SPORT ANALYSIS WITH PYTHON

Warsaw
October 19-20, 2017

Speaker:

- ✓ **ThuyLe**
- ✓ **Ericsson, Italy**

Football analysis

1. Match (score, time, …)
2. Players (performance, goal, award, match, time…)
3. Fan (loyalty, number, scandal…)

   ...

Match analysis (*real-time match tracking*)

OUTLINE

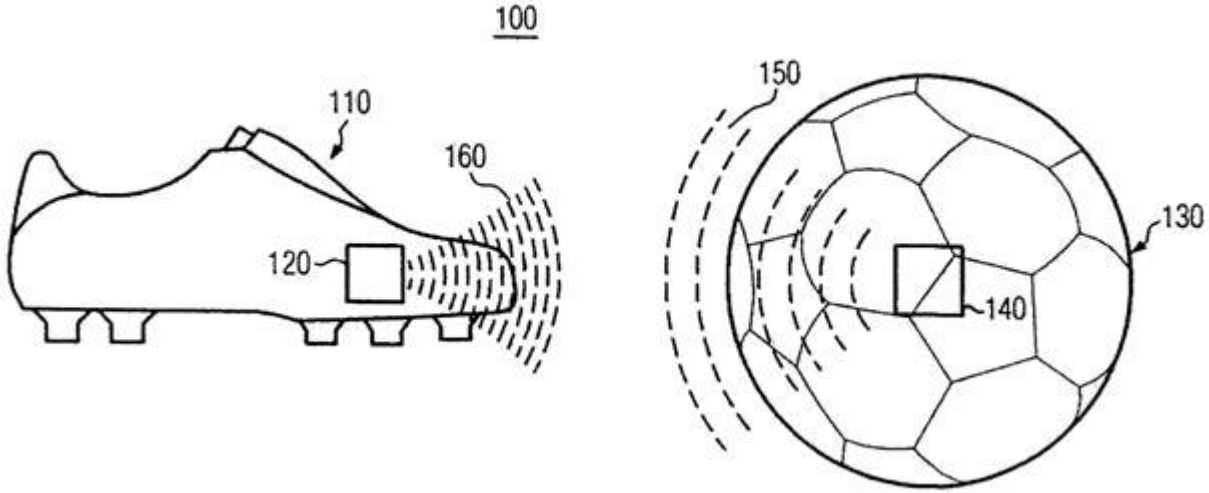
1. Data gathering
2. Data analysis with Python
3. Data visualization with Tableau

Trackers

Sensors

Data gathering

Data gathering



Data analysis

OUTLINE

1. Data gathering
2. Data analysis with Python
3. Data visualization with Tableau

Load Data

ThuyLe_Part1_Demo.csv

```
#==========================================
# # IMPORT LIBRARIES
#==========================================
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import random as rd
import os, time, math, datetime
from math import sqrt


#==========================================
# # Load Data
#==========================================
def loadCSV(csvName, _sep):
    data = pd.read_csv(csvName, sep=_sep)
    return data


print os.getcwd()
```

```
/home/mapr/notebookHome/THUYLE_TEST
```

Load Data

```
path = "/home/mapr/notebookHome/THUYLE_TEST"

os.chdir(path)
sep = "|"
df = loadCSV("ThuyLe_Part1_Demo.csv", sep)
```

```
df.head(15)
```

	timestamp	device_id	x_pos	y_pos
0	1.237747e+12	9	50.875526	25.792036
1	1.237747e+12	8	28.525500	41.909308
2	1.237747e+12	2	21.108100	12.481253
3	1.237747e+12	6	45.247400	26.818432
4	1.237747e+12	7	32.231844	12.636959
5	1.237747e+12	5	28.107198	40.277207
6	1.237747e+12	11	41.571300	61.727806
7	1.237747e+12	1	4.295718	45.317342
8	1.237747e+12	4	35.724664	29.717300
9	1.237747e+12	3	26.620200	56.002000
10	1.237747e+12	10	51.444286	49.769524
11	1.237747e+12	9	50.907026	25.734036
12	1.237747e+12	8	28.521800	41.885908
13	1.237747e+12	2	21.154600	12.468753
14	1.237747e+12	6	45.247400	26.818432

Analysis Data

```
#=================
# # Shape of Data
#=================

df.shape
```

```
(594011, 4)
```

```
#====================
# # Columns of Data
#====================

df.columns
```

```
Index([u'timestamp', u'device_id', u'x_pos', u'y_pos'], dtype='object')
```

Analysis Data

```
#==========================================
# # Unique value of column "device_id"
#==========================================

devices = df['device_id'].unique()
print devices
```

```
[ 9  8  2  6  7  5 11  1  4  3 10]
```

```
#==========================================
# # Calculate the number of devices
#==========================================

print "Number of device: ", (len(devices))
```

```
Number of device:  11
```

Analysis Data

```
#==========================================
# # Count value of column "device_id"
#==========================================

df["device_id"].value_counts(dropna=False)
```

```
11     54001
10     54001
9      54001
8      54001
7      54001
6      54001
5      54001
4      54001
3      54001
2      54001
1      54001
Name: device_id, dtype: int64
```

Analysis Data

```
#=======================
# # Information of Data
#=======================

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 594011 entries, 0 to 594010
Data columns (total 4 columns):
timestamp    594011 non-null float64
device_id    594011 non-null int64
x_pos        594011 non-null float64
y_pos        594011 non-null float64
dtypes: float64(3), int64(1)
memory usage: 18.1 MB
```

Check null

Analysis Data

Check null

```
#==========================================================
# # Information of each device
#==========================================================

for d in devices:
    dfi = df[df["device_id"] == d]
    print "\nDevice :  ", str(d), "  len : ", len(dfi)
    print dfi.info()
```

Analysis Data

```
Device :   9   len :  54001
<class 'pandas
Int64Index: 54
Data columns (
timestamp     5
device_id     5
x_pos         5
y_pos         5
dtypes: float6
memory usage:
None

Device :   8
<class 'pandas
Int64Index: 54
Data columns (
timestamp     5
device_id     5
x_pos         5
y_pos         5
dtypes: float6
memory usage:
None
```

```
Device :   2   len :  54001
<class 'panda
Int64Index: 5
Data columns
timestamp
device_id
x_pos
y_pos
dtypes: float
memory usage:
None

Device :   6
<class 'panda
Int64Index: 5
Data columns
timestamp
device_id
x_pos
y_pos
dtypes: float
memory usage:
None
```

```
Device :   7   len :  54001
<class 'panda
Int64Index: 5
Data columns
timestamp
device_id
x_pos
y_pos
dtypes: float
memory usage:
None

Device :   5
<class 'panda
Int64Index: 5
Data columns
timestamp
device_id
x_pos
y_pos
dtypes: float
memory usage:
None
```

```
Device :   11   len :  54001
<class 'pandas.co
Int64Index: 54001
Data columns (tot
timestamp     5400
device_id     5400
x_pos         5400
y_pos         5400
dtypes: float64(3
memory usage: 2.1
None

Device :   1   le
<class 'pandas.co
Int64Index: 54001
Data columns (tot
timestamp     5400
device_id     5400
x_pos         5400
y_pos         5400
dtypes: float64(3
memory usage: 2.1
None
```

```
Device :   4   len :  54001
<class 'pandas.core.frame.DataFrame'>
Int64Index: 54001 entries, 8 to 594008
Data columns (
timestamp     5
device_id     5
x_pos         5
y_pos         5
dtypes: float6
memory usage:
None

Device :   3
<class 'pandas
Int64Index: 54
Data columns (
timestamp     5
device_id     54001 non-null int64
x_pos         54001 non-null float64
y_pos         54001 non-null float64
dtypes: float64(3), int64(1)
memory usage: 2.1 MB
None
```

```
Device :   10   len :  54001
<class 'pandas.core.frame.DataFrame'>
Int64Index: 54001 entries, 10 to 594010
Data columns (total 4 columns):
timestamp     54001 non-null float64
device_id     54001 non-null int64
x_pos         54001 non-null float64
y_pos         54001 non-null float64
dtypes: float64(3), int64(1)
memory usage: 2.1
None
```

Check null

Analysis Data

```
#==============================================================================
# # Calculate max, min of "x_pos" and "y_pos" of each device
#==============================================================================

for d in devices:
    df_d = df[df['device_id'] == d]
    print "Decices:  ", d
    print "\tx_pos (Max, Min):  ", df_d["x_pos"].max(), df_d["x_pos"].min()
    print "\ty_pos (Max, Min):  ", df_d["y_pos"].max(), df_d["y_pos"].min()
```

```
Decices:   9
        x_pos (Max, Min):   103.313426218 1.08849121823
        y_pos (Max, Min):   66.4642358073 -3.70983419268
Decices:   8
        x_pos (Max, Min):   97.0162 -1.77253
        y_pos (Max, Min):   90.8256076371 25.5194176371
Decices:   2
        x_pos (Max, Min):   99.1654 -3.8516901715
        y_pos (Max, Min):   62.7800532904 -0.210896709556
Decices:   6
```

Analysis Data

```
#=====================================
# # Describe data of each device
#=====================================

for d in devices:
    df_d = df[df['device_id'] == d]
    print "Decices:  ", d
    print df_d.describe()
```

```
Decices:   9
          timestamp  device_id         x_pos         y_pos
count  5.400100e+04    54001.0  54001.000000  54001.000000
mean   1.237748e+12        9.0     59.556763     32.680166
std    7.794445e+05        0.0     21.863482     14.985547
min    1.237747e+12        9.0      1.088491     -3.709834
25%    1.237747e+12        9.0     50.267726     21.258236
50%    1.237748e+12        9.0     61.875126     32.874136
75%    1.237749e+12        9.0     73.415926     42.902736
max    1.237749e+12        9.0    103.313426     66.464236
Decices:   8
          timestamp  device_id         x_pos         y_pos
count  5.400100e+04    54001.0  54001.000000  54001.000000
mean   1.237748e+12        8.0     42.540067     53.853585
std    7.794445e+05        0.0     19.451981     12.487354
min    1.237747e+12        8.0     -1.772530     25.519418
25%    1.237747e+12        8.0     29.681500     43.801308
50%    1.237748e+12        8.0     44.009100     54.215608
75%    1.237749e+12        8.0     54.623700     63.520808
max    1.237749e+12        8.0     97.016200     90.825608
```

Analysis Data

```
#==================
# # Describe Data
#==================

df.describe()
```

	timestamp	device_id	x_pos	y_pos
count	5.940110e+05	594011.00000	594011.000000	594011.000000
mean	1.237748e+12	6.00000	40.995397	43.856025
std	7.794380e+05	3.16228	23.624175	19.094993
min	1.237747e+12	1.00000	-3.851690	-3.709834
25%	1.237747e+12	3.00000	22.013650	29.903948
50%	1.237748e+12	6.00000	42.212144	43.642300
75%	1.237749e+12	9.00000	57.166875	57.709330
max	1.237749e+12	11.00000	104.214000	91.831700

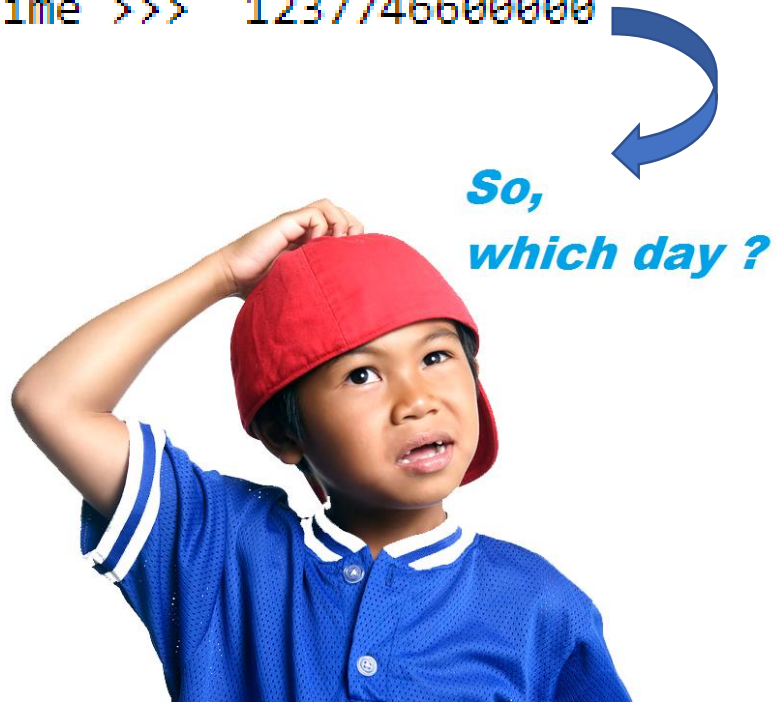
Analysis Data

```
#==================================================================
# # Find start time
#==================================================================

startTime = df["timestamp"].min()
print startTime, " startTime >>> ", int(df["timestamp"].min())
```

```
1.2377466e+12  startTime >>>  1237746600000
```

```
#==================================================================
# # Find stop time
#==================================================================

stopTime = df["timestamp"].max()
print stopTime, " Time Stop >>> ", int(df["timestamp"].max())
```

```
1.2377493e+12  Time Stop >>>  1237749300000
```



Analysis Data

```
#==========================================================
# # Convert date to '%Y-%m-%d %H:%M:%S' format
#==========================================================

def tsConvert(_t):
    s, ms = divmod(_t, 1000)
    ts = '%s.%03d' % (time.strftime('%Y-%m-%d %H:%M:%S',\
                                    time.gmtime(s)), ms)
    return ts
```

Analysis Data

```
#==========================================
# # Time Start
#==========================================

print "Time Start:  ", tsConvert(startTime)
```

```
Time Start:   2009-03-22 18:30:00.000
```

```
#==========================================
# # Time Stop
#==========================================

print "Time Stop :  ", tsConvert(stopTime)
```

```
Time Stop :   2009-03-22 19:15:00.000
```

Analysis Data

```
#==================================================================
# # Total time of the part 1
#==================================================================

part1 = stopTime - startTime
minutesPart1 = part1/60/1000

print "Total time of the part 1:  ", minutesPart1, " (minutes)"
```

```
Total time of the part 1:   45.0  (minutes)
```

No extra time

Data processing

DATA

1. Timestamp
2. Devices_id
3. Positions (x, y)

Calculate

1. Real Time
2. Distance
3. Total distance

Next workshops

- Velocity
- Zone
- Turning points

...

Calculate real time

Calculate real time

```
#==========================================================
# # Calculate real time (match Time - minute)
#==========================================================

ts = (df["timestamp"]- startTime)/1000
def matchTime_minute(_df, _ts):
    _df["matchTime_minute"] = (_ts/60).astype('int') + 1

    return _df
```

```
df = matchTime_minute(df, ts)
df
```

Calculate real time

	timestamp	device_id	x_pos	y_pos	matchTime_minute
0	1.237747e+12	9	50.875526	25.792036	1
1	1.237747e+12	8	28.525500	41.909308	1
2	1.237747e+12	2	21.108100	12.481253	1
3	1.237747e+12	6	45.247400	26.818432	1
4	1.237747e+12	7	32.231844	12.636959	1
5	1.237747e+12	5	28.107198	40.277207	1
6	1.237747e+12	11	41.571300	61.727806	1
7	1.237747e+12	1	4.295718	45.317342	1
8	1.237747e+12	4	35.724664	29.717300	1
9	1.237747e+12	3	26.620200	56.002000	1
10	1.237747e+12	10	51.444286	49.769524	1
11	1.237747e+12	9	50.907026	25.734036	1

Calculate real time

```
#==========================================================
# # Calculate real time (match Time - second)
#==========================================================

def matchTime_second(_df, _ts):
    _df["matchTime_second"] = (_ts%60).astype('int') + 1

    return _df
```

```
df = matchTime_second(df, ts)
df
```

Calculate real time

	timestamp	device_id	x_pos	y_pos	matchTime_minute	matchTime_second
0	1.237747e+12	9	50.875526	25.792036	1	1
1	1.237747e+12	8	28.525500	41.909308	1	1
2	1.237747e+12	2	21.108100	12.481253	1	1
3	1.237747e+12	6	45.247400	26.818432	1	1
4	1.237747e+12	7	32.231844	12.636959	1	1
5	1.237747e+12	5	28.107198	40.277207	1	1
6	1.237747e+12	11	41.571300	61.727806	1	1
7	1.237747e+12	1	4.295718	45.317342	1	1
8	1.237747e+12	4	35.724664	29.717300	1	1
9	1.237747e+12	3	26.620200	56.002000	1	1
10	1.237747e+12	10	51.444286	49.769524	1	1
11	1.237747e+12	9	50.907026	25.734036	1	1
12	1.237747e+12	8	28.521800	41.885908	1	1

Calculate real time

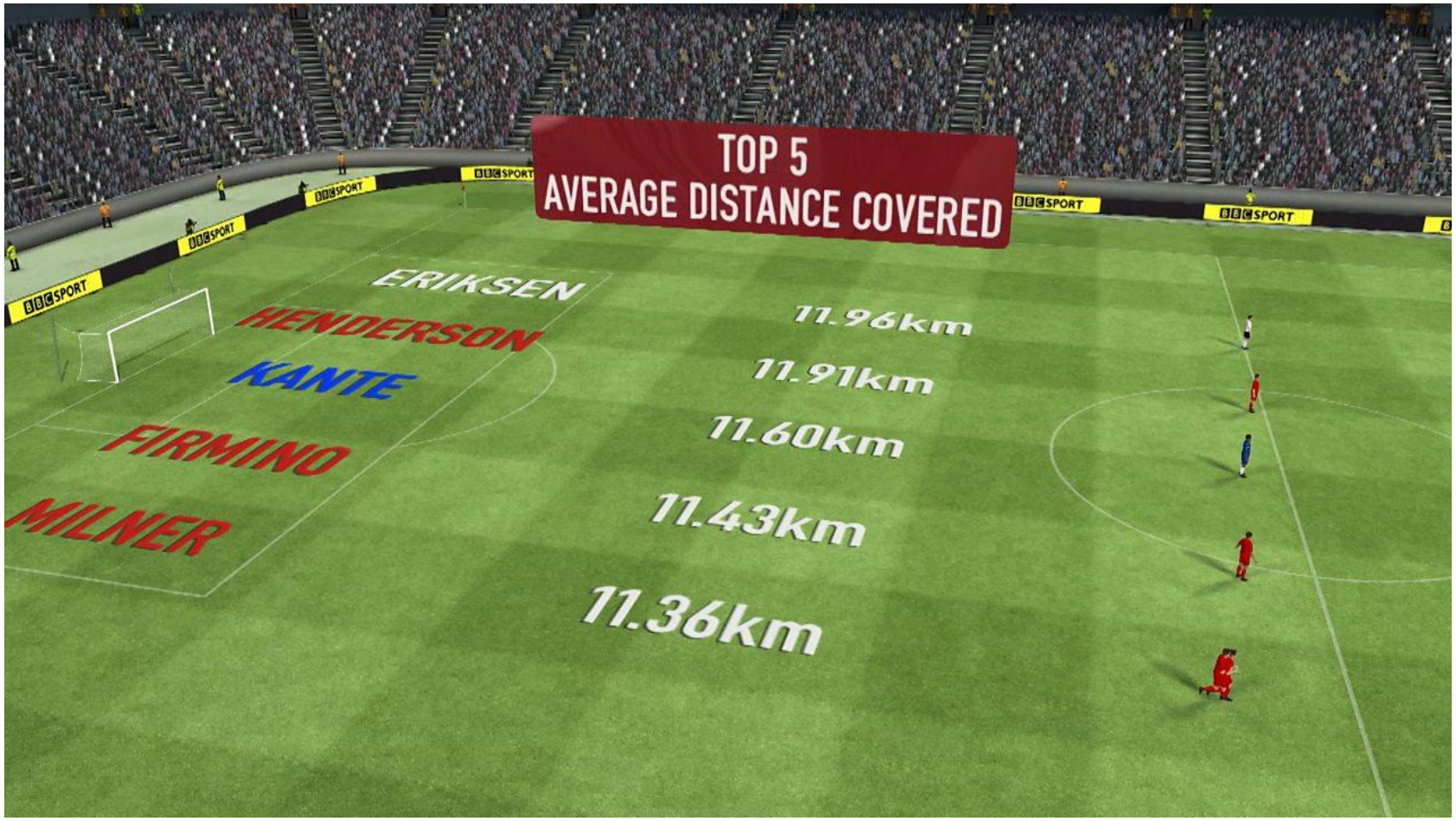
```python
#==========================================
# # Calculate real time (match Time)
#==========================================

def matchTime(_df, _ts):
    m, s = divmod(_ts, 60)
    _df["matchTime_minute"] = (m + 1).astype(int)
    _df["matchTime_second"] = (s + 1).astype(int)

    return _df
```

```python
df = matchTime(df, ts)
df
```

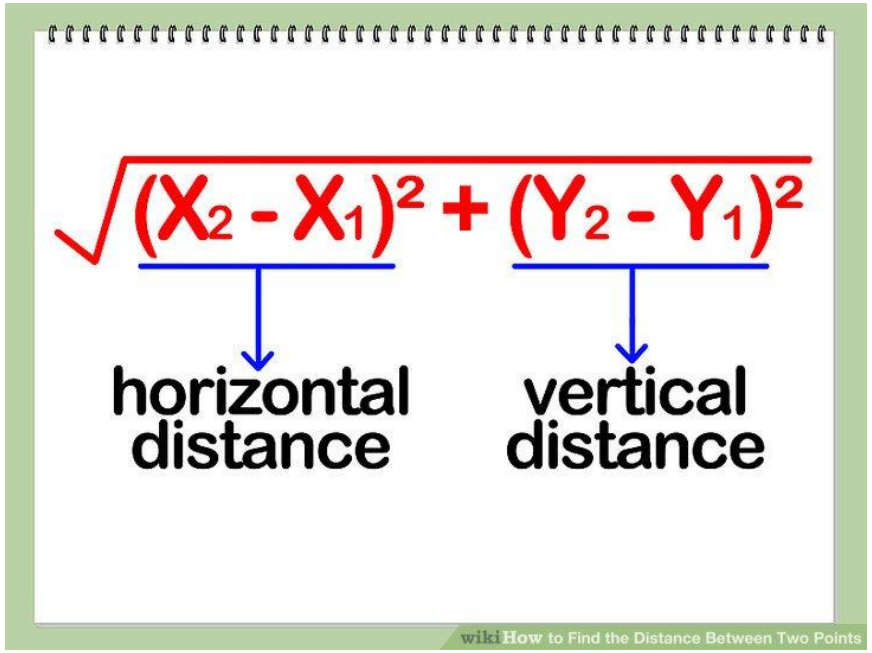
Calculate real time

	timestamp	device_id	x_pos	y_pos	matchTime_minute	matchTime_second
0	1.237747e+12	9	50.875526	25.792036	1	1
1	1.237747e+12	8	28.525500	41.909308	1	1
2	1.237747e+12	2	21.108100	12.481253	1	1
3	1.237747e+12	6	45.247400	26.818432	1	1
4	1.237747e+12	7	32.231844	12.636959	1	1
5	1.237747e+12	5	28.107198	40.277207	1	1
6	1.237747e+12	11	41.571300	61.727806	1	1
7	1.237747e+12	1	4.295718	45.317342	1	1
8	1.237747e+12	4	35.724664	29.717300	1	1
9	1.237747e+12	3	26.620200	56.002000	1	1
10	1.237747e+12	10	51.444286	49.769524	1	1
11	1.237747e+12	9	50.907026	25.734036	1	1

Calculate distance



Calculate distance




```
#==========================================================
# # Calculate the distance between 2 points
#==========================================================

def dist(x1, y1, x2, y2):
    return "%.4f" %  sqrt((x2 - x1)**2 + (y2 - y1)**2 )
```

Calculate distance

```
#==================================================================
# # Calculate distance
#==================================================================

def distance(_df, _devices):
    _df['distance'] = 0
    for d in _devices:
        idd = _df[_df['device_id'] == d].index
        lend = len(idd)

        for i in range (1, lend):
            _df.loc[idd[i], 'distance']= dist(_df.loc[idd[i], 'x_pos'],\
            _df.loc[idd[i], 'y_pos'], _df.loc[idd[i-1], 'x_pos'],\
            _df.loc[idd[i-1], 'y_pos'])
    return _df
```

```
df = distance(df, devices)
df
```

Time run so long?? Why??

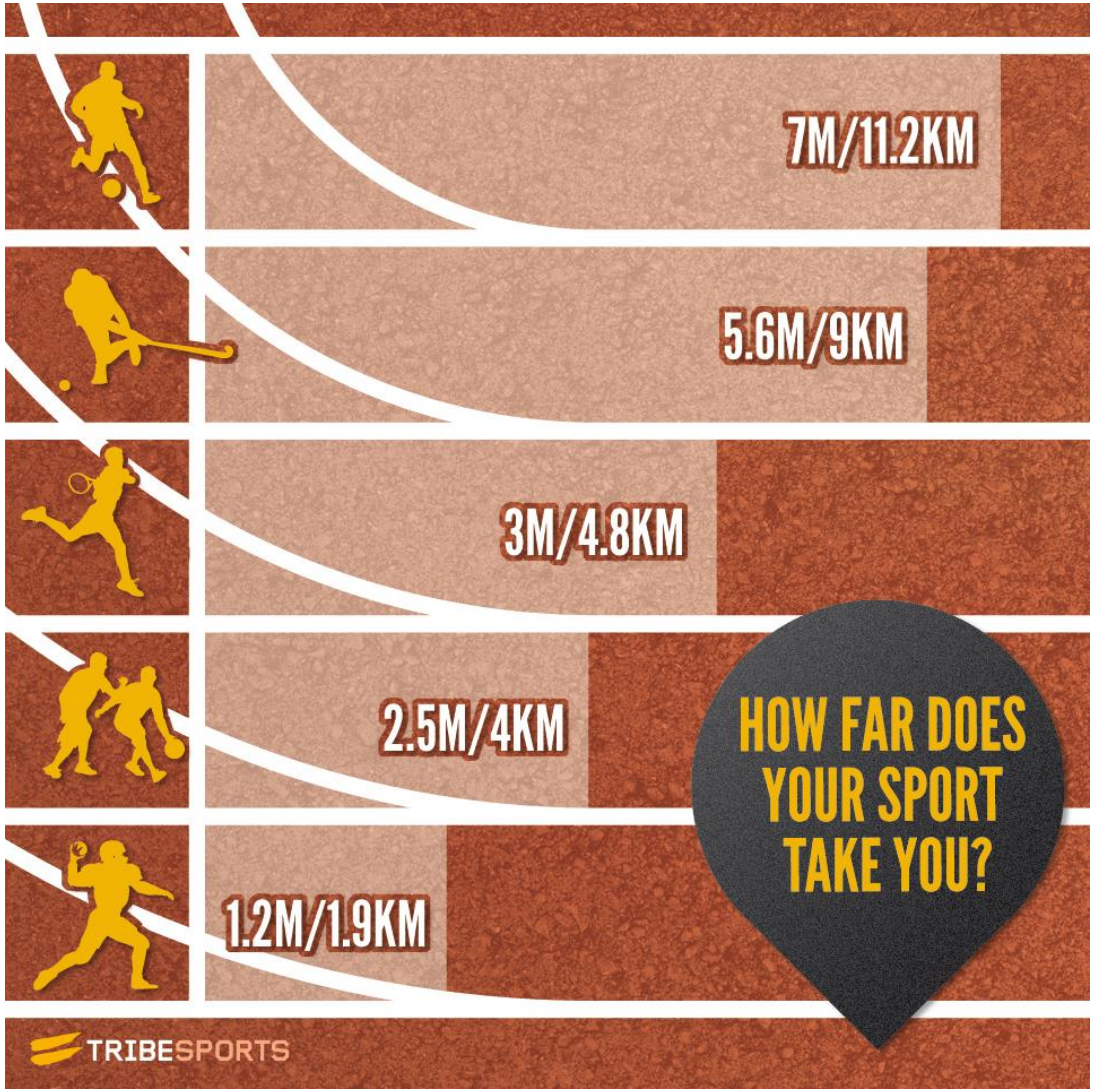
Calculate distance

	timestamp	device_id	x_pos	y_pos	matchTime_minute	matchTime_second	distance
0	1.237747e+12	9	50.875526	25.792036	1	1	0
1	1.237747e+12	8	28.525500	41.909308	1	1	0
2	1.237747e+12	2	21.108100	12.481253	1	1	0
3	1.237747e+12	6	45.247400	26.818432	1	1	0
4	1.237747e+12	7	32.231844	12.636959	1	1	0
5	1.237747e+12	5	28.107198	40.277207	1	1	0
6	1.237747e+12	11	41.571300	61.727806	1	1	0
7	1.237747e+12	1	4.295718	45.317342	1	1	0
8	1.237747e+12	4	35.724664	29.717300	1	1	0
9	1.237747e+12	3	26.620200	56.002000	1	1	0
10	1.237747e+12	10	51.444286	49.769524	1	1	0
11	1.237747e+12	9	50.907026	25.734036	1	1	0.0660
12	1.237747e+12	8	28.521800	41.885908	1	1	0.0237
13	1.237747e+12	2	21.154600	12.468753	1	1	0.0482

Calculate total distance

Length

1.2	=	1.93121
Mile		Kilometre



Calculate total distance

```
#==============================================================================
# # Calculate Total Distance
#==============================================================================

def distanceTotal(_df, _devices):
    _df['totalDistance'] = 0
    for d in _devices:
        idd = _df[_df['device_id'] == d].index
        lend = len(idd)

        for i in range (1, lend):
            _df.loc[idd[i], 'totalDistance'] = _df.loc[idd[i-1], 'totalDistance'] +\
                                                float(_df.loc[idd[i], 'distance'] )
    return _df
```

```
df = distanceTotal(df, devices)
df
```

Calculate total distance

	timestamp	device_id	x_pos	y_pos	matchTime_minute	matchTime_second	distance	totalDistance
0	1.237747e+12	9	50.875526	25.792036	1	1	0	0.0000
1	1.237747e+12	8	28.525500	41.909308	1	1	0	0.0000
2	1.237747e+12	2	21.108100	12.481253	1	1	0	0.0000
3	1.237747e+12	6	45.247400	26.818432	1	1	0	0.0000
4	1.237747e+12	7	32.231844	12.636959	1	1	0	0.0000
5	1.237747e+12	5	28.107198	40.277207	1	1	0	0.0000
6	1.237747e+12	11	41.571300	61.727806	1	1	0	0.0000
7	1.237747e+12	1	4.295718	45.317342	1	1	0	0.0000
8	1.237747e+12	4	35.724664	29.717300	1	1	0	0.0000
9	1.237747e+12	3	26.620200	56.002000	1	1	0	0.0000
10	1.237747e+12	10	51.444286	49.769524	1	1	0	0.0000
11	1.237747e+12	9	50.907026	25.734036	1	1	0.0660	0.0660
12	1.237747e+12	8	28.521800	41.885908	1	1	0.0237	0.0237
13	1.237747e+12	2	21.154600	12.468753	1	1	0.0482	0.0482

Describe Data

```
#==================
# # Describe Data
#==================

df.describe()
```

	timestamp	device_id	x_pos	y_pos	matchTime_minute	matchTime_second	distance	totalDistance
count	5.940110e+05	594011.00000	594011.000000	594011.000000	594011.000000	594011.000000	594011.000000	594011.000000
mean	1.237748e+12	6.00000	40.995397	43.856025	23.000426	30.499454	0.112526	3062.034685
std	7.794380e+05	3.16228	23.624175	19.094993	12.987441	17.318422	0.070992	1757.739790
min	1.237747e+12	1.00000	-3.851690	-3.709834	1.000000	1.000000	0.000000	0.000000
25%	1.237747e+12	3.00000	22.013650	29.903948	12.000000	15.000000	0.058500	1598.513550
50%	1.237748e+12	6.00000	42.212144	43.642300	23.000000	30.000000	0.099900	3025.199000
75%	1.237749e+12	9.00000	57.166875	57.709330	34.000000	45.000000	0.155600	4474.828350
max	1.237749e+12	11.00000	104.214000	91.831700	46.000000	60.000000	0.488900	7938.709100

Describe Data

```
#==========================================
# # Describe Data of each device
#==========================================

for d in devices:
    dfi = df[df["device_id"] == d]
    print "Device >>> ", d, "\n", dfi.describe()
```

Describe Data

```
Device >>>  5
          timestamp  device_id         x_pos         y_pos  matchTime_minute  \
count  5.400100e+04    54001.0  54001.000000  54001.000000      54001.000000
mean   1.237748e+12        5.0     36.478316     47.720794         23.000426
std    7.794445e+05        0.0     17.621164      8.959426         12.987550
min    1.237747e+12        5.0     -0.550356     28.417807          1.000000
25%    1.237747e+12        5.0     24.025098     40.926907         12.000000
50%    1.237748e+12        5.0     39.065498     46.599707         23.000000
75%    1.237749e+12        5.0     47.971998     53.998407         34.000000
max    1.237749e+12        5.0     76.083398     71.084607         46.000000

       matchTime_second      distance  totalDistance
count      54001.000000  54001.000000   54001.000000
mean          30.499454      0.101851    2771.902602
std           17.318568      0.066091    1530.431334
min            1.000000      0.000000       0.000000
25%           15.000000      0.052000    1490.923000
50%           30.000000      0.089000    2719.112100
75%           45.000000      0.140100    4054.630200
max           60.000000      0.463000    5500.053200
```

Describe Data

1. Time : 45 (m)
2. Players: 11
3. No changed players

1. Timestamp
2. Devices_id
3. Positions (x, y)
4. matchTime_minute
5. matchTime_second
6. Distance
7. totalDistance

Write to CSV

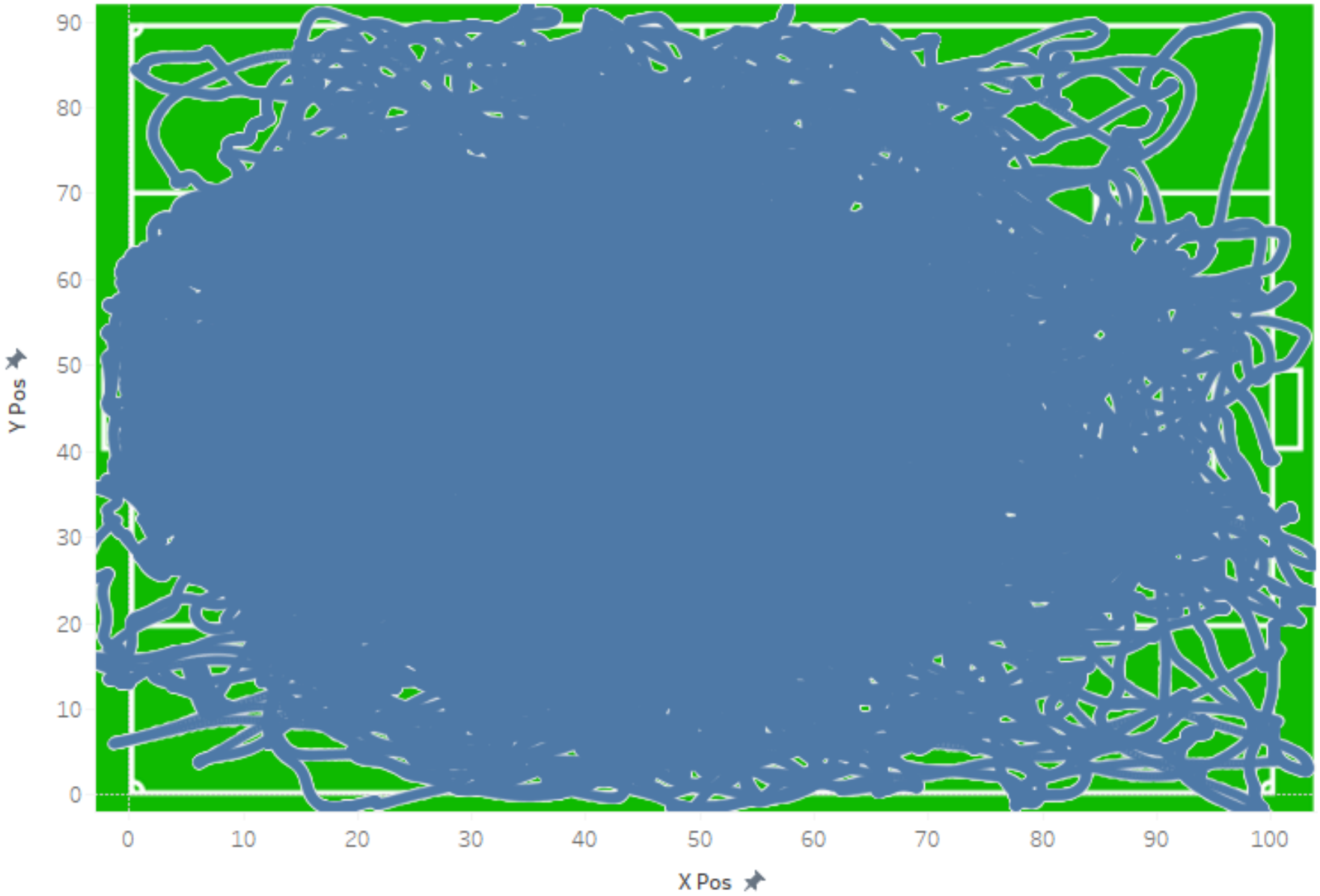
```
#==============================================================
# Write Data to CSV file
#==============================================================

def writeCSV(_df, _csvName, _sep):
    _df.to_csv(_csvName, sep= _sep, header=True, index=False)
```

```
#==============================================================
# # Write to csv file
#==============================================================

writeCSV(df, "ThuyLe_P1_Demo_Distance.csv", sep)
```

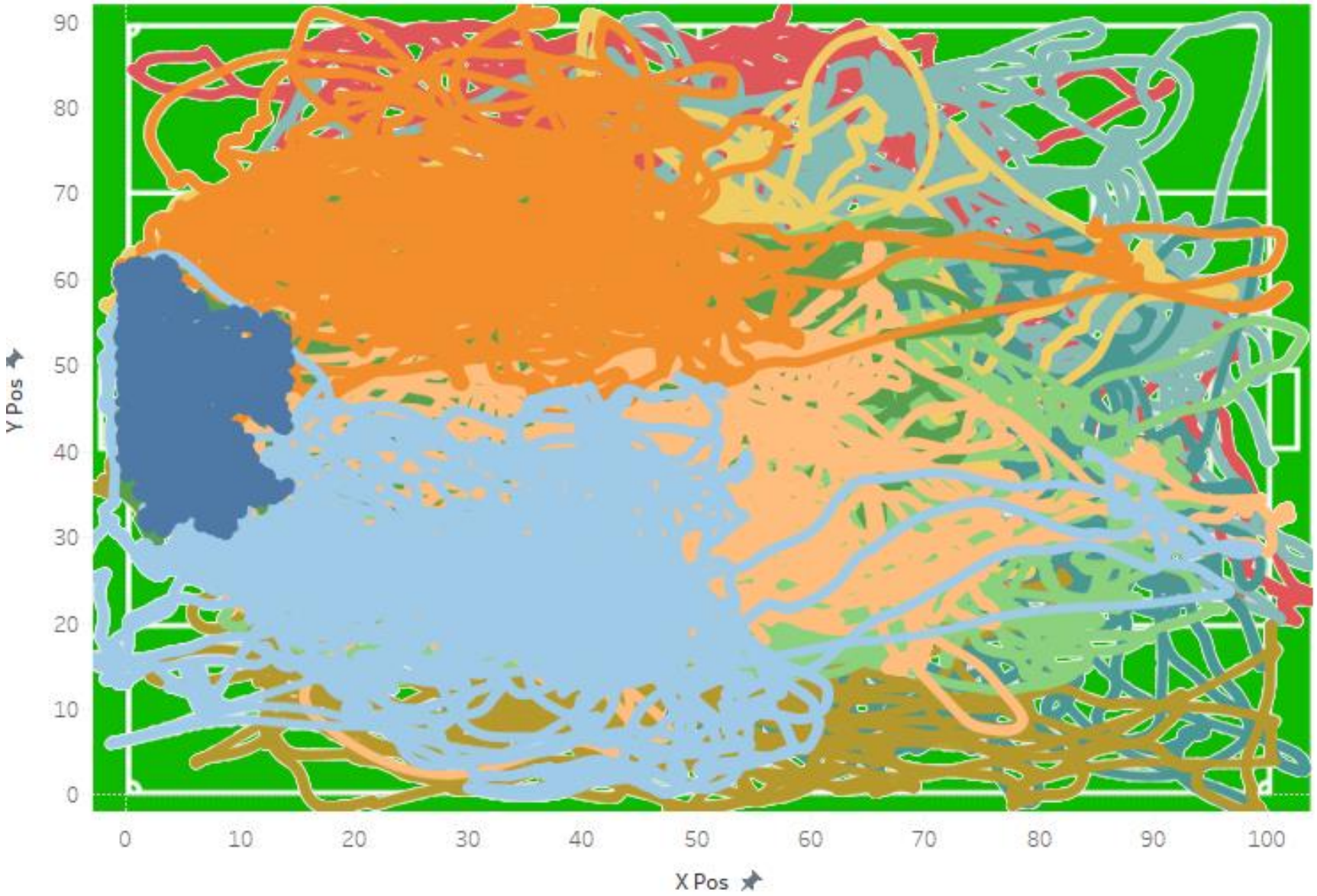
OUTLINE

1. Data gathering
2. Data analysis with Python
3. Data visualization with Tableau

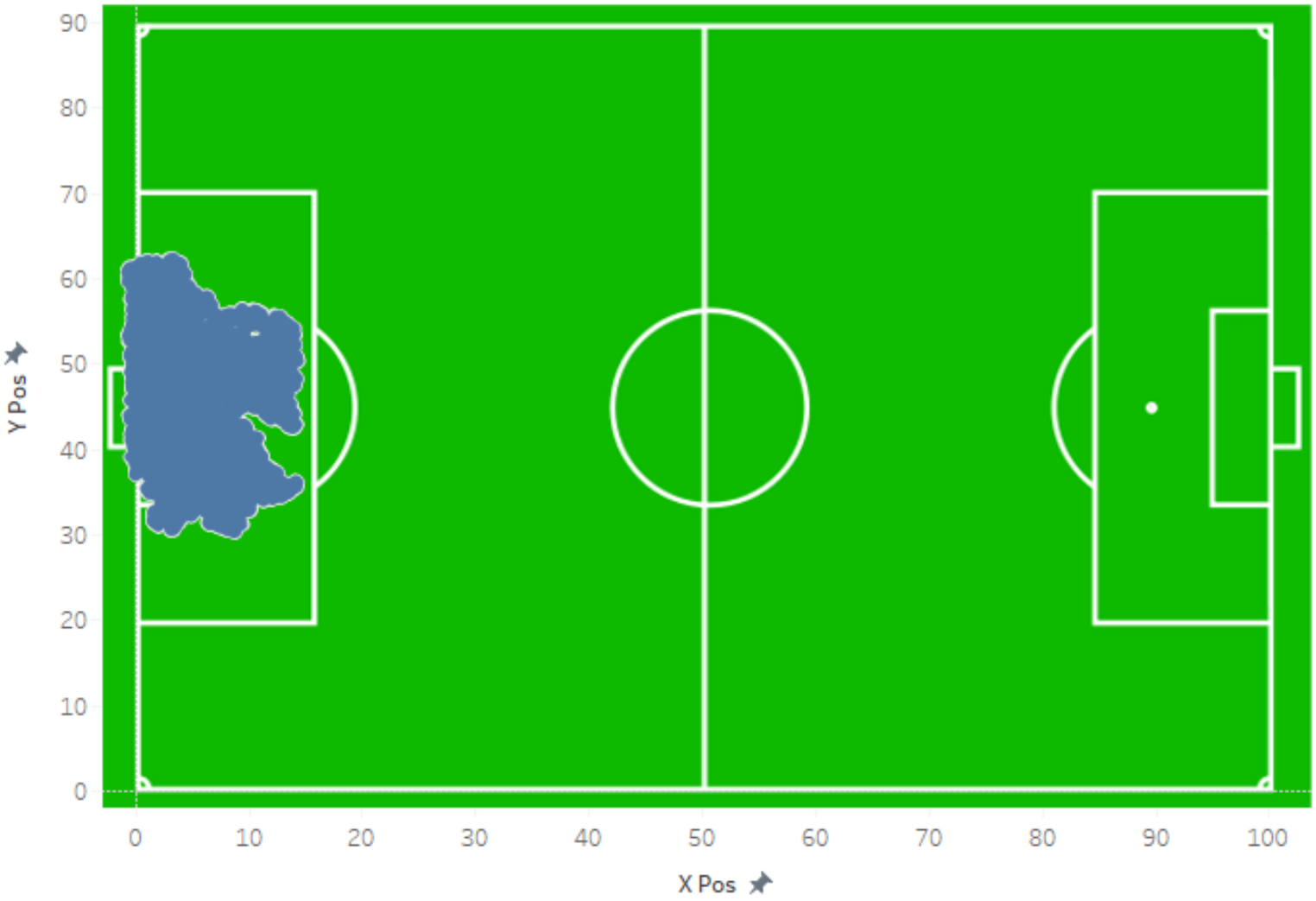
Visualize



Visualize

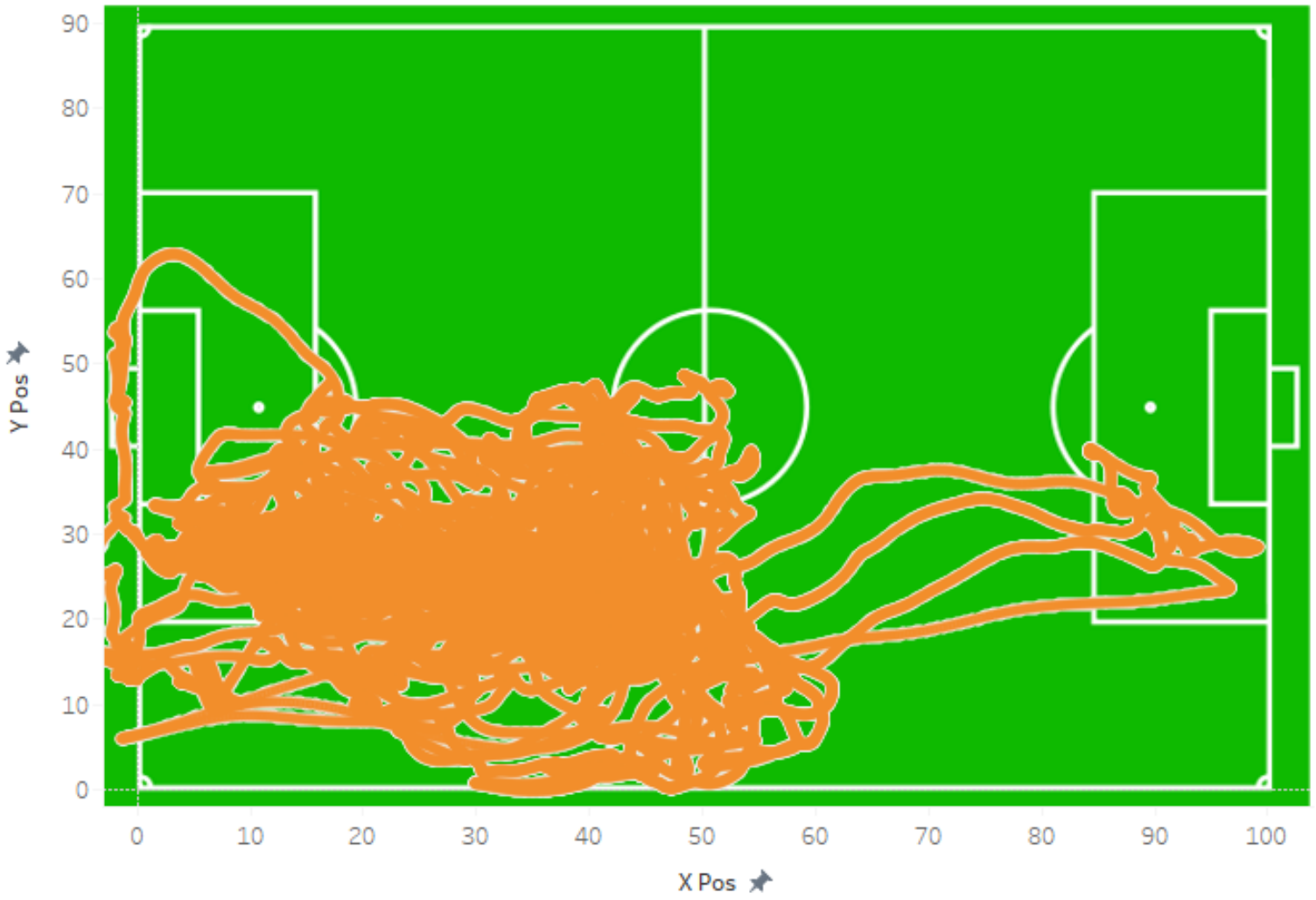


Visualize



Goalkeeper

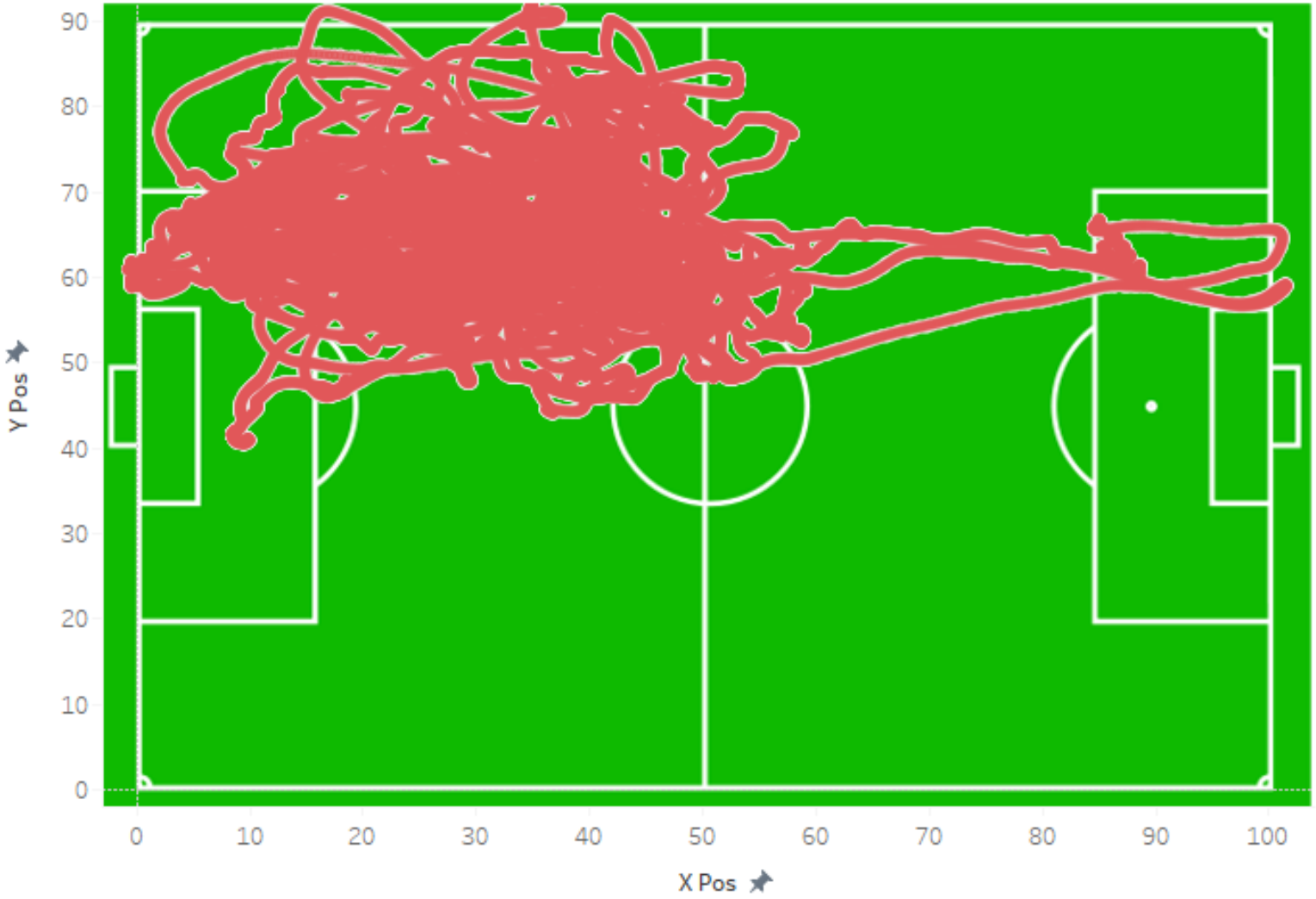
Visualize



Defender

Right back

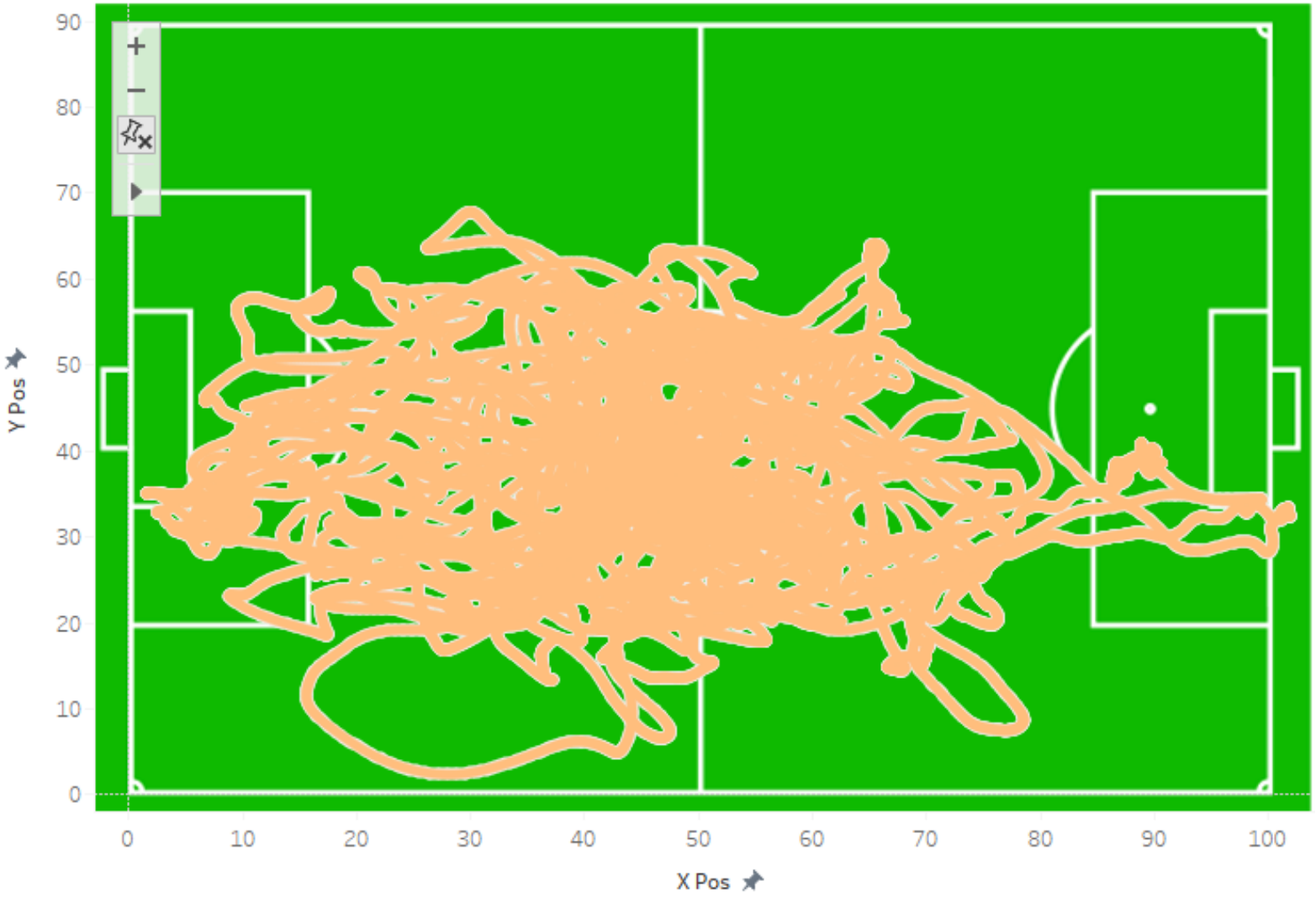
Visualize



Defender

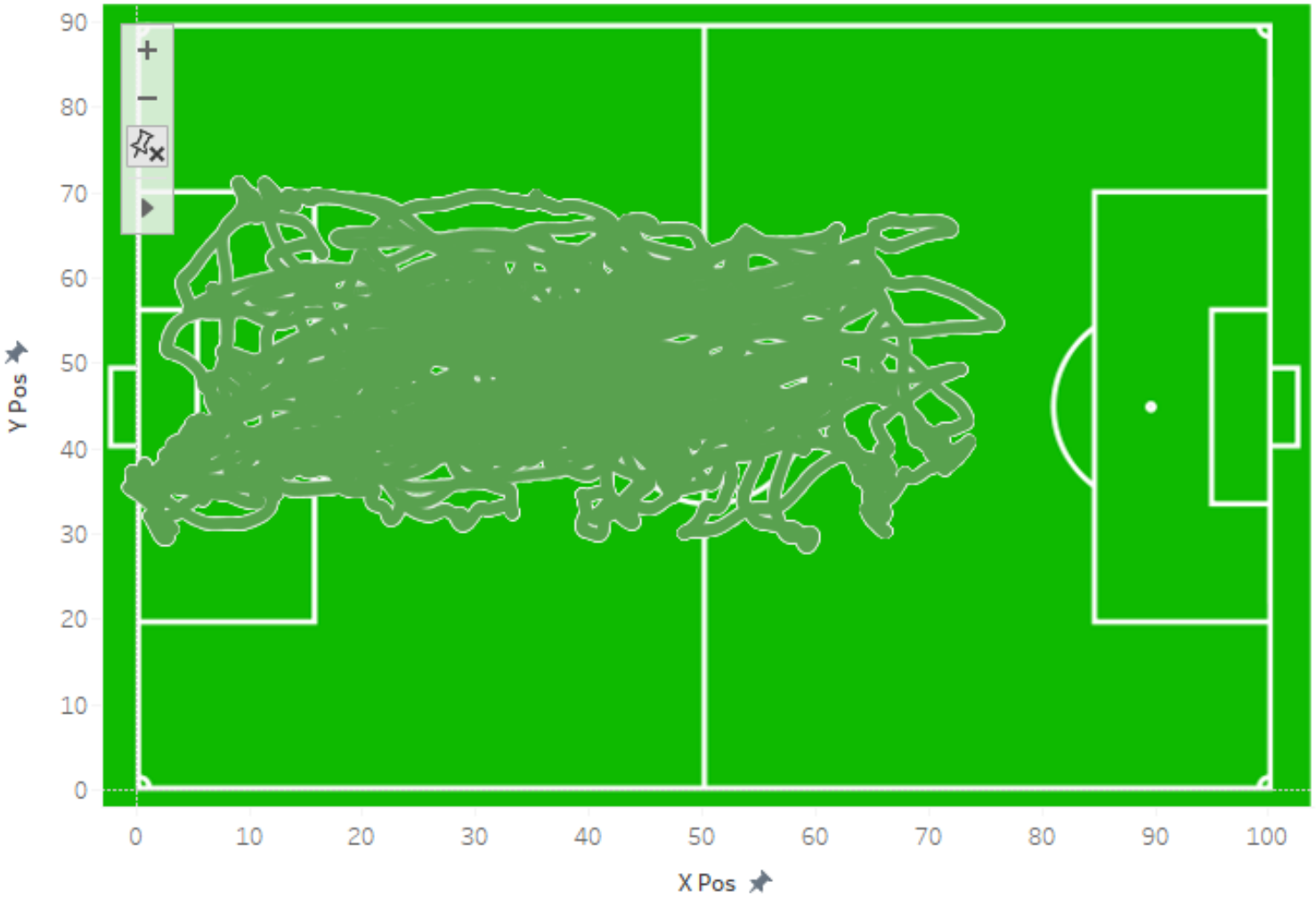
Left back

Visualize



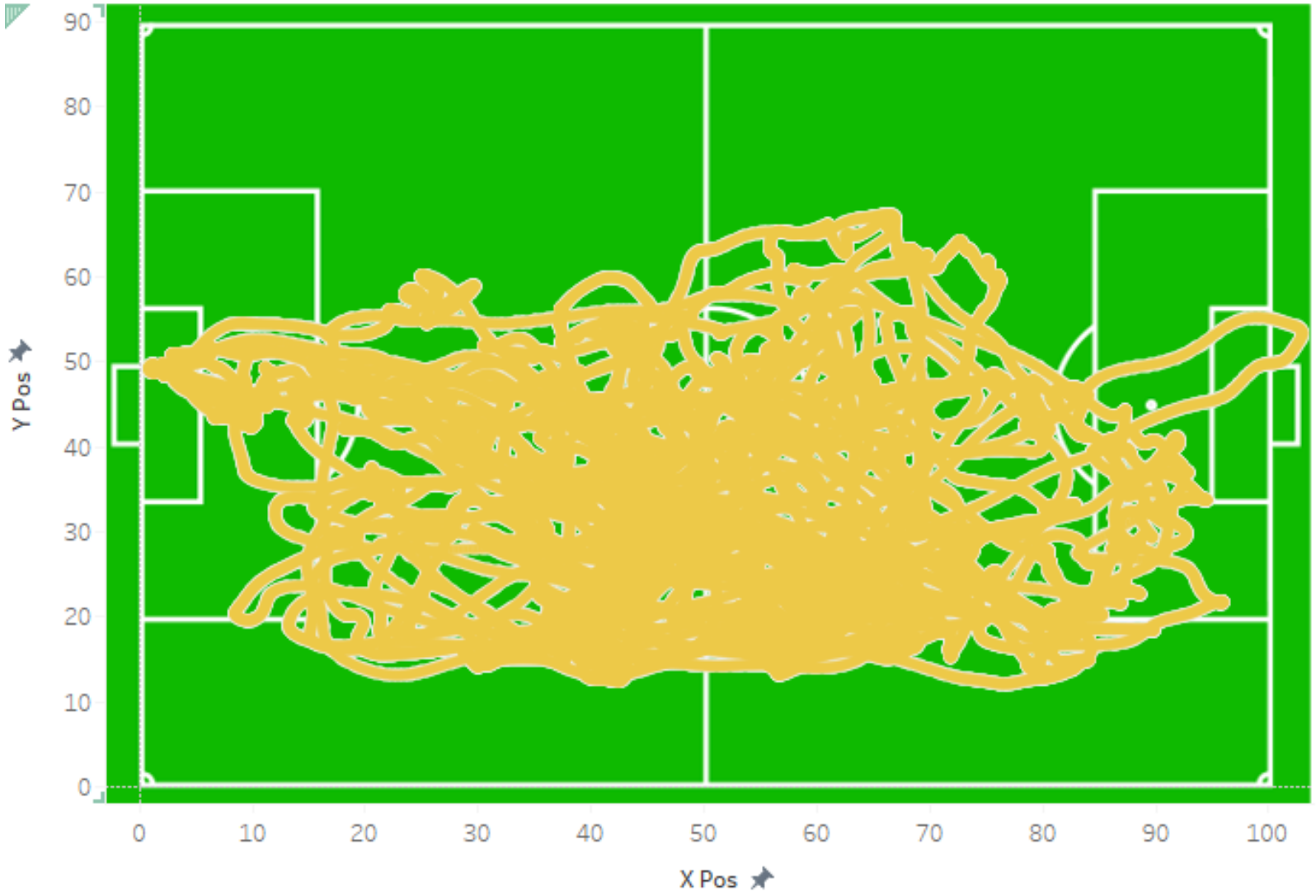
Defender

Visualize



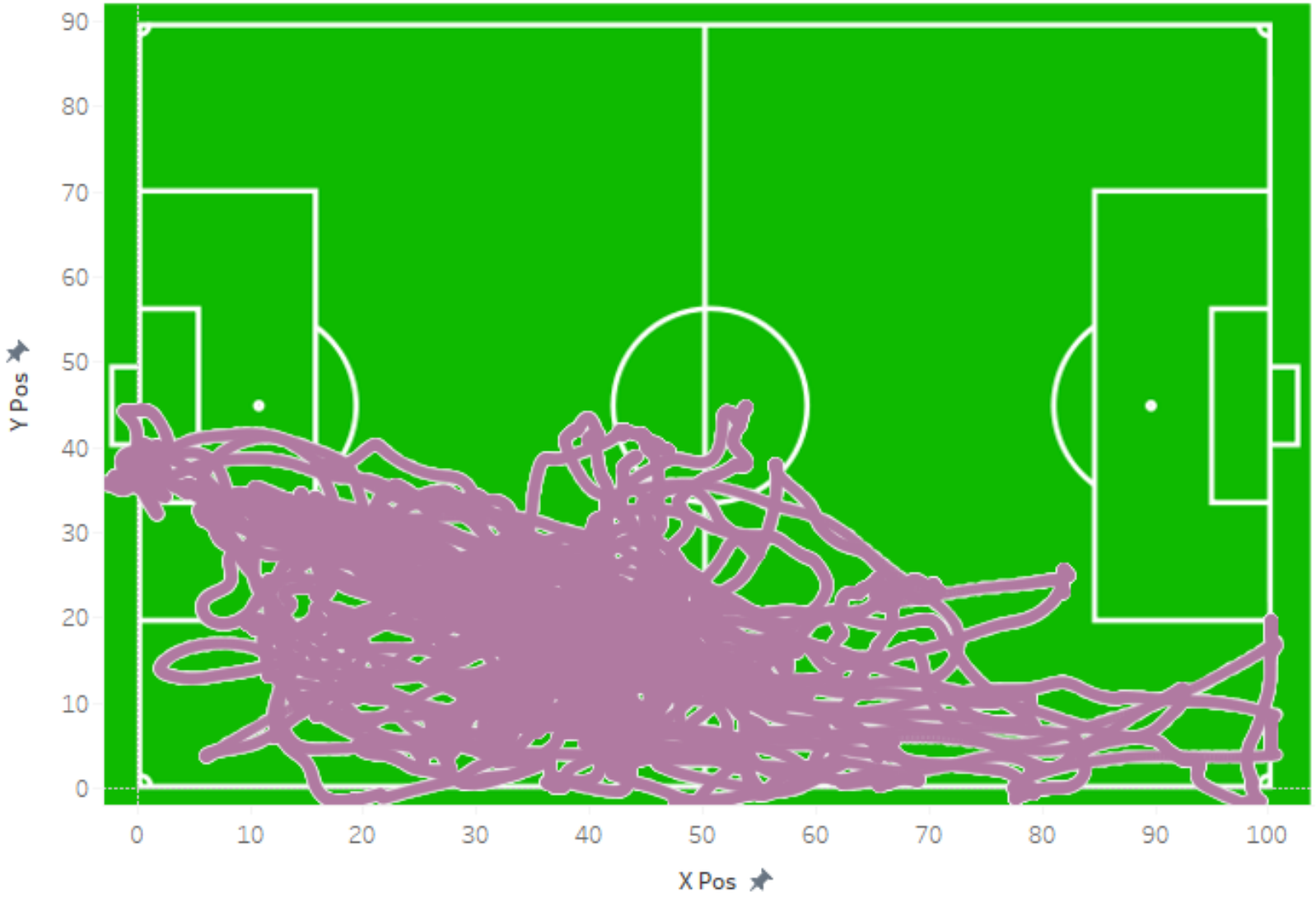
Defender

Visualize



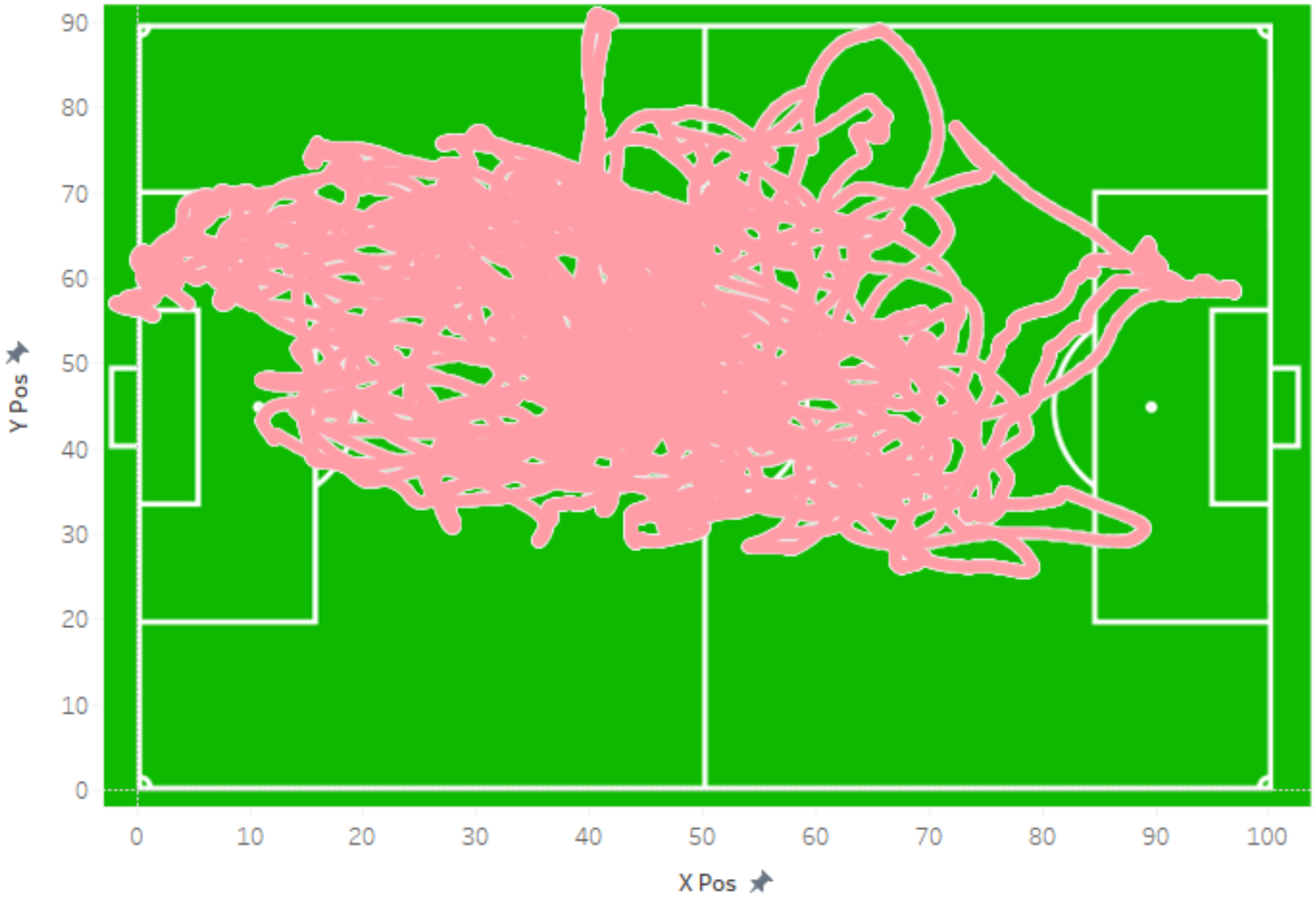
Midfielder

Visualize



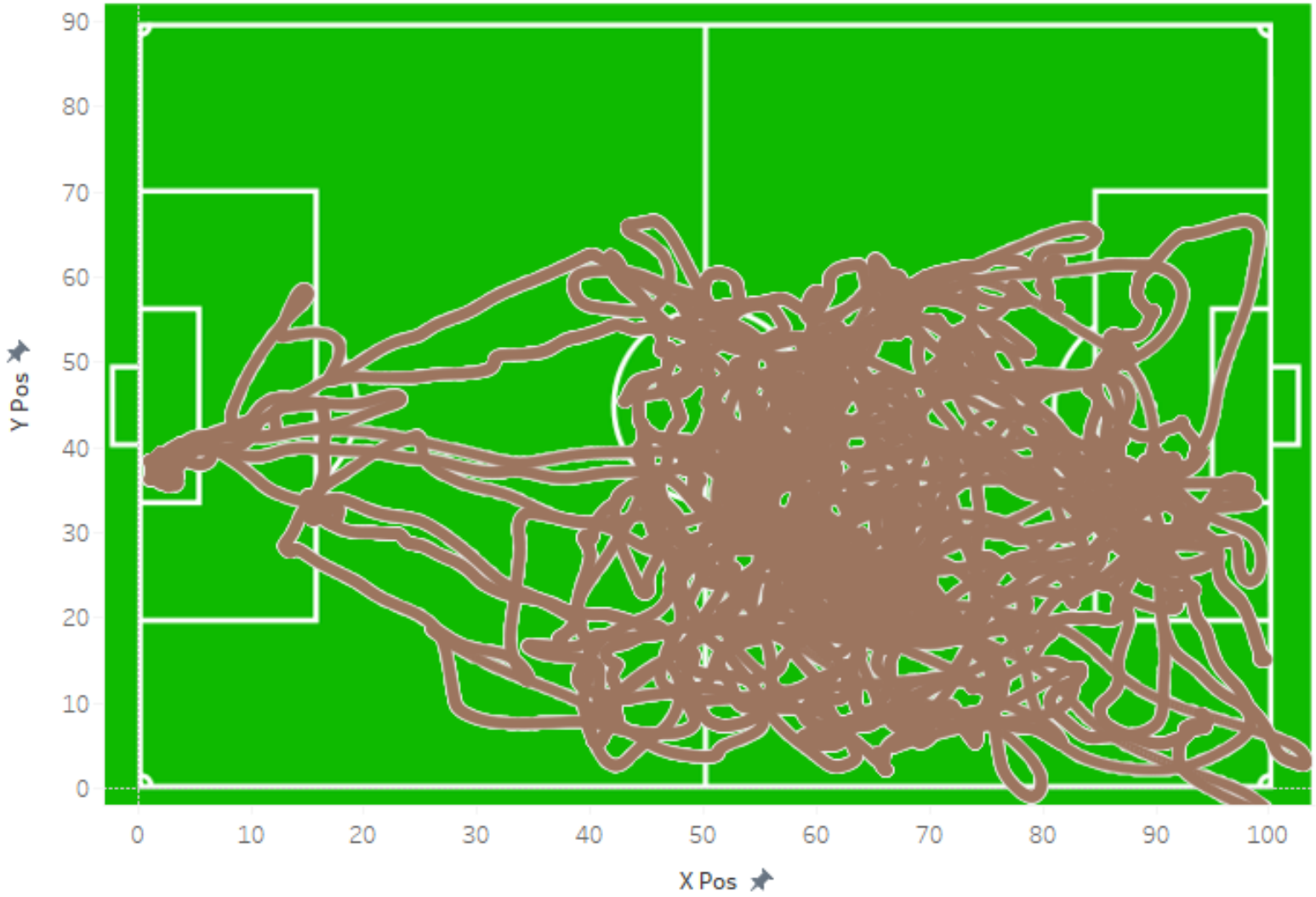
Right midfielder

Visualize



Midfielder

Visualize



Striker

Visualize

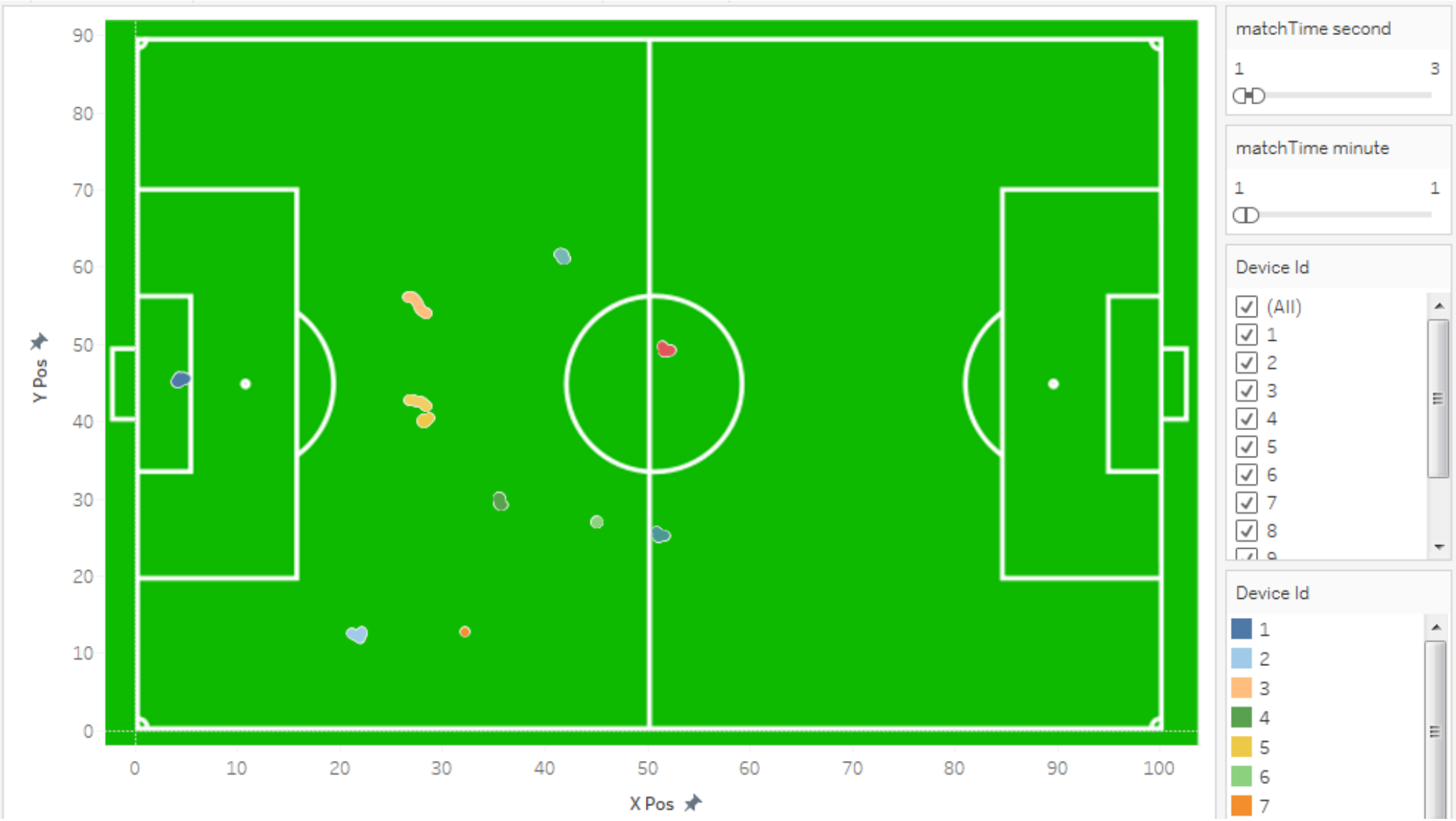
Forward

Visualize

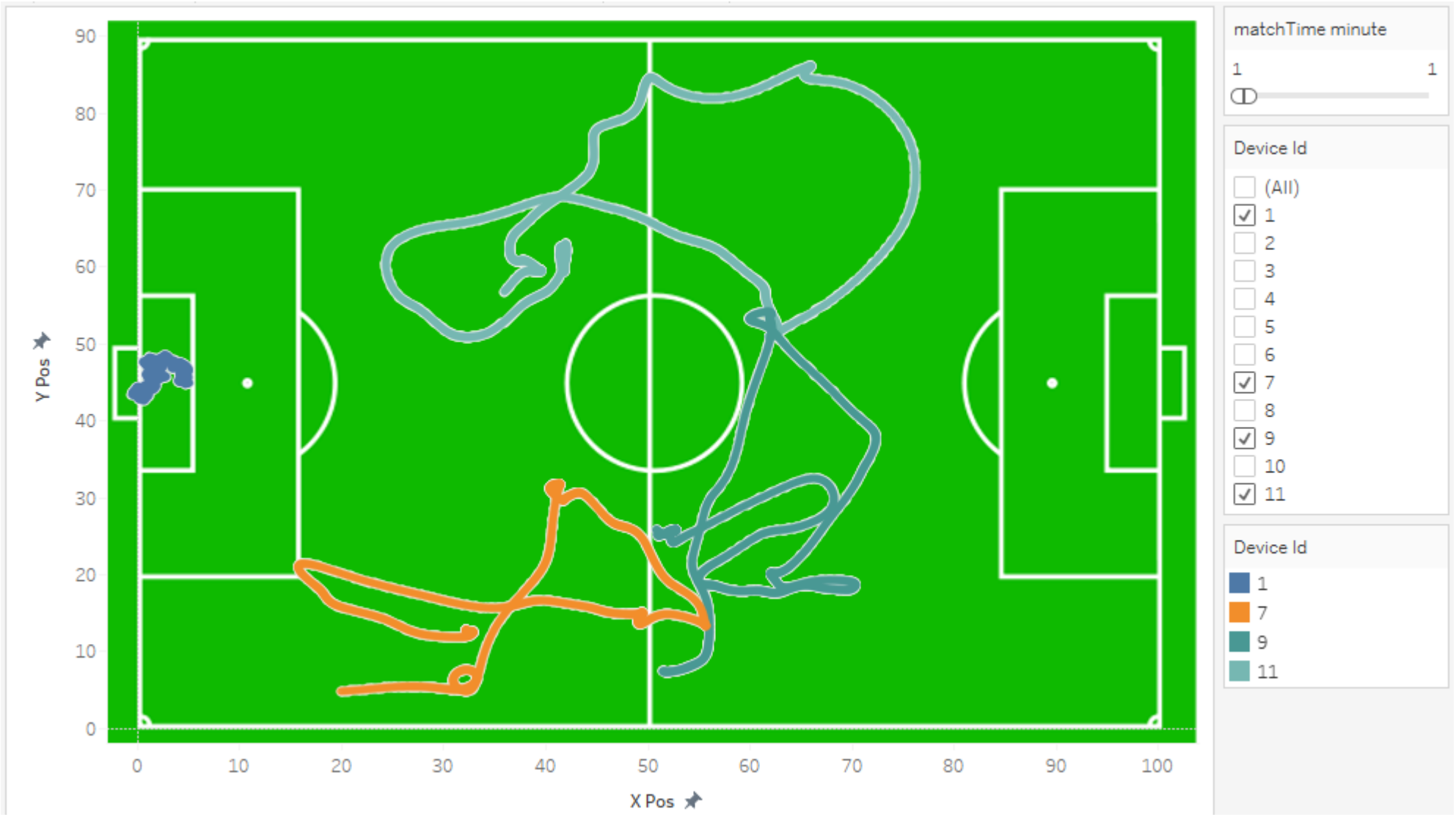
Left midfielder


The 4-4-2 Formation
3
11
5
8
10
1
9
4
6
2
7

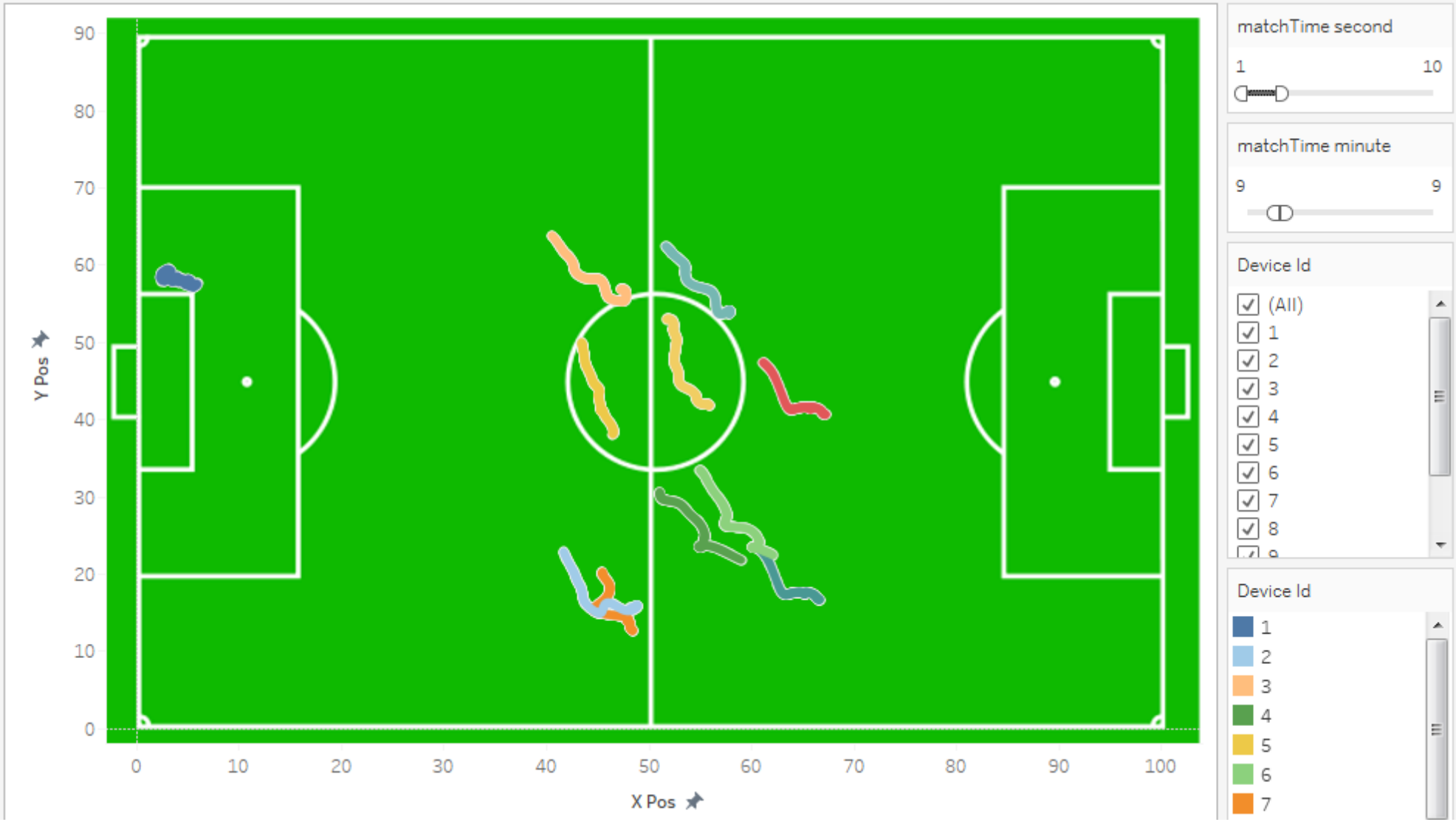
Visualize


Y Pos
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
X Pos
matchTime second
1
3
matchTime minute
1
1
Device Id
(All)
1
2
3
4
5
6
7
8
Device Id
1
2
3
4
5
6
7

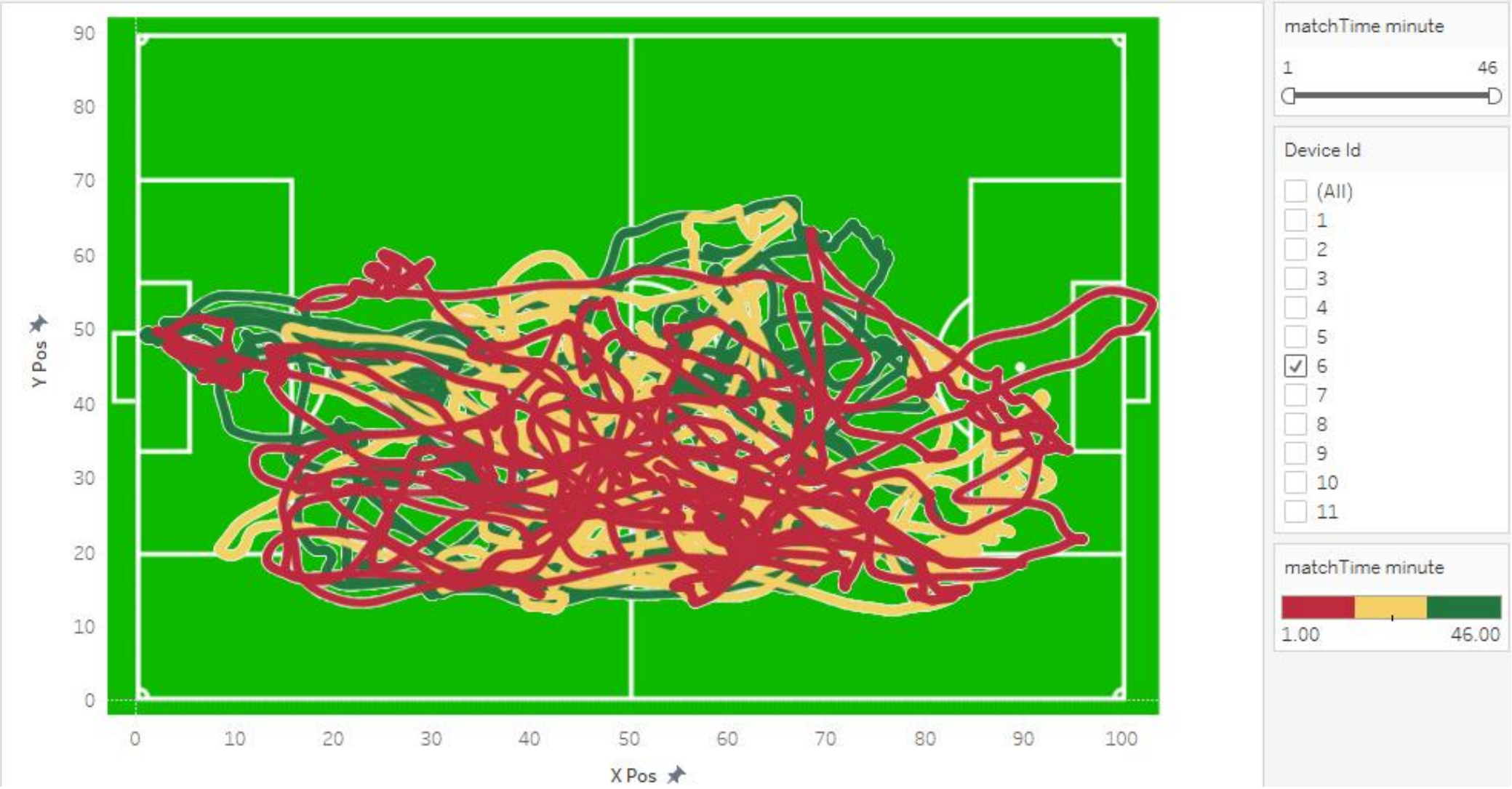
Visualize


Y Pos
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
X Pos
matchTime minute
1
1
Device Id
(All)
1
2
3
4
5
6
7
8
9
10
11
Device Id
1
7
9
11

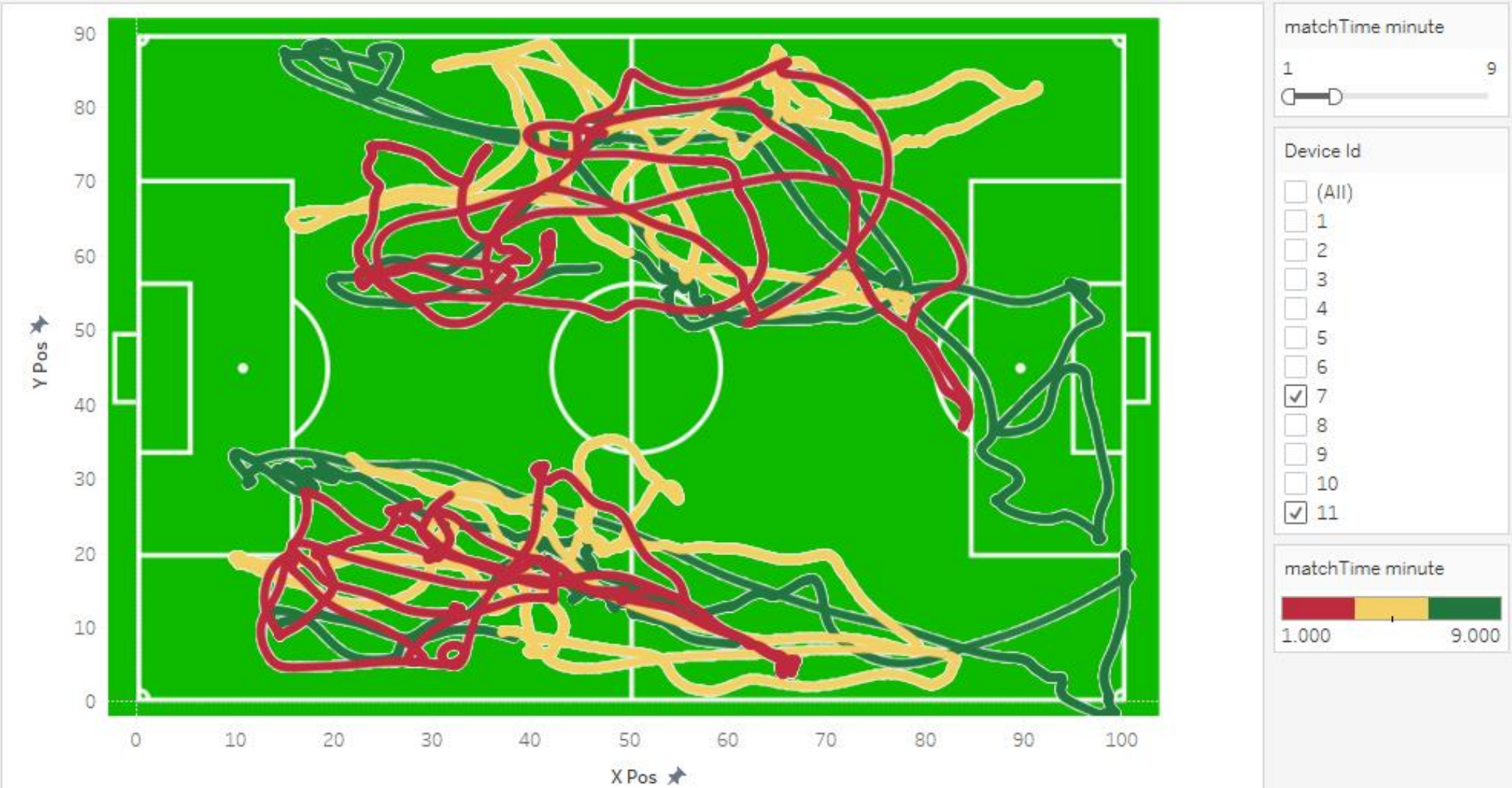
Visualize


Y Pos
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
X Pos
matchTime second
1
10
matchTime minute
9
9
Device Id
(All)
1
2
3
4
5
6
7
8
Device Id
1
2
3
4
5
6
7

Visualize


90
80
70
60
50
40
30
20
10
0
Y Pos
0
10
20
30
40
50
60
70
80
90
100
X Pos
matchTime minute
1
46
Device Id
(All)
1
2
3
4
5
6
7
8
9
10
11
matchTime minute
1.00
46.00

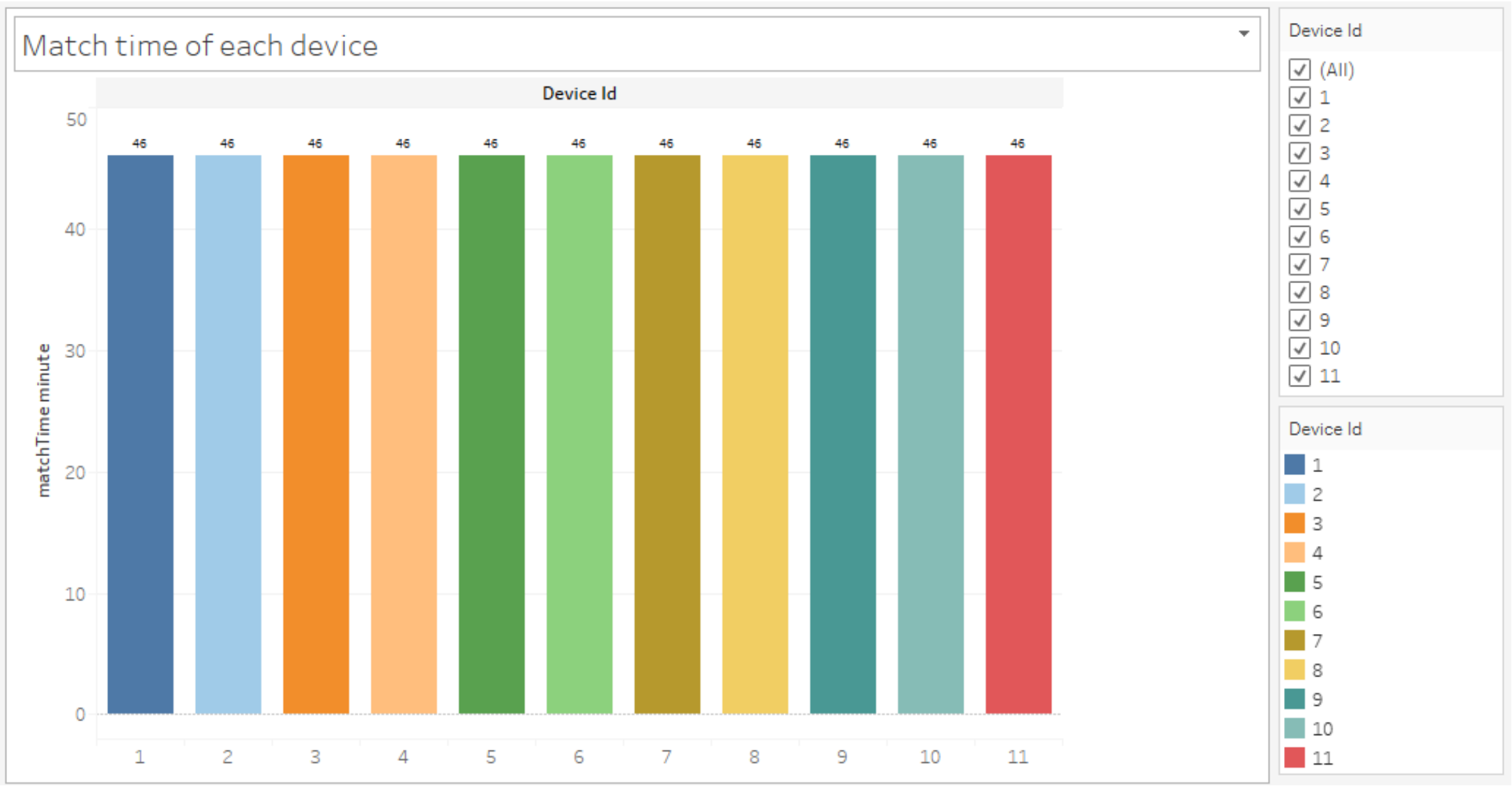
Visualize


90
80
70
60
50
40
30
20
10
0
Y Pos
0
10
20
30
40
50
60
70
80
90
100
X Pos
matchTime minute
1
9
Device Id
(All)
1
2
3
4
5
6
7
8
9
10
11
matchTime minute
1.000
9.000

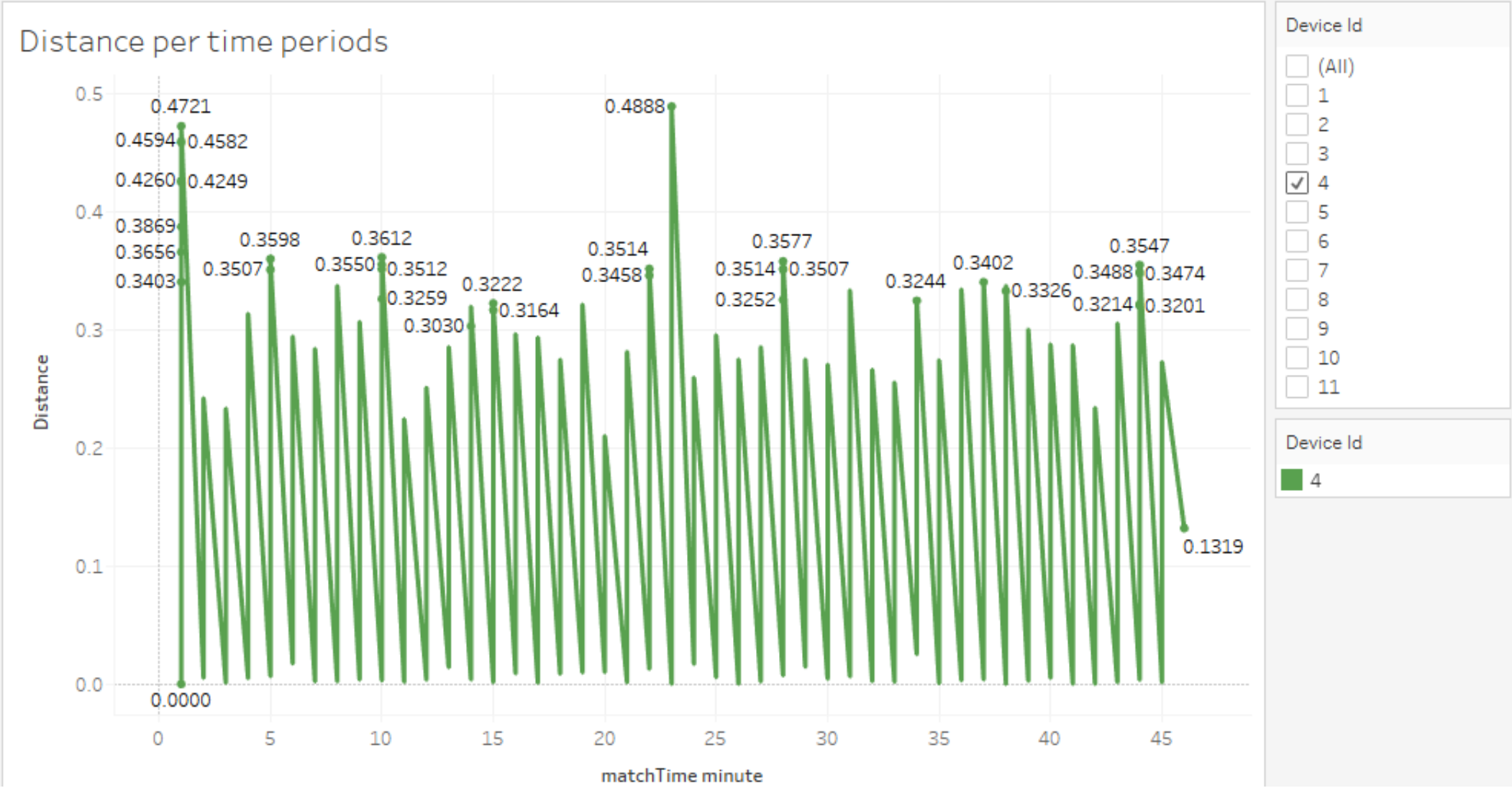
Visualize


Y Pos
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
X Pos
matchTime second
1
60
matchTime minute
1
9
Device Id
(All)
1
2
3
4
5
6
7
8
9
10
11
matchTime second
1.00
60.00

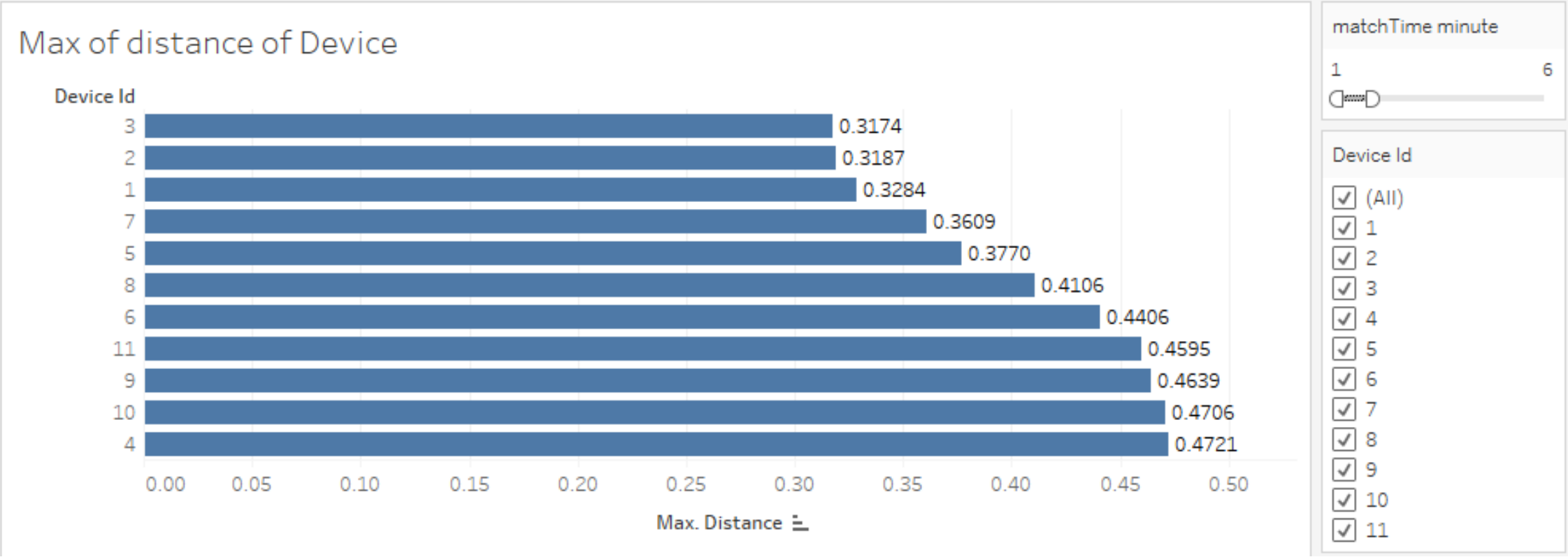
Visualize


Match time of each device
Device Id
matchTime minute
50
40
30
20
10
0
46
1
2
3
4
5
6
7
8
9
10
11
Device Id
(All)
1
2
3
4
5
6
7
8
9
10
11
Device Id
1
2
3
4
5
6
7
8
9
10
11

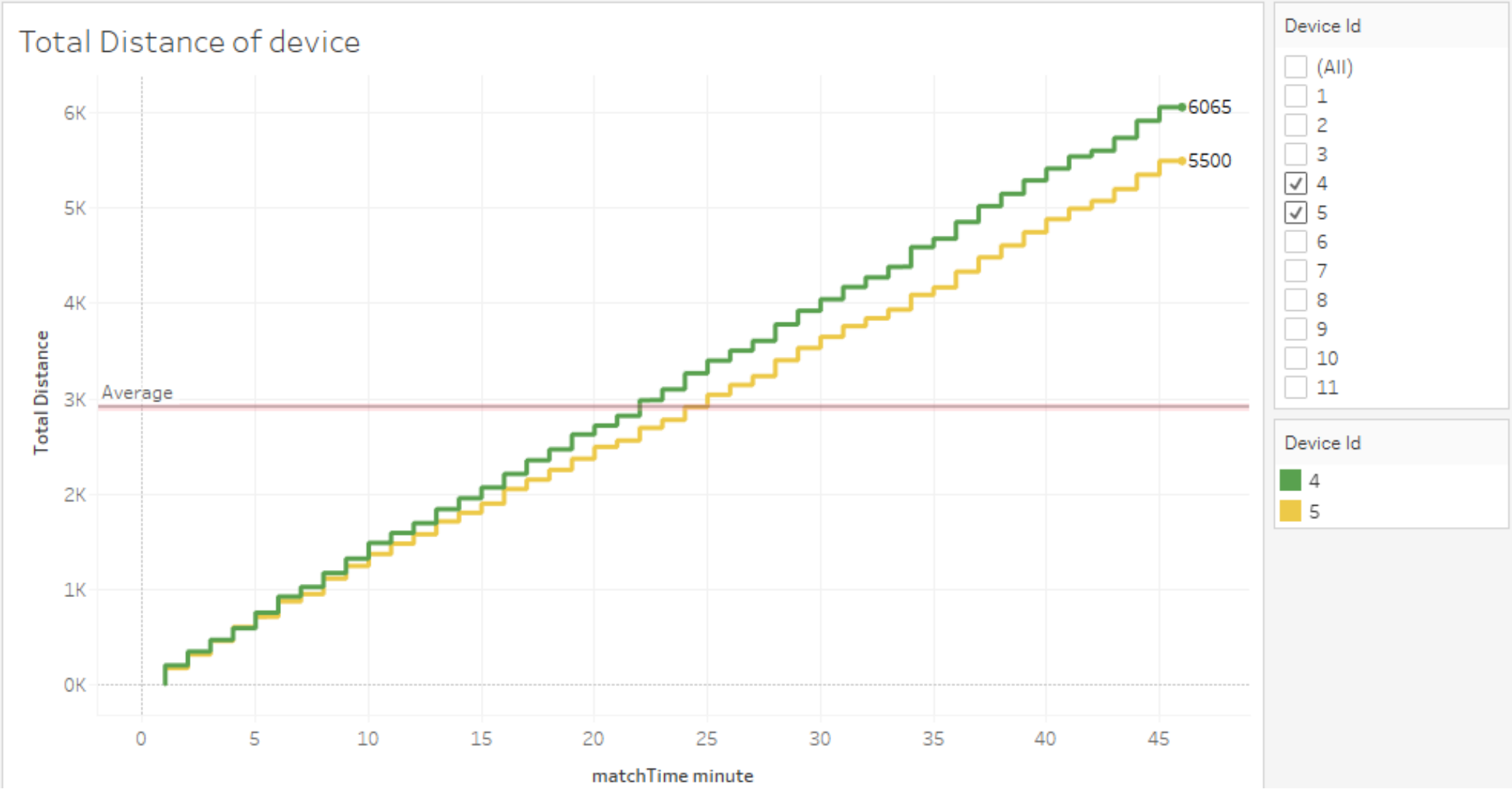
Visualize


Distance per time periods
Distance
matchTime minute
0.5
0.4
0.3
0.2
0.1
0.0
0
5
10
15
20
25
30
35
40
45
0.4721
0.4594
0.4582
0.4260
0.4249
0.3869
0.3656
0.3403
0.3598
0.3507
0.3612
0.3550
0.3512
0.3259
0.3030
0.3222
0.3164
0.3514
0.3458
0.4888
0.3577
0.3514
0.3507
0.3252
0.3244
0.3402
0.3326
0.3547
0.3488
0.3474
0.3214
0.3201
0.1319
0.0000
Device Id
(All)
1
2
3
4
5
6
7
8
9
10
11
Device Id
4

Visualize


Max of distance of Device
Device Id
3
2
1
7
5
8
6
11
9
10
4
0.3174
0.3187
0.3284
0.3609
0.3770
0.4106
0.4406
0.4595
0.4639
0.4706
0.4721
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
0.50
Max. Distance
matchTime minute
1
6
Device Id
(All)
1
2
3
4
5
6
7
8
9
10
11

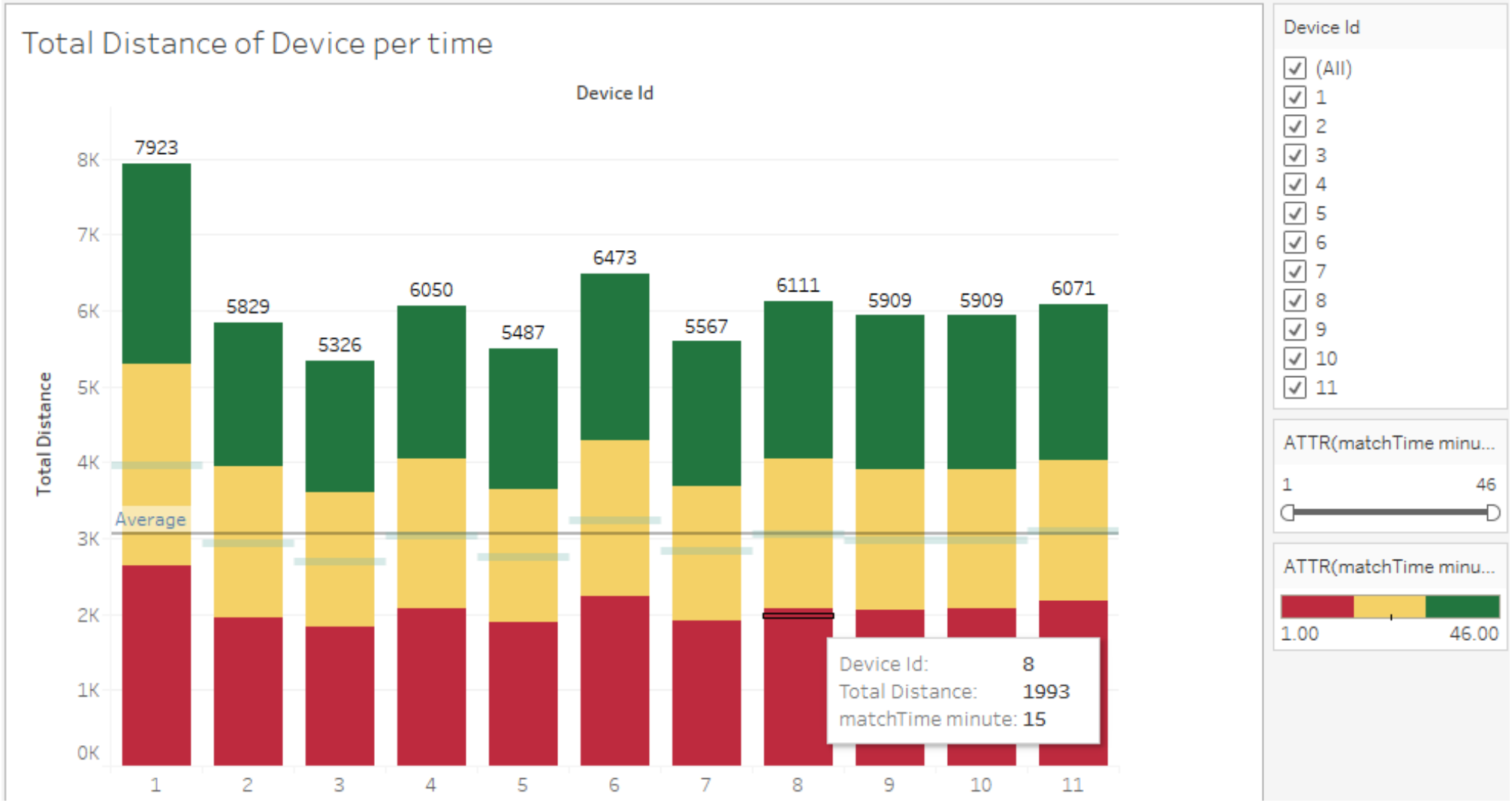
Visualize


Total Distance of device
6K
5K
4K
3K
2K
1K
0K
Total Distance
Average
6065
5500
0
5
10
15
20
25
30
35
40
45
matchTime minute
Device Id
(All)
1
2
3
4
5
6
7
8
9
10
11
Device Id
4
5

Visualize


Device Id
Total Distance
8K
7K
6K
5K
4K
3K
2K
1K
0K
Average
1
2
3
4
5
6
7
8
9
10
11
Device Id
(All)
1
2
3
4
5
6
7
8
9
10
11

Visualize


Total Distance of Device per time
Device Id
Total Distance
8K
7K
6K
5K
4K
3K
2K
1K
0K
Average
7923
5829
5326
6050
5487
6473
5567
6111
5909
5909
6071
1
2
3
4
5
6
7
8
9
10
11
Device Id: 8
Total Distance: 1993
matchTime minute: 15
Device Id
(All)
1
2
3
4
5
6
7
8
9
10
11
ATTR(matchTime minu...
1
46
ATTR(matchTime minu...
1.00
46.00

Calculate

1. Real time
2. Distance
3. Total distance

- Velocity
- Zone
- Turning points
- Energy
- Pass
- Combination
- Dashboards

....

See you in next workshops

PYDATA @ PYCON DE
OCTOBER 25-27, 2017

THANK YOU FOR YOUR ATTENTION


Q&A

Tea break 5 minutes