NFL Working

December 5, 2019

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.patches as patches
sns.set_style('whitegrid')
```

[3]:
```
playlist = pd.read_csv('PlayList.csv')
inj = pd.read_csv('InjuryRecord.csv')
trk = pd.read_csv('PlayerTrackData.csv')
```

[4]:
```
playlist.head()
```

[4]:
```
   PlayerKey   GameID    PlayKey RosterPosition  PlayerDay  PlayerGame  \
0      26624  26624-1  26624-1-1    Quarterback          1           1
1      26624  26624-1  26624-1-2    Quarterback          1           1
2      26624  26624-1  26624-1-3    Quarterback          1           1
3      26624  26624-1  26624-1-4    Quarterback          1           1
4      26624  26624-1  26624-1-5    Quarterback          1           1

  StadiumType  FieldType  Temperature         Weather PlayType  \
0     Outdoor  Synthetic           63  Clear and warm     Pass
1     Outdoor  Synthetic           63  Clear and warm     Pass
2     Outdoor  Synthetic           63  Clear and warm     Rush
3     Outdoor  Synthetic           63  Clear and warm     Rush
4     Outdoor  Synthetic           63  Clear and warm     Pass

   PlayerGamePlay Position PositionGroup
0               1       QB            QB
1               2       QB            QB
2               3       QB            QB
3               4       QB            QB
4               5       QB            QB
```

[70]:
```
trk.head()
```

```
[70]:      PlayKey  time                event      x      y     dir   dis       o  \
      0  26624-1-1   0.0  huddle_start_offense  87.46  28.93  288.24  0.01  262.33
      1  26624-1-1   0.1                   NaN  87.45  28.92  283.91  0.01  261.69
      2  26624-1-1   0.2                   NaN  87.44  28.92  280.40  0.01  261.17
      3  26624-1-1   0.3                   NaN  87.44  28.92  278.79  0.01  260.66
      4  26624-1-1   0.4                   NaN  87.44  28.92  275.44  0.01  260.27

            s
      0  0.13
      1  0.12
      2  0.12
      3  0.10
      4  0.09
```

```
[154]: inj.head()
```

```
[154]:    PlayerKey   GameID      PlayKey BodyPart    Surface  DM_M1  DM_M7  DM_M28  \
       0      39873  39873-4  39873-4-32     Knee  Synthetic      1      1       1
       1      46074  46074-7  46074-7-26     Knee    Natural      1      1       0
       2      36557  36557-1  36557-1-70    Ankle  Synthetic      1      1       1
       3      46646  46646-3  46646-3-30    Ankle    Natural      1      0       0
       4      43532  43532-5  43532-5-69    Ankle  Synthetic      1      1       1

          DM_M42
       0       1
       1       0
       2       1
       3       0
       4       1
```

```
[155]: inj['PlayKey'].isna().sum()
```

```
[155]: 28
```

```
[156]: inj.groupby('BodyPart').count()
```

```
[156]:           PlayerKey  GameID  PlayKey  Surface  DM_M1  DM_M7  DM_M28  DM_M42
       BodyPart
       Ankle            42      42       35       42     42     42      42      42
       Foot              7       7        6        7      7      7       7       7
       Heel              1       1        0        1      1      1       1       1
       Knee             48      48       36       48     48     48      48      48
       Toes              7       7        0        7      7      7       7       7
```

1 Data Cleaning

1.0.1 Playlist

[48]:

```
playlist['StadiumType'].nunique()
```

[48]:

```
29
```

[49]:

```
outdoor = ['Outdoor', 'Outdoors', 'Cloudy', 'Heinz Field',
              'Outdor', 'Ourdoor', 'Outside', 'Outddors',
              'Outdoor Retr Roof-Open', 'Oudoor', 'Bowl']

indoor_closed = ['Indoors', 'Indoor', 'Indoor, Roof Closed', 'Indoor, Roof␣
 ↪Closed',
                   'Retractable Roof', 'Retr. Roof-Closed', 'Retr. Roof -␣
 ↪Closed', 'Retr. Roof Closed']

indoor_open = ['Indoor, Open Roof', 'Open', 'Retr. Roof-Open', 'Retr. Roof -␣
 ↪Open']

dome_closed = ['Dome', 'Domed, closed', 'Closed Dome', 'Domed', 'Dome, closed']

dome_open = ['Domed, Open', 'Domed, open']
```

[50]:

```
playlist.loc[playlist['StadiumType'].isin(outdoor),'StadiumType'] = 'Outdoor'

playlist.loc[playlist['StadiumType'].isin(indoor_closed),'StadiumType'] =␣
 ↪'indoor_closed'

playlist.loc[playlist['StadiumType'].isin(indoor_open),'StadiumType'] =␣
 ↪'indoor_open'

playlist.loc[playlist['StadiumType'].isin(dome_closed),'StadiumType'] =␣
 ↪'dome_closed'

playlist.loc[playlist['StadiumType'].isin(dome_open),'StadiumType'] =␣
 ↪'dome_open'
```

[53]:

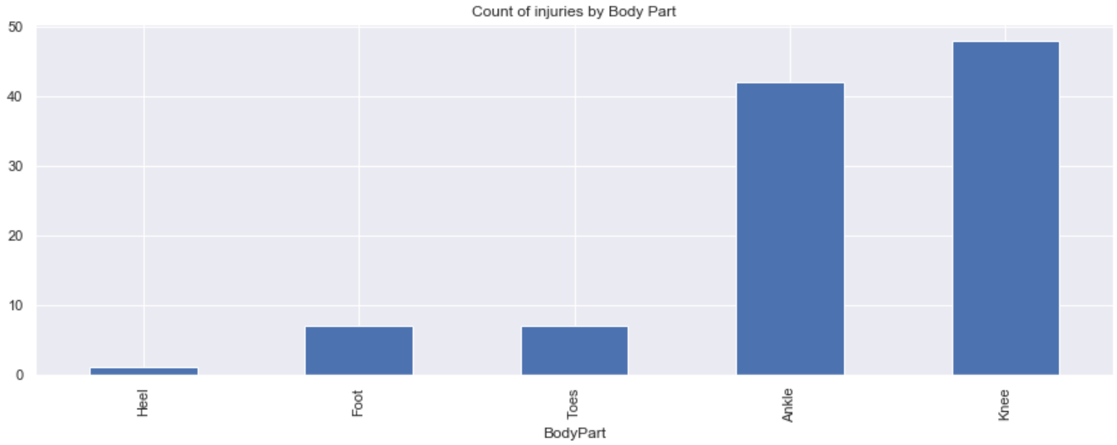
```
playlist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 267005 entries, 0 to 267004
Data columns (total 14 columns):
PlayerKey         267005 non-null int64
GameID            267005 non-null object
PlayKey           267005 non-null object
```

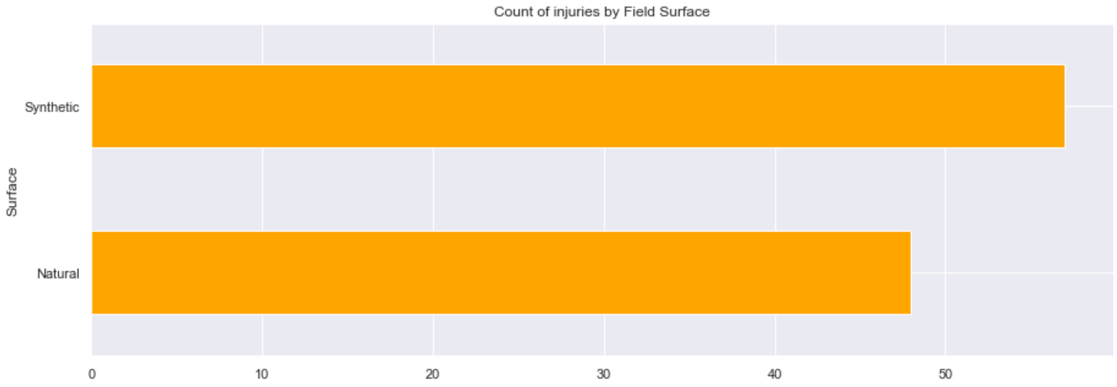
```
RosterPosition    267005 non-null object
PlayerDay         267005 non-null int64
PlayerGame        267005 non-null int64
StadiumType       250095 non-null object
FieldType         267005 non-null object
Temperature       267005 non-null int64
Weather           248314 non-null object
PlayType          266638 non-null object
PlayerGamePlay    267005 non-null int64
Position          267005 non-null object
PositionGroup     267005 non-null object
dtypes: int64(5), object(9)
memory usage: 28.5+ MB
```

2 EDA

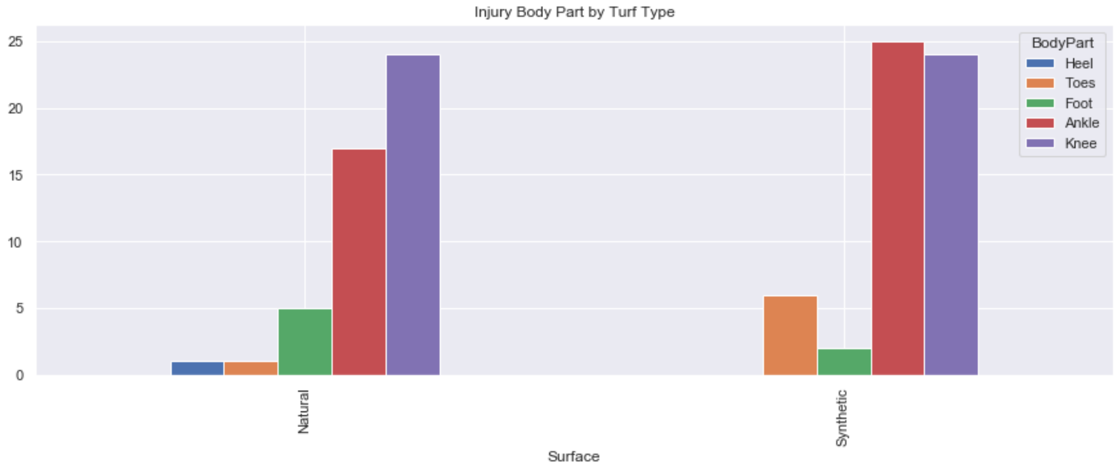
[157]:
```
inj.groupby('BodyPart').count()['PlayerKey'] \
    .sort_values() \
    .plot(kind='bar', figsize=(15, 5), title='Count of injuries by Body Part')
plt.show()
```



[158]:
```
inj.groupby('Surface').count()['PlayerKey'] \
    .sort_values() \
    .plot(kind='barh', figsize=(15, 5), title='Count of injuries by Field␣
↪Surface', color='orange')
plt.show()
```



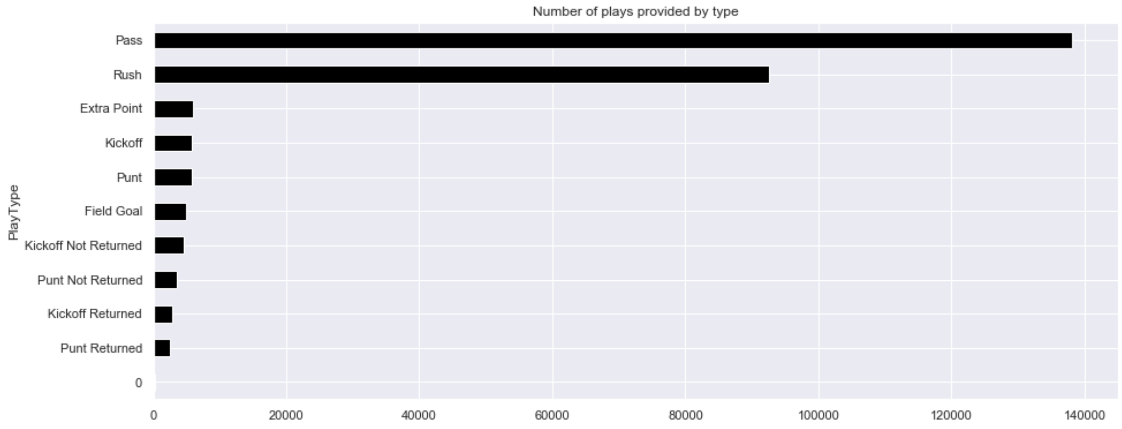
[159]:
```
inj.groupby(['BodyPart','Surface']) \
    .count() \
    .unstack('BodyPart')['PlayerKey'] \
    .T.sort_values('Natural').T \
    .sort_values('Ankle') \
    .plot(kind='bar', figsize=(15, 5), title='Injury Body Part by Turf Type')
plt.show()
```



[160]:
```
playlist['PlayKey'].nunique()
```

[160]:
```
267005
```

[161]:
```
playlist[['PlayKey','PlayType']].drop_duplicates() \
    .groupby('PlayType').count()['PlayKey'] \
    .sort_values() \
    .plot(kind='barh',
        figsize=(15, 6),
          color='black',
         title='Number of plays provided by type')
plt.show()
```

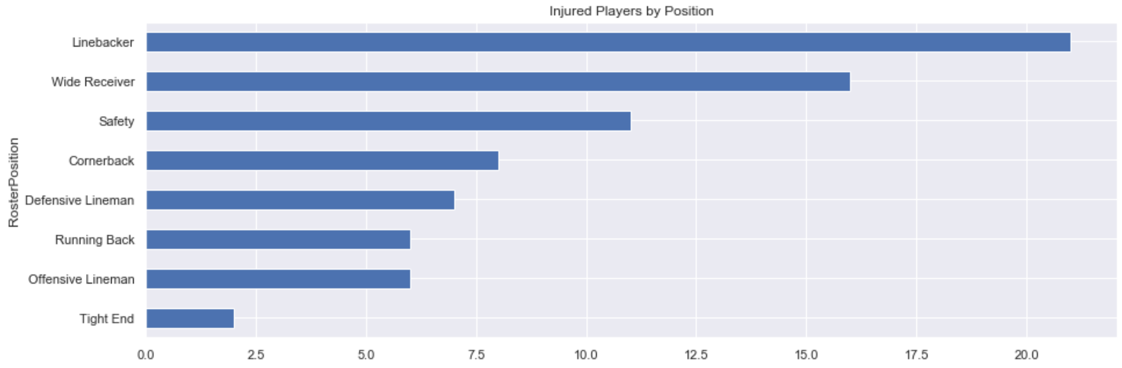


[162]:

```
inj_detailed = inj.merge(playlist)
```

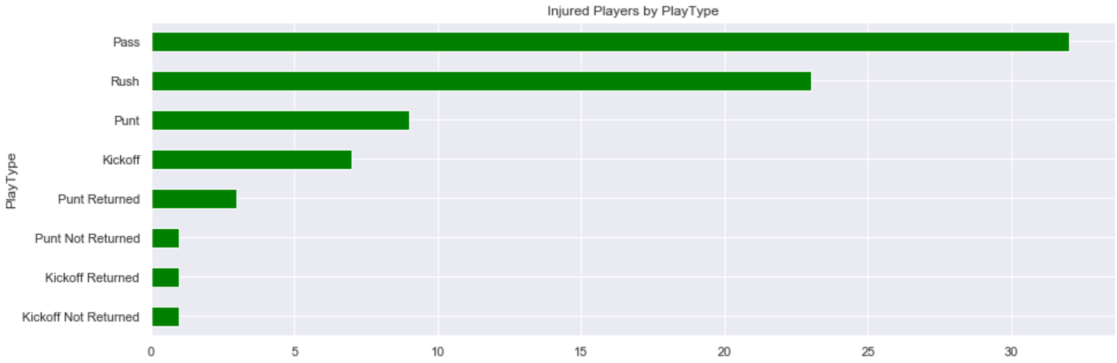
[163]:

```
inj_detailed.groupby('RosterPosition').count()['PlayerKey'] \
    .sort_values() \
    .plot(figsize=(15, 5), kind='barh', title='Injured Players by Position')
plt.show()
```



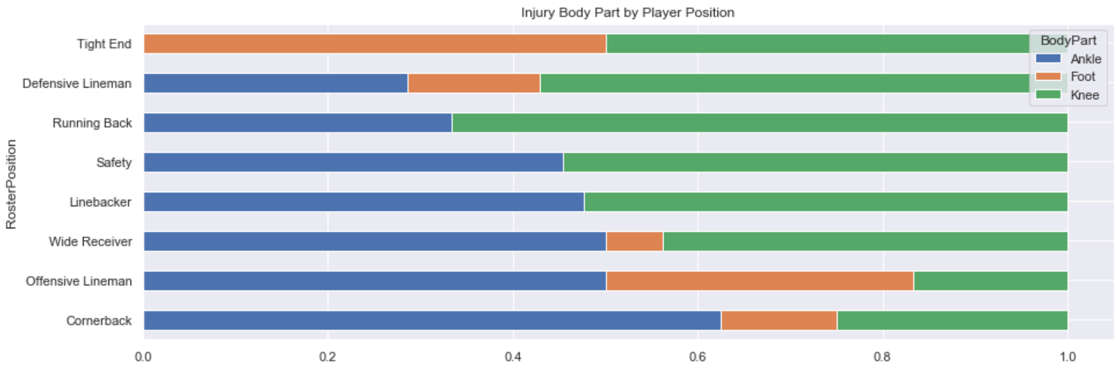
[164]:

```
inj_detailed.groupby('PlayType').count()['PlayerKey'] \
    .sort_values() \
    .plot(figsize=(15, 5), kind='barh', title='Injured Players by PlayType',␣
 ↪color='green')
plt.show()
```



[165]:

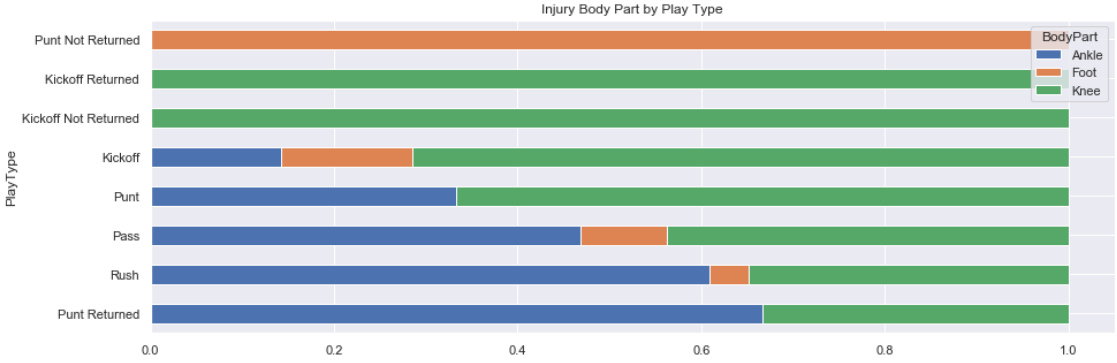
```
inj_detailed.groupby(['RosterPosition','BodyPart']) \
    .count() \
    .unstack('BodyPart')['PlayerKey'] \
    .T.apply(lambda x: x / x.sum()) \
    .sort_values('BodyPart').T.sort_values('Ankle', ascending=False) \
    .plot(kind='barh',
          figsize=(15, 5),
          title='Injury Body Part by Player Position',
          stacked=True)
plt.show()
```



[166]:

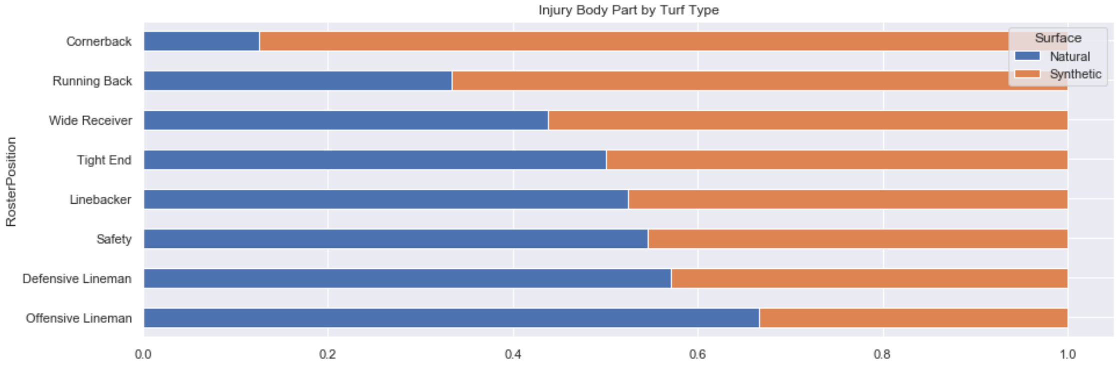
```
inj_detailed.groupby(['PlayType','BodyPart']) \
    .count() \
    .unstack('BodyPart')['PlayerKey'] \
    .T.apply(lambda x: x / x.sum()) \
    .sort_values('BodyPart').T.sort_values('Ankle', ascending=False) \
    .plot(kind='barh',
          figsize=(15, 5),
          title='Injury Body Part by Play Type',
          stacked=True)
```

```
plt.show()
```



[167]:

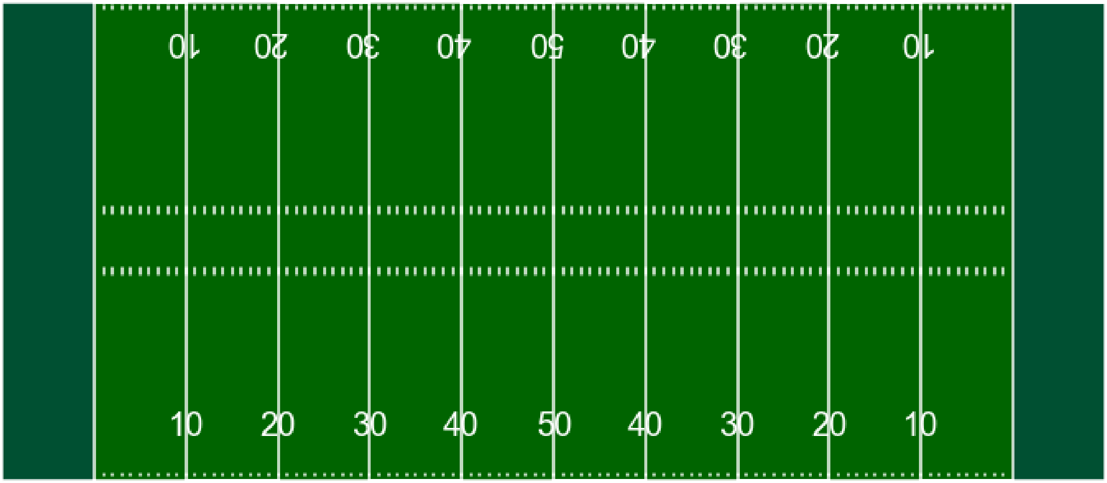
```
inj_detailed.groupby(['RosterPosition','Surface']) \
    .count() \
    .unstack('Surface')['PlayerKey'] \
    .T.apply(lambda x: x / x.sum()) \
    .sort_values('Surface').T.sort_values('Natural', ascending=False) \
    .plot(kind='barh',
          figsize=(15, 5),
          title='Injury Body Part by Turf Type',
          stacked=True)
plt.show()
```



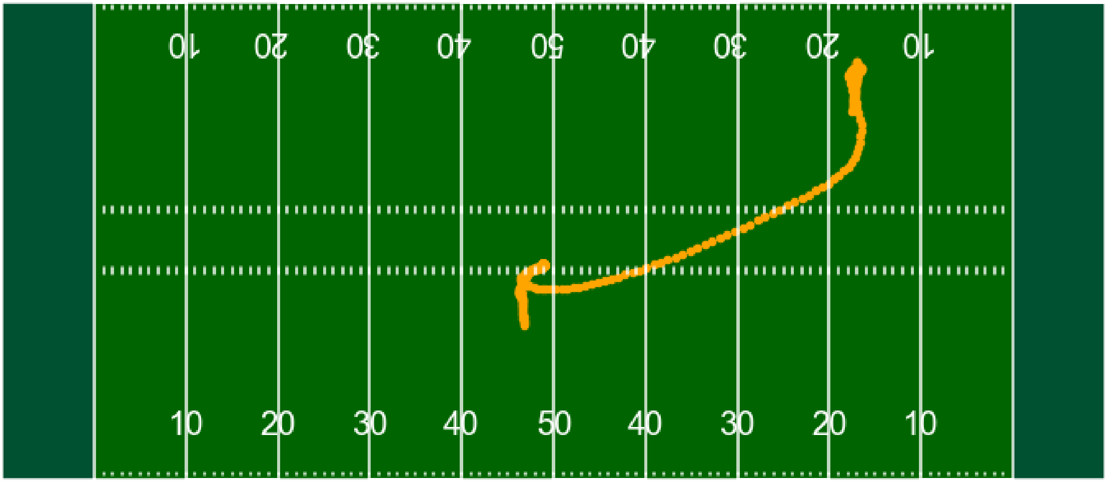
2.0.1 Function

[168]:

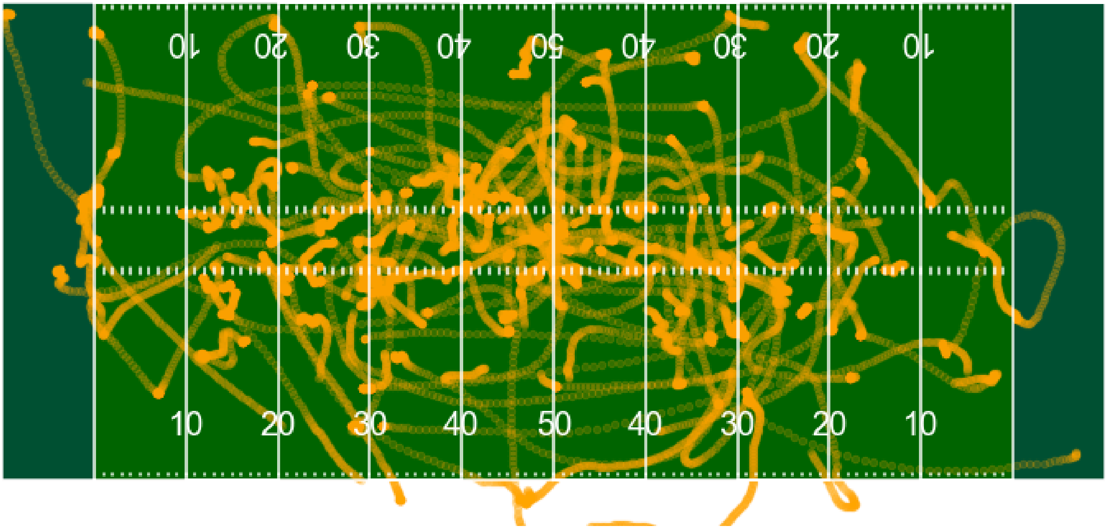
```
def create_football_field(linenumbers=True,
                          endzones=True,
                          highlight_line=False,
                          highlight_line_number=50,
                          highlighted_name='Line of Scrimmage',
                          fifty_is_los=False,
                          figsize=(12, 6.33)):
    """
    Function that plots the football field for viewing plays.
    Allows for showing or hiding endzones.
    """
    rect = patches.Rectangle((0, 0), 120, 53.3, linewidth=0.1,
                             edgecolor='r', facecolor='darkgreen', zorder=0)

    fig, ax = plt.subplots(1, figsize=figsize)
    ax.add_patch(rect)

    plt.plot([10, 10, 10, 20, 20, 30, 30, 40, 40, 50, 50, 60, 60, 70, 70, 80,
              80, 90, 90, 100, 100, 110, 110, 120, 0, 0, 120, 120],
             [0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3,
              53.3, 0, 0, 53.3, 53.3, 0, 0, 53.3, 53.3, 53.3, 0, 0, 53.3],
             color='white')
    if fifty_is_los:
        plt.plot([60, 60], [0, 53.3], color='gold')
        plt.text(62, 50, '<- Player Yardline at Snap', color='gold')
    # Endzones
    if endzones:
        ez1 = patches.Rectangle((0, 0), 10, 53.3,
                                linewidth=0.1,
                                edgecolor='r',
                                facecolor='blue',
                                alpha=0.2,
                                zorder=0)
        ez2 = patches.Rectangle((110, 0), 120, 53.3,
                                linewidth=0.1,
                                edgecolor='r',
                                facecolor='blue',
                                alpha=0.2,
                                zorder=0)
        ax.add_patch(ez1)
        ax.add_patch(ez2)
    plt.xlim(0, 120)
    plt.ylim(-5, 58.3)
    plt.axis('off')
    if linenumbers:
        for x in range(20, 110, 10):
            numb = x
            if x > 50:
                numb = 120 - x
            plt.text(x, 5, str(numb - 10),
                     horizontalalignment='center',
                     fontsize=20,  # fontname='Arial',
                     color='white')
            plt.text(x - 0.95, 53.3 - 5, str(numb - 10),
                     horizontalalignment='center',
                     fontsize=20,  # fontname='Arial',
                     color='white', rotation=180)
    if endzones:
        hash_range = range(11, 110)
    else:
        hash_range = range(1, 120)

    for x in hash_range:
        ax.plot([x, x], [0.4, 0.7], color='white')
        ax.plot([x, x], [53.0, 52.5], color='white')
        ax.plot([x, x], [22.91, 23.57], color='white')
        ax.plot([x, x], [29.73, 30.39], color='white')

    if highlight_line:
        hl = highlight_line_number + 10
        plt.plot([hl, hl], [0, 53.3], color='yellow')
        plt.text(hl + 2, 50, '<- {}'.format(highlighted_name),
                 color='yellow')
    return fig, ax

create_football_field()
plt.show()
```



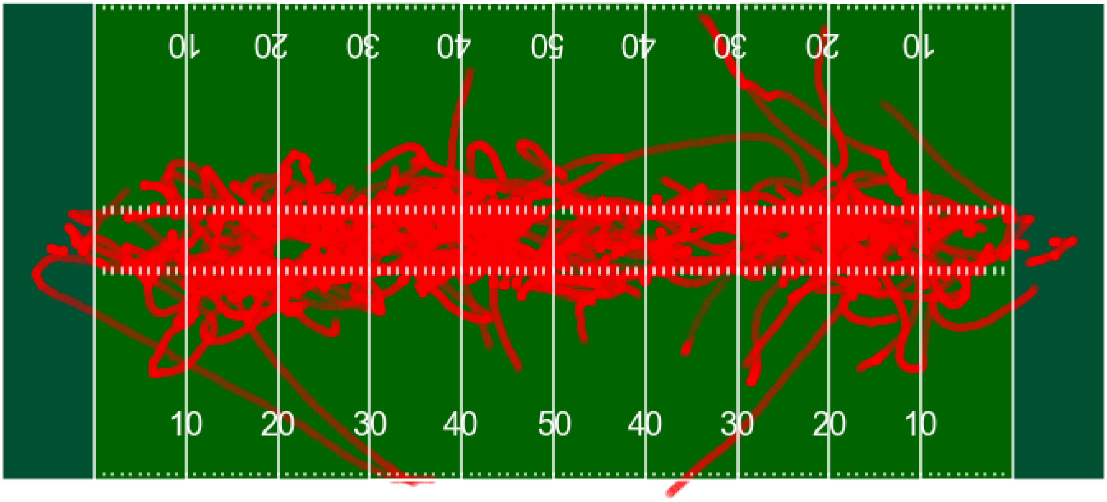
[169]:
```
example_play_id = inj['PlayKey'].values[0]
fig, ax = create_football_field()
trk.query('PlayKey == @example_play_id').plot(kind='scatter', x='x', y='y',␣
 ↪ax=ax, color='orange')
plt.show()
```



[170]:
```
inj_play_list = inj['PlayKey'].tolist()
fig, ax = create_football_field()
for playkey, inj_play in trk.query('PlayKey in @inj_play_list').
 ↪groupby('PlayKey'):
    inj_play.plot(kind='scatter', x='x', y='y', ax=ax, color='orange', alpha=0.
 ↪2)
plt.show()
```

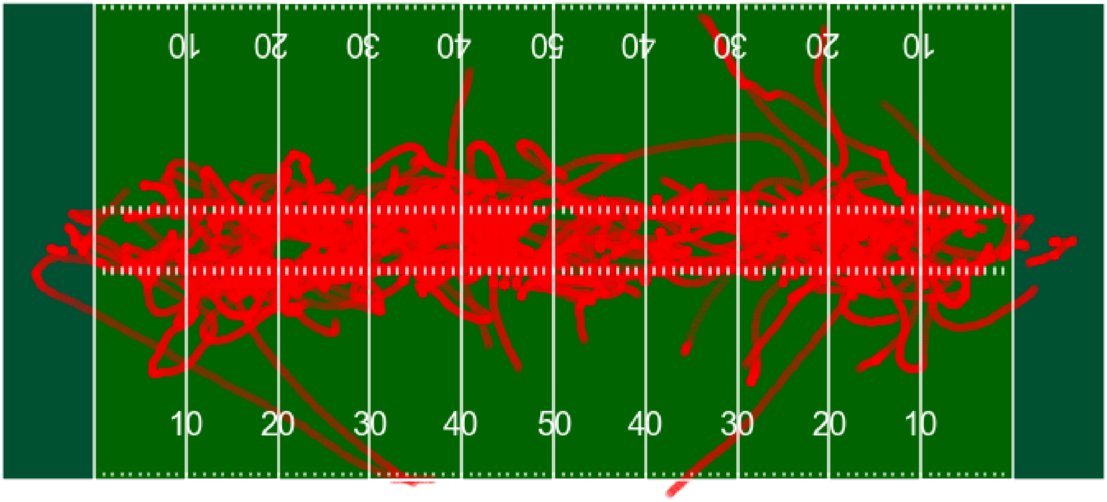


```
[171]: fig, ax = create_football_field()
       for playkey, inj_play in trk.query('PlayKey not in @inj_play_list').head(50000).
        ↪groupby('PlayKey'):
           inj_play.plot(kind='scatter', x='x', y='y', ax=ax, color='red', alpha=0.2)
       plt.show()
```

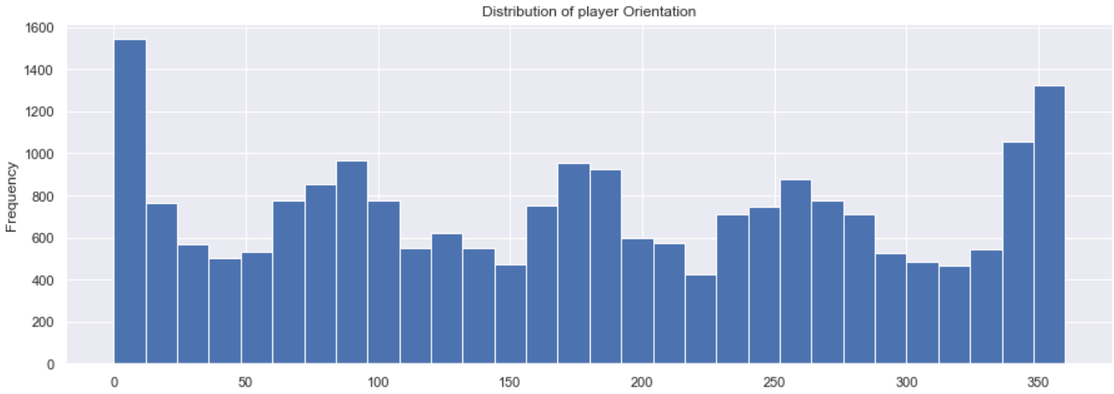


```
[172]: fig, ax = create_football_field()
```

```
for playkey, inj_play in trk.query('PlayKey not in @inj_play_list').head(50000).
 ↪groupby('PlayKey'):
    inj_play.plot(kind='scatter', x='x', y='y', ax=ax, color='red', alpha=0.2)
plt.show()
```



```
[173]: trk.query('PlayKey in @inj_play_list')['o'].plot(kind='hist',
                                                         title='Distribution of player␣
        ↪Orientation',
                                                         figsize=(15, 5), bins=30)
       plt.show()
```



3 Plots distance run of injury and uninjury groups

[71]:

```
t1 = playlist[['PlayKey','RosterPosition','GameID']]
```

[72]:

```
t2 = t1.merge(trk[['PlayKey','dis']],on = 'PlayKey')
```

[134]:

```
t2['injury'] = 0
```

[136]:

```
t2.loc[t2['PlayKey'].isin(inj['PlayKey']),'injury'] = 100
```

[138]:

```
t2.loc[t2['GameID'].isin(inj['GameID']) & ~t2['PlayKey'].isin(inj['PlayKey'])␣
 ↪,'injury'] = 200
```

[139]:

```
t2['injury'].value_counts()
```

[139]:

```
0      75400885
200      943958
100       21905
Name: injury, dtype: int64
```

[144]:

```
injury_group = t2[t2['injury'] == 100]
```

[146]:

```
injury_group.shape
```

[146]:

```
(21905, 5)
```

[145]:

```
nonInj_group =  t2[t2['injury'] == 0]
```

[148]:

```
nonInj_group.shape
```

[148]:

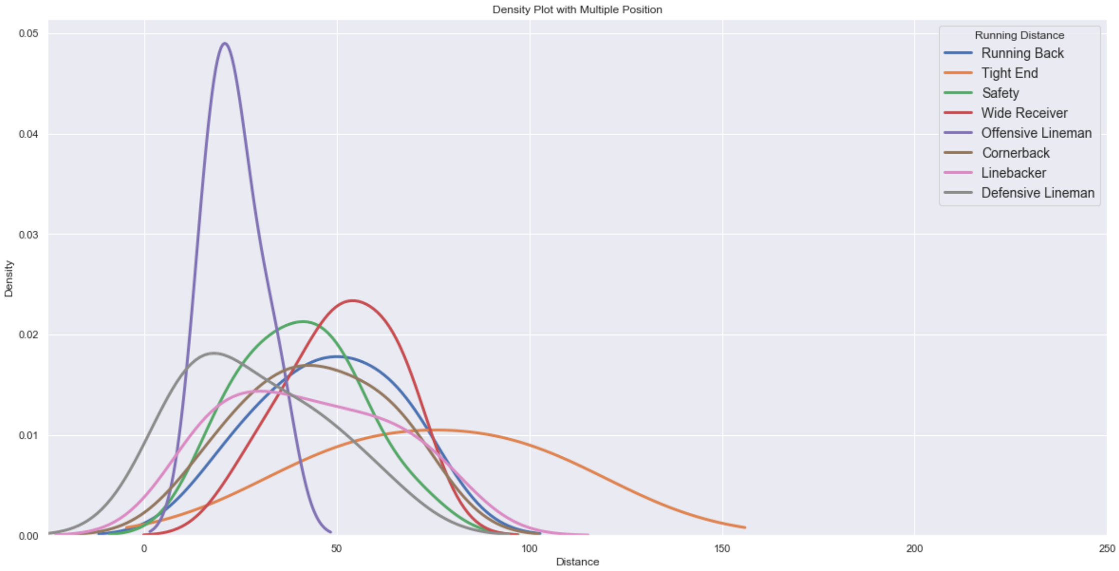
```
(75400885, 5)
```

[149]:

```
res1 = injury_group.groupby(['PlayKey','RosterPosition']).sum().reset_index()
res2 = nonInj_group.groupby(['PlayKey','RosterPosition']).sum().reset_index()
```

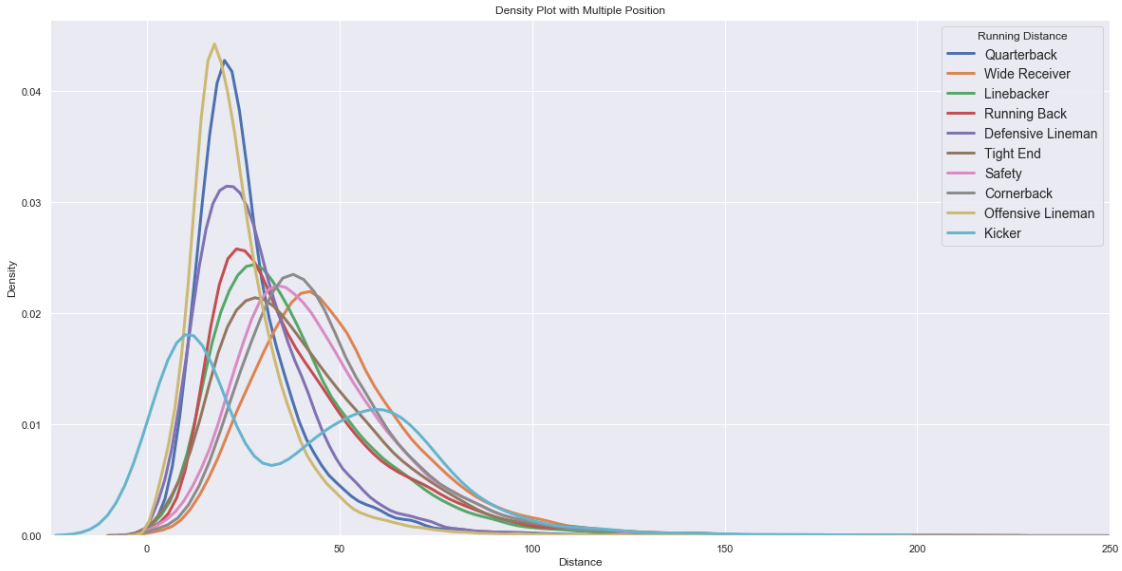
[150]:

```
# List of five airlines to plot

pos = res1['RosterPosition'].unique().tolist()

sns.set(rc={'figure.figsize':(20,10)})


# Iterate through the five airlines
for each in pos:
    # Subset to the airline
    subset = res1[res1['RosterPosition'] == each]
    # Draw the density plot
    sns.distplot(subset['dis'], hist = False, kde = True,
                 kde_kws = {'linewidth': 3},
                 label = each)

# Plot formatting
plt.legend(prop={'size': 14}, title = 'Running Distance')
plt.title('Density Plot with Multiple Position')
plt.xlabel('Distance')
plt.ylabel('Density')
plt.xlim(-25,250)
#plt.figure(figsize=(30,20))
#plt.figure(num=None, figsize=(30, 15), dpi=80, facecolor='w', edgecolor='k')
plt.savefig('injury.png')
```



[151]:

```
# List of five airlines to plot

pos = res2['RosterPosition'].unique().tolist()

sns.set(rc={'figure.figsize':(20,10)})


# Iterate through the five airlines
for each in pos:
    # Subset to the airline
    subset = res2[res2['RosterPosition'] == each]

    # Draw the density plot
    sns.distplot(subset['dis'], hist = False, kde = True,
                 kde_kws = {'linewidth': 3},
                 label = each)

# Plot formatting
plt.legend(prop={'size': 14}, title = 'Running Distance')
plt.title('Density Plot with Multiple Position')
plt.xlabel('Distance')
plt.ylabel('Density')
plt.xlim(-25,250)
#plt.figure(figsize=(30,20))
#plt.figure(num=None, figsize=(30, 15), dpi=80, facecolor='w', edgecolor='k')
plt.savefig('nonInj.png')
```



```
[55]: res1.to_csv('res1.csv',index = False)
```

```
[56]: res2.to_csv('res2.csv',index = False)
```