Spanish Translation - A/B Testing

Mitul Shah

03/07/2020

Goal

A/B tests play a huge role in website optimization. Analyzing A/B tests data is a very important data scientist responsibility. Especially, data scientists have to make sure that results are reliable, trustworthy and conclusions can be drawn.

Furthermore, companies often run tens, if not hundreds, of A/B tests at the same time. Manually analyzing all of them would require lot of time and people. Therefore, it is common practice to look at the typical A/B test analysis steps and try to automate as much as possible. This frees up time for the data scientist to work on more high level topics.

Challenge Description

Company XYZ is a worldwide e-commerce site with localized versions of the site.

A data scientist at XYZ noticed that Spain-based users have a much higher conversion rate than any other Spanish-speaking country.

Spain and LatAm country manager suggested that one reason could be translation. All Spanish-speaking countries had the same translation of the site which was written by a Spaniard. Therefore, they agreed to try a test where each country would have its own translation written by a local. That is, Argentinian users would see a translation written by an Argentinian, Mexican users written by a Mexican and so on. Obviously, nothing would change for users from Spain.

After they run the test however, they are really surprised because the test is negative. That is, it appears that the non-highly localized translation was doing better!

You are asked to:

1. Confirm that test is actually negative. I.e., the old version of the site with just one translation across Spain and LatAm performs better
2. Explain why that might be happening. Are the localized translations really worse?
3. If you identified what was wrong, design an algorithm that would return FALSE if the same problem is happening in the future and TRUE if everything is good and results can be trusted.

Solution

A crucial assumption behind an A/B test is that the only difference between test and control has to be the feature we are testing. This implies that test and control user distributions are comparable. If this is true, we can then exactly estimate the impact of the feature change on whichever metric we are testing.

Comparable test and control user distribution means that, for each relevant segment, the relative proportion of users in test and control is similar. That is, if US users are 10% of users in the test group, we expect to also have ~10% of US users in control. If we have 50% of repeat users in test, we should have a similar percentage in control, and so on.

From a purely statistical standpoint, the above should be true over a large enough number of users. And in A/B testing, we are looking for very small gains, so sample size is large, and, therefore, test and control distributions should be the same.

In practice, it is pretty frequent that test and control distributions are different, invalidating the test results. The number one reason for that is bugs or bias in the randomization algorithm that assigns users to test and control, leading to over/under representation of certain segments. That is, we might have more US users in control, but those users have higher conversion rate, so the difference we see in the metric is not only affected by the feature change that we are testing.

It is therefore extremely important to check that test and control distributions are similar before doing the statistical test. Let's see how.

Load the data

```
## Libraries needed
require(dplyr)

## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

require(rpart)

## Loading required package: rpart

require(ggplot2)

## Loading required package: ggplot2
```

```r
## Read the test data
test = read.csv("/Users/mitulshah/Downloads/Downloads/Giulio Palombo Book
Solutions/Spanish Translation - A:B Testing/Translation_Test/test_table.csv")

## Look at the first few rows of the test data
knitr::kable(head(test))
```

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

user_id	date	source	device	browser_language	ads_channel	browser	conversion	test
315281	2015-12-03	Direct	Web	ES	NA	IE	1	0
497851	2015-12-04	Ads	Web	ES	Google	IE	0	1
848402	2015-12-04	Ads	Web	ES	Facebook	Chrome	0	0
290051	2015-12-03	Ads	Mobile	Other	Facebook	Android_App	0	1
548435	2015-11-30	Ads	Web	ES	Google	FireFox	0	1
540675	2015-12-03	Direct	Mobile	ES	NA	Android_App	0	1

Columns:

user_id: the id of the user. Unique by user. Can be joined to user id in the other table. For each user, we just check whether conversion happens the first time they land on the site since the test started.

date : when they came to the site for the first time since the test started

source : marketing channel: Ads, SEO, Direct. Direct means everything except for ads and SEO, such as directly typing site URL on the browser, downloading the app w/o coming from SEO or Ads, referral friend, etc.

device : device used by the user. It can be mobile or web

browser_language : in browser or app settings, the language chosen by the user. It can be EN, ES, Other (Other means any language except for English and Spanish)

ads_channel : if marketing channel is ads, this is the site where the ad was displayed. It can be: Google, Facebook, Bing, Yahoo ,Other. If the user didn’t come via an ad, this field is NA

browser : user browser. It can be: IE, Chrome, Android_App, FireFox, Iphone_App, Safari, Opera

conversion : whether the user converted (1) or not (0). This is our label. A test is considered successful if it increases the proportion of users who convert.

test : users are randomly split into test (1) and control (0). Test users see the new translation and control the old one. For Spain-based users, this is obviously always 0 since there is no change there.

```
## Read the user data
user = read.csv("/Users/mitulshah/Downloads/Downloads/Giulio Palombo Book
Solutions/Spanish Translation - A:B Testing/Translation_Test/user_table.csv")

## Look at the first few rows of the test data
knitr::kable(head(user))

## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

user_id	sex	age	country
765821	M	20	Mexico
343561	F	27	Nicaragua
118744	M	23	Colombia
987753	F	27	Venezuela
554597	F	20	Spain
62371	M	29	Ecuador

Columns:

user_id : the id of the user. It can be joined to user id in the other table sex : user sex: Male or Female

age : user age (self-reported)

country : user country based on ip address

Checking Data Quality

```
## Dimension of the user table
dim(user)

## [1] 452867      4

## Dimension of the test table
dim(test)

## [1] 453321      9
```

Let’s check that data makes sense.

```
## Are there duplicates in either dataset?
length(unique(test$user_id)) == length(test$user_id)

## [1] TRUE

length(unique(user$user_id)) == length(user$user_id)

## [1] TRUE

## Everyone in one table also in the other one?
length(user$user_id) - length(test$user_id)

## [1] -454
```

Looks like the user table is busted and we have some user ids missing. It is so few values that we can safely get rid of those with an inner join. Nothing will change. However, separately, we should still try to find out why that happened. There might be some bug somewhere.

```
## Join
data = merge(test, user, by = "user_id")

## Convert the mode of date to date
data$date = as.Date(data$date)

## Look at the first few rows of the data
knitr::kable(head(data))

## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

user_id	date	source	device	browser_language	ads_channel	browser	conversion	test	sex	age	country
1	2015-12-02	SEO	Web	EN	NA	Chrome	0	0	M	38	Chile
2	2015-11-30	SEO	Mobile	ES	NA	Android_App	0	0	M	27	Colombia
3	2015-12-03	SEO	Mobile	ES	NA	Iphone_App	0	1	M	18	Guatemala
5	2015-11-30	Ads	Web	ES	Facebook	Chrome	0	1	M	22	Argentina
8	2015-12-03	Ads	Mobile	ES	Google	Android_App	0	1	M	19	Venezuela
11	2015-12-03	Ads	Web	ES	Yahoo	Chrome	0	1	F	28	Colombia

```
## Look at the summary of the data
summary(data)
```

```
##     user_id            date                source              device         
##  Min.   :      1   Min.   :2015-11-30   Length:452867      Length:452867     
##  1st Qu.: 249819   1st Qu.:2015-12-01   Class :character   Class 
:character  
##  Median : 500019   Median :2015-12-03   Mode  :character   Mode  
:character  
##  Mean   : 499945   Mean   :2015-12-02                                        
##  3rd Qu.: 749543   3rd Qu.:2015-12-04                                        
##  Max.   :1000000   Max.   :2015-12-04                                        
##  browser_language   ads_channel          browser            conversion     
##  Length:452867      Length:452867      Length:452867      Min.   :0.00000  
##  Class :character   Class :character   Class :character   1st Qu.:0.00000  
##  Mode  :character   Mode  :character   Mode  :character   Median :0.00000  
##                                                            Mean   :0.04956  
##                                                            3rd Qu.:0.00000  
##                                                            Max.   :1.00000  
##       test            sex                  age          country         
##  Min.   :0.0000   Length:452867      Min.   :18.00   Length:452867     
##  1st Qu.:0.0000   Class :character   1st Qu.:22.00   Class :character  
##  Median :0.0000   Mode  :character   Median :26.00   Mode  :character  
##  Mean   :0.4765                      Mean   :27.13                     
##  3rd Qu.:1.0000                      3rd Qu.:31.00                     
##  Max.   :1.0000                      Max.   :70.00                     
```

1. Confirm that test is actually negative. I.e., the old version of the site with just one translation across Spain and LatAm performs better

First question is about checking the test results. But even before that, let's make sure it is true Spain converts much better than LatAm countries.

```
## Conversion by country
data_conversion_country = data %>%
                          group_by(country) %>%
                          # we check the old version
                          summarize( conversion = mean(conversion[test ==
0]))%>%
                          arrange (desc(conversion))

## Print conversion by country
knitr::kable(data_conversion_country)

## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

country	conversion
Spain	0.0797188
El Salvador	0.0535540
Nicaragua	0.0526470
Costa Rica	0.0522556
Colombia	0.0520895
Honduras	0.0509058
Guatemala	0.0506429
Venezuela	0.0503437
Peru	0.0499140
Mexico	0.0494946

country	conversion
Bolivia	0.0493694
Ecuador	0.0491538
Paraguay	0.0484932
Chile	0.0481072
Panama	0.0467955
Argentina	0.0150705
Uruguay	0.0120482

Yes. Definitely true.

```
## A simple t-test here should work. We have collected ~0.5MM data and
test/control split is ~50/50.

## Nothing changed in Spain, so no point in keeping those users
data = subset(data, country != "Spain")
t.test(data$conversion[data$test == 1], data$conversion[data$test == 0])

## 
##  Welch Two Sample t-test
## 
## data:  data$conversion[data$test == 1] and data$conversion[data$test == 0]
## t = -7.3539, df = 385258, p-value = 1.929e-13
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.006181421 -0.003579837
## sample estimates:
##  mean of x  mean of y 
## 0.04341116 0.04829179
```

Control users are converting at 4.8% while users in the test just at 4.3%. That's a 10% drop, which would be dramatic if it were true. The most likely reason for weird A/B test results are:

1) We didn't collect enough data

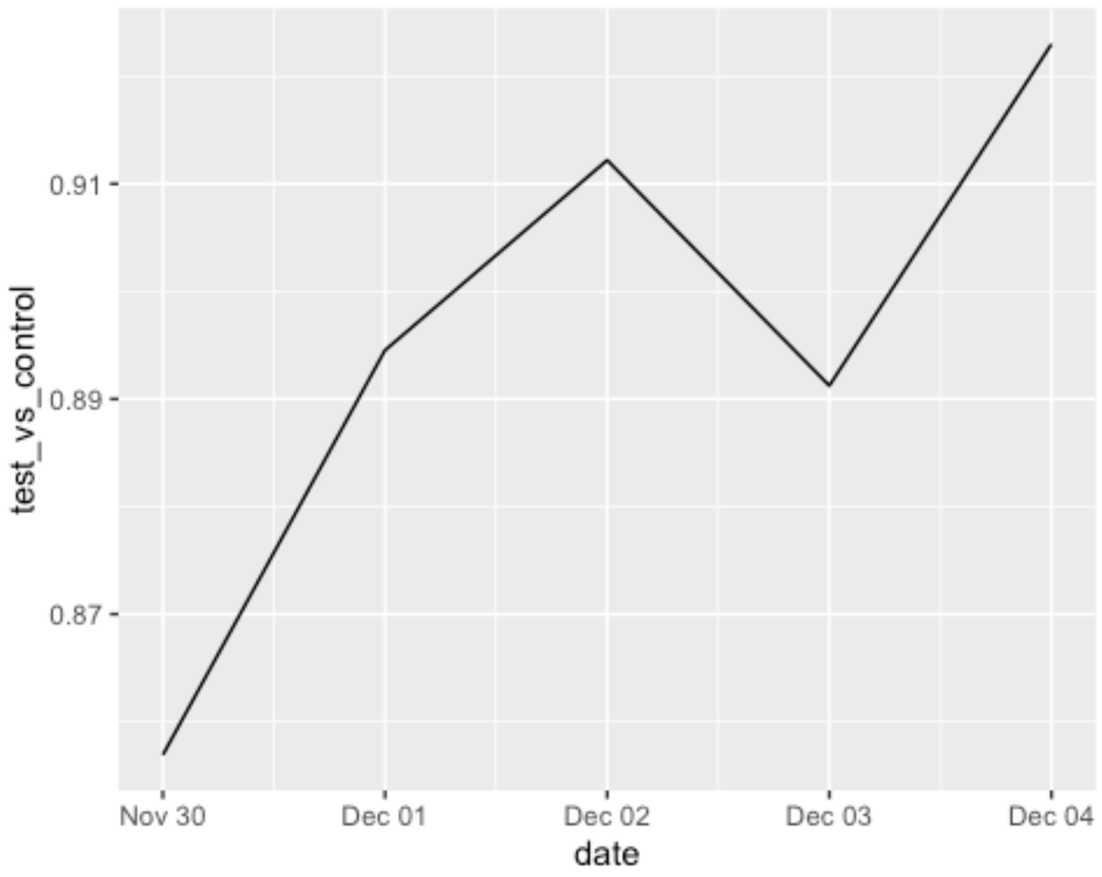
2) Some bias has been introduced in the experiment so that test/control people are not really random

Firstly, let's plot day by day, to see if these weird results have been constantly happening or they just started happening all of a sudden.

```
## Test Results by day
data_test_by_day = data %>%
                   group_by(date) %>%
                   summarize(test_vs_control = mean(conversion[test==1])/
                                                mean(conversion[test==0])
                             )
```

```
## Plot it!
qplot(date, test_vs_control, data= data_test_by_day, geom = "line")

## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



From the plot, we notice a couple of things:

1. Test has constantly been worse than control and there is relatively little variance across days. That probably means that we do have enough data, but there was some bias in the experiment set up

2. As a side note, we just ran it for 5 days. We should always run the test for at least 1 full week to capture weekly patterns, 2 weeks would be much better

Time to find out the bias! Likely, there is some segment of users more likely to end up in test or in control. This segment had a significantly above/below conversion rate and this affected the overall results.

Check A/B Test Randomization

Checking that randomization worked well simply means making sure that all variables have the same distribution in test and control. So, taking for instance the first variable, source, it would mean checking that proportion of users coming from ads, SEO, and direct is the same.

This can easily be done the following way:

```r
## Let's group by source and estimate relative frequencies
knitr::kable(data %>%
  group_by(source) %>%
  summarize(
   test_relative_frequency = 
length(source[test==1])/nrow(data[data$test==1,]),
   control_relative_frequency = 
length(source[test==0])/nrow(data[data$test==0,])
  ))
```

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

source	test_relative_frequency	control_relative_frequency
Ads	0.4006414	0.4012282
Direct	0.1995004	0.2009487
SEO	0.3998582	0.3978231

As we can see, relative frequency of source for different segments is the same. That is, we have basically the same proportion of users coming from Ads, Direct, and SEO in both test and control.

We could potentially keep checking all the variables like this. But it would be extremely time consuming (and boring), especially when you start considering numerical variables and categorical variables with many levels.

So we turn this into a machine learning problem and let an algorithm do the boring work for us. The approach is:

1. Get rid of the conversion variable for now. We don't care about it here. We are just checking if the two user distributions are the same. This is before we check conversion rate for the groups

2. Use the variable test as our label. Try to build a model that manages to separate the users whose test value is 0 vs those whose test value is 1. If randomization worked well, this will be impossible because the two groups are exactly the same. If all

variable relative frequencies were the same as for source, no model would be able to separate test == 1 vs test == 0. If randomization did not work well, the model will manage to use a given variable to separate the two groups.

3. As a model, pick a decision tree. This will allow you to clearly see which variable (if any) is used for the split. That's where randomization failed.

```
## Load the required libraries
library(rpart)
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.3.3

## prepare the data set to check if randomization worked well

## Make it into a classification problem
data$test = as.factor(data$test)

## Build the decision tree
tree = rpart (test ~ .,
              ## Get rid of conversion, not needed here
              subset(data, select=-conversion),
              ## Make classes balanced, easier to understand the output.
              parms = list(prior = c(0.5, 0.5))
              )

## Print the tree!
tree

## n= 401085
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 401085 200542.50 1 (0.5000000 0.5000000)
##   2) country=Bolivia,Chile,Colombia,Costa Rica,Ecuador,El
Salvador,Guatemala,Honduras,Mexico,Nicaragua,Panama,Paraguay,Peru,Venezuela
350218 162347.50 0 (0.5391991 0.4608009) *
##   3) country=Argentina,Uruguay 50867  10574.12 1 (0.2168199 0.7831801) *
```

So we can see that test and control are not the same! Users from Argentina and Uruguay are way more likely to be in test than control (~78% vs 22%, respectively).

Let's double check this. Let's check proportion of Argentinian and Uruguayan users in control vs test.

```
## Dummy variable just for Argentinia and Uruguay users
data$is_argentina = ifelse(data$country == "Argentina",1, 0)
data$is_uruguay =   ifelse(data$country == "Uruguay",1, 0)

knitr::kable(data %>%
  group_by(test) %>% ## Let's check for both test and control
  summarize(
             Argentina_relative_frequency = mean(is_argentina),
             Uruguay_relative_frequency = mean(is_uruguay)
  ))

## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

test	Argentina_relative_frequency	Uruguay_relative_frequency
0	0.0504881	0.0022395
1	0.1732229	0.0172356

Our tree was right! In test, 17% of users are from Argentina, but in control, only 5% of users are from Argentina. Uruguay is even more extreme: test has 1.7% of users from Uruguay and control has basically no Uruguayan users (0.2%).

And this is a big problem because that means we are not comparing anymore apples to apples in our A/B test. The difference we might see in conversion rate might very well depend on the fact that users between the two groups are different.

Let’s check it in practice.

```
## This is the test results using the orginal dataset
original_data = t.test(data$conversion[data$test==1],
data$conversion[data$test==0])

## This is after removing Argentina and Uruguay
data_no_AR_UR = t.test(data$conversion[data$test==1 &
                                        !data$country%in%c("Argentina",
"Uruguay")
                                        ],
                       data$conversion[data$test==0 &
                                        !data$country%in%c("Argentina",
"Uruguay")
                                        ]
                       )

knitr::kable(data.frame(data_type = c("Full", "Removed_Argentina_Uruguay"),
          p_value = c(original_data$p.value, data_no_AR_UR$p.value),
          t_statistic = c(original_data$statistic, data_no_AR_UR$statistic)
))

## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

data_type	p_value	t_statistic
Full	0.0000000	-7.3538952
Removed_Argentina_Uruguay	0.7200849	0.3583456

Huge difference! The biased test where some countries are over/under represented is statistically significant with negative t statistics. So test is worse than control! After removing those two countries, we get non-significant results.

We can also get to the same conclusion by checking how each country is doing taken by itself:

```
data_test_country = data %>%
                    group_by(country) %>%
                    summarize( p_value = t.test( conversion[test==1],
                                                 conversion[test==0]
                                               )$p.value,
                               conversion_test = t.test( conversion[test==1],
                                                         conversion[test==0]
                                                       )$estimate[1],
                               conversion_control = t.test(
conversion[test==1],

conversion[test==0]
                                                          )$estimate[2]
                             ) %>%
                    arrange (p_value)

knitr::kable(data_test_country)
```

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

country	p_value	conversion_test	conversion_control
Mexico	0.1655437	0.0511863	0.0494946
El Salvador	0.2481267	0.0479469	0.0535540
Chile	0.3028476	0.0512950	0.0481072
Argentina	0.3351465	0.0137250	0.0150705
Colombia	0.4237191	0.0505710	0.0520895
Honduras	0.4714629	0.0475398	0.0509058

country	p_value	conversion_test	conversion_control
Guatemala	0.5721072	0.0486472	0.0506429
Venezuela	0.5737015	0.0489783	0.0503437
Costa Rica	0.6878764	0.0547376	0.0522556
Panama	0.7053268	0.0493703	0.0467955
Bolivia	0.7188852	0.0479010	0.0493694
Peru	0.7719530	0.0506043	0.0499140
Nicaragua	0.7804004	0.0541768	0.0526470
Uruguay	0.8797640	0.0129067	0.0120482
Paraguay	0.8836965	0.0492291	0.0484932
Ecuador	0.9615117	0.0489884	0.0491538

After we control for country, the test clearly appears non significant. Not a great success given that the goal was to improve conversion rate, but at least we know that a localized translation didn't make things worse!

At this point, you have two options:

Acknowledge that there was a bug, go talk to the software engineer in charge of randomization, figure out what went wrong, fix it and re-run the test. Note that when you find a bug, it might be a sign that more things are messed up, not just the one you found. So when you find a bug, always try to get to the bottom of it

If you do find out that everything was fine, but for some reason there was only a problem with those two countries, you can potentially adjust the weights for those two segments so that relative frequencies become the same and then re-check the test results