Lecture 04 – The central limit theorem

ENVX1002 Statistics in Life and Environmental Sciences

Januar Harianto

The University of Sydney

Feb 2026

Learning Outcomes

At the end of this week, you will be able to:

1. Define probability distributions and their key properties
2. Apply the normal distribution to calculate probabilities
3. Standardise variables using the normal distribution
4. Differentiate between population, sample and sampling distributions
5. Explain the difference between standard deviation and standard error
6. Apply the central limit theorem and understand its significance
7. Use R to implement these statistical concepts

Quick checklist

- ☐ Create and edit Quarto documents
- ☐ Understand the concept of central tendency and spread/dispersion
- ☐ Interpret skewness (and kurtosis) in a distribution of data
- ☐ Know that different data types require different approaches to summarising data
- ☐ Generate some plots in `ggplot2`

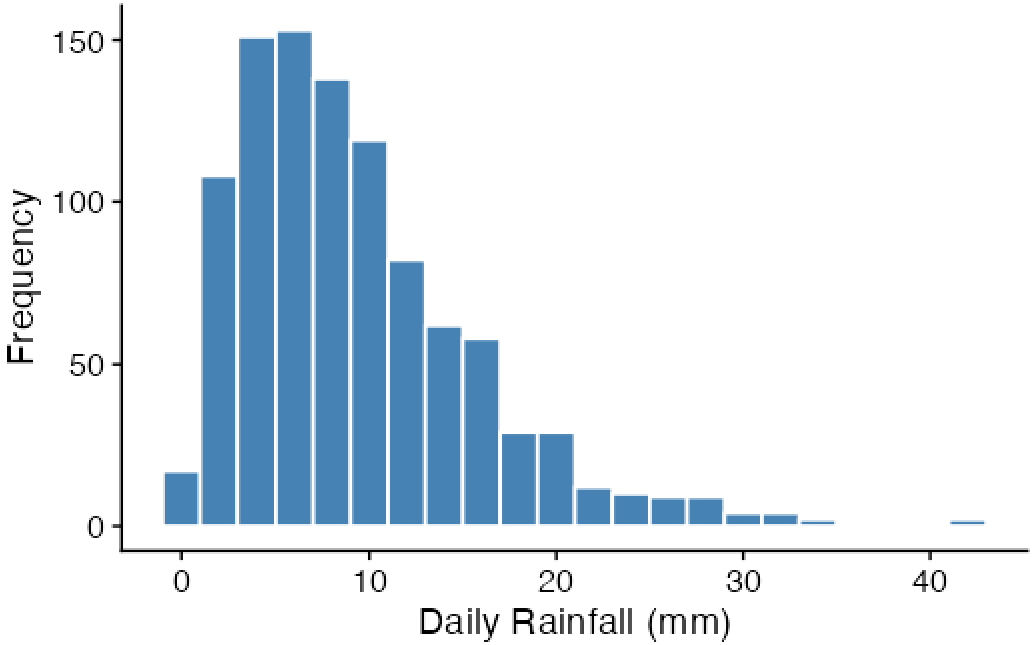
Probability distributions

Example: rainfall

Consider measuring daily rainfall amounts in a coastal wetland. *Some* days have no rain. **Most** days have small to moderate amounts. *Few* days have very heavy rainfall.

This creates a pattern of variation in rainfall amounts, which at some point we can predict. This pattern is what we call a **probability distribution** (i.e. we can calculate the probability of a certain amount of rainfall on a given day).

```r
set.seed(123)
rainfall_data <- data.frame(
  rainfall = rgamma(1000, shape = 2, scale = 5)
)

p_rainfall <- ggplot(rainfall_data, aes(x = rainfall)) +
  geom_histogram(binwidth = 2, fill = "steelblue", colour = "white") +
  labs(
    x = "Daily Rainfall (mm)",
    y = "Frequency"
  )
p_rainfall
```


150
100
50
0
Frequency
0
10
20
30
40
Daily Rainfall (mm)

Probability distribution?

A probability distribution is a function or equation that tells us:

- How likely different values are to occur
- The pattern of variation in our measurements
- The range and spread of possible values

It is like a mathematical recipe for how values are distributed.

Different types of distributions

Depending on the data, distributions can be:

- Discrete (e.g. number of trees in 100m² of forest) – probability **distribution** function $P = f(x)$
  - What is the probability that 100m² of forest contains 50 trees?
- Continuous (e.g. rainfall amounts over a period) – probability **density** function $f(x)$
  - What is the probability of rainfall between 10-20mm a day?
- Both discrete and continuous (e.g. age of trees) $F(x) = P(X \leq x)$
  - What is the probability that a tree is older than 50 years?

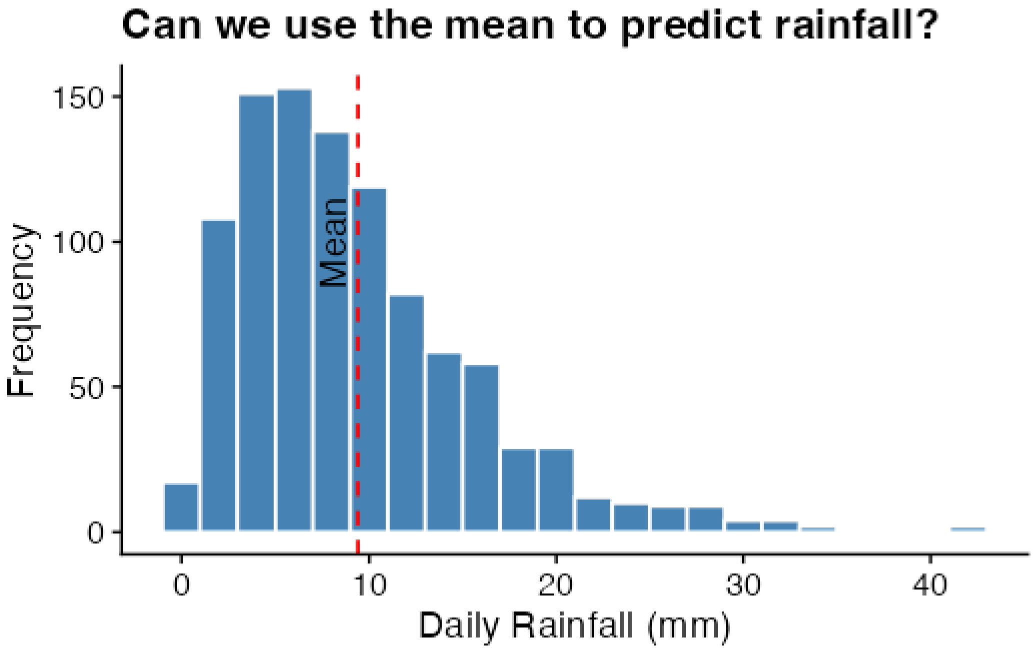
Properties of distributions

We now combine all of what we have learnt about data in the last 3 weeks:

- **Central tendency** – mean, median, mode
- **Spread** – standard deviation, variance
- **Shape** – symmetry, skewness
- **Tails** – how quickly they “decay” towards zero

But how do we use these properties to make predictions? **Do some distributions have special properties?**

```r
# Annotate our rainfall distribution with key properties
p_rainfall +
  geom_vline(
    xintercept = mean(rainfall_data$rainfall),
    colour = "red", linetype = "dashed"
  ) +
  annotate("text",
    x = mean(rainfall_data$rainfall), y = 100,
    label = "Mean", angle = 90, vjust = -0.5
  ) +
  labs(
    x = "Daily Rainfall (mm)",
    y = "Frequency",
    title = "Can we use the mean to predict rainfall?"
  )
```



The general normal distribution

Tree heights

Tree heights in a forest often follow a normal distribution:

- Most trees cluster around the average height
- Fewer trees are very short or very tall
- The pattern is symmetrical – a bell-shaped curve with two parameters (mean μ and standard deviation σ) such that

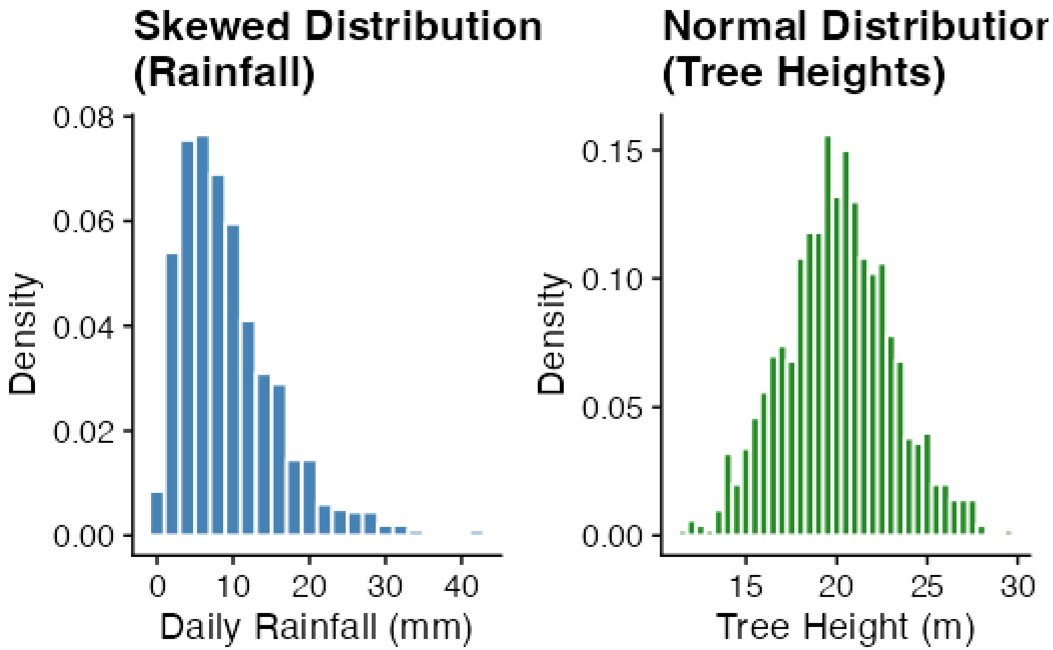
$$X \sim N(\mu, \sigma^2)$$

Comparing distributions

```r
# Set up the data
set.seed(123)
tree_heights <- data.frame(
  height = rnorm(1000, mean = 20, sd = 3)
)

# Create plots for comparison
p1 <- ggplot(rainfall_data, aes(x = rainfall)) +
  geom_histogram(aes(y = ..density..),
    binwidth = 2,
    fill = "steelblue",
    colour = "white"
  ) +
  labs(
    x = "Daily Rainfall (mm)",
    y = "Density",
    title = "Skewed Distribution\n(Rainfall)"
  )
```

```r
p2 <- ggplot(tree_heights, aes(x = height)) +
  geom_histogram(aes(y = ..density..),
    binwidth = 0.5,
    fill = "forestgreen",
    colour = "white"
  ) +
  labs(
    x = "Tree Height (m)",
    y = "Density",
    title = "Normal Distribution\n(Tree Heights)"
  )

# Arrange plots side by side
plot_grid(p1, p2, ncol = 2)
```

```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
ℹ Please use `after_stat(density)` instead.
```

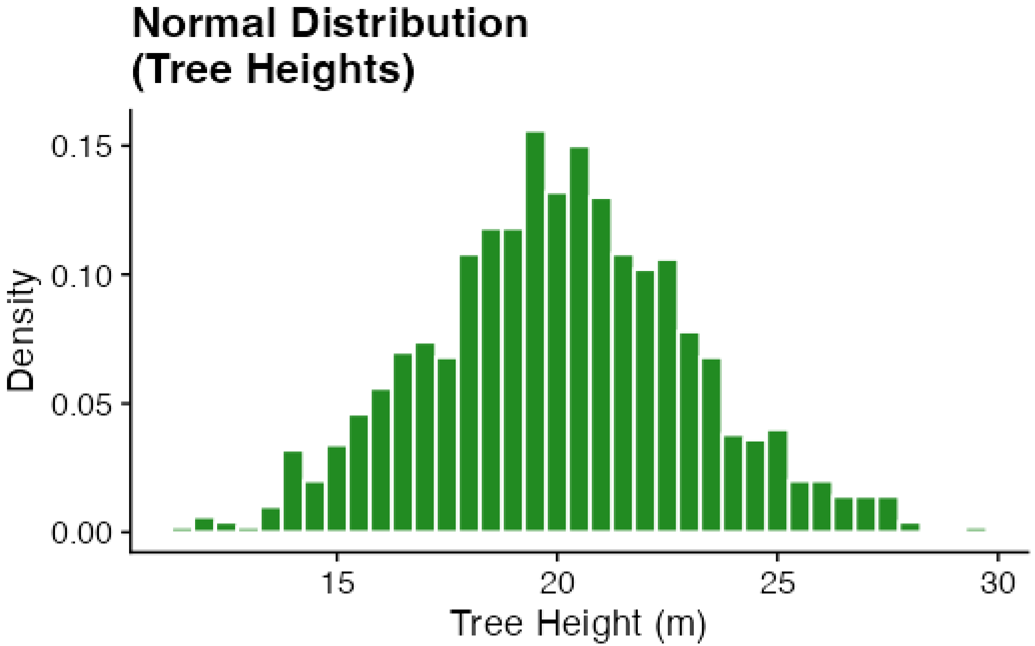

Skewed Distribution
(Rainfall)
Density
0.08
0.06
0.04
0.02
0.00
0
10
20
30
40
Daily Rainfall (mm)
Normal Distributior
(Tree Heights)
Density
0.15
0.10
0.05
0.00
15
20
25
30
Tree Height (m)

Why do tree heights follow a normal distribution?

1. The middle (mean) height is most common – most trees are around 20 metres tall
2. Heights spread out evenly on both sides – as many short trees as tall trees, like a mirror image around the middle
3. Extreme heights are rare – very few extremely short or tall trees – numbers decrease gradually as we move away from the middle

The normal distribution is a natural phenomenon in many real-world situations – a **fundamental** concept in statistics.

```
p2
```


Normal Distribution
(Tree Heights)
Density
0.15
0.10
0.05
0.00
15
20
25
30
Tree Height (m)

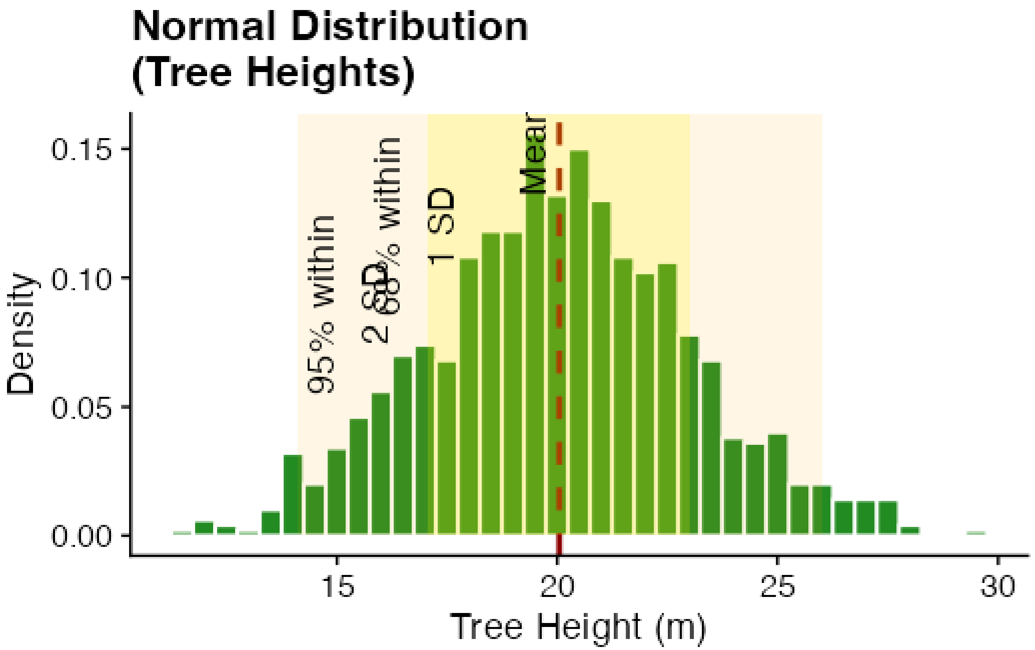
A continuous distribution

- There is an infinite number of possible heights within a certain range (e.g. 0-40m)
- We can calculate the probability of a tree being a certain height or within a range
- Some heights have a probability of zero (e.g. negative heights, or heights above 40m)

We can use these properties to make reasonable predictions about tree heights:

```r
# Calculate mean and standard deviations
mean_height <- mean(tree_heights$height)
sd_height <- sd(tree_heights$height)

# Create plot with standard deviation ranges
p2 +
  # Add vertical lines for mean and SDs
  geom_vline(
    xintercept = mean_height,
    colour = "darkred",
    size = 1,
    linetype = "dashed"
  ) +
# Add SD ranges
annotate("rect",
  xmin = mean_height - sd_height,
  xmax = mean_height + sd_height,
  ymin = 0, ymax = Inf,
  fill = "yellow", alpha = 0.2
) +
annotate("rect",
  xmin = mean_height - 2 * sd_height,
  xmax = mean_height + 2 * sd_height,
  ymin = 0, ymax = Inf,
  fill = "orange", alpha = 0.1
) +
# Add labels
annotate("text",
  x = mean_height, y = 0.15,
  label = "Mean", angle = 90, vjust = -0.5
) +
annotate("text",
  x = mean_height - sd_height/2, y = 0.12,
  label = "68% within\n1 SD", angle = 90, vjust = -0.5
) +
annotate("text",
  x = mean_height - sd_height, y = 0.09,
  label = "95% within\n2 SD", angle = 90, vjust = -0.5
) +
labs(
  x = "Tree Height (m)",
  y = "Density",
  title = "Normal Distribution\n(Tree Heights)"
)
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
ℹ Please use `linewidth` instead.
```



Normal Distribution
(Tree Heights)

The 68-95-99.7 rule

A data that follows a normal distribution:

- About 68% of values fall within 1 standard deviation
- About 95% fall within 2 standard deviations
- About 99.7% fall within 3 standard deviations
- The remaining 0.3% are extreme values and extend to infinity

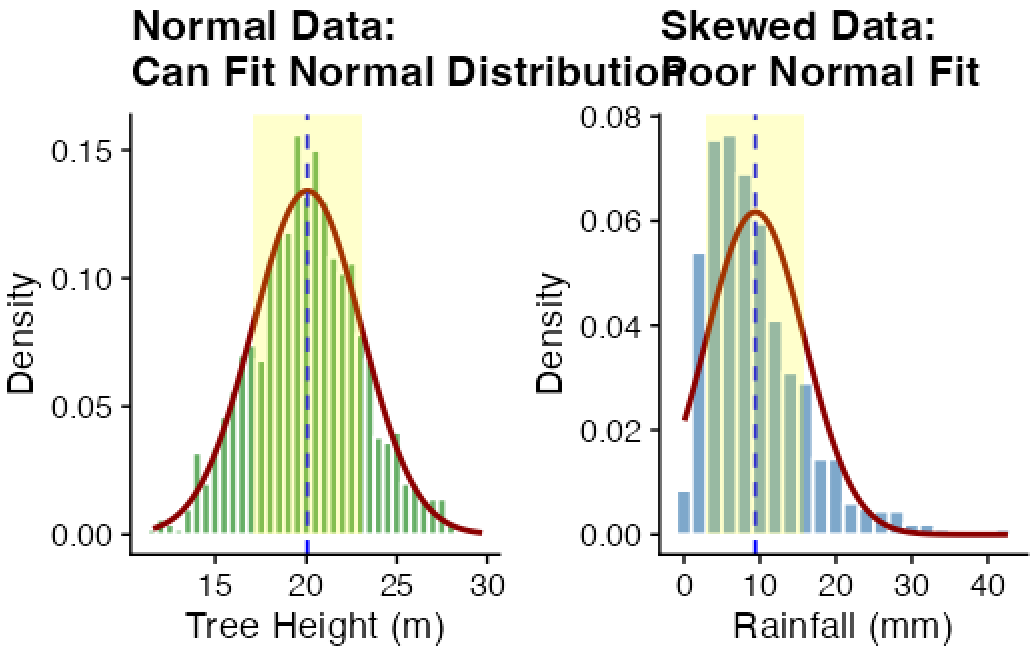
This pattern is the same for ALL normal distributions – a **universal** rule.

Importantly, it is a **predictable** pattern (although not exact).

From data to normal distribution

- When data follows a reasonably normal pattern, we can “fit” a normal distribution to make *equally* reasonable predictions.
- It is possible to fit a normal distribution to a skewed distribution (but **not recommended** – other techniques exist)

```r
# Calculate statistics for both distributions
tree_mean <- mean(tree_heights$height)
tree_sd <- sd(tree_heights$height)
rain_mean <- mean(rainfall_data$rainfall)
rain_sd <- sd(rainfall_data$rainfall)

# Create comparison plots showing fitted normal curves
p1 <- ggplot(tree_heights, aes(x = height)) +
  geom_histogram(aes(y = ..density..),
    binwidth = 0.5,
    fill = "forestgreen",
    colour = "white",
    alpha = 0.7
  ) +
  # Add fitted normal curve
  stat_function(
    fun = dnorm,
    args = list(mean = tree_mean, sd = tree_sd),
    colour = "darkred",
    size = 1
  ) +
  # Add mean line and 1 SD range
  geom_vline(xintercept = tree_mean, colour = "blue", linetype = "dashed") +
  annotate("rect",
    xmin = tree_mean - tree_sd,
    xmax = tree_mean + tree_sd,
    ymin = 0, ymax = Inf,
    fill = "yellow", alpha = 0.2
  ) +
  labs(
    x = "Tree Height (m)",
    y = "Density",
    title = "Normal Data:\nCan Fit Normal Distribution"
  )

p2 <- ggplot(rainfall_data, aes(x = rainfall)) +
  geom_histogram(aes(y = ..density..),
    binwidth = 2,
    fill = "steelblue",
    colour = "white",
    alpha = 0.7
  ) +
  # Try to fit normal curve (shows poor fit)
  stat_function(
    fun = dnorm,
    args = list(mean = rain_mean, sd = rain_sd),
    colour = "darkred",
    size = 1
  ) +
  geom_vline(xintercept = rain_mean, colour = "blue", linetype = "dashed") +
  annotate("rect",
    xmin = rain_mean - rain_sd,
    xmax = rain_mean + rain_sd,
    ymin = 0, ymax = Inf,
    fill = "yellow", alpha = 0.2
  ) +
  labs(
    x = "Rainfall (mm)",
    y = "Density",
    title = "Skewed Data:\nPoor Normal Fit"
  )

plot_grid(p1, p2, ncol = 2)
```



When data is normal…

1. We can use normal probability functions to:
   - Calculate exact probabilities
   - Find specific percentiles
   - Make reliable predictions
2. Properties are predictable (the 68-95-99.7 rule)
3. Can standardise measurements (**more on this later**)
   - Compare different variables
   - Use z-scores
   - **Apply statistical tests**

Working with normal distributions in R

R provides three main functions that help us work with normal distributions. Think of them as different ways to ask questions about our data:

1. `pnorm()`: "What's the probability?"

- Like asking "What proportion of trees are shorter than X meters?"
  - Probability of a tree being under 15m tall
  - Chance of rainfall being less than 50mm
  - Proportion of measurements below average

Working with normal distributions in R

R provides three main functions that help us work with normal distributions. Think of them as different ways to ask questions about our data:

2. `qnorm()`: "What's the value?"

- Like asking "How tall are the shortest 10% of trees?"
  - Finding the height that only 5% of trees exceed
  - Determining rainfall threshold for "extreme" events
  - Identifying typical range boundaries

Working with normal distributions in R

R provides three main functions that help us work with normal distributions. Think of them as different ways to ask questions about our data:

3. `dnorm()`: "What's the relative likelihood?"

- Like asking "How common is this exact height?"
  - Comparing how likely different heights are
  - Finding peak probability
  - Plotting the normal curve shape

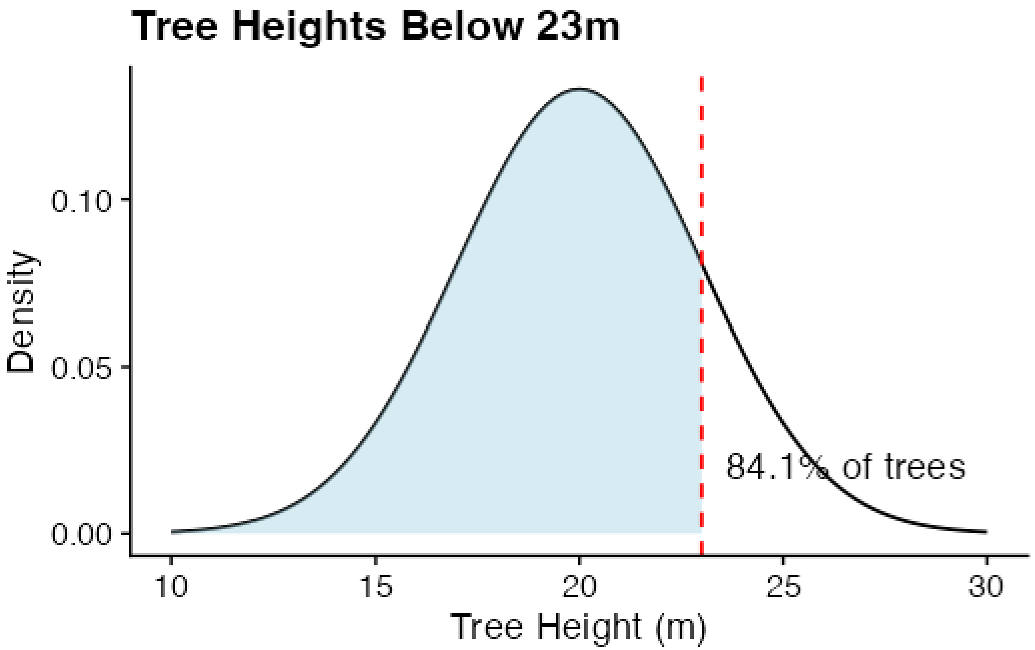
Using `pnorm()` for probabilities

What proportion of trees in our forest are shorter than 23m, knowing that mean height is 20m and the sd is 3m?

```r
prob_under_23 <- pnorm(q = 23, mean = 20, sd = 3) # Calculate proportion of trees under 23m
prob_under_23
```

```
[1] 0.8413447
```

```r
# Visualise the probability
percent_under_23 <- prob_under_23 * 100

ggplot(data.frame(x = c(10, 30)), aes(x = x)) +
  stat_function(fun = dnorm, args = list(mean = 20, sd = 3)) +
  geom_area(
    stat = "function", fun = dnorm, args = list(mean = 20, sd = 3),
    fill = "lightblue", alpha = 0.5, xlim = c(10, 23)
  ) +
  geom_vline(xintercept = 23, linetype = "dashed", colour = "red") +
  annotate("text",
    x = 23, y = 0.02,
    label = sprintf("%.1f%% of trees", percent_under_23),
    hjust = -0.1
  ) +
  labs(
    x = "Tree Height (m)", y = "Density",
    title = "Tree Heights Below 23m"
  )
```


Tree Heights Below 23m
Density
0.10
0.05
0.00
84.1% of trees
10
15
20
25
30
Tree Height (m)

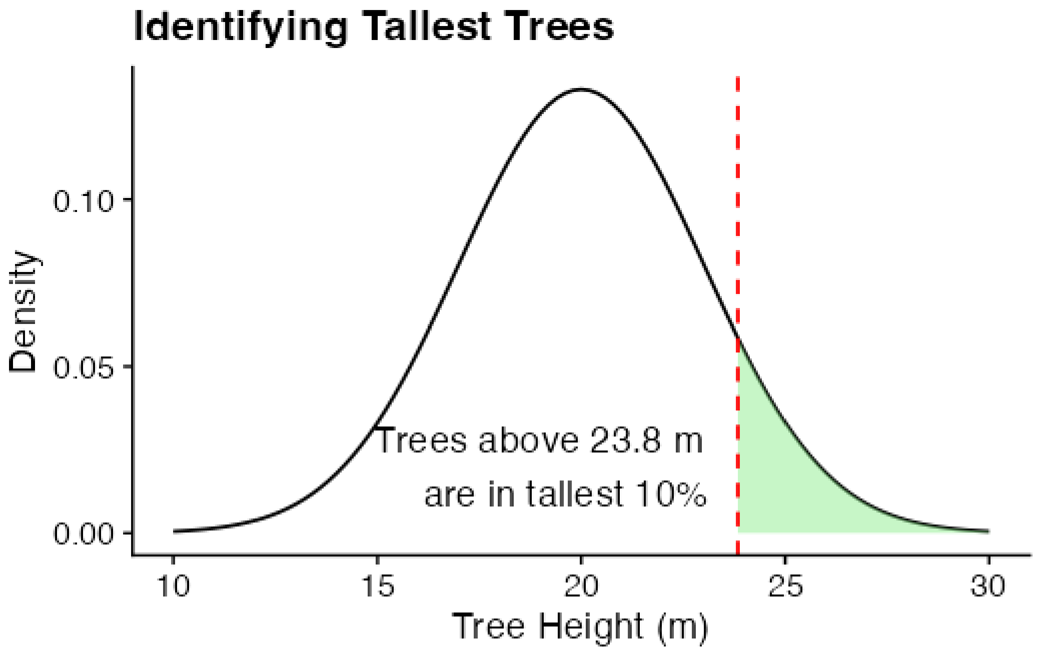
Using `qnorm()` for thresholds

How tall are the largest 10% of trees in our forest?

```r
# Find the height threshold for tallest 10%
height_90th <- qnorm(p = 0.90, mean = 20, sd = 3)
height_90th
```

```
[1] 23.84465
```

```r
# Visualise the threshold
ggplot(data.frame(x = c(10, 30)), aes(x = x)) +
  stat_function(fun = dnorm, args = list(mean = 20, sd = 3)) +
  geom_area(stat = "function", fun = dnorm, args = list(mean = 20, sd = 3),
            fill = "lightgreen", alpha = 0.5, xlim = c(height_90th, 30)) +
  geom_vline(xintercept = height_90th, linetype = "dashed", colour = "red") +
  annotate("text", x = height_90th, y = 0.02,
           label = sprintf("Trees above %.1f m\nare in tallest 10%%", height_90th),
           hjust = 1.1) +
  labs(x = "Tree Height (m)", y = "Density",
       title = "Identifying Tallest Trees")
```



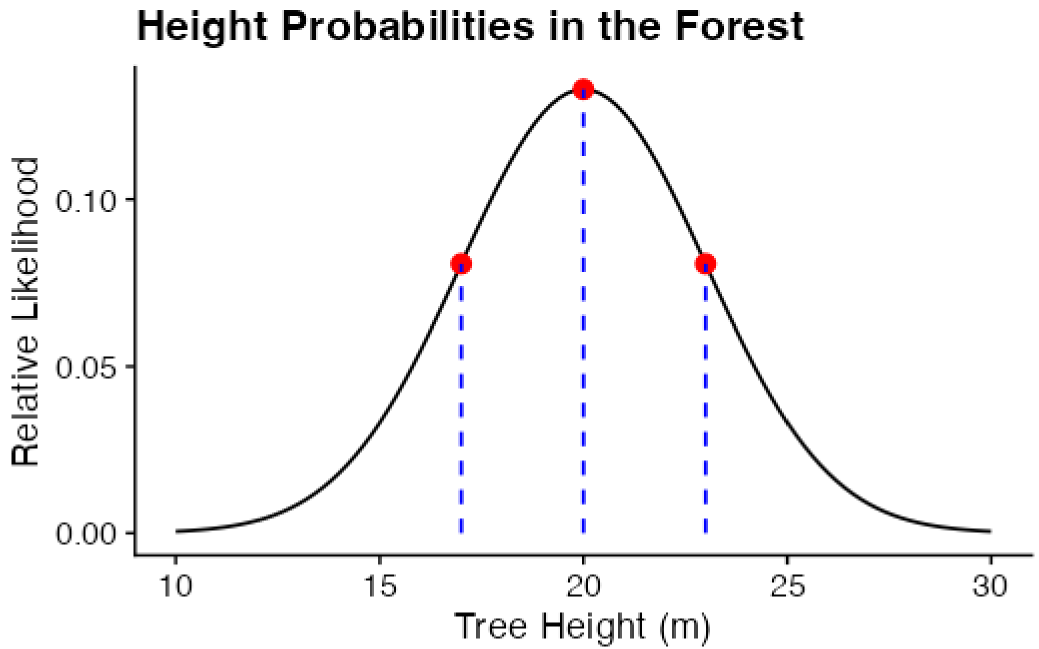
Using `dnorm()` for relative likelihood

Which tree height is most common in our forest: 17m, 20m, or 23m? (Or, how likely are these heights?)

```r
# Find relative likelihood at different heights
heights <- c(17, 20, 23) # Example heights to compare
densities <- dnorm(heights, mean = 20, sd = 3)
densities
```

```
[1] 0.08065691 0.13298076 0.08065691
```

```r
# Plot relative likelihoods
ggplot(data.frame(x = c(10, 30)), aes(x = x)) +
  stat_function(fun = dnorm, args = list(mean = 20, sd = 3)) +
  geom_point(data = data.frame(x = heights, y = densities),
             aes(y = y), colour = "red", size = 3) +
  geom_segment(data = data.frame(x = heights),
              aes(x = x, xend = x, y = 0, yend = dnorm(x, 20, 3)),
              linetype = "dashed", colour = "blue") +
  labs(x = "Tree Height (m)", y = "Relative Likelihood",
       title = "Height Probabilities in the Forest")
```



Trees closest to 20m are most common, with likelihood decreasing as we move away.

Practical uses of normal distribution functions

These functions help us answer questions like:

- What proportion of trees are between 18m and 22m tall?

```r
# Probability between 18m and 22m - calculate the difference
prob_between <- pnorm(22, 20, 3) - pnorm(18, 20, 3)
prob_between
```

```
[1] 0.4950149
```

- How tall are the top 5% of trees?

```r
# Height of tallest 5% of trees
tall_threshold <- qnorm(0.95, 20, 3)
tall_threshold
```

```
[1] 24.93456
```

- What are the height thresholds for “extreme” trees (top 1% and bottom 1%)?

```r
# Height thresholds for "extreme" trees
extreme_thresholds <- qnorm(c(0.01, 0.99), 20, 3)
extreme_thresholds
```

```
[1] 13.02096 26.97904
```

The standard normal curve

Standardising variables

We can convert any normal distribution to a standard form where:

$$X \sim N(\mu, \sigma^2) \to Z \sim N(0, 1)$$

Why standardise?

Standardising helps us:

1. Compare measurements on different scales
2. Identify unusual values (e.g., z > 2 is unusual)
3. Calculate probabilities using standard normal tables

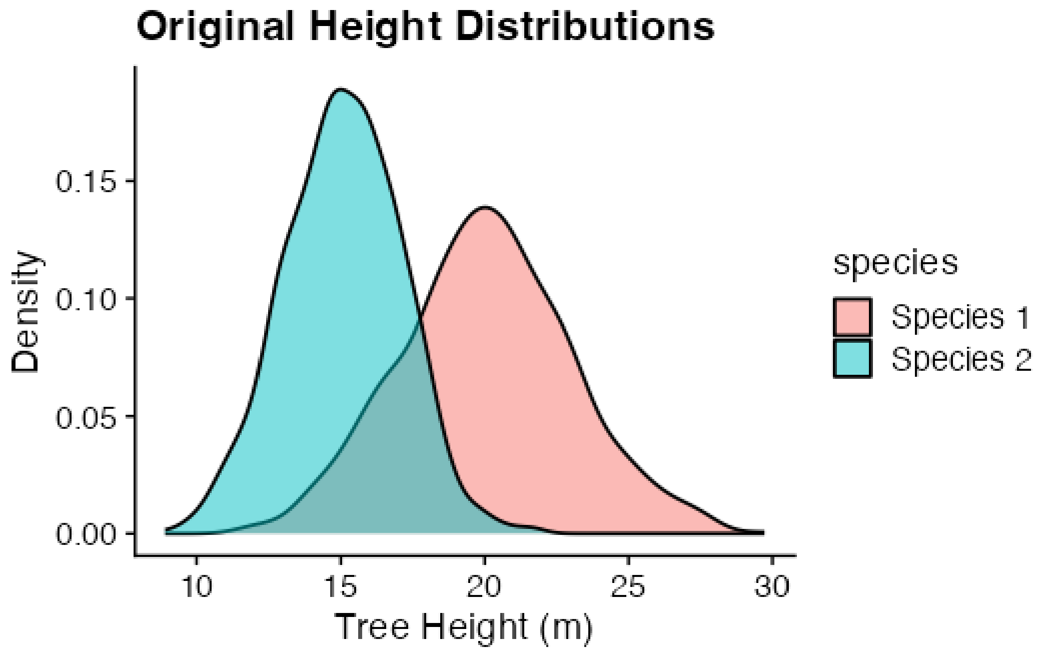
Formula:

$$z = \frac{x - \mu}{\sigma}$$

(subtract mean, divide by SD)

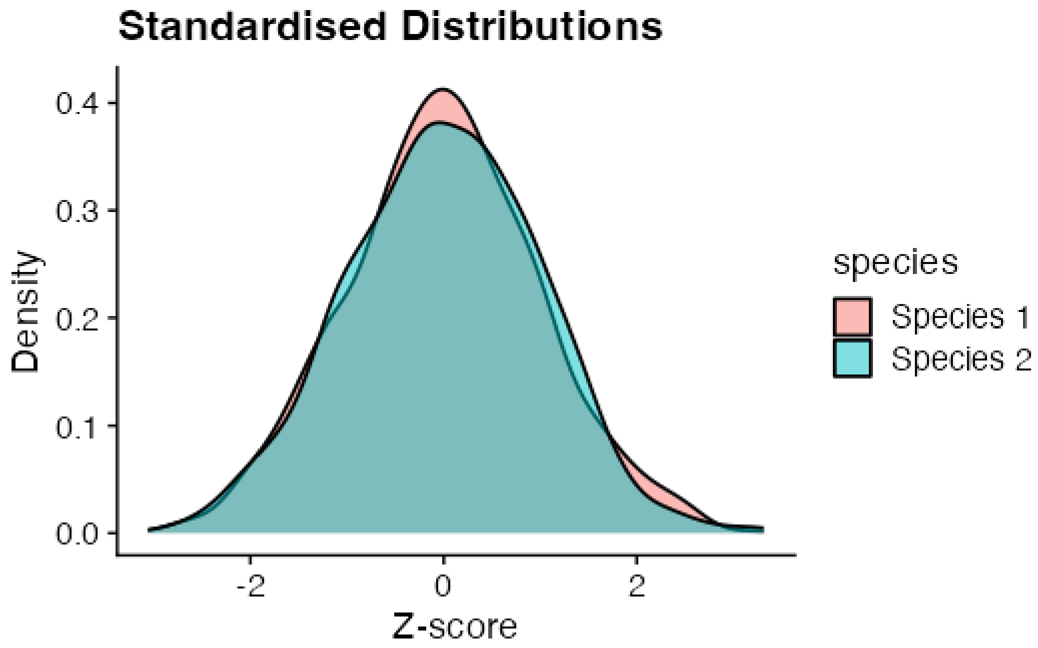
Comparing different species

```r
# Create data for two tree species
set.seed(123)
species1 <- rnorm(1000, mean = 20, sd = 3)  # Tall species
species2 <- rnorm(1000, mean = 15, sd = 2)  # Shorter species

# Plot original heights
data.frame(
  height = c(species1, species2),
  species = rep(c("Species 1", "Species 2"), each = 1000)
) |>
  ggplot(aes(x = height, fill = species)) +
  geom_density(alpha = 0.5) +
  labs(x = "Tree Height (m)", y = "Density",
       title = "Original Height Distributions")
```


Original Height Distributions
Density
0.15
0.10
0.05
0.00
10
15
20
25
30
Tree Height (m)
species
🟥 Species 1
🟦 Species 2

After standardisation

```r
# Convert to z-scores
z1 <- scale(species1)
z2 <- scale(species2)

# Plot standardised distributions
data.frame(
  zscore = c(z1, z2),
  species = rep(c("Species 1", "Species 2"), each = 1000)
) |>
  ggplot(aes(x = zscore, fill = species)) +
  geom_density(alpha = 0.5) +
  labs(x = "Z-score", y = "Density",
       title = "Standardised Distributions")
```


Standardised Distributions
Density
0.4
0.3
0.2
0.1
0.0
-2
0
2
Z-score
species
🟥 Species 1
🟦 Species 2

Example

A z-score of 2 means:

- Species 1: trees above 26 m
- Species 2: trees above 19 m
- Both are equally tall for their species

```r
# Calculate z-scores for specific heights
z_26 <- (26 - 20) / 3
z_26
```

```
[1] 2
```

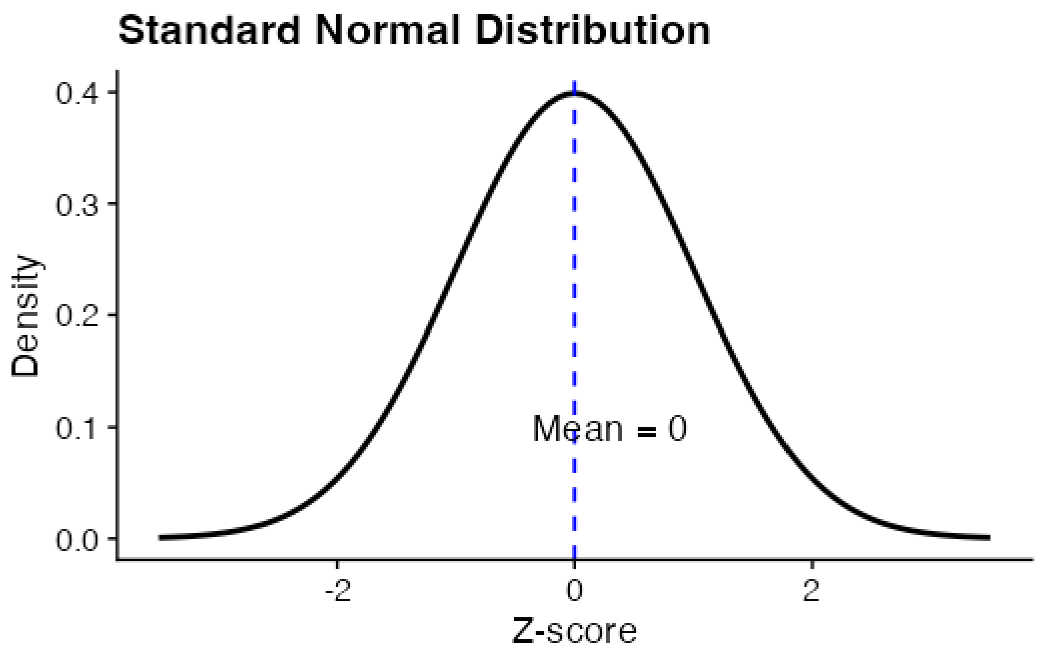
```r
z_19 <- (19 - 15) / 2
z_19
```

```
[1] 2
```

Working with standardised data: The standard normal curve

Once we standardise our data to z-scores, we can calculate probabilities without knowing the original mean or standard deviation.

```r
# Create a visual of the standard normal curve
ggplot(data.frame(x = c(-3.5, 3.5)), aes(x = x)) +
  stat_function(fun = dnorm, size = 1, colour = "black") +
  geom_vline(xintercept = 0, linetype = "dashed", colour = "blue") +
  annotate("text", x = 0.3, y = 0.1, label = "Mean = 0") +
  labs(x = "Z-score", y = "Density",
       title = "Standard Normal Distribution")
```


Standard Normal Distribution
Density
0.4
0.3
0.2
0.1
0.0
Mean = 0
-2
0
2
Z-score

Answering questions with standardised data

1. `pnorm()`: What proportion of values fall below a given z-score?
2. `qnorm()`: What z-score corresponds to a specific percentile?
3. `dnorm()`: How likely is a specific z-score to occur?

Let’s see how these help us understand our tree height data:

```
# Example: For a tree with z-score = 1.5 (taller than average)
# What percentage of trees are shorter than this tree?
pnorm(1.5)
```

```
[1] 0.9331928
```

```
# Example: How tall must a tree be to be in the tallest 5%?
# (Find the z-score at the 95th percentile)
qnorm(0.95)
```

```
[1] 1.644854
```

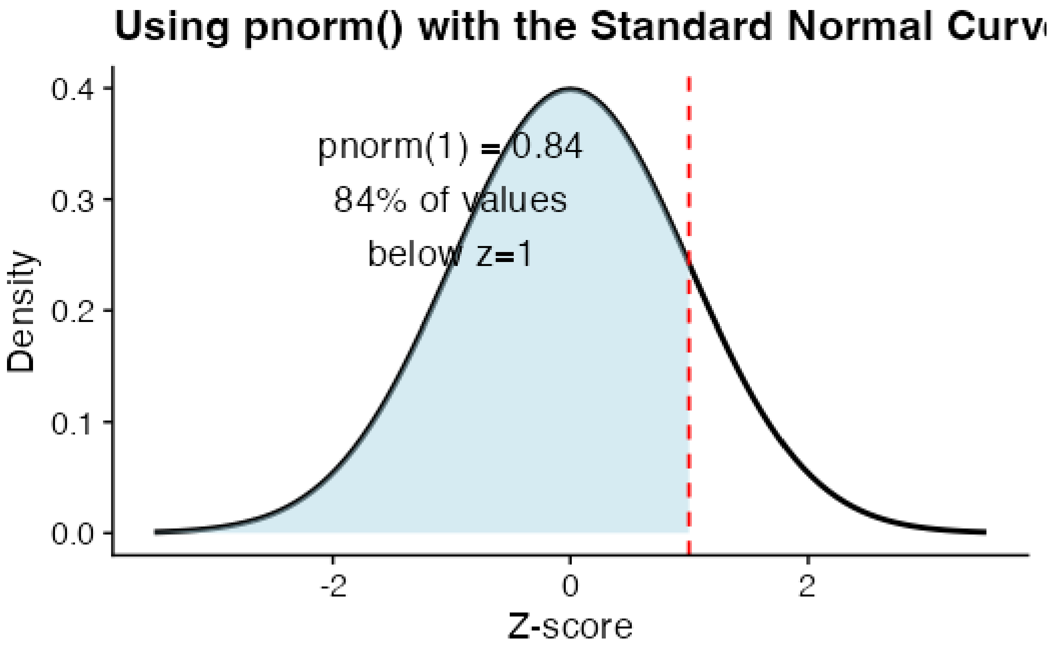
```
# Example: Compare likelihood of average height vs. very tall trees
dnorm(0)/dnorm(2)  # How much more common is z=0 compared to z=2
```

```
[1] 7.389056
```

Visualising standard normal functions

The blue shaded area shows `pnorm(1)` = 0.84, meaning 84% of trees have a z-score less than 1.

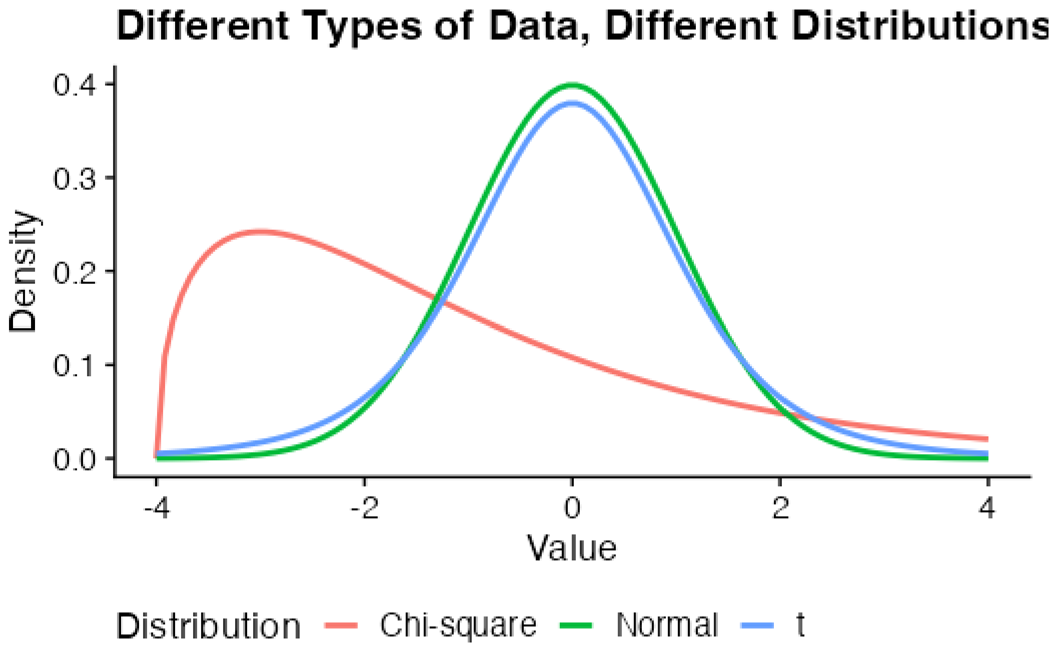
```r
# Create a visual showing pnorm in action
ggplot(data.frame(x = c(-3.5, 3.5)), aes(x = x)) +
  stat_function(fun = dnorm, size = 1, colour = "black") +
  geom_area(stat = "function", fun = dnorm,
            fill = "lightblue", alpha = 0.5, xlim = c(-3.5, 1)) +
  geom_vline(xintercept = 1, linetype = "dashed", colour = "red") +
  annotate("text", x = -1, y = 0.3,
           label = "pnorm(1) = 0.84\n84% of values\nbelow z=1") +
  labs(x = "Z-score", y = "Density",
       title = "Using pnorm() with the Standard Normal Curve")
```


Using pnorm() with the Standard Normal Curv
pnorm(1) = 0.84
84% of values
below z=1
Density
0.4
0.3
0.2
0.1
0.0
-2
0
2
Z-score

Beyond the normal distribution

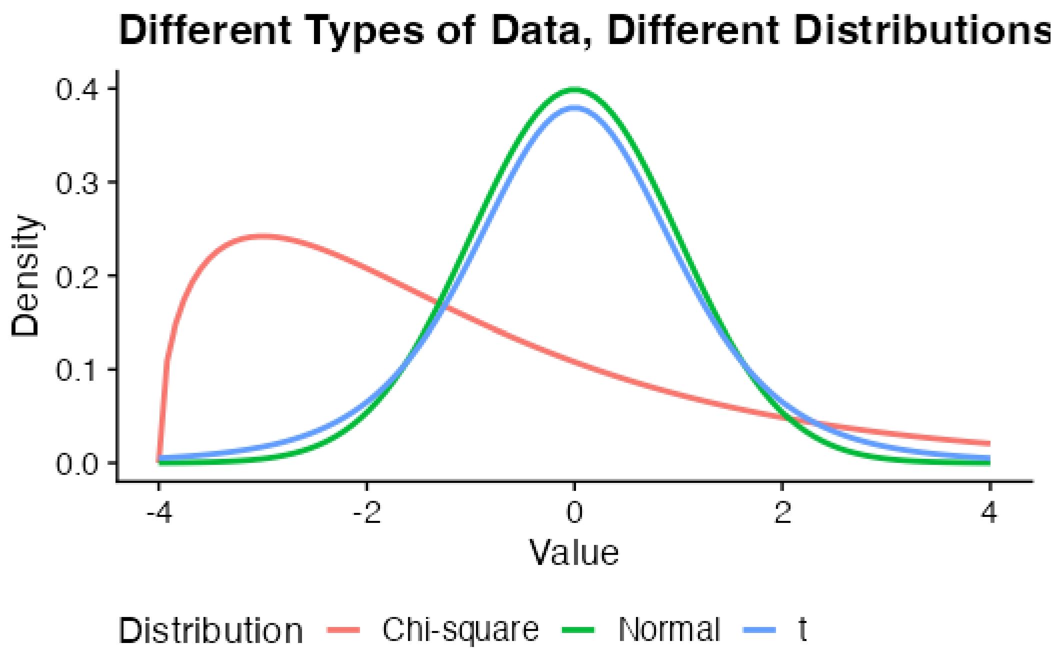
Different data types need different distributions:

```r
# Create sequence for x-axis
x ← seq(-4, 4, length = 100)

# Create data frame for plotting
distributions ← data.frame(
  x = rep(x, 3),
  density = c(
    dt(x, df = 5),        # t-distribution
    dchisq(x + 4, df = 3),# chi-square (shifted)
    dnorm(x)              # normal for comparison
  ),
  Distribution = rep(c("t", "Chi-square", "Normal"), each = 100)
)

# Plot distributions
ggplot(distributions, aes(x = x, y = density, colour = Distribution)) +
  geom_line(size = 1) +
  labs(x = "Value", y = "Density",
       title = "Different Types of Data, Different Distributions") +
  theme(legend.position = "bottom")
```



Beyond the normal distribution

```r
# Plot distributions
ggplot(distributions, aes(x = x, y = density, colour = Distribution)) +
  geom_line(size = 1) +
  labs(
    x = "Value", y = "Density",
    title = "Different Types of Data, Different Distributions"
  ) +
  theme(legend.position = "bottom")
```



We use these for:

- Comparing means of small samples (t-distribution)
- Analysing counts and proportions (Chi-square)
- Testing other hypotheses (many more distributions)

We'll learn when to use each one as we need them.

Sampling and sampling distributions

What is a sample?

A sample is a subset of a population used to represent the entire group. It allows us to make inferences about the population without examining every member.

- **Population**: all possible measurements
- **Sample**: a subset of the population
- **Sample size**: number of measurements in the sample
- **Sampling distribution**: the distribution of sample means from multiple samples

Sampling distribution

A sampling distribution tells us:

- How sample statistics (like means) vary
- What patterns emerge when we take many samples
- Why we get different results with different samples

Think of it as **distribution of sample statistics**, not individual measurements. **Your variable and all its values are a** *single* **number in this distribution.**

Taking samples from our rainfall data

Let’s take samples from our skewed rainfall data and see what happens:

```r
# Our population: all rainfall measurements (simulated)
set.seed(123)
rainfall_population <- rainfall_data$rainfall

# Take a small sample of 5 days
small_sample <- sample(rainfall_population, size = 5)

# Calculate the sample mean
mean(small_sample)
```

```
[1] 6.49786
```

If we take another sample, we get a different mean:

```r
# Take another sample
another_sample <- sample(rainfall_population, size = 5)
mean(another_sample)
```

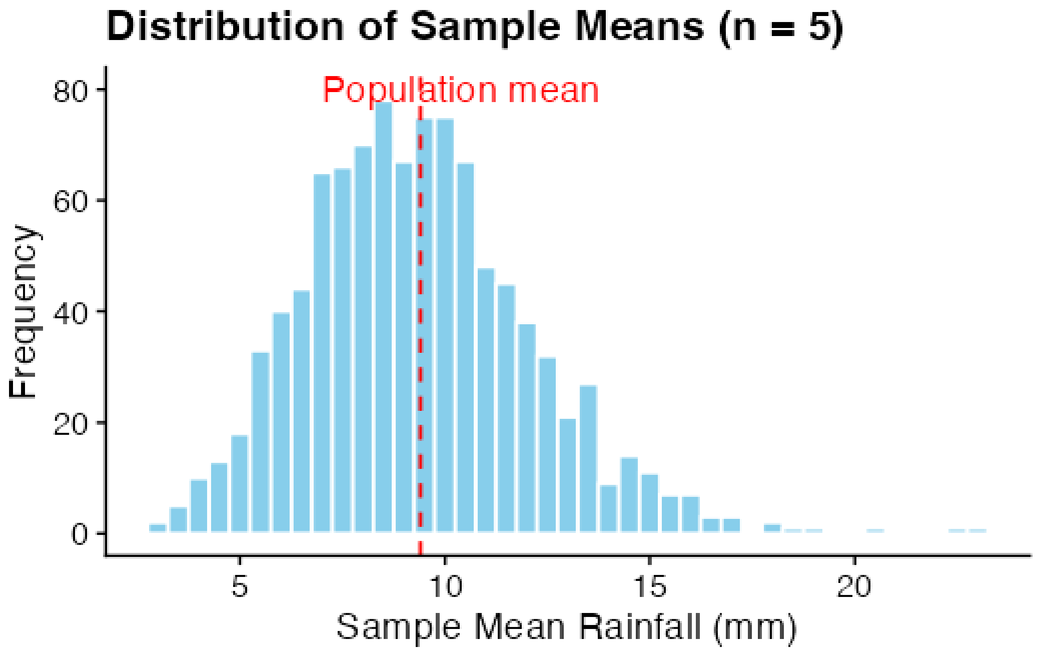
```
[1] 6.656836
```

Each sample gives us a slightly different estimate of the true mean rainfall.

Distribution of sample means for rainfall

What happens if we take many samples and look at all their means?

```r
# Take 1000 samples of size 5 and calculate their means
set.seed(456)
rain_means_n5 <- replicate(1000, mean(sample(rainfall_population, size = 5)))

# Plot the distribution of sample means
ggplot(data.frame(sample_mean = rain_means_n5), aes(x = sample_mean)) +
  geom_histogram(binwidth = 0.5, fill = "skyblue", colour = "white") +
  geom_vline(xintercept = mean(rainfall_population),
             colour = "red", linetype = "dashed") +
  annotate("text", x = mean(rainfall_population) + 1, y = 80,
           label = "Population mean", colour = "red") +
  labs(x = "Sample Mean Rainfall (mm)", y = "Frequency",
       title = "Distribution of Sample Means (n = 5)")
```



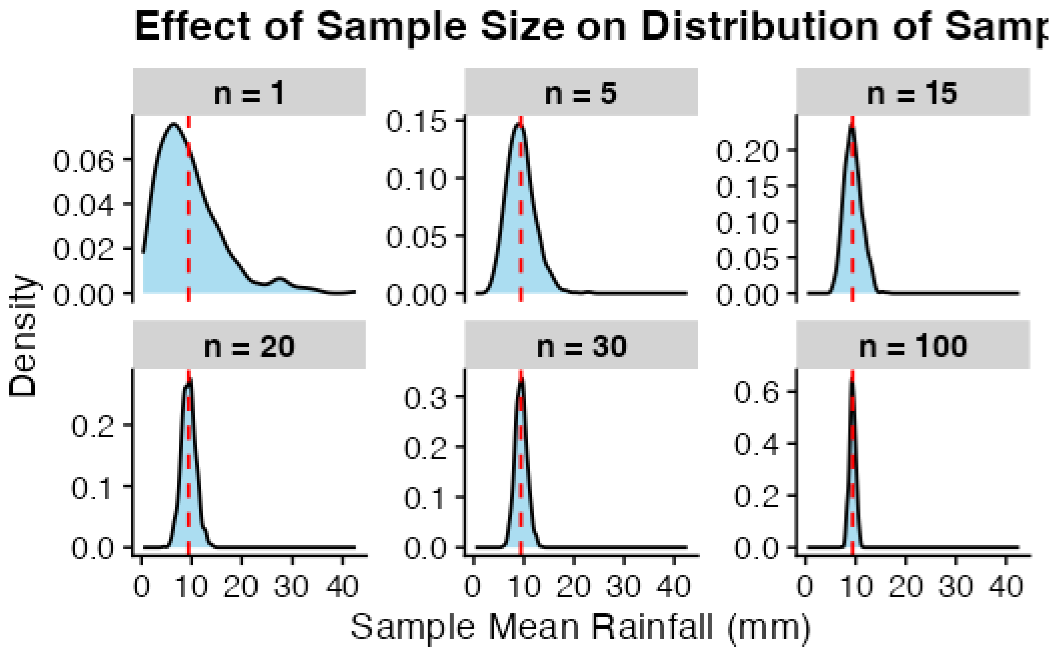
Notice how the sample means are distributed around the true population mean, but the distribution is still somewhat skewed.

Effect of sample size on rainfall means

What happens when we increase our sample size?

```r
# Take 1000 samples of different sizes
set.seed(404)
rain_means_n1 <- replicate(1000, mean(sample(rainfall_population, size = 1)))
# We already have n5 from previous slide
rain_means_n15 <- replicate(1000, mean(sample(rainfall_population, size = 15)))
rain_means_n20 <- replicate(1000, mean(sample(rainfall_population, size = 20)))
rain_means_n30 <- replicate(1000, mean(sample(rainfall_population, size = 30)))
rain_means_n100 <- replicate(1000, mean(sample(rainfall_population, size = 100)))

# Combine data for comparison with more sample sizes
rain_means_df <- data.frame(
  sample_mean = c(rain_means_n1, rain_means_n5, rain_means_n15,
                  rain_means_n20, rain_means_n30, rain_means_n100),
  sample_size = factor(rep(c(1, 5, 15, 20, 30, 100), each = 1000),
                       levels = c(1, 5, 15, 20, 30, 100))
)
```

```r
# Create faceted plot
ggplot(rain_means_df, aes(x = sample_mean)) +
  geom_density(fill = "skyblue", alpha = 0.7) +
  geom_vline(xintercept = mean(rainfall_population),
             colour = "red", linetype = "dashed") +
  facet_wrap(~ sample_size, ncol = 3, scales = "free_y",
             labeller = labeller(sample_size = function(x) paste("n =", x))) +
  labs(x = "Sample Mean Rainfall (mm)", y = "Density",
       title = "Effect of Sample Size on Distribution of Sample Means") +
  theme(strip.background = element_rect(fill = "lightgrey"),
        strip.text = element_text(size = 12, face = "bold"))
```



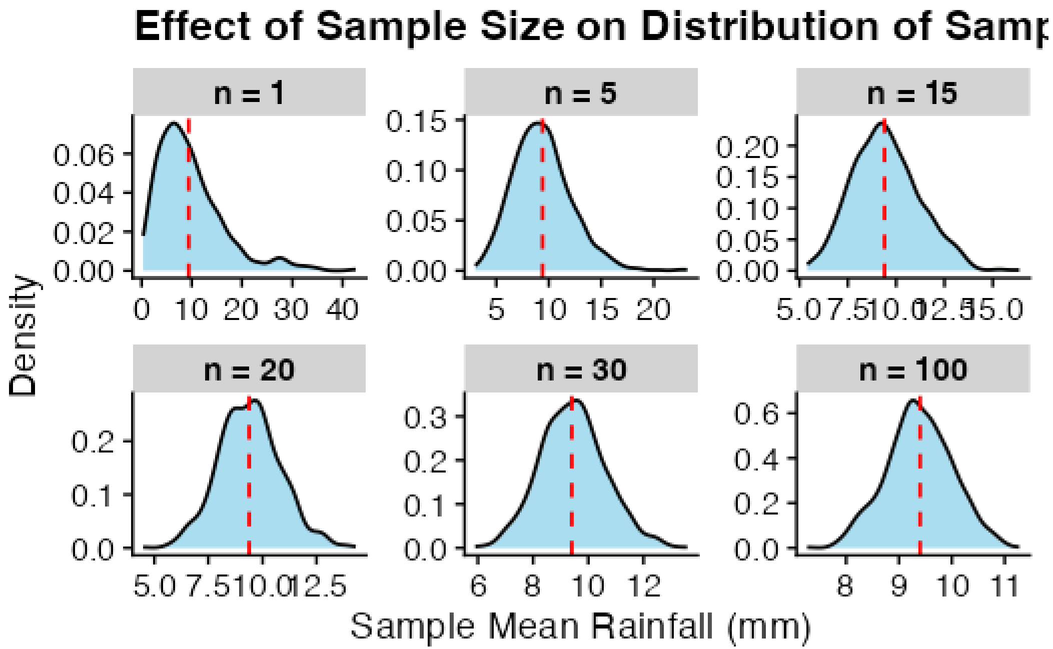
As sample size increases, something remarkable happens:

1. The distribution becomes more symmetrical and bell-shaped
2. The spread of sample means decreases
3. The distribution approaches a normal distribution - **did it become better with 100 samples?**

Zoomed-in

What happens when we increase our sample size?

```r
ggplot(rain_means_df, aes(x = sample_mean)) +
  geom_density(fill = "skyblue", alpha = 0.7) +
  geom_vline(xintercept = mean(rainfall_population),
             colour = "red", linetype = "dashed") +
  facet_wrap(~ sample_size, ncol = 3, scales = "free",
             labeller = labeller(sample_size = function(x) paste("n =", x))) +
  labs(x = "Sample Mean Rainfall (mm)", y = "Density",
       title = "Effect of Sample Size on Distribution of Sample Means") +
  theme(strip.background = element_rect(fill = "lightgrey"),
        strip.text = element_text(size = 12, face = "bold"))
```



As sample size increases, something remarkable happens:

1. The distribution becomes more symmetrical and bell-shaped
2. The spread of sample means decreases
3. The distribution approaches a normal distribution

The Central Limit Theorem

> I know of scarcely anything so apt to impress the imagination as the wonderful form of cosmic order expressed by the law of frequency of error. The law would have been personified by the Greeks if they had known of it. It reigns with serenity and complete self-effacement amidst the wildest confusion. The larger the mob, the greater the apparent anarchy, the more perfect is its sway. It is the supreme law of unreason."

– Sir Francis Galton (source) (1822-1911)

What is the CLT?

The Central Limit Theorem (CLT) states that:

When we take sufficiently large random samples from any population:

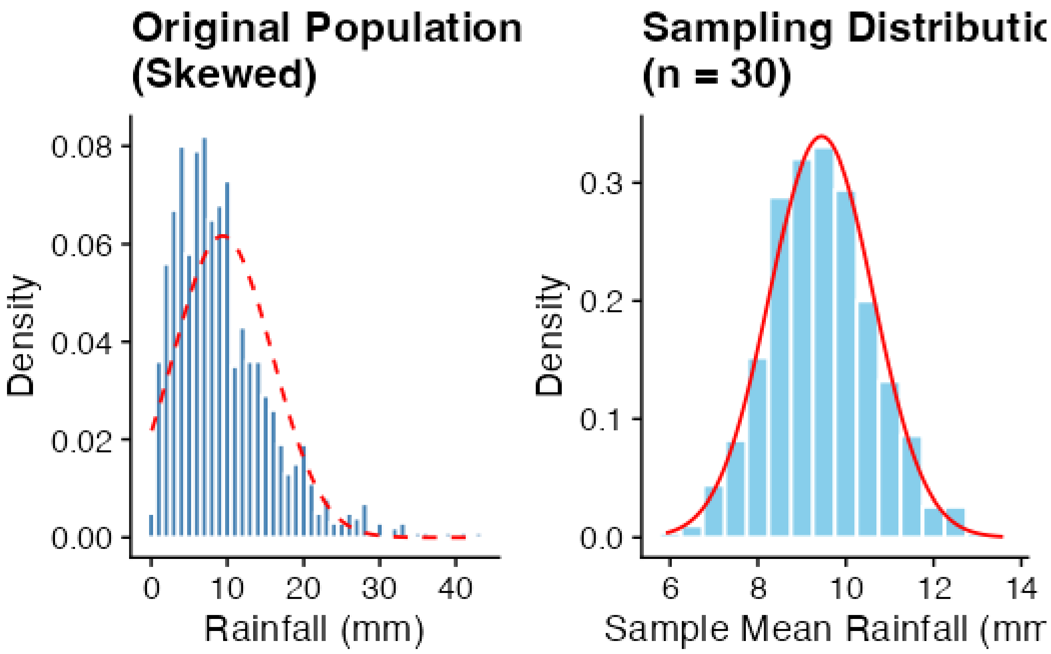
1. The distribution of **sample means** will be approximately normal
2. The mean of the sample means will equal the population mean
3. The standard deviation of sample means (standard error) equals $\frac{\sigma}{\sqrt{n}}$

This is true regardless of the shape of the original population distribution.

Comparing original data with sampling distribution

Let’s compare our original skewed rainfall data with the distribution of sample means:

```r
# Plot original population and sampling distributions
p1 <- ggplot(data.frame(x = rainfall_population), aes(x = x)) +
  geom_histogram(aes(y = ..density..), binwidth = 1,
                 fill = "steelblue", colour = "white") +
  stat_function(fun = dnorm,
                args = list(mean = mean(rainfall_population),
                            sd = sd(rainfall_population)),
                colour = "red", linetype = "dashed") +
  labs(x = "Rainfall (mm)", y = "Density",
       title = "Original Population\n(Skewed)")

p2 <- ggplot(data.frame(x = rain_means_n30), aes(x = x)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.5,
                 fill = "skyblue", colour = "white") +
  stat_function(fun = dnorm,
                args = list(mean = mean(rain_means_n30),
                            sd = sd(rain_means_n30)),
                  colour = "red") +
  labs(x = "Sample Mean Rainfall (mm)", y = "Density",
       title = "Sampling Distribution\n(n = 30)")

plot_grid(p1, p2, ncol = 2)
```

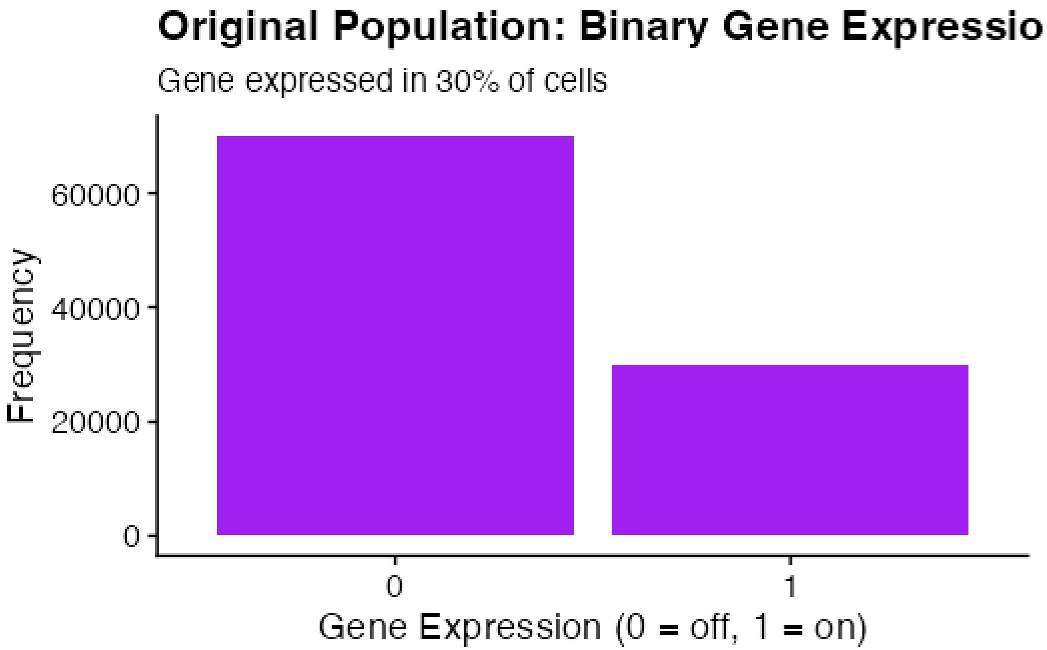


Even with skewed data, the sampling distribution becomes normal as sample size increases!

The CLT with binary biological data

In biology, many processes have binary outcomes: a gene is expressed or not, a cell divides or not, an organism survives or dies. Let's demonstrate the CLT with gene expression data:

```r
# Create a binary population (gene expression: 0 = off, 1 = on)
# Assume the gene is expressed in 30% of cells
set.seed(123)
n_cells <- 100000
p_expressed <- 0.3
gene_expression <- rbinom(n_cells, size = 1, prob = p_expressed)

# Plot the original population distribution
ggplot(data.frame(expression = factor(gene_expression)), aes(x = expression)) +
  geom_bar(fill = "purple") +
  labs(x = "Gene Expression (0 = off, 1 = on)", y = "Frequency",
       title = "Original Population: Binary Gene Expression",
       subtitle = "Gene expressed in 30% of cells")
```

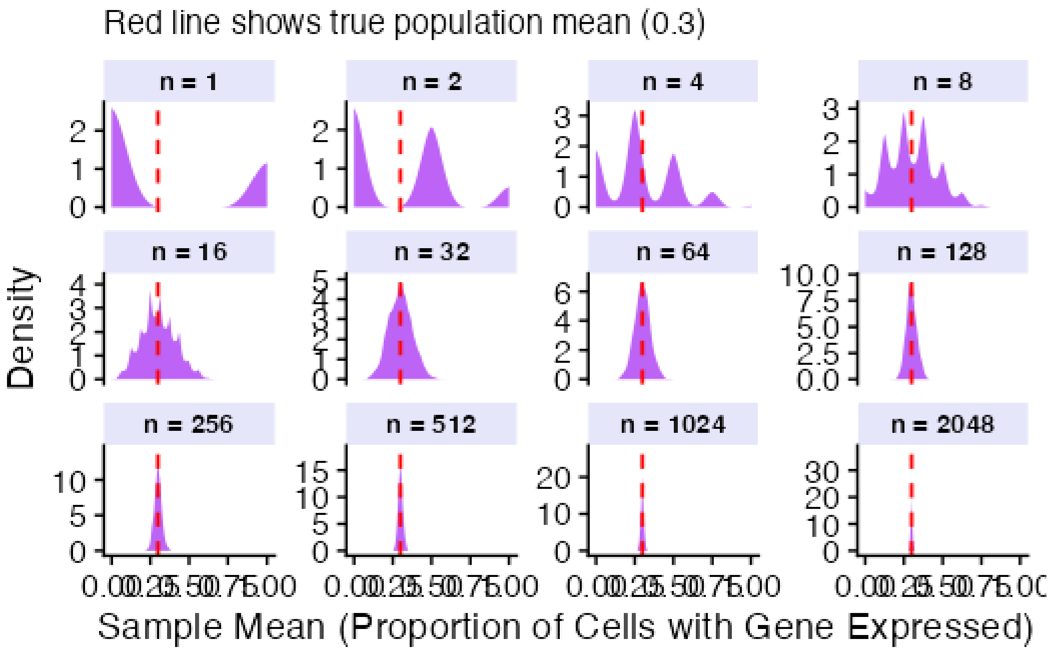


This is a highly non-normal distribution - it’s discrete with only two possible values!

Sampling distribution with increasing sample sizes

```r
# Define sample sizes (powers of 2)
sample_sizes <- c(1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048)

# Function to generate sample means for a given sample size
generate_sample_means <- function(sample_size, n_samples = 1000) {
  replicate(n_samples, mean(sample(gene_expression, size = sample_size, replace = TRUE)))
}

# Generate sample means for each sample size
set.seed(456)
all_sample_means <- lapply(sample_sizes, generate_sample_means)
names(all_sample_means) <- paste("n =", sample_sizes)

# Combine into a data frame
sample_means_df <- data.frame(
  sample_mean = unlist(all_sample_means),
  sample_size = factor(rep(names(all_sample_means), each = 1000),
                      levels = names(all_sample_means))
)
```

```r
# Create faceted plot with density plots instead of histograms
ggplot(sample_means_df, aes(x = sample_mean)) +
  geom_density(fill = "purple", color = "white", alpha = 0.7) +
  geom_vline(xintercept = mean(gene_expression),
             color = "red", linetype = "dashed") +
  facet_wrap(~ sample_size, scales = "free_y", ncol = 4) +
  labs(x = "Sample Mean (Proportion of Cells with Gene Expressed)",
       y = "Density",
       subtitle = "Red line shows true population mean (0.3)") +
  theme(strip.background = element_rect(fill = "lavender"),
        strip.text = element_text(size = 10, face = "bold"))
```



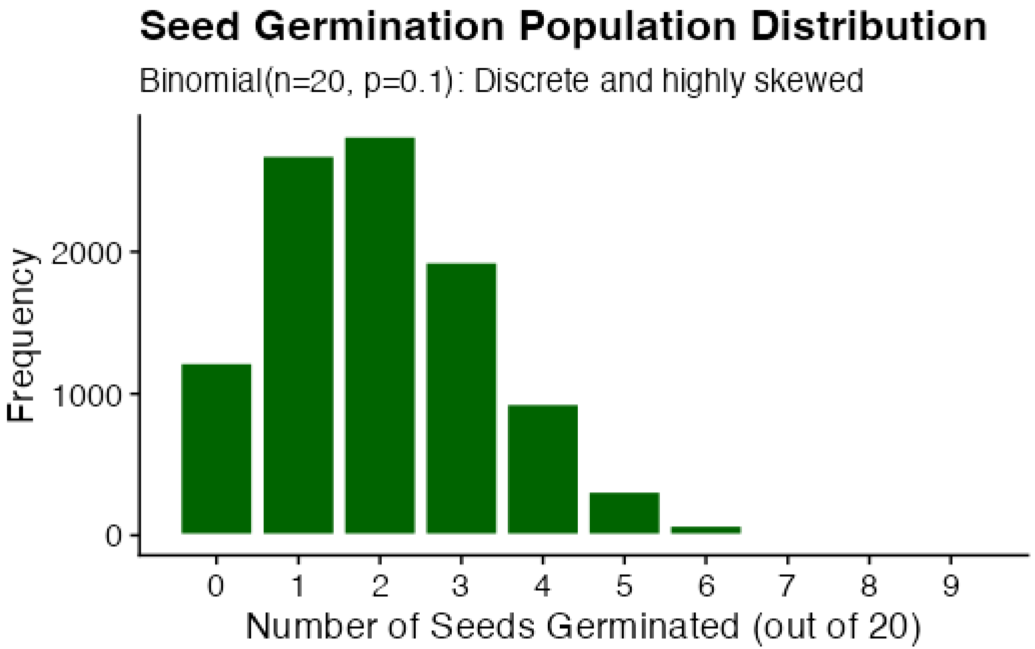
Even with the most non-normal data possible (binary data), the sampling distribution of means becomes normal with sufficient sample size.

Another example: Seed germination (discrete data)

Let’s explore another example with a discrete, highly skewed distribution:

In ecological restoration, practitioners need to estimate germination rates of native seeds. Imagine each seed has only a 10% chance of germinating (p = 0.1).

```r
# Create a binomial population - number of germinated seeds in batches of 20
set.seed(789)
n_trials <- 20  # 20 seeds per batch
p_success <- 0.1  # 10% germination rate
n_batches <- 10000  # 10,000 batches to create our "population"

# Generate the population data
germination_population <- rbinom(n_batches, size = n_trials, prob = p_success)

# Plot the population distribution
ggplot(data.frame(germinated = germination_population), aes(x = germinated)) +
  geom_bar(fill = "darkgreen", color = "white") +
  scale_x_continuous(breaks = 0:10) +
  labs(x = "Number of Seeds Germinated (out of 20)",
       y = "Frequency",
       title = "Seed Germination Population Distribution",
       subtitle = "Binomial(n=20, p=0.1): Discrete and highly skewed")
```



This distribution is: - Discrete (only whole numbers of seeds can germinate) - Highly skewed (most batches have very few germinating seeds) - Very different from our continuous rainfall example

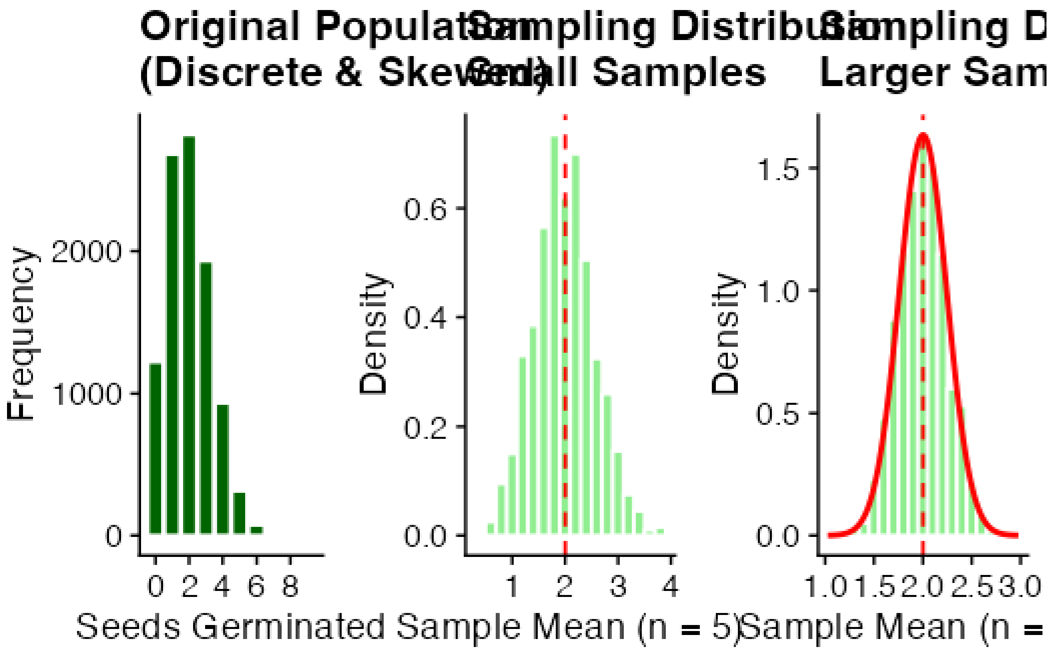
Sampling distribution for seed germination data

What happens when we take samples from this discrete, skewed distribution?

```r
# Take samples of different sizes and calculate means
set.seed(123)
seed_means_n5 <- replicate(1000, mean(sample(germination_population, size = 5)))
seed_means_n30 <- replicate(1000, mean(sample(germination_population, size = 30)))

# Calculate theoretical standard error for n=30
se_n30 <- sd(germination_population)/sqrt(30)

# Plot the original population vs. sampling distributions
p1 <- ggplot(data.frame(x = germination_population), aes(x = x)) +
  geom_bar(fill = "darkgreen", color = "white") +
  scale_x_continuous(breaks = seq(0, 10, by = 2)) +
  labs(x = "Seeds Germinated", y = "Frequency",
       title = "Original Population\n(Discrete & Skewed)")

p2 <- ggplot(data.frame(x = seed_means_n5), aes(x = x)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.2,
                fill = "lightgreen", colour = "white") +
  geom_vline(xintercept = mean(germination_population),
             colour = "red", linetype = "dashed") +
  labs(x = "Sample Mean (n = 5)", y = "Density",
       title = "Sampling Distribution\nSmall Samples")

# For n=30, use density scale and properly scaled normal curve
p3 <- ggplot(data.frame(x = seed_means_n30), aes(x = x)) +
  geom_histogram(aes(y = ..density..), binwidth = 0.1,
                fill = "lightgreen", colour = "white") +
  geom_vline(xintercept = mean(germination_population),
             colour = "red", linetype = "dashed") +
  stat_function(fun = dnorm,
                args = list(mean = mean(germination_population),
                           sd = se_n30),
                colour = "red", size = 1) +
  labs(x = "Sample Mean (n = 30)", y = "Density",
       title = "Sampling Distribution\nLarger Samples") +
  # Set x-axis limits to focus on the relevant range
  xlim(mean(germination_population) - 4*se_n30,
       mean(germination_population) + 4*se_n30)
```

```r
# Arrange plots
plot_grid(p1, p2, p3, ncol = 3)
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```



Even with discrete, highly skewed data, the CLT still applies:

- The sampling distribution becomes more normal as sample size increases
- The sampling distribution centers on the population mean (≈ 2, since 20 × 0.1 = 2)
- The standard error decreases as sample size increases

Standard error: Measuring the precision of sample means

The standard error (SE) tells us how much sample means typically vary:

$$SE = \frac{\sigma}{\sqrt{n}}$$

Where: - σ is the population standard deviation - n is the sample size

```
# Calculate theoretical standard errors
pop_sd ← sd(rainfall_population)
se_n5 ← pop_sd / sqrt(5)
se_n15 ← pop_sd / sqrt(15)
se_n30 ← pop_sd / sqrt(30)

# Compare standard errors
se_n5
```

```
[1] 2.89159
```

```
se_n15
```

```
[1] 1.66946
```

```
se_n30
```

```
[1] 1.180487
```

As sample size increases, standard error decreases, making our estimates more precise.

Standard error vs. standard deviation

While we have been using the standard deviation to measure variability in our data, the standard error measures variability in our **sample means**.

```r
# Create comparison table between Standard Deviation and Standard Error
sd_se_comparison <- data.frame(
  Aspect = c("Definition", "Formula", "Measures", "Effect of sample size", "Typical use"),
  Standard_Deviation = c(
    "Measures spread of individual data points around their mean",
    "σ = √[Σ(x-μ)²/N] or s = √[Σ(x-x̄)²/(n-1)]",
    "Variability within a dataset",
    "Unaffected by sample size",
    "Describing the spread of data in a single sample"
  ),
  Standard_Error = c(
    "Measures precision of a sample statistic (usually the mean)",
    "SE = σ/√n or SE = s/√n",
    "Variability between sample means from the same population",
    "Decreases as sample size increases (∝ 1/√n)",
    "Inferential statistics, confidence intervals, hypothesis testing"
  )
)

kable(sd_se_comparison,
      col.names = c("Aspect", "Standard Deviation (SD)", "Standard Error (SE)"),
      caption = "Standard Deviation vs. Standard Error")
```

Aspect	Standard Deviation (SD)	Standard Error (SE)
Definition	Measures spread of individual data points around their mean	Measures precision of a sample statistic (usually the mean)
Formula	$\sigma = \surd[\Sigma(x-\mu)^2/N]$ or $s = \surd[\Sigma(x-\bar{x})^2/(n-1)]$	$SE = \sigma/\surd n$ or $SE = s/\surd n$
Measures	Variability within a dataset	Variability between sample means from the same population
Effect of sample size	Unaffected by sample size	Decreases as sample size increases ($\propto 1/\surd n$)
Typical use	Describing the spread of data in a single sample	Inferential statistics, confidence intervals, hypothesis testing

Practical implications of the CLT

The Central Limit Theorem enables modern inferential statistics:

1. **Making inferences from samples to populations**
   - Estimate population parameters with quantifiable precision
   - Calculate confidence intervals for parameters (coming soon!)
   - Perform hypothesis tests with known error rates
2. **Works regardless of population distribution**
   - Non-normal data still produces normally distributed sample means
   - Enables parametric tests even when original data is skewed
   - Only requires sufficiently large samples ($n \geq 30$ is often adequate)
3. **Foundation for statistical theory**

Thanks!