Class 15 Demo: Sample Size Planning

Announcement

Make sure to update your `mktg482` so that you have the sample size functions. You can do this by entering the following into the R console

```
update_mktg482()
```

You only need to do this once.

One proportion

This document illustrates how to obtain a sample size for a proportion that yields a given level of precision as assessed by the halfwidth of the confidence interval at a given level of confidence (the total width of the interval is of course twice the halfwidth). We begin with the case of a single proportion and the corresponding function `ss.proportion`.

Question: What does it mean for a confidence interval to have an x% confidence level?

Answer: It means that if one were to take many, many different samples and compute an x% confidence interval for each sample, then x% of those confidence intervals would contain the true value and (1-x)% would not.

Okay, now suppose we want to estimate a single proportion with precision 0.03 at 95% confidence. The sample size we would need is:

```
library(mktg482)
ss.proportion(0.03)
```

```
[1] 1068
```

If we required more (could tolerate less) precision, we would need a larger (smaller) sample size:

```
ss.proportion(0.01)
```

```
[1] 9604
```

```
ss.proportion(0.05)
```

```
[1] 385
```

Similarly, if we required more (could tolerate less) confidence, we would need a larger (smaller) sample size:

```
ss.proportion(0.03, confidence=0.99)
```

```
[1] 1844
```

```
ss.proportion(0.03, confidence=0.90)
```

```
[1] 752
```

Note: The default value of `confidence` is 0.95; that will be the confidence level unless we explicitly tell the function otherwise.

The sample size calculations above are conservative in that they assume the true (unknown) proportion is the one that gives the widest confidence interval. If we have a ballpark estimate of the true proportion (e.g., 20%), we typically need a smaller sample size:

```
ss.proportion(0.03, proportion=0.2)
```

```
[1] 683
```

```
ss.proportion(0.01, proportion=0.2)
```

```
[1] 6147
```

```
ss.proportion(0.05, proportion=0.2)
```

```
[1] 246
```

```
ss.proportion(0.03, confidence=0.99, proportion=0.2)
```

```
[1] 1180
```

```
ss.proportion(0.03, confidence=0.90, proportion=0.2)
```

```
[1] 481
```

Note: The default value of `proportion` is 0.5 because this is the conservative value that gives the widest confidence interval; that will be the value unless we explicitly tell the function otherwise.

In sum, the `ss.proportion` function requires us to specify a desired `halfwidth` and allows us to also optionally specify a level of `confidence` and ballpark estimate of the `proportion`. How do we use this in practice? Let's walk through how to choose each of the three input parameters starting with `halfwidth`.

The `halfwidth` parameter is the "plus or minus" amount we can tolerate in our application. For example, in political polling, it is common to want to estimate a candidate's support up to +/-3% and so this requires a sample size of:

```
ss.proportion(0.03)
```

```
[1] 1068
```

Next, let's consider the `proportion` parameter. Oftentimes, we have a good sense of baseline customer behavior. For instance, suppose baseline churn has historically been about 20%. In that case, if we wanted to estimate churn up to +/-3% but have reason to believe it is about 20% (i.e., because we do not expect the new strategy to change churn dramatically), then we would also set `halfwidth` equal to 0.03 and `proportion` equal to 0.2.

```
ss.proportion(0.03, proportion=0.2)
```

```
[1] 683
```

On the other hand, if we did have strong expectations about how the new strategy might change churn, say to 10%, we would set `proportion` equal to 0.1.

```
ss.proportion(0.03, proportion=0.1)
```

```
[1] 385
```

As for the `confidence` parameter, 95% is the default used overwhelmingly in practice. However, some applications require more confidence and others less so we should always think hard about what level of confidence ours requires!

Two proportions

Suppose now instead of comparing a new churn reduction strategy to a known baseline, we want to compare two new churn reduction strategies to each other (Note: the additional strategy could even be our current

strategy, which might be a good thing to test if we are concerned times have changed, i.e., lack of situational invariance).

We now use the function `ss.2proportion`. Like `ss.proportion`, `ss.2proportion` requires us to specify a desired `halfwidth` and allows us to also optionally specify a level of `confidence` and ballpark estimates of `proportion1` and `proportion2`.

The `confidence`, `proportion1`, and `proportion2` parameters are exactly as in `ss.proportion`. However, the `halfwidth` parameter now refers to the halfwidth of the confidence interval of the *difference* between the two proportions.

As before, we choose the `halfwidth` parameter based on what "plus or minus" we can tolerate in our application, bearing in mind that this "plus or minus" is on the difference of the two proportions not the proportions themselves. Suppose we want estimate this difference up to +/-3%. We would set the `halfwidth` paramter to 0.03.

```
ss.2proportion(0.03)
```

```
[1] 2135
```

The sample size we need is roughly double what we needed before. But, because we are now testing two strategies, this means we need a sample size of four times what we needed before (i.e., the function gives the sample size per strategy).

Again, as before if baseline churn has historically been about 20% and we do not expect the new strategies to change churn dramatically, we can also set `proportion1` and`proportion2` equal to 0.2

```
ss.2proportion(0.03, proportion1=0.2, proportion2=0.2)
```

```
[1] 1366
```

On the other hand, if we did have strong expectations about how the new strategies might change churn, say to 10% and 15% respectively, we would set `proportion1` equal to 0.1 and `proportion2` equal to 0.15.

```
ss.2proportion(0.03, proportion1=0.1, proportion2=0.15)
```

```
[1] 929
```

Additional Considerations

Sample size planning is complicated and highly technical. Some issues we have not touched on:

1. When the sample size calculated by the function is large relative to the total population size (e.g., greater than five percent), we may want to consider a so-called "finite population correction". This can happen in B2B contexts where the total size of the customer base is relatively small.
2. When customers fall into heterogeneous customer groups (i.e., so there are large differences between groups but small differences within groups), we may want to consider stratified random sampling.
3. When it is too costly or infeasible to randomly sample in a simple fashion, we may want to consider cluster sampling. This performs best when there are small differences between groups and large differences within groups.
4. When the differences within groups themselves differ across groups or some groups are more costly to sample than other groups, we may want to sample an unequal number per group.
5. What if there are more than two groups? We have only touched on the case of one group or two groups.
6. Are we sure we can sample randomly (or sample randomly within groups if applicable)?

Note: In all of the above, we can replace "customer groups" with "churn strategies" or other interventions.

Be sure to discuss these issues with your data scientist!