Binomial Tests

Riley Smith

09 November 2016

Contents

Setting Things Up in R

```
source("SETUP.R")
options(width = 70)
knitr::opts_chunk$set(
    tidy = FALSE,
    echo = TRUE,
    cache = FALSE,
    fig.keep = 'high',
    fig.show = 'asis',
    results = 'asis',
    # autodep = T,
    Rplot = TRUE,
    dev = 'pdf',
    fig.path = 'graphics/binomials/rplot_',
    fig.width = 7,
    fig.asp = 1,
    out.width = "\\linewidth",
    collapse = TRUE
    )
library(foreign) ## read.spss() ##
library(car) ## recode() ##
dat <- read.spss("data/reuters.sav", to.data.frame = TRUE)
dat <- within(dat, {recode(response, c("1=0", "2=1", "3=2"))})
```

NOTES ON THE R-code ABOVE: SETUP.R is the default R-script I source in the "setup" code-chunk[1] at the beginning of all R markdown documents. It contains global arguments for loading commonly used packages, setting options, and defining various R-object utilities and functions. I also use this script as a record of functions I create while working in R. The script is heavily commented throughout for explanatory purposes, as well as for giving credit where it is due (I've tried to keep up with all of the R-code, LaTex, and R markdown sources, but may have missed some along the way).

[1] see help(package = knitr)

AFTER SOURCING SETUP.R and setting a few output options (options() & opts_chunk$set()), I load the {foreign} and {car} packages for the read.spss() and recode() functions, respectively. Then I read in (read.spss()) the Reuters polling dataset, "reuters.sav" and store it in R as a dataframe named "dat". Finally, I re-code the values for the "dat$response" variable, using the {car} package's recode() function to undo default numeric values 1,2,3 and label values.

Note: *This last recoding step seems unneccessary to me. I included it here based on Jason Newsom's code and in-class explanation, but when I run the rest of the analysis below without recoding, the results are the same. Is there a specific situation in which setting a discrete variable's values to '0, 1, 2, ...', versus '1, 2, 3, ...' is consequential to the analysis? Or is this a matter of personal preference for coding discrete variables? I think it would make more logical sense to me if, in this case, '0' strictly reflected 'No Response', but that is not the case here, as '0' reflects both 'Other' and 'No Response'. Further, should 'Other' responses not be coded as separate from 'No Response', given that an 'Other' response is absolutely qualitatively distinct from 'No Response'?*

See Newsom 2016-CDA Handout-4

Data-Cleaning & Preparation

A few things need to happen to help optimize the information we put into and get out of the binomial hypothesis testing for the polling data. Specifically, I want set all values of "other" to "NA" in dat$response, then drop all "NA" values since the analysis is only interested in responses for the two major candidates (i.e., *Clinton and Trump*).

Note that the common method for binomial tests in R is to use exact.test = TRUE, which is actually the most conservative approach and also no ideal for binomial tests. Use prop.test() for *approximate* tests. *(source: Jason Newsom)*

HOWEVER, there are also some existing "NAs" in "dat$party", which need to be re-coded as well to avoid excluding rows with acceptable data values in dat$response.

sapply(), apply(), and vapply() are great function to get to know if you find yourself doing a ton of data cleaning/mining.

```
Risna <- function(x) sum(is.na(x))
#   ## Getting a count of NA values in the original dataframe ##
sapply(dat, Risna)
```

id	response	party	partmiss	ind
0	0	377	0	0

```
R.na <- function(x, v = 0){
    ## x = object to be manipulated,
    ## v = value to assign to NAs ##
```

```
x <- ifelse(is.na(x), v, x)
return(x)
}
```

`R.na()`: *"If x = NA (`is.na()`), replace x with v, otherwise leave x alone."*

Table 2: Summary information for '**party**' data column *before* recoding `NA`s

M	0.42
SD	0.49
Min	0.00
Max	1.00
NAs	377.00

```
unique(dat$party)
```

0, 1 and *NA*

```
dat$party <- sapply(dat$party, R.na, v = 99)
unique(dat$party)
```

0, 1 and *99*

Table 3: Summary information for '**party**' data column *after* recoding `NA`s

M	24.59
SD	42.42
Min	0.00
Max	99.00
NAs	0.00

```
dat$response <- recode_factor(dat$response,
                              "other/no opinion" = NA_character_)
## see "recode_factor()" in the {dplyr} package ##
dat <- na.omit(dat)
sapply(dat, R.isna) ## bye-bye NAs! ... again ##
```

id	response	party	partmiss	ind
0	0	0	0	0

```
## but this time we only lost data for rows with NA
## in dat$response (but we did lose ALL of the data
## for those rows, as these were removed from the
## dataframe entirely, though the original datafile remains untouched).
```

Now the data are, in my opinion, ready for analysis.

Single-Sample Binomial Tests: Differences in proportions of (non-"other") polling responses.

```
levels(dat$response)
```

Trump and *Clinton*

```
poll.t <- table(dat$response)
## Why not make a table of the poll response counts for each candidate? ... ##
```

Table 5: Frequency Table of Polling Data

Response	Frequency
Trump	554
Clinton	677

DEFINITION OF A CONVENIENCE FUNCTION FOR THE BINOMIAL TEST. I'm combining the `prop.test()` & `binom.test()` functions (`{pkg:stats}`) because I think it's kind of ridiculous that there is not already a combined function for these. I also don't particularly enjoy the default output format for either of these functions, so I'm breaking the function writing rule of simplicity (AKA: *"Curly's Law"*) and implementing some formatting tasks within the function as well.

Arguments (`R.binom_test()`):

`p`. The target proportion to be tested against the null hypothesis (H_0; π_0; see `pi0` below). *Synonymous Arguments:* `x` in `prop.test()` & `binom.test()`.

`N`. The size of the sample from which '*p*' is taken. *Synonymous Arguments:* `n` in `prop.test()` & `binom.test()`. *Synonymous Arguments:* `n` in `prop.test()` & `binom.test()`.

`pi0`. [Default = `0.5`]. A vector of probabilities of success corresponding to the value(s) in `p`. These probabilities represent the null hypothesis value (H_0; π_0) against which `p` is to be tested. *Synonymous Arguments:* `p` & `conf.level` (inverse) in `prop.test()` & `binom.test()`.

`exact`. Logical [Default = `FALSE`]. Should the the hypothesis be tested using an exact binomial test (i.e., `binom.test()`). If `FALSE` (the default), a test of equal or given proportions, depending on the lengths of `p` and `pi0` is conducted using `prop.test()`

`correct`. Logical Default = `FALSE`]. Synonymous with the `correct`

argument in `prop.test()`.

`digits`. [Default = 2]. Number of digits to use when rounding (`round()`) the final output values (does not influence the test calculation).

`...`. Additional arguments to be passed to either `prop.test()` or `binom.test()`, depending on the value set for `exact` (e.g., `alternative`).

Value (`R.binomTest()`): Returns a `data.frame` object containing the values returned by either `prop.test()` or `binom.test()`, depending on the value set for `exact`.

```
R.binom_test <- function(p, N, pi0 = 0.5, exact = FALSE, correct = FALSE,
                         digits = 2, ...){
    if (exact) { ## Hypothesis Testing
        BT <- stats::binom.test(x = p, n = N, p = pi0, ...)
        }
        else {
            BT <- stats::prop.test(x = p, n = N, p = pi0,
                                   correct = correct, ...)
        }
    ## The rest deals with formatting the output ##
        BT$data.name <- paste0(p, " out of ", N, " null probability ",
                               BT$null.value)
        ## Above, I modified the default output value for *.test$data.name
            ## to print the actual data values, rather than the object
            ## names the values are stored under (see output below) ##
        BTCI <- paste0(round(BT$conf.int[[1]], digits = digits), ", ",
                       round(BT$conf.int[[2]], digits = digits))
        BT$p.value <- round(BT$p.value, digits = 7)
        BT.df <- data.frame(c(BT[c("alternative",
                                   "null.value",
                                   "parameter",
                                   "estimate",
                                   "statistic",
                                   "p.value")],
                              BTCI))
        row.names(BT.df) <- NULL
        return(BT.df)
}
```

Approximate Test (`prop.test()`)

```
pt <- R.binom_test(p = poll.t["Clinton"], N = nrow(dat), pi0 = 0.5)
    ## ... Now we know where values for prop.test() came from :) ##
        ## There’s more than one way to do that, by the way, but
        ## creating the table will come in handy later on too. ##
```

Table 6: 1-sample proportions test without continuity correction: *677 out of 1231*

H_1	π_0	df	p	χ_2	*p-value*	CI
two.sided	0.5	1	0.55	12	0.00046	0.52, 0.58

Exact Test (`binom.test()`)

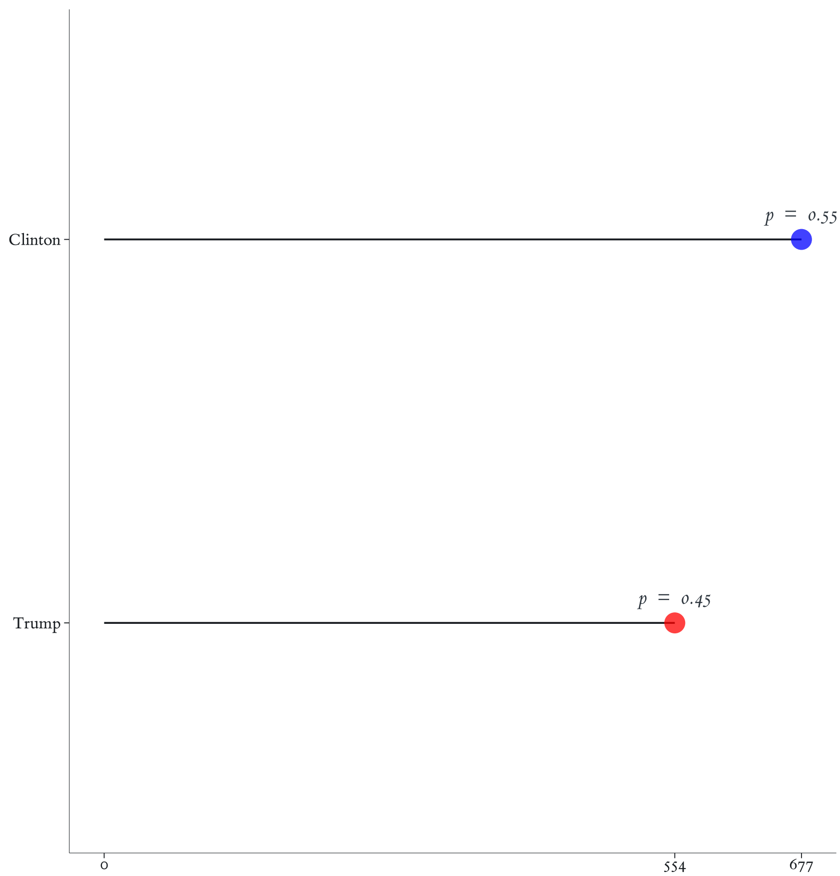
```
et <- R.binom_test(p = poll.t["Clinton"],
                   N = nrow(dat), pi0 = 0.5,
                   exact = TRUE)
```

Table 7: 1-sample *exact* binomial test with continuity correction: *677 out of 1231*

H_1	π_0	n_{trials}	p	$n_{successes}$	*p-value*	CI
two.sided	0.5	1231	0.55	677	5e-04	0.52, 0.58

```
poll.df <- as.data.frame(poll.t)
names( poll.df) <- c("Response", "Frequency")
 poll.df$N <- rep(x = nrow(dat), times = nrow( poll.df))
n <-  poll.df[, 2]

electpal <- c("red", "blue")
electpal <- sapply(electpal, adjustcolor, alpha = 0.75, USE.NAMES = FALSE)

bpoll <- ggplot( poll.df, aes(x = Frequency, y = Response)) +
    geom_segment(aes(yend = Response), xend = 0, colour = mypal[20]) +
    geom_point(size = 5, aes(colour = Response)) +
```

```
    scale_colour_manual(values = electpal, guide = FALSE) +
    labs(y = "", x = "") + thm_tft(xline = TRUE, yline = TRUE)

bpoll2 <- bpoll + scale_x_continuous(breaks = c(0, n),
                                     limits = c(0, max(n)))

bpoll2 + geom_text(vjust = -1.5, hjust = 0.5, stat = 'identity',
                   position = 'identity', colour = mypal[19],
                   size = rel(4), aes(family = "ETBembo",
                                      fontface = "italic",
                                      label = paste("p  = ",
                                                    round(n/ poll.df$N,
                                                          digits = 2))))
```



See Newsom 2016-CDA Handout-3

z-Score Test

Here we are taking one of the favorability proportions (i.e. "sucess proportions") and comparing it to the *Null Hypothesis* (H_0) represented by π_0 (i.e., 0.50).[2]

[2] Agresti, *Catagorical Data Analysis*; Agresti and Coull, "Approximate Is Better Than 'Exact' for Interval Estimation of Binomial Proportions."
Data: 2016 Polling data

$$z = \frac{0.55 - 0.5}{\sqrt{\frac{0.5(1-0.5)}{1231}}} \Rightarrow \frac{0.05}{\sqrt{\frac{0.5(0.5)}{1231}}} \Rightarrow \frac{0.05}{\sqrt{\frac{0.25}{1231}}} \Rightarrow \frac{0.05}{\sqrt{0}} \Rightarrow \frac{0.05}{0.0142509}$$

$$z = \mathbf{3.508561}$$

$$z = \frac{p - \pi_0}{SE_\pi}$$
$$SE_\pi = \sqrt{\frac{\pi_0(1-\pi_0)}{n}}$$

z-Score Test: Lower and Upper Confidence Limits

Confidence limits are calculated by the favorability proportion (p) $\pm$ the $z_{critical}$ value multiplied by the standard error of the estimate (SE_π).

$$CI = p \pm (z_{critical})(SE_\pi)$$
$$CI_z = p \pm (z_{critical})(SE_\pi)$$
$$z_{critical} = 1.96$$
$$p = 0.55$$

$$LCL_z = p - (1.96)(0.0141794) \Rightarrow p - 0.0277917 \Longrightarrow \mathbf{0.5222083}$$

$$UCL_z = p + (1.96)(0.0141794) \Rightarrow p + 0.0277917 \Longrightarrow \mathbf{0.5777917}$$

$$CI_z = \mathbf{0.52} < p < \mathbf{0.58}$$

z-Score Test Margin of Error (ME_z)

$$ME_z = \left(\frac{CI_z^+ - CI_z^-}{2}\right)(100)$$

$$ME_z = \frac{(0.5222083 - 0.5777917)}{2}(100) \Rightarrow \frac{(-0.0555833)}{2}(100) \Rightarrow (-0.0277917)(100)$$

$$ME_z = \mathbf{-2.779167}$$

Goodness-of-Fit Tests (χ^2)

Table 8: Summary of Known Values and Parameters for Pearson's χ^2 Goodness-of-Fit Test

N	$n_{Clinton}$ †	n_{Trump} ‡	$z_{critical}$ $(df = 1)$
1231	677	554	3.84

Evaluate the observed χ^2 value to the χ^2 distribution ($f_k(x)$).

$\chi^2 = \Sigma \frac{(O_i - Ei)^2}{E_i}$

$O_1 = 677$

$O_2 = 554$

$$E_i = \frac{(O_1 + O_2)}{2}$$

The χ^2 test's flexibility allows for additional comparison analyses. The Likelihood Ratio χ^2 is similar to the pearson χ^2.

$$E_i = \frac{(677 + 554)}{2} \Rightarrow \frac{(1231)}{2}$$

$$E_i = \mathbf{615.5}$$

$$\chi^2 = \frac{(677 - 615.5)^2}{615.5} + \frac{(554 - 615.5)^2}{615.5}$$

$$\Longrightarrow \frac{(61.5)^2}{615.5} + \frac{(-61.5)^2}{615.5}$$

$$\Longrightarrow \frac{(3782.25)}{615.5} + \frac{(3782.25)}{615.5}$$

$$\Longrightarrow 6.1450041 + 6.1450041$$

$$\chi^2 = \mathbf{12.2900081}$$

References[3]

Agresti, Alan. *Catagorical Data Analysis*. Third Edition. New York, NY: Wiley & Sons, 2013.

Agresti, Alan, and Brent A Coull. "Approximate Is Better Than 'Exact' for Interval Estimation of Binomial Proportions." *The American Statistician* 52, no. 2 (1998): 119–26.

Allaire, JJ, Joe Cheng, Yihui Xie, Jonathan McPherson, Winston Chang, Jeff Allen, Hadley Wickham, Aron Atkins, and Rob Hyndman. *rmarkdown: Dynamic Documents for R*, 2016. https://CRAN.R-project.org/package=rmarkdown.

Arnold, Jeffrey B. *Ggthemes: Extra Themes, Scales and Geoms for Ggplot2*, 2016. https://CRAN.R-project.org/package=ggthemes.

Aust, Frederik, and Marius Barth. *Papaja: Create APA Manuscripts with RMarkdown*, 2015. https://github.com/crsh/papaja.

Boettiger, Carl. *knitcitations: Citations for Knitr Markdown Files*, 2015. https://CRAN.R-project.org/package=knitcitations.

Chang, Winston. *Extrafont: Tools for Using Fonts*, 2014. https://CRAN.R-project.org/package=extrafont.

Daroczi, Gergely, and Roman Tsegelskyi. *Pander: An R Pandoc Writer*, 2015. https://CRAN.R-project.org/package=pander.

Fox, John, and Sanford Weisberg. *An R Companion to Applied Regression*. Second. Thousand Oaks CA: Sage, 2011. http://socserv.socsci.mcmaster.ca/jfox/Books/Companion.

Francois, Romain. *Bibtex: Bibtex Parser*, 2014. https://CRAN.R-project.org/package=bibtex.

———. *Highlight: Syntax Highlighter*, 2015. https://CRAN.R-project.org/package=highlight.

Qiu, Yixuan. *Showtext: Using Fonts More Easily in RGraphs*, 2015. https://CRAN.R-project.org/package=showtext.

Qiu, Yixuan, and others. *Sysfonts: Loading System Fonts into R*, 2015. https://CRAN.R-project.org/package=sysfonts.

R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing, 2016. https://www.R-project.org/.

Wickham, Hadley. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. http://ggplot2.org.

———. *Gtable: Arrange Grobs in Tables*, 2016. https://CRAN.R-project.org/package=gtable.

———. *Scales: Scale Functions for Visualization*, 2016. https://CRAN.R-project.org/package=scales.

———. "The Split-Apply-Combine Strategy for Data Analysis." *Journal of Statistical Software* 40, no. 1 (2011): 1–29. http://www.

[3] **Note:** This document was created using ***R**-v3.3.2* R Core Team, *R*, and the following ***R**-packages*: *base-v3.3.* R Core Team, *R*, *bibtex-v0.4.* Francois, *Bibtex*, *car-v2.1.* Fox and Weisberg, *An R Companion to Applied Regression*, *dplyr-v0.5.* Wickham and Francois, *Dplyr*, *DT-v0.2.* Xie, *DT*, *extrafont-v0.17.* Chang, *Extrafont*, *ggplot2-v2.1.* Wickham, *Ggplot2*, *knitcitations-v1.0.* Boettiger, *knitcitations*, *knitr-v1.14.* Xie, *Dynamic Documents with R and Knitr*, *pander-v0.6.* Daroczi and Tsegelskyi, *Pander*, *papaja-v0.1.* Aust and Barth, *Papaja*, *plyr-v1.8.* Wickham, "The Split-Apply-Combine Strategy for Data Analysis.", *rmarkdown-v1.1.* Allaire et al., *rmarkdown*, *scales-v0.4.* Wickham, *Scales*, *tidyr-v0.6.* Wickham, *Tidyr*, *ggthemes-v3.2.* Arnold, *Ggthemes*, *gtable-v0.2.* Wickham, *Gtable*, *kableExtra-v0.0.* Zhu, *KableExtra*, *tufte-v0.2.* Xie and Allaire, *Tufte*, *devtools-v1.12.* Wickham and Chang, *Devtools*, *highlight-v0.4.* Francois, *Highlight*, *sysfonts-v0.5.* Qiu and others, *Sysfonts*, and *showtext-v0.4.* Qiu, *Showtext*

jstatsoft.org/v40/i01/.

———. *Tidyr: Easily Tidy Data with Spread() and Gather() Functions*, 2016. https://CRAN.R-project.org/package=tidyr.

Wickham, Hadley, and Winston Chang. *Devtools: Tools to Make Developing RPackages Easier*, 2016. https://CRAN.R-project.org/package=devtools.

Wickham, Hadley, and Romain Francois. *Dplyr: A Grammar of Data Manipulation*, 2015. https://CRAN.R-project.org/package=dplyr.

Xie, Yihui. *DT: A Wrapper of the Javascript Library Datatables*, 2015. https://CRAN.R-project.org/package=DT.

———. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC, 2015. http://yihui.name/knitr/.

Xie, Yihui, and JJ Allaire. *Tufte: Tufte's Styles for Rmarkdown Documents*, 2016. https://CRAN.R-project.org/package=tufte.

Zhu, Hao. *KableExtra: Decorate Kable Output Using Pipe Syntax*, 2016. https://github.com/haozhu233/kableExtra.