Lecture: Probability Distributions and Frequentist Inference

Selective R refresher

Preface: Base R vs tidyverse

- Base R:
  - collection of native commands
  - historical legacies: not necessarily a "clean" language/syntax
- `tidyverse`:
  - an "opinionated collection of R packages" for data visualization, transformation, tidying, and import
  - more or less uniform syntax
  - Packages: `dplyr`, `ggplot2`, `tibble`, `readr`, `tidyr`, `purrr`, and many others
- both are valid
- both are fully compatible with one another
- both are fully compatible with external packages
- you can do amazing things by relying on one, on the other, or on both
- we will use whichever best suits our purposes (but, mostly, specialized packages like `brms`, `modelsummary` and `marginaleffects`)

Some important base R

- En route to using `brms`, we will explore some manual implementations of *Markov Chain Monte Carlo* (MCMC) samplers
- The produce *numeric samples* from the posterior distributions of (a) parameter(s)
  - for a single parameter, these are stored in a *vector* of length S (S posterior samples, one parameter)
  - for K parameters, these are stored in a *matrix* of dimensions $S \times K$ (S posterior samples in rows, K parameters in columns)

Numeric vectors

A vector is a serial listing of data elements of the same type (e.g. `integer`, `double`, `logical`, or `character`).

Example

```
numeric_vector <- c(.99, 1.49, 1.99, 2.79, 1.89)
```

Numeric matrices

A matrix is a rectangular arrangement of data elements of the same type.

```
numeric_matrix <- matrix(seq(-.35, .35, .1), nrow = 2, ncol = 4)
```

```
##       [,1]  [,2] [,3] [,4]
## [1,] -0.35 -0.15 0.05 0.25
## [2,] -0.25 -0.05 0.15 0.35
```

Note: By default, we fill the 8 elements of the 2×4 matrix by columns. Specify the option `byrow = TRUE` if you want to fill by rows instead.

Arithmetic operators and common transformations

When working with numeric objects like vectors or matrices, we can apply arithmetic operators and mathematical transformations...

```
x + y     # addition
x - y     # subtraction
x * y     # multiplication
x / y     # division
x ^ y     # exponentiation
x %% y    # modulus
x %/% y   # integer division
log(x)    # natural logarithm
exp(x)    # exponential
```

```r
sqrt(x)   # square root
t(a)      # vector/matrix transposition
```

Generic mathematical functions

...as well as generic mathematical functions:

```r
mean()      # arithmetic mean()
median()    # median
quantile()  # quantile(s)
sd()        # standard deviation
var()       # variance
```

apply

Use `apply()` on matrices or arrays to apply a function across selected dimension(s) of a matrix.

Row- and column-means

```
##       [,1]  [,2] [,3] [,4]
## [1,] -0.35 -0.15 0.05 0.25
## [2,] -0.25 -0.05 0.15 0.35
```

```
## [1] -0.05  0.05
```

```
## [1] -0.3 -0.1  0.1  0.3
```

Custom functions You can apply custom functions to all apply functions:

```
## [1] 2 2
```

```
## [1] 2 2 0 0
```

Probability distributions

Univariate probability distributions

Univariate probability distributions give the distribution of probabilities for all feasible realizations of a single *random variable*.

It characterizes the (cumulative) probability mass of each possible outcome for a *discrete/categorical* variable or the (cumulative) probability density of each value within the support of a *continuous* variable.

Functions

For common probability distributions, R features the following four commands (where `dist` is a placeholder):

1. `ddist(x)`: Probability density/mass function (pdf/pmf). Takes a value x and returns the probability density/mass $P(X = x)$.

2. `pdist(x)`: Cumulative distribution function (CDF). Takes a value x and returns the cumulative probability $P(X \leq x)$.
3. `qdist(q)`: Quantile function or inverse CDF. Takes a cumulative probability $P(X \leq x)$ and returns the corresponding value x.
4. `rdist(n)`: Produces n random draws from the distribution.

Discrete case: The binomial distribution

A binomial distribution characterizes the probability of outcomes for a series of `n` independent realization of a binary random variable.

Numerically, we define a binary variable as a variable that can either take on a value of zero ("failure") or one ("success").

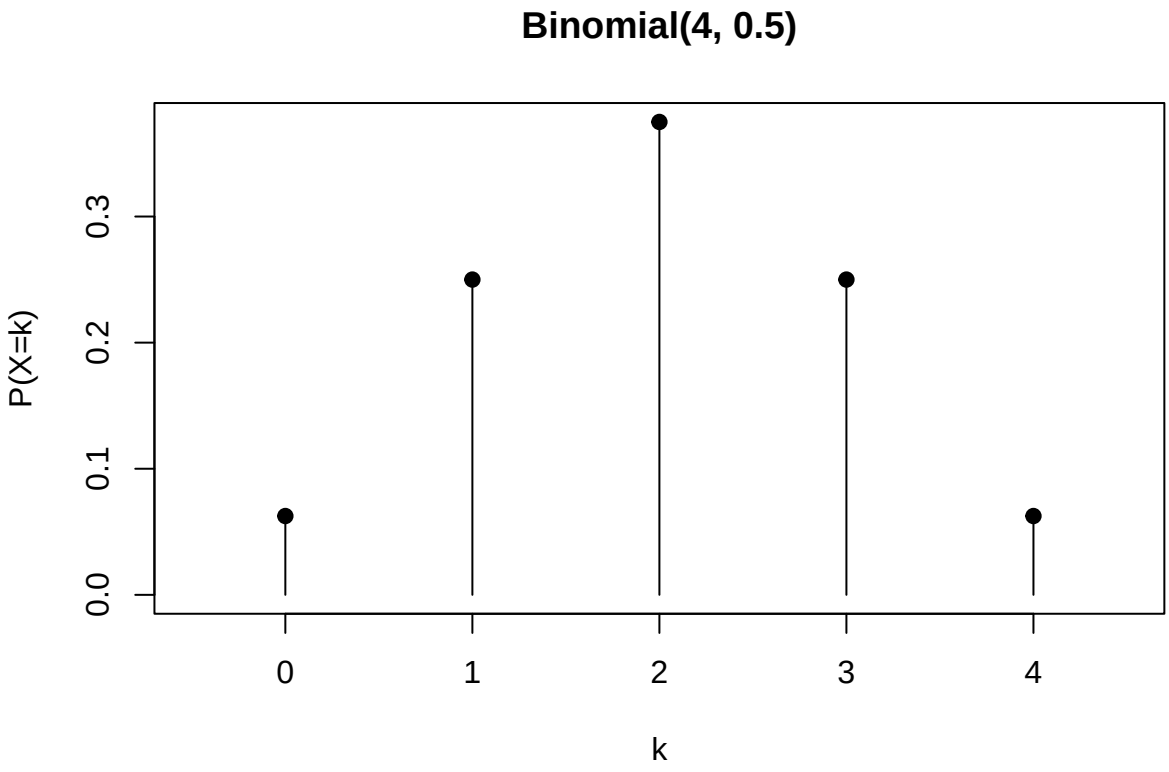
An example is a series of coin flips n, each of which produces either heads or tails. The numerical assignment of ones and zeroes is arbitrary. Here, we will consider heads as ones or successes. We denote the number as heads/successes as k.

The binomial distribution is governed by a single *probability parameter*, $\pi \in [0, 1]$. It determines the probability of successes/ones for each realization. Accordingly, $1 - \pi$ gives the probability of failure/zeroes.

In our example, π determines the fairness of a coin. A fair coin with $\pi = 0.5$ is equally likely to produce heads and tails for each flip.

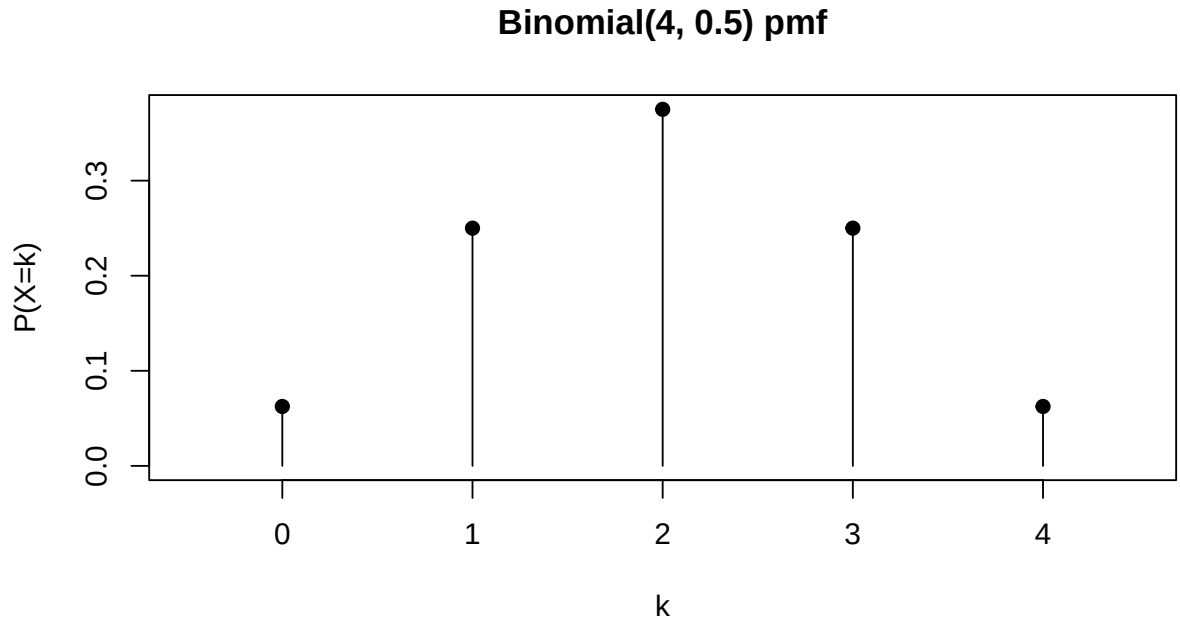
Probability mass function (pmf)

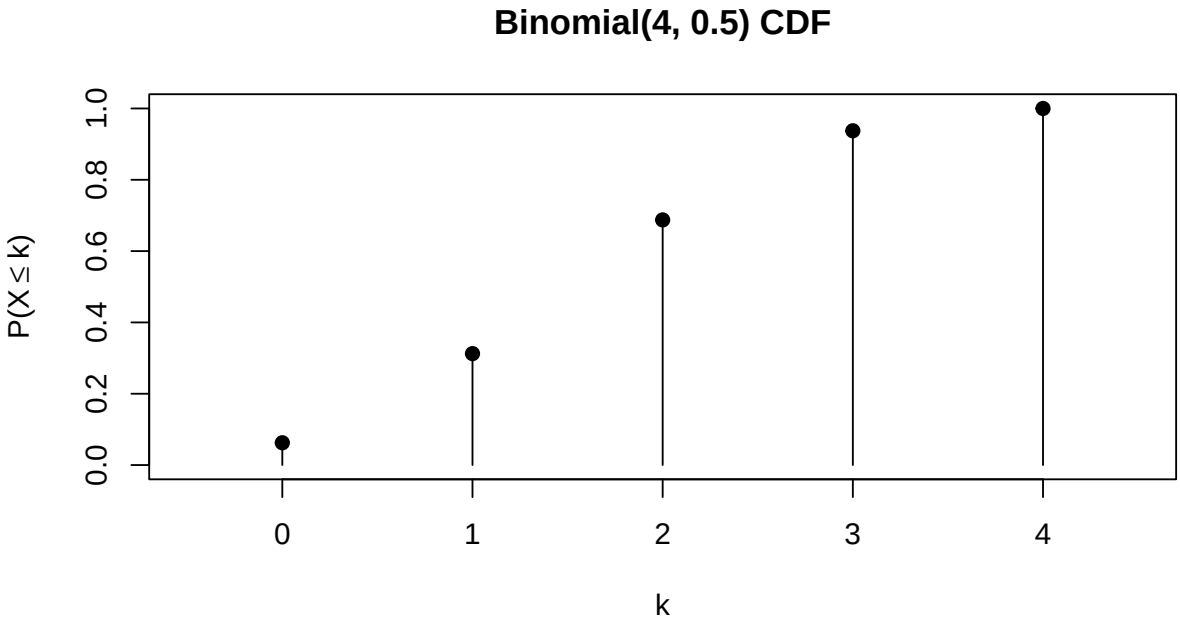
Below, we show the probability mass function of a random variable X that records the probability of each count of heads $k \in \{0, 1, 2, 3, 4\}$ out of $n = 4$ flips with a fair coin, i.e., $\pi = 0.5$: $k \sim \text{Binom}(4, 0.5)$.



What do these probabilities sum to?

Cumulative distribution function (CDF)



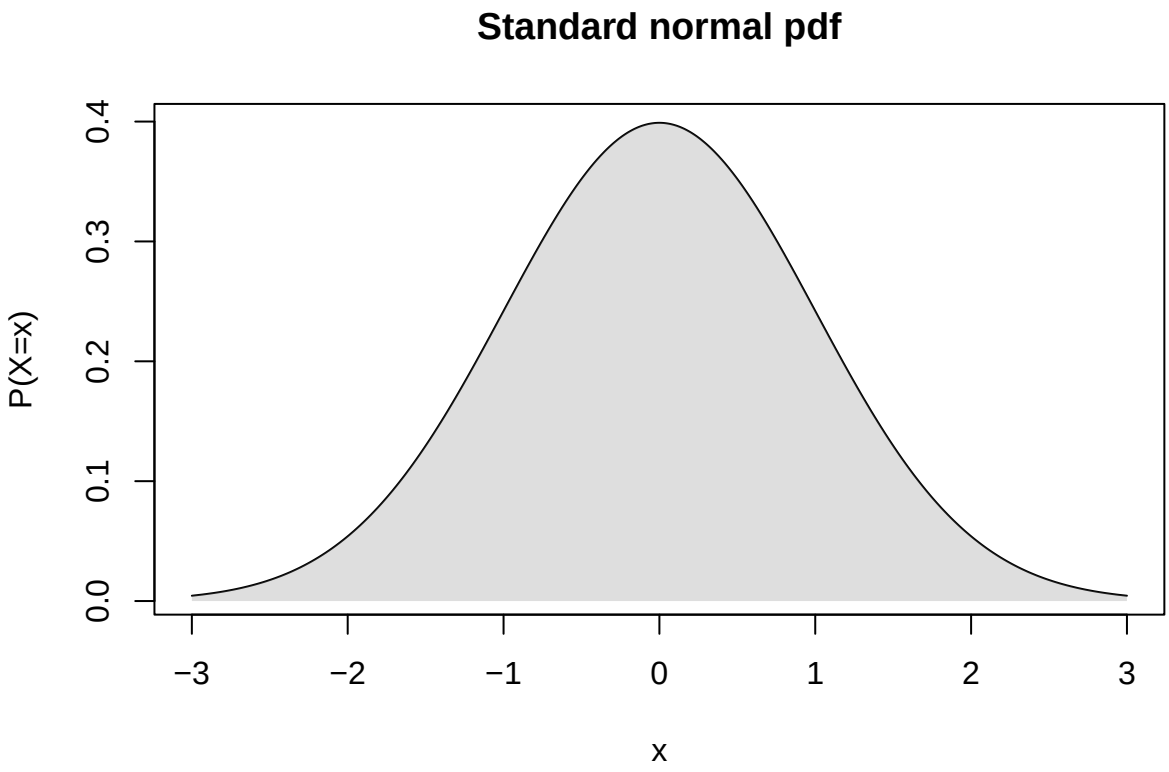


Continuous case: The standard normal distribution

Let's illustrate the continuous case using everyone's favorite, the standard normal.

The normal distribution is characterized by two parameters: Mean μ and *standard deviation* σ.

The standard normal distribution is a special case where $\mu = 0$ and $\sigma = 1$, $\mathrm{N} \sim (0, 1)$:

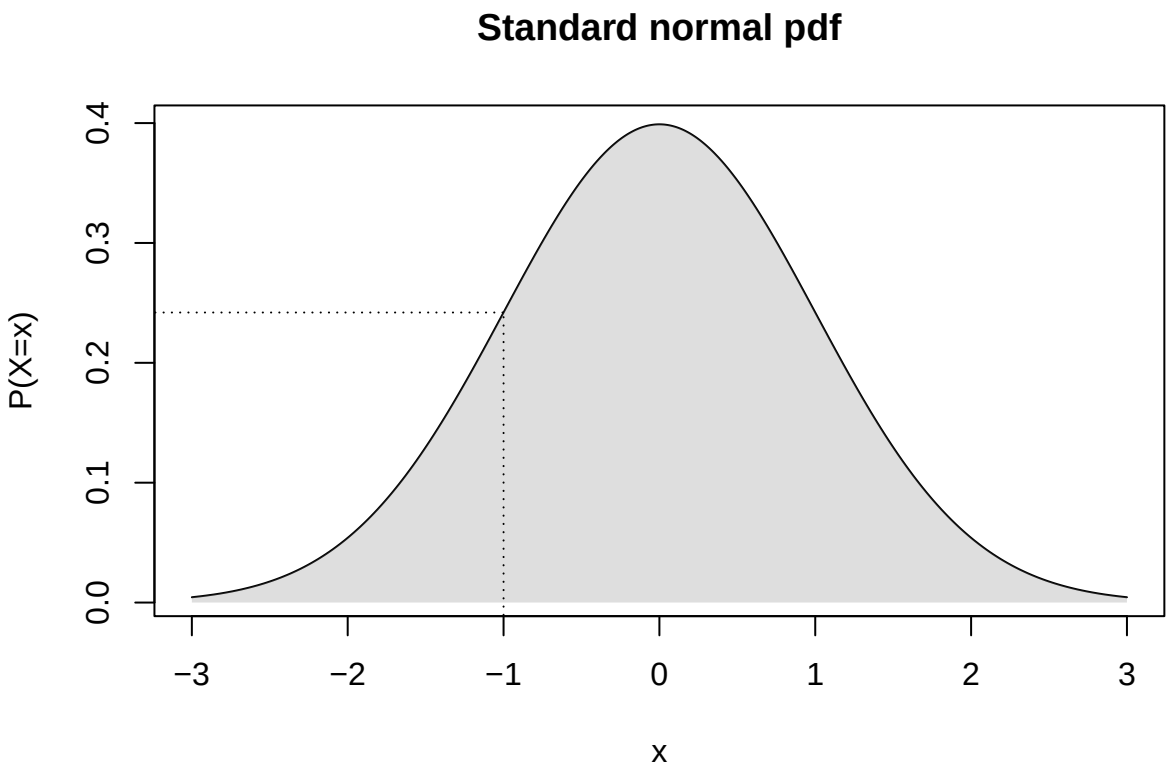


pdf

To get the probability density at any value x, e.g., $x = -1$, run:

```
dnorm(-1, mean = 0, sd = 1)
```

```
## [1] 0.2419707
```

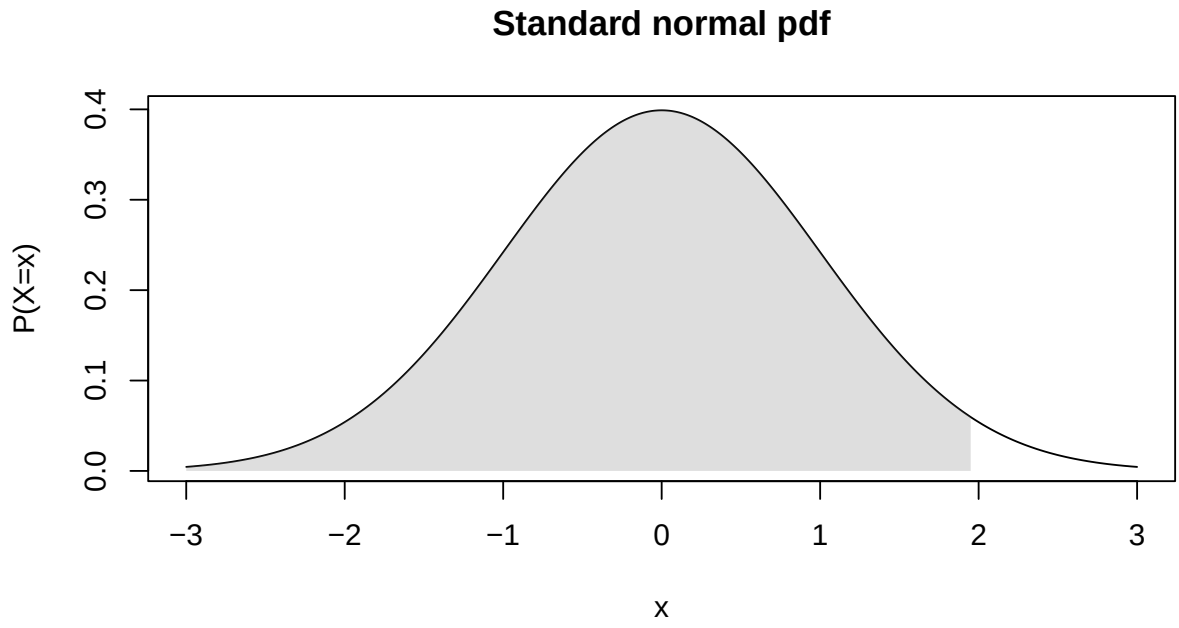


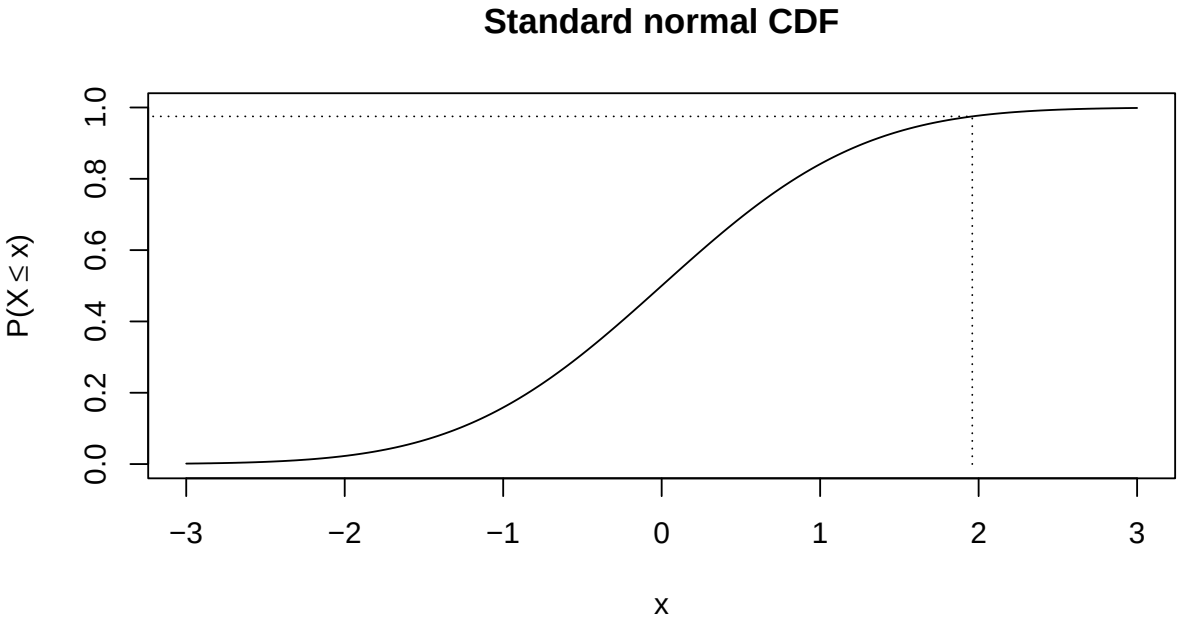
CDF and quantile function

To get the cumulative probability up to any value x, e.g., $x = 1.9599$, run:

```
pnorm(1.9599, mean = 0, sd = 1)
```

```
## [1] 0.9749963
```

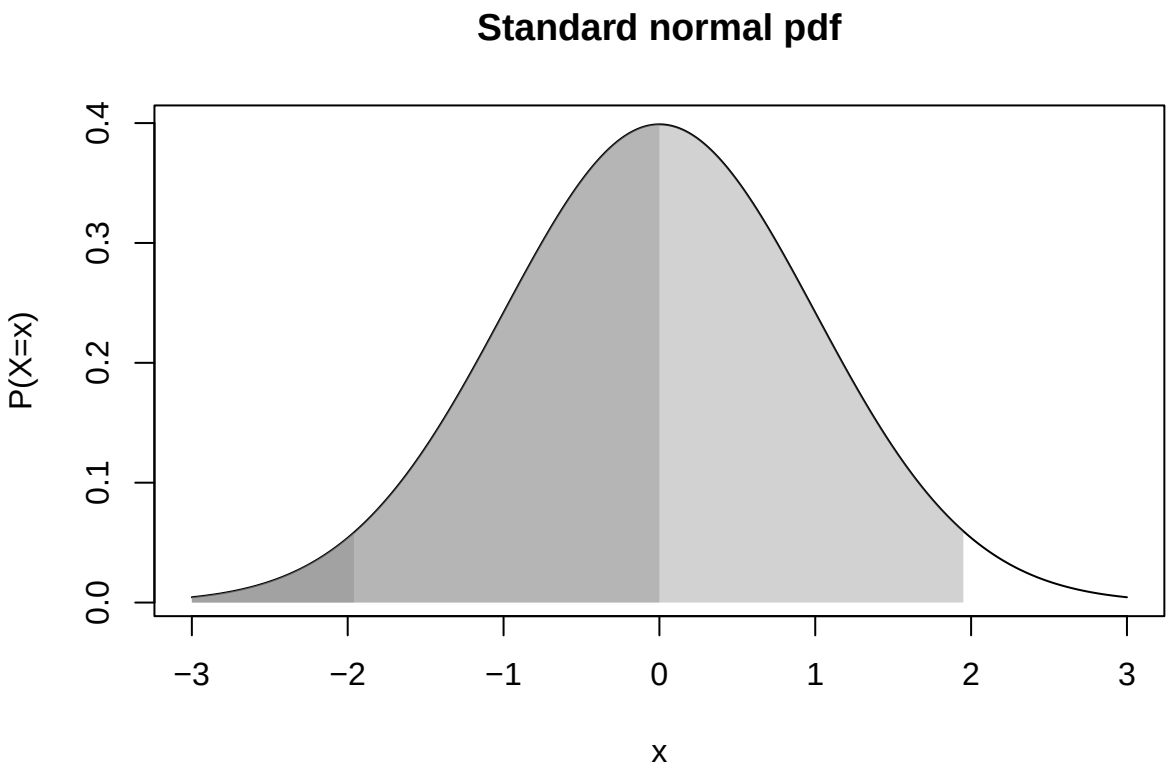




Conversely, use `qnorm` to get the x values for any desired cumulative probability:

```
qnorm(c(.025, .5, .975))
```

```
## [1] -1.959964  0.000000  1.959964
```



Random number generation

Lastly, to generate random draws from a distribution with specific parameters, use `rnorm()`. The higher the number of draws, the better the chances that the frequency distribution of your random draws will approximate the underlying pdf:

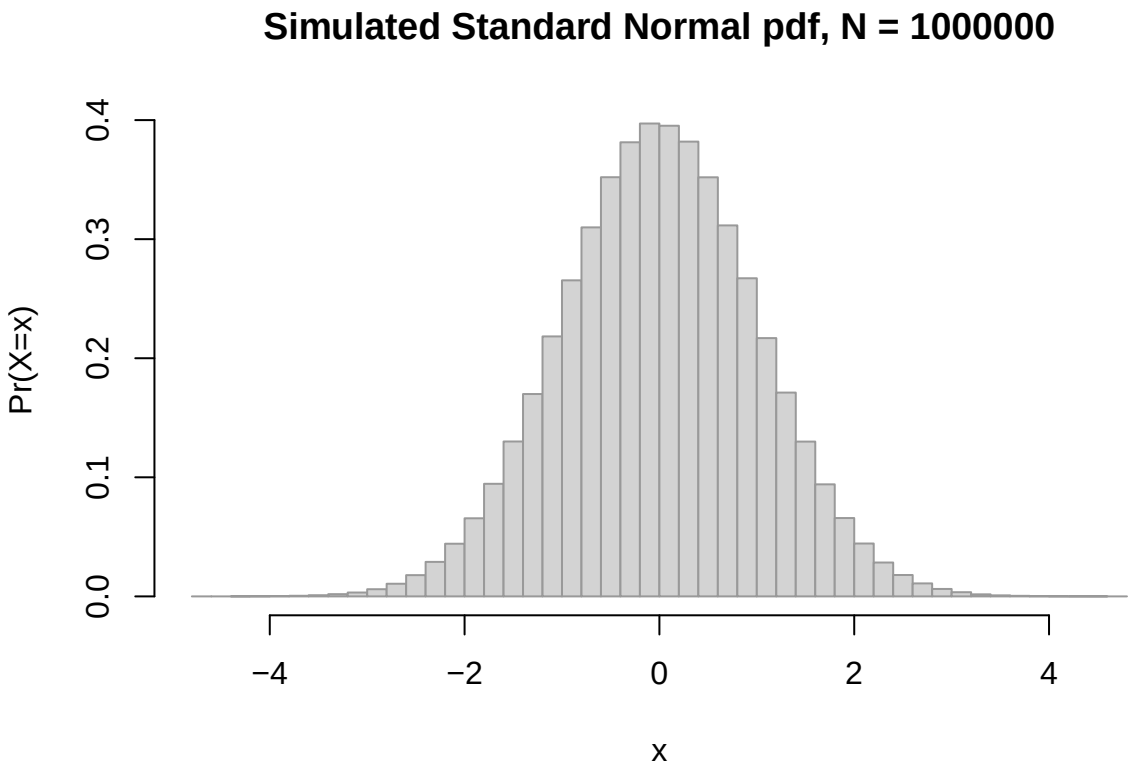
```
# Define number of draws
n_sim <- 1000000L

# Set a seed for replicability
set.seed(20231122L)

# Take random draws
x_sim <- rnorm(n_sim, mean = 0, sd = 1)

# Visualize
```

```r
hist(
  x_sim,
  main = paste0("Simulated Standard Normal pdf, N = ", n_sim),
  xlab = "x",
  ylab = "Pr(X=x)",
  freq = FALSE,
  breaks = log(n_sim) * 3,
  border = "gray60"
)
```



Note: `set.seed()` ensures the replicability of random number generation (!).

Multivariate probability distributions

Bivariate probability distributions give the distribution of probabilities for all feasible *joint* realizations of *two* random variables.

Multivariate probability distributions generalize this idea to *three or more* random variables.

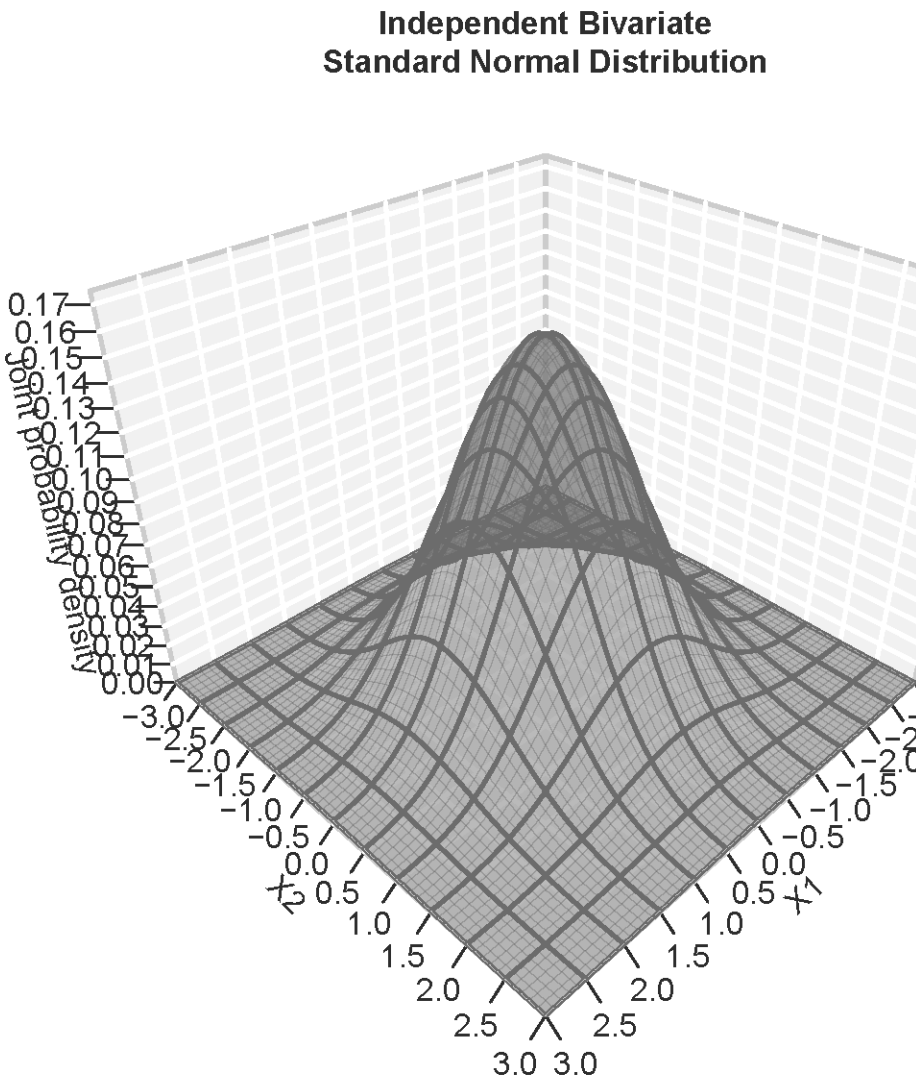
Multivariate probability distributions carry the same information for the constitutive variables as univariate probability distributions. We call these variable-specific distributions *marginal probability distributions*.

Additionally, multivariate probability distributions carry information about the *mutual dependence* of the constitutive variables. This is typically captured by *covariance parameters*. Independent random variables share a covariance of zero.

Example: The bivariate standard normal distribution (independent case)

The following shows the joint probability density function of a two standard normal distributions.

We can think of it as a bell-shaped hill with a circular base whose height reaches its peak at $X1 = 0$ and $X2 = 0$.

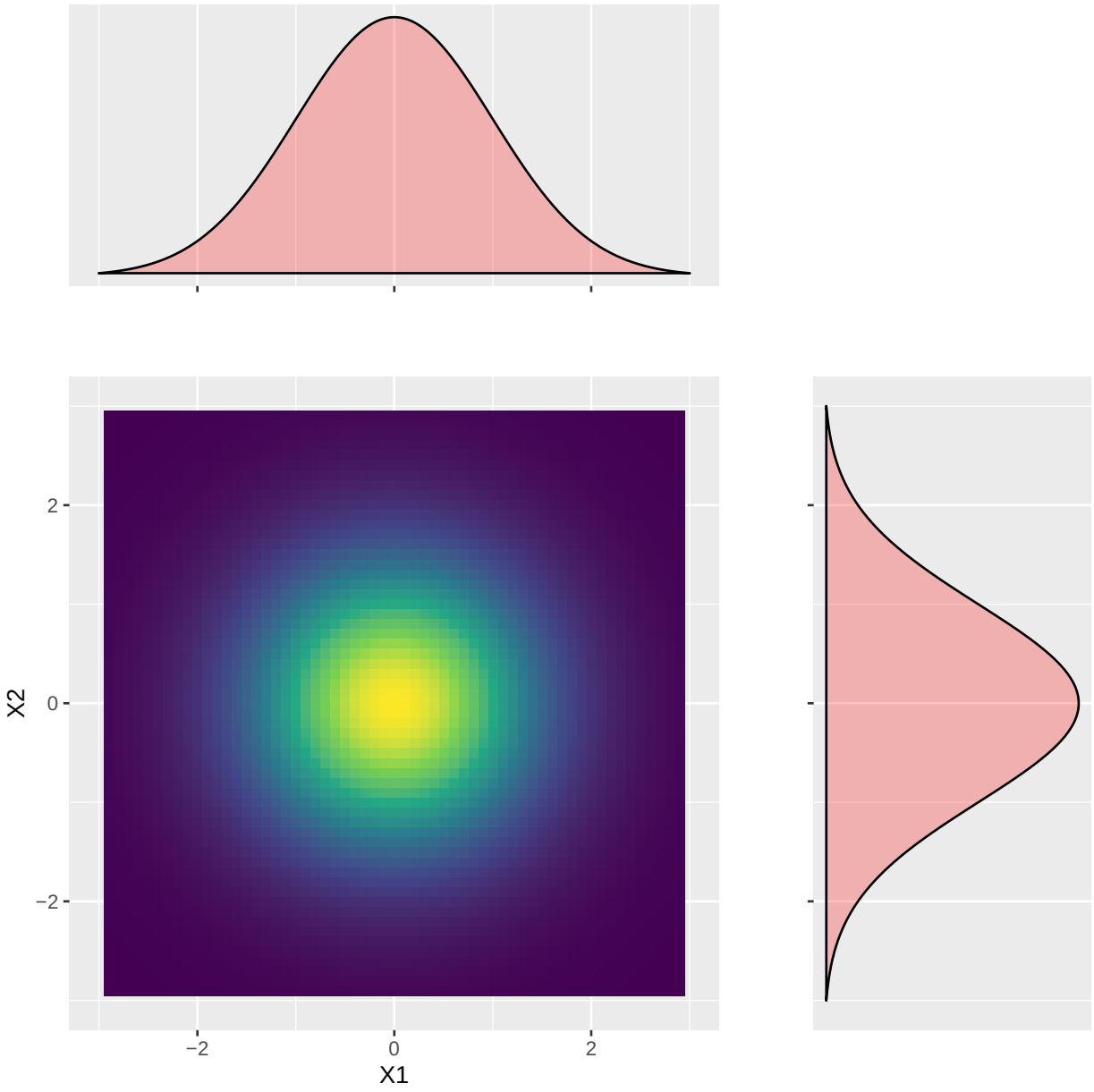


Heatmaps

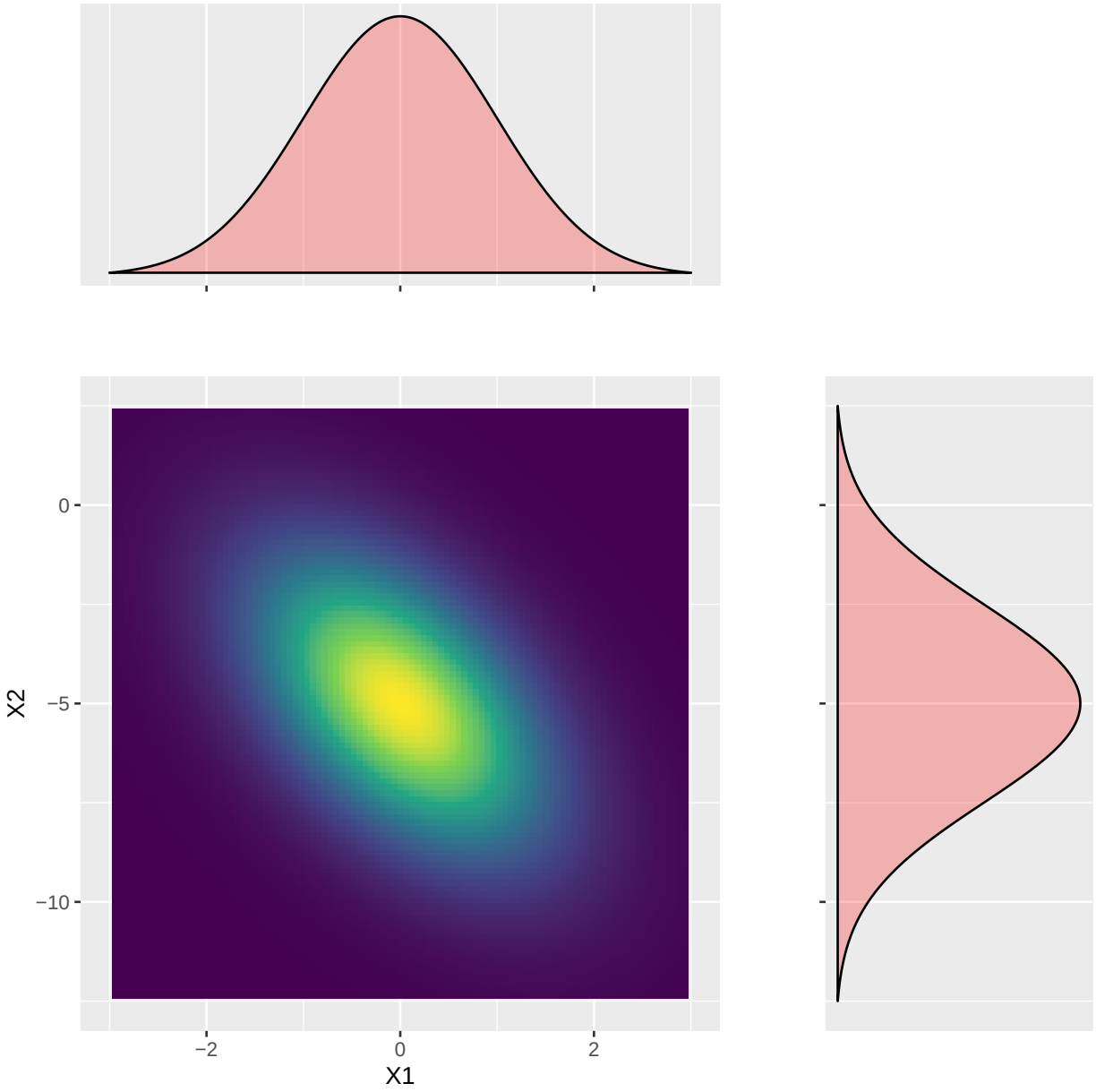
Heatmaps are a great way of visualizing joint densities in 2D.

They take a bird's eye perspective and re-express the third dimension (height) through the intensity of color shadings.

They allow us to add information on the marginal distributions, too.


2
0
−2
X2
−2
0
2
X1

Example: The bivariate normal distribution (interdependent case)



Can you guess:

- The approximate means and standard deviations of X_1 and X_2?
- Whether the correlation between X1 and X2 is positive or negative?

Frequentist inference

Ask Wikipedia?

The first two sentences of the English-language Wikipedia article defines statistical inference as follows:

1. Statistical inference is the process of using data analysis to infer properties of an underlying distribution of probability.
2. Inferential statistical analysis **infers properties of a population**, for example by testing hypotheses and deriving estimates.

The first sentence is universally true across both frequentist and Bayesian frameworks of statistical inference.

However, as we will learn in this course, the bold-printed is exclusive to the *frequentist* framework of statistical inference.

Key characteristics

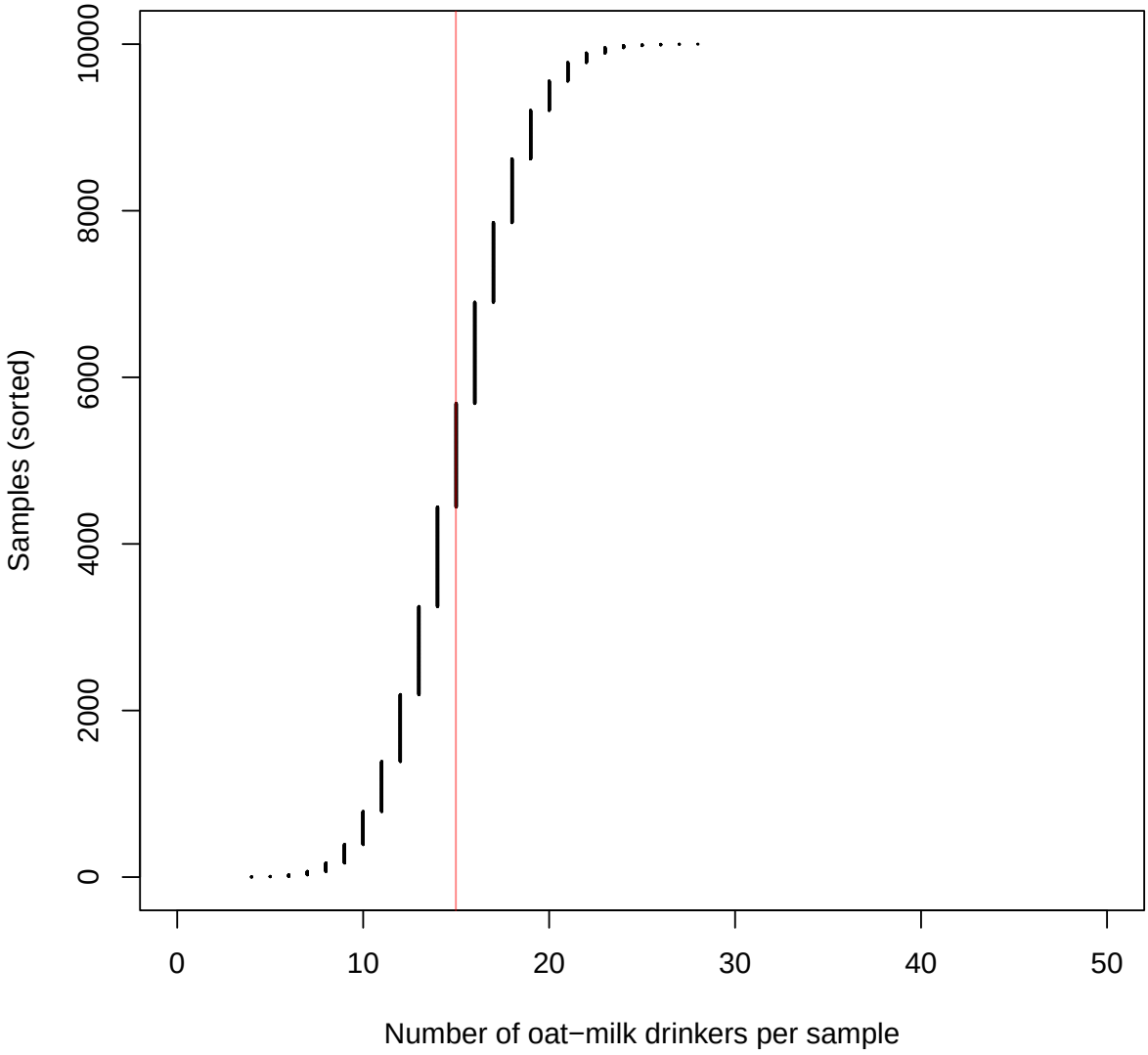
- Frequentist analysis seeks to infer *population parameters*. Such parameters – e.g., the population mean of a variable X – are considered *fixed*: Their values are true, exact, and unknown.
- To learn about such population parameters, we rely on a finite sample. Every such sample is considered one out of many possible random samples from the underlying population.
- This implies *repeatable data*: In theory, we should have a constant stream or pool of independently and identically distributed data, from which we could sample as often as we like.
- Every random sample is an imperfect representation of the underlying population due to *sampling variability*.
- Imagine we took many independent samples from the same underlying population, and computed the sample analogue of the population parameter for each of them – using a suitable estimator/statistic, the resulting distribution of sample statistics would yield a probability distribution known as the *sampling distribution*.

Repeated sampling: A though experiment

- Consider an infinite population.

- We want to infer the population proportion of a binary variable X – say, the number of café-goers who order beverages with oat milk.
- For this task, we will simulate $S = 10,000$ samples of size $N = 50$.
- Unbeknownst to us, the true population proportion is $\pi = 0.3$.
- Given that X is binary, we treat each sample as a series of independent Bernoulli trials – i.e., a draw from a Binomial$(50, 0.3)$ distribution, where the probability parameter is equal to the population proportion of X.

Let us first look at the variability in the number of oat-milk drinkers in our samples:



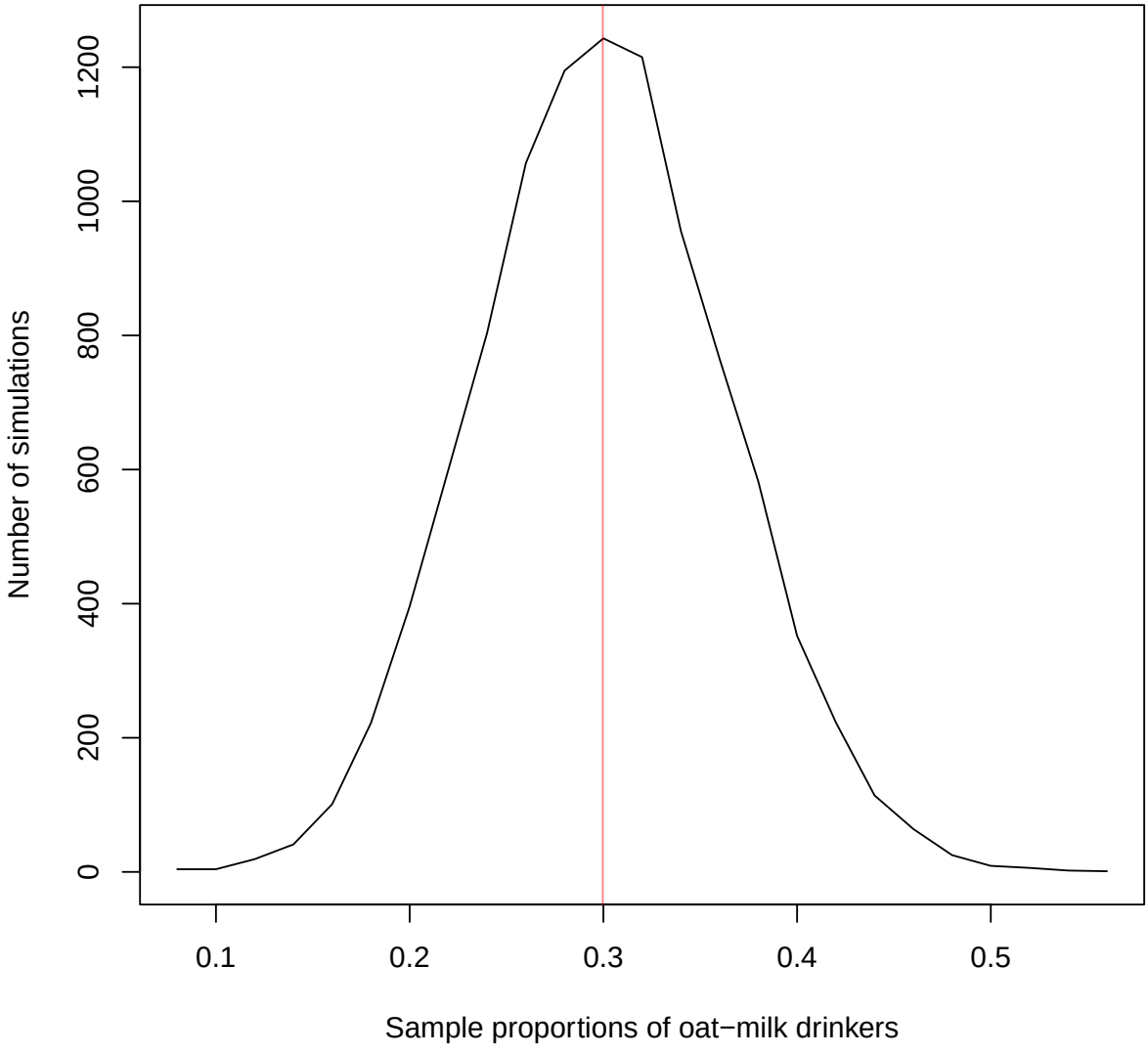
As we can see, there is quite some variability: While many (precisely, 9165) of our 10,000 samples produce between 10 and 20 oat-milk drinkers (with a mean of roughly 14.99), the numbers go as low as 4 and as high as 28.

This is indicative of the significant sampling variability we face when working with small samples.

A simulated sampling distribution

We can re-express the raw sample counts by calculating the *sample proportion* within each sample. This is also known as an *estimator* or *statistic* – a function that takes data as inputs to produce an aggregate output. The specific value this function produces is called an *estimate*.

Instead of plotting the number of the index of the sorted samples along the x-axis, we will now plot the **number of simulations** that produce a specific estimate.



The above is the **sampling distribution of the estimator** – in this case, the sampling distribution of the sample proportion for a sample of size 50 from the same underlying population.

Sampling distributions

You will recognize this distribution as *approximately normal*. This is due to the **central limit theorem**: With sufficiently large samples, the probability distribution of the sample mean (analogously, the sample proportion) across many such samples converges to a normal distribution.

Some facts about sampling distributions:

- Its mean is the true population mean.
- Its standard deviation gives the **standard error** of the estimator.
- Its 2.5 and 97.5 percentiles give the **95% confidence interval**. Since the sampling distribution is normal, the 95% confidence interval can be derived as mean $\pm$ 1.96 $\times$ std. err..
- The true sampling distribution of an estimator is **never known**. It can only be approximated by actually taking many, many independent samples.

The sampling distribution in practice

So, if the *true* sampling distribution is never known, and cannot be recovered from a single sample, what do we do?

We take the sample statistics of a singular sample at face value:

- Take the sample proportion as the best available estimate of the true population proportion: $\hat{p} = \frac{1}{N}\sum_{i=1}^{N} X_i$
- Take the standard error of the sample proportion as the best available estimate of the standard error of the estimator: $\hat{\sigma}_p = \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$
- Derive the 95% confidence interval accordingly (remember to use the correct t-value in place of the z-score!)

Example

The first of our 10,000 samples produces the following number of oat-milk drinkers: 9:

- The sample proportion is 0.18.
- The standard error of the sample proportion is 0.054.

- The corresponding 95% confidence interval is 0.071, 0.289.
- It does not contain the true population parameter.

What if we had ended up with our second sample instead?

- The number of oat-milk drinkers is 19:
- The sample proportion is 0.38.
- The standard error of the sample proportion is 0.079.
- The corresponding 95% confidence interval is 0.221, 0.539.
- It does contain the true population parameter.

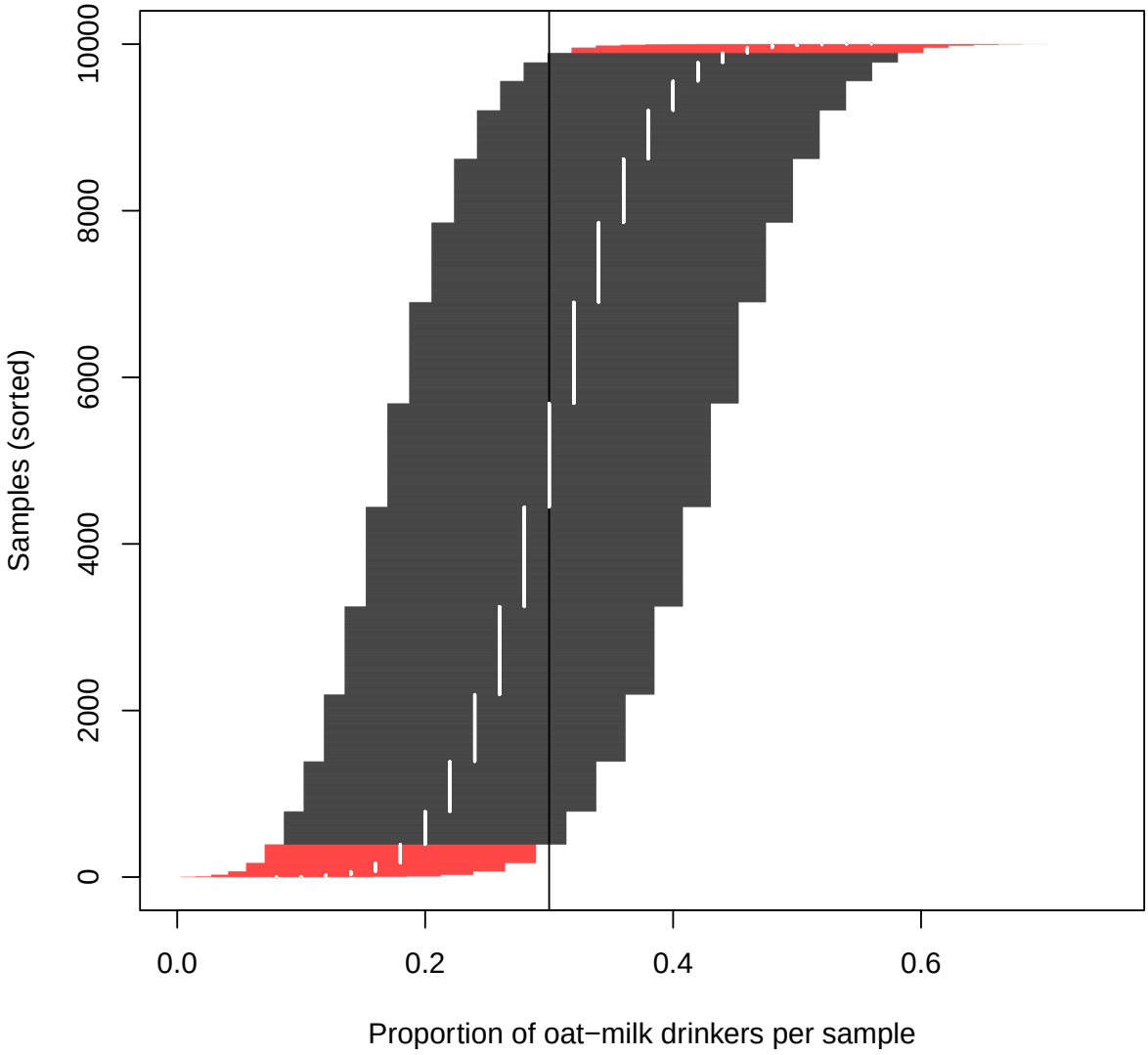
The correct interpretation of a 95% confidence interval

As we have just learned, whether a 95% confidence interval estimated from a given sample covers the true population parameter depends how (un)lucky we are with our sample.

A given 95% confidence interval does **not** contain the true parameter with 95% probability or "certainty". It either does or it doesn't, so the probability is either zero or one.

A 95% confidence interval *"represents the long-run proportion of CIs (at the given confidence level) that theoretically contain the true value of the parameter"*.

We can see this by looking back at our 10,000 samples:



The proportion of 95% CIs that do contain the true mean is 0.9502.

p-values and the logic of frequentist hypothesis testing

Similarly, *p*-values do *not* give the probability of support for an (alternative) hypothesis.

They give probability of finding other results at least as extreme as the present result, given a null hypothesis, as the long-run frequency across many samples.

Example: Let our hypothesis be that the proportion of oat-milk drinkers in the population is 0.5.

Remember our first sample had 9 oat milk drinkers, a proportion of 0.08.

To test our hypothesis based on this sample data, we first need a t-score:

```
t <- (p[1] - 0.5) / se[1]
t
```

```
## [1] -10.94701
```

We can the calculate the p-value for a two-tailed test as

```
2 * pt(-abs(t), df = n_obs - 1)
```

```
## [1] 9.172696e-15
```

This value is near zero. This shows that we would expect that hardly any other sample would produce a result as extreme as ours if the null hypothesis were true. We would therefore *reject* the null hypothesis.