STA303/1002 Portfolio

An exploration of linear mixed models and common misconceptions in statistics

Jiahao(Green) Bai

2022-02-03

Contents

List of Figures

Introduction

This portfolio is a course project of STA303 with the usage of R.

STA303 is a statistic course at the University of Toronto. Its concepts include but are not limited to the following: wrangle and explore data, understand commonly used statistic models, data visualization, make proper conclusions based on data analysis.

This portfolio has three sections: statistical skills sample, writing sample, and reflection.

The statistical skill sample section includes five tasks focused on three concepts:
1. Application of linear mixed model
2. The common misunderstandings of statistic terms
3. Reprex

Task 1 is to set up the libraries required for this portfolio.
Task 2 is related to the linear mixed model, based on the dataset of patches of strawberries with different treatments. It conducts visualization of patches using tidyverse package, comparison between models with REML, and interpretation of the final model.
Task 3 and 5 concentrate on the interpretation of confidence intervals and p-values. For task 3, I would implement interpreters of CI and p-values, along with instructions and disclaimer. For task 5, I would simulate p-values of normal distributions with different parameters and use histograms along with Q-Q plots to find a connection with the definition of the p-value when the null hypothesis is true.
Task 4 is about creating the reprex, explaining what reprex is, and what needs to be considered before making it.

The writing sample section is a summary and discussion about an article that is about common misconceptions about data analysis and statistics.

The reflection section requires reviewing the whole portfolio and writing about good things done and something that I could do differently next time. Also, the application of things I learned in future work and study.

Statistical skills sample

Task 1: Setting up libraries and seed value

```r
# Setup the libraries need for this portfolio
library(tidyverse)
library(lme4)

# Create a variable holds 100 + last 3 digits of student number
last3digplus = 100 + 097
```

Task 2a: Return to Statdew Valley: exploring sources of variance in a balanced experimental design (teaching and learning world)

Growinng your (grandmother's) strawberry patch

```r
# Don't edit this file
# Sourcing it makes a function available
source("grow_my_strawberries.R")

# Create the strawberry patch and make treatment a factor variable with levels
# as required
my_patch = grow_my_strawberries(seed = last3digplus)

my_patch <- my_patch %>% mutate(treatment =
                                  fct_relevel(treatment, "No netting", after = 0))
```

Plotting the strawberry patch

```r
# Plot the strawberry patch: patch vs yield with different treatment
my_patch %>% ggplot(aes(x = patch, y = yield, fill = treatment,
                        color = treatment)) +
             geom_point(pch = 25) +
             scale_fill_manual(values = c("#78BC61", "#E03400", "#520048")) +
             scale_color_manual(values = c("#78BC61", "#E03400", "#520048")) +
             theme_minimal() +
        labs(caption =
                "Created by Jiahao(Green) Bai in STA303/1002, Winter 2022")
```

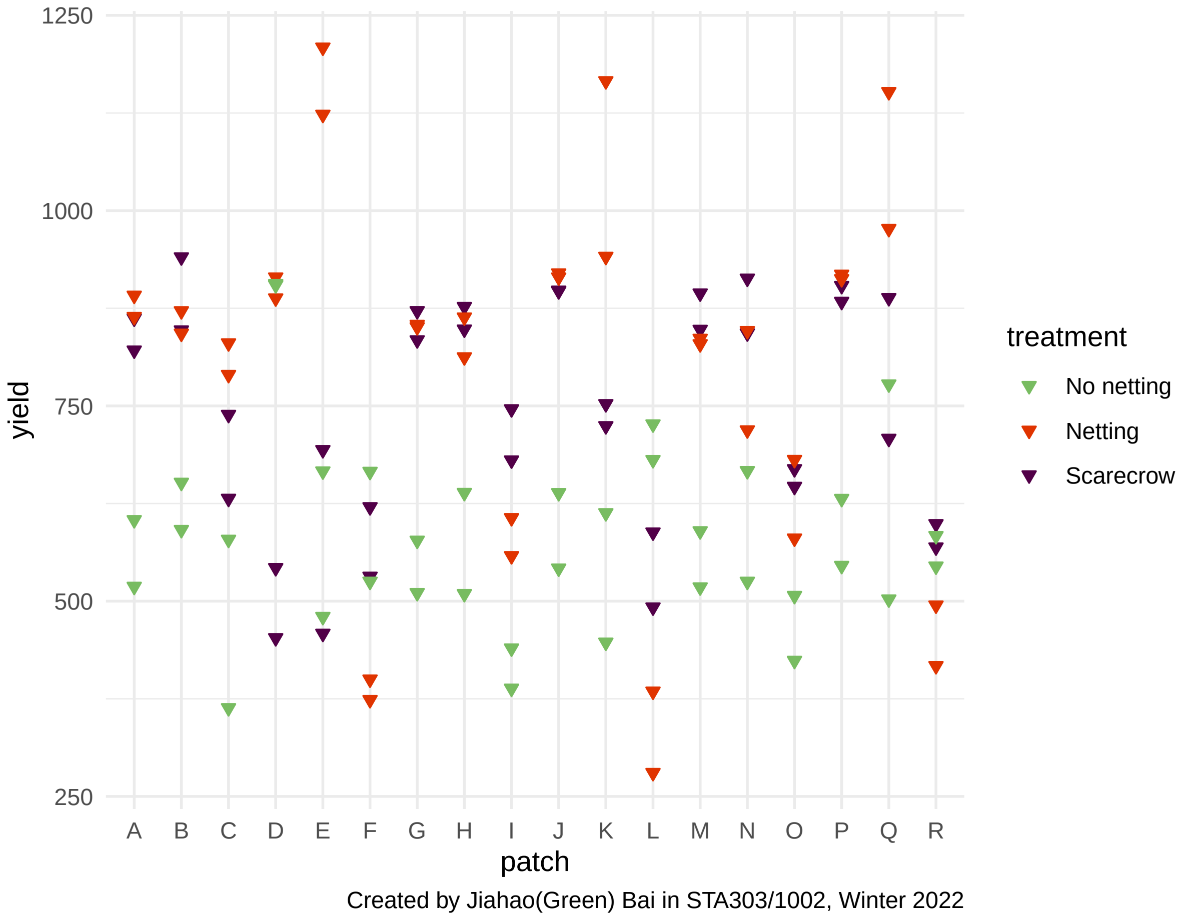


Figure 1: Strawberry Patch vs Yield Scatter Plot with 3 Different Treatments Colored

Demonstrating calculation of sources of variance in a least-squares modelling context

Model formula

$$y_{ijk} = \mu + \alpha_i + b_j + (\alpha b)_{ij} + \epsilon_{ijk}$$

where:

- y_{ijk} is the amount of strawberry yielded (in kilograms) in the k^{th} harvest time by the j^{th} patch with i^{th} treatment
- μ is the grand mean yield of strawberry
- α_i are the I fixed effects for treatment, where $i = 1, 2, 3$

- b_j are the random effect of patch j, where $j = 1, \dots, 18$
- $(\alpha b)_{ij}$ are the IJ interaction terms (54 terms in total) for the interaction between the treatment and the patch
- ϵ_{ijk} is the error term at each k^{th} harvest time by the j^{th} patch with i^{th} treatment
- $(\alpha b)_{ij} \sim N(0, \sigma^2_{\alpha b})$, $b_j \sim N(0, \sigma^2_b)$ and $\epsilon_{ijk} \sim N(0, \sigma^2)$

```r
# Create tibbles
# Tibble of aggregate patch data, which grouped by patch and treatment
# yield_avg_int is average strawberry yield for each patch and treatment
# combination
agg_int <- my_patch %>% group_by(patch, treatment) %>%
                        summarize(yield_avg_int = mean(yield), .groups = "drop")

# Tibble of aggregate patch data, which grouped by patch and yield_avg_int is
# average strawberry yield for each patch and treatment combination
agg_patch <- my_patch %>% group_by(patch) %>%
                         summarise(yield_avg_patch = mean(yield), groups = "drop")
```

```r
# Create models
# Interaction model which includes main effects
int_mod = lm(yield ~ patch * treatment, data = my_patch)

# Main effects model
agg_mod = lm(yield_avg_int ~ patch + treatment, data = agg_int)

# Intercept-only model
patch_mod = lm(yield_avg_patch ~ 1, data = agg_patch)
```

```r
# Variance in average yield patch-to-patch
var_patch = summary(patch_mod)$sigma^2 - (1 / 3) * (summary(agg_mod)$sigma^2)
                                         # K = 3 because there are 3 treatments
# Residual variance after fitting the version of this linear model with this
# most parameters
var_int = summary(int_mod)$sigma^2

# Variance in yield explained by the interaction between patch and treatment,
# after accounting for the fixed effects and other sources.
var_ab = summary(agg_mod)$sigma^2 - (1 / 2) * var_int
```

```
# Create the variances and proportions table
sum_var = sum(var_int, var_ab, var_patch) # Sum of the three variances
tibble(`Source of variation` = c("patch",
                                 "treatment:patch",
                                 "residual"),
       Variance = c(var_patch, var_ab, var_int),
       Proportion = c(round(var_patch / sum_var, 2),
                      round(var_ab / sum_var, 2),
                      round(var_int / sum_var, 2) )) %>%
  knitr::kable(caption = "Table of Variances and Proportions of Variances in
                          Yield Explained by The Corresponding Sources")
```

Table 1: Table of Variances and Proportions of Variances in Yield Explained by The Corresponding Sources

Source of variation	Variance	Proportion
patch	3567.179	0.12
treatment:patch	21668.003	0.71
residual	5408.529	0.18

Task 2b: Applying linear mixed models for the strawberry data (practical world)

```
# Create linear model
mod0 = lm(yield ~ treatment, data = my_patch)

# Create linear mixed models
mod1 = lmer(yield ~ treatment + (1|patch), data = my_patch)
mod2 = lmer(yield ~ treatment + (1|patch) +
                                (1|patch:treatment), data = my_patch)
```

For the comparisons, the default setting of R: REML is used, since the mod1 and mod2 involve both fixed and random effects as model parameters. ML should be used when we want to compare the fixed effects of nested models only, so it's not the method we should consider for this task.

```
# Comparison between three models with likelihood ratio test
lmtest::lrtest(mod1, mod2)
```

```
## Likelihood ratio test
##
## Model 1: yield ~ treatment + (1 | patch)
## Model 2: yield ~ treatment + (1 | patch) + (1 | patch:treatment)
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   5 -689.46
## 2   6 -664.80  1 49.323  2.171e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
lmtest::lrtest(mod0, mod2)
```

```
## Warning in modelUpdate(objects[[i - 1]], objects[[i]]): original model was of
## class "lm", updated model is of class "lmerMod"
```

```
## Likelihood ratio test
##
## Model 1: yield ~ treatment
## Model 2: yield ~ treatment + (1 | patch) + (1 | patch:treatment)
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   4 -708.27
## 2   6 -664.80  2 86.939  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Summary of the final model
summary(mod2)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: yield ~ treatment + (1 | patch) + (1 | patch:treatment)
##    Data: my_patch
##
## REML criterion at convergence: 1329.6
```

```
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.87950 -0.53646  0.08453  0.43338  1.89883
##
## Random effects:
##  Groups          Name        Variance Std.Dev.
##  patch:treatment (Intercept) 21668    147.20
##  patch           (Intercept)  3567     59.73
##  Residual                     5409     73.54
## Number of obs: 108, groups:  patch:treatment, 54; patch, 18
##
## Fixed effects:
##                    Estimate Std. Error t value
## (Intercept)          582.97      39.40  14.797
## treatmentNetting     209.14      52.04   4.019
## treatmentScarecrow   159.20      52.04   3.059
##
## Correlation of Fixed Effects:
##             (Intr) trtmnN
## trtmntNttng -0.660
## trtmntScrcr -0.660  0.500
```

Justification and interpreation

The most appropriate final model is mod2, the linear mixed model with treatment, patch and the interaction of treatment and patch. The p-value of the likelihood ratio test between mod1 and mod2 is 2.171e-12, and that of the test between mod0 and mod2 is < 2.2e-16. These two p-values are small. Thus, we have strong evidence against the hypothesis of each test that mod1, which does not have patch/treatment interaction, and mod0, which contains only the fixed effect, are as good as mod2.
The fixed effect coefficients of final model mod2 show that no netting treatment (control group) would have 582.97 kilograms strawberry in yield. The netting treatment and scarecrow treatment would have increases of 209.14 kilograms and 159.20 kilograms strawberries in yield on average compared to no netting treatment, respectively. For the variances showed under random effects section of above summary table. The values are matched with the variance table in task 2a. There are approximately 12%, 71%, and 18% of variances are explained by the following sources of variance: patch, patch/treatment interaction term, and residual, respectively.

Task 3a: Building a confidence interval interpreter

```
interpret_ci <- function(lower, upper, ci_level, stat){
  if(!is.character(stat)) {
    # produce a warning if the statement of the parameter isn't a character string
    # the spacing is a little weird looking so that it prints nicely in your pdf
    warning("
    Warning:
    stat should be a character string that describes the statistics of
    interest.")
  } else if(!is.numeric(lower)) {
    # produce a warning if lower isn't numeric
    warning("Warning:
             lower should be a numeric value that describes the lower bound
             of the confidence interval.")
  } else if(!is.numeric(upper)) {
    # produce a warning if upper isn't numeric
    warning("Warning:
             upper should be a numeric value that describes the lower bound
             of the confidence interval.")
  } else if(!is.numeric(ci_level) | ci_level < 0 | ci_level > 100) {
    # produce a warning if ci_level isn't appropriate
    warning("Warning:
             ci_level should be a numeric value between 0 and 100 that describes
             the confidence level this interval was calculated at.")
  } else{
    # print interpretation
    # this is the main skill I want to see, writing a good CI interpretation.
  str_c("With ", ci_level, "% confidence that ", stat, " is between ",lower,
        " and ", upper)
  }
}

# Test 1
ci_test1 <- interpret_ci(10, 20, 99, "mean number of shoes owned by students")

# Test 2
ci_test2 <- interpret_ci(10, 20, -1, "mean number of shoes owned by students")

# Test 3
ci_test3 <- interpret_ci(10, 20, -1, tibble(stat = 3))
```

CI function test 1: With 99% confidence that mean number of shoes owned by students is between 10 and 20

CI function test 2: Warning: ci_level should be a numeric value between 0 and 100 that describes the confidence level this interval was calculated at.

CI function test 3: Warning: stat should be a character string that describes the statistics of interest.

Task 3b: Building a p value interpreter

```r
# message=FALSE means we will not get the warnings
interpret_pval <- function(pval, nullhyp){
  if(!is.character(nullhyp)) {
    warning("Warning: nullhyp should be a character string that describes the
             null hypothesis.")
  } else if(!is.numeric(pval)) {
    warning("")
  }  else if(pval > 1) {
    warning("
            Warning: p value should be a numeric value between 0 and 1 inclusive
            , instead of greater than 1.")
  } else if(pval < 0){
    warning("
            Warning: p value should be a numeric value between 0 and 1 inclusive
            , instead of less than 0.")
  } else if(pval >= 0.1){
    str_c("The p value is ", round(pval, 3),
          ", we have no evidence against the null hypothesis: ", nullhyp, ".")
  } else if(pval >= 0.05){
    str_c("The p value is ", round(pval, 3),
          ", we have weak evidence against the null hypothesis: ", nullhyp, ".")
  } else if(pval >= 0.01) {
    str_c("The p value is ", round(pval, 3),
          ", we have some evidence against the null hypothesis: ", nullhyp, ".")
  } else if(pval >= 0.001){
    str_c("The p value is ", round(pval, 3),
          ", we have strong evidence against the null hypothesis: ", nullhyp,
          ".")
  } else {
    str_c("The p value is <.001, we have very strong evidence against the null
          hypothesis: ", nullhyp, ".")
  }
}

pval_test1 <- interpret_pval(0.0000000003,
                             "the mean grade for statistics students is the same as
                             ↪   for non-stats students")

pval_test2 <- interpret_pval(0.0499999,
                             "the mean grade for statistics students is the same as
                             ↪   for non-stats students")

pval_test3 <- interpret_pval(0.050001,
                             "the mean grade for statistics students is the same as
                             ↪   for non-stats students")

pval_test4 <- interpret_pval("0.05", 7)
```

p value function test 1: The p value is <.001, we have very strong evidence against the null hypothesis: the mean grade for statistics students is the same as for non-stats students.

p value function test 2: The p value is 0.05, we have some evidence against the null hypothesis: the mean grade for statistics students is the same as for non-stats students.

p value function test 3: The p value is 0.05, we have weak evidence against the null hypothesis: the mean grade for statistics students is the same as for non-stats students.

p value function test 4: Warning: nullhyp should be a character string that describes the null hypothesis.

Task 3c: User instructions and disclaimer

Instructions

Confidence Interval Interpreter:
The confidence interval interpreter helps you to interpret confidence intervals correctly. It takes the following inputs: 1. lower: a numeric value which is lower bound of the confidence interval. 2. upper: a numeric value which is upper bound of the confidence interval. 3. ci_level: a numeric value between 0 and 100 which is the confidence of level this interval was calculated at. 4. stat: a character sting description of the statistic of interest. The inputs must be entered in same order just stated. The confidence interval is a specific range

of values in which we believe the population parameter lies. The population parameter is a quantity of the population we are interested in that describes this population. An example of a population parameter is the average height of all men on Earth, referred to as the mean. The confidence level is the probability that the interval estimates would contain the population parameter. The ideas of confidence level and confidence interval could be easily mixed up. The confidence interval is not the probability that the population parameter would lie in. The population parameter either lies in the given range: 100% or does not appear in it at all: 0%. Also, some may say that the range of values that sample data/statistics would lie in is incorrect. Moreover, the confidence interval can only be used once. That is, you may not reuse the confidence interval with a repeated experiment of the same sample statistics.

P-value Interpreter:
The P-value interpreter helps you to interpret the p-value correctly. It takes the following inputs: 1. pval: a positive numeric value which is the p-value that you want to interpret. 2. nullhyp: a character string which is the null hypothesis. The inputs must be entered in the same order just stated. The null hypothesis is a statement we want to test, which is related to the population parameters. It's stated as the following: there is no relationship or difference between the population parameters we're interested in. For example, a correct null hypothesis about the average heights of all men and women is: there is no difference between the average heights of all men and women. We could only either reject or do not reject the null hypothesis, but for the wording of the comment made based on the p-value, we should use the strength of evidence instead. For example, if the p-value is greater than 0.1, we should say: "No evidence against the null."

Disclaimer

For the ***P-value Interpreter***, there are several things you need to be care of. Firstly, a p-value must be a positive number. A negative p-value is impossible. Please double check your work if you obtain a negative p-value. Secondly, do not directly connect the p-value to alternative hypothesis or put it as an input of the interpreter. P-value should always bond with the null hypothesis. There is only a very little chance to interpret alternative hypothesis directly with p-value. Thirdly, the interpreted would round the p-values in two decimal places. Therefore, if you enter 0.49999 and 0.50001, then they would have different interpretations printed but with same p-values in it: 0.50. Lastly, the thresholds are set to be lower inclusive. That is, the p-value ranges includes the lower bound in it. For example, a p-value: 0.05 would be considered as weak evidence against the null hypothesis (in range $0.05 <= \text{p value} < 0.1$), not Moderate/some evidence against the null hypothesis.

Task 4: Creating a reproducible example (reprex)

Reprex is the minimal example you want to show others, which helps them reproduce your errors or any other things you want to demonstrate to them. To make a working reprex, you need to consider the following: 1. What are the packages required for the selected code? These packages make sure that the code runs without missing packages issue so that others can run your reprex. Therefore, you should include the code of installing/importing packages needed for the reprex. 2. Code you selected is enough for reprex. Too many unnecessary lines or not enough lines would cause problems for others when they reproduce your work. 3. Make sure you select the code. Otherwise, you will not be able to create the correct reprex since no code source is given.

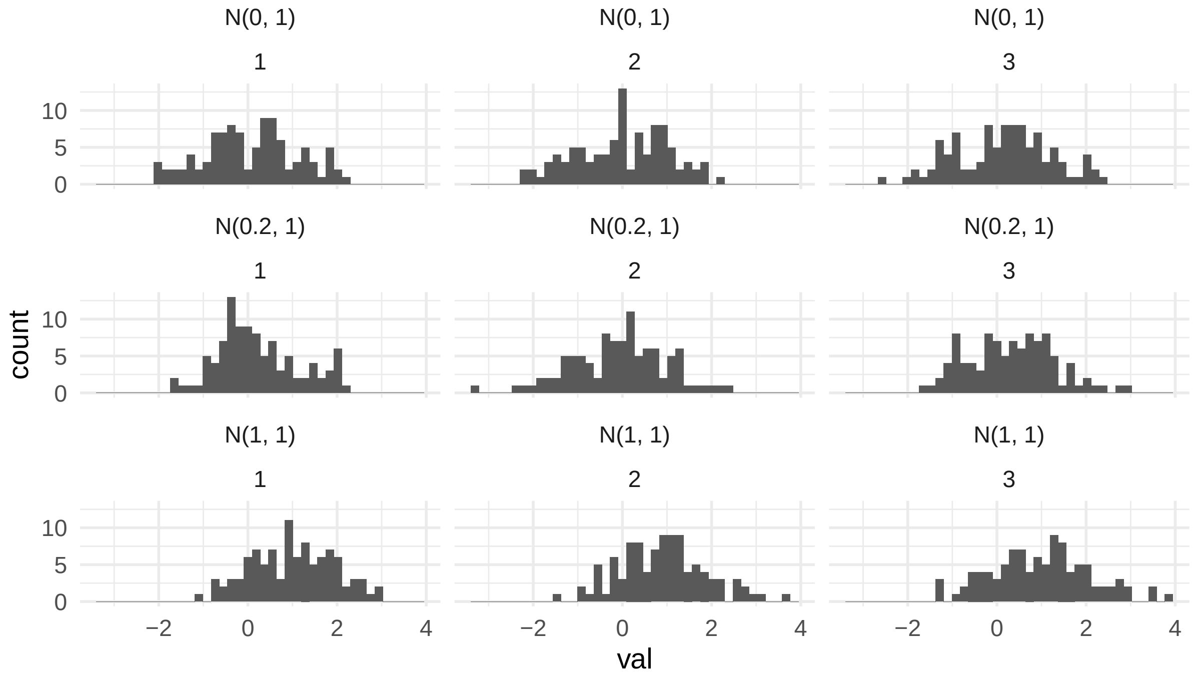
```r
library(tidyverse)
my_data <- tibble(group = rep(1:10, each=10),
                  value = c(16, 18, 19, 15, 15, 23, 16, 8, 18, 18, 16, 17, 17,
                            16, 37, 23, 22, 13, 8, 35, 20, 19, 21, 18, 18, 18,
                            17, 14, 18, 22, 15, 27, 20, 15, 12, 18, 15, 24, 18,
                            21, 28, 22, 15, 18, 21, 18, 24, 21, 12, 20, 15, 21,
                            33, 15, 15, 22, 23, 27, 20, 23, 14, 20, 21, 19, 20,
                            18, 16, 8, 7, 23, 24, 30, 19, 21, 25, 15, 22, 12,
                            18, 18, 24, 23, 32, 22, 11, 24, 11, 23, 22, 26, 5,
                            16, 23, 26, 20, 25, 34, 27, 22, 28))

my_summary <- my_data %>%
  summarize(group_by = group, mean_val = mean(value))

glimpse(my_summary)
#> Rows: 100
#> Columns: 2
#> $ group_by <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3...
#> $ mean_val <dbl> 19.67, 19.67, 19.67, 19.67, 19.67, 19.67, 19.67, 19.67, 19.67...
```

Task 5: Simulating p-values

Setting up simulated data

```
# Sets up simulated data
set.seed(last3digplus)

sim1 = tibble(group = rep(1:1000, each = 100), val = rnorm(100000, 0, 1))

sim2 = tibble(group = rep(1:1000, each = 100), val = rnorm(100000, 0.2, 1))

sim3 = tibble(group = rep(1:1000, each = 100), val = rnorm(100000, 1, 1))

# Stack the datasets together into one new dataset
all_sim = bind_rows(sim1, sim2, sim3, .id = "sim")

# Create sim_description
# Dataset to merge with improved simulation names
sim_description <- tibble(sim = 1:4, desc = c("N(0, 1)", "N(0.2, 1)",
                                              "N(1, 1)", "Pois(5)"))

# Make the values of sim column in all_sim be numeric values
all_sim <- all_sim %>% mutate(sim = as.numeric(sim))

# Join all_sim and sim_description
all_sim <- all_sim %>% left_join(sim_description, by = "sim")
```

```
# Plot the histograms
all_sim %>%
        filter(group <= 3) %>%
        ggplot(aes(x = val)) +
        geom_histogram(bins = 40) +
        facet_wrap(desc~group, nrow = 3) +
        theme_minimal() +
        labs(caption = "Created by Jiahao(Green) Bai in STA303/1002, Winter 2022")
```



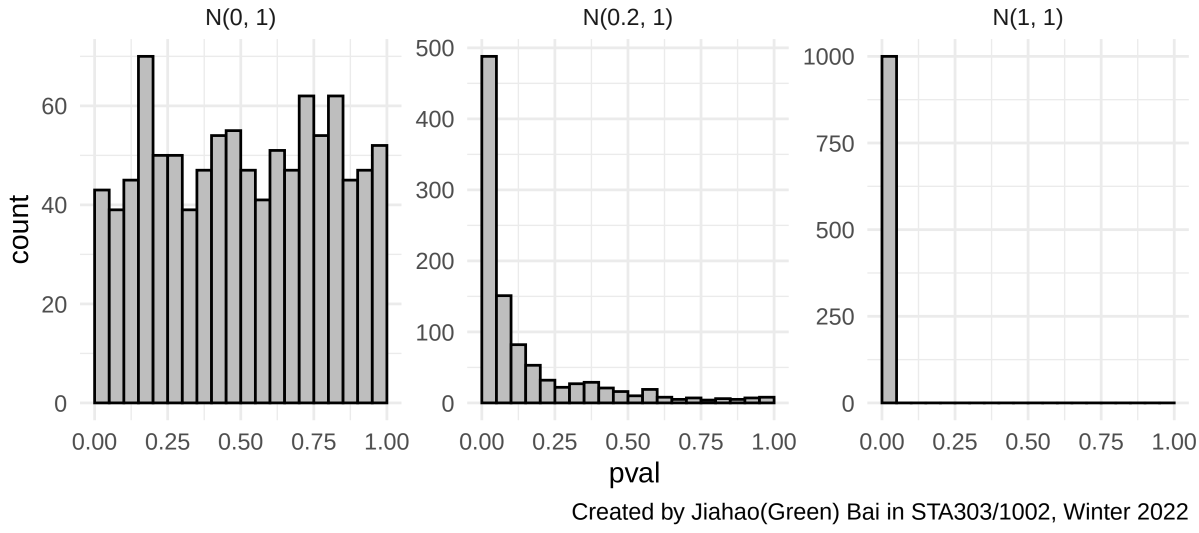
Created by Jiahao(Green) Bai in STA303/1002, Winter 2022

Figure 2: Histograms for The First Three Groups for Simulated Datasets: sim1, sim2, sim3

Calculating *p* values

```r
# Calculate p values
pvals <- all_sim %>% group_by(desc, group) %>%
         summarise(pval = t.test(val, mu = 0)$p.value, .groups = "drop")
```

```r
# Plot the p values
pvals %>% ggplot(aes(x = pval)) +
          geom_histogram(boundary = 0, binwidth = 0.05, fill = "grey",
                         color = "black") +
          xlim(0, 1) +
          facet_wrap(~desc, scales = "free_y") +
          theme_minimal() +
          labs(caption = "Created by Jiahao(Green) Bai in STA303/1002, Winter 2022")
```

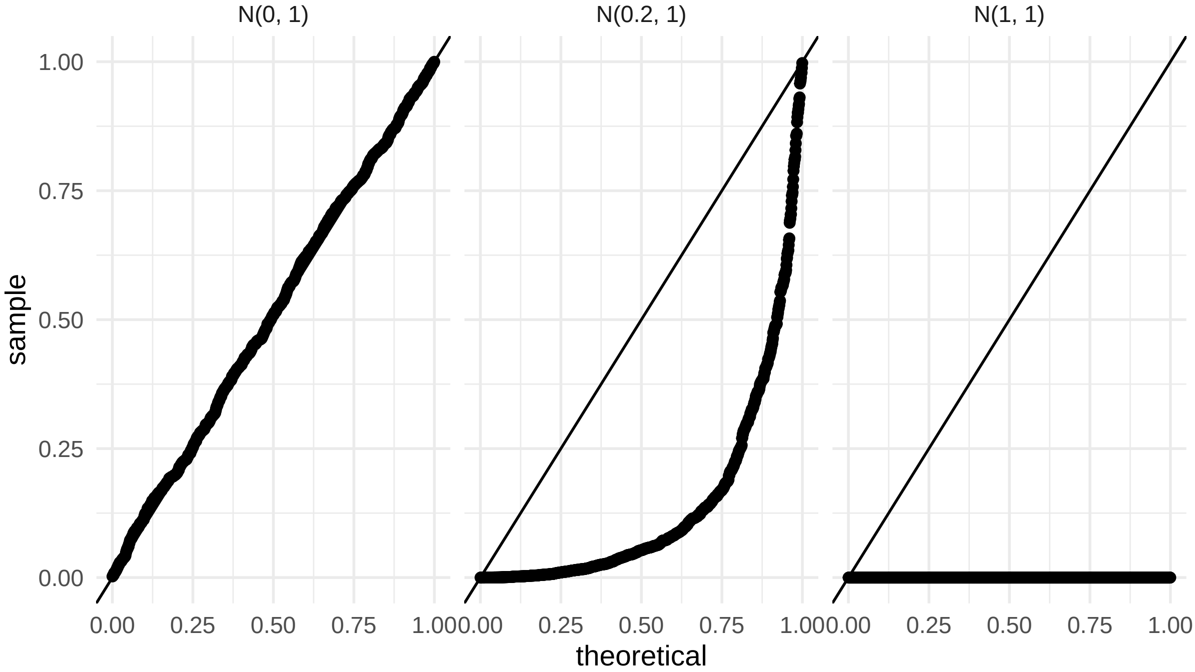


Created by Jiahao(Green) Bai in STA303/1002, Winter 2022

Figure 3: Histograms of p-values Under Different Normal Distributions

Drawing Q-Q plots

```r
# Plot the QQ plots for each simulation
pvals %>%
  ggplot(aes(sample = pval)) +
  geom_qq(distribution = qunif) +
  geom_abline(intercept = 0, slope = 1) +
  facet_wrap(~desc) +
  theme_minimal() +
  labs(caption = "Created by Jiahao(Green) Bai in STA303/1002, Winter 2022")
```



Created by Jiahao(Green) Bai in STA303/1002, Winter 2022

Figure 4: Q-Q plots for each simulation

Conclusion and summary

P-value is an essential concept in hypothesis testing. In this task, the definition of p-values connects to the histograms and Q-Q plots. According to the histograms of p-values of three normal distributions with different parameters, the p-values of N(0, 1) show a uniformly distributed pattern. Meanwhile, the p-values of the other two normal distributions with non-zero means are not uniformly distributed. Also, the range of these p-values is between 0 and 1. From the Q-Q plots, p-values of N(0,1) are uniformly distributed on the diagonal line while p-values of the other two plots barely touch the diagonal line. Therefore, when the null hypothesis is true, the distribution of p values is Uniform(0, 1). The definition of p-value claims that p-value should be a number between 0 and 1, which agrees on the range of p-values in both histograms and Q-Q plots. Consequently, the correct answer should be: approximately 10% of the p-values will be between 0.9 and 1. Other choices are wrong since we could make claims about p-values when the null hypothesis is true, and due to the uniform distribution between 0 and 1, p-values would not cluster at one point.

Writing sample

Rigorous and precise are essential concepts in statistics. Harvey J. Motulsky writes an article introducing a few common misconceptions about data analysis and statistics. After reading this article, I learn about p-hacking and agree with the idea of providing details in our work.

To begin with, as a statistic student, I have completed many data analysis projects. Most of them conduct model optimization, such as variable selection. One of the multiple linear regression models I worked with contains numerous predictors. Since a small p-value of a predictor indicates significance, I deliberately manually select or use stepwise regression to obtain a combination of predictors that only include significant predictors with the lowest possible p-values. Referring to the article, the author introduces various forms of p-hacking and results from data that involves p-hacking in the p-hacking misconception section. Surprisingly, the idea of trying various combinations of independent variables to include in a multiple regression is a typical example of P-hacking (Motulsky, 2014). This idea blows my mind, and now I realize doing so would make my work biased instead. From this, I would be more careful about variable selection and less restrictive to p-values in my future career or study. Furthermore, I should label any conclusions “preliminary” if p-hacking is involved, as Motulsky suggests (Motulsky, 2014). Therefore, my work could be more rigorous and close to actual values.

The last misconception that Motulsky mentions is: you do not need to report the details. Motulsky mainly focused on examples of providing details that increase reproducibility in this section. I always consider reproducibility is vital in statistics. For others to reproduce the work you have done, it’s crucial to include details of the research. If there is any programming part in the work, one should provide commented codes so that others can trace the codes and reproduce the results quickly. If any changes are made in your data or analysis methods, you should point them out and be clear about how you changed. Motulsky (2014) suggests that if you eliminate any outliers, state how many outliers you eliminated, the rule used to identify them, and a statement whether this rule was chosen before collecting data. I agree with it, and it coincides with my opinion stated above. There are more suggestions like this that Motulsky says, which I could use as reminders for my future works. Thus, my work will be more reproducible.

Overall, there are five common misconceptions that Motulsky discusses in the article, and I’m interested in two of them: p-hacking and reporting details. From this, I learned to avoid p-hacking as much as possible, specify my conclusion as “preliminary” if there is any p-hacking, and include more information in my future projects.

Word count: 451

References

> Motulsky, H. J. (2014). Common misconceptions about data analysis and statistics. *Naunyn-Schmiedeberg's Archives of Pharmacology*, *387*(11), 1017–1023. https://doi.org/10.1007/s00210-014-1037-6

Reflection

What is something specific that I am proud of in this portfolio?

There are three things I'm proud of in this portfolio. Firstly, I'm proud that I keep the habit of commenting on my R codes. Doing so would allow people tracing my code to know what I'm doing instantly. Codes without comments are hard to follow and confuse the reader. I did my best to provide a comment for each major step to avoid that. The comments are more informative and shorter than what I did in the mini-portfolio project. Another thing that I'm proud of is the instruction part in task 3c. I brought the idea of docstring of python functions into this part and combined it with the interpretations of confidence intervals and p values. I explained the critical concepts, but I also did a brief description of the inputs of interpreters so that the users would not pass something weird to the function. Lastly, I'm proud that I could make graphs faster than before. I'm more familiar with the graphing functions this time. The graphs I made in this portfolio require less time to create, and they look better than the graphs in the mini-portfolio. Overall, these are the things I'm proud of.

How might I apply what I've learned and demonstrated in this portfolio in future work and study, after STA303/1002?

Data collected from the real world can be messy and contains problems. In future work or studies, I may encounter data with dependency issues. Therefore, linear regression can not be used since independence is one of its assumptions. The linear mixed-effects model I learned and used in this portfolio would be an ideal choice since it's designed to deal with dependent data. Also, task 3a and 3b helped me practice writing and improving a function. Therefore, I could build functions to deal with works that share the same logic to increase my reproductivity in the future, such as a function for repeated calculations with different parameters. Moreover, task 3c and task 5 make me have a deeper understanding of confidence interval and p value, so I could explain them better whenever I have to. The reprex I learned in task 4 allows me to ask questions in detail. I could let others reproduce my work quickly in my future inquiries. Finally, all of the writings in this portfolio could help me write shorter, more informative, and understandable statistics reports that would be present to the general audience in the future.

What is something I'd do differently next time?

I would be more patient with project instructions for future projects and do better time management. The week before reading week was filled with midterms and assignments, so I did not have too much time for this portfolio. I could finish the first three tasks of this portfolio before the Wednesday class, but I kept switching the material I was working with. My time was shattered into pieces. More specifically, I was doing this portfolio one minute, but the next minute I was looking at other subjects, such as environmental science, which has no relationship with statistics. This lousy time management made me suffer from resuming what I was doing for the portfolio. Slowly, I lost my patience, and I missed many parts in task 2. Task 2 was barely finished on Wednesday. Consequently, I have to ask for an extension. Thus, I definitely would plan my time and focus on one thing at a time. Also, I would pay more attention to the instructions so that I don't have to find out what I missed when some errors pop out saying something is not defined.