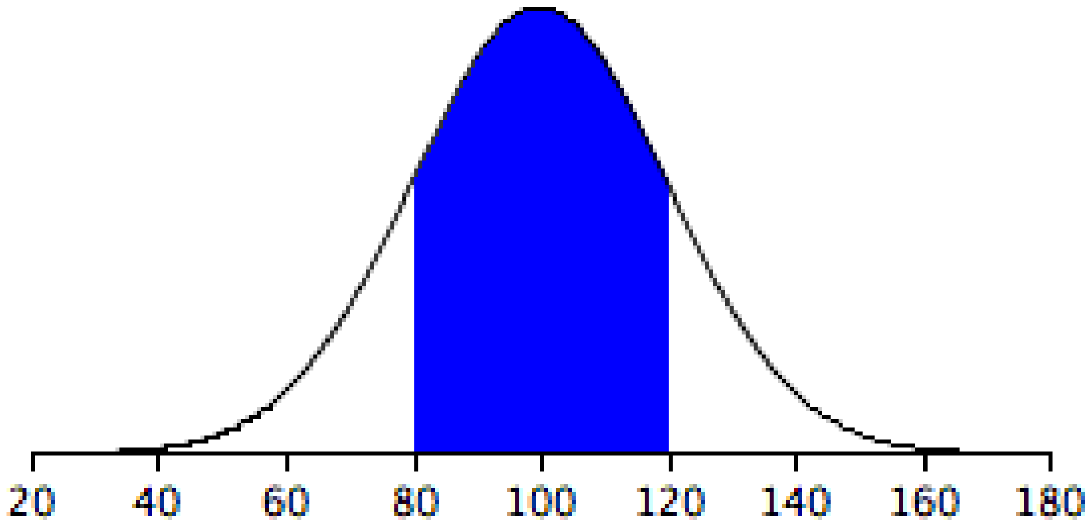
Data and sampling distributions



Overview

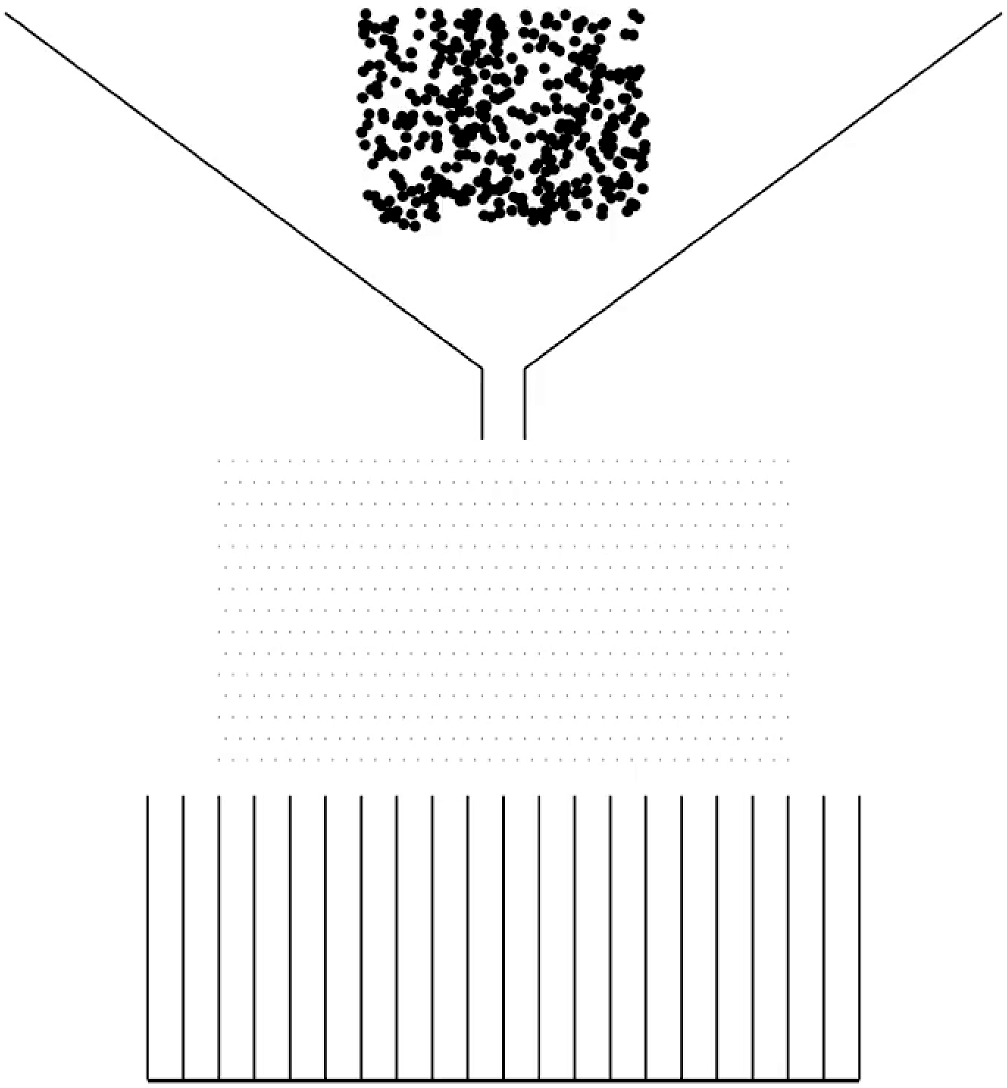
Very quick review

For loops

Probability functions

- Generating random numbers
- Probability density functions
- Cumulative distribution functions

Sampling distributions



Announcements

Change in my office hours: 3-4pm on Tuesdays and Thursdays

Homework 2 has been posted

- Due Sunday (9/18) at 11pm
- Start early on it!
  - You can do problems 1, and 2 after today's class.
- How was homework 1?

Where we are in the plan for the semester

			Analysis	R
1	Sep 2	Course overview, introduction to R, descriptive statistics		base R
2	Sep 7-9	Review of central statistical concepts and exploratory analysis using R	resampling methods	

We will be using some simulations to justify and validate methods we use throughout the semester

Where we are in the plan for the semester

How would describe the pace of the class so far?

Way too slow	1 respondent	1 %
Too slow	7 respondents	6 %
About right	91 respondents	78 %
Too fast	17 respondents	15 %
Way too fast		0 %

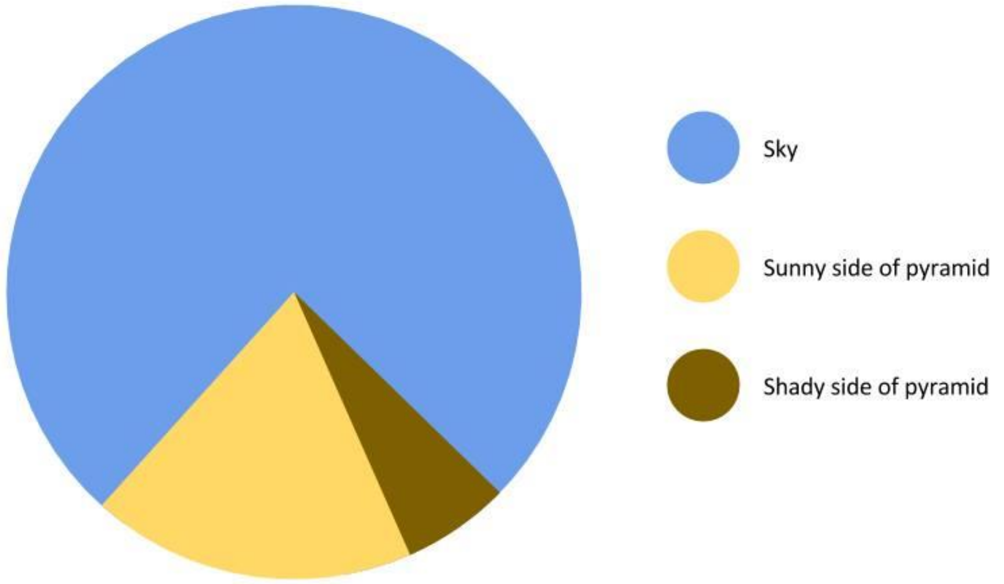
Quick review

Basics of R

```
> my_vec <- c(5, 28, 19)
> my_vec[3]
> my_vec[3]  <-  7
```

How to plot categorical data

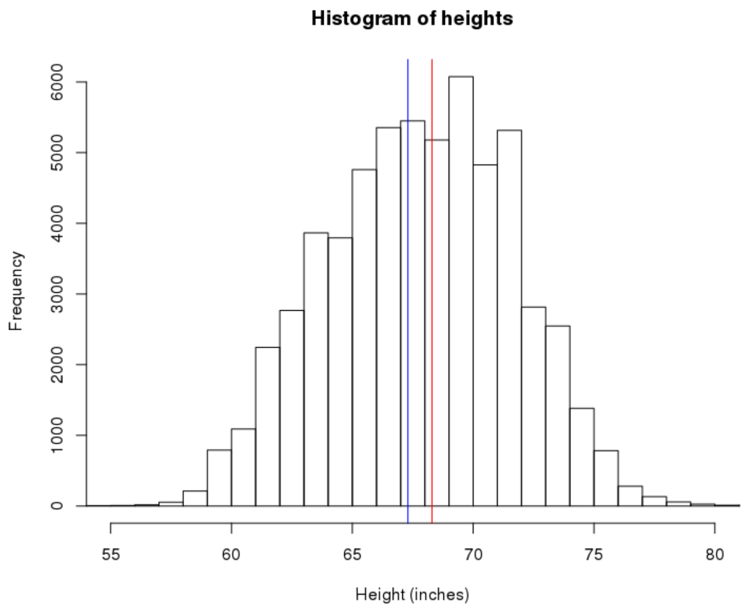
```
> drinks_table <- table(profiles$drinks)
> barplot(drinks_table)
> pie(drinks_table)
```

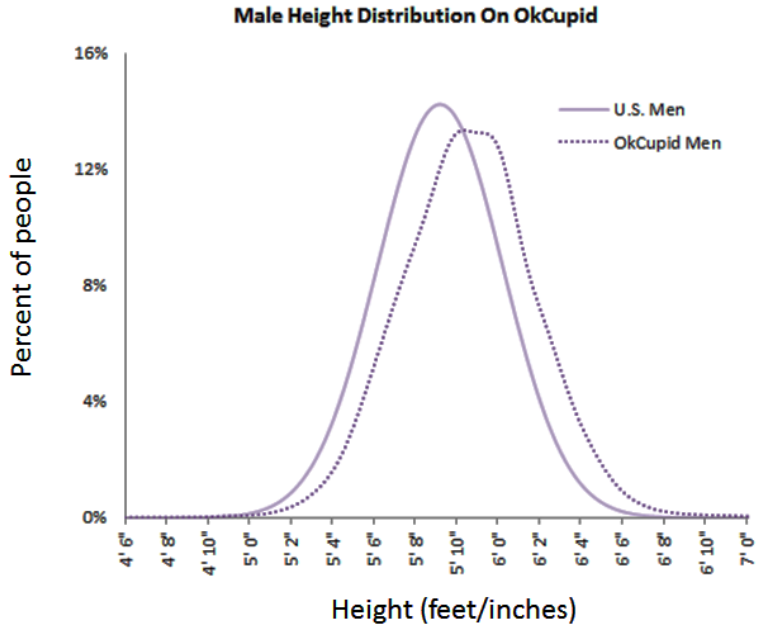


Quick review

How to plot quantitative data:

```
> hist(profiles$height)
> abline(v = 67)
```





For loops

For loops are useful when you want to repeat a piece of code many times under similar conditions

The syntax for a for loop is:

```
for (i in 1:100) {
        # do something
}
```

This is repeated 100 times

i is incremented by 1 each time

For loops

For loops are particular useful in conjunction with vectors…

```
my_results <- NULL      # create an empty vector to store the results
for (i in 1:100) {
        my_results[i] <- i^2
}
```

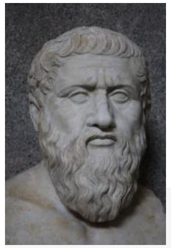
Try this at home!: Use a for loop to create a vector that holds the values at multiples of 3 from 3 to 300

- i.e., 3, 6, 9, …, 300

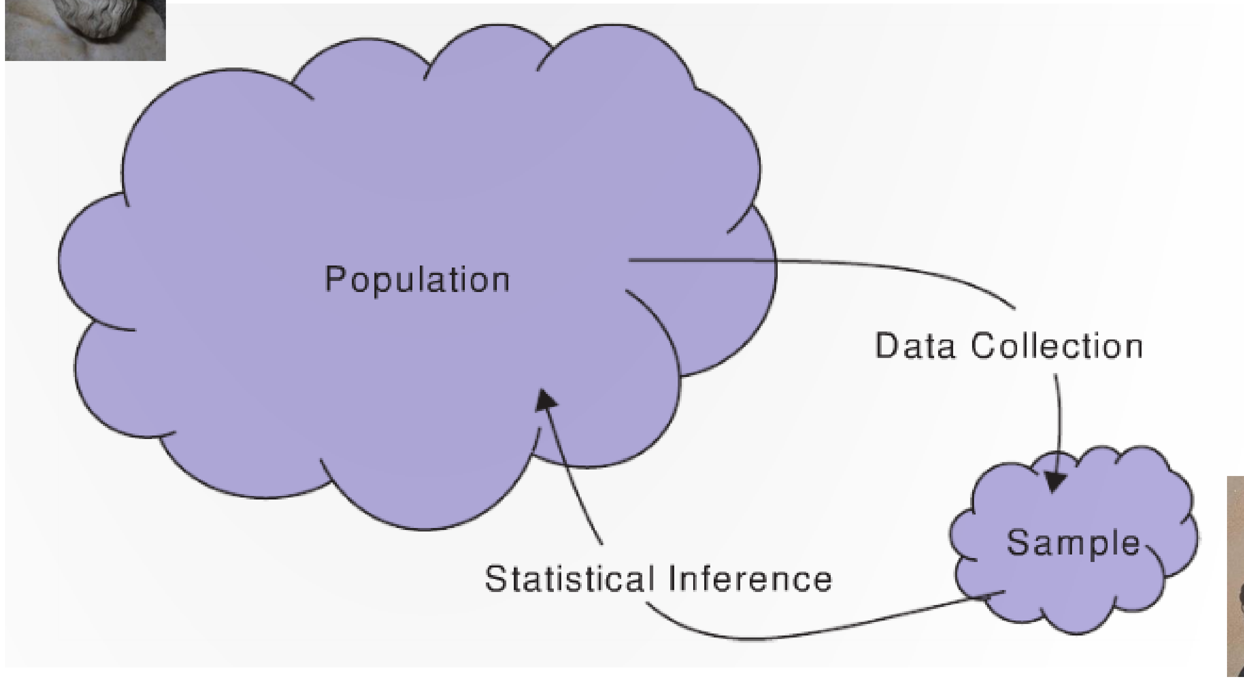
Questions?

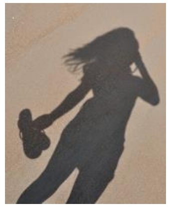
Review and extension of statistical concepts

Where does data come from?



Population: all individuals/objects of interest

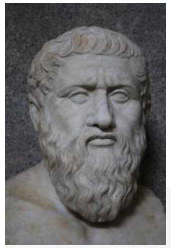




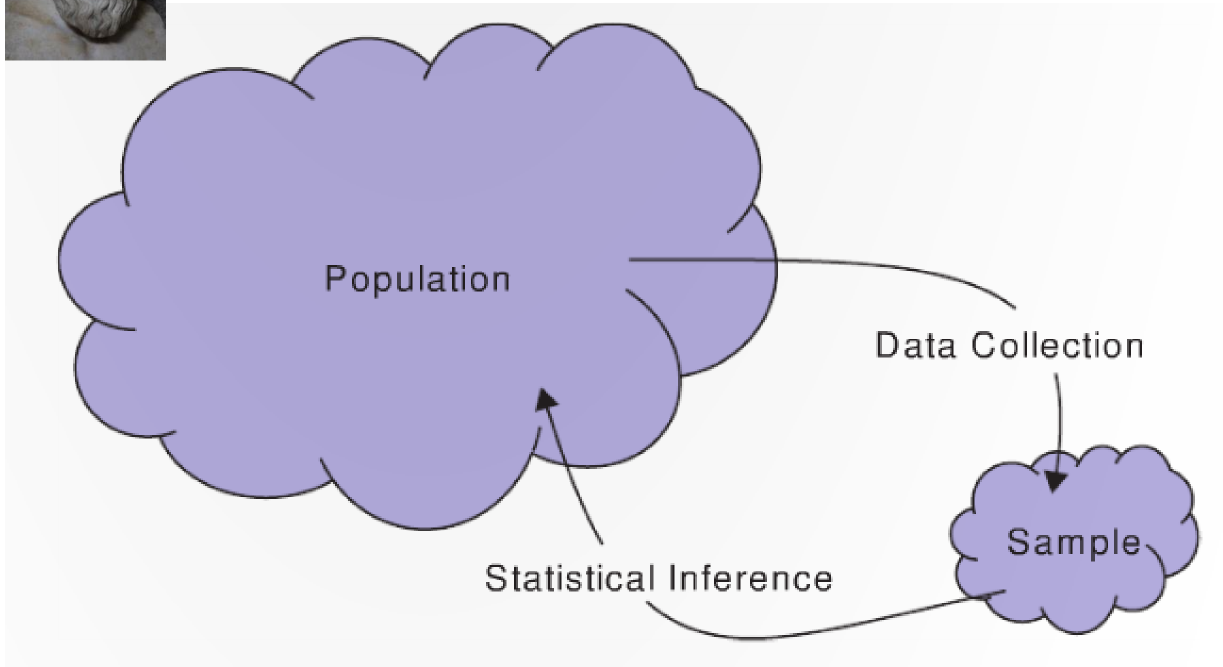
Sample: A subset of the population

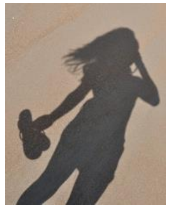
Where does data come from?

Question: Is the okcupid profiles data frame a population or a sample?



Parameters: π, μ, σ, ρ, β





Question: If the OkCupid profiles data frame is a sample, what is the population?

Statistics: $\hat{p}$, $\bar{x}$, s, r, b

How do we get sample of data?

Simple random sample: each member in the population is equally likely to be in the sample

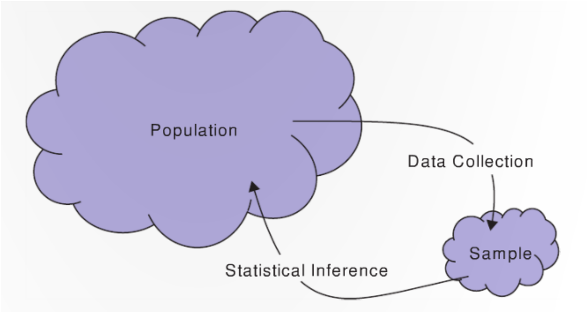
"Random selection"

Soup analogy!

Q: Why is this good?

A: Allows for generalizations to the population!

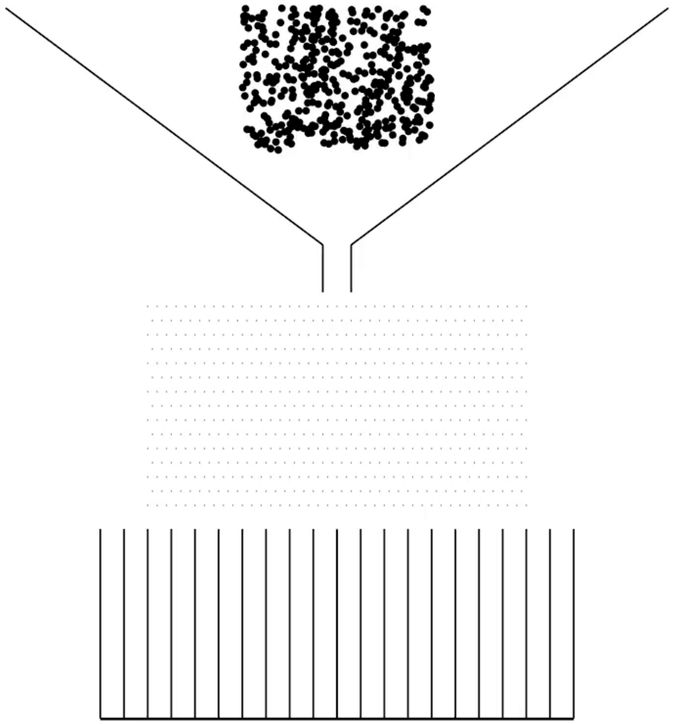
- No sampling bias
- Statistic (on average) equal parameter
  - E.g., $E[\bar{x}] = \mu$



Questions:

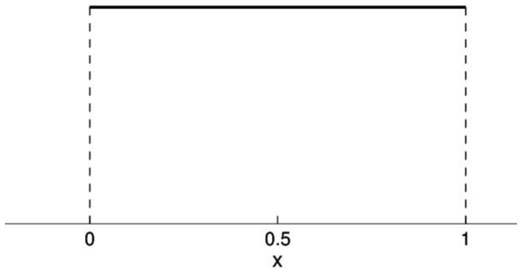
- Is the OkCupid profiles data a simple random sample?
- Would we expect sampling bias from statistics computed from the OkCupid profiles?

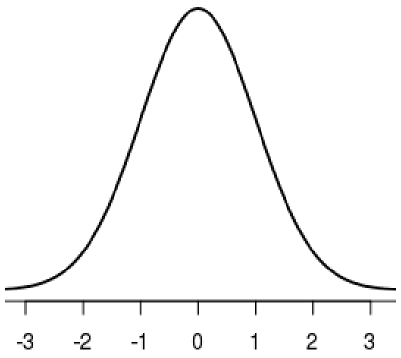
Big picture for the week



We can use R to:

- Simulate randomly sampling individual data points from a population
  - E.g., random heights of each person in this classroom
- Simulate statistics calculated from many random samples; i.e., simulate ***sampling distributions***
  - E.g., suppose we took the average height of everyone in this class, and several other classes, and created a distribution of these average heights
- Use randomly generated data to assess the validity of statistical methods.
- And much more…





Big picture for the week

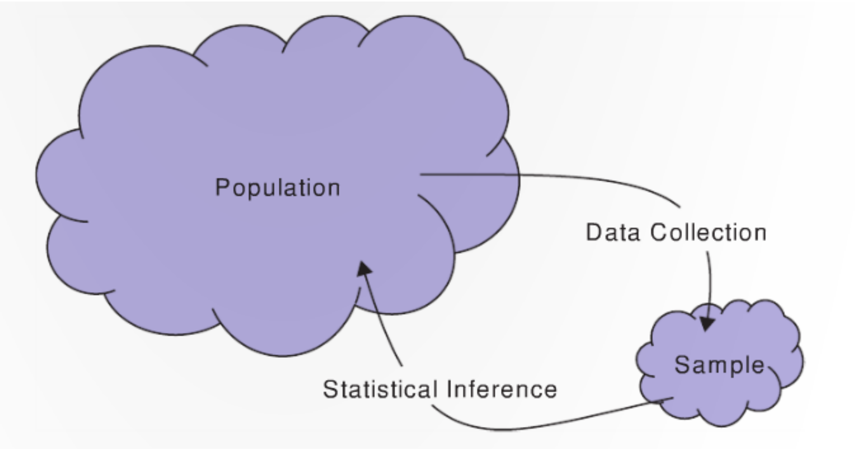
Statistics are point estimates of parameters

We can use sampling distributions (i.e., distributions of statistics) to tell us how much we can trust ***any one statistic*** to be a good point estimate of a parameter

-> confidence interval

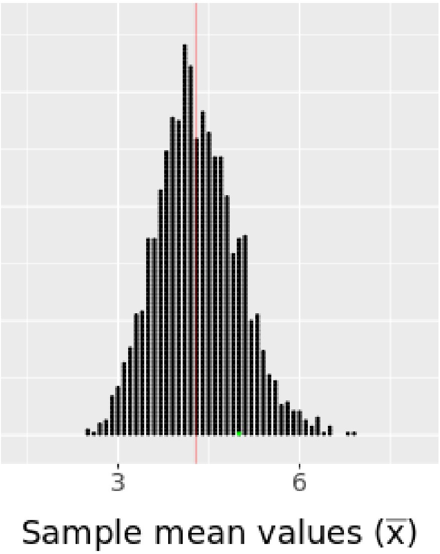
Let's starts on this now…

parameter: μ



statistic: $\bar{x}$

μ



Sampling distribution of $\bar{x}$

Generating random data and probability models

To understand our data, it is often useful to be able to:

1. Simulate data in a way that replicates key properties of the data

2. Create mathematical (probability) models of our data

Generating random data and probability models

To understand our data, it is often useful to be able to:

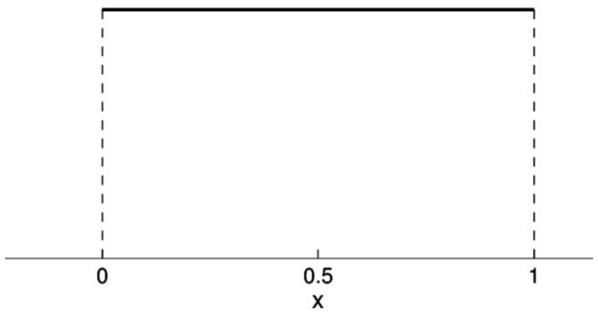
1. Simulate data in a way that replicates key properties of the data

2. Create mathematical (probability) models of our data

Generating random data

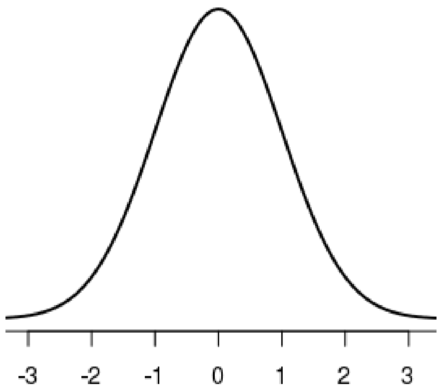
R has built in functions to generate data from different distributions
- All these functions start with the letter ***r***

The uniform distribution




```
# generate n = 100 points from U(0, 1)
    > rand_data <- runif(100)
    > hist(rand_data)
```

The normal distribution




```
 # generate n = 100 points from N(0, 1)
    > rand_data  <- rnorm(100)
    > hist(rand_data)
```

Generating random data

R has built in functions to generate data from different distributions

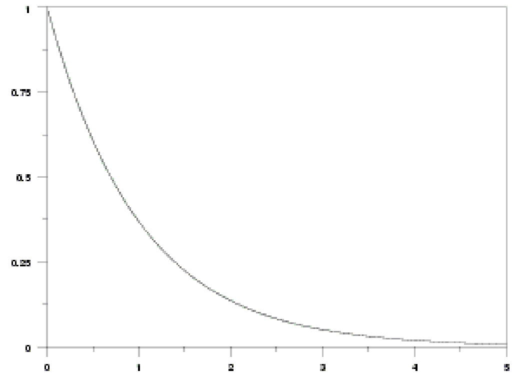
- All these functions start with the letter ***r***

The exponential distribution

generate n = 100 points from exponential(λ = 1)

> Homework 2

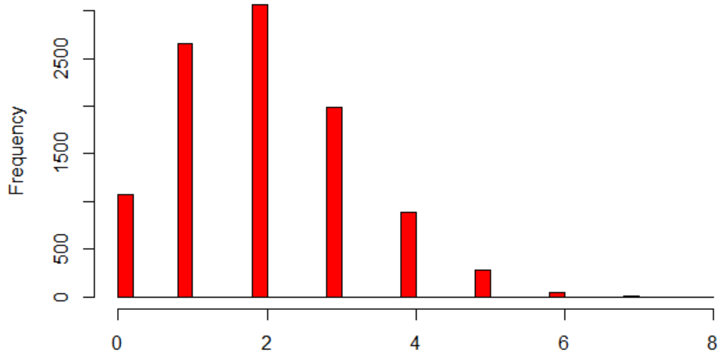
>



The binomial distribution

generate n = 100 points from binomial(n = 8, π = .2)

```
> rand_data  <- rbinom(100, 8, .2)
> hist(rand_data)
```



Generating random data

If we want the same sequence of random numbers we can set the random number generating seed

```
> set.seed(123)
> runif(100)
```

Q: Why would we want the same sequence of random number?

A: Reproducibility!

Generating random data and probability models

To understand our data, it is often useful to be able to:

1. Simulate data in a way that replicates key properties of the data

2. Create mathematical (probability) models of our data

Density Curves

A **density curve** is a mathematical function f(x) that can be used to model data

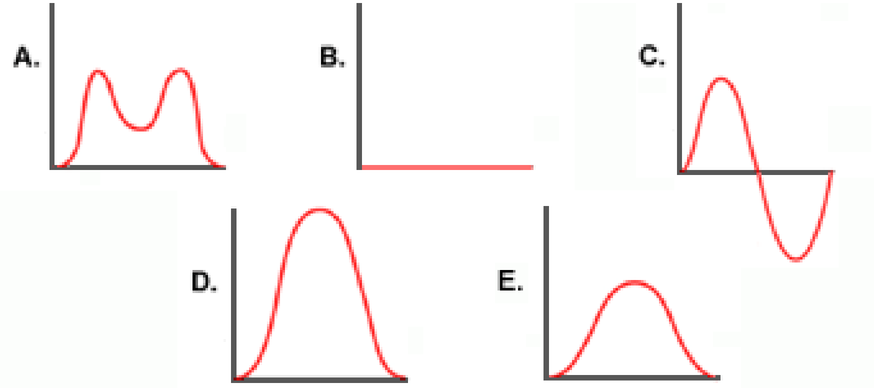
- We can imagine density curves as histograms that have:
  - Infinitely large data sample
  - With infinitely small bins sizes
  - Normalized to have an area of 1

Density curves have two defining properties:

1. The total area under the curve f(x) is equal to 1

2. The curve is always ≥ 0

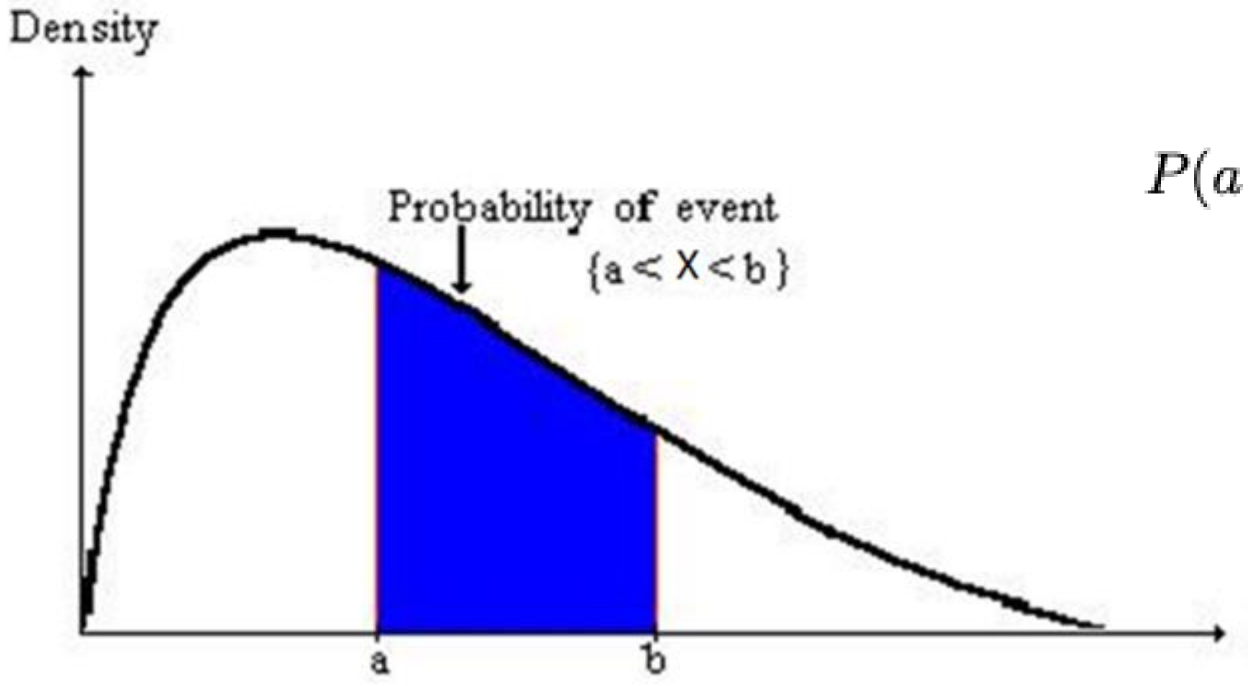
Which of these could ***not*** be a density curve?



Density Curves

The area under the density curve in an interval [a, b] models the probability that a random number X will be in the interval

Pr(a < X < b) is the area under the curve from a to b



$$P(a < X < b) = \int_a^b f(x)dx$$

Examples of density curves

R has built in functions to create density curves

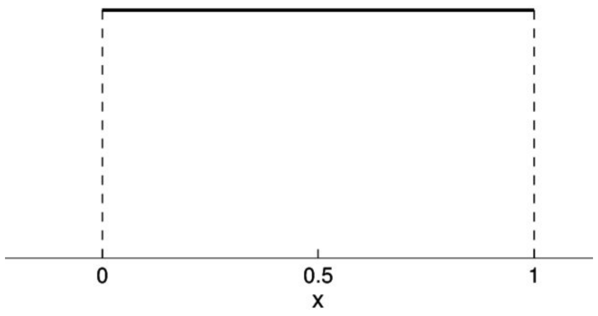
- All these functions start with the letter ***d***

The uniform distribution

- (here b = 1, a = 0)

```
> x <- seq(-.2, 1.2, by = .001)
> y <- dunif(x)
>  plot(x, y, type = "l")
```

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \le x \le b, \\ 0 & \text{for } x < a \text{ or } x > b \end{cases}$$

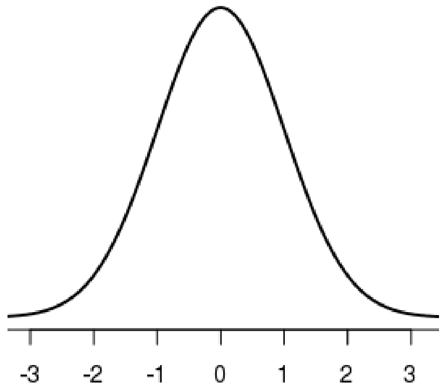


The normal distribution

- (here μ = 0, σ = 1)

```
>  x <- seq(-3, 3, by = .001)
>  y <- dnorm(x)
> plot(x, y, type = "l")
```

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Examples of density curves

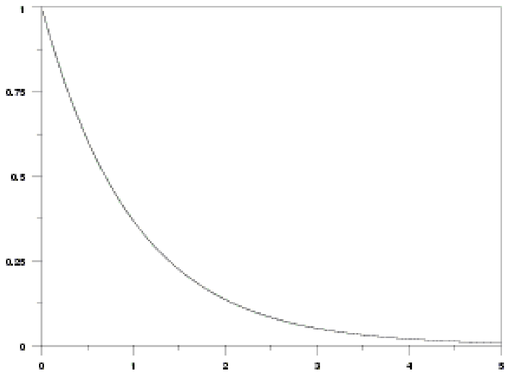
R has built in functions to create density curves

- All these functions start with the letter ***d***

The exponential distribution

> Homework 2
>
>
>

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0, \\ 0 & x < 0. \end{cases}$$

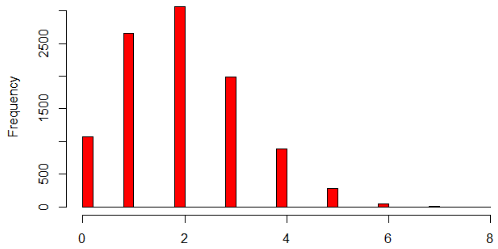


The binomial distribution

- (actually a probably mass function)

```
> x <- 0:8
> y <- dbinom(x, 8, .2)
> names(y) <- x
> barplot(y)
```

$$f(k,n,p) = \binom{n}{k} p^k (1-p)^{n-k}$$



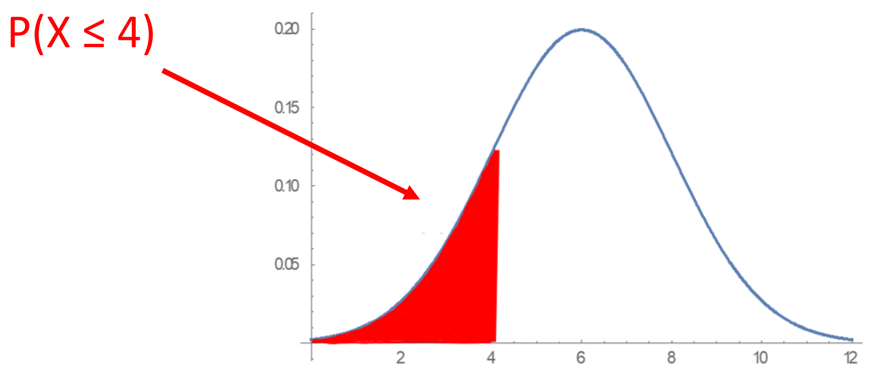
Cumulative distribution functions

Cumulative distribution functions give the probability of getting a random value X less than or equal to a value x: $P(X \leq x)$

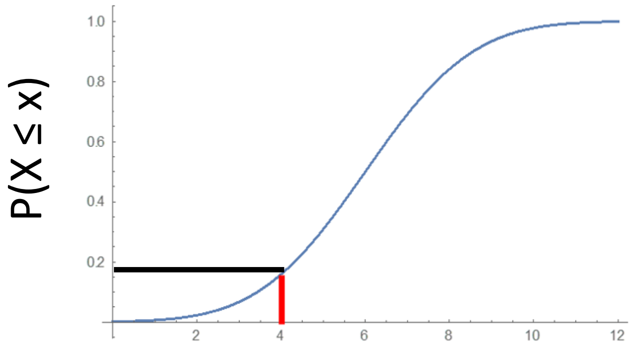
- For example, we would write the probability of getting a random number X less than 2 as: $P(X \leq 2)$

Cumulative distribution functions are obtained by calculating the area under a probability density function

Density: f(x)



Cumulative distribution F(x)



$$P(X \leq x)$$

$$= F(x)$$

$$= \int_{-\infty}^{x} f(x)dx$$

Examples of cumulative distributions in R

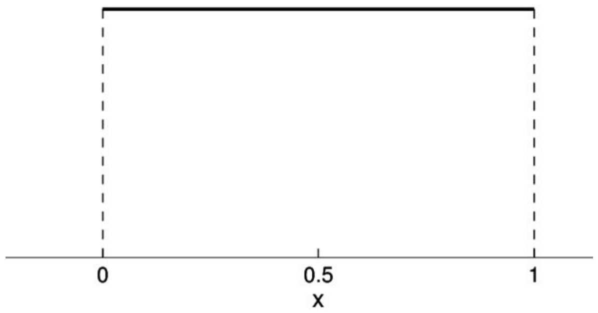
R has built in functions to get probabilities from different distributions

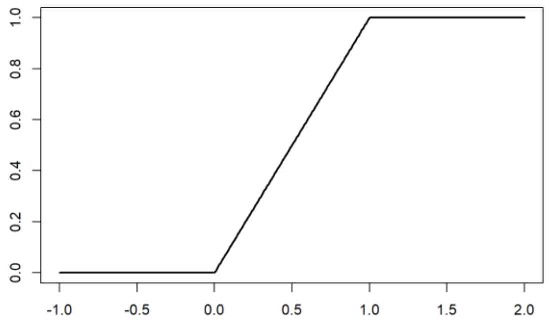
- All these functions start with the letter ***p***

The uniform distribution

\# P(X ≤ .25)

punif(.25)

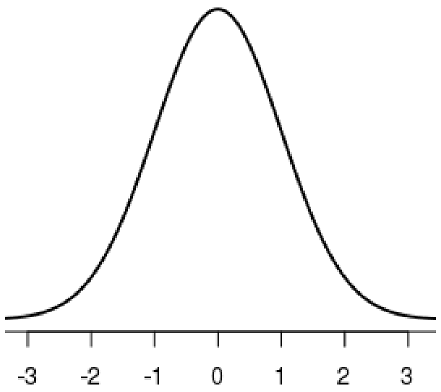


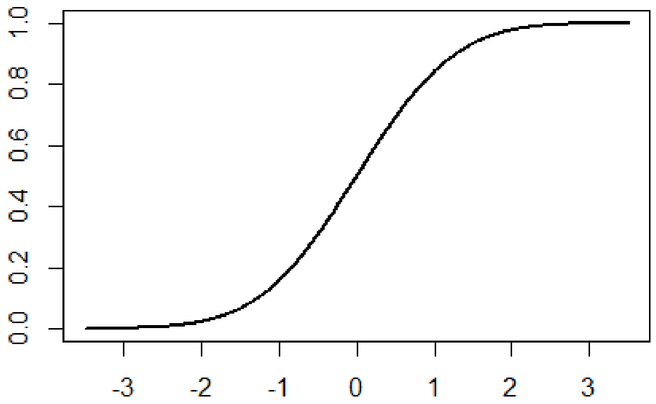


The normal distribution

\# P(X ≤ 2)

pnorm(2)





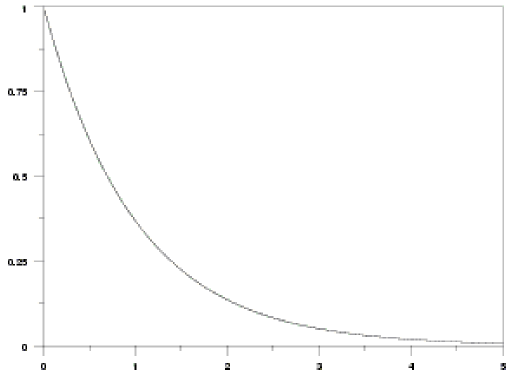
Examples of cumulative distributions in R

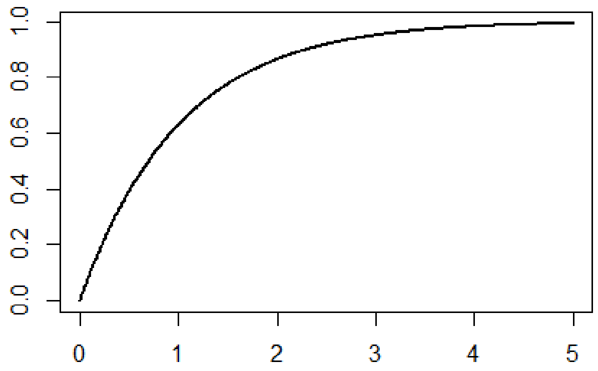
R has built in functions to get probabilities from different distributions
- All these functions start with the letter ***p***

The exponential distribution

P(X ≤ 2)

pexp(2)

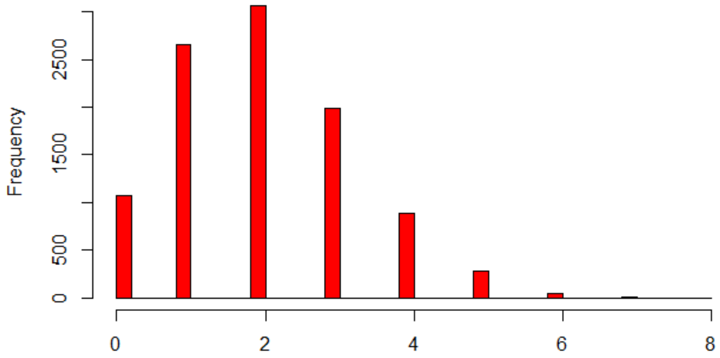


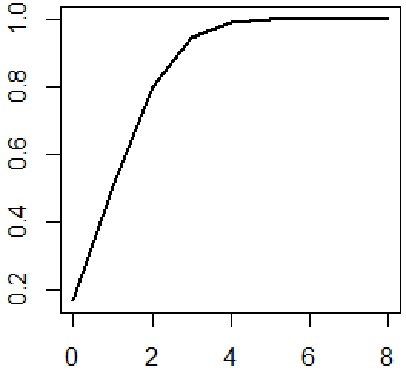


The binomial distribution

P(X ≤ 2; n = 8, π = .2)

pbinom(2, 8, .2)





Sampling distributions

Sample statistics

Q: What is a statistic?

A: A statistic is number computed from a function on a sample of data

The sample mean $\bar{x}$ (shadow of the parameter μ)

```
> rand_data <- runif(100)        # generate n = 100 points from U(0, 1)
> mean(rand_data)
```

Q: If we repeat the code above will we get the same statistic?

- A: unlikely

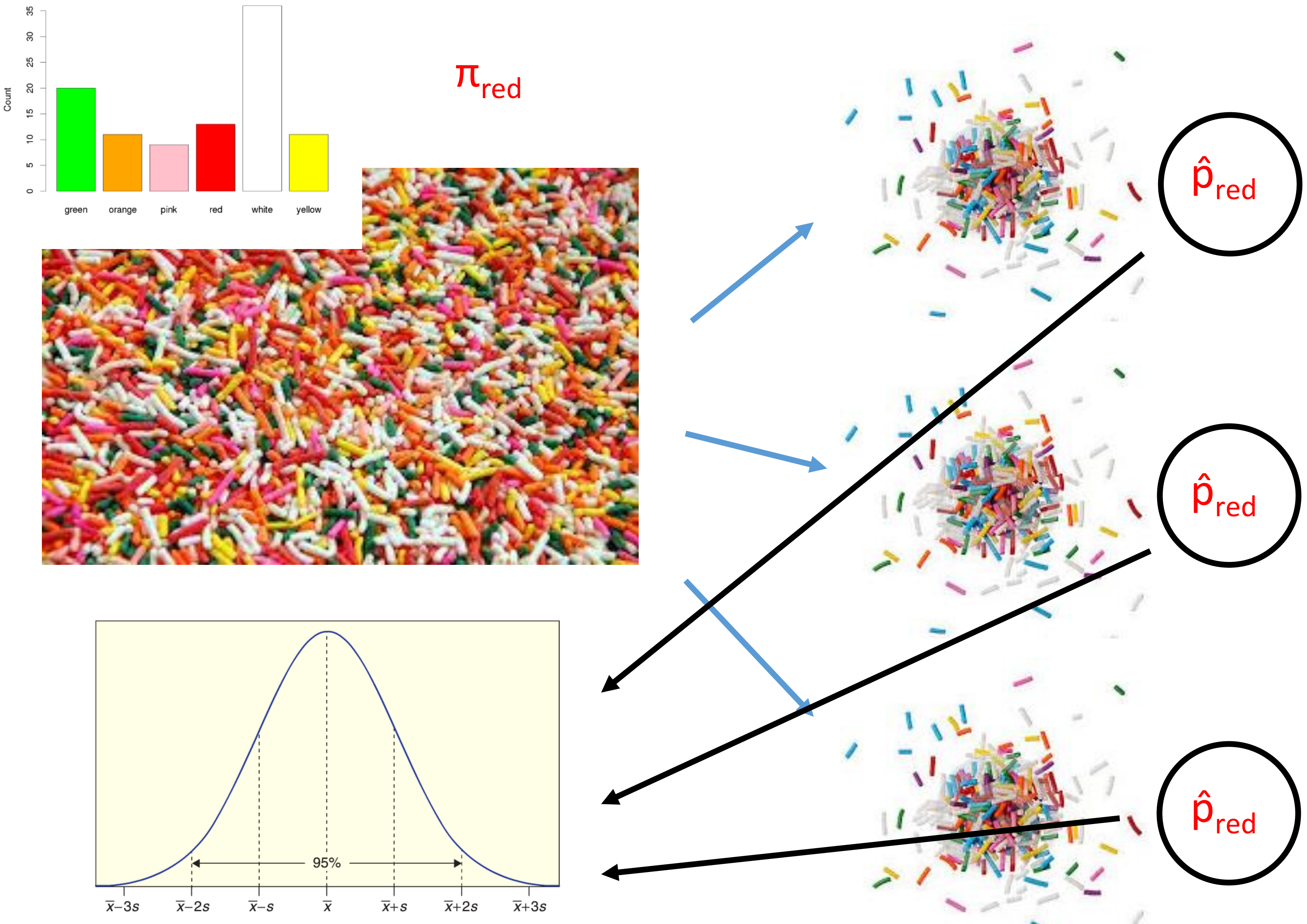
Sampling distributions

A ***sampling distribution*** is a distribution of ***statistics***

<u>Reminder</u>: For a *single* ***categorical variable***, the main statistic of interest is the ***proportion*** ($\hat{p}$) in each category

- (shadow of the parameter π)

$$\hat{p} = \text{Proportion in a category} = \frac{\text{number in that category}}{\text{total number}}$$

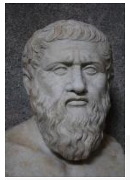


Sampling distribution!

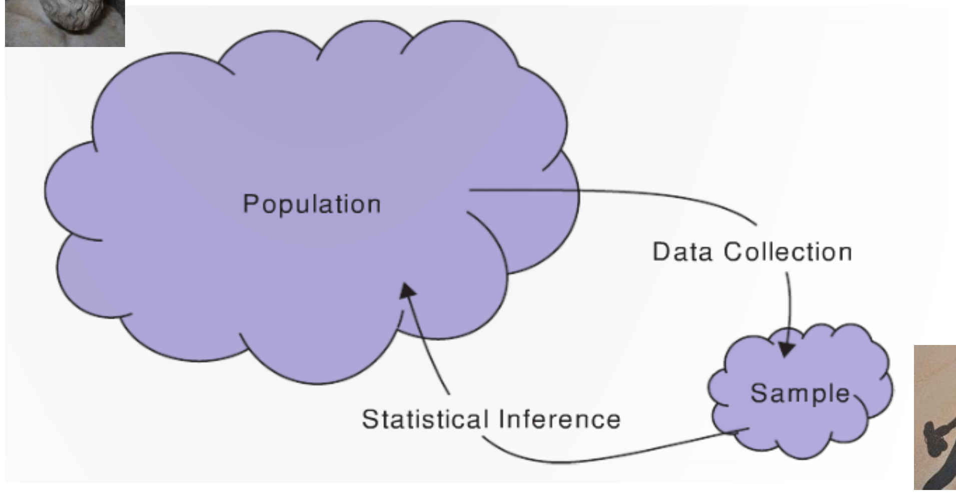
Sampling distribution

Why would we be interested in the sampling distribution?

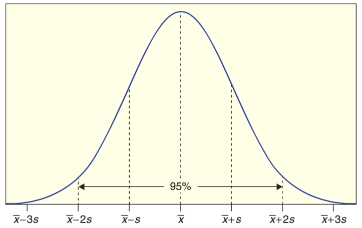
- If we knew what the sampling distribution was, then we could evaluate how much we should trust individual statistics



Parameters: π, μ, σ, ρ, β



Sampling distribution



Statistics: $\hat{p}$, $\bar{x}$, s, r, b

Simulating sampling distributions

```
sampling_dist <- NULL
for (i in 1:1000) {
	rand_data <- runif(100)   # generate n = 100 points from U(0, 1)
	sampling_dist[i] <- mean(rand_data)   # save the mean
}

hist(sampling_dist)
```

Simulating sampling distributions

Distribution of OkCupid user's heights n = 100

```
heights <- profiles$height

# get one random sample of heights from 100 people
height_sample <- sample(heights, 100)

# get the mean of this sample
mean(height_sample)
```

Simulating sampling distributions

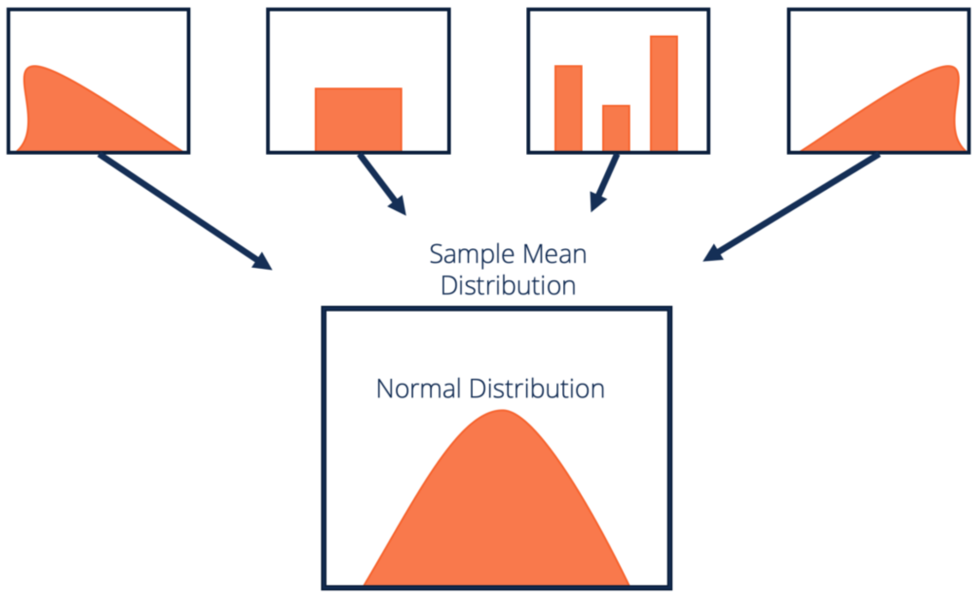
Distribution of OkCupid user’s heights n = 100

```
sampling_dist <- NULL
for (i in 1:1000) {
        height_sample <- sample(heights, 100)    # sample 100 random heights
        sampling_dist[i] <- mean(height_sample)    # save the mean
}

hist(sampling_dist)
```

The central limit theorem

The **central limit theorem** establishes that, in many situations, when independent random variables are summed up, their properly normalized sum tends toward a normal distribution.



Since many statistics we use are the sum of randomly data, many of our sampling distributions will be approximately normal

- You will explore this more on homework 2

Statistics: $\hat{p}$, $\bar{x}$, s, r, b

If there is extra time…

Confidence intervals

Point Estimate

We use the statistics from a sample as a **point estimate** for a population parameter

- $\bar{x}$ is a point estimate for…? μ

A NPR/PBS NewHour/Marist poll listed Biden's approval rating at 43%

Symbols:

π: Biden's approval for all voters

$\hat{p}$: Biden's approval for those voters in our sample

Interval estimate based on a margin of error

An **interval estimate** give a range of plausible values for a population parameter

One common form of an interval estimate is:

Point estimate ± margin of error

Where the **margin of error** is a number that reflects the precision of the sample statistic as a point estimate for this parameter

Example: Fox news poll

43% of American approve of Biden's job performance, plus or minus 3%

How do we interpret this?

Says that the <u>population parameter</u> (π) lies somewhere between 40% to 46%

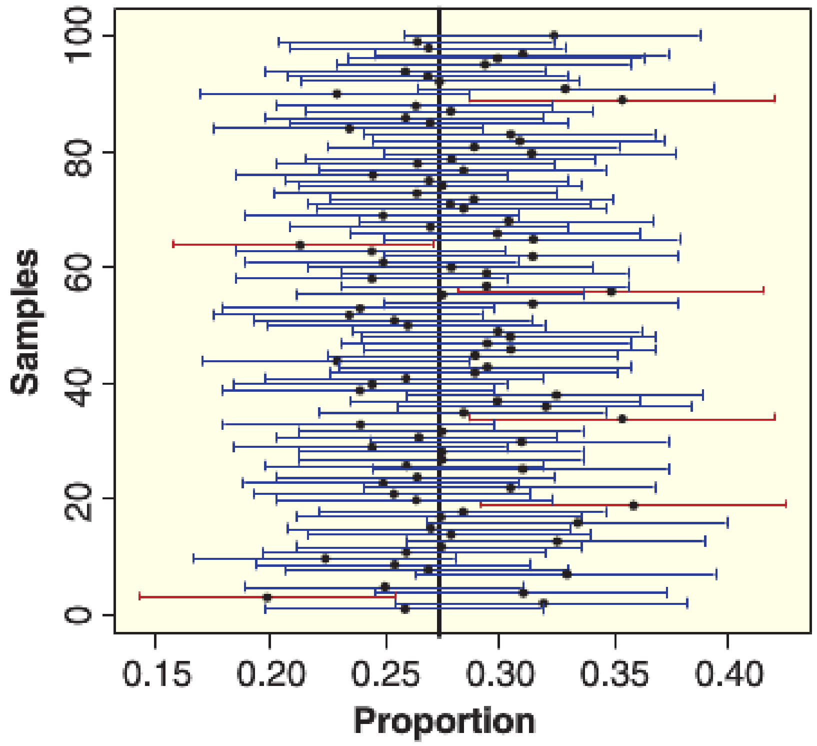
i.e., if they sampled all voters the true population proportion (π) would be likely be in this range

Confidence Intervals

A **confidence interval** is an interval <u>computed by a method</u> that will contain the ***parameter*** a specified percent of times

- i.e., if the estimation were repeated many times, the interval will have the parameter x% of the time

The **confidence level** is the percent of all intervals that contain the parameter



Think ring toss…

Parameter exists in the ideal world

We toss intervals at it

95% of those intervals capture the parameter