R Tutorial-07

T. Scofield

You may click here to access the .qmd file.

In this issue, we demonstrate the Central Limit Theorem at work. As in class, I will take my population to be 2019 professional baseball players (as reflected in the data set), and the variable X to be `Salary`. This data needed to be drawn from an external source, https://www.lock5stat.com/datapage3e.html.

```
mlb19 = read.csv("https://www.lock5stat.com/datasets3e/BaseballSalaries2019.csv")
head(mlb19)
```

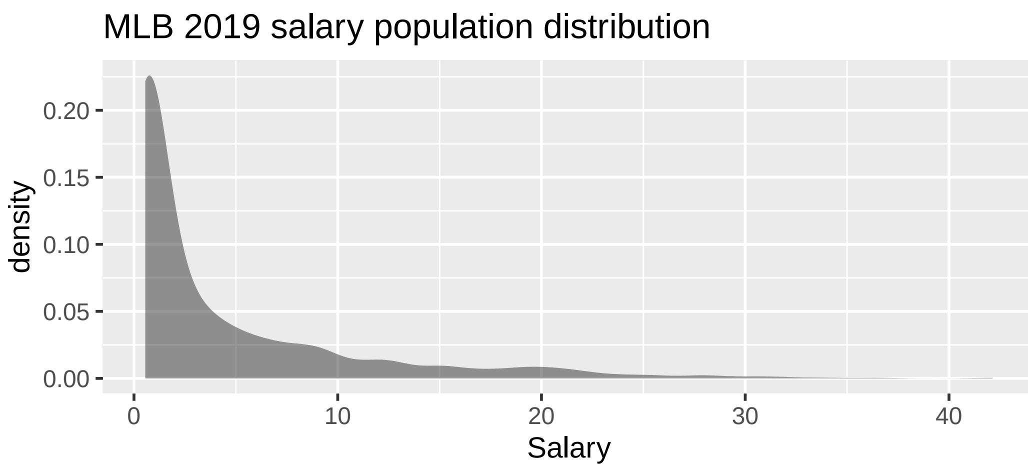
```
               Name Salary Team POS
1      Max Scherzer 42.143 WSH   SP
2 Stephen Strasburg 36.429 WSH   SP
3        Mike Trout 34.083 LAA   CF
4      Zack Greinke 32.422 ARI   SP
5       David Price 31.000 BOS   SP
6   Clayton Kershaw 31.000 LAD   SP
```

```
nrow(mlb19)
```

```
[1] 877
```

The salaries in this population are decidedly non-normal:

```
gf_density(~Salary, data=mlb19) |>
  gf_labs(title="MLB 2019 salary population distribution")
```




```
mean(~Salary, data=mlb19)    # this is mu
```

```
[1] 4.509924
```

```
sd(~Salary, data=mlb19)    # this is sigma
```

```
[1] 6.334217
```

The Central Limit Theorem addresses the behavior of sums and means calculated from iid random samples.

iid samples of size 5

I will use `resample()` (iid is like sampling with replacement) to draw one sample of size 5, and use it to calculate the mean of those 5 players.

```
oneSampleOfFive = resample(mlb19, size=5)
oneSampleOfFive   # shows the players involved
```

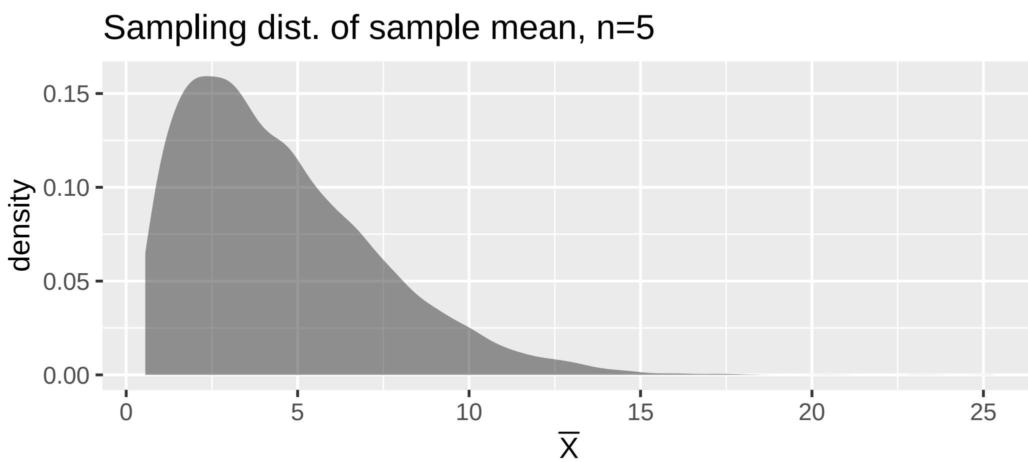
```
                Name Salary Team POS orig.id
188   Eugenio Suarez  7.286 CIN   3B     188
680      Julio Urias  0.565 LAD   SP     680
565      Colin Moran  0.580 PIT   3B     565
418 Kyle Barraclough  1.725 WSH   RP     418
768    Dakota Hudson  0.559 STL   SP     768
```

```
mean(~Salary, data=oneSampleOfFive)
```

```
[1] 2.143
```

The point with the CLT is that the **sampling distribution** of means such as these—those calculated from iid samples of size 5 using this population—is increasingly normal, but less spread-out, as n grows. We only get a sense of this sampling distribution by repeating the process of the previous code block many times. Below I do so 10000 times.

```
manyMeans5 = replicate(10000, mean(~Salary, data=resample(mlb19, size=5)))
gf_density(~manyMeans5) |>
  gf_labs(title="Sampling dist. of sample mean, n=5", x=TeX(r"( $\bar{X}$ )"))
```




```
mean(manyMeans5)
```

```
[1] 4.506374
```

```
sd(manyMeans5)
```

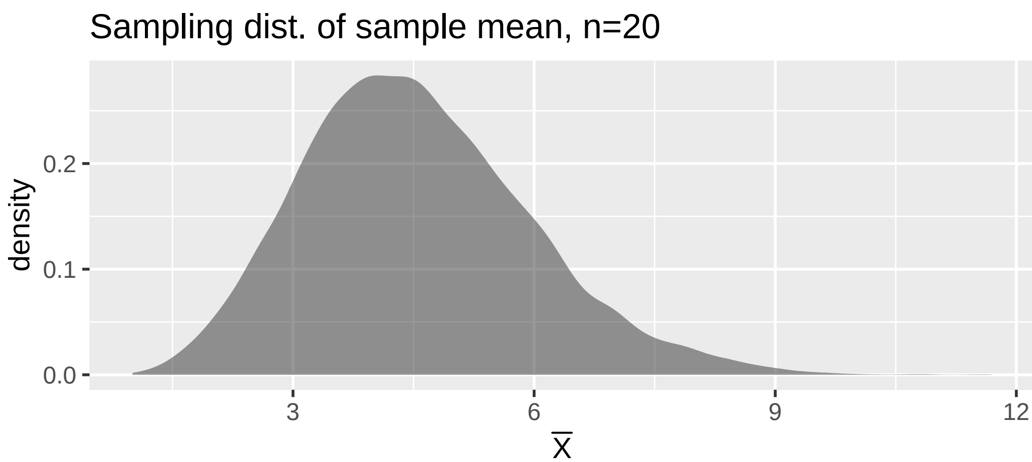
```
[1] 2.853796
```

While this (approximate) sampling distribution is still non-normal, it isn't as starkly skewed as the population distribution (displayed above). It is still centered on the same value μ as the population, but its **standard error** (the standard deviation of $\bar{X}$-values) is less than σ: specifically, it is approximate $\sigma/\sqrt{5}$.

iid samples of size 20

Upping the sample size, now to $n = 20$ (4 times larger) should cut the standard error by half, while producing an approximate sampling distribution that looks even more *normal*:

```
manyMeans20 = replicate(10000, mean(~Salary, data=resample(mlb19, size=20)))
gf_density(~manyMeans20) |>
  gf_labs(title="Sampling dist. of sample mean, n=20", x=TeX(r"( $\bar{X}$ )"))
```




```
mean(~manyMeans20)    # approximate mu = 4.51
```

```
[1] 4.542446
```

```
sd(~manyMeans20)      # approximately 6.334/sqrt(20)
```

```
[1] 1.419002
```

As it becomes more normal, it should have about the same shape as the Norm(μ,$\sigma/\sqrt{n}$) distribution. At $n = 20$, there is still some discrepancy, but you see it is happening when overlaying the one with the other:

```
gf_density(~manyMeans20) |> gf_dist("norm", mean=4.51, sd=6.334/sqrt(20), color="red")
```