Assignment 1

Kshitij Kavimandan, Pablo Alves, Pooja Mangal (Group 15)

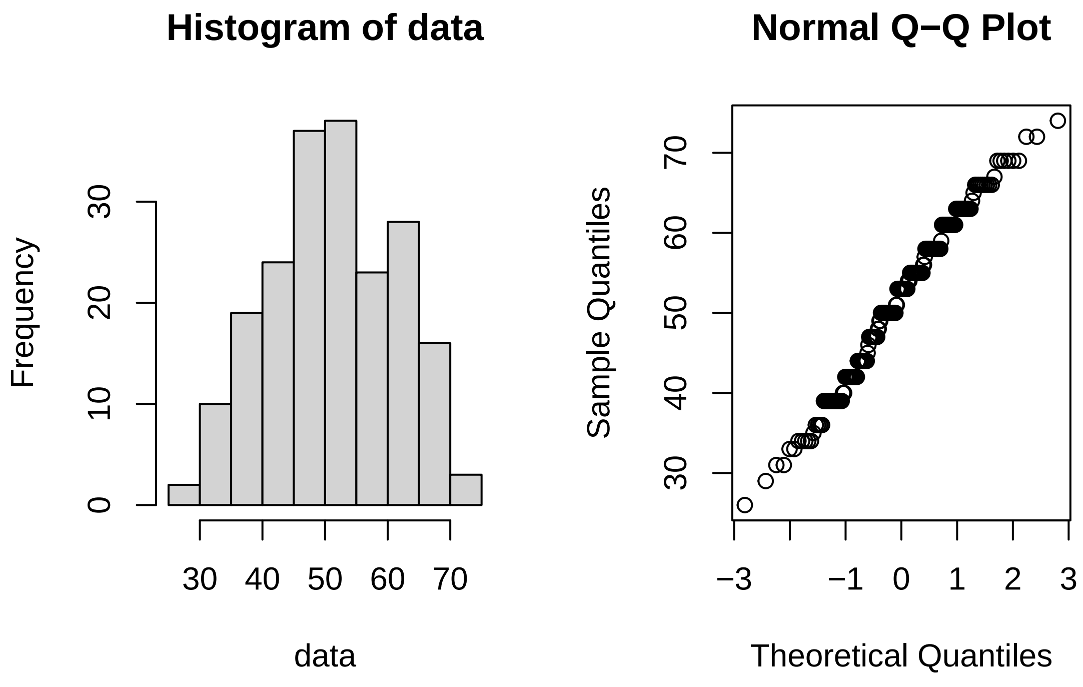
20 February 2024

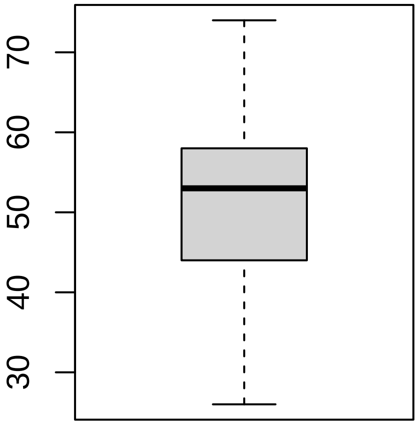
Exercise 1. Ice cream

a) Relevant plots and normality assessment:

```
# Load the dataset
ice_cream_dataset <- read.csv("Ice_cream.csv")
data = ice_cream_dataset$video
library("drcarlate")
```

```
# Do basic plots
par(mfrow=c(1,2)); hist(data); qqnorm(data); boxplot(data)
```





The histogram of the videogame scores looks normally distributed because it is roughly symetrical around 50, with values getting smaller towards its extremes. Despite this, the histogram bar for values between 60 and 65 is higher than what the corresponding analytical normal distribution should have.

Secondly, the Q-Q Plot also shows that our sample is densely populated in the center and its shape is very close to a line, which is also consistent with normality.

Finally, the boxplot is also consistent with normality, despite its median being not symetrical with its surrounding quantiles. As shown with the histogram, this can be explained with the additional values which are more frequently present in that particular region.

In short, our inspection of these basic plots suggest that this sample is normally distributed.

Because we are assuming by definition that our sample is normally distributed, we can apply the 68-95-99.7% rule. Because our sample has a mean of 51.85 and a standard deviation of 9.9, we can then see that the desired 97% confidence interval for the mean should lie inside the 6 sigma interval, which has a length of 59.9. With this rough estimation, our 97% confidence value for our mean is 51.85 +- 29.7, which is akin to the [22.15,81.55] interval.

To get the 97% CI, we need to evaluate the margin of error with the formula

margin = T(0.97)*(std(x)/sqrt(N))

where T is the t distribution with N-1 degrees of freedom, N our sample size (200)…

```r
options(digits = 3)
n <- length(data)
margin <- qt(0.97, df=n-1)*sd(data)/sqrt(n)
min_mean = mean(data) - margin
max_mean = mean(data) + margin
length = (max_mean - min_mean)/2; length
```

```
## [1] 1.32
```

```r
paste("97 CI: ", mean(data), " +-", length)
```

```
## [1] "97 CI:  51.85  +- 1.32428938516609"
```

Thus, the 97% CI for the mean is [50.52571,53.1749], with a length of 2.648579

In order to compute the bounded 97% CI for mu, we need to use the following formula:

```r
m_calculator<-function(p, data) sqrt(sd(data)/length(data)*(4*norminv(p+(1-p)/2)))
margin = m_calculator(p=0.97,data=data)
min_mean = mean(data) - margin
max_mean = mean(data) + margin
length = max_mean - min_mean; length
```

```
## [1] 1.31
```

```r
paste("97 CI: ", mean(data)," +-", length)
```

```
## [1] "97 CI:  51.85  +- 1.31105538574369"
```

In order to compute the neccesary number of samples needed for a particular maximun length of the CI we can use the following formula

```r
n_calculator <- function(p,length,data) sd(data)*(2*norminv(p+(1-p)/2)/length)^2
samples_needed = n_calculator(p=0.97,length=3,data=data)
paste("For a margin of: ", 3 ," we need", samples_needed)
```

```
## [1] "For a margin of:  3  we need 20.7227503006415"
```

To determine the 97% bootstrap CI we can apply the following code

```r
p = 0.97
B = 100

Tstar = numeric(B)
for(i in 1:B){Tstar[i]=mean(sample(data,replace=TRUE))}

Tstar985 = quantile(Tstar,0.985)
Tstar15 = quantile(Tstar, 0.015)

c(2*mean(Tstar) - Tstar985,2*mean(Tstar)-Tstar15)
```

```
## 98.5%  1.5%
##  50.0  53.3
```

```r
d = ((2*mean(Tstar)-Tstar15) - (2*mean(Tstar) - Tstar985))
paste("Length:",d)
```

```
## [1] "Length: 3.24025"
```

```r
paste("Margin on each side:",d/2)
```

```
## [1] "Margin on each side: 1.620125"
```

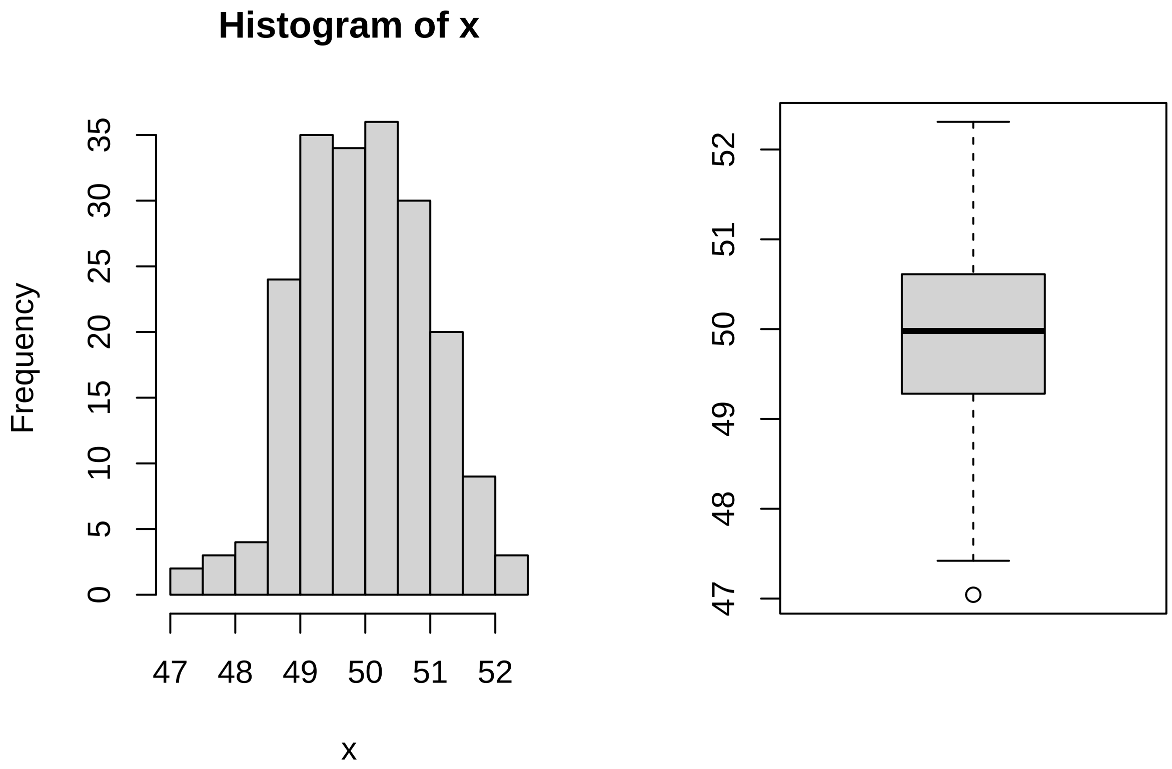
Our Bootstrap margin is bigger than our previous margins, which is consistent with the fact that the Bootstrap method does not assume normality, and thus is expected to have a bigger uncertainty on its estimation.

---

b) T-test to verify mean score:

The t-test compares the mean of the sample to the hypothesized mean and calculates a t-statistic and a p-value.

```r
mu = 50
x = rnorm(length(data), mu, 1)
par(mfrow=c(1,2))
hist(x)
boxplot(x)
```



```
t_test_result <- t.test(x, alternative="g", mu = mu)
print(t_test_result)
```

```
##
##  One Sample t-test
##
## data:  x
## t = -0.3, df = 199, p-value = 0.6
## alternative hypothesis: true mean is greater than 50
## 95 percent confidence interval:
##  49.9  Inf
## sample estimates:
## mean of x
##        50
```

```
t_test_result_mu_51 <- t.test(x, alternative="g", mu = 51)
t_test_result_mu_51
```

```
##
##  One Sample t-test
##
```

```
## data:  x
## t = -15, df = 199, p-value = 1
## alternative hypothesis: true mean is greater than 51
## 95 percent confidence interval:
##  49.9  Inf
## sample estimates:
## mean of x
##        50
```

- In the first test, with mu_0 = 50, the p-value is more than the significance level of 0.05 which suggests that there isn't enough evidence to reject the null hypothesis, indicating that the mean score on the video game is not significantly different from 50.
- Similarly, when testing against mu_0 = 51, the high p-value of 1 implies strong evidence in favor of the null hypothesis, meaning that the mean score is not significantly greater than 51.
- In the output, the 95% confidence interval for the first test is (49.9, infinity). This means that we are 95% confident that the true population mean lies in the above interval. The upper bound of infinity indicates that the upper limit is unbounded.
- The confidence interval for the second test is the same, as it is based on the sample mean and standard deviation. It doesn't change with the hypothesized mean mu_0.

c) Sign test and test based on ranks:

```r
# Sign test
sign_test <- binom.test(sum(data > 50), length(data), p = 0.5, alternative = "g")
print(sign_test)
```

```
##
##  Exact binomial test
##
## data:  sum(data > 50) and length(data)
## number of successes = 108, number of trials = 200, p-value = 0.1
## alternative hypothesis: true probability of success is greater than 0.5
## 95 percent confidence interval:
##  0.479 1.000
## sample estimates:
## probability of success
##                   0.54
```

```r
# Test based on ranks (Wilcoxon signed-rank test)
wilcox_test <- wilcox.test(data, mu = 50, alternative = "g")
print(wilcox_test)
```

```
##
##  Wilcoxon signed rank test with continuity correction
```

```
##
## data:  data
## V = 9836, p-value = 0.005
## alternative hypothesis: true location is greater than 50
```

```
# Test for fraction of scores less than 42
binom_test <- binom.test(sum(data < 42), length(data), p = 0.25, alternative = "l")
print(binom_test)
```

```
##
##  Exact binomial test
##
## data:  sum(data < 42) and length(data)
## number of successes = 31, number of trials = 200, p-value = 8e-04
## alternative hypothesis: true probability of success is less than 0.25
## 95 percent confidence interval:
##  0.000 0.203
## sample estimates:
## probability of success
##                  0.155
```

- For the median score, the sign test results in a p-value of 0.144, suggesting no significant difference between the median score and mu_0 = 50. The 95% confidence interval for the success probability of the sign test ranges from 0.479 to 1.
- Comments on comparison of the above results with the previous question (b)): *The t-test evaluates whether the mean score is significantly greater than 50, while the sign test assesses if the median score is greater than 50. Both tests yield p-values greater than 0.05, indicating that there's insufficient evidence to reject the null hypotheses. The confidence interval from the t-test provides a range of plausible values for the population mean, whereas the sign test directly assesses the proportion of observations above 50.*
- The Wilcoxon signed-rank test results in a p-value of 0.005, indicating a significant difference between the median score and mu_0 = 50. The alternative hypothesis suggests that the true location is greater than 50.
- Lastly, the binomial test to check the fraction of scores less than 42 yields a p-value of 0.0007, suggesting that the probability of success (scores less than 42) is less than 0.25.

d) Bootstrap test with test statistic T:

```
# Bootstrap test (slides)
n = length(data); t=min(data); t
```

```
## [1] 26
```

```r
B = 100; tstar = numeric(B)
for (i in 1:B) {xstar=rexp(n,1)
 tstar[i]=min(xstar)}
pl = sum(tstar<t)/B; pr=sum(tstar>t)/B
p=2*min(pl,pr); p
```

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

```r
# Bootstrap test (google)
# Define the test statistic function
test_statistic <- function(data) {
  return(min(data))
}

# Number of bootstrap samples
B <- 1000

# Bootstrap test
bootstrap_test <- function(mu) {
  tstar <- numeric(B)
  for (i in 1:B) {
    bootstrap_sample <- rnorm(B, mean = mu, sd = 10)  # Generate bootstrap sample
    tstar[i] <- test_statistic(bootstrap_sample)       # Compute minimum
  }

  # Compute observed minimum
  obs_min <- test_statistic(data[1:100])

  # Compute p-value
  p_value <- mean(tstar >= obs_min)

  # Determine if null hypothesis is rejected
  count <- 0
  if (p_value > 0.05) {
    reject_null <- TRUE
  } else {
    reject_null <- FALSE
    count <- count + 1
  }

  return(list(mu = mu, p_value = p_value, reject_null = reject_null))
}

# Perform bootstrap test for different values of mu in the range [0, 100]
results <- lapply(seq(0, 100, by = 1), bootstrap_test)

# Display results
```

```
for (result in results) {
  cat("mu:", result$mu, "p-value:", result$p_value, "Reject H0:", result$reject_null, "\n")
}
```

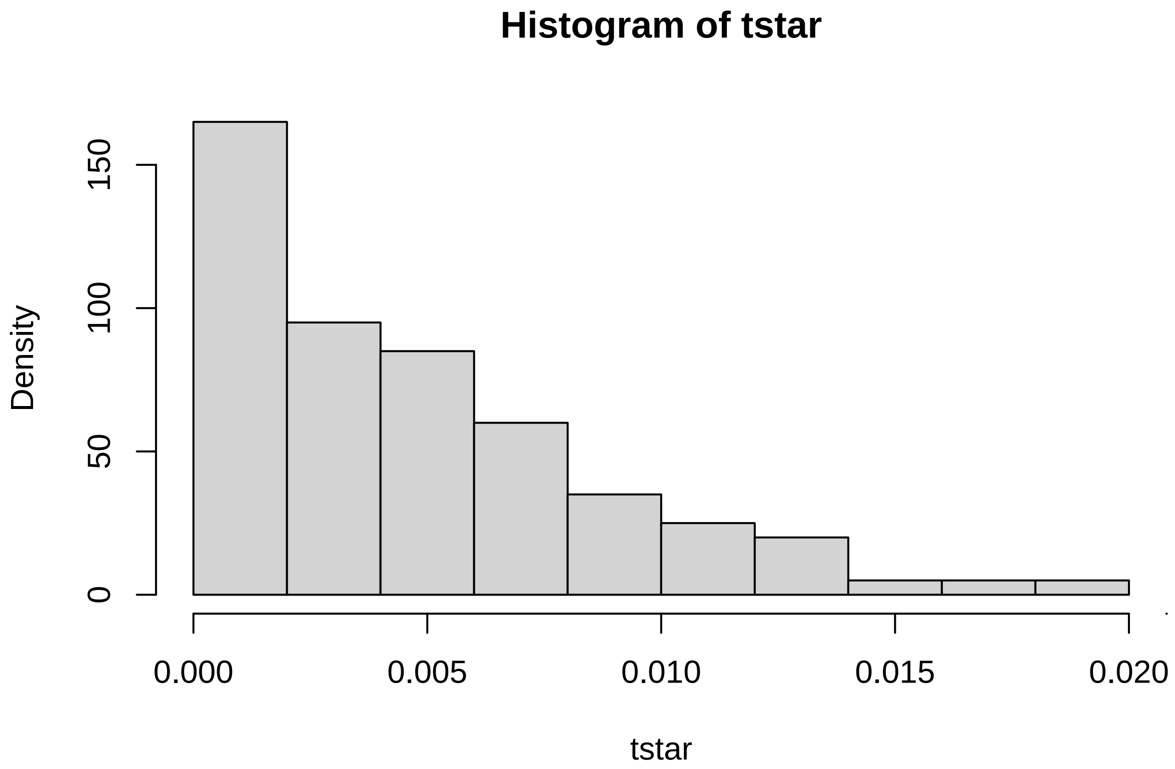
```
## mu: 0 p-value: 0 Reject H0: FALSE
## mu: 1 p-value: 0 Reject H0: FALSE
## mu: 2 p-value: 0 Reject H0: FALSE
## mu: 3 p-value: 0 Reject H0: FALSE
## mu: 4 p-value: 0 Reject H0: FALSE
## mu: 5 p-value: 0 Reject H0: FALSE
## mu: 6 p-value: 0 Reject H0: FALSE
## mu: 7 p-value: 0 Reject H0: FALSE
## mu: 8 p-value: 0 Reject H0: FALSE
## mu: 9 p-value: 0 Reject H0: FALSE
## mu: 10 p-value: 0 Reject H0: FALSE
## mu: 11 p-value: 0 Reject H0: FALSE
## mu: 12 p-value: 0 Reject H0: FALSE
## mu: 13 p-value: 0 Reject H0: FALSE
## mu: 14 p-value: 0 Reject H0: FALSE
## mu: 15 p-value: 0 Reject H0: FALSE
## mu: 16 p-value: 0 Reject H0: FALSE
## mu: 17 p-value: 0 Reject H0: FALSE
## mu: 18 p-value: 0 Reject H0: FALSE
## mu: 19 p-value: 0 Reject H0: FALSE
## mu: 20 p-value: 0 Reject H0: FALSE
## mu: 21 p-value: 0 Reject H0: FALSE
## mu: 22 p-value: 0 Reject H0: FALSE
## mu: 23 p-value: 0 Reject H0: FALSE
## mu: 24 p-value: 0 Reject H0: FALSE
## mu: 25 p-value: 0 Reject H0: FALSE
## mu: 26 p-value: 0 Reject H0: FALSE
## mu: 27 p-value: 0 Reject H0: FALSE
## mu: 28 p-value: 0 Reject H0: FALSE
## mu: 29 p-value: 0 Reject H0: FALSE
## mu: 30 p-value: 0 Reject H0: FALSE
## mu: 31 p-value: 0 Reject H0: FALSE
## mu: 32 p-value: 0 Reject H0: FALSE
## mu: 33 p-value: 0 Reject H0: FALSE
## mu: 34 p-value: 0 Reject H0: FALSE
## mu: 35 p-value: 0 Reject H0: FALSE
## mu: 36 p-value: 0 Reject H0: FALSE
## mu: 37 p-value: 0 Reject H0: FALSE
## mu: 38 p-value: 0 Reject H0: FALSE
## mu: 39 p-value: 0 Reject H0: FALSE
## mu: 40 p-value: 0 Reject H0: FALSE
## mu: 41 p-value: 0 Reject H0: FALSE
## mu: 42 p-value: 0 Reject H0: FALSE
## mu: 43 p-value: 0 Reject H0: FALSE
## mu: 44 p-value: 0 Reject H0: FALSE
## mu: 45 p-value: 0 Reject H0: FALSE
## mu: 46 p-value: 0 Reject H0: FALSE
## mu: 47 p-value: 0 Reject H0: FALSE
## mu: 48 p-value: 0 Reject H0: FALSE
## mu: 49 p-value: 0 Reject H0: FALSE
## mu: 50 p-value: 0 Reject H0: FALSE
## mu: 51 p-value: 0.002 Reject H0: FALSE
## mu: 52 p-value: 0.01 Reject H0: FALSE
## mu: 53 p-value: 0.03 Reject H0: FALSE
## mu: 54 p-value: 0.075 Reject H0: TRUE
## mu: 55 p-value: 0.16 Reject H0: TRUE
## mu: 56 p-value: 0.249 Reject H0: TRUE
## mu: 57 p-value: 0.372 Reject H0: TRUE
## mu: 58 p-value: 0.518 Reject H0: TRUE
## mu: 59 p-value: 0.621 Reject H0: TRUE
## mu: 60 p-value: 0.728 Reject H0: TRUE
## mu: 61 p-value: 0.78 Reject H0: TRUE
## mu: 62 p-value: 0.849 Reject H0: TRUE
## mu: 63 p-value: 0.894 Reject H0: TRUE
## mu: 64 p-value: 0.943 Reject H0: TRUE
## mu: 65 p-value: 0.951 Reject H0: TRUE
## mu: 66 p-value: 0.979 Reject H0: TRUE
## mu: 67 p-value: 0.982 Reject H0: TRUE
## mu: 68 p-value: 0.992 Reject H0: TRUE
## mu: 69 p-value: 0.988 Reject H0: TRUE
## mu: 70 p-value: 0.991 Reject H0: TRUE
## mu: 71 p-value: 1 Reject H0: TRUE
## mu: 72 p-value: 0.995 Reject H0: TRUE
## mu: 73 p-value: 0.998 Reject H0: TRUE
## mu: 74 p-value: 0.998 Reject H0: TRUE
## mu: 75 p-value: 1 Reject H0: TRUE
## mu: 76 p-value: 0.999 Reject H0: TRUE
## mu: 77 p-value: 1 Reject H0: TRUE
## mu: 78 p-value: 1 Reject H0: TRUE
## mu: 79 p-value: 1 Reject H0: TRUE
## mu: 80 p-value: 1 Reject H0: TRUE
## mu: 81 p-value: 1 Reject H0: TRUE
## mu: 82 p-value: 1 Reject H0: TRUE
## mu: 83 p-value: 1 Reject H0: TRUE
## mu: 84 p-value: 1 Reject H0: TRUE
## mu: 85 p-value: 1 Reject H0: TRUE
## mu: 86 p-value: 1 Reject H0: TRUE
## mu: 87 p-value: 1 Reject H0: TRUE
## mu: 88 p-value: 1 Reject H0: TRUE
## mu: 89 p-value: 1 Reject H0: TRUE
## mu: 90 p-value: 1 Reject H0: TRUE
## mu: 91 p-value: 1 Reject H0: TRUE
## mu: 92 p-value: 1 Reject H0: TRUE
## mu: 93 p-value: 1 Reject H0: TRUE
## mu: 94 p-value: 1 Reject H0: TRUE
## mu: 95 p-value: 1 Reject H0: TRUE
## mu: 96 p-value: 1 Reject H0: TRUE
## mu: 97 p-value: 1 Reject H0: TRUE
## mu: 98 p-value: 1 Reject H0: TRUE
## mu: 99 p-value: 1 Reject H0: TRUE
## mu: 100 p-value: 1 Reject H0: TRUE
```

```r
# Apply Kolmogorov-Smirnov test
# Perform Kolmogorov-Smirnov test for different values of mu in the range [0, 100]
mu_values_not_rejected <- c()

for (mu in seq(0, 100, by = 1)) {
  # Generate the first 100 samples from N(mu, 100)
  sample_data <- rnorm(100, mean = mu, sd = 10)

  # Compute the Kolmogorov-Smirnov statistic and p-value
  ks_test_result <- ks.test(sample_data, "pnorm", mean = mu, sd = 10)

  # Check if null hypothesis is rejected
  if (ks_test_result$p.value > 0.05) {
    mu_values_not_rejected <- c(mu_values_not_rejected, mu)
  }
}

# Display mu values for which null hypothesis is not rejected
cat("Mu values for which null hypothesis is not rejected:", mu_values_not_rejected, "\n")
```

```
## Mu values for which null hypothesis is not rejected: 0 1 2 3 4 5 6 8 9 10 11 12 13 14 15 16
```

Since p-value=0.007, H0 is rejected.

```r
hist(tstar,prob=T, main="Histogram of tstar")
lines(rep(t,2),c(0,p), col="red",lwd=2)
axis(1,t,expression(paste("t")))
```



e) Tests for male and female students:

```
data <- read.csv("Ice_cream.csv")

# Separate scores for male and female students
male_scores <- data$video[data$female == 0]
female_scores <- data$video[data$female == 1]
# Perform two-sample t-test
t_test_gender <- t.test(male_scores, female_scores, alternative = "g"); t_test_gender
```

```
##
##  Welch Two Sample t-test
##
## data:  male_scores and female_scores
## t = 2, df = 177, p-value = 0.04
## alternative hypothesis: true difference in means is greater than 0
## 95 percent confidence interval:
##  0.186   Inf
## sample estimates:
## mean of x mean of y
##      53.2      50.7
```

```
# Perform Mann-Whitney test
mannwhitney_test <- wilcox.test(male_scores, female_scores,
alternative = "g"); mannwhitney_test
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  male_scores and female_scores
## W = 5748, p-value = 0.03
## alternative hypothesis: true location shift is greater than 0
```

```
# Perform Kolmogorov-Smirnov test
ks_test_gender <- ks.test(male_scores, female_scores); ks_test_gender
```

```
##
##  Exact two-sample Kolmogorov-Smirnov test
##
## data:  male_scores and female_scores
## D = 0.2, p-value = 0.07
## alternative hypothesis: two-sided
```

f) Correlation and comparison between video game and puzzle scores:

```
# Investigate correlation
correlation <- cor(data$video, data$puzzle); correlation
```

```
## [1] 0.465
```

```
# Test if puzzle scores are higher than video game scores
wilcox_test_puzzle <- wilcox.test(data$puzzle, data$video, alternative = "g"); wilcox_test_puzz
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  data$puzzle and data$video
## W = 21044, p-value = 0.2
## alternative hypothesis: true location shift is greater than 0
```

Exercise 2. Hemoglobin in trout

a) R-code for randomization process: a) R-code for randomization process:

```r
# Load the data
hemoglobin_data <- read.table("hemoglobin.txt", header = TRUE)
# Create a data frame to store the combinations of rate and method
combinations <- expand.grid(rate = c(1, 2, 3, 4), method = c("A", "B"))
# Randomly assign 80 fishes to the combinations
set.seed(123) # Set seed for reproducibility
hemoglobin_data$fish <- sample(rep(1:80,
length.out = nrow(hemoglobin_data)),
replace = FALSE)
# Merge the combinations with the hemoglobin data
final_data <- merge(combinations, hemoglobin_data, by = "rate", all = TRUE)
```

b) Two-way ANOVA:

```r
# Perform two-way ANOVA
anova_result <- aov(hemoglobin ~ rate * method, data = hemoglobin_data)
# Print ANOVA table
print(summary(anova_result))
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## rate         1   27.9   27.93   11.93 0.00091 ***
## method       1    2.4    2.42    1.03 0.31296
## rate:method  1    1.2    1.24    0.53 0.46837
## Residuals   76  177.9    2.34
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

c) Influence of factors and combination yielding highest hemoglobin:

```r
# Calculate means for rate and method
mean_hemoglobin <- aggregate(hemoglobin ~ rate + method,
data = hemoglobin_data, FUN = mean)
# Find combination yielding highest hemoglobin
max_hemoglobin <- mean_hemoglobin[which.max(mean_hemoglobin$hemoglobin), ]
# Estimate mean hemoglobin value for rate 3 by using method A
mean_hemoglobin_rate3_methodA <-
mean_hemoglobin$hemoglobin[which(mean_hemoglobin$rate == 3 &
mean_hemoglobin$method == "A")]
# Estimate mean hemoglobin value for each rate
mean_hemoglobin_rate <- aggregate(hemoglobin ~ rate,
data = hemoglobin_data, FUN = mean)
```

d) One-way ANOVA:

```
# Perform one-way ANOVA ignoring the variable method
anova_result_rate <- aov(hemoglobin ~ rate, data = hemoglobin_data)
# Print ANOVA table
print(summary(anova_result_rate))
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## rate         1   27.9   27.93      12 0.00087 ***
## Residuals   78  181.6    2.33
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

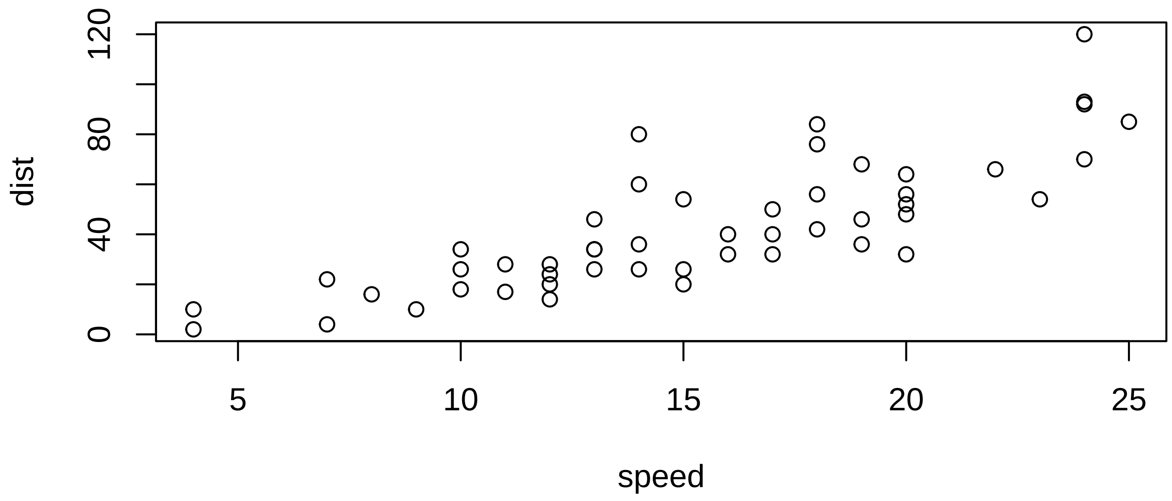
e) Kruskal-Wallis test:

```
# Perform Kruskal-Wallis test
kruskal_test_result <- kruskal.test(hemoglobin ~ rate, data = hemoglobin_data)
# Print Kruskal-Wallis test result
print(kruskal_test_result)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  hemoglobin by rate
## Kruskal-Wallis chi-squared = 34, df = 3, p-value = 2e-07
```

Figures

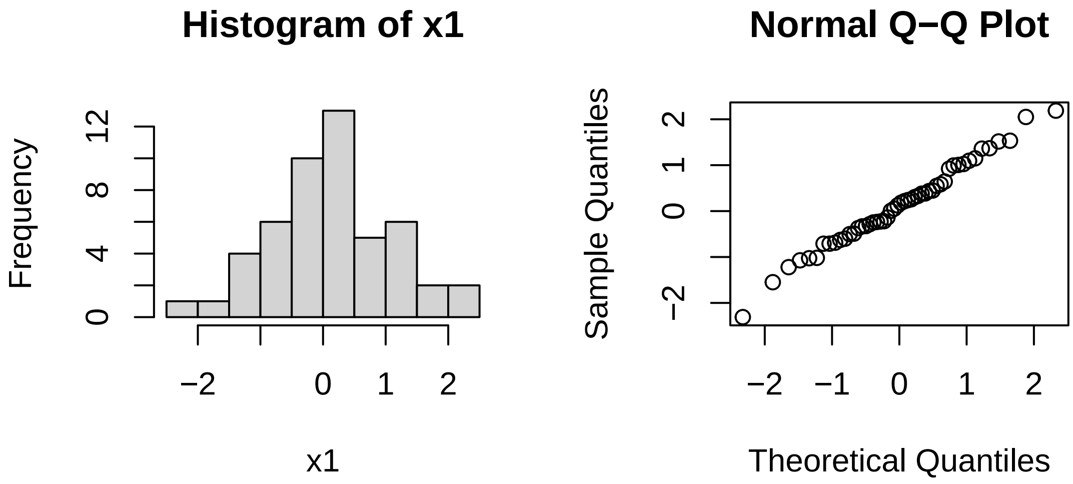
You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot. Use knitr options to style the output of a chunk. Place options in brackets above the chunk. Other options with the defaults are: the `eval=FALSE` option just displays the R code (and does not run it); `warning=TRUE` whether to display warnings; `tidy=TRUE` wraps long code so it does not run off the page.

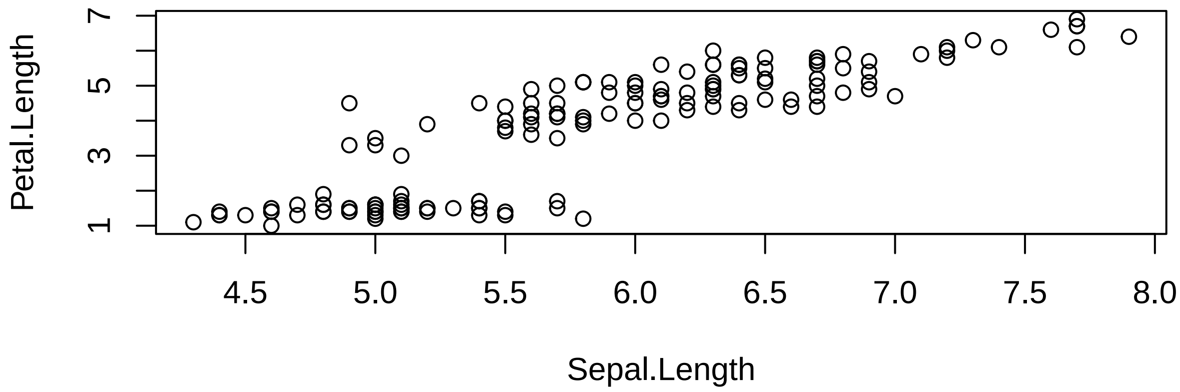
You can control the size and placement of figures. For example, you can put two figures (or more) next to each other. Use `par(mfrow=c(n,m))` to create `n` by `m` plots in one picture in R. You can adjust the proportions of figures by using the `fig.width` and `fig.height` chunk options. These are specified in inches, and will be automatically scaled down to fit within the handout margin. Chunk option `fig.align` takes values `left`, `right`, or `center` (to align figures in the output document).

```
par(mfrow=c(1,2)); x1=rnorm(50); hist(x1); qqnorm(x1)
```



You can arrange for figures to span across the entire page by using the `fig.fullwidth` chunk option.

```
plot(iris$Sepal.Length,iris$Petal.Length,xlab="Sepal.Length",ylab="Petal.Length")
```



More about chunk options can be found at https://yihui.name/knitr/options/.

Equations

To produce mathematical symbols, you can also include LaTeX expessions/equations in your report: inline $\frac{d}{dx}\left(\int_0^x f(u)\,du\right) = f(x)$ and in the display mode:

$$\frac{d}{dx}\left(\int_0^x f(u)\,du\right) = f(x).$$

To be able to use this functionality, LaTeX has to be installed.

Footnotes

Here is the use of a footnote[1].

Images

Want an image? This will do it. To depict an image (say, `my_image.png` which should be in your current working directory), use this command

Tables

Want a table? This will create one (note that the separators *do not* have to be aligned).

Table Header	Second Header
Table Cell	Cell 2

[1] This is a footnote.

Table Header	Second Header
Cell 3	Cell 4

You can also make table by using knit's `kable` function:

Table 2: A knit kable.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.62	16.5	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.88	17.0	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.32	18.6	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.21	19.4	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.44	17.0	0	0	3	2

Block quote

> This will create a block quote, if you want one.

Verbatim

```
This text is displayed verbatim/preformatted.
```

Links

Links: http://example.com, in-text link to Google.

This is a hyperlink.

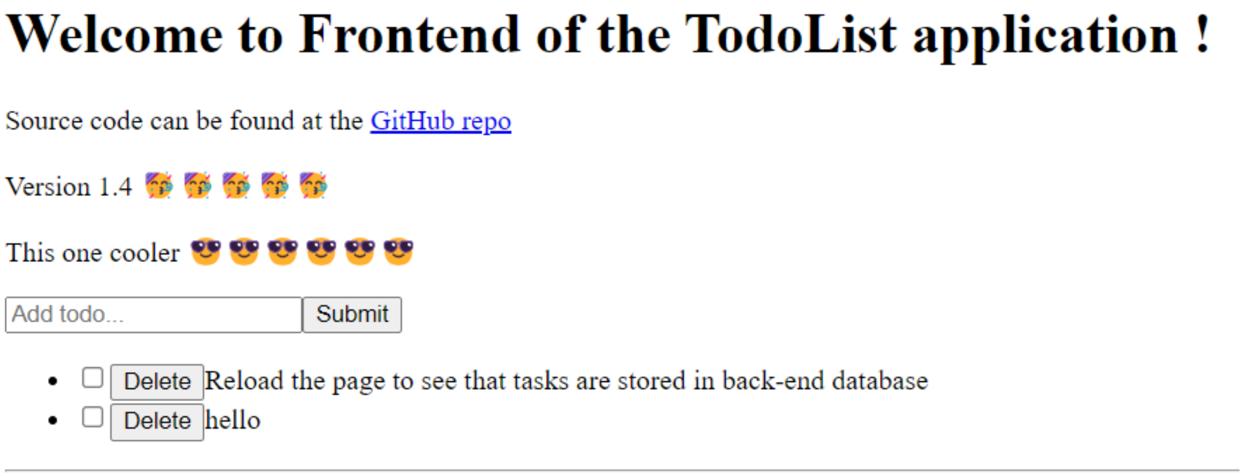
This

is where the hyperlink jumps to.

Itimization, italicized and embolded text

- Single asterisks italicize text *like this.*
- Double asterisks embolden text **like this**.

One more way to italicize and embold: *italic* and **bold**.



Welcome to Frontend of the TodoList application !

Source code can be found at the GitHub repo

Version 1.4 🥳 🥳 🥳 🥳 🥳

This one cooler 😎 😎 😎 😎 😎 😎

- □ Reload the page to see that tasks are stored in back-end database
- □ hello

Figure 1: caption for my image