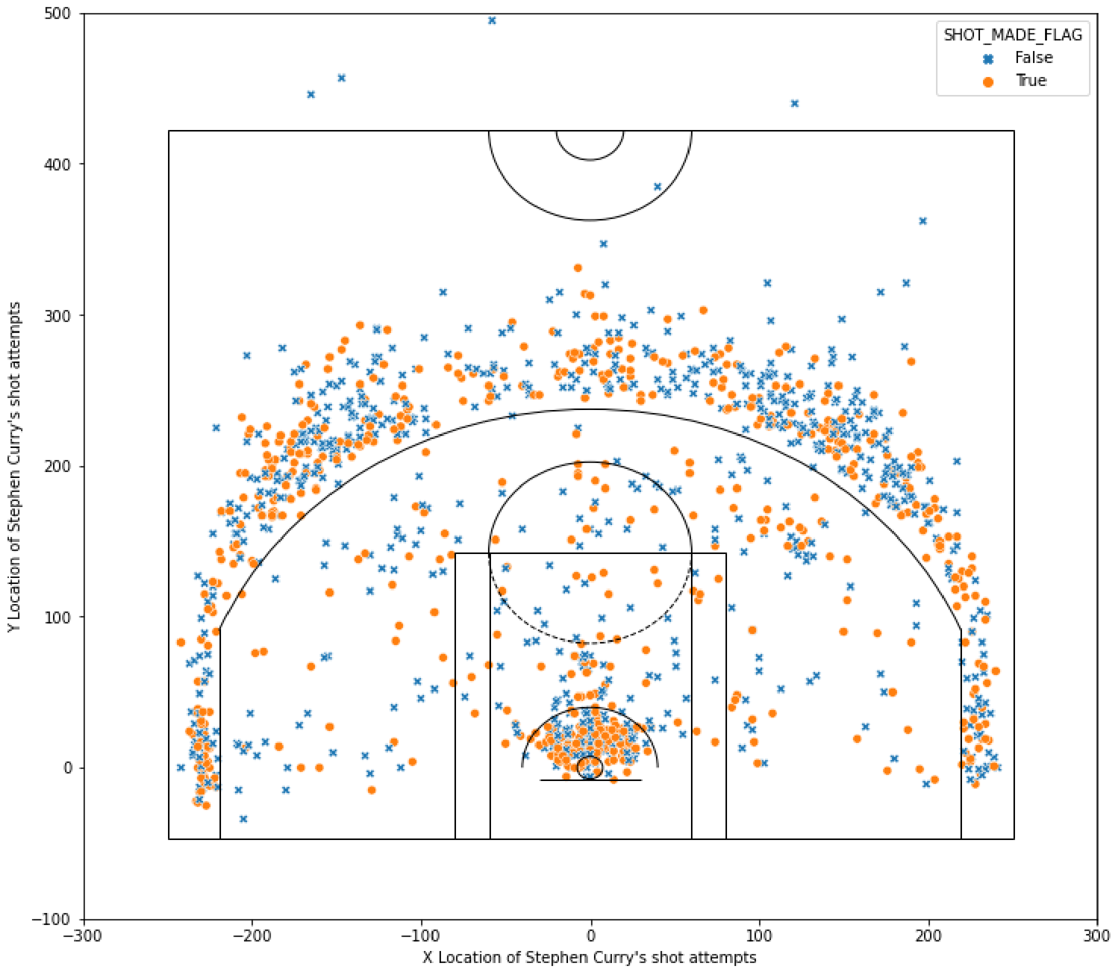
0.0.1 Question 2b: All Shots Scatter Plot + Court Outline

Again use seaborn to make a scatter plot of Stephen Curry's shots. Again, set the x-axis limits to span (-300, 300), the y-axis limits to span (-100, 500) color the points by whether the shot was made or missed. Set the missed shots to have an 'x' symbol and made shots to be a circular symbol. Call the `draw_court` function with `outer_lines` set to to be true. Save the `Axes` returned by the plot call in a variable called `ax`.

```
In [18]: plt.figure(figsize=(12, 11))
         markers = {0 : "X", 1 : "o"}
         ax = sns.scatterplot(data=curry_data, x=curry_data['LOC_X'], y=curry_data['LOC_Y'],
                              hue='SHOT_MADE_FLAG', style='SHOT_MADE_FLAG', markers=markers)
         ax.set(xlabel="X Location of Stephen Curry's shot attempts", ylabel="Y Location of Stephen Cur
                xlim=(-300, 300), ylim=(-100, 500))
         draw_court(outer_lines=True)
         plt.show()
```

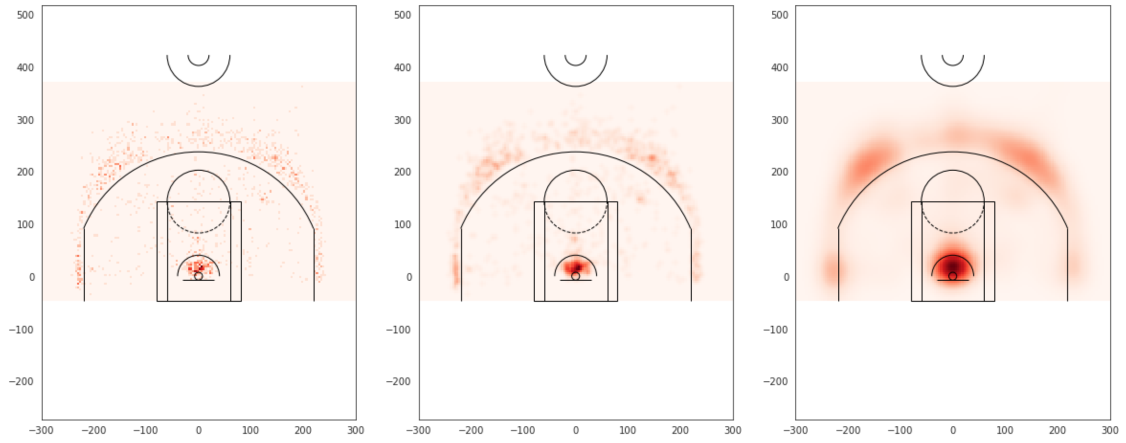


0.0.2 Question 2c: Analyzing the Visualization

In a few sentences, discuss what makes this an effective or ineffective visualization for understanding the types of shots that Stephen Curry likes to take and is good at taking, relative to other players in the league. Are there ways it can be improved?

This plot is effective due to the contrasting, bright colors and the difference in symbols. I would say the symbols are slightly too small, so an increase in their size would make it clearer.

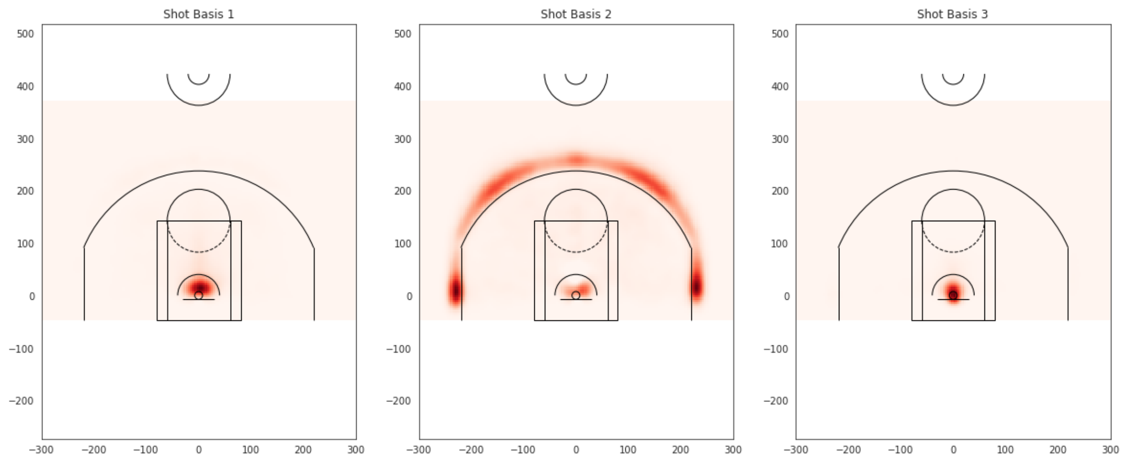
```
In [26]: fig, ax = plt.subplots(1, 3, figsize=(20,60))
         plot_shotchart(curry_binned_unsmoothed, xedges, yedges, ax[0])
         plot_shotchart(curry_binned_smoothed1, xedges, yedges, ax[1])
         plot_shotchart(curry_binned_smoothed5, xedges, yedges, ax[2])
         fig.show()
```



0.0.3 Question 4b: Visualizing Shot Types

Plot the first three basis images by calling `plot_vectorized_shot_chart` below on the columns of `W3`.

```
In [31]: def plot_vectorized_shot_chart(vec_counts, xedges, yedges, ax=None, use_log=False, cmap = 'Red
             """Plots 2d heatmap from vectorized heatmap counts

             Args:
                 hist_counts: vectorized output of numpy.histogram2d
                 xedges, yedges: bin edges in arrays
                 ax: figure axes [None]
                 use_log: will convert count x to log(x+1) to increase visibility [False]
                 cmap: Set the color map https://matplotlib.org/examples/color/colormaps_reference.html
             Returns:
                 ax: axes with plot
             """

             nx = xedges.size - 1
             ny = yedges.size - 1

             two_d_counts = np.reshape(vec_counts, [nx, ny])

             return(plot_shotchart(two_d_counts, xedges, yedges, ax=ax, use_log=use_log, cmap=cmap))

         fig, ax = plt.subplots(1, 3, figsize=(20,60))

         ## Write a for loop
         for i in range(3):
             plot_vectorized_shot_chart(W3[:, i], xedges, yedges, ax[i])
             ax[i].set_title('Shot Basis %i' % (i+1))
```



0.0.4 Question 4c: Reconstruction Error

Below we re-construct the shooting pattern for a single player. By “reconstructing” we mean use the approximation

$$\hat{X} = WH$$

obtained via NMF. Find $\hat{X}$ by multipling W and H. In python the `@` symbol is used for matrix multiplication.

```
In [32]: X3_hat = W3 @ H3
```

0.0.5 Question 4d: Choice of Colormap

Why does it make sense to use a *sequential* palette for the original and reconstructed shot charts and a *diverging* palette for the residual? *Hint:* Read the introduction to colormaps here.

A diverging palette shows us whether the residual was positive or negative, which is very useful to identify more nuanced patterns in our reconstructed shot charts.

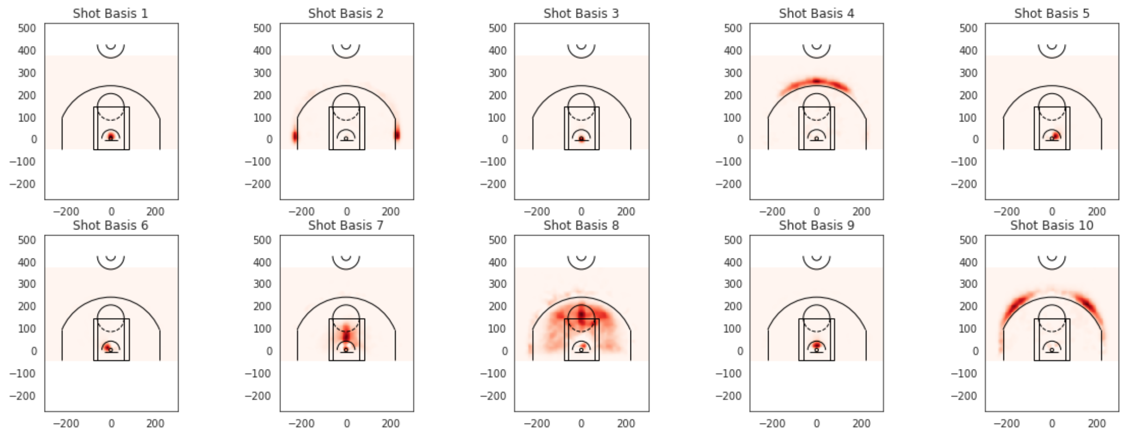
0.0.6 Question 4e: More Detailed Modeling

Re-run the analysis, this time for 10 basis vectors instead of 3. Again plot the bases using `plot_vectorized_shotchart` on the columns of `W10`.

Hint: Study the following code

```
fig, ax = plt.subplots(2, 5, figsize=(20, 7))
ax = ax.flatten() # turn ax into a flat array
ax[0].set_title('hello')
ax[9].set_title('there')
fig.show()
```

```
In [34]: decomp2 = non_negative_matrix_decomp(10, X)
         W10 = decomp2[0]
         H10 = decomp2[1]

         fig, ax = plt.subplots(2, 5, figsize=(20, 7))

         ## Write a for loop
         for i in range(10):
             plot_vectorized_shot_chart(W10[:, i], xedges, yedges, ax[i//5, i % 5])
             ax[i//5, i % 5].set_title('Shot Basis %i' % (i+1))
```

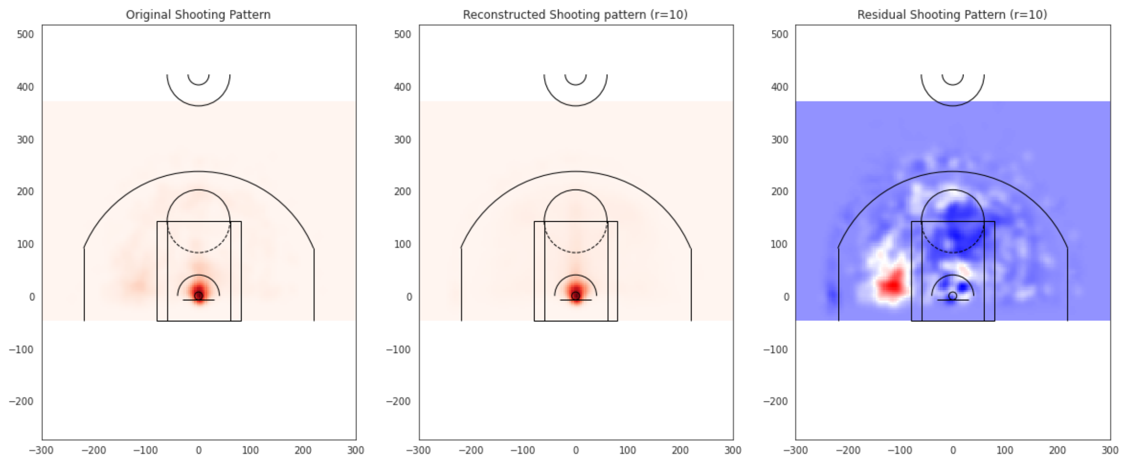


If you did things correctly, you should be really impressed! We've identified potentially interesting patterns of shooting styles without actually specifying anything about the way basketball is played or where the relevant lines are on the court. The resulting images are based only on the actual behavior of the players. Even more impressive is that we're capturing similarity in regions that are far apart on the court. One reason we can do this is that a basketball court is symmetric along the length of the court (i.e. symmetric about x=0). However, people tend to be left or right hand dominant, which might affect their preferences. Look carefuly at the shot basis plots above: is there any evidence of *asymmetry* in player shooting behavior? Refer to specific basis images in your answer.

At first glance it seems that every single basis image is symmetrical, but Shot Basis 8 shows that players tend to shoot more from the left side of the court behind the three-point line than from the right side. This may have to do with the fact that right-handed people outnumber left-handed people, so it makes sense to say the same for basketball players. This could also explain the spot in Shot Basis 8 that shows players tend to shoot more from the right side of the net when they are close to the net.

Repeat part 5b, and again plot original, reconstructed and residual shot chats for LaMarcus Aldridge.

```
In [35]: X10_hat = W10 @ H10

         fig, ax = plt.subplots(1, 3, figsize=(20,60))

         # Find the player_id of LaMarcus Aldridge
         player_id = allshots[allshots.PLAYER_NAME == 'LaMarcus Aldridge'].index[0]

         ## find index in X corresponding to that player
         to_plot_idx = np.where(pids == player_id)[0][0]

         ## Call plot_vectorized_shot_chart
         original_shotchart = plot_vectorized_shot_chart(X[:, to_plot_idx], xedges, yedges, ax[0])
         reconstructed_shotchart = plot_vectorized_shot_chart(X10_hat[:, to_plot_idx], xedges, yedges, a
         residual_chart = plot_vectorized_shot_chart((X - X10_hat)[:, to_plot_idx], xedges, yedges, ax[

         ax[0].set_title('Original Shooting Pattern')
         ax[1].set_title('Reconstructed Shooting pattern (r=10)')
         ax[2].set_title('Residual Shooting Pattern (r=10)');
```



0.0.7 Question 4f: Comparing Players

With `H10` matrix, it is possible to compare any pair of players. For all players pairwise, i and j, compare using euclidean distance between their coefficients:

$$\text{player-distance}(i,j) = \|H_i - H_j\|_2 = \left(\sum_{k=1}^{10}(H_{ki} - H_{kj})^2\right)^{1/2}$$

Create a heatmap for comparing pair-wise player distance matrix. Find the two pairs of players with smallest distances. Also, find two pairs of players with largest distances.

```
In [70]: pairwise = []
         for i in range(allplayers.size):
             for j in range(allplayers.size):
                 eudist = 0
                 for k in range(10):
                     eudist += ((H10[i, k] - H10[j, k]) ** 2)
                 pairwise.append([(eudist ** 0.5)])
         pairwise = np.array(pairwise)
         plt.imshow(pairwise, cmap='hot', interpolation='nearest')
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipykernel_244/490415730.py in <module>
      4         eudist = 0
      5         for k in range(10):
----> 6             eudist += ((H10[i, k] - H10[j, k]) ** 2)
      7         pairwise.append([(eudist ** 0.5)])
      8 pairwise = np.array(pairwise)

IndexError: index 10 is out of bounds for axis 0 with size 10
```

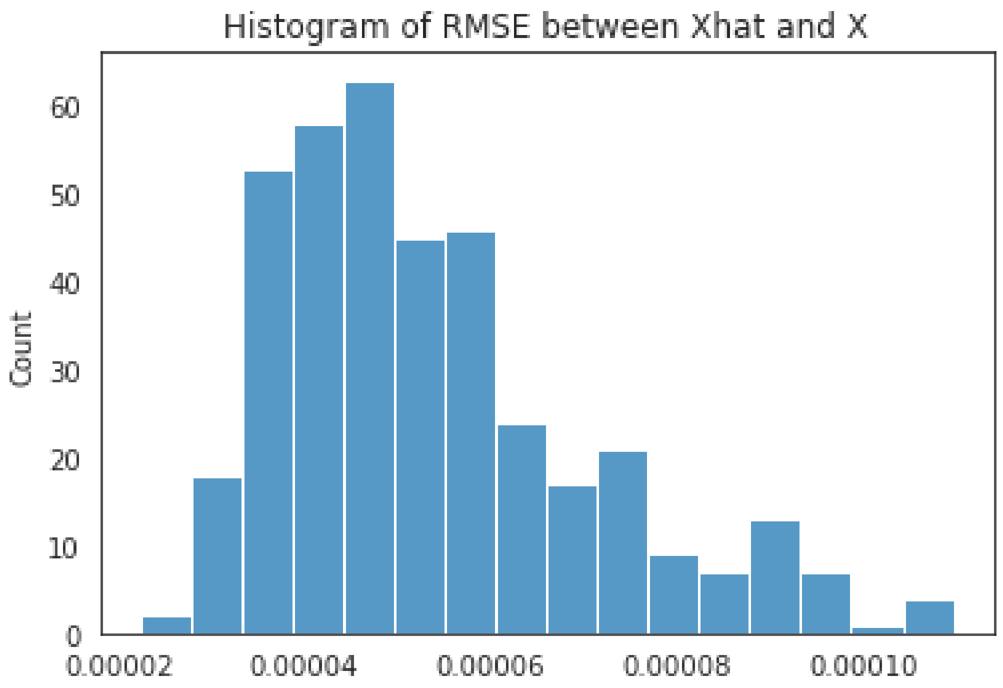
Type your answer here, replacing this text.

0.0.8 Question 4g: Residuals

The residual betwene `Xhat` and `X` gives a sense of how well a player is decribed by NMF computed matrices `W` and `H`. Calculate RMSE for each player, and plot the histogram. Comment on this distribution and players with smallest and largest RMSEs (use 10 components).

```
In [75]: from sklearn.metrics import mean_squared_error as mse
         from math import sqrt
         l = []
         residuals = X10_hat - X
         for i in residuals.T:
             rmse = sqrt(np.square(i).mean())
             l.append(rmse)
         sns.histplot(l).set(title='Histogram of RMSE between Xhat and X')
```

```
Out[75]: [Text(0.5, 1.0, 'Histogram of RMSE between Xhat and X')]
```



The peak RMSE is between 0.00004 and 0.00006, suggesting that Xhat is extremely close to X, almost a perfect fit. The player

0.0.9 Question 4h: Proposing improvements

One of the main purposes of exploratory data analysis is to generate new ideas, directions, and hypothesis for future analyses and experiments. Take two players of your choice and compare their shooting patterns with various visualizations.

State any insights and defend your conclusions with visual and/or numerical comparisons.

```
In [ ]:
```

Type your answer here, replacing this text.