Density Estimation: Lecture Notes

06-12-2023

This file will be my first attempt to take notes via an `.Rmd` file. Today's morning lecture for SURE 2023 is about density estimation.

Building Intuition about Density Estimation

Bold idea: **mass of objects can be measured**

BUT many comparisons are not "like with like" because volumes vary

Density = mass/volume to the rescue!

Probability's bold idea: **probability of events can be measured**

How do we measure probability?

For continuous variables: probability density function

Rules:

- P(event) is between 0,1 (inclusive)
- Area under PDF integrates to 1 (like fixing the volume in density, directly comparable)

The Idea on Estimation

The notion of inference:

- Samples are snapshots into population but we care about population-level questions

So, we must make assumptions and estimations...

- Must assume the event space follows some distribution
- Must assume the specific distribution that generates the data
- Must *estimate* the parameters for specific distribution

  This estimation comes from statistical techniques

The Data

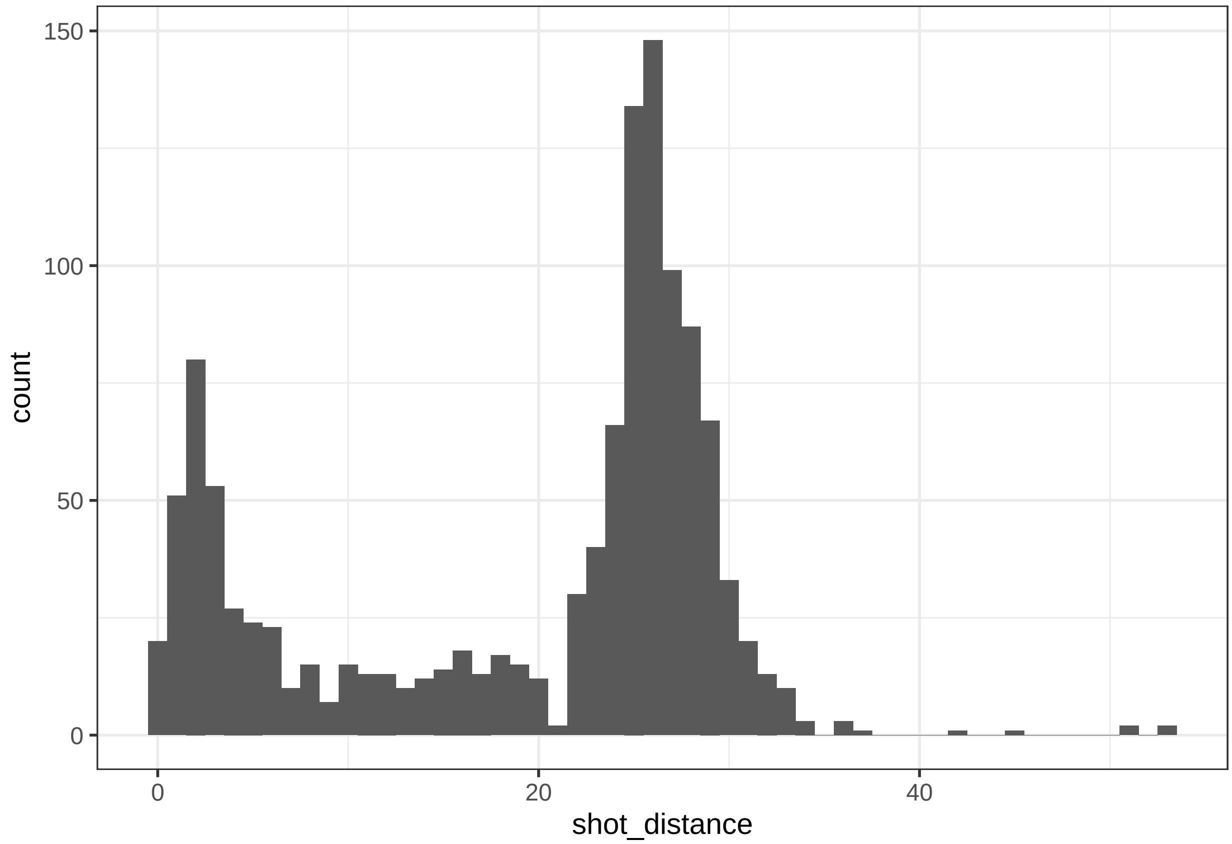
The data for today is about Steph Curry's shots.

```
curry_shots <-  read_csv("https://shorturl.at/xFI18")
head(curry_shots)
```

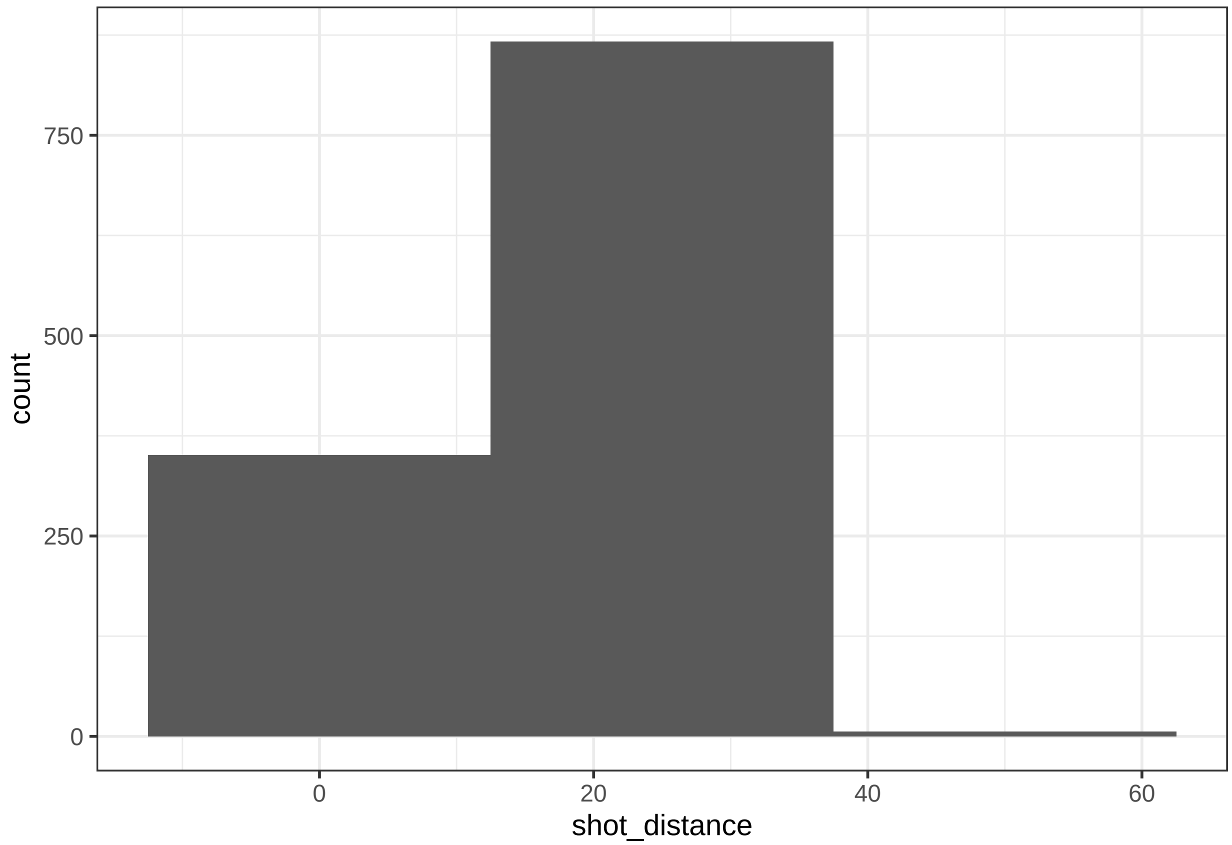
```
## # A tibble: 6 x 8
##   shot_x shot_y shot_distance is_shot_made period fg_type    shot_zone shot_type
##    <dbl>  <dbl>         <dbl> <lgl>         <dbl> <chr>      <chr>     <chr>
## 1   -109    260            28 FALSE             1 3PT Field~ Above th~ Pullup J~
## 2     48    257            26 FALSE             1 3PT Field~ Above th~ Running ~
## 3   -165    189            25 TRUE              1 3PT Field~ Above th~ Jump Shot
## 4    -13     12             1 FALSE             1 2PT Field~ Restrict~ Driving ~
## 5    -15     22             2 FALSE             1 2PT Field~ Restrict~ Layup Sh~
## 6     18     16             2 FALSE             1 2PT Field~ Restrict~ Driving ~
```

Histograms

```
curry_shots %>%
ggplot(aes(x = shot_distance)) +
geom_histogram(binwidth = 1) +
theme_bw()
```




```
curry_shots %>%
ggplot(aes(x = shot_distance)) +
geom_histogram(binwidth = 25) +
theme_bw()
```

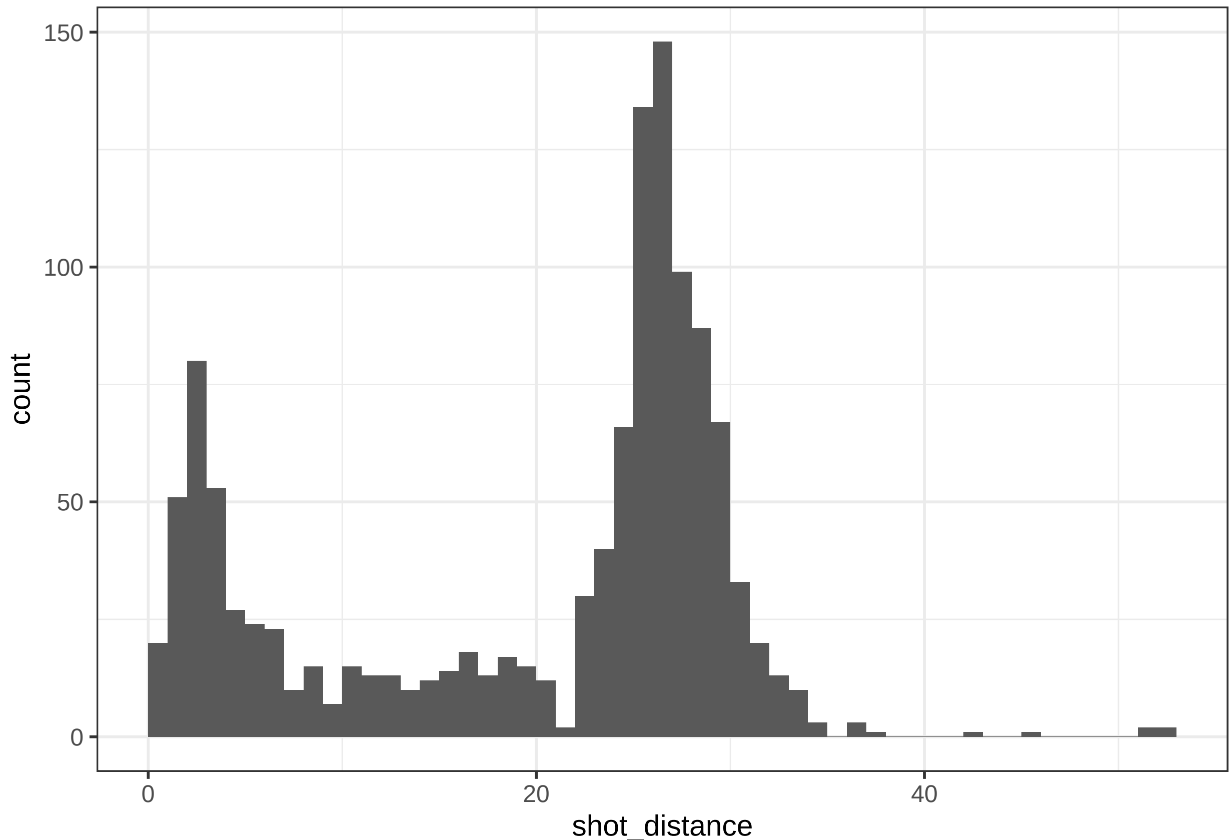


Subtle Point about Histogram Code

By default:

- left-closed, right-open intervals
- centered on the integers (e.g. starting at 0.5 and going to 1.5)

```
curry_shots %>%
ggplot(aes(x = shot_distance)) +
geom_histogram(binwidth = 1, center = 0.5,
closed = "left") +
theme_bw()
```



NOTE: Freedman-Diaconis Rule for bin width

Kernel Density Estimation

GOAL: estimate the PDF f(x) for all possible values (assuming it is continuous/smooth)

Formula:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i)$$

Kernel is a "continuous weighting of f" that satisfies:

(1) non-negative

(2) symmetric about 0

(3) goes to 0 as x goes to infinity or negative infinity

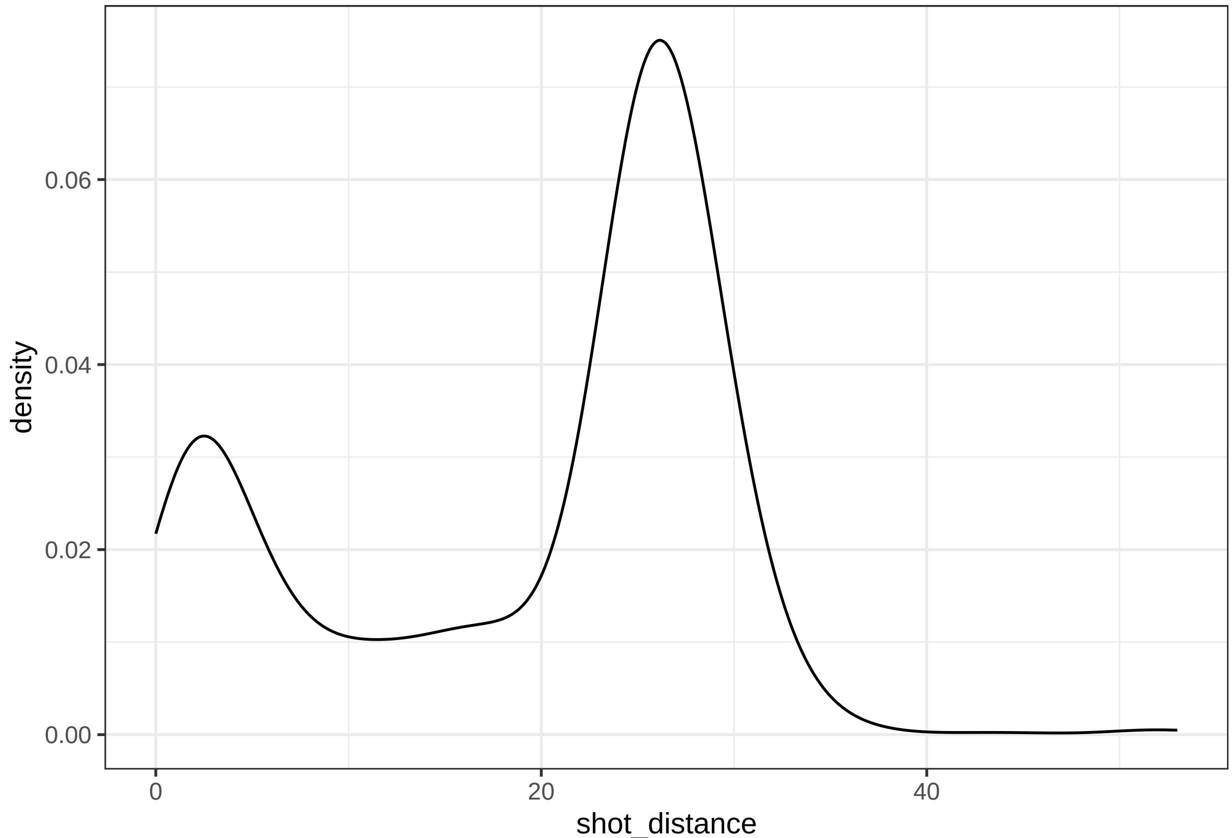
Other Notes:

- n = sample size, x = new point estimate f(x) [does not have to be in the dataset!]
- h = bandwidth analogous to histogram bin width, ensures $\hat{f}(x)$ integrates to 1
- x_i = ith observation in the data set
- $K_h(x - x_i)$ is the **Kernel** function, creates **weight** given distance of ith observation from new point
- As bandwidth (h) increases, weights are more evenly spread out (and as h decreases, more concentrated around x)

- typically use **Gaussian/Normal** kernel [can look up online]
- $K_h(x - x_i)$ is large when x_i is close to x

Histogram is basically a step-function version of Kernel Density Estimation

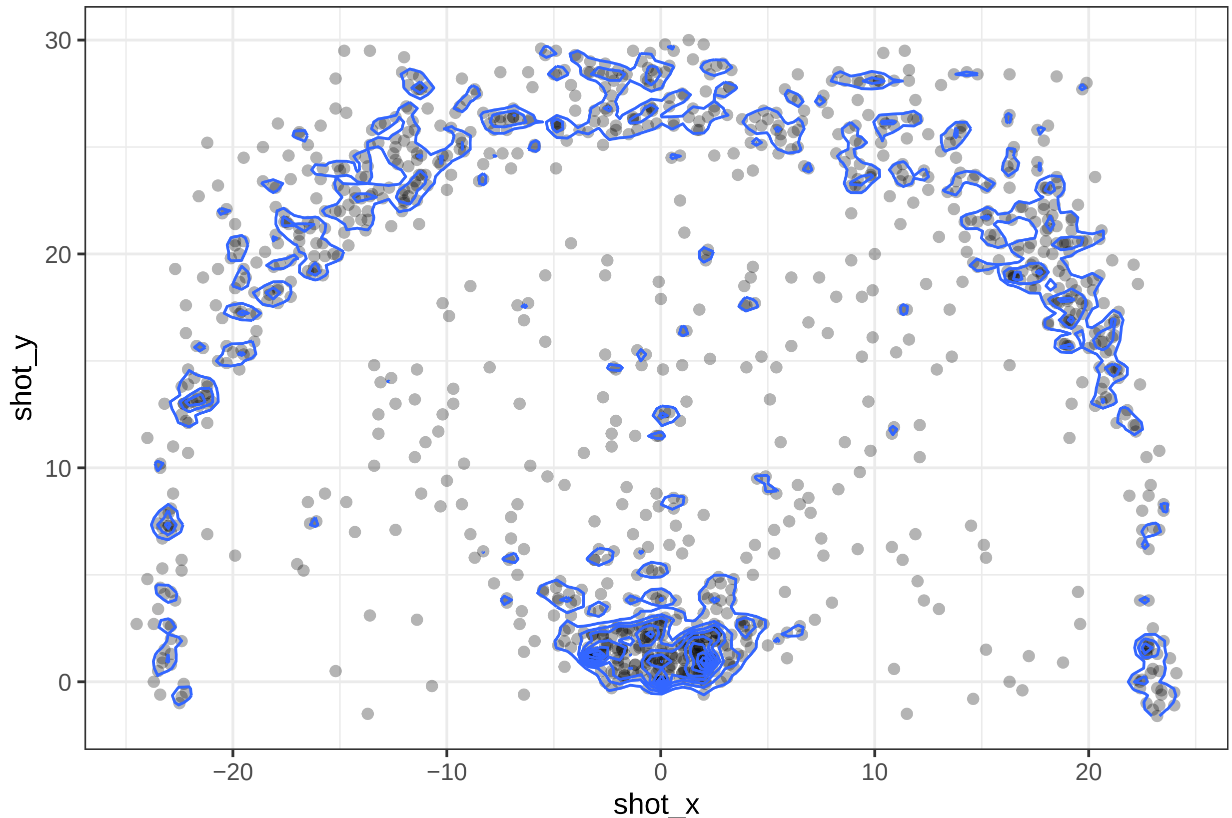
```
curry_shots %>%
ggplot(aes(x = shot_distance)) +
geom_density() +
theme_bw()
```



What about for 2D? (two continuous variables)

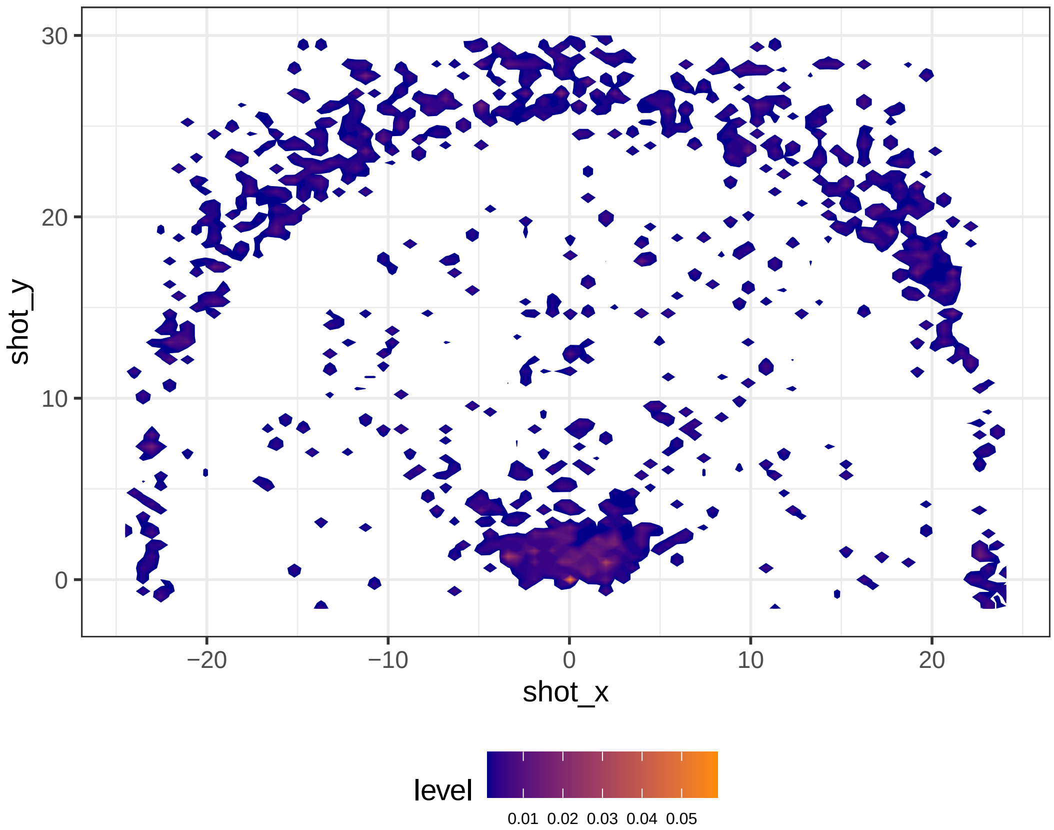
Contours

```
curry_shots %>% # Modify the shot coordinates
  mutate(shot_x = -shot_x / 10, shot_y = shot_y / 10) %>%
  filter(shot_y <= 30) %>%
  ggplot(aes(x = shot_x, y = shot_y)) +
  geom_point(alpha = 0.3) +
   geom_density2d(adjust = 0.1) + #adds blue level sets for contours
  coord_fixed() +
  theme_bw()
```



Heatmaps

```
curry_shots %>%  mutate(shot_x = -shot_x / 10, shot_y = shot_y / 10) %>%  filter(shot_y <= 30) %>%
ggplot(aes(x = shot_x, y = shot_y)) +
stat_density2d(h = 0.5, bins = 60, aes(fill = after_stat(level)),                 geom = "polygon") +
scale_fill_gradient(low = "darkblue", high = "darkorange") +
theme_bw() +
theme(legend.position = "bottom", legend.text = element_text(size = 6))+
  coord_fixed()
```


30
20
10
0
−20
−10
0
10
20
shot_x
shot_y
level
0.01 0.02 0.03 0.04 0.05