Simulation of distributions for the experiment on the relationship between the Illusion of Causality and the Foreign Language Effect

1. Fitting a Theoretical Distribution

To determine an appropriate Effect Size (ES) for detecting the difference between two groups exposed to a null contingency illusory condition (one in a native language – NL, and one in a foreign language – FL), we first examine the expected distribution of the dependent variable for the NL group. This distribution is based on data from a previous study (Dalla Bona & Vicovaro, 2024). We then extend this analysis to consider the expected differences between the NL and FL groups.

Given that our current Contingency Learning Task (CLT) experiment will be conducted online using Psychopy (Peirce et al., 2019), and that a previously conducted online experiment (Dalla Bona & Vicovaro, 2024) was using it (with data available at: https://osf.io/c26qa/files/osfstorage), we will use data from this prior study as a reference distribution. Specifically, we focus on the control group, which was not exposed to any manipulations of perceptual features, in the null contingency illusory condition. Our objective is to identify which generative distribution most likely underlies the observed empirical data points.

To identify potential candidate distributions, we employ a Cullen and Frey graph that plots kurtosis against the square of skewness (Frey & Cullen, 1995). This visualization helps us assess the distributional characteristics of the data and select the most appropriate generative model.

1.1 Importing Data and Describing It

We begin by loading the data from the Excel file and filtering it to include only participants with valid responses (`valido == "si"`). The `tempoistr` (time spent in the experiment) variable is then converted to numeric format, and we exclude any rows where the time spent is less than 10 seconds, as this was an exclusion criterion in the previous experiment. Next, we extract the causal evaluation scores for participants in the control group who were exposed to the null contingency scenario (identified by `gruppo == 1`). These scores are stored in the `vector_illusion` variable.

```r
# Reading the data from the Excel file
library(readxl)

data <- as.data.frame(read_xlsx("Dataframeclt_online.xlsx"))


# Structure of the dataset
str(data)
```

```
'data.frame':	249 obs. of  15 variables:
 $ cp1       : chr  "A" "A" "A" "B" ...
 $ cp2       : num  1 2 3 4 5 6 7 8 9 10 ...
 $ REF       : num  1 2 3 5 6 7 8 9 10 11 ...
 $ email     : logi  NA NA NA NA NA NA ...
 $ sesso     : chr  "F" "F" "F" "M" ...
 $ eta       : num  24 24 22 27 25 23 22 23 30 30 ...
 $ valido    : chr  "si" "si" "si" "si" ...
 $ gruppo    : num  3 2 1 2 1 4 4 3 4 3 ...
 $ tempoistr : chr  "33.7546" "33.2288" "67.6089" "138.8366" ...
 $ tempotask : chr  "347.3093" "336.9458" "328.1303" "265.2705" ...
 $ tempovd   : chr  "10.4335" "10.8814" "9.2636" "44.5247" ...
 $ tempocheck: chr  "12.1336" "9.3648" "7.7313" "9.0483" ...
 $ tempotot  : chr  "486.661" "434.2955" "469.9609" "537.1056" ...
 $ vd        : num  59 50 69 40 70 80 75 68 70 80 ...
 $ check     : num  1 6 1 7 2 6 5 1 6 1 ...
```

```
# Subset the first 209 rows
data <- data[1:209,]

# Filter data for valid responses
data <- subset(data, data$valido == "si")

# Convert 'tempoistr' to numeric and filter out fast responders (time < 10 seconds)
data$tempoistr <- as.numeric(data$tempoistr)
data <- data[-c(which(data$tempoistr < 10)), ]

# Extract causal evaluation scores for the control group
vector_illusion <- data$vd[data$gruppo == 1]
vector_general <-  data$vd
```

The causal evaluation is treated as a metric (interval-like) variable, with scores ranging from 0 to 100. For the control group (N = 60), the distribution of the dependent variable is fairly spread out, with a mean of 58.58, a median of 63, and a standard deviation of 18.51. The range spans from a minimum of 16 to a maximum of 90, and 50% of the data falls between 50 and 73 (1st Quartile = 49.5, 3rd Quartile = 73). The distribution exhibits a moderate negative skew (skewness = -0.63), which may be partially attributable to the data being bounded between 0 and 100. Additionally, the distribution is slightly platykurtic (kurtosis = -0.45), indicating a flatter shape compared to a normal distribution.

To visualize this distribution, we use a combination of plots: the boxplot highlights the quartiles and potential outliers, the half-eye plot provides a kernel density estimate, and the dot plot emphasizes individual data points.

```r
# View the length and summary of the vector
length(vector_illusion)
```

```
[1] 60
```

```r
summary(vector_illusion)
```

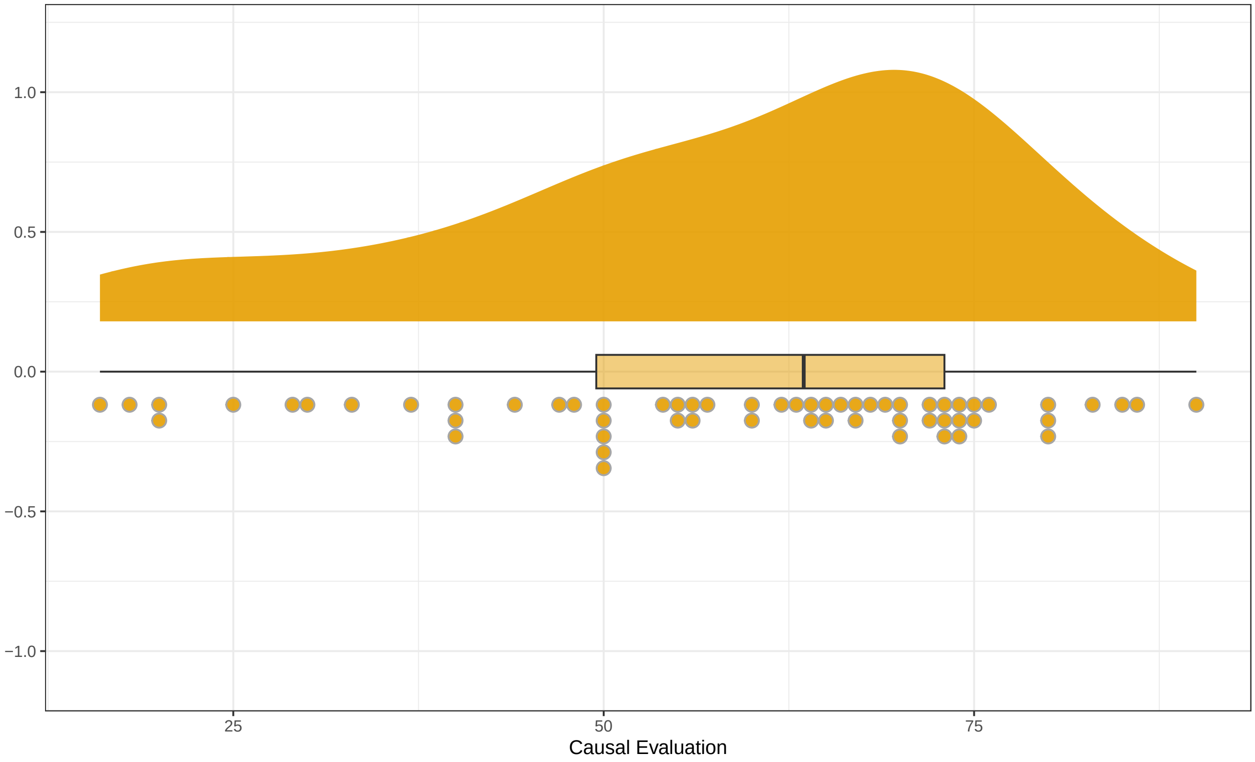
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  16.00   49.50   63.50   58.58   73.00   90.00 
```

```r
# Calculate detailed descriptive statistics
library(pastecs)
round(stat.desc(vector_illusion, norm = TRUE), 2)
```

```
     nbr.val     nbr.null       nbr.na          min          max        range 
       60.00         0.00         0.00        16.00        90.00        74.00 
         sum       median         mean      SE.mean CI.mean.0.95          var 
     3515.00        63.50        58.58         2.39         4.78       342.59 
     std.dev     coef.var     skewness     skew.2SE     kurtosis     kurt.2SE 
       18.51         0.32        -0.63        -1.02        -0.45        -0.37 
  normtest.W   normtest.p 
        0.95         0.01 
```

```r
# Load ggplot packages
library(ggdist); library(ggthemes); library(ggplot2)

library(ggokabeito) #Colorblind friendly palette

# Visual representation
```

```
ggplot(mapping = aes(y = vector_illusion, fill = factor(1))) +
  scale_fill_okabe_ito( alpha = .9) +
  stat_halfeye(adjust = 0.9, justification = -0.2, .width = 0, point_colour = NA) +
  geom_boxplot(width = 0.12, outlier.color = NA, alpha = 0.5) +
  stat_dots(side = "left", justification = 1.1, binwidth = 1) +
  labs(y = "Causal Evaluation", x = "") +
  coord_flip() +
  theme_bw() +
  theme(legend.position = "none")
```



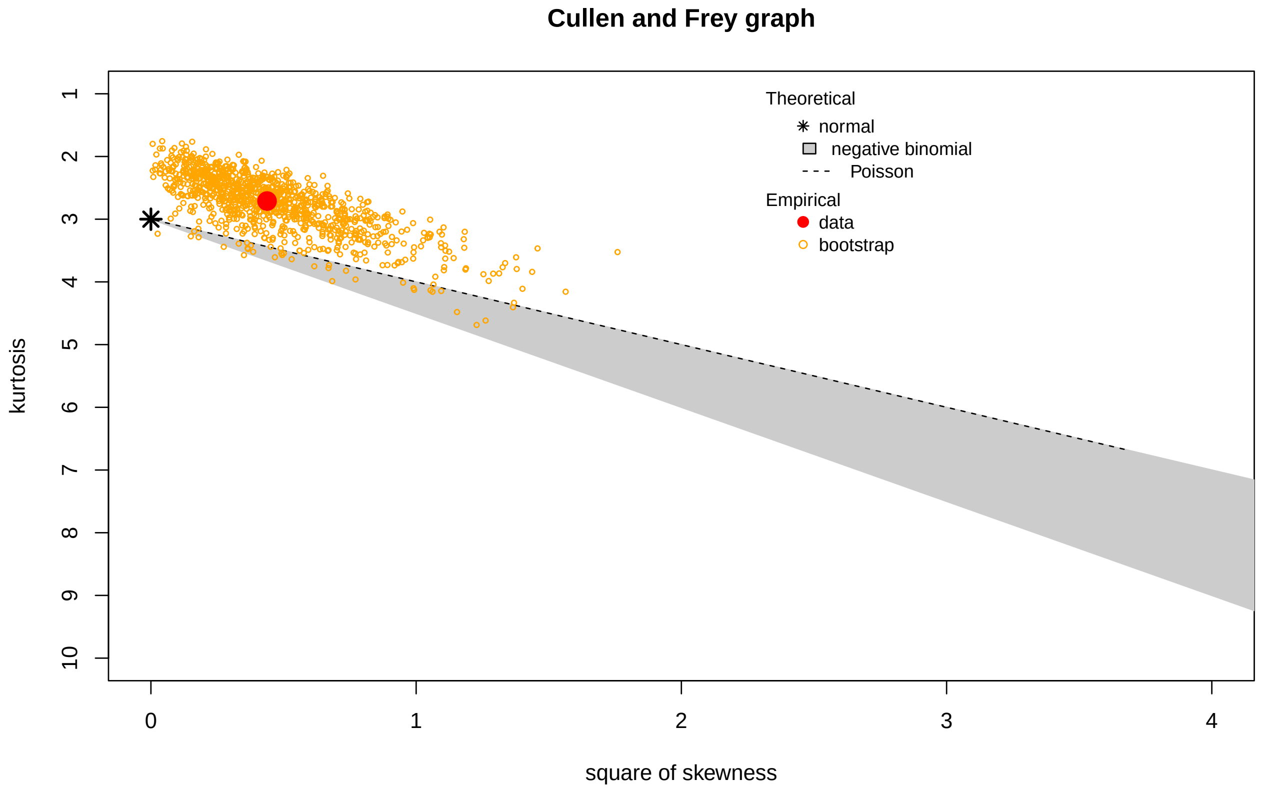
1.2 Generative Distributions

In this section, we aim to identify the distribution that best describes our sample data. To do so, we compare our data to various theoretical distributions using the Cullen and Frey graph. This graph plots kurtosis against the square of skewness (Frey & Cullen, 1995), allowing us to visually assess how well our data aligns with different distribution families.

The data are represented as a red point on the graph, indicating how closely the distribution resembles known distribution families. To enhance the analysis, we also include bootstrapped data (generated from the sample) to assess how the distribution holds up under resampling.

To generate the graph and perform the analysis, we use the `fitdistrplus` package. The following code fits the data to various distributions and plots the Cullen and Frey graph, treating the variable as discrete (assuming the random variable takes countable values).

```
# Generate the Cullen and Frey graph with bootstrapping (1000 resamples)
library(fitdistrplus)


descdist(vector_illusion, boot = 1000, discrete = TRUE)
```



```
summary statistics
------
min:  16   max:  90
median:  63.5
```

```
mean:  58.58333

estimated sd:  18.50908

estimated skewness:  -0.6617163

estimated kurtosis:  2.710791
```

We consider two candidate distributions for the data: the Normal distribution and the Poisson distribution. To determine the most suitable model, we perform a series of diagnostic plots and statistical tests that compare the empirical distribution of the data against the theoretical distributions.

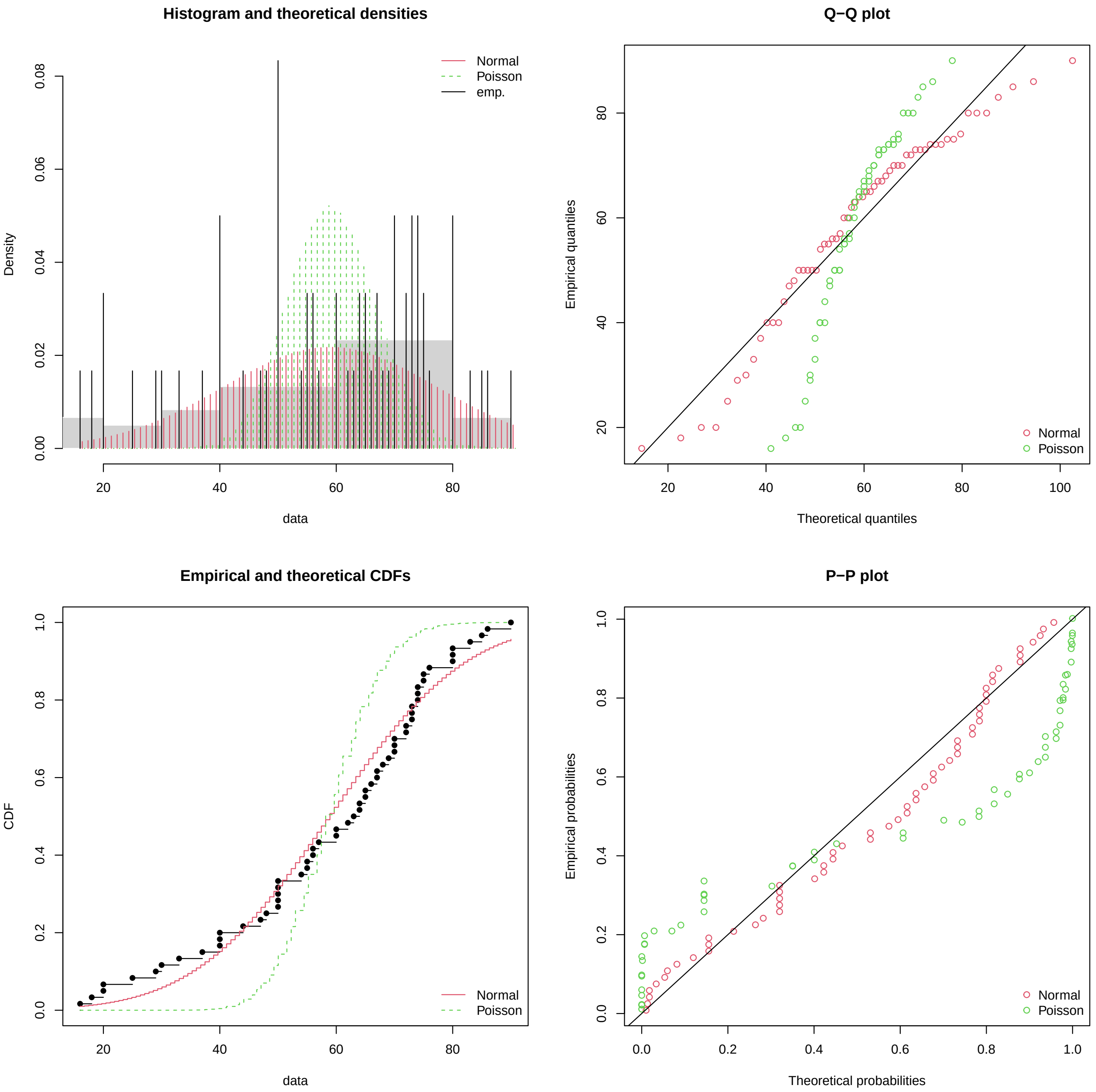
The following diagnostic plots are used:

- **Density comparison (`denscomp`)**: Compares the histogram of the empirical data to the fitted density functions of the theoretical distributions.
- **Q-Q plot (`qqcomp`)**: Plots the theoretical quantiles of the fitted distributions against the quantiles of the empirical data.
- **CDF comparison (`cdfcomp`)**: Compares the cumulative distribution function (CDF) of the empirical data to the CDFs of the fitted distributions.
- **P-P plot (`ppcomp`)**: Plots the theoretical cumulative probabilities against the empirical cumulative probabilities.

From these diagnostic plots, we observe that the Normal distribution fits the data better than the Poisson distribution. This observation is further confirmed by the goodness-of-fit tests, which indicate that the Normal distribution is the most probable model for our data. Specifically, the Akaike weight for the Normal distribution is approximately 1, suggesting that it is the best-fitting model.

It is important to note that the Normal distribution fitted to the data is effectively truncated, particularly on the right-hand side. This truncation occurs because the mean of the distribution exceeds the median point of 50, and the data are constrained within the boundaries of 0 to 100. As a result, values that would typically fall outside the upper boundary are clipped, leading to skewness, especially in the left tail. Based on these observations, we assume that the truncated Normal distribution is the most appropriate generative distribution for further analysis.

```r
# Fit distributions to the data
fnorm <- fitdist(vector_illusion, "norm")  # Fitting Normal distribution
fpois <- fitdist(vector_illusion, "pois")  # Fitting Poisson distribution

plot.legend <- c("Normal", "Poisson")
par(mfrow = c(2, 2))

# Density comparison
denscomp(list(fnorm, fpois), legendtext = plot.legend)

# Q-Q plot comparison
qqcomp(list(fnorm, fpois), legendtext = plot.legend)

# CDF comparison
cdfcomp(list(fnorm, fpois), legendtext = plot.legend)

# P-P plot comparison
ppcomp(list(fnorm, fpois), legendtext = plot.legend)
```



```
# Goodness-of-fit tests

gofstat(list(fnorm, fpois), fitnames = c("Normal", "Poisson"))
```

```
Goodness-of-fit statistics

                              Normal    Poisson

Kolmogorov-Smirnov statistic 0.1160484  0.2933251

Cramer-von Mises statistic   0.1692820  1.8065351
```

```
Anderson-Darling statistic   1.0540738 30.3656046


Goodness-of-fit criteria
                                  Normal  Poisson
Akaike's Information Criterion 523.4556 743.5827
Bayesian Information Criterion 527.6443 745.6771
```

```r
# Akaike weights
C <- gofstat(list(fnorm, fpois), fitnames = c("Normal", "Poisson"))


AIC_W_pois <- exp(-1/2 * (C$aic[2] - C$aic[1])) /
  (exp(-1/2 * (C$aic[1] - C$aic[2])) + exp(-1/2 * (C$aic[2] - C$aic[1])))


AIC_W_norm <- exp(-1/2 * (C$aic[1] - C$aic[2])) /
  (exp(-1/2 * (C$aic[1] - C$aic[2])) + exp(-1/2 * (C$aic[2] - C$aic[1])))


as.numeric(AIC_W_norm); round(as.numeric(AIC_W_pois),3)
```

```
[1] 1
```

```
[1] 0
```

2. Simulating Approach to Estimate ES

In this section, we employ a simulation-based approach to estimate a meaningful ES. Specifically, we aim to generate simulated data from a theoretical distribution (the truncated Normal distribution) and hypothesize about the potential meaningful difference between two groups, namely NL (i.e., the control group) and FL (i.e., the treatment group).

2.1 Simulating the Distribution

We have already established that the Normal distribution is the most likely model for our data. However, we must account for the fact that our data is discrete and constrained within the boundaries of 0 and 100. To address these constraints, we simulate the data using a truncated normal distribution, which respects both the boundaries and the characteristics of the data. The truncated normal distribution ensures that all generated values fall within the specified range of 0 to 100.

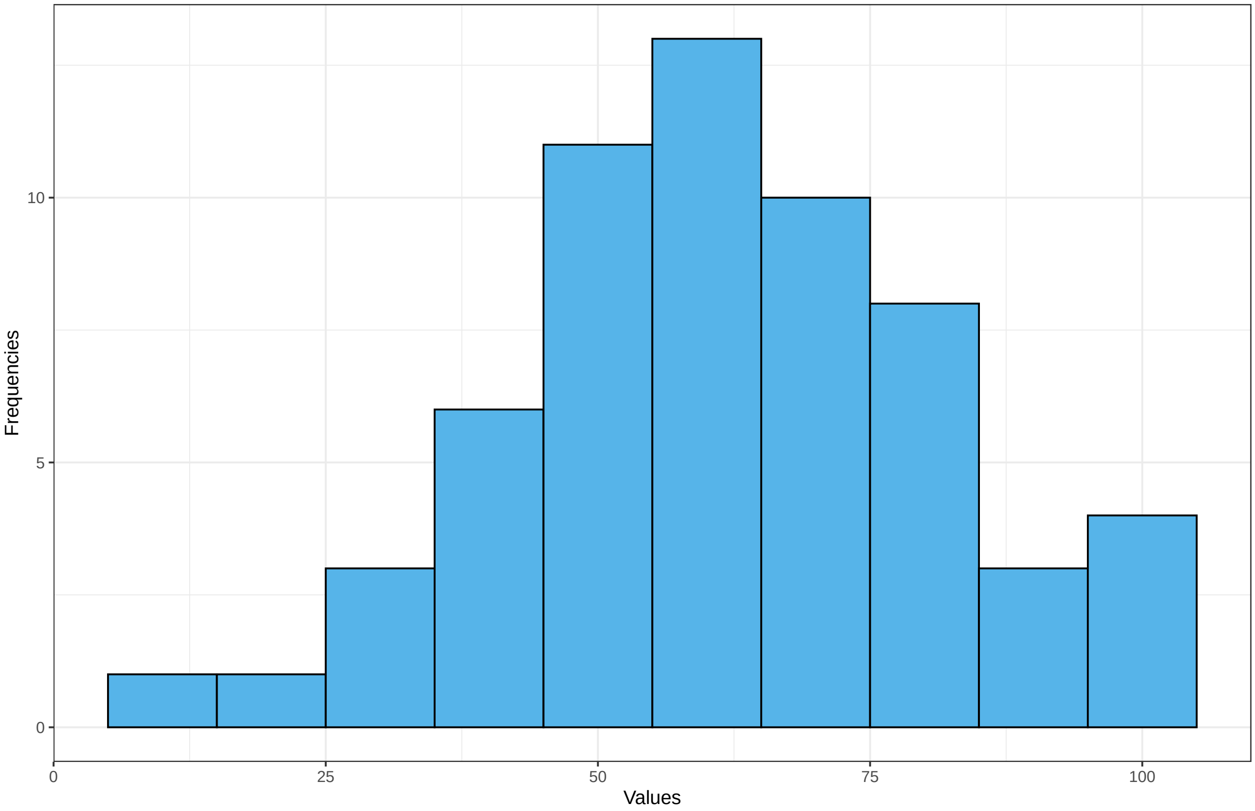
We will use a custom function, based on the `qnorm` and `runif` functions, to generate truncated normal data:

- `qnorm`: The quantile function for the normal distribution, which transforms uniformly distributed random numbers into quantiles of the normal distribution.
- `runif`: The function used to generate uniformly distributed random numbers, which are then transformed by `qnorm` into the desired normal distribution.
- `round`: Since our data needs to be discrete, we round the simulated values to the nearest integer.

By imposing boundaries on the simulated data, we truncate the distribution. This means that any data points that would normally fall outside the specified range (0 to 100) are trimmed. Consequently, truncation introduces skewness in the distribution, particularly when the mean is positioned closer to one of the boundaries. This behavior is expected when dealing with bounded data.

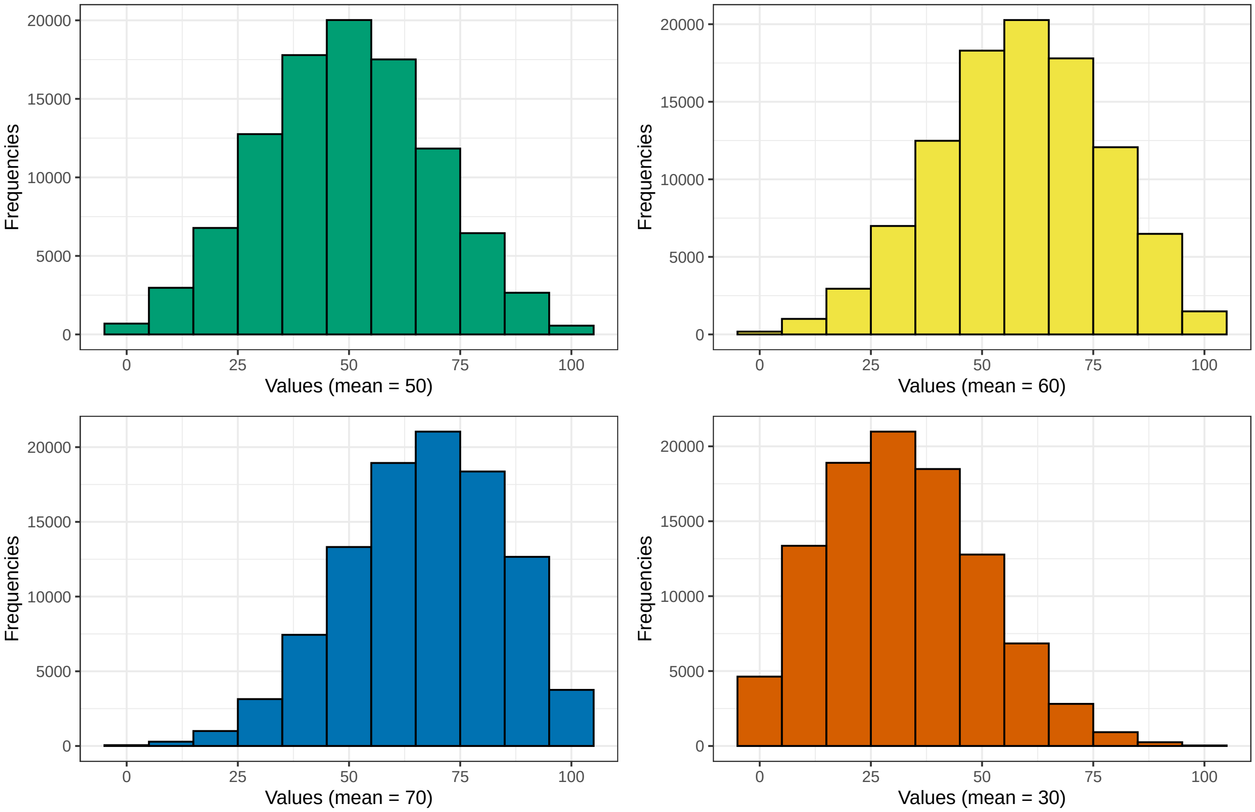
When the mean of the distribution passed to the function is more extreme (i.e., closer to the boundaries), the simulation generates a distribution with a slightly more centered mean. This occurs because truncation removes values in the tails, effectively shrinking the range of the distribution. As a result, the standard deviation (SD) of the distribution is reduced as the mean moves toward the boundaries, because truncation limits the spread of the data.

```r
# Generation from a "truncated" normal distribution
tnorm_f <- function(n, mean, sd, a = 0, b = 100) {
  qnorm(runif(n, pnorm(a, mean, sd), pnorm(b, mean, sd)), mean, sd)
} # Truncated at 0 and 100 (a and b)

# Using the truncated normal distribution
set.seed(4234)
x <- round(tnorm_f(length(vector_illusion), 58.58, 18.51))

# Plot values
ggplot(mapping = aes(x, fill = factor(1))) +
  geom_histogram(binwidth = 10, color = "black") +
  theme_bw() +
  scale_fill_okabe_ito(order = 2) +
  theme(legend.position = "none") +
  labs(x = "Values", y = "Frequencies")
```



```
# Simulate with different means to observe the effects of truncation
a <- round(tnorm_f(10e4, 50, 20))  # Mean = 50 (n=10000)
b <- round(tnorm_f(10e4, 60, 20))  # Mean = 60 (n=10000)
c <- round(tnorm_f(10e4, 70, 20))  # Mean = 70 (n=10000)
d <- round(tnorm_f(10e4, 30, 20))  # Mean = 30 (n=10000)



gga <- ggplot(mapping = aes(a, fill = factor(1))) +
  geom_histogram(binwidth = 10, color = "black") +
  theme_bw() +
  scale_fill_okabe_ito(order = 3) +
  theme(legend.position = "none") +
  labs(x = "Values (mean = 50)", y = "Frequencies")
```

```r
ggb <-ggplot(mapping = aes(b, fill = factor(1))) +
  geom_histogram(binwidth = 10, color = "black") +
  theme_bw() +
  scale_fill_okabe_ito(order = 4) +
  theme(legend.position = "none") +
  labs(x = "Values (mean = 60)", y = "Frequencies")

ggc <-ggplot(mapping = aes(c, fill = factor(1))) +
  geom_histogram(binwidth = 10, color = "black") +
  theme_bw() +
  scale_fill_okabe_ito(order = 5) +
  theme(legend.position = "none") +
  labs(x = "Values (mean = 70)", y = "Frequencies")

ggd <-ggplot(mapping = aes(d, fill = factor(1))) +
  geom_histogram(binwidth = 10, color = "black") +
  theme_bw() +
  scale_fill_okabe_ito(order = 6) +
  theme(legend.position = "none") +
  labs(x = "Values (mean = 30)", y = "Frequencies")

gridExtra::grid.arrange(gga,ggb,ggc,ggd)
```



```r
# Simulate with different means to observe the effects of truncation
x <- round(tnorm_f(100000, 50, 20))  # Mean = 50
y <- rnorm(100000, 50, 20)  # Untruncated normal

# Calculate means and SDs
mean(x); sd(x)
```

```
[1] 49.93504
[1] 19.07987
```

```r
mean(y); sd(y)
```

```
[1] 49.98902
[1] 19.91653
```

```
# Simulate with higher mean to observe the effect more clearly
x <- round(tnorm_f(100000, 65, 20))  # Mean = 65
y <- rnorm(100000, 65, 20)  # Untruncated normal

# Calculate means and SDs
mean(x); sd(x)
```

```
[1] 63.28526
```

```
[1] 18.18063
```

```
mean(y); sd(y)
```

```
[1] 65.06018
```

```
[1] 19.9791
```

2.2 Comparisons

In this section, we compare how well the truncated normal distribution simulates the empirical data, relative to the normal distribution without boundaries. We visually examine the three distributions (empirical, normal, and truncated normal) and compute the `overlap` indices between the empirical data and the theoretical simulated distributions (with the same sample size). The overlap metric quantifies how much the distributions overlap, providing insight into how well each simulated distribution fits the empirical data.

From this analysis, we observe that the truncated normal distribution, with a inputted mean of 60 and a standard deviation of 20 (closely approximating the empirical mean of 58.58 and standard deviation of 18.51 of the original data – M = 58.97, SD = 18.70), provides a marginally better fit for the data compared to the normal distribution without boundaries. Nonetheless, we continue to use the truncated normal distribution, as we anticipate that, by approaching more extreme values for the mean, this simulation approach will even better capture the behavior of the data. Specifically, the truncated normal distribution allows us to more effectively model the data's characteristics based on the positioning of the mean within the bounded range.

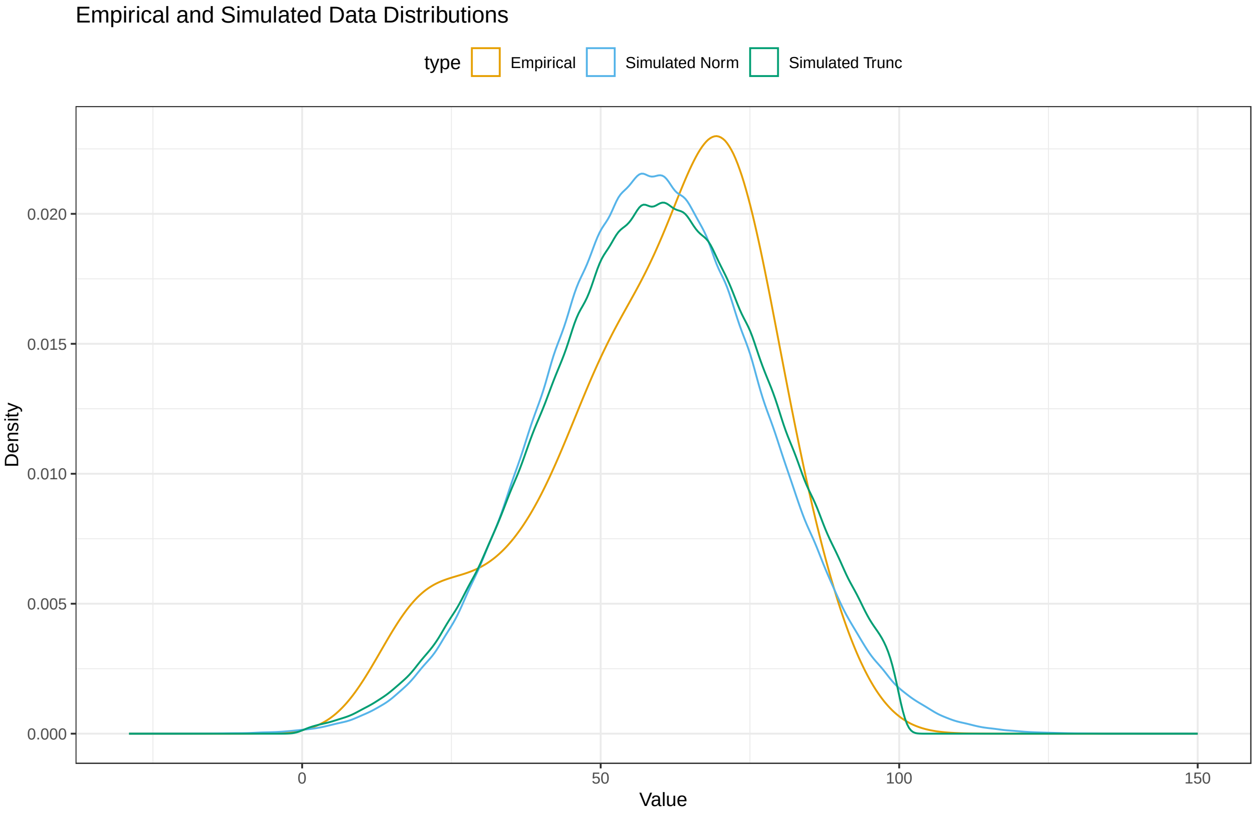
```
# Simulate data for normal and truncated normal distributions
set.seed(4234)

simulated_norm <- round(rnorm(1000000, mean(vector_illusion), sd(vector_illusion)))
mean(simulated_norm); sd(simulated_norm)
```

```
[1] 58.57679

[1] 18.51333
```

```
simulated_trunc <- round(tnorm_f(1000000, 60, 20))
mean(simulated_trunc); sd(simulated_trunc)
```

```
[1] 58.96533

[1] 18.70319
```

```
# Plot the distributions
empirical_df <- data.frame(value = vector_illusion, type = "Empirical")
simulated_df <- data.frame(value = simulated_norm, type = "Simulated Norm")
simulated_df1 <- data.frame(value = simulated_trunc, type = "Simulated Trunc")

combined_df <- rbind(empirical_df, simulated_df, simulated_df1)

ggplot(combined_df, aes(x = value, col = type)) +
  geom_density(alpha = 0.5) +
  scale_color_okabe_ito()  +
  labs(title = "Empirical and Simulated Data Distributions",
       x = "Value",
       y = "Density") +
  theme_bw() +
  theme(legend.position = "top")
```



```
# Calculate overlap coefficient with normal distribution
library(overlapping)


overlap_norm_values <- rep(NA, 1000)
for(i in 1:1000){
  simulated_sample <- round(rnorm(length(vector_illusion), 58.58, 18.41))
  overlap_norm_values[i] <- overlap(list(simulated_sample, vector_illusion), type = "1")
}
median(as.numeric(overlap_norm_values)); mean(as.numeric(overlap_norm_values))
```

```
[1] 0.863875
```

```
[1] 0.8617332
```

```
# Calculate overlap coefficient with truncated normal distribution
overlap_trunc_values <- rep(NA, 1000)
for(i in 1:1000){
  simulated_sample <- round(tnorm_f(length(vector_illusion), 60, 20))
  overlap_trunc_values[i] <- overlap(list(simulated_sample, vector_illusion), type = "1"
}
median(as.numeric(overlap_trunc_values)); mean(as.numeric(overlap_trunc_values))
```

```
[1] 0.8783599
```

```
[1] 0.8763157
```

2.3 ES estimation

In this analysis, we are focused on estimating a meaningful ES within the context of a Likert scale ranging from 0 to 100. Here, 0 represents the lowest possible rating (i.e., “Definitely not”), and 100 represents the highest possible rating (i.e., “Definitely yes”). Supposing we were interested in a 10-point change on this scale, that can be defined as a target ES, this would correspond to a 1/10 shift in the range of causality evaluations. This shift could represent the degree to which participants’ evaluations of a causal relationship differ when presented in a FL versus their NL. This shift is perhaps plausible and it can reflects a moderate change in the associative strength between the cue and the outcome, providing a clear benchmark for evaluating the effect of the manipulation.

To simulate data for the NL group, we assume that a truncated normal distribution with a mean rating of 60 and a standard deviation (SD) of 20 is the generative distribution. This parametrization is based on the fit of data from our previous experiment, where the observed mean was 58.58 and the SD was 18.21. Using this distribution, the simulation produced a mean of 58.98 and a SD of 18.70, which closely mirrors the characteristics of the control group data.

For the FL group, we simulate a second truncated normal distribution with the mean shifted 10 points lower, representing the effect size. This results in a parameterized mean of 50, while the parameterized standard deviation remains the same at 20, ensuring that the spread of values in the FL group is similar to the NL group. The simulation for the FL group yielded a mean of 49.10 and a standard deviation of 19.03.

When parameterizing from a truncated normal distribution, we observe that the more extreme the initial parameter (e.g., a mean larger than 50), the more the actual computed mean is reduced (also, the computed standard deviation decreases). This behavior is expected, as truncation naturally compresses the distribution. However, in the next simulations, we will be cautious to ensure that the difference in means between the two groups remains at least 9.9, changing a little bit the initial parametrization of the FL group, as the truncated distribution may sometimes produce a smaller difference than the target 10-point shift.

```r
# Simulate for NL (Mean = 60, SD = 20)
set.seed(4234)
a <- round(tnorm_f(100000, 60, 20))

# Simulate for FL (Mean = 60 - 10, SD = 20)
b <- round(tnorm_f(100000, mean(a) - 10, 20))

# Calculate the means and SDs for both groups
mean(a); sd(a)  # NL group
```

```
[1] 58.97929
```

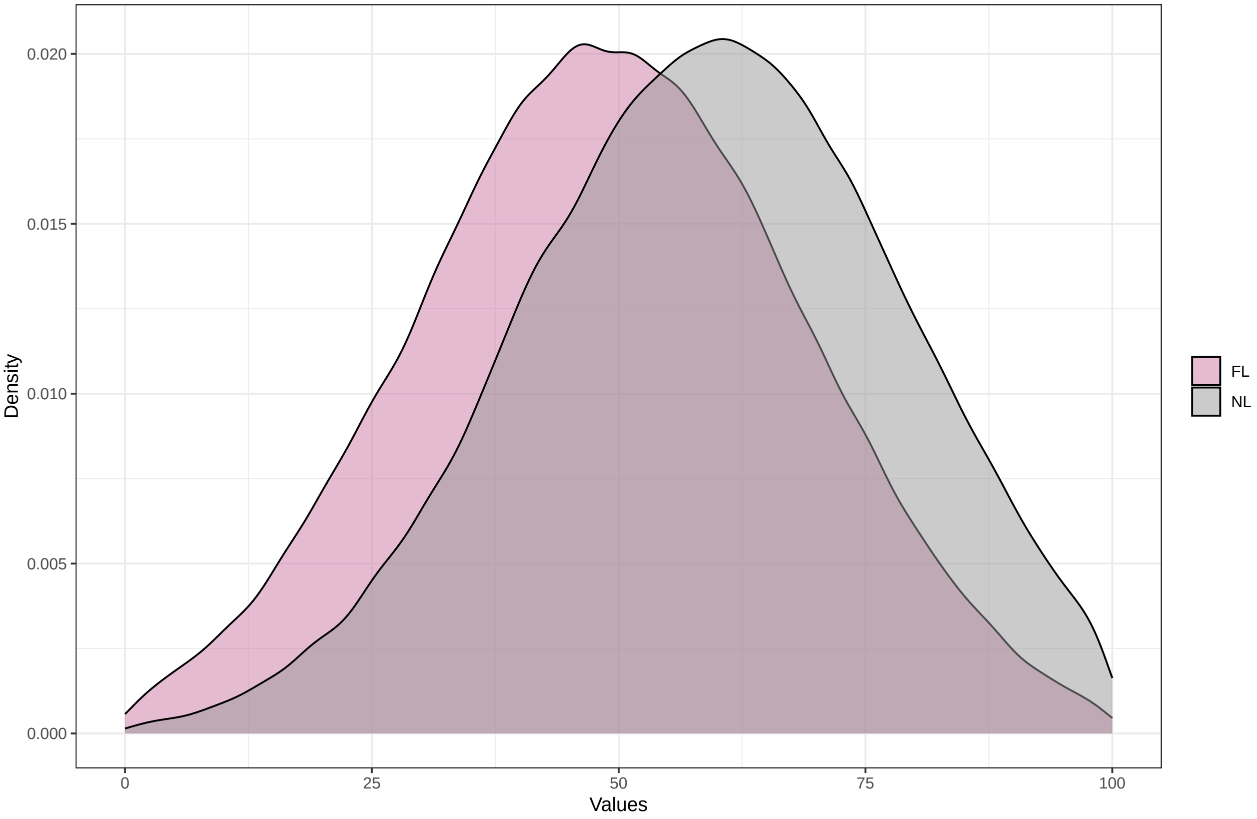
```
[1] 18.70429
```

```r
mean(b); sd(b)  # FL group
```

```
[1] 49.1039
```

```
[1] 19.03031
```

```r
# Plot the densities
data <- data.frame(
  value = c(a, b),
  group = factor(rep(c("NL", "FL"), each = length(a)))
)

ggplot(data, aes(x = value, fill = group)) +
  geom_density(alpha = 0.5) +
  labs(x = "Values", y = "Density") +
  scale_fill_okabe_ito(order=c(7,8)) +
  theme_bw() +
  theme(legend.title = element_blank())
```



```
# Cohen's D
library(effectsize)
cohens_d(a,b)
```

```
Cohen's d |        95% CI
------------------------
0.52      | [0.51, 0.53]

- Estimated using pooled SD.
```

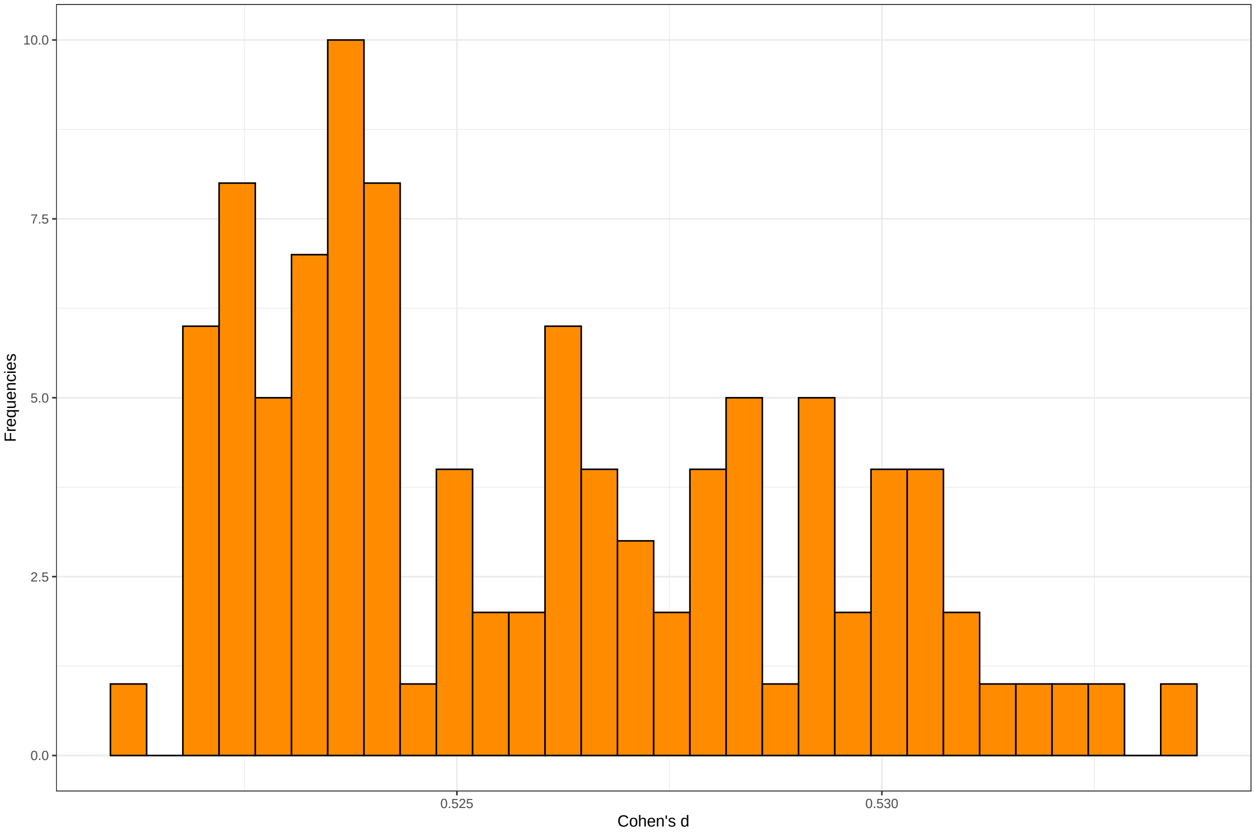
We are uncertain about the exact mean for the NL group in the upcoming experiment, as some features are expected to change. However, we can assume that the NL group mean will vary within a range of -5 to +5 points compared to the previously observed data. Since the SD is likely to decrease when the mean shifts toward the extremes of the distribution and increase when the mean moves toward the center, we will use a fixed SD parameter value of 20 for all the simulated truncated normal distributions.

The mean of the FL group is set to be 10 points lower than the corresponding NL group mean, and we maintain the SD parameter at 20. The ES for each pair of NL and FL distributions is calculated using Cohen's d. The following code simulates NL and FL group data for a range of NL group means, specifically from 55 to 65. The simulation ensures that the mean difference between the two groups is always at least 10 points, adjusting the FL group mean parameter as necessary to meet this condition.

The histogram displays the distribution of the calculated ESs, and a line plot shows how the ES changes as the NL group mean shifts. The results suggest that the hypothesized ES is medium, as in all simulations, Cohen's d did not drop below 0.5.

```
# Function to extract ES in Cohen's d
calculate_effect_size <- function(NLmean, FLdiff, n=10000, sd=20) {
  v <- rep(NA, 5)
  NLc_data <- round(tnorm_f(n, NLmean, sd))  # NL group
  FLc_data <- round(tnorm_f(n, NLmean - FLdiff, sd))  #  FL group
  x <- 0

  # Ensuring that the difference between means is at least 9.9
  if ((mean(NLc_data) - mean(FLc_data) < 9.9)) {
    while (mean(NLc_data) - mean(FLc_data) < 9.9) {
      x <- x + 0.0005
      FLc_data <- round(tnorm_f(n, NLmean - (FLdiff + x), sd))

      # Break when the mean difference is >= 9.9
      if (mean(NLc_data) - mean(FLc_data) >= 9.9) {
        break
      }
    }
  }

  d <- as.numeric(cohens_d(NLc_data, FLc_data)[1])
  v[1] <- d
  v[2] <- mean(NLc_data)
  v[3] <- sd(NLc_data)
  v[4] <- mean(FLc_data)
  v[5] <- sd(FLc_data)

  return(v)
}

# Defining range of NL means
NLmeans <- seq(55, 65, by = 0.1)

# Preparing vectors to store the results
mean1 <- rep(NA, length(NLmeans))
mean2 <- rep(NA, length(NLmeans))
sd1 <- rep(NA, length(NLmeans))
sd2 <- rep(NA, length(NLmeans))
cohen <- rep(NA, length(NLmeans))

# Simulate through each possible NL mean
for(i in 1:length(NLmeans)) {
  vector <- calculate_effect_size(NLmeans[i], 10, n = 100000)
```

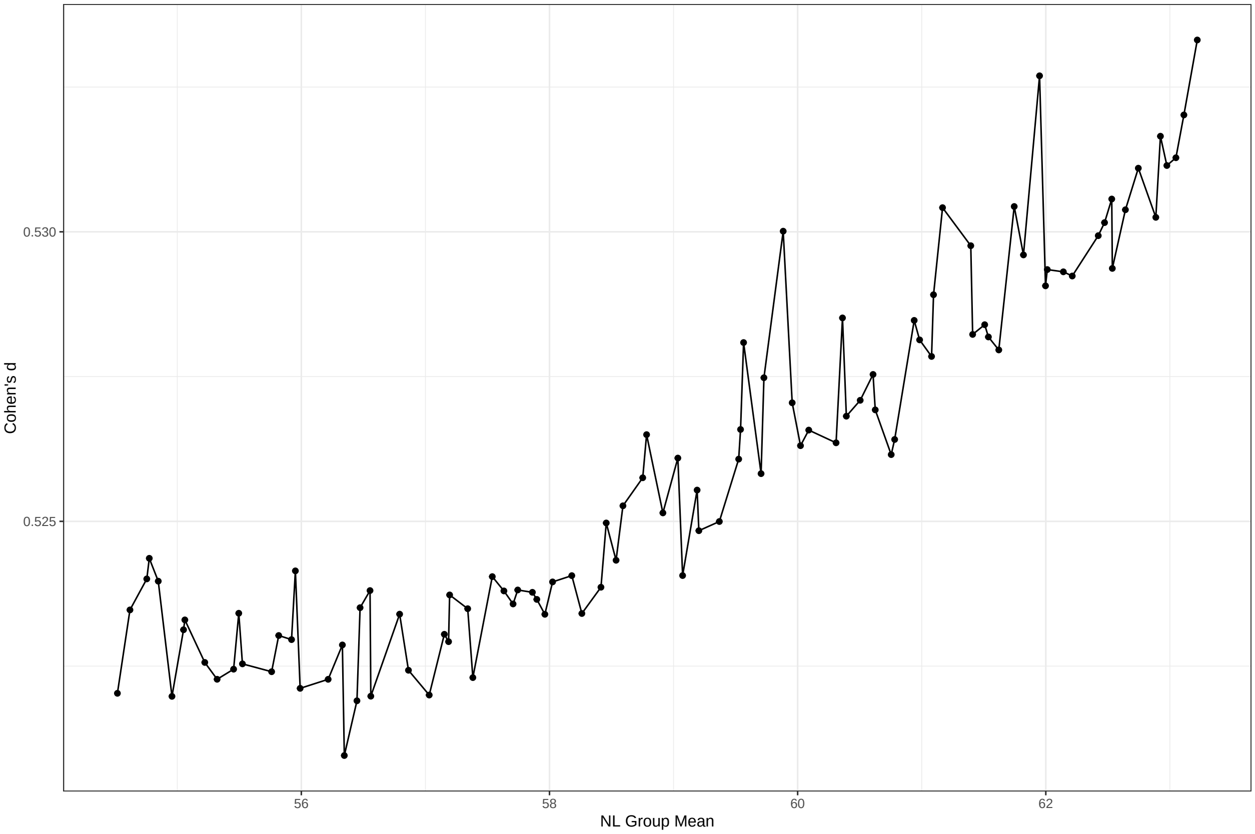
```r
  cohen[i] <- vector[1]
  mean1[i] <- vector[2]
  mean2[i] <- vector[4]
  sd1[i] <- vector[3]
  sd2[i] <- vector[5]
}

# Store the final results
effect_size_df <- data.frame(
  NLmean = mean1,
  FLmean = mean2,
  NLsd = sd1,
  FLsd = sd2,
  d = cohen
)

# Cohen's d distribution
ggplot(effect_size_df, aes(x = d)) +
  geom_histogram(fill="darkorange", col="black")+
  labs(y = "Frequencies", x = "Cohen's d") +
  theme_bw()
```



```
summary(effect_size_df$d)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.5210  0.5234  0.5253  0.5259  0.5284  0.5333 
```

```
#  Cohen's d
ggplot(effect_size_df, aes(x = mean1, y = d)) +
  geom_line() +
  geom_point() +
  labs(x = "NL Group Mean", y = "Cohen's d") +
  theme_bw()
```



2.4 Analysis prior

In the experiment by Dalla Bona and Vicovaro (2024), we observed that approximately 36% of participants tend to select Likert scale responses divisible by 10 (e.g., 20, 30, 40, 50, etc.), while about 50% of participants prefer responses divisible by 5 (e.g., 20, 25, 30, 35, etc.).

```
# Causality evaluations divisible by 5 and by 10
 sum(vector_general %% 5 == 0) / length(vector_general) *100
```

```
[1] 50.5
```

```
 sum(vector_general %% 10 == 0) / length(vector_general) *100
```

```
[1] 36.5
```

This suggests that Likert scale points ending in 5 or 10 may act as anchor points for participants' responses. This observation can give rise to a heuristic type of reasoning to elicit an analysis prior distribution of ES for our experiment: we can use these anchor points to model the mean difference between the two groups, assuming that a shift in causal evaluation — triggered by an FL manipulation — could lead participants to anchor their responses to lower key points on the scale.

Referring to the original study by @diaz2019thinking, the difference between the two groups in the first experiment was greater than 20 points (NL group M = 64.5, SD = 15.12; FL group M = 40.75, SD = 18.43), which we interpret as a shift of 4 or 5 anchor points (20–25 Likert points). In the second experiment, which is more similar to our upcoming study, the difference between the two groups was slightly less than 20 points (NL group M = 63, SD = 20.49; FL group M = 44.30, SD = 26.18), which we interpret as a shift of 3 or 4 anchor points (15–20 Likert points).

We must also account for the possibility that the estimated difference could be inflated due to a reduced sample size. Therefore, a reduction of 1 or 2 anchor points seems plausible.

We plan to create an analysis prior considering a minimum reduction of 1 anchor (5 points, which corresponds approximately to a 0.2 Cohen's d effect size from meta-analysis) and a maximum shift of 4 anchors (20 points, corresponding approximately to a 1 Cohen's d effect size from the original experiment). We will generate the means of the two groups using the interval between these two anchors (5 to 20 points, with a step size of 1). We will use the entire interval between anchors, as the mean reflects only a theoretical value.

As in the previous simulation, we assume that the NL group mean will vary within a range of -5 to +5 points compared to the previously observed data.

Additionally, we hypothesize that the FL group will exhibit greater variability not only due to a reduction in the mean of the truncated normal distribution but also because individuals in the FL group may show differing responses to the FL exposure. To account for this increased variability, we will simulate different FL group standard deviations, ranging from 20 to 25 (following an increase of approximately 5 points as in the original experiment), with a step size of 1.

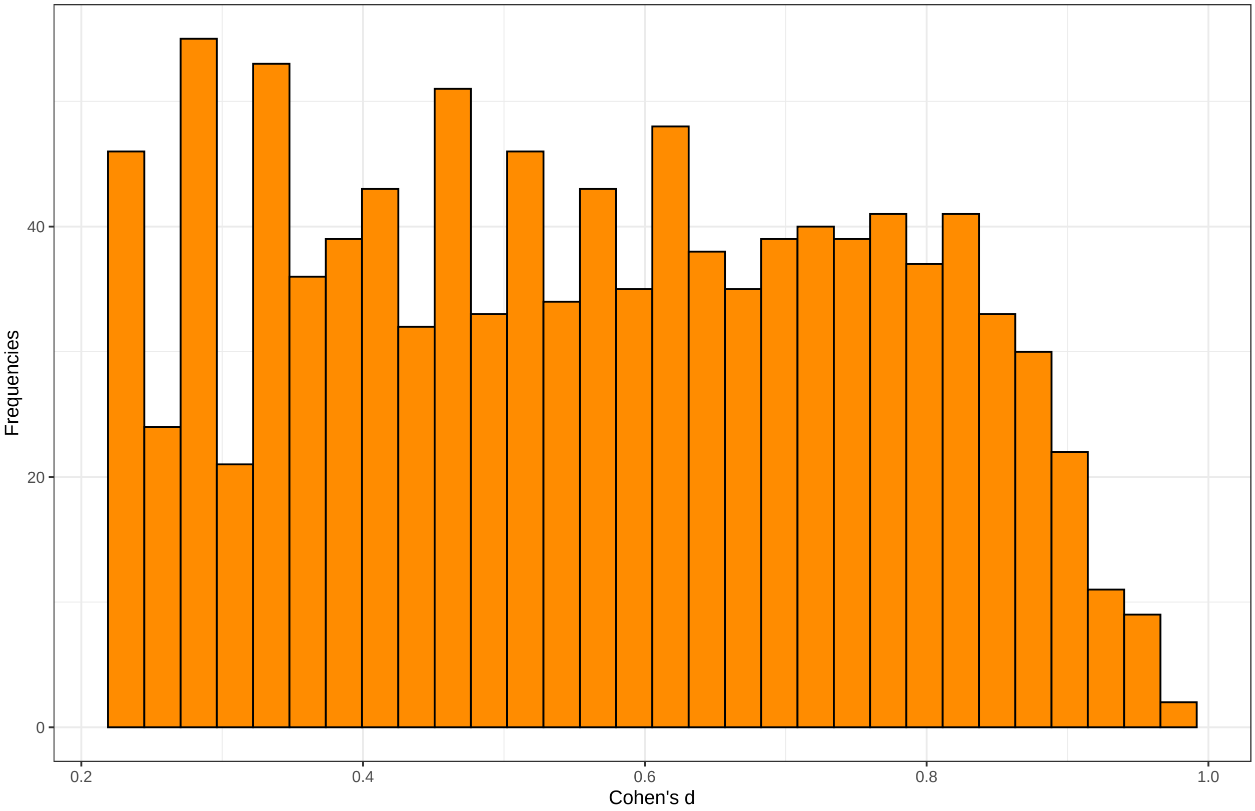
Finally, we will store the results in a dataframe that includes Cohen's d values, which will serve as prior knowledge on the expected effect size (ES) for future analyses. We will also store the updated mean parameterization for the FL group. In each simulation, the FL group's mean parameter will be adjusted to ensure that the difference between the NL and FL group means is always at least the desired mean difference. This updated mean will be crucial for design analysis, as it ensures the simulation maintains the required mean difference and provides an accurate representation of the distributions for subsequent simulations.

This prior will serve as a reference for the analysis prior for a Bayes Factor Design Analysis (the density plot is shown in blue, with the classical "objective analysis prior" shown in red). It reflects the uncertainty about where we believe the true effect size lies, as the meta-analysis suggests a small effect size, whereas the original experiment suggests a large effect size.

Our belief is that the true effect size lies somewhere in between and this belief will be incorporated in the design analysis.

```
# Simulate with varying mean differences
calculate_effect_size_TSD <- function(NLmean, NLsd, treatment_sd, desired_diff = 10, n =
  # NL group with SD = 20
  NLc_data <- round(tnorm_f(n, NLmean, NLsd))
  # FL group with varying SD
  FLc_data <- round(tnorm_f(n, NLmean - desired_diff, treatment_sd))

  # Ensuring that the difference of means is at least the difference required
  x <- 0
  while (abs(mean(NLc_data) - mean(FLc_data)) < (desired_diff - 0.04) & abs(mean(NLc_dat

    if (mean(NLc_data) > mean(FLc_data)) {
      x <- x + 0.05
      FLc_data <- round(tnorm_f(n, NLmean - (desired_diff + x), treatment_sd))
    } else {
      x <- x - 0.05
      FLc_data <- round(tnorm_f(n, NLmean - (desired_diff + x), treatment_sd))
    }
  }

  # Cohen's d
  d <- as.numeric(cohens_d(NLc_data, FLc_data, pooled_sd = TRUE)[1])

  # Return the results and new parameters t
  return(c(d, NLmean - (desired_diff + x), mean(NLc_data), sd(NLc_data), NLmean, mean(FL
}

# NL means (55 to 65)
control_means <- seq(55, 65, by = 1)

#  FL SDs (20 to 25)
treatment_sds <- seq(20, 25, by = 1)

# NL SD
control_sd <- 20

# Vector of mean differences
desired_diffs <- c(5:20)

# Preparing vectors to store results
mean1_var <- rep(NA, length(control_means) * length(treatment_sds) * length(desired_diff
mean1_true <- rep(NA, length(control_means) * length(treatment_sds) * length(desired_dif
mean2_var <- rep(NA, length(control_means) * length(treatment_sds) * length(desired_diff
mean2_true <- rep(NA, length(control_means) * length(treatment_sds) * length(desired_dif
sd1_var <- rep(NA, length(control_means) * length(treatment_sds) * length(desired_diffs)
sd2_var <- rep(NA, length(control_means) * length(treatment_sds) * length(desired_diffs)
cohen_var <- rep(NA, length(control_means) * length(treatment_sds) * length(desired_diff
sd_true <- rep(NA, length(control_means) * length(treatment_sds) * length(desired_diffs)
DI <- rep(NA, length(control_means) * length(treatment_sds) * length(desired_diffs))
counter <- 1

# Simulate for each combination
for (mean_val in control_means) {
  for (treatment_sd_val in treatment_sds) {
    for (diff in desired_diffs) {
      vector <- calculate_effect_size_TSD(mean_val, control_sd, treatment_sd_val, desire
      cohen_var[counter] <- vector[1]
      mean1_true[counter] <- vector[3]
      mean1_var[counter] <- vector[5]
      mean2_true[counter] <- vector[6]
      mean2_var[counter] <- vector[2]
      sd1_var[counter] <- vector[4]
      sd2_var[counter] <- vector[7]
      sd_true[counter] <- vector[8]
      DI[counter] <- diff

      counter <- counter + 1
    }
  }
}
```

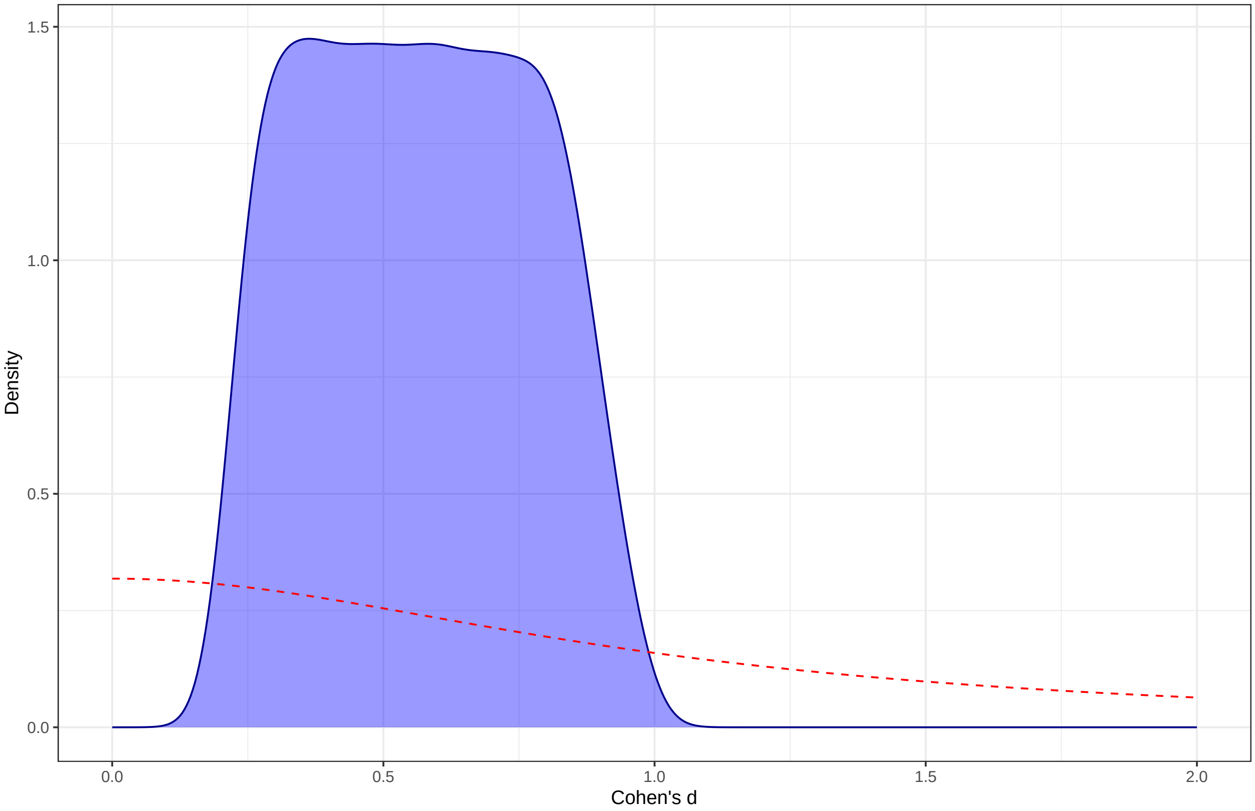
```r
effect_size_analysisP <- data.frame(
  NLmean = mean1_var,
  NLmeantrue = mean1_true,
  FLmean = mean2_var,
  FLmeantrue = mean2_true,
  NLsd = sd1_var,
  FLsd = sd2_var,
  d = cohen_var,
  sd = sd_true,
  diff=DI
)

# Cohen's d
ggplot(effect_size_analysisP, aes(x = d)) +
  geom_histogram(fill="darkorange", col="black") +
  labs(y = "Frequencies", x = "Cohen's d") +
  theme_bw()
```



```
summary(effect_size_analysisP$d)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
 0.2198  0.3836  0.5624  0.5642  0.7344  0.9668 
```

```
# Priors (Our analysis prior and the Cauchy analysis prior)
ggplot(effect_size_analysisP, aes(x = d)) +
  geom_density(fill="blue", alpha=.4, col="darkblue") +
  stat_function(fun = dt, args = list(df=1), lty="dashed", col="red") +
  xlim(c(0,2)) +
  labs(x = "Cohen's d", y = "Density") +
  theme_bw()
```



```
save(effect_size_analysisP, file="Effsize.Rda")
```

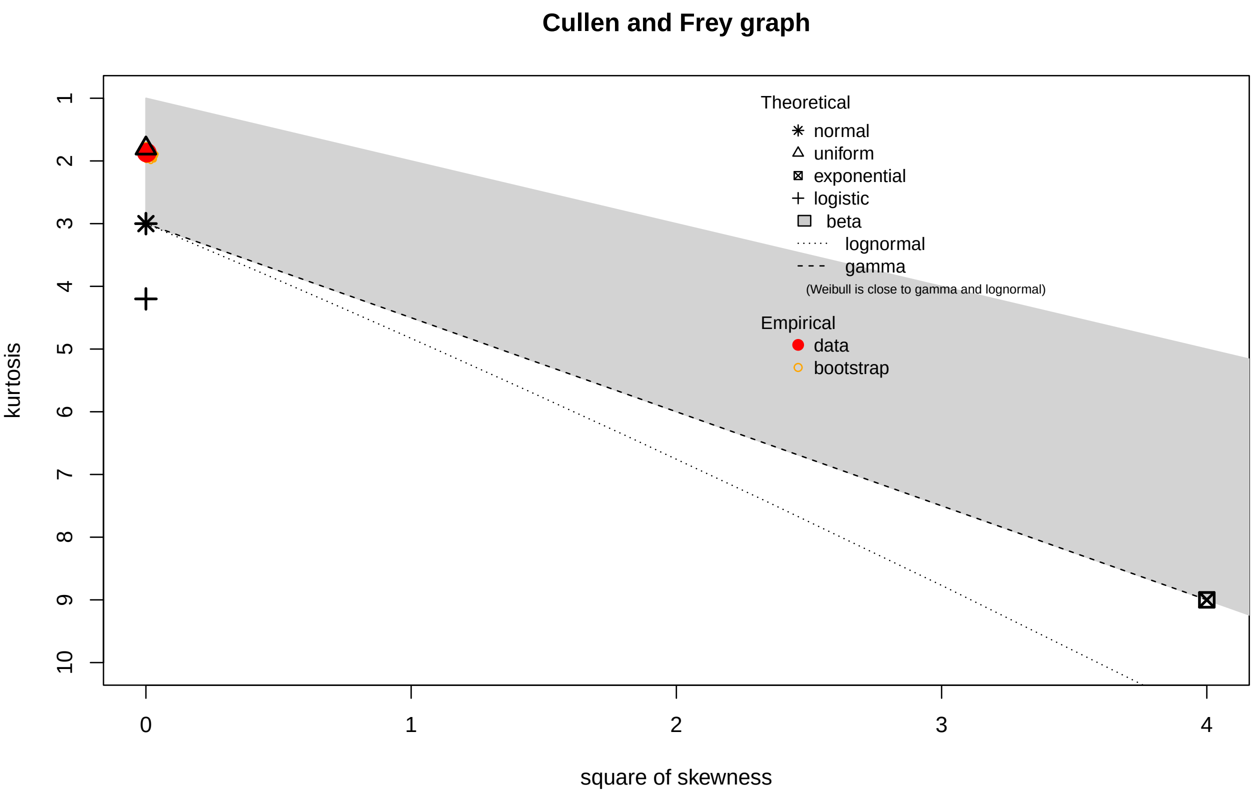
Our analysis prior under H1 can be described as a uniform (min = .22, max = .96), but we think that a personalized function could better describe the smoothness of the empirical distribution tails. For this reason, we came up with the following formula, normalized so that the area under the curve is equal to 1, and the .95 probability lies between the two inflection points (.2 and .9):

$$f(x) = \left(\frac{1}{1+e^{-40(x-0.2)}} + \frac{1}{1+e^{35(x-0.9)}} - 1\right) \cdot \frac{10}{7}$$

The analytical formula will be used as the analysis prior under H1, whereas the empirical distribution will be used as the design prior under which are going to sample data to calculate the necessary sample size for the experiment.

Our analysis prior under H0 is that there are no difference between means (i.e., ES is equal to 0).

```
#Comparison with a Uniform distribuion
descdist(effect_size_analysisP$d, boot = 1000)
```



```
summary statistics
------
min:  0.2197644   max:  0.9667542
median:  0.5623741
mean:  0.5641723
estimated sd:  0.2008705
estimated skewness:  0.05674132
estimated kurtosis:  1.86834
```

```
# Analysis prior analytic formula
analysis_priors <- function(x, lower = 0.2, upper = .9, steepness = 35) {
  ((1 / (1 + exp(-40 * (x - lower))) + 1 / (1 + exp(steepness * (x - upper)))) - 1) * (1
}
```

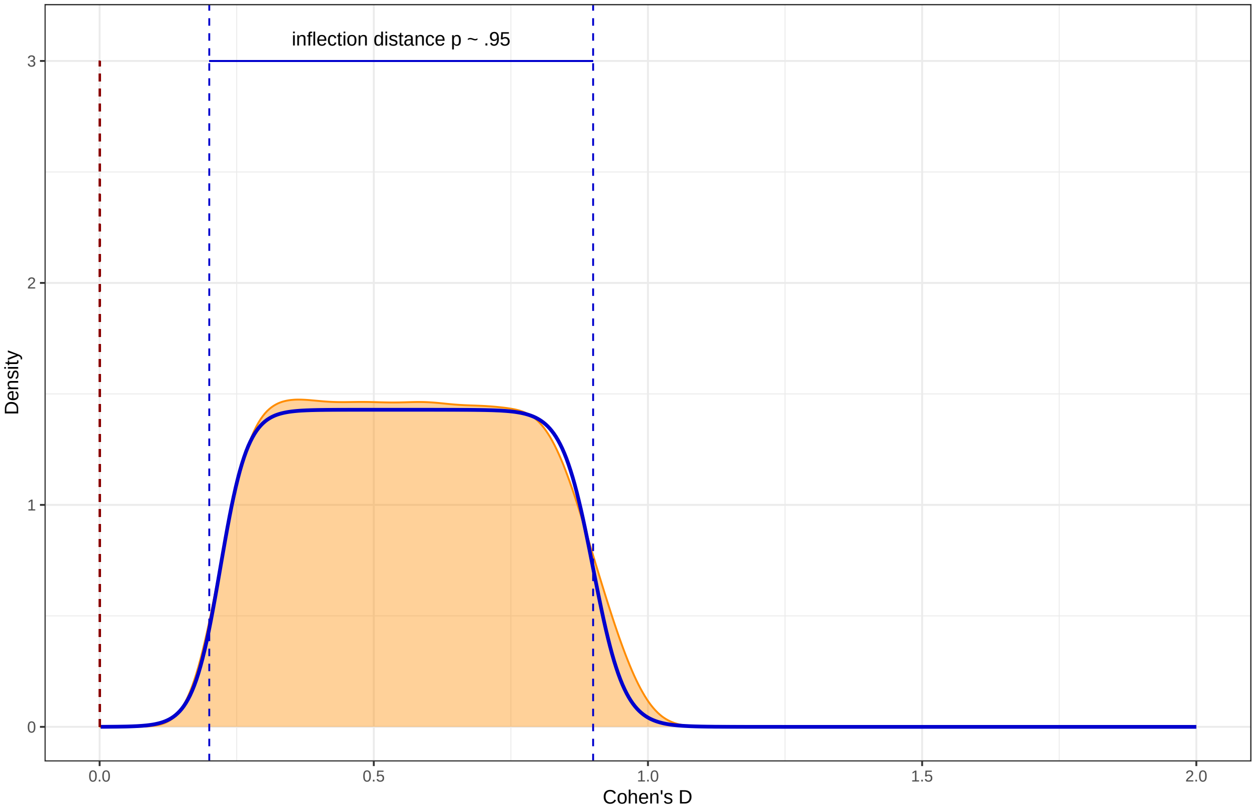
```
#Area under the curve between 0 and 1
round(integrate(analysis_priors, 0, 1)$value,2)
```

```
[1] 1
```

```
#Area under the curve between .2 and .2
round(1 - (integrate(analysis_priors, -10, .2)[[1]] +
  integrate(analysis_priors, .9, 10)[[1]]), 2)
```

```
[1] 0.95
```

```
# Showing the analytical function and the empirical distribution
delta_values <- seq(-2, 2, length.out = 1056)

analysis_prior <- ((1 / (1 + exp(-40 * (delta_values - .22))) + 1 / (1 + exp(35 * (delta

data <- data.frame(delta = delta_values,
                   analysis_prior = analysis_prior)

ggplot(effect_size_analysisP, aes(x = d)) +
  geom_density(fill="darkorange", alpha=.4, col="darkorange") +
  geom_line(aes(x = data$delta, y = data$analysis_prior), color = "blue3", size = 1) +
  geom_vline(xintercept = .2, lty="dashed", col="blue3") +
  geom_segment(aes(x = .2, xend = .9, y = 3, yend = 3), col="blue3") +
  geom_segment(aes(x = 0, xend = 0, y = 0, yend = 3), col="darkred", lty="dashed") +
  annotate("text", x = .55, y = 3.1, label = "inflection distance p ~ .95") +
  geom_vline(xintercept = .9, lty="dashed", col="blue3") +
  xlim(c(0, 2)) +
  labs(x = "Cohen's D", y = "Density") +
  theme_bw()
```



Arnold, J. B. (2024). *Ggthemes: Extra themes, scales and geoms for 'ggplot2'*. https://CRAN.R-project.org/package=ggthemes

Auguie, B. (2017). *gridExtra: Miscellaneous functions for "grid" graphics*. https://CRAN.R-project.org/package=gridExtra

Barrett, M. (2024). *Ggokabeito: 'Okabe-ito' scales for 'ggplot2' and 'ggraph'*. https://github.com/malcolmbarrett/ggokabeito

Ben-Shachar, M. S., Lüdecke, D., & Makowski, D. (2020). Effectsize: Estimation of effect size indices and standardized parameters. *Journal of Open Source Software*, *5*(56), 2815. https://doi.org/10.21105/joss.02815

Dalla Bona, S., & Vicovaro, M. (2024). Does perceptual disfluency affect the illusion of causality? *Quarterly Journal of Experimental Psychology*, *77*(8), 1727–1744. https://doi.org/10.1177/17470218231220928

Delignette-Muller, M. L., & Dutang, C. (2015). Fitdistrplus: An r package for fitting distributions. *Journal of Statistical Software*, *64*(4), 1–34. https://doi.org/10.18637/jss.v064.i04

Díaz-Lago, M., & Matute, H. (2019). Thinking in a foreign language reduces the causality bias. *Quarterly Journal of Experimental Psychology*, *72*(1), 41–51. https://doi.org/10.1177/1747021818755326

Frey, H. C., & Cullen, A. C. (1995). *Distribution development for probabilistic exposure assessment* (Vol. 11, p. 95).

Grosjean, P., & Ibanez, F. (2024). *Pastecs: Package for analysis of space-time ecological series*. https://CRAN.R-project.org/package=pastecs

Kay, M. (2024). Ggdist: Visualizations of distributions and uncertainty in the grammar of graphics. *IEEE Transactions on Visualization and Computer Graphics*, *30*(1), 414–424. https://doi.org/10.1109/TVCG.2023.3327195

Masked Citation. (n.d.). *Masked Title*.

Pastore, M., Alaimo Di Loro, P., Mingione, M., & Calcagni, A. (2022). *Overlapping: Estimation of overlapping in empirical distributions*. https://CRAN.R-project.org/package=overlapping

Peirce, J., Gray, J. R., Simpson, S., MacAskill, M., Hochenberger, R., Sogo, H., Kastman, E., & Lindelov, J. K. (2019). PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods*, *51*, 195–203. https://doi.org/10.3758/s13428-018-01193-y

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org

Wickham, H., & Bryan, J. (2023). *Readxl: Read excel files*. https://CRAN.R-project.org/package=readxl