Preface

In Statistics or Data Science, we deal with data that is affected by chance in some way:

- data comes from a random sample;
- data is affected by measurement error;
- data measures some outcome that is random in nature.

Being able to quantify the uncertainty introduced by randomness is one of the most important jobs of a data analyst. Statistical inference offers a framework, as well as several practical tools, for doing this. The first step is to learn how to mathematically describe random variables.

Warm-up

- Revisit random variables and their properties starting with their application to games of chance.
- Describe some of the events surrounding the financial crisis of 2007-2008 using probability theory.

- The crisis is partly caused by underestimating the risk of certain securities sold by financial institutions. Specifically, the risks of mortgage-backed securities (MBS) and collateralized debt obligations (CDO) were grossly underestimated.

- These assets were sold at prices that assumed most homeowners would make their monthly payments, and the probability of this not occurring was calculated as being low.

- Several factors resulted in many more defaults than were expected, which led to a price crash of these securities.

- As a result, banks lost so much money that they needed government bailouts to avoid closing down completely.

Random variables

- Random variables are numeric outcomes resulting from random processes. For example, define X to be 1 if a bead is blue and 0 otherwise.

R demonstration:

```
beads <- rep( c("red", "blue"), times = c(2,3))
X <- ifelse(sample(beads, 1) == "blue", 1, 0)
```

-

Random variables

- Random variables are numeric outcomes resulting from random processes. For example, define X to be 1 if a bead is blue and 0 otherwise.

R demonstration:

```
beads <- rep( c("red", "blue"), times = c(2,3))
X <- ifelse(sample(beads, 1) == "blue", 1, 0)
```

- Here X is a random variable: every time we select a new bead the outcome changes randomly. See below:

```
ifelse(sample(beads, 1) == "blue", 1, 0)
#> [1] 1
ifelse(sample(beads, 1) == "blue", 1, 0)
#> [1] 0
ifelse(sample(beads, 1) == "blue", 1, 0)
#> [1] 0
```

- Distribution of X:

$$\mathbb{P}(X = 1) = 0.6, \quad \mathbb{P}(X = 0) = 0.4$$

```
x <- rep(0,100)
for (i in 1:100) {
  x[i] <- ifelse(sample(beads, 1) == "blue", 1, 0)
}
sum(x)
#> [1] 62
mean(x)
#> [1] 0.62
```

Sampling models

Many data generation procedures, those that produce the data we study, can be modeled as draws from an urn.

- Polling: draw 0s (Republicans) and 1s (Democrats) from an urn containing the 0 and 1 code for all likely voters.
- Epidemiological studies, experimental research, etc.

- Casino games offer many examples of real-world situations in which sampling models are used to answer specific questions. We start with such examples.

Example

Suppose a small casino hires you to consult on whether they should set up roulette wheels. To keep it simple, we will assume that 1000 people will play and that the only game you can play on the roulette wheel is to bet on red or black.

- The casino wants you to predict how much money they will make or lose.
- They want a range of values and, in particular, they want to know what's the chance of losing money.
- If this probability is too high, they will pass on installing roulette wheels.

Define a random variable $S = S_{1000}$: representing the casino's total winnings. Let us construct the urn.

- A roulette wheel has 18 red pockets, 18 black pockets and 2 green ones. So playing a color in one game of roulette is equivalent to drawing from this urn:

```
color <- rep(c("Black", "Red", "Green"), c(18, 18, 2))
```

-

Define a random variable $S = S_{1000}$: representing the casino's total winnings. Let us construct the urn.

- A roulette wheel has 18 red pockets, 18 black pockets and 2 green ones. So playing a color in one game of roulette is equivalent to drawing from this urn:

```
color <- rep(c("Black", "Red", "Green"), c(18, 18, 2))
```

- The 1000 outcomes from 1000 people playing are independent draws from this urn. If red comes up, the gambler wins and the casino loses a dollar, so we draw a -$1. Otherwise, the casino wins a dollar and we draw a $1. We construct the random variable S as follows:

```
color <- rep(c("Black", "Red", "Green"), c(18, 18, 2))
n <- 1000
X <- sample(ifelse(color=="Red",-1,1), n, replace=TRUE)
```

- Because we know the proportions of 1s and -1s, we can generate the draws with one line of code, without defining color:

```
X <- sample(c(-1,1), n, replace = TRUE, prob=c(18/38, 20/38))
```

-

```
color <- rep(c("Black", "Red", "Green"), c(18, 18, 2))
n <- 1000
X <- sample(ifelse(color=="Red",-1,1), n, replace=TRUE)
```

- Because we know the proportions of 1s and -1s, we can generate the draws with one line of code, without defining color:

```
X <- sample(c(-1,1), n, replace = TRUE, prob=c(18/38, 20/38))
```

- We call this a sampling model since we are modeling the random behavior of roulette with the sampling of draws from an urn. The total winnings S is simply the sum of these 1000 independent draws:

```
X <- sample(c(-1,1), n, replace = TRUE, prob=c(18/38, 20/38))
S <- sum(X)
```

Probability distribution of a random variable (r.v.)

If you run the code above, you see that S changes every time.

The probability distribution of an r.v. tells us the probability of the observed value falling at any given interval.

- For example, the probability of losing money is the probability that S is in the interval $(-\infty, 0)$.
- Define the *cumulative distribution function* (CDF) $F(a) = \mathbb{P}(S \leq a)$. This F is called the random variable's *distribution function* (d.f.).
- For a *standard normal distribution* — $\mathcal{N}(0,1)$, the CDF is

$$F(a) = (2\pi)^{-1/2} \int_{-\infty}^{a} e^{-t^2/2} \mathrm{d}t.$$

We can estimate the distribution function of S by using a *Monte Carlo simulation* to generate many realizations of this random variable.

- We run the experiment of having 1000 people play roulette, over and over, for $B = 10000$ times:

```
n <- 1000
B <- 10000
roulette_winnings <- function(n){
X <- sample(c(-1,1), n, replace = TRUE, prob=c(9/19, 10/19))
sum(X)
}
S <- replicate(B, roulette_winnings(n))
```

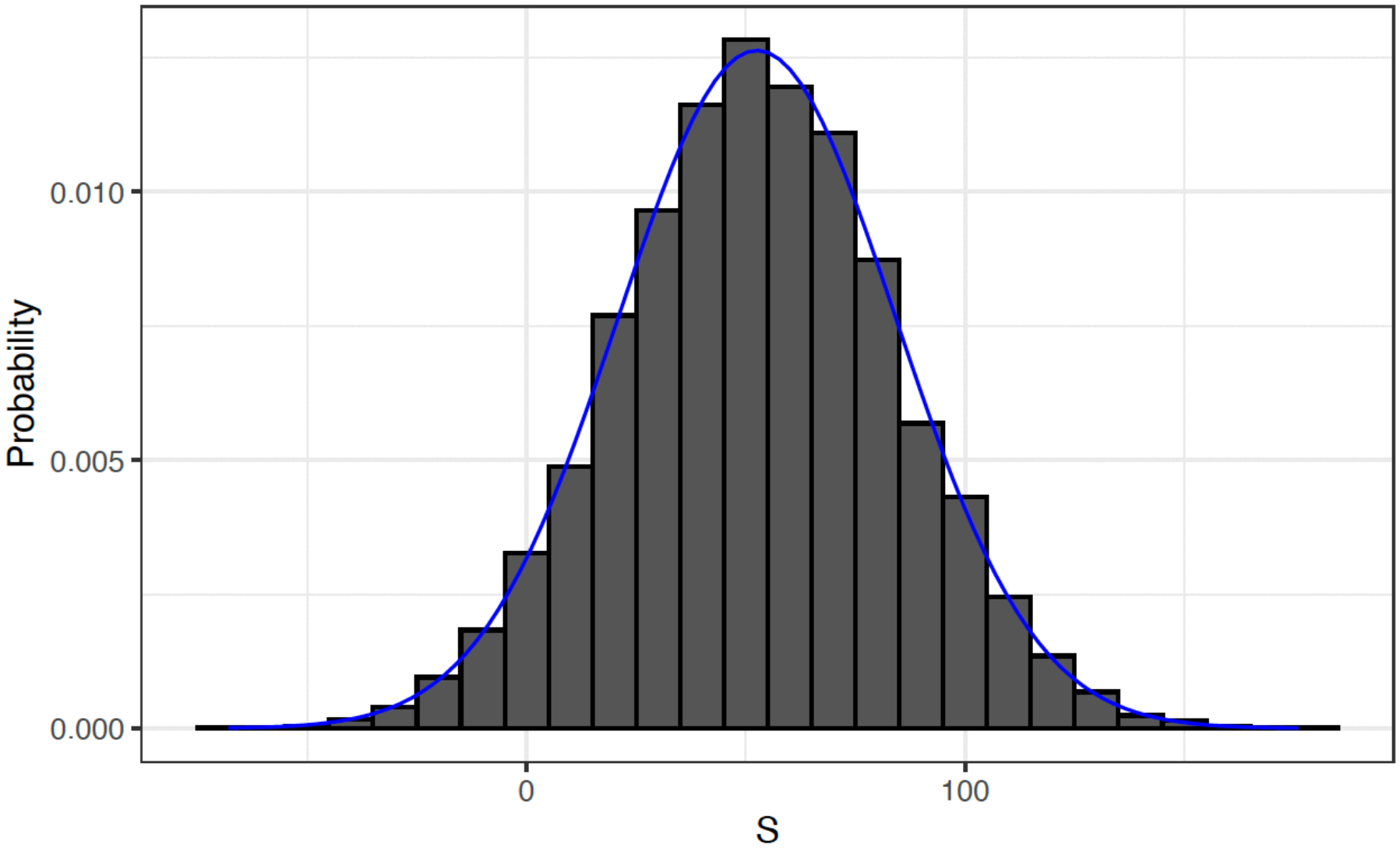
- We can ask: how often did we get sums less than or equal to a?

```
mean(S <= a)
```

- This is a fairly good approximation of $F(a)$. We can easily answer the casino's question: how likely is it that the casino will lose money? It is quite low.

```
mean(S < 0)
#> [1] 0.0456
```

- We can also visualize the distribution of S by creating a *histogram* showing the probability $F(b) - F(a)$ for several intervals $(a, b]$:


Probability
0.010
0.005
0.000
0
100
S

- The distribution appears to be approximately normal.

- If, in fact, the distribution is normal, then all we need to determine the distribution is mean and standard deviation.

- Since we have the original values from which the distribution is created, we can compute these with `mean(S)` (sample mean) and `sd(S)` (sample standard deviation/ standard error).

- The blue curve added to the histogram above is a normal density with this mean and standard deviation.

Statistical theory

- $(S+n)/2$ follows a *binomial distribution*.
- In R, we can use the function `dbinom` and `pbinom` to compute the probabilities of binomial distribution exactly.

- For example, note that

$$\mathbb{P}(S<0)=\mathbb{P}((S+n)/2<n/2).$$

```
n <- 1000
pbinom(n/2, size = n, prob = 10/19)
#> [1] 0.0511
```

Since this is a *discrete* probability function, to get $\mathbb{P}(S<0)$ rather than $\mathbb{P}(S\leq 0)$, we compute

```
pbinom(n/2-1, size = n, prob = 10/19)
#> [1] 0.0448
```

Notation for random variables

In statistics, upper case letters are often used to denote random variables, and lower case letters are used for observed values.

- For example, we see random events defined as $X \leq x$.
- Let X represent the number on a die roll, and x an actual value we see—1,2,3,4,5 or 6. The prob. of $X = x$ is $1/6$.
- In the context of probability, X is not an observed quantity; instead, it's a random quantity that we will see in the future. We can talk about what we expect it to be, what values are probable, but not what it is. Once we have data, we do see a realization of X. So data scientists talk of what could have been after we see what actually happened.

Expected value (mean) and standard error

Now we briefly go over the mathematical theory that lets us approximate the prob. distributions for the sum of draws.

- Expected value of the random variable X: $\mathbb{E}(X)$.
- A random variable will vary around its expected value—if you take the average of many, many draws, the average of the draws will approximate the expected value, getting closer and closer the more draws you take.

- In the urn used to model betting on red in roulette, we have 20 \$1s and 18 -\$1s. The expected value is

$$\mathbb{E}(X) = \$(20 + -18)/38 \approx 5 \text{ cents.}$$

Counterintuitive? X varies around 0.05, but takes values 1 and -1. One way to make sense of the expected value is by realizing that if we play the game over and over, the casino wins, on average, 5 cents per game. A Monte Carlo simulation confirms this:

```
B <- 10^6
x <- sample(c(-1,1), B, replace = TRUE, prob=c(9/19, 10/19))
mean(x)
#> [1] 0.0517
```

- If the urn has two outcomes, a and b, with probabilities p and $1-p$, the average/mean is $\mathbb{E}(X) = ap + b(1-p)$.

- Property: the *expected value of the sum of the draws* is: number of draws $\times$ average of the numbers in the urn.

- If 1000 people play roulette, the casino expects to win, on average, about 1000 $\times$ \$0.05 = \$50. But this is only an expected value. How different can one observation be from the expected value? What is the range of possibilities? If negative numbers are too likely, the casino will not install roulette wheels.

- The *standard error* (SE) gives us an idea of the size the variation around the mean. We use $\text{se}(X)$ to denote the standard error/deviation of a random variable.

- If *draws are independent*, the standard error of the sum is $\sqrt{\text{number of draws}} \times$ stand deviation of the numbers in the urn.

- If an urn contains two values a and b with proportions p and $1-p$, the standard deviation is $\sqrt{p(1-p)}\,|b-a|$.

- $\mathrm{se}(X) = \sqrt{\mathrm{var}(X)}$

- $\mathrm{var}(X) = \mathbb{E}X^2 - (\mathbb{E}X)^2$

- $\mathbb{E}X^2 = a^2p + b^2(1-p), \quad \mathbb{E}X = ap + b(1-p)$

- $$\begin{aligned}\mathrm{var}(X) &= a^2p + b^2(1-p) - (ap + b(1-p))^2 \\ &= (a-b)^2 p(1-p)\,.\end{aligned}$$

- Back to the roulette example, the s.d. of the values inside the urn is: $2 * \sqrt{(10/19) * (9/19)}$ or

```
2 * sqrt(90)/19
#> [1] 0.999
```

- The sum of 1000 people playing has s.e. of about $32:

```
n <- 1000
sqrt(n) * 2 * sqrt(90)/19
#> [1] 31.6
```

- When 1000 people bet on red, the casino is expected to win $50 with a standard error of $32. Seems like a safe bet. Recall the question: how likely is it to lose money?

- Earlier we have calculated, using the exact binomial distribution, the probability of $S < 0 \approx 4.5\,\%$:

```
pbinom(n/2-1, size = n, prob = 10/19)
#> [1] 0.0448
```

- Alternative method: CLT (central limit theorem) will help. CLT can be generally applied to sums of random variables in a way that the binomial distribution cannot.

Population SD versus sample SD

Consider a height data from the `dslabs` package.

```
install.packages("dslabs"). # first time users
library(dslabs)             # load the package
x <- heights$height
m <- mean(x)
s <- sqrt(mean((x-m)^2))
```

- Mathematical notation:

$$m = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - m)^2}.$$

- The `sd` function returns a slightly different result:

```
s-sd(x)
#> [1] -0.00194
```

- Let $X_1, \ldots, X_N$ be a random sample from a population. The `sd` function computes the square root of an unbiased variance estimator, with the formula

$$\hat{s} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})^2}, \quad \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i.$$

- For large samples, these two are practically equivalent since $\sqrt{(N-1)/N} \approx 1$.

Central limit theorem

CLT tell us: when the number of draws—sample size—is large, the probability distribution of the sum of independent draws is approximately normal.

Previously we ran

```
n <- 1000
B <- 10000
roulette_winnings <- function(n){
X <- sample(c(-1,1), n, replace = TRUE, prob=c(9/19, 10/19))
sum(X)
}
S <- replicate(B, roulette_winnings(n))
```

- CLT tells us, the sum S is approximated by a normal distribution, with mean and standard deviation

```
n * (20-18)/38
#> [1] 52.6
sqrt(n) * 2 * sqrt(90)/19
#> [1] 31.6
```

- These theoretical values match those obtained with the Monte Carlo simulation

```
mean(S)
#> [1] 52.2
sd(S)
#> [1] 31.7
```

- By CLT, we can skip the MC and instead compute the probability of the casino losing money using this approximation:

```
mu <- n * (20-18)/38
se <- sqrt(n) * 2 * sqrt(90)/19
pnorm(0, mu, se)
#> [1] 0.0478
```

- Recall the Monte Carlo result:

```
mean(S < 0)
#> [1] 0.0458
```

When is CLT useful

- CLT works when the number of draws is large. How large is large?
- Rule-of-thumb: in many circumstances, as few as 30 draws is enough to make the CLT useful. Sometimes, as few as 10 is enough.
- When the probability of success is very small, we need much larger sample sizes.

- Counterexample: In the lottery, the chances of winning are less than 1 in a million. Thousands of people buy so the number of draws is very large. Yet the number of winners, the sum of the draws, ranges between 0 and 4. This sum is certainly not well approximated by a normal distribution, so the CLT does not apply, even with very large sample size. This is true when the probability of a success is very low, in which case the Poisson distribution is more appropriate.

Statistical properties of averages/means

- The expected value of the sum of random variables is the sum of each random variable's expected value:

$$\mathbb{E}(X_1 + X_2 + \cdots + X_n) = \mathbb{E}X_1 + \mathbb{E}X_2 + \cdots + \mathbb{E}X_n.$$

- The expected value of a non-random constant times a random variable is the non-random constant times the expected value of a random variable:

$$\mathbb{E}(aX) = a \times \mathbb{E}(X).$$

- The variance of the sum of independent random variables is the sum of the variance of each random variable:

$$\mathrm{var}(X_1 + \cdots + X_n) = \mathrm{var}(X_1) + \cdots + \mathrm{var}(X_n).$$

- The s.e. of a non-random constant a times a random variable is $|a|$ times the random variable's s.e.:

$$\mathrm{se}(aX) = |a| \times \mathrm{se}(X).$$

- Let $X_1, \ldots, X_n$ be independent draws from the same urn with mean $\mu = \mathbb{E}(X_i)$ and standard deviation $\sigma = \sqrt{\text{var}(X_i)}$. Then

$$\mathbb{E}((X_1 + \cdots + X_n)/n) = n\mu/n = \mu$$

and

$$\begin{aligned} &\text{se}((X_1 + \cdots + X_n)/n) \\ &= \text{se}(X_1 + \cdots + X_n)/n \\ &= \sqrt{\text{se}(X_1) + \cdots + \text{se}(X_n)}/n = \sigma/\sqrt{n}. \end{aligned}$$

- If X is a normally distributed random variable—$\mathcal{N}(\mu, \sigma^2)$—with mean μ and variance σ^2, then for non-random constants a and b, $aX + b \sim \mathcal{N}(a\mu + b, (a\sigma)^2)$.

Law of large numbers (LLN)

- The standard error of the average becomes smaller and smaller ($\rightarrow 0$) as n grows larger ($n \rightarrow \infty$). When n is very large, then the standard error is practically 0 and the average of the draws converges to the average of the urn. This is known in statistical textbooks as the law of large numbers.
- Misinterpreting law of averages:
  - Imagine toss a coin 5 times and see a head each time. One might argue that the next toss is probably a tail because of LLN: on average we should see 50% heads and 50% tails.
  - Or saying that red "is due" on the roulette wheel after seeing black come up five times in a row.

These events are independent! The chance of a coin landing heads is 50% regardless of the previous 5.

Case study: The Big Shot

Interest rate explained by chance model

- Suppose you have been running a small bank for years. Historically, in a given year, only 2% of your customers default.
- If you loan money to everybody without interest, you will end up losing money due to this 2%.
- Although you know 2% of your clients will probably default, you don't know which ones.
- By charging everybody just a bit extra in interest, you can make up the losses incurred due to that 2%, and also cover operating costs.
- If you set the interest rates too high, your clients will go to another bank.

☆ We use all above facts and some probability theory to decide what interest rate you should charge.

- Suppose your bank will give out 1000 loans for $180,000 this year. After adding up all costs, suppose your bank loses $200,000 per foreclosure. For simplicity, we assume this includes all operational costs. A sampling model for this scenario can be coded as:

```
n <- 1000
loss_per_foreclosure <- -200000
p <- 0.02
defaults <- sample( c(0,1), n, prob=c(1-p, p), replace = TRUE)
sum(defaults * loss_per_foreclosure)
#> [1] -3600000
```

- The total loss defined by the final sum is a random variable. Every time you run the above code, you get a different answer.

- We can construct a Monte Carlo simulation to get an idea of the distribution of this random variable.

```
B <- 10000
losses <- replicate(B, {
defaults <- sample( c(0,1), n, prob=c(1-p, p), replace = TRUE)
sum(defaults * loss_per_foreclosure)
})
```

-

- We can construct a Monte Carlo simulation to get an idea of the distribution of this random variable.

```
B <- 10000
losses <- replicate(B, {
defaults <- sample( c(0,1), n, prob=c(1-p, p), replace = TRUE)
sum(defaults * loss_per_foreclosure)
})
```

- We can also use CLT to approximate this distribution. The mean and standard error are estimated by

```
n*(p*loss_per_foreclosure + (1-p)*0)
#> [1] -4e+06
sqrt(n)*abs(loss_per_foreclosure)*sqrt(p*(1-p))
#> [1] 885438
```

- We can now set an interest rate to guarantee that, on average, we break even. To do so, we add a quantity $x > 0$ to each loan, which in this case are represented by draws, so that the expected value is 0.

- If we define l to be the loss per foreclosure, we need:

$$lp + x(1-p) = 0.$$

  Solve this gives

```
-loss_per_foreclosure*p/(1-p)
#> [1] 4082
```

  This corresponds to an interest rate of $2.3\% - x/180000$.

- Problem: Although this interest rate guarantees that on average we break even, there is a 50% chance that we lose money. If our bank loses money, we have to close it down.

- Proposal: Pick an interest rate that makes it unlikely for this to happen. Remember that if the interest rate is too high, our clients will go to another bank so we must be willing to take some risks.
- If we want our chance of losing money to be 1 in 100, what does x quantity need to be? In other words, we want the sum S to satisfy

$$\mathbb{P}(S < 0) = 0.01.$$

- Proposal: Pick an interest rate that makes it unlikely for this to happen. Remember that if the interest rate is too high, our clients will go to another bank so we must be willing to take some risks.

- If we want our chance of losing money to be 1 in 100, what does x quantity need to be? In other words, we want the sum S to satisfy

$$\mathbb{P}(S < 0) = 0.01.$$

- Recall that S is approximately normal. The mean of S is

$$\mathbb{E}S = \{lp + x(1-p)\}n,$$

where n is the number of draws/loans. The standard error is

$$\mathrm{sd}(S) = |x - l|\sqrt{np(1-p)} = (x-l)\sqrt{np(1-p)}.$$

- If $\mathbb{P}(S < 0) = 0.01$, then

$$\mathbb{P}\left(\frac{S - \mathbb{E}S}{\mathrm{se}(S)} < \frac{-\mathbb{E}S}{\mathrm{se}(S)}\right) = 0.01.$$

- CLT tells us $\frac{S - \mathbb{E}S}{\text{se}(S)} \overset{\text{d}}{\approx} Z \sim \mathcal{N}(0,1)$.

- The problem is then to find x such that

$$\mathbb{P}\left(Z < \frac{-\{lp + x(1-p)\}n}{(x-l)\sqrt{np(1-p)}} \right) = 0.01,$$

  or equivalently,

$$\frac{-\{lp + x(1-p)\}n}{(x-l)\sqrt{np(1-p)}} = \text{0.01-quantile of } \mathcal{N}(0,1).$$

```
qnorm(0.01)
#> [1] -2.33
```

- $z = \text{qnorm}(0.01)$ gives us the value of z for which $\mathbb{P}(Z \leq z) = 0.01$.

- Solve the equation

$$\frac{-\{lp + x(1-p)\}n}{(x-l)\sqrt{np(1-p)}} = z$$

to obtain

$$x = -l\frac{np - z\sqrt{np(1-p)}}{n(1-p) + z\sqrt{np(1-p)}}.$$

- Solve the equation

$$\frac{-\{lp + x(1-p)\}n}{(x-l)\sqrt{np(1-p)}} = z$$

  to obtain

$$x = -l\frac{np - z\sqrt{np(1-p)}}{n(1-p) + z\sqrt{np(1-p)}}.$$

```
l <- loss_per_foreclosure
z <- qnorm(0.01)
x <- -l*(n*p - z*sqrt(n*p*(1-p)))/(n*(1-p) + z*sqrt(n*p*(1-p)))
x
#> [1] 6249
```

- Our interest rate now goes up to 3.5%, a very competitive one. By choosing this rate, we now have an expected profit per loan of:

```
loss_per_foreclosure*p + x*(1-p)
#> [1] 2124
```

The total expected profit is about

```
n*(loss_per_foreclosure*p + x*(1-p))
#> [1] 2124198
```

dollars!

- Let's run a Monte Carlo simulation to double check the theoretical approximations:

```
B <- 100000
profit <- replicate(B, {
draws <- sample( c(x, loss_per_foreclosure), n,
prob=c(1-p, p), replace = TRUE)
sum(draws)
})
mean(profit)
#> [1] 2118604
mean(profit<0)
#> [1] 0.013
```

The Big Short

- One of your employees points out: since the bank is making $2124 dollars per loan, the bank should give out more loans! Why just n?

- You explain that finding those n clients was hard. You need a group that is predictable and that keeps the chances of defaults low.

- He/she points out: even if the probability of default is higher, as long as our expected value is positive, you can minimize your chances of losses by increasing n and relying on the law of large numbers.

- He/she claims: even if the default rate is twice as high—4%—if we set the rate just a bit higher than

```
p <- 0.04
r <- (- loss_per_foreclosure*p/(1-p)) / 180000
r
#> [1] 0.0463
```

we will profit. At 5%, we are guaranteed a positive expected value of

```
r <- 0.05
x <- r*180000
loss_per_foreclosure*p + x * (1-p)
#> [1] 640
```

- We minimize the chances of losing money by increasing n, because

$$\mathbb{P}(S < 0) \approx \mathbb{P}\left(Z < -\frac{\mathbb{E}X}{\text{se}(X)}\right).$$

Let μ and σ be the mean and standard deviation of the urn—a single urn. Then $\mathbb{E}S = n\mu$ and $\text{se}(X) = \sqrt{n}\sigma$. With $z = \text{qnorm}(0.01)$, solve

$$-\frac{n\mu}{\sqrt{n}\sigma} = -\frac{\sqrt{n}\mu}{\sigma} = z.$$

- If we let

$$n \geq z^2\sigma^2/\mu^2$$

  then (if $\mu > 0$)

$$\mathbb{P}(S < 0) \approx \mathbb{P}\left(Z < -\frac{\mathbb{E}X}{\text{se}(X)}\right) = \mathbb{P}\left(Z < -\frac{\sqrt{n}\mu}{\sigma}\right) \leq \mathbb{P}(Z < z) = 0.01.$$

- Implication: as long as μ is positive, we can find an n that minimizes the probability of losing.

-

- If we let

$$n \geq z^2\sigma^2/\mu^2$$

  then (if $\mu > 0$)

$$\mathbb{P}(S < 0) \approx \mathbb{P}\left(Z < -\frac{\mathbb{E}X}{\text{se}(X)}\right) = \mathbb{P}\left(Z < -\frac{\sqrt{n}\mu}{\sigma}\right) \leq \mathbb{P}(Z < z) = 0.01.$$

- Implication: as long as μ is positive, we can find an n that minimizes the probability of losing.

- Question: with x fixed, what n do we need for the probability to be 0.01?

```
z <- qnorm(0.01)
n <- ceiling((z^2*(x-l)^2*p*(1-p))/(l*p + x*(1-p))^2)
n
#> [1] 22163
```

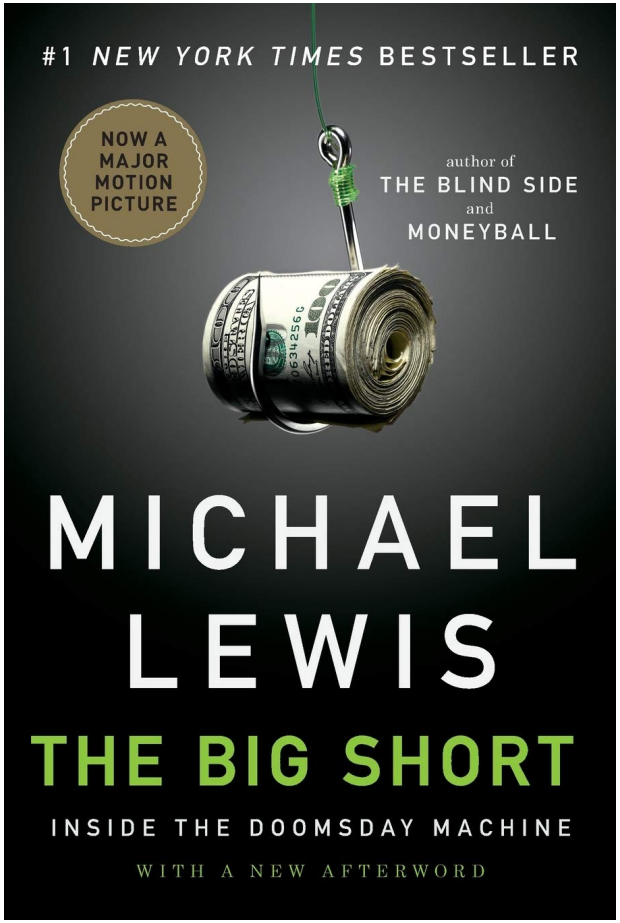
If we give out 22163 loans, the probability of losing is about 1% and we are expected to earn a total of

```
n*(loss_per_foreclosure*p + x * (1-p))
#> [1] 14184320
```

more than $14 million. We confirm this with a Monte Carlo simulation:

```
p <- 0.04
x <- 0.05*180000
profit <- replicate(B, {
draws <- sample( c(x, loss_per_foreclosure), n,
prob=c(1-p, p), replace = TRUE)
sum(draws)
})
mean(profit)
#> [1] 14185724
mean(profit < 0)
#> [1] 0.01066
```

- As a result, your colleague decides to leave your bank and start his own high-risk mortgage company. A few months later, your colleague's bank has gone bankrupt. A book is written and eventually a movie is made relating the mistake your friend, and many others, made. What happened?



- The above scheme was mainly based on the formula

$$\mathrm{se}((X_1 + \cdots + X_n)/n) = \sigma/\sqrt{n}.$$

  By making n large, we minimize the standard error of per-loan profit.

  For this to hold, X_i's must be independent draws: one person defaulting must be independent of others defaulting.

-

-

- The above scheme was mainly based on the formula

$$\mathrm{se}((X_1 + \cdots + X_n)/n) = \sigma/\sqrt{n}.$$

  By making n large, we minimize the standard error of per-loan profit.

  For this to hold, X_i's must be independent draws: one person defaulting must be independent of others defaulting.

- In the case of averaging the same event over and over—an extreme case of events that are not independent—we get a standard error that is $\sqrt{n}$ times bigger:

$$\mathrm{se}((X_1 + \cdots + X_1)/n) = \sigma \gg \sigma/\sqrt{n}.$$

- A more realistic simulation: assume there is a global event that affects everybody with high-risk mortgages and changes their probability. We will assume that with 50-50 chance, all the probabilities go up or down slightly to somewhere between 0.03 and 0.05. But it happens to everybody at once, not just one person. These draws are no longer independent.

```
p <- 0.04
x <- 0.05*180000
profit <- replicate(B, {
      new_p <- 0.04 + sample(seq(-0.01, 0.01, length = 100), 1)
      draws <- sample( c(x, loss_per_foreclosure), n,
      prob=c(1-new_p, new_p), replace = TRUE)
      sum(draws)
})
```

- The expected profit it still large:

```
mean(profit)
#> [1] 14082671 ($14 million)
```

```
p <- 0.04
x <- 0.05*180000
profit <- replicate(B, {
      new_p <- 0.04 + sample(seq(-0.01, 0.01, length = 100), 1)
      draws <- sample( c(x, loss_per_foreclosure), n,
      prob=c(1-new_p, new_p), replace = TRUE)
      sum(draws)
})
```

- The expected profit it still large:

```
mean(profit)
#> [1] 14082671 ($14 million)
```

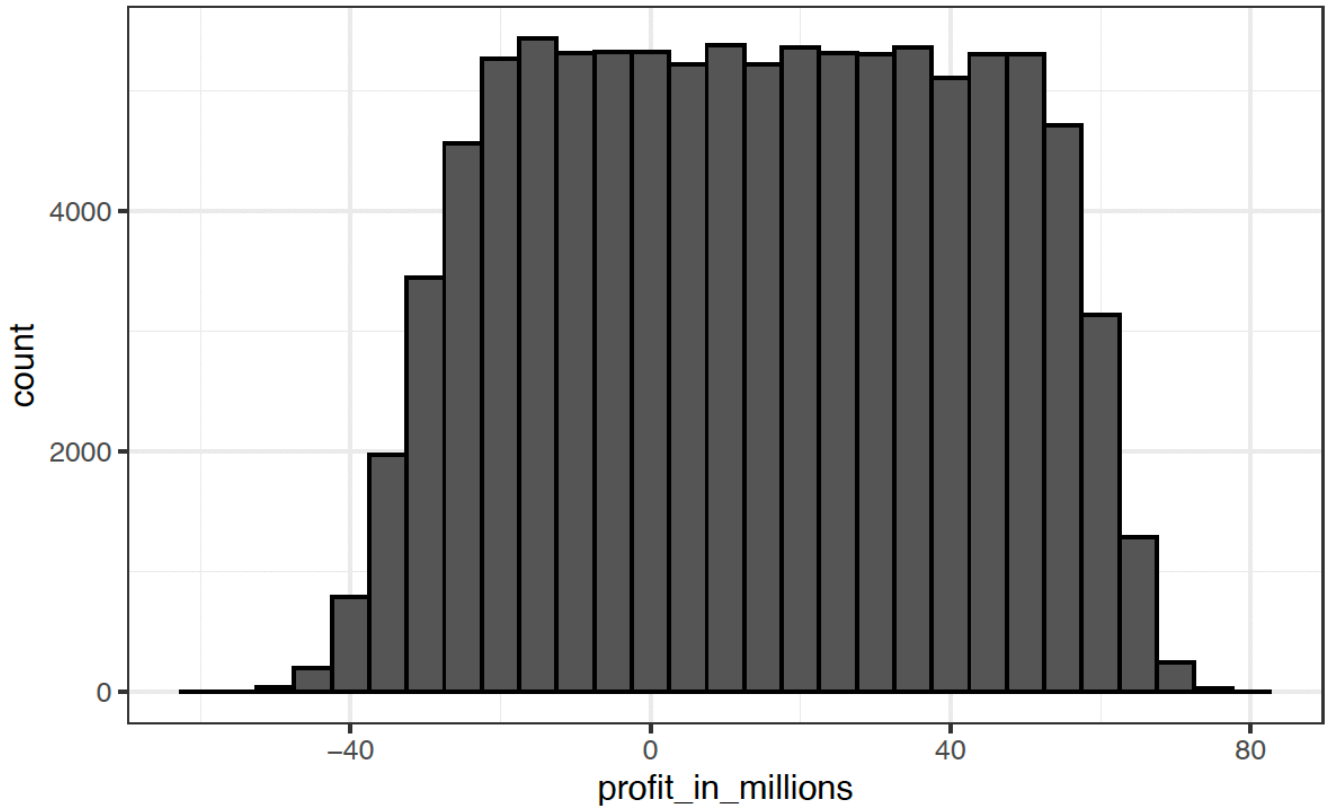
- The probability of the bank having negative earnings shoots up to:

```
mean(profit<0)
#> [1] 0.35
```

- Even scarier is that the probability of losing more than $10 million is:

```
mean(profit < -10000000)
#> [1] 0.244
```

- To understand how this happens, look at the distribution:



- ☆ The theory completely breaks down and the random variable—total earnings—has much more variability than expected. The financial meltdown of 2007 was due, among other things, to financial “experts” assuming independence when there was none.

```
install.packages(“ggplot2”) # first-time users
library(ggplot2)            # load the package
profit_in_millions <- data.frame(profit/10^6)
names(profit_in_millions) <- c("profit_in_millions")

ggplot(profit_in_millions, aes(x = profit_in_millions))
       + geom_histogram(color="black", binwidth = 5)
```