# vis-cricket

November 24, 2019

[3]:
```
import pandas as pd
import plotly.figure_factory as ff
import plotly.graph_objects as go
import plotly.io as pio
import math
# renderer for jupyter notebook
pio.renderers.default='notebook'
%%latex
```

```
UsageError: Line magic function `%%latex` not found.
```

[17]:
```
pio.templates.default = "plotly_dark"
```

[18]:
```
df_scorecard=pd.read_csv(r'./full/odi_scorecard.csv')
```

[19]:
```
df_info=pd.read_csv(r'./full/odi_info.csv')
```

[20]:
```
df_scorecard_agg=pd.read_csv(r'./full/odi_scorecard_agg.csv')
```

[21]:
```
df_total=pd.merge(df_info,df_scorecard,on='match-id')
```

[22]:
```
df_total_agg=pd.merge(df_info,df_scorecard_agg,on='match-id')
```

[23]:
```
df_info.columns
```

[23]:
```
Index(['city', 'competition', 'date', 'match-id', 'gender', 'match-number',
       'match-referee', 'method', 'neutralvenue', 'outcome', 'player-of-match',
       'reserve-umpire', 'season', 'series', 'team-0', 'team-1',
       'toss-decision', 'toss-winner', 'tv-umpire', 'umpire-0', 'umpire-1',
       'venue', 'winner', 'winner-runs', 'winner-wickets', 'year'],
      dtype='object')
```

[24]:
```
df_scorecard_agg.columns
```

[24]:
```
Index(['match-id', 'team-name', 'batting-position', 'over-batsman',
       'runs-scored', 'balls-played', 'dots', 'ones', 'twos', 'threes',
       'fours', 'sixes', 'balls-bowled', 'maiden-overs', 'runs-given',
       'wickets', 'extras', 'fall-of-wicket-score', 'fall-of-wicket-overs',
       'fall-of-wicket-no'],
      dtype='object')
```

[25]:
```
df_scorecard.columns
```

[25]:
```
Index(['match-id', 'team-name', 'innings', 'name', 'batting-position',
       'over-batsman', 'runs-scored', 'balls-played', 'dots', 'ones', 'twos',
       'threes', 'fours', 'sixes', 'wicket-method', 'balls-bowled',
       'maiden-overs', 'runs-given', 'wickets', 'extras',
       'fall-of-wicket-score', 'fall-of-wicket-overs', 'fall-of-wicket-no',
       'fall-of-wicket-bowler'],
      dtype='object')
```

## 0.1 Visualizations

### 0.1.1 Wickets

**Wickets Methods**

[175]:
```
wickets_method=df_scorecard[df_scorecard['wicket-method']!='0']['wicket-method']
```

[176]:
```
wicket_method=wickets_method.value_counts().index
wicket_method_value=wickets_method.value_counts().values
fig=go.Figure(data=[go.
 ↪Bar(x=wicket_method,y=wicket_method_value,text=wicket_method_value,textposition='auto')],la
 ↪of wickets per dismissal method")
fig.update_layout(xaxis_title='Dismissal Method',yaxis_title='Count')
fig.show()
```

**Dismissal Method Distribution**

[177]:
```
fig=go.Figure()
fig.add_trace(go.Pie(labels=wicket_method,values=wicket_method_value))
fig.update_layout(title='Dismissal method distribution')
fig.show()
```

**Fall of Wicket by Runs**

[27]:
```
fow_score=df_scorecard[(df_scorecard['fall-of-wicket-score']!
 ↪=0)]['fall-of-wicket-score'].
 ↪append(df_scorecard[(df_scorecard['fall-of-wicket-score']==0) &␣
 ↪df_scorecard['fall-of-wicket-no']!=0]['fall-of-wicket-score'])
```

[28]:
```
fig = go.Figure(data=[go.Histogram(x=fow_score)])
fig.update_layout(title='Fall of wicket by␣
 ↪runs',xaxis_title='Runs',yaxis_title='Wickets Fallen')
fig.show()
```

**Probability Distribution of fall of wicket by runs**

[29]:
```
fig=ff.create_distplot([fow_score],group_labels=['Fall of wicket Runs'])
fig.update_layout(title='Probability Distribution of wickets fall by␣
 ↪runs',xaxis_title='runs',yaxis_title='Probability')
fig.show()
```

**Fall of Wickets by overs**

[143]:
```
fow_overs=df_scorecard[df_scorecard['fall-of-wicket-overs']>0.
 ↪0]['fall-of-wicket-overs'].apply(lambda x:str(x).split('.')[0])
```

[144]:
```
fig = go.Figure(data=[go.Histogram(x=fow_overs)])
fig.show()
```

**Probability distribution of Fall of wickets by overs**

[145]:
```
fow_overs=fow_overs.astype('int64')
```

[146]:
```
fig=ff.create_distplot([fow_overs],group_labels=['Fall of wicket Overs'])
fig.update_layout(title='Probability Distribution of wickets fall by␣
 ↪over',xaxis_title='Over',yaxis_title='Probability')
fig.show()
```

### 0.1.2 Team Statistics

**Teamwise runs scored**

[147]:
```
team_scores={team:[] for team in df_scorecard_agg['team-name']}
for index,row in df_scorecard_agg.iterrows():
    team_scores[row['team-name']].append(row['runs-scored'])
fig=go.Figure()
for index, value in team_scores.items():
    fig.add_trace(go.Box(y=value,name=index,boxmean='sd'))
fig.update_layout(title='Teamwise Runs scored',xaxis_title='Team␣
 ↪Name',yaxis_title='Runs Scored')
fig.show()
```

**Teamwise Wickets Taken**

[148]:
```
team_scores={team:[] for team in df_scorecard_agg['team-name']}
for index,row in df_scorecard_agg.iterrows():
    team_scores[row['team-name']].append(row['wickets'])
fig=go.Figure()
for index, value in team_scores.items():
    fig.add_trace(go.Box(y=value,name=index,boxmean='sd'))
fig.update_layout(title='Teamwise Wickets Taken',xaxis_title='Team␣
 ↪Name',yaxis_title='Wickets Taken Scored')
fig.show()
```

**Teamwise Extras Given**

[14]:
```
team_scores={team:[] for team in df_scorecard_agg['team-name']}
for index,row in df_scorecard_agg.iterrows():
    team_scores[row['team-name']].append(row['extras'])
fig=go.Figure()
for index, value in team_scores.items():
```

```
    fig.add_trace(go.Box(y=value,name=index,boxmean='sd'))
fig.update_layout(title='Teamwise Extras Given',xaxis_title='Team␣
 ↪Name',yaxis_title='Wickets Taken Scored')
fig.show()
```

**Team performance over the years**

[20]:
```
year=[]
team=[]
matches=[]
team_year_wise_total=df_info.groupby('year')['team-0'].value_counts()
for index,value in team_year_wise_total.iteritems():
    year.append(index[0])
    team.append(index[1])
    matches.append(value)
temp1=pd.DataFrame({'year':year,'team':team,'matches0':matches})
year=[]
team=[]
matches=[]
team_year_wise_total=df_info.groupby('year')['team-1'].value_counts()
for index,value in team_year_wise_total.iteritems():
    year.append(index[0])
    team.append(index[1])
    matches.append(value)
temp2=pd.DataFrame({'year':year,'team':team,'matches1':matches})
year=[]
team=[]
wins=[]
team_year_wise_wins=df_info.groupby('year')['winner'].value_counts()
for index,value in team_year_wise_wins.iteritems():
    year.append(index[0])
    team.append(index[1])
    wins.append(value)
temp3=pd.DataFrame({'year':year,'team':team,'wins':wins})
df_matches_year=pd.merge(temp1,temp2,on=['year','team'])
df_matches_year=pd.merge(df_matches_year,temp3,on=['year','team'])
```

[12]:
```
df_matches_year['matches']=df_matches_year['matches0']+df_matches_year['matches1']
df_matches_year['win-ratio']=round(df_matches_year['wins']/
 ↪df_matches_year['matches'],3)
```

[13]:
```
df_matches_year_dict={i:{'year':[],'ratio':[]} for i in df_matches_year['team'].
 ↪unique()}
for index,row in df_matches_year.iterrows():
    df_matches_year_dict[row['team']]['year'].append(row['year'])
    df_matches_year_dict[row['team']]['ratio'].append(row['win-ratio'])
```

[14]:
```
fig=go.Figure()
for i in df_matches_year['team'].unique():
    fig.add_trace(go.
 ↪Scatter(x=df_matches_year_dict[i]['year'],y=df_matches_year_dict[i]['ratio'],name=i))
fig.update_layout(
    title_text='Win ratio of teams in ODI yearwise',
    xaxis_rangeslider_visible=True
)
fig.show()
```

**Team wise performance over different venues**

[76]:
```
venue=[]
team=[]
matches=[]
team_year_wise_total=df_info.groupby('venue')['team-0'].value_counts()
for index,value in team_year_wise_total.iteritems():
    venue.append(index[0])
    team.append(index[1])
    matches.append(value)
temp1=pd.DataFrame({'venue':venue,'team':team,'matches0':matches})
venue=[]
team=[]
matches=[]
team_year_wise_total=df_info.groupby('venue')['team-1'].value_counts()
for index,value in team_year_wise_total.iteritems():
    venue.append(index[0])
    team.append(index[1])
    matches.append(value)
temp2=pd.DataFrame({'venue':venue,'team':team,'matches1':matches})
venue=[]
team=[]
wins=[]
team_year_wise_wins=df_info.groupby('venue')['winner'].value_counts()
for index,value in team_year_wise_wins.iteritems():
    venue.append(index[0])
    team.append(index[1])
    wins.append(value)
temp3=pd.DataFrame({'venue':venue,'team':team,'wins':wins})
df_matches_venue=pd.merge(temp1,temp2,on=['venue','team'])
df_matches_venue=pd.merge(df_matches_venue,temp3,on=['venue','team'])
```

[77]:
```
df_matches_venue['matches']=df_matches_venue['matches0']+df_matches_venue['matches1']
df_matches_venue['win-ratio']=round(df_matches_venue['wins']/
 ↪df_matches_venue['matches'],3)
```

[78]:
```
matches_count=df_matches_venue.groupby('team',as_index=False).sum()
matches_count=matches_count.sort_values(by='matches',ascending=False)
```

```
matches_count=matches_count.iloc[:8,:]
```

[79]:
```
df_matches_venue_dict={i:{'venue':[],'ratio':[]} for i in␣
 ↪df_matches_venue['team'].unique()}
for index,row in df_matches_venue.iterrows():
    df_matches_venue_dict[row['team']]['venue'].append(row['venue'])
    df_matches_venue_dict[row['team']]['ratio'].append(row['win-ratio'])
```

[80]:
```
fig=go.Figure()
color_v=["rgb(37,102,118)", "rgb(98,240,101)", "rgb(154,72,174)",␣
 ↪"rgb(184,228,80)", "rgb(209,48,255)", "rgb(101,161,14)", "rgb(46,33,208)",␣
 ↪"rgb(241,192,57)"]
j=0
for i in matches_count['team'].unique():
#     print(i)
    fig.add_trace(go.Scatter(
        x=df_matches_venue_dict[i]['venue'],
        y=df_matches_venue_dict[i]['ratio'],
        marker=dict(color=color_v[j],size=12),
        mode='markers',
        name=i))
    j+=1
fig.update_layout(
    title_text='Win ratio of teams in ODI yearwise',
    xaxis_rangeslider_visible=True
)
fig.show()
```

### 0.1.3 Venue Statistics

**Venue wise Runs Scored**

[150]:
```
venues=df_total_agg['venue'].value_counts()
venue_scores={venue:[] for venue in venues.index[:15]}
for index,row in df_total_agg.iterrows():
#     print(venue_scores.get(row['venue'],-1),row['venue'])
    if venue_scores.get(row['venue'],-1)!=-1:
        venue_scores[row['venue']].append(row['runs-scored'])
fig=go.Figure()
for index, value in venue_scores.items():
    fig.add_trace(go.Box(y=value,name=index,boxmean='sd'))
fig.update_layout(title='Venue wise Runs scored',xaxis_title='Venue␣
 ↪Name',yaxis_title='Runs Scored')
fig.show()
```

**Venue wise Wickets Fallen**

[151]:
```
venues=df_total_agg['venue'].value_counts()
venue_scores={venue:[] for venue in venues.index[:15]}
for index,row in df_total_agg.iterrows():
#     print(venue_scores.get(row['venue'],-1),row['venue'])
    if venue_scores.get(row['venue'],-1)!=-1:
        venue_scores[row['venue']].append(row['wickets'])
fig=go.Figure()
for index, value in venue_scores.items():
    fig.add_trace(go.Box(y=value,name=index,boxmean='sd'))
fig.update_layout(title='Venue wise Wickets Fallen',xaxis_title='Venue␣
 ↪Name',yaxis_title='Wickets Fallen')
fig.show()
```

**ODI matches distribution among grounds**

[152]:
```
venues=df_total_agg['venue'].value_counts().iloc[:50]
fig=go.Figure()
fig.add_trace(go.Pie(labels=venues.index,values=venues.values))
fig.update_layout(title='ODI matches distribution among top 50 grounds'
                  ,margin=dict(l=20,r=20,t=40,b=20),
                  autosize=False,
                  width=1000,
                  height=1000)
fig.show()
```

### 0.1.4 Player Statistics

**Matches distibution between genders**

[153]:
```
gender=df_info['gender'].value_counts()
fig=go.Figure()
fig.add_trace(go.Pie(labels=gender.index.unique(),values=gender.values))
fig.update_layout(title='Gender wise matches distribution')
fig.show()
```

**Top 10 Batsmen**

[109]:
```
df_scorecard_batsman_agg=df_scorecard.groupby(['name'],as_index=False).sum()
df_scorecard_batsman_agg=df_scorecard_batsman_agg.
 ↪sort_values(by=['runs-scored'],ascending=False)
df_scorecard_batsman_agg=df_scorecard_batsman_agg.reset_index(drop=True)
```

[110]:
```
df_scorecard_batsman_agg=df_scorecard_batsman_agg.drop(['batting-position',
       'over-batsman','extras',
       'fall-of-wicket-score', 'fall-of-wicket-overs',␣
 ↪'fall-of-wicket-no'],axis=1)
```

[111]:
```
batsman_innings=df_scorecard[df_scorecard['over-batsman']>0.0]['name']
batsman_innings=batsman_innings.value_counts().to_dict()
df_scorecard_batsman_agg['innings']=df_scorecard_batsman_agg['name'].
 ↪map(batsman_innings)
df_scorecard_batsman_agg['strike-rate']=df_scorecard_batsman_agg.apply(lambda␣
 ↪row: round((row['runs-scored']/row['balls-played'])*100,3) if␣
 ↪row['balls-played']>0 else 0 ,axis=1)
df_scorecard_batsman_agg['avg']=df_scorecard_batsman_agg.apply(lambda row:␣
 ↪round(row['runs-scored']/row['innings'],3) if row['innings']>0 else 0,axis=1)
```

[112]:
```
df_scorecard_batsman_agg_sub=df_scorecard_batsman_agg.iloc[:10]
fig=go.Figure()
fig.add_trace(go.Table(
    header=dict(
        values=['Batsman Name','Innings','Runs Scored','Balls␣
 ↪Played','Fours','Sixes','Batting Strike Rate','Batting Average'],
        fill_color='paleturquoise',
        align='left',
        font=dict(color='black',size=14)
    ),
    cells=dict(values=
               ␣
 ↪[df_scorecard_batsman_agg_sub['name'],df_scorecard_batsman_agg_sub['innings'],df_scorecard_
                ␣
 ↪df_scorecard_batsman_agg_sub['balls-played'],df_scorecard_batsman_agg_sub['fours'],df_score
               ␣
 ↪df_scorecard_batsman_agg_sub['strike-rate'],df_scorecard_batsman_agg_sub['avg']],
                align='left'
    )
))
fig.update_layout(title='Top 10 Batsmen')
fig.show()
```

**Batsmen Performance**

[113]:
```
df_scorecard_batsman_agg_sub=df_scorecard_batsman_agg.iloc[:]
hover_text=[]
bubble_size=[]
for index,row in df_scorecard_batsman_agg_sub.iterrows():
    hover_text.append(
        ('Name: {name}<br>'+'Balls Played: {balls}<br>'+'Average:␣
 ↪{avg}<br>'+'Strike Rate: {strike}<br>').format(
            ␣
 ↪name=row['name'],avg=row['avg'],strike=row['strike-rate'],balls=row['balls-played']))
    bubble_size.append(math.sqrt(row['strike-rate'])*2)
fig=go.Figure()
fig.add_trace(
```

```
    go.Scatter(
    x=df_scorecard_batsman_agg_sub['balls-played'],
    y=df_scorecard_batsman_agg_sub['runs-scored'],
    text=hover_text,
        mode='markers',
        marker=dict(
        color=df_scorecard_batsman_agg_sub['innings'],
        colorbar=dict(
        title='Innings Played'
        ),
            colorscale='Viridis',
        size=bubble_size,
            showscale=True
        )
    )
)
fig.update_layout(title='Batsmen Performance',xaxis_title='Balls␣
 ↪Played',yaxis_title='Runs scored')
fig.show()
```

**Top 10 Bowlers**

[158]:
```
df_scorecard_bowler_agg=df_scorecard.groupby(['name'],as_index=False).sum()
df_scorecard_bowler_agg=df_scorecard_bowler_agg.
 ↪sort_values(by=['wickets'],ascending=False)
df_scorecard_bowler_agg=df_scorecard_bowler_agg.reset_index(drop=True)
```

[159]:
```
df_scorecard_bowler_agg=df_scorecard_bowler_agg.drop(['batting-position',␣
 ↪'over-batsman', 'runs-scored',
       'balls-played', 'dots', 'ones', 'twos', 'threes', 'fours', 'sixes',␣
 ↪'extras',
       'fall-of-wicket-score', 'fall-of-wicket-overs',␣
 ↪'fall-of-wicket-no'],axis=1)
```

[160]:
```
bowler_innings=df_scorecard[df_scorecard['balls-bowled']>0]['name']
bowler_innings=bowler_innings.value_counts().to_dict()
df_scorecard_bowler_agg['innings']=df_scorecard_bowler_agg['name'].
 ↪map(bowler_innings)
df_scorecard_bowler_agg['strike-rate']=df_scorecard_bowler_agg.apply(lambda row:
 ↪ round((row['balls-bowled']/row['wickets']),3) if row['wickets']>0 else 0␣
 ↪,axis=1)
df_scorecard_bowler_agg['avg']=df_scorecard_bowler_agg.apply(lambda row:␣
 ↪round(row['runs-given']/row['wickets'],3) if row['wickets']>0 else 0,axis=1)
df_scorecard_bowler_agg['eco']=df_scorecard_bowler_agg.apply(lambda row:␣
 ↪round(row['runs-given']/(row['balls-bowled']/6),3) if row['balls-bowled']>0␣
 ↪else 0,axis=1)
```

[161]:
```
df_scorecard_bowler_agg_sub=df_scorecard_bowler_agg.iloc[:10]
fig=go.Figure()
fig.add_trace(go.Table(
    header=dict(
        values=['Bowler Name','Innings','Balls','Maiden Overs','Runs␣
 ↪Conceded','Wickets','Economy','Bowling Strike Rate','Bowling Average'],
        fill_color='paleturquoise',
        align='left',
        font=dict(color='black',size=14)
    ),
    cells=dict(values=
               ␣
 ↪[df_scorecard_bowler_agg_sub['name'],df_scorecard_bowler_agg_sub['innings'],df_scorecard_bo
                ␣
 ↪df_scorecard_bowler_agg_sub['maiden-overs'],df_scorecard_bowler_agg_sub['runs-given'],df_sc
                 df_scorecard_bowler_agg_sub['eco'],
               ␣
 ↪df_scorecard_bowler_agg_sub['strike-rate'],df_scorecard_bowler_agg_sub['avg']],
                align='left'
    )
))
fig.update_layout(title='Top 10 Bowlers')
fig.show()
```

**Bowlers Performance**

[162]:
```
df_scorecard_bowler_agg_sub=df_scorecard_bowler_agg.iloc[:]
hover_text=[]
bubble_size=[]

for index,row in df_scorecard_bowler_agg_sub.iterrows():
    hover_text.append(
        ('Name: {name}<br>'+'Economy: {eco}<br>'+'Average: {avg}<br>'+'Strike␣
 ↪Rate: {strike}<br>').format(
            ␣
 ↪name=row['name'],avg=row['avg'],strike=row['strike-rate'],eco=row['eco']))
    bubble_size.append(math.sqrt(row['eco'])*6)
fig=go.Figure()
fig.add_trace(
    go.Scatter(
    x=df_scorecard_batsman_agg_sub['balls-bowled'],
    y=df_scorecard_batsman_agg_sub['wickets'],
    text=hover_text,
        mode='markers',
        marker=dict(
        color=df_scorecard_batsman_agg_sub['innings'],
        colorbar=dict(
```

```
        title='Innings Played'
        ),
            colorscale='Viridis',
        size=bubble_size,
            showscale=True
        )
    )
)
fig.update_layout(title='Bowlers Performance',xaxis_title='Balls␣
 ↪Bowled',yaxis_title='Wickets')
fig.show()
```

**Batsmen performance over the years: Runs scored**

[183]:
```
df_player_year=df_total.groupby(by=['year','name'],as_index=False).sum()
```

[164]:
```
top_batsmen=df_scorecard_batsman_agg.iloc[:10]['name'].tolist()
df_batsmen_year=df_player_year[df_player_year['name'].isin(top_batsmen)]
df_batsman_year_grouped=df_batsmen_year.groupby('name')
```

[151]:
```
fig=go.Figure()
for name,group in df_batsman_year_grouped:
    fig.add_trace(go.Scatter(x=group['year'],y=group['runs-scored'],name=name))
fig.update_layout(
    title_text='Runs in a calendar year of top 10 ODI batsmen',
    xaxis_rangeslider_visible=True,
    xaxis_title='Year',
    yaxis_title='Runs'
)
fig.show()
```

**Batsmen performance over the years: Strike rate**

[152]:
```
df_batsmen_year['strike-rate']=round(df_batsmen_year['runs-scored']/
 ↪df_batsmen_year['balls-played'],4)*100
df_batsman_year_grouped=df_batsmen_year.groupby('name')
```

[153]:
```
fig=go.Figure()
for name,group in df_batsman_year_grouped:
    fig.add_trace(go.Scatter(x=group['year'],y=group['strike-rate'],name=name))
fig.update_layout(
    title_text='Strike rate of top 10 batsmen in ODI yearwise',
    xaxis_rangeslider_visible=True,
    xaxis_title='Year',
    yaxis_title='Strike Rate'
)
fig.show()
```

**Bowlers performance over the years: Wickets taken**

[171]:
```
top_bowlers=df_scorecard_bowler_agg.iloc[:10]['name'].tolist()
df_bowlers_year=df_player_year[df_player_year['name'].isin(top_bowlers)]
df_bowlers_year_grouped=df_bowlers_year.groupby('name')
```

[172]:
```
fig=go.Figure()
for name,group in df_bowlers_year_grouped:
    fig.add_trace(go.Scatter(x=group['year'],y=group['wickets'],name=name))
fig.update_layout(
    title_text='Runs in a calendar year of top 10 ODI bowlers',
    xaxis_rangeslider_visible=True,
    xaxis_title='Year',
    yaxis_title='Wickets'
)
fig.show()
```

**Bowlers performance over the years: Economy**

[181]:
```
df_bowlers_year['economy']=round(df_bowlers_year['runs-given']/
 ↪(df_bowlers_year['balls-bowled']/6),2)
df_bowlers_year_grouped=df_bowlers_year.groupby('name')
```

[182]:
```
fig=go.Figure()
for name,group in df_bowlers_year_grouped:
    fig.add_trace(go.Scatter(x=group['year'],y=group['economy'],name=name))
fig.update_layout(
    title_text='Economy of top 10 bowlers in ODI yearwise',
    xaxis_rangeslider_visible=True,
    xaxis_title='Year',
    yaxis_title='Strike Rate'
)
fig.show()
```