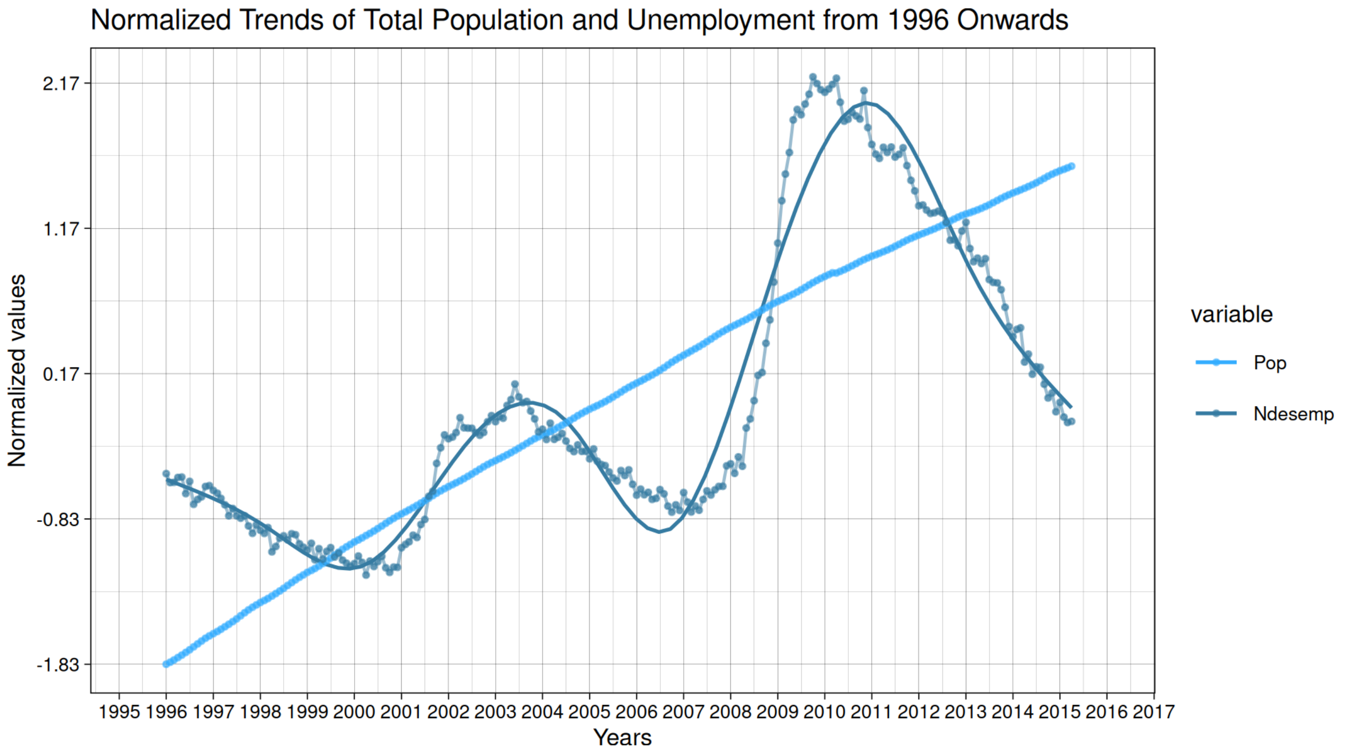
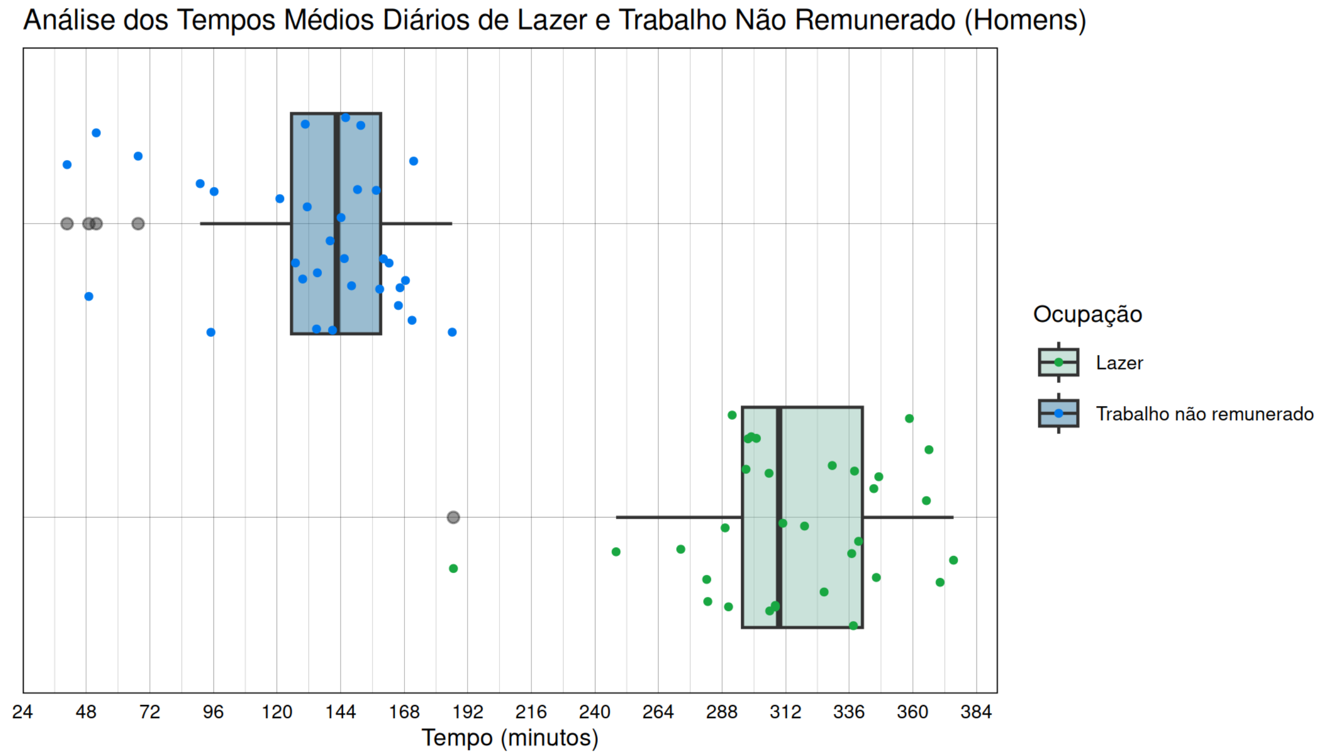
Question 1




```r
# Load required libraries - install.packages("pacman")
pacman::p_load(readxl, tidyverse, reshape2)

# Function to standardize a variable
standardize <- function(x) {
  (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)
}

# Read data and build dataframe
data <- read_xlsx("data/econ.xlsx")
data$tempo <- as.Date(data$tempo, format = "%Y-%m-%d")

df <- data %>%
  filter(tempo >= as.Date("1996-01-01")) %>%
  select(tempo, pop, ndesemp) %>%
  mutate(across(c(pop, ndesemp), standardize))  # Apply variable transformation

# Convert to a melted data frame
meltdf <- melt(df, id="tempo")

# Create plot
plot <- ggplot(data = meltdf, aes(x=tempo, y=value, colour=variable, group=variable)) +
  geom_line(linewidth=0.5, alpha=0.5, na.rm = TRUE) +
  geom_smooth(data = subset(meltdf, variable == "ndesemp"), method = "gam", se = FALSE, linewidth = 0.6) +
  geom_point(size=0.6, alpha=0.6, na.rm = TRUE)

# Apply theming, labels and title
scaleFUN <- function(x) sprintf("%.2f", x)

final_plot <- plot +
  ggtitle("Normalized Trends of Total Population and Unemployment from 1996 Onwards") +
  xlab("Years") + ylab("Normalized values") +
  scale_color_manual(values=c("#34adff","#367ba2"), labels=c("Pop", "Ndesemp")) +
  theme(legend.position = "right", legend.title = element_blank()) +
  theme_linedraw(base_size = 8) +
  scale_x_date(limits=c(as.Date("1995-06-01"), as.Date("2016-01-01")), date_breaks="1.5 years", date_labels="%Y") +
  scale_y_continuous(labels=scaleFUN, breaks=seq(min(meltdf$value, na.rm=TRUE), max(meltdf$value, na.rm=TRUE), by=1))

# Display the resulting plot
print(final_plot)
```

Source Code Q1

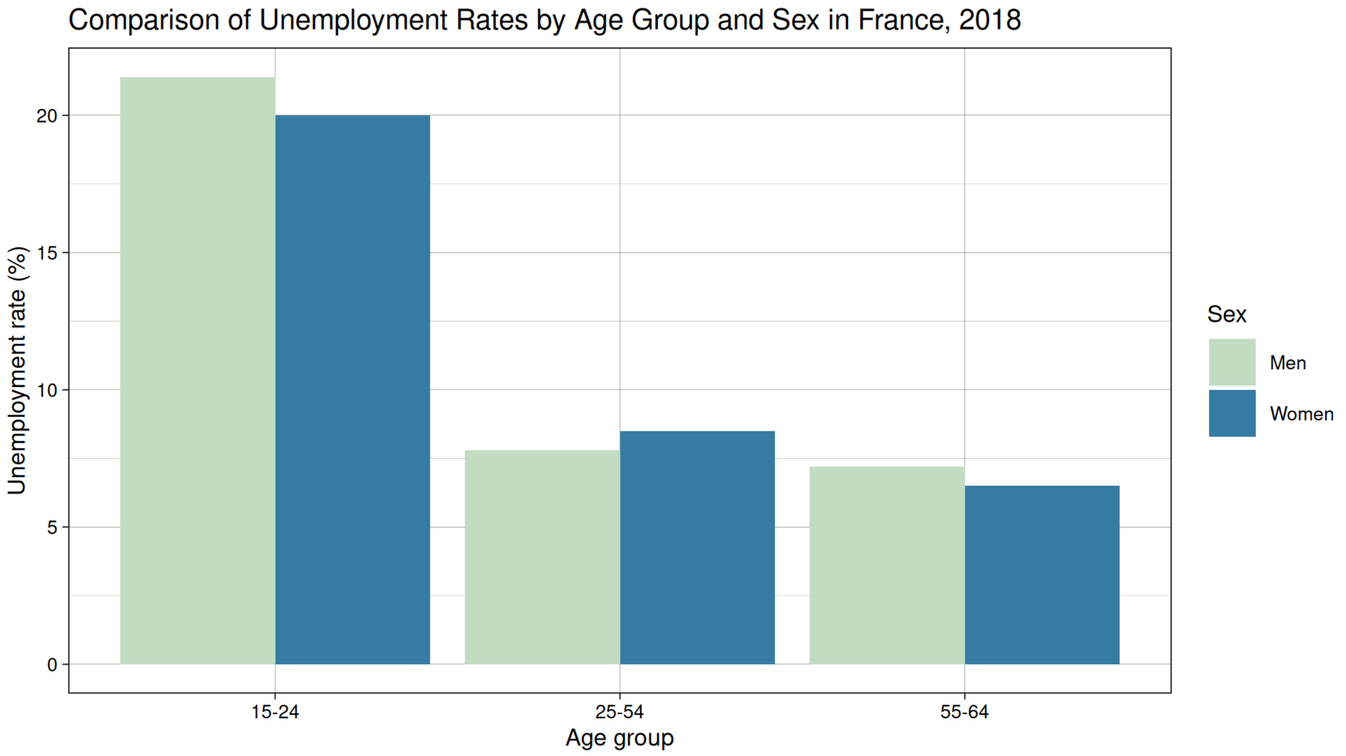
Question 2




```
# Load required libraries - install.packages("pacman")
pacman::p_load(tidyverse) # Includes ggplot2

# Read and filter the data
df <- read_csv("data/TIME_USE_24092022.csv") %>%
  filter(País !='África do Sul' &
           Sexo == 'Homens' &
          (Ocupação=='Lazer' | Ocupação=='Trabalho não remunerado'))

# Remove unwanted column
df <- df %>% select(-Sexo)

# Function to format the y-axis labels
scaleFUN <- function(x) sprintf("%.0f", x)

# Function to create the y-axis breaks
breaksFUN <- function(x) seq(min(x, na.rm = TRUE), max(x, na.rm = TRUE), by = 24)

# Plot the data
plot <- ggplot(df, aes(x = Ocupação, y = Tempo)) +
  geom_boxplot(aes(fill=Ocupação), alpha=0.5) +
  geom_jitter(aes(color=Ocupação), size = 0.75, shape = 19) +
  coord_flip() + theme_linedraw(base_size = 8)

# Add labels and title
final_plot <- plot + ggtitle("Análise dos Tempos Médios Diários de Lazer e Trabalho Não Remunerado (Homens)") +
  xlab("") + ylab("Tempo (minutos)") +
  theme(axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        strip.background = element_blank(),
        strip.text.y = element_blank()) +
  scale_y_continuous(breaks = breaksFUN, labels = scaleFUN) +
  scale_fill_manual(values=c("#95c6b5", "#367ba2")) +
  scale_color_manual(values=c("#19a742", "#0079ee"))

# Display the resulting plot
print(final_plot)
```

Source Code Q2

Question 3




```r
# Load required libraries - install.packages("pacman")
pacman::p_load(ggplot2, dplyr)

# Read in data
data <- read.delim("data/GENDER_EMP_19032023152556091.txt")

# Filter for wanted data
f_data <- data %>%
  filter(Country == "France" &
         Time == 2018 &
         IND == "EMP3" &
         Age.Group %in% c("15-24", "25-54", "55-64") &
         Sex %in% c("Men", "Women"))

# Specify the order of the x-axis
f_data$Age.Group <- factor(f_data$Age.Group, levels = c("15-24", "25-54", "55-64"))

# Develop grouped barplot
final_plot <- ggplot(f_data, aes(x = Age.Group, y = Value, fill = Sex)) +
  geom_bar(position="dodge", stat="identity") +
  ggtitle("Comparison of Unemployment Rates by Age Group and Sex in France, 2018") +
  xlab("Age group") + ylab("Unemployment rate (%)") +
  scale_fill_manual(values = c("Men" = "#C2DCC2", "Women" = "#367ba2")) +
  theme_linedraw(base_size = 8)

print(final_plot) # Display the resulting plot
```

Source Code Q3

Question 4

Valor final: 0.0613

```r
# Fix the random seed for reproducibility
set.seed(2904)

sample_size <- 3117
rate <- 9.5

sample <- rexp(sample_size, rate)   # Generate sample
s <- cumsum(sample)                 # Time at which each event occured
T <- ceiling(max(s))                # Integer value >= time of last event
event_count <- table(floor(s))      # Count the frequency of each value in
                                    # the samples cummulative sum

# Find sample mean and distribuition expected value
mean_counts <- mean(event_count)
expected_value <- rate

# Find absolute deviation
absolute_deviation <- abs(mean_counts - expected_value)
round(absolute_deviation, 4)
```

Source Code Q4

Question 5

Valor final: 0.3389

```r
# Fix the random seed for reproducibility
set.seed(1274)

generate_geom <- function(p) {
  u <- runif(1) # Step 1: Generate a uniform random variable
  x <- floor(log(1 - u) / log(1 - p)) # Step 2: Apply the inverse CDF
  return(x)
}

p <- 0.2
n <- 1185
samples <- replicate(n, generate_geom(p))
sample_mean <- mean(samples)
sample_sd <- sd(samples)

sample_above_mean <- samples[samples > sample_mean]
proportion <- sum(sample_above_mean > (sample_mean + sample_sd))
proportion <- proportion / length(sample_above_mean)

print(round(proportion,4))
```

Source Code Q5

Question 6

Valor final: 0.0209

```r
# Find the probability of the first digit being 1 or 6
prob <- log10(1 + 1/1) + log10(1 + 1/6)

# Define the range of exponents for the powers of 2
exponent_range <- 9:26

# Calculate the powers of 2
powers_of_two <- 2^exponent_range

# Convert to character strings to extract the first digit
first_digits <- substr(powers_of_two, 1, 1)

# Calculate the fraction of powers whose first digit is 3 or 9
fraction <- sum(first_digits %in% c("1", "6")) / length(powers_of_two)

# Find the absolute deviation of the specified parameter
abs_deviation <- abs(prob - fraction)

print(round(abs_deviation, 4))
```

Source Code Q6

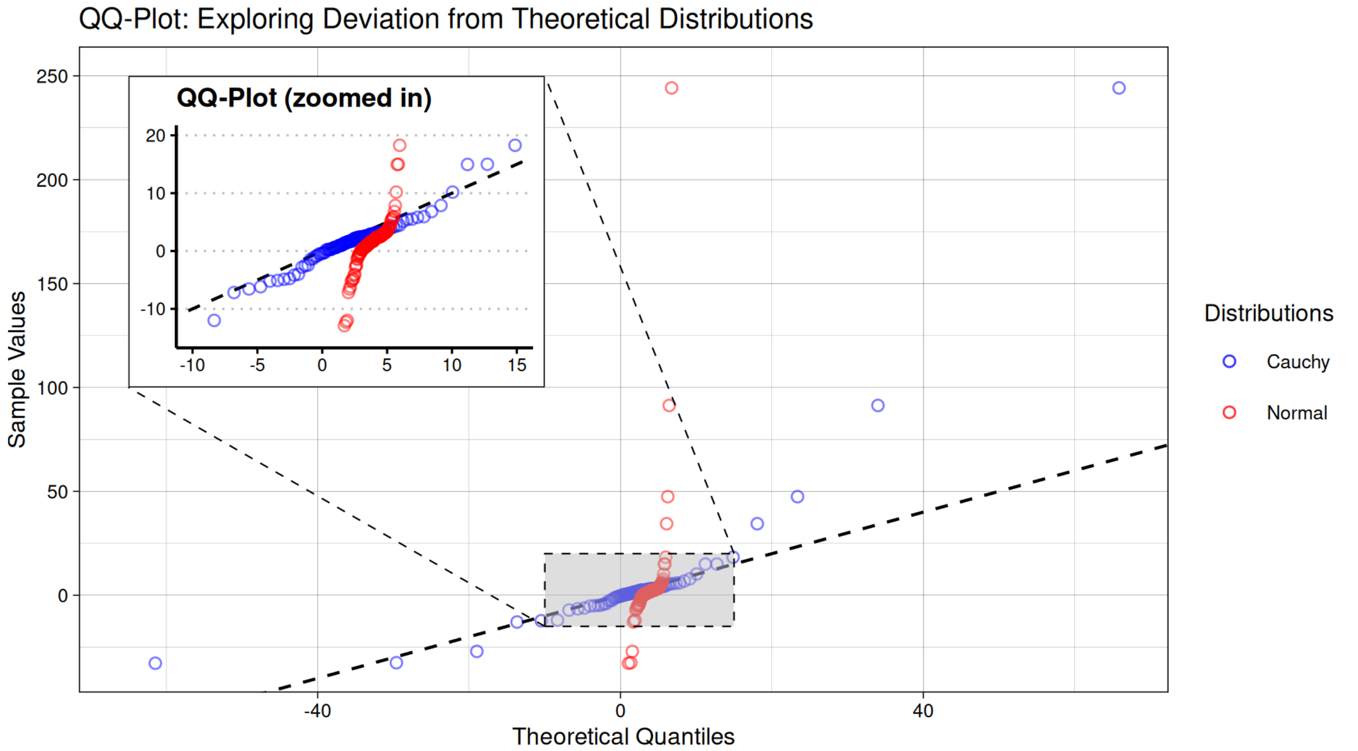
Question 7

Valor final: 0.2125

```r
# Fix the random seed for reproducibility
set.seed(1276)

# Generate m samples each of size n from a standard normal distribution
m <- 2947
n <- 12
samples <- replicate(m, rnorm(n))

# Calculate the sum of squares for each sample and the 0.39 quantile
sum_sq <- apply(samples^2, 2, sum)
sample_quantile <- quantile(sum_sq, 0.39, type = 2)

# Calculate the 0.39 quantile for the theoretical chi-square distribution
theoretical_quantile <- qchisq(0.39, df = n)

# Calculate the absolute difference
abs_diff <- abs(sample_quantile - theoretical_quantile)

# Print the absolute difference
print(round(abs_diff, 4))
```

Source Code Q7

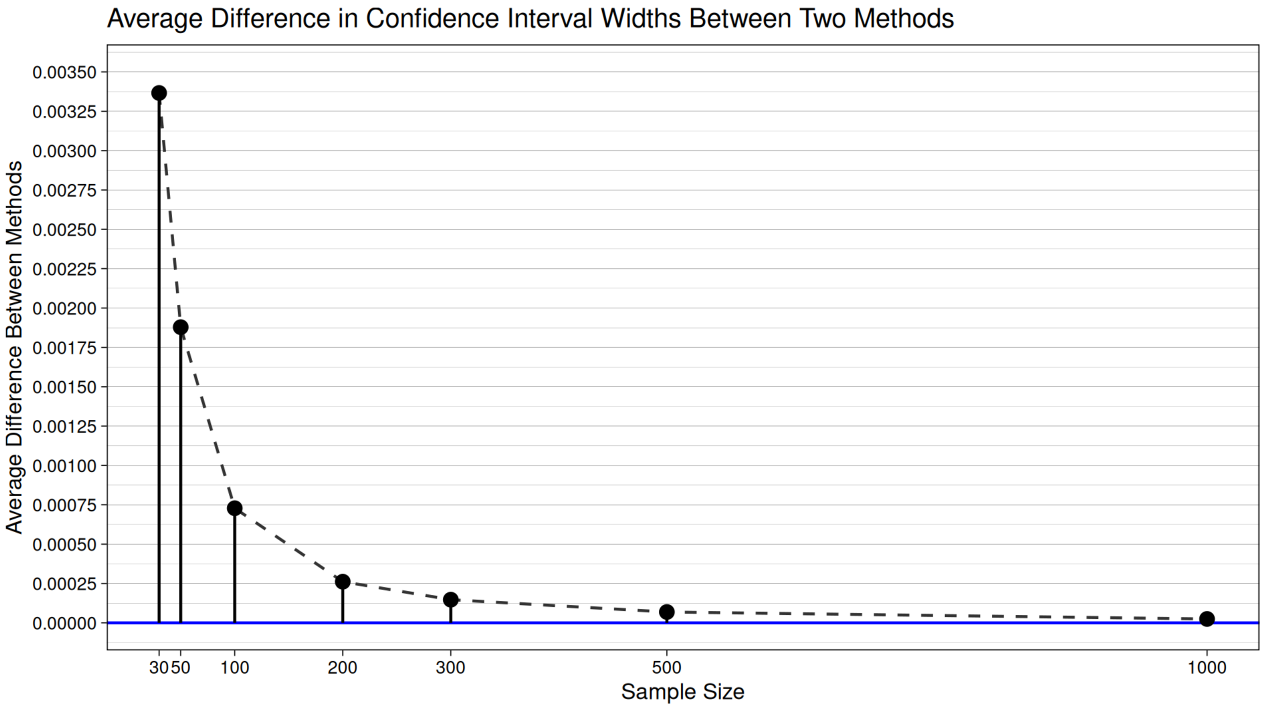
Question 8




```r
# Load required libraries - install.packages("pacman")
pacman::p_load(ggplot2, ggthemes)

# Set parameters
set.seed(1338)  # Fix the random seed for reproducibility
location <- 2.2 # Cauchy distribution parameters
scale <- 1.6
mean <- 3.9      # Normal distribution parameters
sd <- sqrt(1.4)

# Generate sample
sample_size <- 124
sample <- rcauchy(sample_size, location, scale)
sample <- sort(sample) # Sort sample

# Find theoretical quantiles
quantiles <- (1:sample_size) / (sample_size + 1)
theoretical_quantiles_cauchy <- qcauchy(quantiles, location, scale)
theoretical_quantiles_norm <- qnorm(quantiles, mean, sd)

# Create main plot
main_plot <- ggplot() + labs(title = "QQ-Plot: Exploring Deviation from Theoretical Distributions") +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
  geom_point(aes(x = theoretical_quantiles_cauchy, y = sample, colour="Cauchy"), shape=1, alpha=0.5, size=1.5) +
  geom_point(aes(x = theoretical_quantiles_norm, y = sample, colour="Normal"), shape=1, alpha=0.5, size=1.5) +
  labs(x = "Theoretical Quantiles", y = "Sample Values", color = "Distributions") +
  scale_color_manual(values = c("Cauchy" = "blue", "Normal" = "red")) +
  theme_linedraw(base_size = 8)

# Create inset plot
inset_plot <- main_plot + labs(title = "QQ-Plot (zoomed in)") + xlim(-10, 15) + ylim(-15, 20) +
  theme_clean(base_size = 8) + theme(legend.position = "none", axis.title = element_blank())

inset_grob <- ggplotGrob(inset_plot) # Convert to a grid graphical object

# Final result
final_plot <- main_plot + annotation_custom(grob = inset_grob, xmin = -65, xmax = -10, ymin = 100, ymax = 250) +
  annotate("rect", xmin = -10, xmax = 15, ymin = -15, ymax = 20, ...
            color = "black", fill = "gray", size = 0.25, linetype = "dashed", alpha = 0.5) +
  annotate("segment", x = -10, xend = -65, y = -15, yend = 100, linetype = "dashed", color = "black", size = 0.25) +
  annotate("segment", x = 15, xend = -10, y = 20, yend = 250, linetype = "dashed", color = "black", size = 0.25)

print(final_plot) # Display the resulting plot
```

Source Code Q8

Question 9



Comentário: Os dois métodos diferem no cálculo do desvio padrão da média da amostra. O método 1 pressupõe o conhecimento do parâmetro p, calculando o desvio padrão com $\texttt{sqrt}[p(1-p)/n]$ (suposição padrão ao aplicar o Teorema do Limite Central, mas impraticável, já que tal parâmetro é normalmente desconhecido). Por outro lado, o método 2 calcula o desvio padrão com recurso à média da amostra, $\overline{x}$ (i.e., $\texttt{sqrt}[\overline{x}(1-\overline{x})/n]$), eliminando a necessidade de conhecer a verdadeira proporção populacional, p.
Quando o tamanho da amostra, n, é pequeno, $\overline{x}$ é pouco fidedigno, levando a intervalos de confiança maiores devido à maior incerteza na estimativa. No entanto, à medida que n aumenta, ambos os métodos convergem exponencialmente, refletindo a rapidez com que os valores do desvio padrão se alinham, evidenciando o poder do Teorema do Limite Central e da lei dos grandes números (amostras de tamanho superior garantem uma melhor representação da população, o que reduz o impacto da aleatoriedade e variabilidade dos resultados).

```r
# Load required libraries - install.packages("pacman")
pacman::p_load(Rlab, tidyverse)

# Fix the random seed
set.seed(1505)

# Declare both methods
method_1 <- function(sample_mean, sample_size){

  p = 0.8
  gamma <- 0.9
  z = qnorm((1 + gamma) / 2)

  # Apply quadratic formula
  a = 1 + z^2/sample_size
  b = -z^2/sample_size - 2 * sample_mean
  c = sample_mean^2

  lower <- (-b - sqrt(b^2 - 4*a*c))/(2*a)
  upper <- (-b + sqrt(b^2 - 4*a*c))/(2*a)

  interval_width = upper - lower

  # Return width for comparison
  return(interval_width)
}

method_2 <- function(sample_mean, sample_size){

  sd <- sqrt(sample_mean * (1 - sample_mean) / sample_size)
  gamma <- 0.9

  # Compute the 90% confidence interval
  z <- qnorm((1 + gamma) / 2)
  interval_width = 2*z*sd

  # Return width for comparison
  return(interval_width)
}
```

```r
# Compute the differences
sample_size = c(30,50,100,200,300,500,1000)
mean_difference_vector = seq(1,7,by = 1)

for(i in 1:length(sample_size)){
  difference_vector = seq(1,3000,by=1)

  for(n in 1:3000){
    sample_vector = rbinom(n = sample_size[i], size = 1,prob = 0.8)

    sample_mean = mean(sample_vector)
    interval_width_1 = method_1(sample_mean, sample_size[i])
    interval_width_2 = method_2(sample_mean, sample_size[i])

    difference = interval_width_2 - interval_width_1
    difference_vector[n] = difference
  }

  mean_difference_vector[i] = mean(difference_vector)
}

# Create a data frame from the vectors
data <- data.frame(
  SampleSize = sample_size,
  MeanDifference = mean_difference_vector
)

# Display the resulting plot
ggplot(data, aes(x = SampleSize, y = MeanDifference)) +
  ggtitle("Average Difference in Confidence Interval Widths Between Two Methods") +
  geom_line(color = "gray20", linetype = "dashed") +
  geom_hline(yintercept = 0, color = "blue", size = 0.5) +
  geom_segment(aes(xend = SampleSize, yend = 0)) +
  geom_point(shape = 19, size = 2) +
  scale_x_continuous(breaks = unique(sample_size)) +
  scale_y_continuous(limits = c(0, 0.0035), breaks = seq(0, 0.0035, by = 0.00025)) +
  xlab("Sample Size") + ylab("Average Difference Between Methods") +
  theme_linedraw(base_size = 8) +
  theme(panel.grid.major.x = element_blank(), panel.grid.minor.x = element_blank())
```

Source Code Q9

Question 10

Valor final: 0.470

```r
# Fix the random seed for reproducibility
set.seed(717)

# Define the true mean, variance, and sample size
mu <- 51.9
sigma <- sqrt(4)  # standard deviation is the square root of variance
n <- 20

# Generate m samples of size n from the Normal distribution
m <- 100
samples <- replicate(m, rnorm(n, mu, sigma))

# Conduct a z-test on each sample
z.test <- function(x, mu = 0, sigma.x = 1){
  xbar <- mean(x)
  se <- sigma.x / sqrt(length(x))
  z <- (xbar - mu) / se
  p.value <- 2 * (1 - pnorm(abs(z)))  # Two-tailed test
  return(list(statistic = z, p.value = p.value))
}

p_values <- apply(samples, 2, function(x) z.test(x, 50.9, sigma)$p.value)

# Estimate the probability of not rejecting the null hypothesis
alpha <- 0.03
probability <- mean(p_values > alpha)

# Print the estimated probability
print(round(probability, 3))
```

Source Code Q10