mplsoccer

    Parameters:
    touches_data: List of tuples [(x1, y1), (x2, y2), ...]
    containing touch coordinates
    """
    # Create pitch
    pitch = Pitch(pitch_type='statsbomb',
                  pitch_color='white', line_color='black')

    # Create figure and axis with pitch
    fig, ax = pitch.draw(figsize=(15, 10))

    # Convert touch data to numpy arrays
    x = [touch[0] for touch in touches_data]
    y = [touch[1] for touch in touches_data]

    # Create heatmap using kernel density estimation
    sns.kdeplot(
        x=x,
        y=y,
        cmap='Reds_r',
        fill=True,
        alpha=0.6,
        levels=20,
        ax=ax
    )

    # Set title
    ax.set_title('Player Touch Heatmap', color='black', size=20, pad=20)

    return fig
# Example usage
if __name__ == "__main__":
    # Sample touch data (x, y coordinates)
    # This simulates a left-wing player's touches
    np.random.seed(42)

    # Generate some random touches with bias towards left wing
    n_touches = 50
    x_touches = np.random.normal(30, 15, n_touches)
    y_touches = np.random.normal(40, 10, n_touches)

    # Combine into list of tuples
    touch_data = list(zip(x_touches, y_touches))

    # Create and display the heatmap
    fig = create_heatmap(touch_data)
    plt.show()
```

Source: own elaboration.

Unit 3.10 Top visualization libraries in Python

Matplotlib – This is the foundation of plotting in Python.

- Highly customizable and detailed control over every aspect of plots.
- Great for publication-quality figures and scientific visualization.
- Steeper learning curve but highly versatile and powerful.
- Works well with NumPy arrays and Pandas DataFrames.

Seaborn – Seaborn specializes in statistical visualization based on Matplotlib.

- Built for statistical graphics and attractive default styles.
- Excellent for statistical visualizations like distribution plots, heatmaps.
- Integrates smoothly with Pandas DataFrames.
- Simpler API than Matplotlib while maintaining flexibility.

Plotly – Plotly is best for interactive web-based visualizations.

- Creates interactive plots that work well in web browsers and Jupyter notebooks.
- Strong support for financial plots and scientific charts.
- Good for creating dashboards and web applications.
- Has both Python and JavaScript APIs.

Altair - Declarative statistical visualization.

- Based on Vega and Vega-Lite specifications.
- Concise, clear syntax for complex visualizations.
- Excellent for data exploration and analysis.
- Strong support for interactive visualizations.

Bokeh - Another powerful interactive visualization library.

- Creates web-ready visualizations with JavaScript backend.
- Good for building dashboards and applications.

- Handles large datasets well.
- Strong support for streaming data and real-time updates.

Unit 3.11 Key sports data visualization packages in Python

MPLSoccer – https://mplsoccer.readthedocs.io/en/latest/index.html.

MPLBasketball – https://pypi.org/project/mplbasketball/

PySport—https://opensource.pysport.org/—PySport has a list of dozens of open-source sports data analytics and visualization Python packages for a wide variety of sports.

Unit 3.12 Visualizations in Tableau

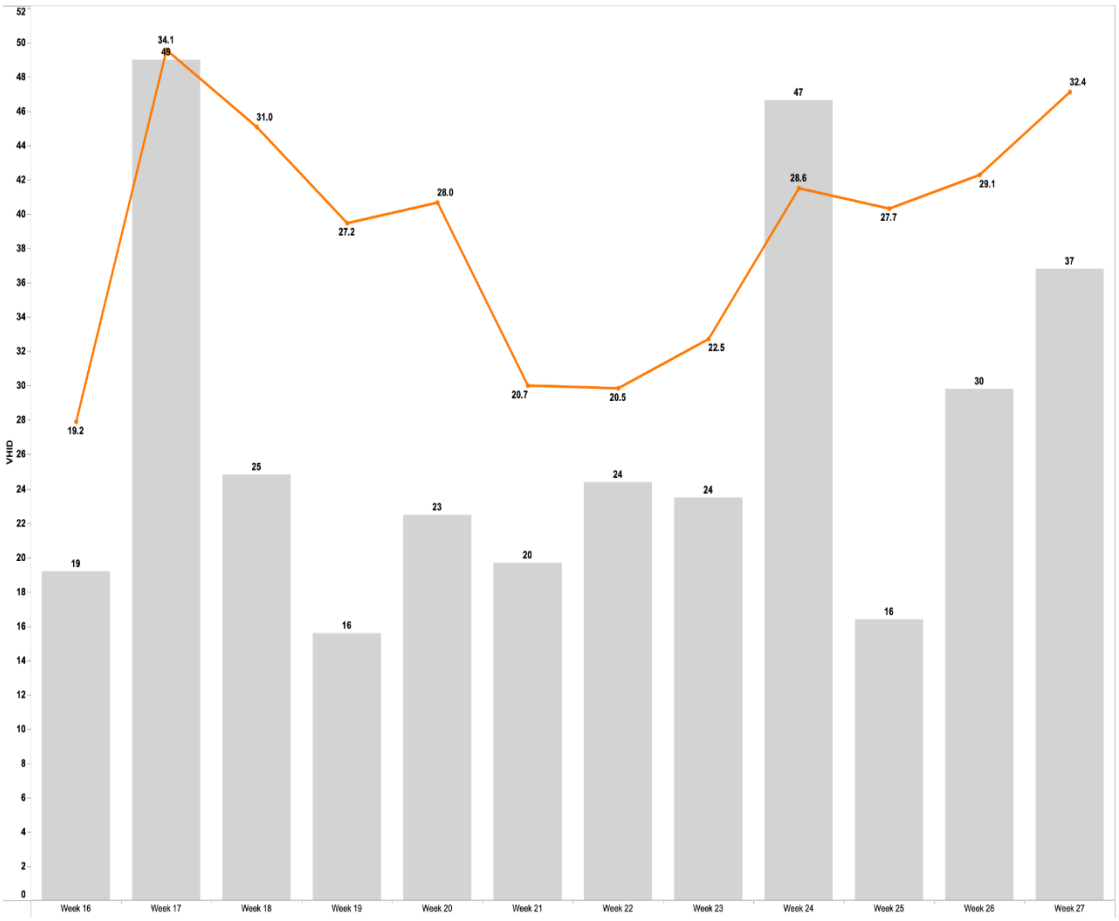
Almost all of the visuals we learned to make in Python in sections 3.8 and 3.9 can also be created in Tableau software. Tableau has a free public version where you can create and share visuals publicly.

Link to Tableau Public - Tableau Public

1) Bar chart with dual axis

One of the best ways to see an overlapping trend is by merging two data points or charts. The chart below shows the daily value with an acute-to-chronic ratio of 1:4 weeks of data points. This helps the coaches understand the trend and how athletes respond to it.

Figure 12. Bar chart with dual axis



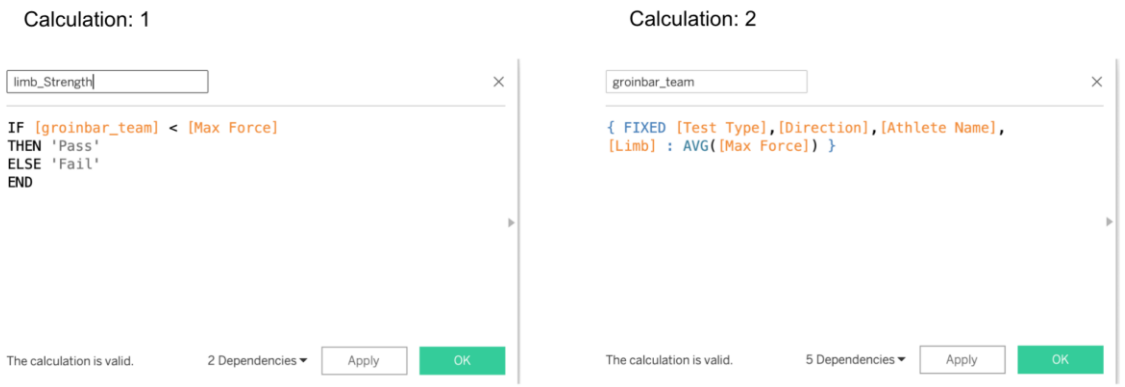
Source: own elaboration.

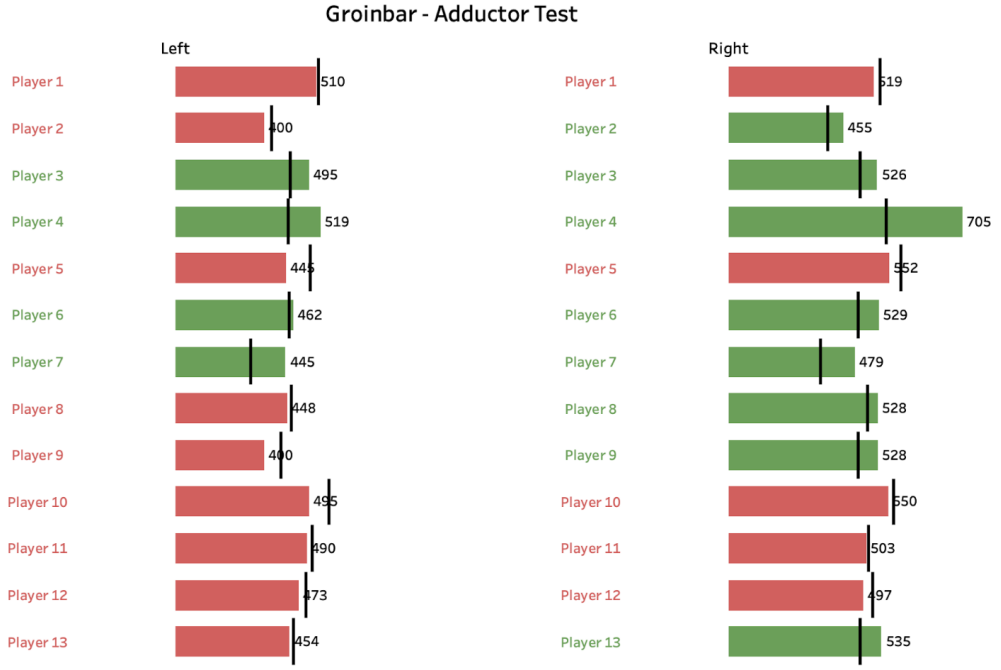
The previous chart highlight (very high-speed running - VHID > 7m/s) showing acute to chronic ratio, weekly total in the bar graph, and line representation of the average for the last 4 weeks.

2) Bar chart with an average line

This chart shows how athletes correspond to strength training in the gym for off-field strength and conditioning.

Figure 13. Bar chart with an average line



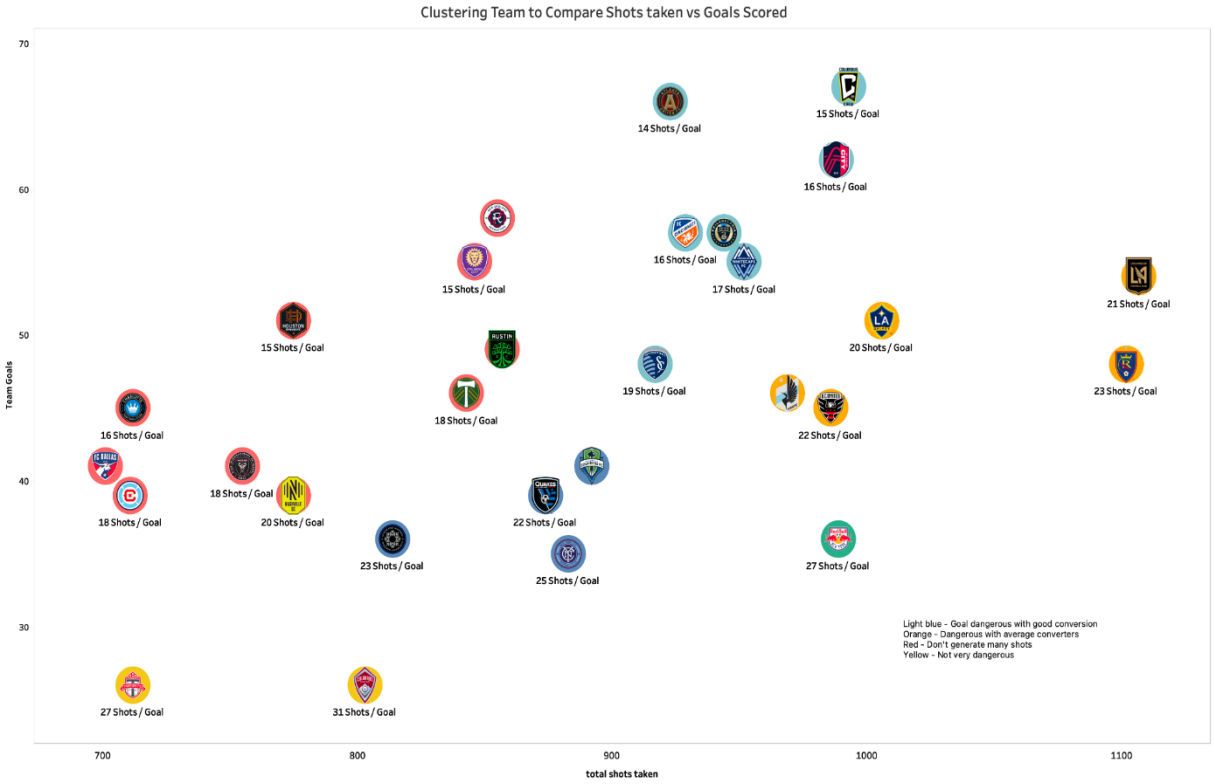


Source: own elaboration.

3) Scatter plot

Tableau inbuilt cluster analysis (k-mean). The chart highlights the shots-to-goals scored ratio, comparing how many shots a team took to score 1 goal.

Figure 14. Scatter plot



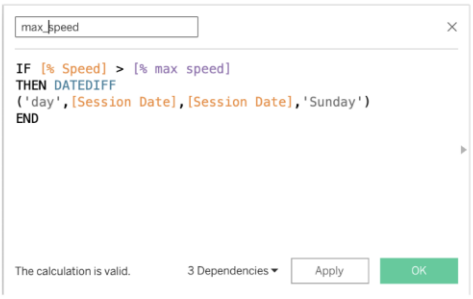
Source: own elaboration.

3) Using Tableau calculations

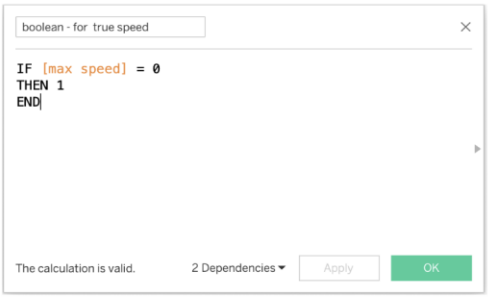
Tableau calculations give much flexibility when it comes to building a visualization. The visual below highlights the percent of max speed a player has achieved, plus how many days have passed since the last max exposure. As hamstring injuries are prevalent in football, these visuals help you digest the information quickly. I have highlighted all the essential calculations one needs to add besides clicking on the drop in the tableau to make the visual.

Figure 15. Tableau calculations

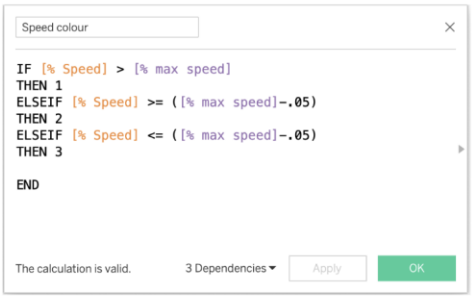
Calculation: 1



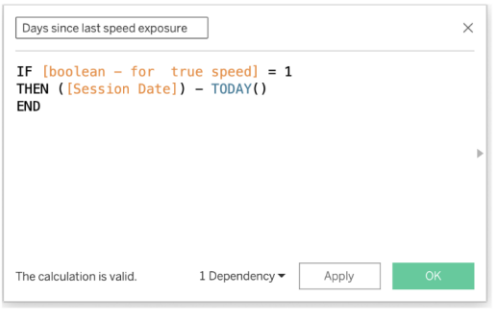
Calculation: 2



Calculation: 3



Calculation: 4



Days since the Selected speed exposure

Session Date

Player Name	November 15, 2023	November 16, 2023	November 17, 2023	November 18, 2023	November 19, 2023	November 22, 2023	November 23, 2023	November 24, 2023	November 25, 2023	November 26, 2023
Player 1	11 days 96%		days 72%	days 64%	days 82%	days 68%	days 83%	days 78%	days 56%	days 86%
Player 2	11 days 95%		days 88%	days 89%	7 days 94%	days 77%	3 days 90%	days 85%	days 75%	0 days 100%
Player 3	11 days 91%		days 80%	days 72%	7 days 96%	days 67%	days 89%	days 81%	days 53%	0 days 96%
Player 4	11 days 95%		days 82%	days 70%	days 89%	days 82%	3 days 95%	days 74%	days 75%	days 69%
Player 5	11 days 93%		days 79%	days 78%	7 days 98%	days 85%	days 85%	days 88%	days 83%	days 44%
Player 6	11 days 93%			days 73%	days 85%	days 78%	3 days 91%	days 74%	days 69%	0 days 95%
Player 7	11 days 92%		days 82%	days 68%	7 days 91%	days 76%	days 89%	days 89%	days 78%	days 88%
Player 8	11 days 94%		days 77%	days 64%	7 days 92%	days 75%	3 days 93%	days 80%	days 65%	0 days 96%
Player 9	days 71%	days 81%	days 79%			days 87%	days 87%	days 75%	days 75%	0 days 99%
Player 10	11 days 96%		days 86%	days 68%	7 days 97%	days 70%	3 days 95%	days 80%	days 69%	
Player 11	11 days 95%		days 79%	days 69%	7 days 98%	days 72%	3 days 91%	days 77%	days 73%	days 82%
Player 12	11 days 98%		days 76%	days 68%	7 days 94%	days 87%	3 days 99%	days 75%	days 73%	days 66%
Player 13	days 81%		days 66%	days 61%	7 days 95%	days 72%	days 83%	days 64%	days 72%	0 days 91%

Source: own elaboration.

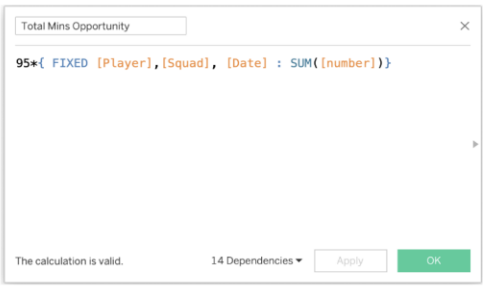
4) Advanced visual

How can teams look into the minutes players per player? Or how many players take up 70-80% of the available team minutes in a season?

Here is a quick look at the formulas needed to build this visual.

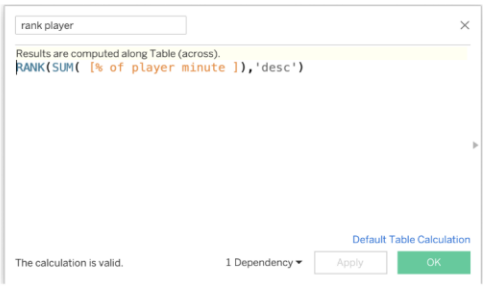
Figure 16. Advanced visual

Calculation: 1

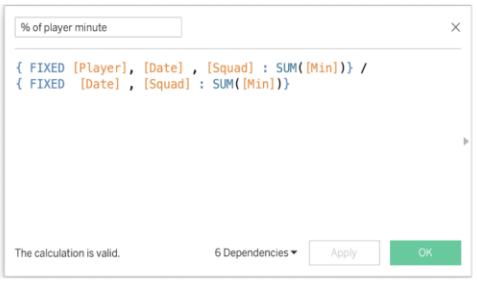


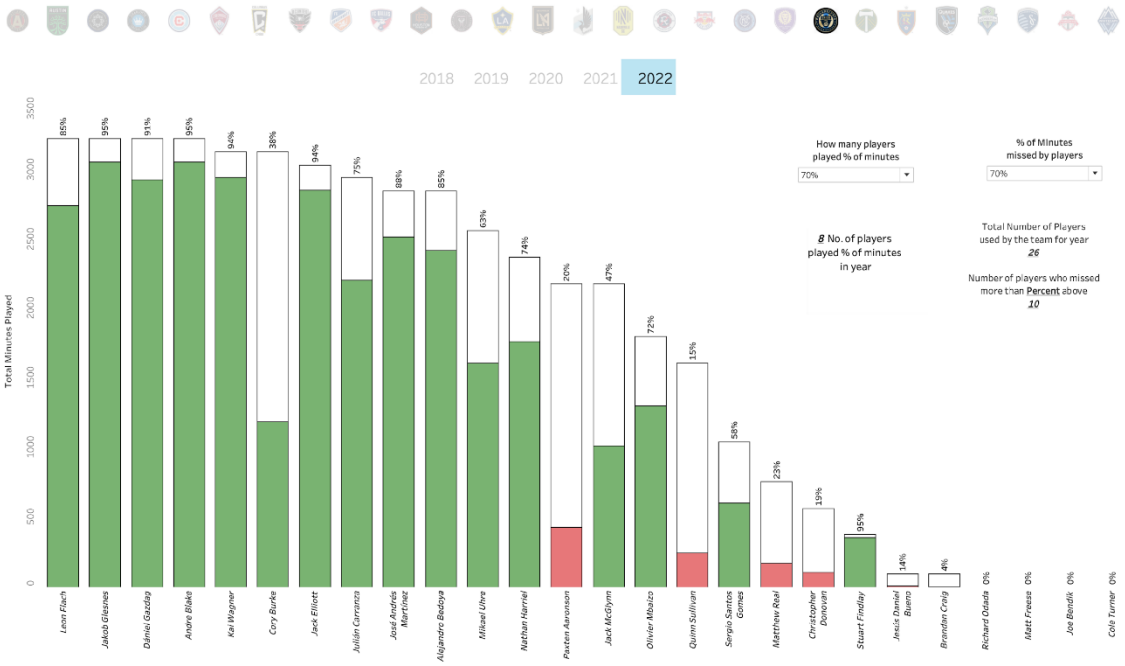
Calculation: 2

Calculation: 3



Calculation: 4





Source: own elaboration.

The white solid bars represent the minutes available for the player and the green bars show the minutes played by every player.

Conclusion

Effective data visualization is both an art and a science. The key to success lies in understanding your data, your audience, and the story you want to tell. By matching the correct visualization to your data type and question, you can create powerful visual narratives that drive understanding and decision-making. Remember that the best visualization is often the simplest one that effectively communicates your message.

References

Mobifly. (n.d.). Graphic Designs. 5 Basic Elements of Good Visual Design for Website and Mobile Apps. –https://mobifly.tech/category/graphic-designs/

Woopra. (n.d.). Data Analytics and Visualization: A Comprehensive Guide to Transforming Raw Data into Strategic Insights.https://www.woopra.com/blog/data-analytics-and-visualization

Yalcin, O. (2023). Cutting Costs and Boosting Efficiency with Data Visualization Tools. *Vizio.AI*. https://www.vizio.ai/blog/cutting-costs-and-boosting-efficiency-with-data-visualization-tools