Voter Targeting Exercise

DPI 610

Week 2

February 3, 2021

Introduction

You are a campaign consultant for the Republican candidate in the US Senate race in North Carolina in 2016. Your boss, incumbent Republican Richard Burr (R-NC), is running against challenger Deborah Ross. By all accounts, you are in for a close race. This is challenging because you are trying to learn how to use R and also do a good job as a campaign consultant.

In this exercise, we will be using **R** to load survey data, identify targeted voters, and evaluate the efficiency of that targeting.

Then, we will use a prediction model to construct an (improved) list of voters for persuasion targeting.

Preliminaries: Some Basic R Operations and Object Types

The first step to using **R** is understanding some basic object types and operations, such as the ones below:

Object Types

- Numbers:
  - Integers: 1, 2, 10456
  - Doubles: 1.2, 3.99, -15.75769567
- Strings / characters:
  - Enclosed in single or double quotes.
    - "Joe Biden", 'Bernie Sanders'
    - Numbers in quotes are treated as strings:
- Booleans:
  - Object that can only have the value `TRUE` or `FALSE`.
  - Can be abbreviated to `T` or `F`.
- Groups of Objects:
  - The function `c()` creates vectors and lists:
    - Vectors are sets of the same data type.
    - Lists may have different types.

Question 1

Suppose we conducted a survey and we wanted to record the ages of 5 people we surveyed, aged 21, 36, 27, 55, and 78. In the code chunk below, write code to assign a vector with those five ages to an object called 'ages'. Print your answer using the function `print()`.

Answer 1

```r
#Insert Answer Here
```

Operations

Symbol	Use
`<-`	Assign value to object
`==`	Test equality
`!=`	Test not equal
`&`	And
|	Or
`!`	Not

Question 2

Use the object you just defined, `ages`, and apply a logical operation to identify the ages in that vector that are over the age of 40 years old. Print your answer using the function `print()`.

Answer 2

```r
#Insert Answer Here
```

Indexing and Subsetting

Sometimes we want to refer to just some values in a dataset or variable. We can subset using square brackets: `[]`

- Use a sequence `x[1:5]`
- Use a vector `x[c(1, 3, 9, 5)]`
- Exclude using a negative number: `x[-3]`; `x[-length(x)]`
- The first value in a vector is indexed to `1` (not `0`, as in some other languages).

Question 3

Subset the vector defined in Question 1 (`ages`) to ages over 40.

Answer 3

```r
#Insert Answer Here
```

Analysis of CCES Survey

Key Variables in CCES Survey

The CCES is a political survey that contains some key demographic variables of voters, as well as their pre-election vote intentions. We have pre-loaded the dataset in the R environment. The dataset is saved in an object called `nc_cces`.

Let's examine a few key variables from this survey.

Variable name	Description
`case_id`	Unique identification number of respondents
`age`	Age
`gender`	Gender (F or M)
`vv_regstatus`	Validated registration status
`race`	Race
`familyincome`	Family income
`college_grad_prob`	College graduates (1) or not (0)
`married_prob`	Married (1) or not (0)
`intent_sen`	Voting intention for Senate race

Question 4

Subset columns of dataset `nc_cces` so that it only contains `"case_id"`,`"age"`,`"gender"`,`"party"`,`"vv_regstatus"`, and `"race"`. Use `head()` function to view the first 5 rows of the subsetted dataset.

Answer 4

```r
#insert answer here
```

Accessing Variables in Data Frame

There are many ways to access variables within data frames.

`object_name$variable_name` refers to `variable_name` within `object_name`.

```r
nc_cces$age[1:10]
```

```
##  [1] 53 74 42 58 55 52 77 40 31 47
```

```r
mean(nc_cces$age)
```

```
## [1] 47.56986
```

```r
mean(nc_cces$married_prob)
```

```
## [1] 0.5893214
```

```r
class(nc_cces$married_prob)
```

```
## [1] "integer"
```

```r
class(nc_cces$race)
```

```
## [1] "factor"
```

Mathematical operations on a vector/variable apply to every value:

```
nc_cces$age[1:10] * 2
```

```
##  [1] 106 148  84 116 110 104 154  80  62  94
```

```
#Suppose we wanted to create a new age-squared variable
nc_cces$age_squared <- nc_cces$age^2

nc_cces$age[1:10]
```

```
##  [1] 53 74 42 58 55 52 77 40 31 47
```

```
nc_cces$age_squared[1:10]
```

```
##  [1] 2809 5476 1764 3364 3025 2704 5929 1600  961 2209
```

Persuadable Voters

Before you started working for the Burr campaign, the previous consultant decided to stick to a simple criterion for identifying persuadable voters. The previous consultant cut a list from the voter file that identified persuadable voters as all people who: (1) Had active registration, (2) Did not belong to either party, and (3) Were White.

Let's identify who fits these criteria.

```
nc_cces$old_list <- as.numeric(nc_cces$vv_regstatus=="Active" &
                                 nc_cces$party %in% c("NPA","LIB") &
                                 nc_cces$race == "C")
```

The old list comprises 15.22% of the total sample.

Question 5

Suppose instead the previous consultant had not used race to identify persuadable voters but had instead used age, particularly people 65 and over. Define a new variable in the dataset called 'old_list2' based on this. What percent of the sample would that list have comprised?

Answer 5

```
#Insert Answer Here
```

To determine the "efficiency" of the actual old list, let's examine the share of voters in the list who when polled said that they did not intend to vote for either the Democratic or Republican candidate.

We use the `%in%` operator preceded by an "!", which means "not". So the statement below says to code as "undecided" anyone who did not say their intent was to vote for a D or R candidate.

```
nc_cces$undecided <- as.numeric(!(nc_cces$intent_sen %in% c("[Democrat / Candidate 1]",
                        "[Republican / Candidate 2]")))
```

Now, to find the share of voters in the old list who were really undecided (i.e., the efficiency), just take the mean of the binary variable "undecided", conditional on being part of the old list.

```r
old_list_efficiency<-mean(nc_cces$undecided[nc_cces$old_list==1])
print(old_list_efficiency)
```

```
## [1] 0.3081967
```

```r
pop_efficiency <- mean(nc_cces$undecided)
```

The efficiency for the old list is 30.82%. This compares to an efficiency of 31.74% in the full population. So the old method was actually doing a worse job than just randomly contacting people, if the goal was to target undecided voters.

Question 6

If the previous consultants had replaced the race criterion with age (65 and over), would it have resulted in a more or less efficient list targeting persuadable voters?

Answer 6

```r
#Insert Answer Here
```

We can also determine the old list's coverage (recall coverage is the share of target voters in the full population actually reached by the list). In this case, to determine coverage we find the percent of all undecided voters in the old list.

```r
old_list_coverage <- sum(nc_cces$undecided==1 & nc_cces$old_list==1)/sum(nc_cces$undecided==1)
```

The coverage for the old list is 14.78%.

Overall, it is safe to say that the old method of identifying persuadable target voters was not working too well.

Candidate Support Model

Let's now create a simple model for understanding who supports our candidate, Richard Burr (R-NC). Let's be sure to make use of the new information we have from our survey on who various voters are likely to support. (Also, for this exercise, let's ignore any issues with survey weights – we'll cover this in a later part of the course.)

We want to split our data into a training and a test set, so we can get a sense of how well our model would perform on new data. Note that this was done at the start in the 'setup' code chunk.

```r
# Create binary variable that has value 1 for people who say they
#intend to vote for Burr, and 0 otherwise.

nc_cces$burr <- as.numeric(nc_cces$intent_sen == "[Republican / Candidate 2]")

# Run a logistic model
model1 <- glm(burr~as.factor(gender)+as.factor(race)+age
              +I(age^2)+log(1+familyincome)+college_grad_prob
              +married_prob,
              data=subset(nc_cces,case_id %in% train_id),family="binomial")

summary(model1)
```

```
##
## Call:
## glm(formula = burr ~ as.factor(gender) + as.factor(race) + age +
##     I(age^2) + log(1 + familyincome) + college_grad_prob + married_prob,
##     family = "binomial", data = subset(nc_cces, case_id %in%
##         train_id))
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3181  -0.8892  -0.5461   1.1778   2.6550
##
## Coefficients:
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -4.2271581  1.1072238  -3.818 0.000135 ***
## as.factor(gender)M     0.2164099  0.1318997   1.641 0.100856
## as.factor(race)H      -0.0617720  0.2634154  -0.235 0.814594
## as.factor(race)M       0.1890033  0.5962853   0.317 0.751268
## as.factor(race)B      -1.7291082  0.2315744  -7.467 8.22e-14 ***
## as.factor(race)N       0.0108522  0.7660095   0.014 0.988697
## as.factor(race)A      -0.2098470  0.5359274  -0.392 0.695384
## age                    0.0417638  0.0257564   1.621 0.104912
## I(age^2)              -0.0002453  0.0002593  -0.946 0.344097
## log(1 + familyincome)  0.1767028  0.0950447   1.859 0.063005 .
## college_grad_prob     -0.3160819  0.1491456  -2.119 0.034066 *
## married_prob           0.5343004  0.1491183   3.583 0.000340 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1612.4  on 1336  degrees of freedom
## Residual deviance: 1440.3  on 1325  degrees of freedom
##   (167 observations deleted due to missingness)
## AIC: 1464.3
##
## Number of Fisher Scoring iterations: 5
```

Now let's use the fitted values from the model to predict a support score for Burr. Someone who has a score around 0.5 should be most on the fence. Let's start by classifying anyone between 0.3 and 0.7 as undecided. And then calculate efficiency and coverage when using our model in the test set.

```r
#predicted probabilities
yhat <- predict(model1,newdata=subset(nc_cces,case_id %in% test_id),type="response")

#predicting persuadable voters
nc_cces$undecided_predict[nc_cces$case_id %in% test_id] <- as.numeric(yhat>=0.3 & yhat < 0.7)
nc_cces$yhat[nc_cces$case_id %in% test_id] <- yhat

#new list based on persuadable voters
new_list <- subset(nc_cces,undecided_predict==1 & case_id %in% test_id)

#calculate efficiency of new list
burr_test_efficiency <- mean(new_list$undecided,na.rm=T)
old_test_efficiency <- mean(nc_cces$undecided[nc_cces$old_list ==1
                                              & nc_cces$case_id %in% test_id],na.rm=T)
```

```
burr_test_coverage <- sum(new_list$undecided)/sum(nc_cces$undecided[nc_cces$case_id %in% test_id])
old_test_coverage <- sum(nc_cces$undecided[nc_cces$old_list ==1
                                           & nc_cces$case_id %in% test_id],na.rm=T)/
                                            sum(nc_cces$undecided
                                            [nc_cces$case_id %in% test_id],na.rm=T)
```

In the test set, our model results in an efficiency of 26.58% as compared to an efficiency of 31.34% in the same test data for the old list. The level of coverage in the new list based off of our model is 36.42% as compared to 12.96% using the old list.

This is not great. Let's assess how well the model is actually working using accuracy, precision and recall. We will assess it's predictions in the test set (comparing them to actual vote intention).

```
nc_test_data <- subset(nc_cces,case_id %in% test_id)

print("2x2 Table, Predicted vs. Actual")
```

```
## [1] "2x2 Table, Predicted vs. Actual"
```

```
out_table<-table(nc_test_data$undecided_predict,nc_test_data$undecided)
print(out_table)
```

```
##
##       0   1
##   0 134  93
##   1 163  59
```

```
accuracy<-(out_table[1,1]+out_table[2,2])/sum(out_table)
print(paste("Accuracy, i.e., share of predictions correct is:",percentit(accuracy)))
```

```
## [1] "Accuracy, i.e., share of predictions correct is: 42.98%"
```

```
precision <- out_table[2,2]/(out_table[2,2]+out_table[2,1])
recall <- out_table[2,2]/(out_table[2,2]+out_table[1,2])

print(paste("Precision is:",percentit(precision)))
```

```
## [1] "Precision is: 26.58%"
```

```
print(paste("Recall is:",percentit(recall)))
```

```
## [1] "Recall is: 38.82%"
```

Question 7

Your boss says that this level of model performance and efficiency is not good enough. Can you come up with an alternative model and/or threshold that improves on the results from above?

Answer 7

```
#Insert Answer Here
```