**Nominal Categorical Response in Regression-Based Analysis**

Benjamin Bui

October 10, 2024

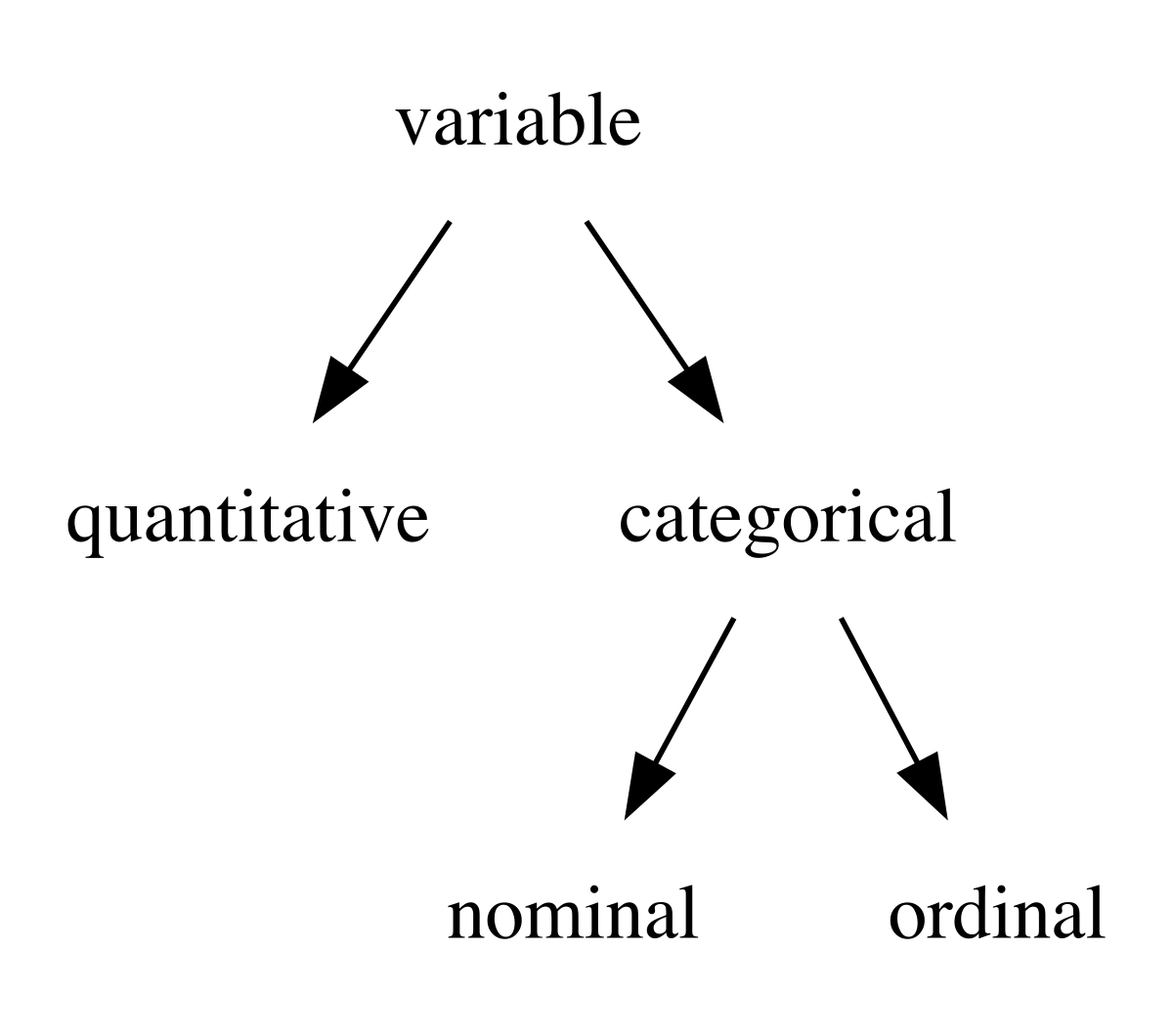
Categorical variables provide an inherent problem when attempting to describe relationships mathematically. Many analyses we run involve regression analyses which inherently require that **ALL** variables be treated numerically. Categorical variables are normally converted into numerical variables in one of two ways, 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 statistically problematic and offers alternative solutions to the issue.

This memo is divided into three main sections. The first section describes categorical variables, both statistically and how they are handled in R. This is followed by a statistical 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. Finally, we go through solutions for how to handle this statistical problem in R.

# Categorical variables, in general and in R

## What is a categorical and dummy variable?

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



A variable is quantiative if it is inherently ordered, and has an internally consistent, scalable definition of difference. For example, consider a variable representing age. Given any two values of age we can state that one age value is greater than or the same as the other age value. Moreover, this relationship is transitive as can be seen with the following.

12 years of age is greater than 8 years of age. 8 years of age is greater than 4 years of age. Thus 12 years of age is necessarily also greater than 4 years of age.

The difference property can be seen with how we can describe age with arbitrary precision. We can understand what is meant when we say something is 5 years of age but we can also understand what is meant when we say something is 4.999999999 years of age. Moreover the distance between 4.999999999 and 5 years of age is the same as 40.999999999 and 50 years of age. Although these properties seem trivial, indeed they are just a subset of the properties of the real numbers, categorical variables breaks with some of these assumptions.

First we will break with the scalability property. Suppose we have a variable with the possible values: {“Less than high school”, “High school”, “Bachelor’s Degree”, “Greater than Bachelor’s Degree”}. We can order this such that the ordering represents the average amount of time for completion: {“Less than high school” < “High school” < “Bachelor’s Degree” < “Greater than Bachelor’s Degree”}. The variable as it is represented does’t allow for any further precision because their are no in-between values. This type of variable is known as an ordered categorical variable, or an ordinal variable.

Now we will break with the ordering property. Here we’ll use a fruit variable with the possible values of {“apple”, “orange”, “grape”}. There is no agreed upon way to order these values so we will refer to this as a nominal categorical variable, or a nominal variable for short.

It is important to consider that many variables can be considered quantitiative, ordinal, or nominal depending on your interpretation and goals. Age was what we used as an exmaple of a quantitiative variable. We can effectively transform this into an ordinal variable by breaking up the age into the possible values: {“Younger than 3”, “3 to 18 years”, “Older than 18”}. We can also transform this into a nominal variable by making it into the variable {“3 to 18 years”, “Not 3 to 18 years”}. All three of these variables describe age despite the differences in their representation.

The nominal variable we made also qualifies as a dummy variable, or a binary variable. Binary and dummy variables are, for most intents and purposes, the same thing just from the perspective of a programmer and a statistician, respectively. The only difference between them is that dummy variables are technically always coded as a {0, 1} variable and that they are used to indicate the presence of some categorical trait. Dummy variables break up a single categorical variable into multiple different variables, each representing a specific possible value of the original categorical. Dummy variables are coded with a value space of {0, 1} because it allows for convenient mathematical manipulation and interpretation. It is possible to create dummy variables from a categorical variable with possible levels; however, dummy variables encodes the same amount of information. For example, consider our fruit categorical variable {“apple”, “orange”, “grape”}. We can separate this into an apple dummy variable that is 1 if the fruit is “apple” and 0 otherwise and an orange dummy variable that is 1 if the fruit is “orange” and 0 otherwise. If both apple and orange is 0, then we know that the fruit must be “grape” which makes creating an explicit grape variable redundant and can actually introduce issues of perfect multicollinearity.

## Variable types as R classes

These three types of variables: quantitative, nominal, and ordered are actually represented in R as different classes of vector. Quantitative variables should be represented by the numeric class, nominal variables by the factor class, and ordinal variable by a combination of the ordered and factor classes.

$ has 25 different types of vectors at its lowest level, most relevant here are integer, double, and character (**WICHHAM201912?**). You may notice that factor is not included here, that is because factor isn’t actually implemented as a low level vector type, rather it is implemented via a integer vector and character vector. A factor stores its data as an integer vector but also includes an attribute called levels This levels attribute is a character vector representing the sample space of your categorical variable. 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 very 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.

Let’s take at this using iris$Species.

# Taking a subset for readability  
iris\_subset <- iris[sample(seq\_len(nrow(iris)), 20), ]  
iris\_subset$Species

## [1] versicolor setosa virginica versicolor virginica setosa   
## [7] versicolor setosa versicolor setosa virginica virginica   
## [13] virginica versicolor virginica versicolor setosa versicolor  
## [19] virginica versicolor  
## Levels: setosa versicolor virginica

The actual values of Species are stored as the integer vector:

typeof(iris\_subset$Species)

## [1] "integer"

unclass(iris\_subset$Species)

## [1] 2 1 3 2 3 1 2 1 2 1 3 3 3 2 3 2 1 2 3 2  
## attr(,"levels")  
## [1] "setosa" "versicolor" "virginica"

And the levels are stored as the character vector:

levels(iris$Species)

## [1] "setosa" "versicolor" "virginica"

This mapping means that all values of 1 correspond to “setosa”, 2 corresponds to “versicolor”, and 3 corresponds to “virginica”. Convertng this into an ordered factor is fairly simple (although I recommend being more explicit with your ordering).

as.ordered(iris\_subset$Species)

## [1] versicolor setosa virginica versicolor virginica setosa   
## [7] versicolor setosa versicolor setosa virginica virginica   
## [13] virginica versicolor virginica versicolor setosa versicolor  
## [19] virginica versicolor  
## Levels: setosa < versicolor < virginica

Here, we see the only difference in presentation is that the levels now use < instead of , as the delimiter which represents which levels are greater than which. R the language doesn’t treat these variables differently, again the only difference is the addition of the ordered class.

attributes(iris\_subset$Species)

## $levels  
## [1] "setosa" "versicolor" "virginica"   
##   
## $class  
## [1] "factor"

attributes(as.ordered(iris\_subset$Species))

## $levels  
## [1] "setosa" "versicolor" "virginica"   
##   
## $class  
## [1] "ordered" "factor"

This additional class is used by modelling packages to differentiate between nominal and ordered categorical variables and to apply different methods to these variables. Consider the most common method for fitting linear models: stats::lm. When given a plain factor as a covariate, stats::lm creates dummy variables and runs the model treating these dummy variable as if they were quantitative. However, if you pass an ordered factor as a covariate, then stats::lm will treat the variable as a polynomial model with degrees. A polynomial model assumes that the levels of the ordered factor are equally spaced and that the space is continuous. The results it presents evaluate how well the relationship between the ordered factor and response variable can be described as linear, quadratic, cubic, etc. This is a choice by the authors of the stats package on how to handle ordered factor covariates and the exact way that any modelling package handles these variables can differ. Treating the categorica variable as a polynomial model isn’t better or worse than treating it as dummies, it is just important that you select the appropraite method for your data and intention.

It can be confusing to think of a plain factor as unordered because the order that the levels attribute is defined as can still have a practical meaning even if it doesn’t have a statistical or mathematical one. For example, suppose we want to sort a dataset or plot such that certain values of a categorical variable appear first. We aren’t implying that any level is greater or less than the other, we just want our data to appear a certain way. A simple way to do this would be to adjust the levels such that the values we want to see first are first in the actual levels attribute character vector. The first value in the levels attribute is also used when calculating contrasts as the “baseline”. For example, consider our fruit variable: {“apple”, “orange”, “grape”}. If we used this variable in a model and requested contrasts on fruit, then our results would, by default, present the impact of “orange” compared to “apple” and the impact of “grape” compared to “apple”. This doesn’t have any impact in the computation of the model itself but it affect the direct interpretability of the coefficients. However, because we should’t make a variable an ordered factor unless we explicitly want it to be treated as an ordinal, we can instead just change the level attribute. A simple way of doing this would be using stats::relevel or forcats::fct\_relevel

iris\_subset$Species

## [1] versicolor setosa virginica versicolor virginica setosa   
## [7] versicolor setosa versicolor setosa virginica virginica   
## [13] virginica versicolor virginica versicolor setosa versicolor  
## [19] virginica versicolor  
## Levels: setosa versicolor virginica

# relevel is great to just set 1 new level at the start  
relevel(iris\_subset$Species, "virginica")

## [1] versicolor setosa virginica versicolor virginica setosa   
## [7] versicolor setosa versicolor setosa virginica virginica   
## [13] virginica versicolor virginica versicolor setosa versicolor  
## [19] virginica versicolor  
## Levels: virginica setosa versicolor

# fct\_relevel is useful for more complex re-ordering operations  
forcats::fct\_relevel(iris\_subset$Species, "virginica", "versicolor", "setosa")

## [1] versicolor setosa virginica versicolor virginica setosa   
## [7] versicolor setosa versicolor setosa virginica virginica   
## [13] virginica versicolor virginica versicolor setosa versicolor  
## [19] virginica versicolor  
## Levels: virginica versicolor setosa

# The potential problems with dummy response variables

This section will cover why we do not want always want to substitute nominal cateogrical variables with dummy variables.

Tranforming nominal categorical variables into dummy variables is accepted practice when being done on predictor variables for a mode. As stated previously, this is how most statistical modelling packages in R automatically handle factor variables. It intuitively makes sense then that we apply the same to nominal categorical response variables; to just run models for each dummy variable. However, consider the interpretability of our results if only 1 of these analyses provides a statistically significant p-value. Does that mean our explanatory variable only effectively predicts for one possible value of our categoricals? That conclusion is difficult to reconcile with our knowledge that our dummy variables are collinear and mutually exclusive with each other. For this eason, a single, shared statistical test would be preferred when evaluating a categorical variable. This would allow us to better evaluate the impact of the explanatory variable on the categorical response as a whole.

Another important consideration pertains to multiple hypothesis testing. Given some impact analysis, we have a null hypothesis and an alternate hypothesis . Although the specifics will differ, these two hypotheses can be generally summarised as:

The true population treatment and control groups are the same.

The true population treatment and control groups are not the same.

The p-value result you get from running a statistical test tells you the likelehood that, given the assumptions particular to your choice of test, that you would have achieved the result found given . A p-value of 0.05 thus means that there is a 5% chance that, given the treatment haa no effect, you would have oberved the differences found. Although 5% is a fairly small number, it is not 0 and it becomes practically inevitable that we encounter a erroneously find a statistically significant result the more tests we run. This issue is commonly referred to as the multiple comparisons problem and the use of dummied outcome variables can inflate the number of outcomes such that this can become a much more significant issue.

## Multiple comparisons with dummy variables

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. This dataset will consist 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

library(tidyverse, warn.conflicts = FALSE, quietly = TRUE)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.1 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ tidyr::extract() masks magrittr::extract()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ purrr::set\_names() masks magrittr::set\_names()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

# 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, all of these variables were randomly and independently generated so there is no true relationship between any of them.

| p&lt;X | n |
| --- | --- |
| 0.05 | 3 |
| 0.1 | 1 |
| >=0.1 | 42 |

We see no statistical significance in 42 of our 46 tests using a threshold of 0.1; however, there are four tests that display some significance, or about 8.7% of the results. Recall that we are really trying to determine the relationship between our treatment variable and five response variables, not 46. Although presenting 8.7% of these results as statistically significant isn’t too problematic in and of itself, it would be disingenous to present the statistically significant results without either the context of the other, non-significant results, or with a more formalized multiple comparisons test.

# Avoiding dummy response variables

We ideally want a test that provides a singular result on a our categorical response variable. Although we did discuss ordinal variables previously, that largely to avoid the accidental conflation of nominal and ordinal variables. The solutions presented hencforth should only be applied to nominal response variables.

The simplest solution to the issue is to use a quantitative variable instead. In cases where a categorical variable is generated from a quantitative variable, say when creating age blocks when exact birth dates are known, using the quantitative variable as the response instead can avoid much of this headache. Moreover, the choice of category boundaries itself, intentionally or not, can geerate false significance (Stefan & Schönbrodt, 2023). There are many cases where this option is not available, whether because the categorical variable is not derived from a quantitative variable or because the funder/PI wants results in terms of the categorical variable. If this is the case then continue on to one of the method below.

## Binary response variables

Given a binary response variable, the most commonly prescribed model is the logistic regression (logit). In practical terms, the difference between a linear model provides a “close enough” result to the logit model with more simply interpretable results. Please see this [QMG memo](https://mdrc365.sharepoint.com/sites/QuantMethods/Shared%20Documents/Forms/AllItems.aspx?id=%2Fsites%2FQuantMethods%2FShared%20Documents%2FGuidance%2FQMG%2DMemo%2Don%2DBinary%2DOutcomes%20%28January%202024%29%2Epdf&parent=%2Fsites%2FQuantMethods%2FShared%20Documents%2FGuidance) on the topic for more details.

References

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*. <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.