Nominal Categorical Response in Regression-Based Analysis

Benjamin Bui

May 20, 2024

# Introduction

Categorical variables provide an inherent problem when attempting to describe relationships mathematically. Many analyses we run involve regression analysis which requires that variables be treated numerically. The simple sounding solution to categorical variables then is to convert them into a numeric, either via a direct scale mapping or by converting to {0,1} dummy variables. This memo argues why this method, when applied to non-binary dependent variables is problematic statistically and offers alternative solutions to the issue.

This memo is divded into three main sections. The first section describes categorical variables, both statistically and how they are handled in . This is followed by a statistically argument against the use of multiple dummy variables as a proxy for a nominal categorical variable when dealing with a multinomial nominal categorical response variable, i.e. a categorical outcome with more than 2 distinct levels. Finally, we go through solutions for how to actually handle this statistical problem in .

## Table of Contents

[1. Introduction 1](#_Toc174096388)

[1.1. Table of Contents 1](#_Toc174096389)

[2. Categorical variables, in general and in 2](#_Toc174096390)

[2.1. What is a categorical and dummy variable? 2](#_Toc174096391)

[2.2. Variable Types as classes 4](#_Toc174096392)

[2.2.1. A slightly deeper dive in factors and their implementation in 4](#_Toc174096393)

[3. Why not to use dummy variables 5](#_Toc174096394)

[3.1. P-value abuse with dummy responses 6](#_Toc174096395)

[4. How to avoid dummy response variables 8](#_Toc174096396)

[4.1. The simple test 9](#_Toc174096397)

[4.1.1. Non-binary analysis 10](#_Toc174096398)

[4.2. Multinomial Logistic Regression (MNL) 11](#_Toc174096399)

[4.2.1. -Computation 15](#_Toc174096400)

[4.2.2. Seemingly Unrelated Regression (SUR) 15](#_Toc174096401)

[4.3. Linear mixed-effects model without “lower-level” predictors 17](#_Toc174096402)

[4.4. Linear mixed-effects model with higher-level predictors 21](#_Toc174096403)

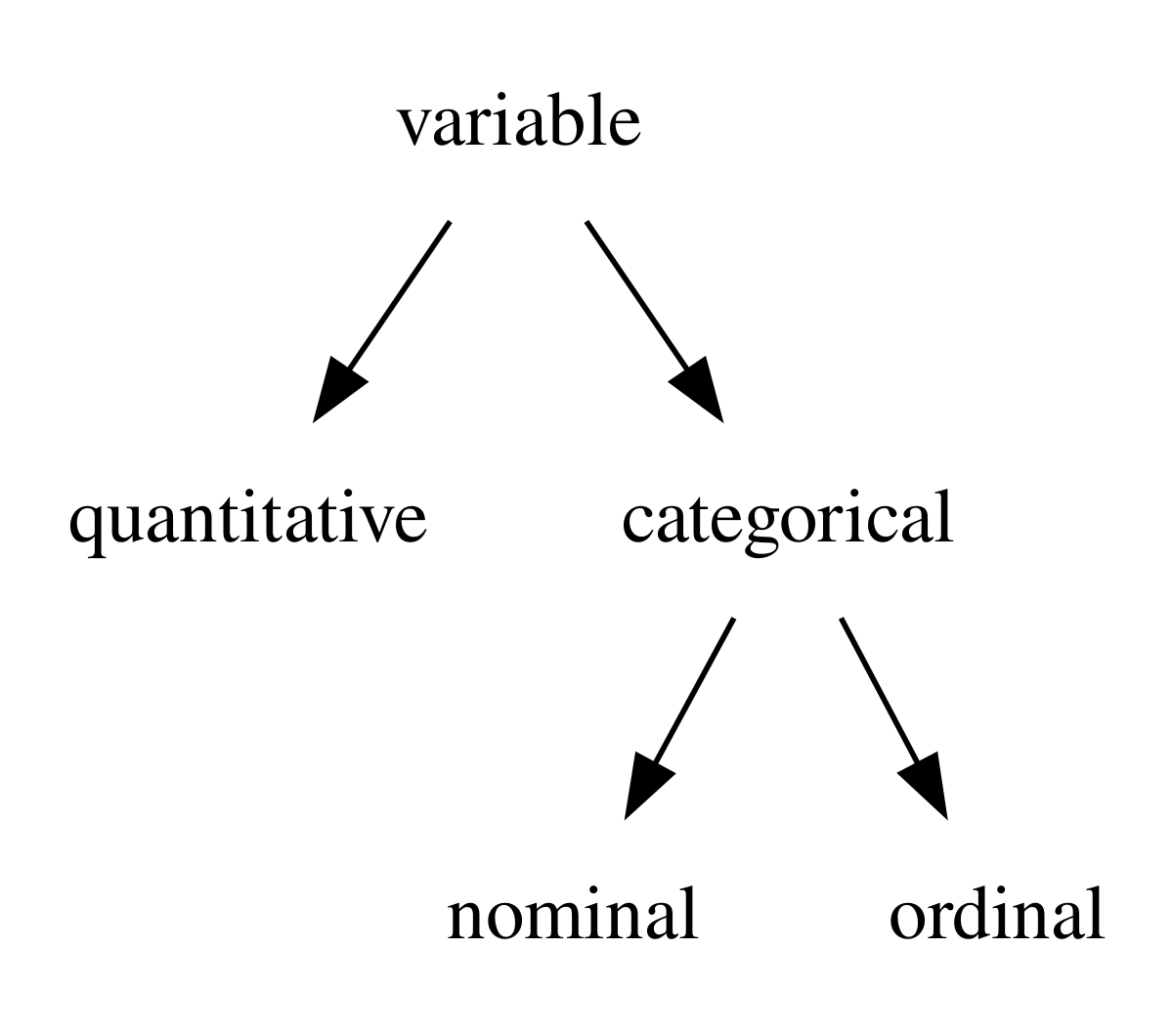
[4.4.1. Alternative methods for mixed-effect models 23](#_Toc174096404)

[5. References 23](#_Toc174096405)

# Categorical variables, in general and in

## What is a categorical and dummy variable?

For our purposes, we can characterize variables as one of three types: quantitative, nominal categorical, ordinal categorical (Sinharay, 2010). There is also a meaningful distinction between categorical variables with just 2 levels, i.e. binary, and those with more than 2 levels that we will go into shortly.



A variable is quantitative if it is inherently ordered and have an internally consistent, scalable definition of difference. For example, consider a variable representing age. It is ordered in that we can always state that one value of age is greater than or equal to any other value of age and this property is transitive. The difference property can be seen when comparing ages. We can describe age with arbitrary precision, say 5.5 years or 5.44444444449 years. Moreover, the distance from 5 years of age to 5.5 years of age is the same as 5.5 years of age to 6 years of age. Although this seems trivial, categorical variables break some of these assumptions. In practically all but the most extreme cases, it is preferrable to conduct analyses using quantitative variables as the dependent when both options are present.

A variable is categorical if it is missing the property of “internally consistent, scalable defintion of difference”. For example, suppose we have a variable describing fruits with the possible values {“apple”, “orange”, “grape”}. These three values are different but we don’t have any fruit that is say 50% of the way between an “apple” and a “grape” the same way we could have said that 5.5 years is 50% of the way between 5 years and 6 years of age. Moreover, even if you did try to devise up some apple-grape gradient, this wouldn’t make any sense when applied to an orange. Categorical variables can be divided into nominal or ordered. Nominal categoricals lack inherent ordering such you can’t compare different values of this categorical and make a statement about one being greater than or less than another. Ordered categoricals do have an inherent ordering such that we can always state that some level is greater than another level and that if level and , then . Our previous example of {“apple”, “orange”, “grape”} is a nominal categorical as we aren’t treating any of the fruit types as greater than the other. However, suppose we have an education level cateogrical: {“Less than high school”, “High school”, “Bachelors degree”, “Greater than bachelors degree”}. The values of this variable can be ordered: {“Less than high school” < “High school” < “Bachelors degree” < “Greater than bachelors degree”}.

It is important to consider that many variables can be interpreted to quantitive or categorical. For example, consider age in the form of number of years. This variable can be interpreted as quantitative and is often encoded as just a number. However, we could also re-encode age as a categorical variable with possible values {“3 or younger”, “4 or older”}. This re-encoding technically loses information, especially if our original source has more granularity in age but it is still a valid operation. Likert scales are a classic example of an ordered categorical variable; however, these are often re-encoded into continuous scales for the purposes of measure creation or aggregation despite the lack of rigor in ensuring that the difference between “Strongly Disagree” and “Disagree” is the same as the difference between “Disagree” and “Indifferent”.

Both nominal and ordinal categorical variable can be converted into a set of binary variables. Each category corresponds to a binary variable. . Binary Dummy variables are typically coded as 1 for those observations in the category and 0 otherwise. It is typical for a variable with n possible values to create n new variables; however, creating variables contains the same amount of information. It is typical for a variable with possible values to create new variables; however, creating variables contains the same amount of information. For example consider a categorical variable encoding the color options “Blue”, “Green” and “Orange”. This could be separated into dummy variables, a Blue variable that is 1 if “Blue” and 0 if otherwise, a Green variable that is 1 if “Green” and 0 if otherwise, and an Orange variable that is 1 if “Orange” and 0 if otherwise. However, if we only include dummy variables, say by keeping only the Green and Blue dummy variables, we still know that the object is “Orange” if both Green and Blue are 0.

## Variable Types as classes

These 3 types of variables: quantitative, ordinal categorical, and nominal (unordered) categoricals are actually treated as separate classes of vector in R. The class of a vector, which can be found by calling class() on an object, has some distinction from the type of a vector which can be found by calling typeof() on an object. Quantitative variables should be represented by the numeric class, nominal categorical variables should be represented by the factor class and ordered categorical variables should be possess both the “ordered” and “factor” classes when passing these objects into a modelling function.

It is important to understand what ordered factors are. As stated previously, these are used to encode ordinal categoricals. Ordered factors are treated differently in analyses and so should only be used if you understand the implications of them in your model.

### A slightly deeper dive in factors and their implementation in

R has 25 different types of vectors at its lowest level, most relevant here are integer, double, and character (*Advanced R, Second Edition*, 2019). The double type, oftentimes confusingly called the numeric type in encomapasses all real numbers (technically anything with absolute value less than and greater than ) and should be used for any quantitative variable being passed into a modelling function. The integer type encompasses all real integers up to about and isn’t used very extensively in such that you don’t need to ever consider their direct usage when modelling. The integer and double type are both under the numeric mode which is why both display TRUE when calling is.numeric. The character type is used to encode all strings, including numbers that should not be treated quantitatively like ID number.

Notably, factor is not included here because factor is a class, not a type which is a higher level of object definiton. Factors are actually implemented as an integer vector with the addition of an attribute called levels. The levels attribute is assigned a character vector representing the sample space of your categorical variable. The integer vector component effectively functions as the actual storage of the data and the levels attribute serves as a map between the integers and what they actually represent. You can think of a factor like a numbered coloring book where the levels are a guide saying 1 corresponds to a category, 2 corresponds to another category and the integer vector component is the actual picture, or vector we are trying to store. Unless the factor is ordered, the integers do **NOT** store or intend any meaning as a number. They are only stored as integer because it is space efficient to do so. Adding ordered as a class to a factor, usually by setting ordered = TRUE when defining a factor just adds ordered as a class but doesn’t do anything else to the underlying implementation.

The difference arises when we use these variables as predictors in a model. We’ll go into more detail later but as an example, stats::lm coerces plain factor into dummy variables and runs the model normally. However, an ordered factor goes another step by applying polynomial contrasts. How this exactly works depends on the amount of distinct levels in the ordered factor but say for a three-level ordered factor, R would report coefficients describing how much the relationship between the ordered factor resembles a linear and a quadratic relationship [See here for more details](https://people.linguistics.mcgill.ca/~morgan/qmld-book/practical-regression-topics-2-ordered-factors-nonlinear-effects-model-predictions-post-hoc-tests.html) Neither treatment is inherently wrong, it is just important that you select the appropriate method for your data and purpose although you should be using unordered factors in most circumstances. Also, factor and character are both treated as nominal categorical variables by practically all modelling p ackages, the primary advantage of factor over character is the higher space efficiency of the vector and its ability to store levels that could have existed but do not exist in the data, e.g. an option in a survey that was never selected but was possible to select.

The crux of this confusion arises because the order of strings in the levels attribute in an unordered factor does have practical meaning even if it doesn’t have a mathematical or statistical meaning. For example, if we want to sort a dataset or plot such that certain values of a categorical variable appear first, then a simple way to do this is by explicitly setting the levels such that the values we want to see first are first in the input character vector. The first value in the levels attribute is also used when calculating contrasts as the “baseline”. For example, if we have a race categorical variable with “White” as the first value of levels, then computed contrasts will show the effects of “Black” compared to “White”, of “Asian” compared to “White”, and so on. This doesn’t at all change the computation of the model but it does affect the direct interpretability of coefficients. Again though, unless you desire the resulting analytical implications, you shouldn’t add the ordered class to your factor when doing this, just changing the levels attribute is sufficient. A simple way of doing that is by using stats::relevel or forcats::fct\_relevel

# Why not to use dummy variables

We will now be considering how we want to treat nominal categorical variables as a response variable.

It is accepted practice to transform nominal categorical variables into dummy variables when used as a predictor variable; in fact, performs this step automatically if you include a factor as a predictor variable in your formula. It seems logical then to just do the same with a response variable; just run the model times for each dummy variable. First consider the non-rigorous implications if only 1 of these analyses provides a statistically significant p-value Does that mean our research condition or explanatory variable only effectively predicts for one possible value of our categorical? That conclusion is somewhat difficult to reconcile with the tautological fact that the statistically significant case is necessarily collinear with a set of non-statistically significant variables. For this reason, a single, shared statistical test when evaluating a categorical variable such that we have a single measure of significance would be preferred.

Another important consideration pertains to the implications of the definition of a p-value. In impact analyses, a p-value of 0.05 states that we reject the null hypothesis and conclude our results are unlikely to have occurred if the true population treatment and control groups were the same. Implicit in this is the fact that these differences can be observed even if the null hypothesis, that the two groups are the same, is true in reality. However, if one runs more and more hypothesis tests, this can be abused, as it becomes more likely we will generate significant p-values by random chance. (Stefan & Schönbrodt, 2023). Running a test on multiple dummy variables provides a seemingly innocent way to accomplish this.

## P-value abuse with dummy responses

Multinomial categorical response variables are often of interest at MDRC, say if we want to deduce impacts on categorical hourly wages or letter grades. Even in cases where a quantiative version of the data may exist as in wages and grading, we may not have access to that fine level of data or we may be interested in the impacts on the discrete categories in particular.

Here I will be creating a randomly generated dataset in order to demonstrate the potential pitfalls of using dummy variables as a proxy for a nominal categorical variable in the context of a response variable.

Here I am making a dataset of 1000 observations with the following variables:

* Environment - Random sampling with choices: {“Urban”, “Suburban”, “Rural”}
* Education - Random sampling with choices: {“Bachelors”, “Graduate”}
* Month - Random sampling with choices: {“Jan”, “Feb”, “Mar”, “Apr”, “May”, “Jun”, “Jul”, “Aug”, “Sep”, “Oct”, “Nov”, “Dec”}
* State - Random sampling with choices: {“NY”, “CA”, “DC”}
* Letter - Random sampling from all available lowercase letters in the english alphabet

suppressPackageStartupMessages(library(magrittr))  
suppressPackageStartupMessages(library(dplyr))  
# Setting random seed for consistency in reference  
set.seed(-1619558820)  
# Generating dataset with 5 nominal categorical variables drawn from a uniform  
# distribution  
test\_categoricals <- data.frame(  
 Environment = sample(factor(c("Urban", "Suburban", "Rural")), 1000, replace = TRUE),  
 Education = sample(factor(c("Bachelors", "Gradauate")), 1000, replace = TRUE),  
 Month = sample(factor(month.abb), 1000, replace = TRUE),  
 State = sample(factor(c("NY", "DC", "CA")), 1000, replace = TRUE),  
 Letter = sample(factor(letters), 1000, replace = TRUE),  
 # The variable we are using as our independent  
 Intervention = sample(factor(c("Control", "Intervention")), 1000, replace = TRUE)  
)

# Creating dummied version of dataset  
test\_dummies <- fastDummies::dummy\_cols(  
 test\_categoricals,  
 c("Environment", "Education", "Month", "State", "Letter"),  
 remove\_selected\_columns = TRUE  
)

Since our variables are created randomly, there should be no relationship between the intervention group and any of the nominal categorical generated. However, running a simple linear model and looking at the results suggests otherwise.

# Running linear model comparisons  
test\_comparisons <- mdrcAnalysis::lm\_extract(test\_dummies,  
 .dependents = setdiff(  
 names(test\_dummies),  
 "Intervention"  
 ),  
 .treatment = "Intervention",  
 .inc\_sample = FALSE,  
 .inc\_trail = FALSE  
)

In lieu of printing the results of each test, lets take a look at how the significance levels are distributed. Remember again that the parametric truth of the dataset is there are no associations among the variables, so no differences should be statistically significant.

| p<X | n |
| --- | --- |
| 0.01 | 1 |
| 0.05 | 2 |
| 0.1 | 4 |
| >=0.1 | 39 |

We see no statistical significance in 39 of our 46 tests using a threshold of 0.1; however, there are 7 tests that display some significance, or about 15% of the results. First, looking at this from the perspective of a multiple comparisons problem, we effectively increased the number of statistical tests performed from 5 - testing on Environment, Education, Month, State, and Letter to 46 different tested. This truth is shown in 75 of our 47 tests; however, there are 7 tests that display some significance with a p-value of 0.1 or lower, or about 15% of the results. First we’ll consider this result in the context of the multiple comparisons problem.

The multiple comparisons problem, to simplify, states that the likelihood of actually seeing an erroneous statistical test increases as the number of tests increase. With a threshold of 0.1, we would expect that, even with randomly generated samples, about 10% of our tests should show statistical significance. If we report these significant results while providing context on the multiple other tests that do not show significance, then the multiple comparisons problem is of relatively minor concern as we should be expecting that 10% of the results are erroneously significant.

The multiple comparisons problem arises whenever we have a large amount of statistical tests which is amplified by creating multiple dummy tests for each categorical variable. Another issue, more specifically to using dummies as a proxy for multinomial categorical variables as a response variable is one of collinearity and interpretability. Consider our Environment variable which has 3 possible values. If we make all possible dummy variables of this variable, we end up with three, perfectly collinear variables because each dummy variable must be, by definition, mutually exclusive.

test\_dummies %>%  
 count(Environment\_Rural, Environment\_Suburban, Environment\_Urban)

| Environment\_Rural | Environment\_Suburban | Environment\_Urban | n |
| --- | --- | --- | --- |
| 0 | 0 | 1 | 331 |
| 0 | 1 | 0 | 338 |
| 1 | 0 | 0 | 331 |

This introduces a multicollinearity problem when viewing the full array of tests in aggregate. There is also a significant interpretability issue when a statistically significant difference is found with some but not all of the dummy variables representing a categorical variable.

test\_comparisons %>%  
 select(Dependent, ProbF) %>%  
 filter(str\_detect(Dependent, "Environment"))

| Dependent | ProbF |
| --- | --- |
| Environment\_Rural | 0.03742112 |
| Environment\_Suburban | 0.10963481 |
| Environment\_Urban | 0.63710155 |

Here, we see that Rural is below our threshold of 0.1, Suburban is slightly above it, and Urban is well above our threshold. Without a single, aggregate measure of significance, we can’t rigorously say whether our Intervention has a significant impact on Environment as it doesn’t make very much sense to claim that Intervention impacts Rural but not Suburban or Urban given that we know that these variables are collinear. The aim then of the methods present henceforth are aimed to create this aggregate measure of significance to create a more interpretable result that also avoids compounding on our multiple comparisons problem.

# How to avoid dummy response variables

After having hopefully cleared up why the use of dummy variables as a response variable is inherently problematic, we see that there is a need to analyze nominal categorical variables. The simplest solution is to just use the continuous variable instead of the categorical variable if there exist two analogous encodings of the same information. In fact, the selection of category boundaries itself is a method that, intentionally or not, can generate false significance (Stefan & Schönbrodt, 2023). However, if we still want to or must use a categorical variable as our response, there are methods to do so. Technically, binary dependent variables “should” be run with a logistic regression model although practically speaking, with an RCT, a linear model can offer an unbiased estimate with more simply interpretable results when testing for impacts. Please see this [QMG memo](https://mdrc365.sharepoint.com/:b:/r/sites/QuantMethods/Shared%20Documents/Guidance/QMG-Memo-on-Binary-Outcomes%20(January%202024).pdf?csf=1&web=1&e=0lM9V5) for more details.

In the case of non-binary nominal categorical the process is more complex. Below I offer two methods depending on the type of model you are running on the analogous quantitative variables. These focus on providing a technical guide on using R to calculate these results.

## The simple test

Although a bit of an oversimplification, a test serves an analogous function to categorical variables as a -test is to quantitative variables. For the purposes of this memo, you would use a test when attempting to determine the relationship between two categorical variables, e.g. race and your research group. It is not useful when attempting to account for other “covariates” in the relationship (although the related log-linear analysis can be used with categorical variables, that’s outside the current scope of this memo). These tests are often used at MDRC for early investigations of data regarding baseline equivalence and non-response bias, both cases where there is a relatively clear categorical vs categorical relationship we are attempting to ascertain.

Programming a test in is fairly simple. Here we are using the mdrcAnalysis::sim\_data\_robust\_reg and comparing Employment (employed\_01) and treatment group (treatment). Both of these are dummy variables which also allows us to compare these results against a -test treating them like quantitative variables.

Here both variables are encoded as numerics. Note that this test requires passing of direct atomic vectors to the function, not a dataset and the names of the variables you want. stats::chisq.test accepts both numeric and factor variables and treats them identically.

data(sim\_data\_robust\_reg, package = "mdrcAnalysis")  
# Noting here that Yates' continuity correciton is automatically  
# applied here but can be disabled by setting correct = FALSE  
chisq.test(  
 sim\_data\_robust\_reg$employed\_01,  
 sim\_data\_robust\_reg$treatment  
)

##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: sim\_data\_robust\_reg$employed\_01 and sim\_data\_robust\_reg$treatment  
## X-squared = 60.527, df = 1, p-value = 7.256e-15

If you want to extract this data, say for output into Excel or a table, the simplest option is to use the broom package although direct subsetting is doable if there is a specific parameter you want. Please read ?stats::chisq.test for details on what these columns are referring to.

library(broom)  
tidy(chisq.test(  
 sim\_data\_robust\_reg$employed\_01,  
 sim\_data\_robust\_reg$treatment  
))

| statistic | p.value | parameter | method |
| --- | --- | --- | --- |
| 60.5275 | 7.255772e-15 | 1 | Pearson's Chi-squared test with Yates' continuity correction |

If we compare these results to a paired two-sided -test, we see slightly different results in our test p-value.

t.test(  
 sim\_data\_robust\_reg$employed\_01,  
 sim\_data\_robust\_reg$treatment,  
 paired = TRUE  
)

##   
## Paired t-test  
##   
## data: sim\_data\_robust\_reg$employed\_01 and sim\_data\_robust\_reg$treatment  
## t = 8.2071, df = 322, p-value = 5.511e-15  
## alternative hypothesis: true mean difference is not equal to 0  
## 95 percent confidence interval:  
## 0.1741833 0.2840210  
## sample estimates:  
## mean difference   
## 0.2291022

Both p-values are low, on the order of but the test is testing for independence of these two categorical variables but the paired t.test is testing for the difference in the population mean of the two variables, i.e. the test is treating the two variables as categorical but the t.test is treating them as quantitative. Both of these tests show a statistically significant result but the difference in interpretation does result in variable p-values.

### Non-binary analysis

Using a non-binary variable does not at all complicate the process of performing a ^2 analysis. Here I am using the same dataset but replacing Employment with Favorite Animal (favAnimals), a trinary variable.

chisq.test(sim\_data\_robust\_reg$favAnimals, sim\_data\_robust\_reg$treatment) %>%  
 tidy()

| statistic | p.value | parameter | method |
| --- | --- | --- | --- |
| 1.674965 | 0.4327987 | 2 | Pearson's Chi-squared test |

Here we don’t see evidence of a relationship between favAnimals and treatment. However, if you did see a statistically significant relationship, then you can perform a post-hoc test to try and identify which categorical levels are “driving” the difference.

chisq.posthot.test is a package for quickly performing a post-hoc test using the bonferroni method. To determine which combination of variables are driving the difference, you can just look for statistically significant p-values in the resulting output.

library(chisq.posthoc.test)  
chisq.posthoc.test(  
 # Creating contingency table for test  
 table(sim\_data\_robust\_reg$favAnimals, sim\_data\_robust\_reg$treatment),  
 # Explicitly specifying method although this is default  
 method = "bonferroni"  
)

| Dimension | Value | 0 | 1 |
| --- | --- | --- | --- |
| cat | Residuals | 1.0618930 | -1.0618930 |
| cat | p values | 1.0000000 | 1.0000000 |
| croc | Residuals | 0.1166707 | -0.1166707 |
| croc | p values | 1.0000000 | 1.0000000 |
| dog | Residuals | -1.1753351 | 1.1753351 |
| dog | p values | 1.0000000 | 1.0000000 |

## Multinomial Logistic Regression (MNL)

A test offers a very quick and simple comparison of a singular categorical variable against another categorical variable. However, if we want or need to perform more complex analyses, say if we want to account for any covariates in the relationship then tests are insufficient. In this scenario, then the “go-to” option would be a multinomial logistic regression (mlogit) model.

Noting again that a logistic model is the proper treatment for a categorical outcome with 2 levels, a multinomial logistic model practically generalizes the method for arbitrarily many levels. There is also a [QMG memo](https://mdrc365.sharepoint.com/:b:/r/sites/QuantMethods/Shared%20Documents/Guidance/QMG%20Categorical%20Outcomes%20Memo%20(December%202013).pdf?csf=1&web=1&e=wqRkvO) For this example, we will be determining the effect of treatment on favAnimals while including cities, eduLevel and pre\_income\_raw as covariates, drawing from the mdrcAnalysis::sim\_data\_robust\_reg dataset.

# Loading dataset  
data(sim\_data\_robust\_reg, package = "mdrcAnalysis")  
# Making categoricals explicit and with desired reference levels for treatment  
sim\_data\_robust\_reg <- sim\_data\_robust\_reg %>%  
 mutate(  
 favAnimals = factor(favAnimals, levels = c("cat", "croc", "dog")),  
 treatment = factor(treatment, levels = c(0, 1)),  
 cities = factor(cities)  
 )  
sim\_data\_robust\_reg %>%  
 select(treatment, favAnimals, cities, eduLevel, pre\_income\_raw) %>%  
 head()

| treatment | favAnimals | cities | eduLevel | pre\_income\_raw |
| --- | --- | --- | --- | --- |
| 0 | dog | 1 | 2 | 7774.086 |
| 1 | cat | 1 | 5 | 29456.055 |
| 1 | cat | 1 | 3 | 21378.514 |
| 0 | croc | 1 | 2 | 7887.680 |
| 1 | cat | 1 | 3 | 31355.188 |
| 1 | dog | 1 | 1 | 11980.657 |

Here we will be using the nnet package to perform our analyses. While the package is focused on neural networks (a type of machine learning), it also includes a function, multinom() for multinomial logistic regression.

Running the model proceeds similarly to the analogous linear model; however, here I am scaling the quantitative pre\_income\_raw for performance and convergence reasons. This occurs because many if these algorithms rely on the distance between data points when determining effects. Unscaled variables run the risk of “dominating” the algorithm if one variable is of a much wider range than another. This is not done automatically by these algorithms because there are a variety of ways a user may want to scale a variable.

library(nnet)  
mlogit\_model <- multinom(  
 favAnimals ~ cities + eduLevel + scale(pre\_income\_raw) + treatment,  
 data = sim\_data\_robust\_reg  
)

## # weights: 27 (16 variable)  
## initial value 354.851769   
## iter 10 value 347.147846  
## iter 20 value 346.622167  
## final value 346.622122   
## converged

Here we see that calling multinom actually produces output despite our direct assignment to a variable. MNL is an iterative estimation process and this output exists to inform the user of its progress every 10 iterations. This output can be removed by setting trace = FALSE.

mlogit\_model <- multinom(  
 favAnimals ~ cities + eduLevel + scale(pre\_income\_raw) + treatment,  
 data = sim\_data\_robust\_reg,  
 trace = FALSE  
)

We can see simply cleaned results by once again using the broom package. The -value listed here is The result of a Wald z-test which is not actually calculated by nnet but rather by broom. One significant reason why logistic models are often avoided is because of the difficulty in interpreting the results in human terms aside from direction of effect.

# Using broom  
tidy(mlogit\_model) %>%  
 filter(term == "treatment1")

| y.level | term | estimate | std.error | statistic | p.value |
| --- | --- | --- | --- | --- | --- |
| croc | treatment1 | 0.1875968 | 0.2798265 | 0.670404 | 0.5026003 |
| dog | treatment1 | 0.4497142 | 0.2807859 | 1.601627 | 0.1092382 |

The broom also makes it trivial to exponentiate results which aids significantly in interpretability by providing direct odds ratios. Below, we can interpret the results as:

* Given every other covariate in the model, being in the treatment group increases the odds of being in the “croc” category vs the baseline “cat” category by 1.21.
* Given every other covariate in the model, being in the treatment group increases the odds of being in the “dog” category vs the baseline “cat” category by 1.57.
* The -value indicates the probability that, given all other coefficients are present, that the specified coefficient is equal to 0.

# Using broom  
tidy(mlogit\_model, exponentiate = TRUE) %>%  
 filter(term == "treatment1")

| y.level | term | estimate | std.error | statistic | p.value |
| --- | --- | --- | --- | --- | --- |
| croc | treatment1 | 1.206347 | 0.2798265 | 0.670404 | 0.5026003 |
| dog | treatment1 | 1.567864 | 0.2807859 | 1.601627 | 0.1092382 |

Showing equivalence to manually calculated Wald z-test.

# Calculating z-scores  
z\_mlogit\_model <- summary(mlogit\_model)$coefficients / summary(mlogit\_model)$standard.errors  
print(z\_mlogit\_model)

## (Intercept) cities2 cities3 cities4 cities5 eduLevel  
## croc 0.3840999 0.623674 1.5101232 1.759013 1.007643 -1.654061  
## dog 0.4303137 1.304249 0.3653124 1.879263 1.683399 -2.195643  
## scale(pre\_income\_raw) treatment1  
## croc 0.3394571 0.670404  
## dog -0.2351130 1.601627

# Calculating 2-tailed z-score test  
(1 - pnorm(abs(z\_mlogit\_model), 0, 1)) \* 2

## (Intercept) cities2 cities3 cities4 cities5 eduLevel  
## croc 0.7009044 0.5328417 0.1310120 0.07857525 0.31362605 0.09811518  
## dog 0.6669674 0.1921486 0.7148783 0.06020858 0.09229782 0.02811751  
## scale(pre\_income\_raw) treatment1  
## croc 0.7342654 0.5026003  
## dog 0.8141210 0.1092382

A single, -value can be derived using a Wald chi-square test. This can be easily found using the car package. This -value is what you would report on when describing the categorical variable effect as a whole.

suppressPackageStartupMessages(library(car))  
Anova(mlogit\_model)

| LR Chisq | Df | Pr(&gt;Chisq) |
| --- | --- | --- |
| 8.6880135 | 8 | 0.3692944 |
| 5.3485530 | 2 | 0.0689567 |
| 0.3130017 | 2 | 0.8551308 |
| 2.6148527 | 2 | 0.2705154 |

Cleaning up the result using broom, we get a -value of 0.271 with a null hypothesis that the coefficient is 0 in all logistic models.

tidy(Anova(mlogit\_model)) %>%  
 filter(term == "treatment")

| term | statistic | df | p.value |
| --- | --- | --- | --- |
| treatment | 2.614853 | 2 | 0.2705154 |

### -Computation

The [QMG memo](https://mdrc365.sharepoint.com/:b:/r/sites/QuantMethods/Shared%20Documents/Guidance/QMG%20Categorical%20Outcomes%20Memo%20(December%202013).pdf?csf=1&web=1&e=wqRkvO) actually recommends supplementing MNL with -computation to calculate proper p-values and estimates on impacts. The code for this in SAS was written by the RTU; however, to my knowledge no equivalent has been written in R as of yet. This method uses a bootstrap estimate of standard error when performing its Wald statistic calculation which can significantly increase the time complexity of the calculation as the model and data grows more complex. Because this method has not yet been formally written in R, I recommend consulting the RTU/QMG for assistance on setting this up.

### Seemingly Unrelated Regression (SUR)

Another method mentioned by the QMG is to use SUR to obtain more relevant p-values. This method does involve the use of a linear probability model and is not an extension of a logistic model which may be disagreeable to some funders.

Below I use the systemfit package to run the regression. Because this is more of a linear probability model extension, we will need to create a dummy variable representation of our categorical response variable.

sim\_data\_robust\_reg\_dummied <- sim\_data\_robust\_reg %>%  
 # Generating dummy variable version of categorical variable  
 fastDummies::dummy\_cols("favAnimals", remove\_selected\_columns = TRUE)  
# Showing the new columns made  
sim\_data\_robust\_reg\_dummied %>%  
 select(matches("favAnimals")) %>%  
 head()

| favAnimals\_cat | favAnimals\_croc | favAnimals\_dog |
| --- | --- | --- |
| 0 | 0 | 1 |
| 1 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 0 | 0 | 1 |

library(systemfit)

## Loading required package: Matrix

## Loading required package: lmtest

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

##   
## Please cite the 'systemfit' package as:  
## Arne Henningsen and Jeff D. Hamann (2007). systemfit: A Package for Estimating Systems of Simultaneous Equations in R. Journal of Statistical Software 23(4), 1-40. http://www.jstatsoft.org/v23/i04/.  
##   
## If you have questions, suggestions, or comments regarding the 'systemfit' package, please use a forum or 'tracker' at systemfit's R-Forge site:  
## https://r-forge.r-project.org/projects/systemfit/

# Omitting dog to avoid perfect multicollinearity  
sur\_fit <- systemfit(  
 list(  
 favAnimals\_cat ~ cities + eduLevel + scale(pre\_income\_raw) + treatment,  
 favAnimals\_croc ~ cities + eduLevel + scale(pre\_income\_raw) + treatment  
 ),  
 data = sim\_data\_robust\_reg\_dummied,  
 # Make sure to set method = "SUR"  
 method = "SUR"  
)  
tidy(sur\_fit) %>%  
 filter(str\_detect(term, "treatment"))

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| --- | --- | --- | --- | --- | --- | --- |
| eq1\_treatment1 | -0.068272363 | 0.05259214 | -1.2981476 | 0.1951863 | -0.1717486 | 0.03520391 |
| eq2\_treatment1 | -0.009277224 | 0.05305290 | -0.1748674 | 0.8612962 | -0.1136601 | 0.09510561 |

We then run an F-test using the car package to arrive at a singular p-value testing the null hypothesis that all treatment impacts are 0.

library(car)  
linearHypothesis(sur\_fit,  
 hypothesis.matrix = c("eq1\_treatment1 = 0", "eq2\_treatment1 = 0")  
)

| Res.Df | Df | F | Pr(&gt;F) |
| --- | --- | --- | --- |
| 632 | NA | NA | NA |
| 630 | 2 | 1.290065 | 0.2759789 |

Note that there has not been much investigation done yet to determine perfect equivalence between this R implementation and the SAS version using syslin and stest to calculate the f-test.

## Linear mixed-effects model without “lower-level” predictors

A mixed-effect model, which for our purposes are identical to hierarchical-level models (HLM) and mixed-level models (MLM) are used at MDRC to model data with a natural “nesting” structure, e.g children within classrooms within schools. While mathematically possible, we have not yet found an R package to easily work with multinomial logistic mixed-effect models. Instead, we use linear mixed effects model which necessitates a quantitative response variable instead of a categorical response variable. This method does use dummy variables but accounts for them in aggregate, thereby avoiding the previously listed issues with multiple dummy variable analyses.

For mixed effects models, we reference a omnibus test method that was proposed by Amy Taub and Marie-Andree Somers for the VIQI project. This method works around the quantitative/binary response limitation of a linear model by using the treatment variable as the response variable and using our outcome of interest as a covariate. By running this mixed-effects model in addition to one identically formulated without the outcome of interest as a covariate, we can pass both models into an ANOVA in order to determine if the inclusion of the outcome of interest improves the ability of the model to predict the treatment variable which serves as an effective proxy for our desired p-value.

Note that there exists a mixed extract in the mdrcAnalysis package but I am not using it here for increased consistency between the quantitative and categorical response analysis code. Here I am synthetically generating data with nesting structure. It is extremely important to note here that the numbers that show up as a result of these analyses are basically nonsensical. The data and model construction are used in the absence of publicly available, unproblematic data.

# Modifying sim\_data\_robust\_reg so that treatment is  
# block randomized and consistent across sites,  
# Each block has 2 sites  
sim\_data\_robust\_reg <- sim\_data\_robust\_reg %>%  
 mutate(Block = cut(as.integer(factor(sites)),  
 n\_distinct(sites) / 2,  
 labels = seq\_len(n\_distinct(sites) / 2)  
 )) %>%  
 group\_by(Block) %>%  
 mutate(treatment = as.integer(sites == head(sites, 1)))  
sim\_data\_robust\_reg\_dummied <- sim\_data\_robust\_reg %>%  
 # Generating dummy variable version of categorical variable  
 fastDummies::dummy\_cols("favAnimals", remove\_selected\_columns = TRUE)

If we want to calculate the treatment effect on pre\_income\_raw, a quantitative variable, while accounting for sites as a random intercept effect we can simply specify the model as such using the lme4 package. I am also calling the emmeans for adjusted means calculations and broom.mixed for its method extensions on broom.

suppressPackageStartupMessages(library(lme4))  
suppressPackageStartupMessages(library(emmeans))  
suppressPackageStartupMessages(library(broom.mixed))  
# Specifying the model  
lmer(pre\_income\_raw ~ treatment + (1 | sites), data = sim\_data\_robust\_reg) %>%  
 # Getting adjusted means by treatment but using satterthwaite degres of freedom  
 emmeans("treatment", options = get\_emm\_option("satterthwaite")) %>%  
 # Showing treatment contrasts  
 pairs()

## contrast estimate SE df t.ratio p.value  
## treatment0 - treatment1 477 1397 18 0.342 0.7366  
##   
## Degrees-of-freedom method: kenward-roger

Suppose then we wanted to identify the relationship between favAnimal (which is at the individual level) and the treatment while accounting for the site nesting. The following method will work provided we do not want to include block because it is a site level randomized variable. The next section, titled “Linear mixed-effects model with other predictors”, will discuss how to conduct this analysis with additional variables.

Calculating this for the categorical response actually separates the calculation of the p-value and the contrast estimate. First to calculate the p-value, we need to ensure that treatment is a true dummy because we will be using it as a quantitative response variable.

# Showing this is a 0, 1 integer vector  
str(unique(sim\_data\_robust\_reg$treatment))

## int [1:2] 1 0

Now we calculate two models, a null and effect model. Both models use our new treatment variable as a response variable and account for classroom nesting but only the effect model contains our categorical variable as a predictor.

# Note that in actual work, these warnings indicate something deeply  
# wrong with your model, i.e. a lack of true nesting effects  
# However, I am ignoring them here for demonstration purposes  
mixed\_null\_model <- lmer(treatment ~ (1 | sites), data = sim\_data\_robust\_reg)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :  
## Model failed to converge with max|grad| = 0.648388 (tol = 0.002, component 1)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unidentifiable: very large eigenvalue  
## - Rescale variables?

mixed\_result\_model <- lmer(treatment ~ favAnimals + (1 | sites), data = sim\_data\_robust\_reg)

## Warning in optwrap(optimizer, devfun, getStart(start, rho$pp), lower =  
## rho$lower, : convergence code -4 from nloptwrap: NLOPT\_ROUNDOFF\_LIMITED:  
## Roundoff errors led to a breakdown of the optimization algorithm. In this case,  
## the returned minimum may still be useful. (e.g. this error occurs in NEWUOA if  
## one tries to achieve a tolerance too close to machine precision.)

Now we pass both of these models to an anova test which compares the model. Because these models differ only by the inclusion of favAnimals as a predictor, our anova result will tell us if this, more complex model is significantly better than the model without favAnimals. This basically tells us if there is a statistically significant difference between the treatment and control accounting for all of favAnimals in aggregate.

# anova comparison of models  
mixed\_anova <- anova(mixed\_result\_model, mixed\_null\_model)

## refitting model(s) with ML (instead of REML)

# tidycleans up the output of anova into a simple table  
tidy(mixed\_anova)

| term | npar | AIC | BIC | logLik | deviance | statistic | df | p.value |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| mixed\_null\_model | 3 | -9738.286 | -9726.953 | 4872.143 | -9744.286 | NA | NA | NA |
| mixed\_result\_model | 5 | -9731.135 | -9712.246 | 4870.567 | -9741.135 | 0 | 2 | 1 |

The actual p-value can be extracted as such:

na.omit(mixed\_anova$`Pr(>Chisq)`)

## [1] 1  
## attr(,"na.action")  
## [1] 1  
## attr(,"class")  
## [1] "omit"

Contrasts are calculated individually using the dummy variables as outcomes; however, we will be ignoring the p-values because we already have the aggregated p-value we calculated with our ANOVA. Using these p-values will result in the same issues discussed in section “Why not to use dummy variables”. This is done functionally but the essence of the code is similar to the mixed effects quantitative response example shown previously.

mixed\_contrasts <- paste0("favAnimals\_", c("cat", "croc", "dog")) %>%  
 purrr::map(function(.response\_variable) {  
 # Generating formulas for each response variable and attaching data  
 dummy\_formula <- as.formula(paste(.response\_variable, "~ treatment + (1 | sites)"),  
 env = rlang::env(!!!sim\_data\_robust\_reg\_dummied)  
 )  
 # Run model given the generated formula  
 lmer(dummy\_formula) %>%  
 # Adjusted means by treatment  
 emmeans("treatment", options = emmeans::get\_emm\_option("satterthwaite")) %>%  
 # Adjusted mean contrasts  
 pairs() %>%  
 # Convert adjusted mean contrasts into data.frame  
 tidy() %>%  
 mutate(  
 Categorical\_Level = .response\_variable,  
 Contrast = estimate,  
 .keep = "none"  
 )  
 }) %>%  
 bind\_rows()

## boundary (singular) fit: see help('isSingular')  
## boundary (singular) fit: see help('isSingular')

We can now combine the p-value and contrasts into a single table.

mixed\_contrasts %>%  
 mutate(Categorical\_Level, Contrast,  
 P\_Value = na.omit(mixed\_anova$`Pr(>Chisq)`),  
 .keep = "none"  
 )

| Categorical\_Level | Contrast | P\_Value |
| --- | --- | --- |
| favAnimals\_cat | -0.03454622 | 1 |
| favAnimals\_croc | 0.07209695 | 1 |
| favAnimals\_dog | -0.03290382 | 1 |

## Linear mixed-effects model with higher-level predictors

We are interested in determining treatment impacts accounting for site level effects. Because of this, variables at the site level or randomized at the site level can introduce problems because of insufficient individual level variation. This lack of variation can be addressed by aggregating our dataset and running the model using the new aggregated level as the individual.

First to calculate the p-value, we create a new dataset by aggregating by our nesting variable, in this case sites.

sim\_data\_robust\_reg\_aggregated <- sim\_data\_robust\_reg\_dummied %>%  
 group\_by(sites, treatment, Block) %>%  
 summarise(across(matches("^favAnimals"), mean),  
 .groups = "drop"  
 )

Now we calculate two linear models, a null and effect model. A linear model is used instead of a mixed effects model because we aggregated data up to the sites level. Both models use our dummy treatment variable as a response variable and account for classroom nesting and block but only the effect model contains our aggregated dummies as a predictor.

# Generating null model  
mixed\_null\_model <- stats::lm(treatment ~ Block,  
 data = sim\_data\_robust\_reg\_aggregated  
)  
# Generating result formula based on the names in the aggregated ddataset  
mixed\_result\_formula <- as.formula(  
 paste(  
 "treatment ~",  
 paste0(  
 setdiff(  
 names(sim\_data\_robust\_reg\_aggregated),  
 c("treatment", "sites")  
 ),  
 collapse = " + "  
 )  
 ),  
 env = rlang::env(!!!sim\_data\_robust\_reg\_aggregated)  
)  
print(mixed\_result\_formula)

## treatment ~ Block + favAnimals\_cat + favAnimals\_croc + favAnimals\_dog  
## <environment: 0x562af58c6b30>

mixed\_result\_model <- lm(mixed\_result\_formula)

We now run an ANOVA test on the two models to get a test that accounts for all components of our categorical variable.

mixed\_anova <- stats::anova(mixed\_result\_model, mixed\_null\_model)  
tidy(mixed\_anova)

| term | df.residual | rss | df | sumsq | statistic | p.value |
| --- | --- | --- | --- | --- | --- | --- |
| treatment ~ Block + favAnimals\_cat + favAnimals\_croc + favAnimals\_dog | 11 | 6.2304 | NA | NA | NA | NA |
| treatment ~ Block | 13 | 6.5000 | -2 | -0.2696003 | 0.2379946 | 0.7921621 |

The actual p-value can be extracted as such:

na.omit(mixed\_anova$`Pr(>F)`)

## [1] 0.7921621  
## attr(,"na.action")  
## [1] 1  
## attr(,"class")  
## [1] "omit"

The actual contrasts are calculated using a multi-level model for each dummied predictor.

mixed\_contrasts <- paste0("favAnimals\_", c("cat", "croc", "dog")) %>%  
 purrr::map(function(.response\_variable) {  
 dummy\_formula <- as.formula(  
 paste(.response\_variable, "~ Block + treatment + (1|sites)"),  
 env = rlang::env(!!!sim\_data\_robust\_reg\_dummied)  
 )  
 lmer(dummy\_formula) %>%  
 emmeans("treatment",  
 options = get\_emm\_option("satterthwaite")  
 ) %>%  
 pairs() %>%  
 tidy() %>%  
 mutate(  
 Categorical\_Level = .response\_variable,  
 Contrast = estimate,  
 False\_P\_Value = p.value,  
 .keep = "none"  
 )  
 }) %>%  
 bind\_rows()

## boundary (singular) fit: see help('isSingular')  
## boundary (singular) fit: see help('isSingular')  
## boundary (singular) fit: see help('isSingular')

mixed\_contrasts

| Categorical\_Level | Contrast | False\_P\_Value |
| --- | --- | --- |
| favAnimals\_cat | -0.06184277 | 0.3057566 |
| favAnimals\_croc | 0.10250428 | 0.1072057 |
| favAnimals\_dog | -0.04066151 | 0.5020427 |

Just for demonstration purposes, you can see here the range of p-values reported by running these individual tests on the dummy variables.

We can now combine the p-value and effect sizes into a single table.

mixed\_contrasts %>%  
 mutate(Categorical\_Level, Contrast,  
 P\_Value = na.omit(mixed\_anova$`Pr(>F)`),  
 .keep = "none"  
 )

| Categorical\_Level | Contrast | P\_Value |
| --- | --- | --- |
| favAnimals\_cat | -0.06184277 | 0.7921621 |
| favAnimals\_croc | 0.10250428 | 0.7921621 |
| favAnimals\_dog | -0.04066151 | 0.7921621 |

### Alternative methods for mixed-effect models

Note that by setting family = binomial, there is already a treatment for binary outcome variables using lme4::glmer. One method I saw in the [lme4 github](https://github.com/lme4/lme4/issues/594) would be to convert a multinomial model into the equivalent Poisson regression (Lee et al., 2017). lme4::glmer supports every GLM family listed in stats::glm which does includes Poisson regression which means we can use this equivalent Poisson formulation in lme4::glmer. However, this is easier said than done and there are also concerns about the computational complexity of such a technique.

A Bayesian mixed effects model should also theoretically handle this situation using the brms package. This option has seen some traction online but it may be difficult to practically implement because it, like many software libraries designed around probabilistic programming and bayesian modelling, uses the “Stan” programming language in its backend which basically required another compilation step on program run.

# References

*Advanced r, second edition*. (2019). Chapman; Hall/CRC. <https://doi.org/10.1201/9781351201315>

Lee, J. Y. L., Green, P. J., & Ryan, L. M. (2017). *On the "poisson trick" and its extensions for fitting multinomial regression models*. <http://arxiv.org/abs/1707.08538>

Sinharay, S. (2010). An overview of statistics in education. In P. Peterson, E. Baker, & B. McGaw (Eds.), *International encyclopedia of education (third edition)* (Third Edition, pp. 1–11). Elsevier. <https://doi.org/https://doi.org/10.1016/B978-0-08-044894-7.01719-X>  
There have been numerous applications of statistical methods to the field of education, mostly in educational measurement. This article provides short descriptions of several topics in statistics that have found applications to education and provides examples of applications of some of these topics to education.

Stefan, A. M., & Schönbrodt, F. D. (2023). Big little lies: A compendium and simulation of p-hacking strategies. *Royal Society Open Science*, *10*(2). <https://doi.org/10.1098/rsos.220346>  
In many research fields, the widespread use of questionable research practices has jeopardized the credibility of scientific results. One of the most prominent questionable research practices is p-hacking. Typically, p-hacking is defined as a compound of strategies targeted at rendering non-significant hypothesis testing results significant. However, a comprehensive overview of these p-hacking strategies is missing, and current meta-scientific research often ignores the heterogeneity of strategies. Here, we compile a list of 12 p-hacking strategies based on an extensive literature review, identify factors that control their level of severity, and demonstrate their impact on false-positive rates using simulation studies. We also use our simulation results to evaluate several approaches that have been proposed to mitigate the influence of questionable research practices. Our results show that investigating p-hacking at the level of strategies can provide a better understanding of the process of p-hacking, as well as a broader basis for developing effective countermeasures. By making our analyses available through a Shiny app and R package, we facilitate future meta-scientific research aimed at investigating the ramifications of p-hacking across multiple strategies, and we hope to start a broader discussion about different manifestations of p-hacking in practice.