Categorical Data Visualization

Riley Smith

14 November 2016

Reuters 2016 Polling Data

Data-Cleaning \& Preparation

```r
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 1: 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 _ _

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

Table 2: Summary information for '**party**' data column *after* recoding `NAs`

M	24.59
SD	42.42
Min	0.00
Max	99.00
NAs	0.00

0, 1 and *99*

```
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 & plotting.

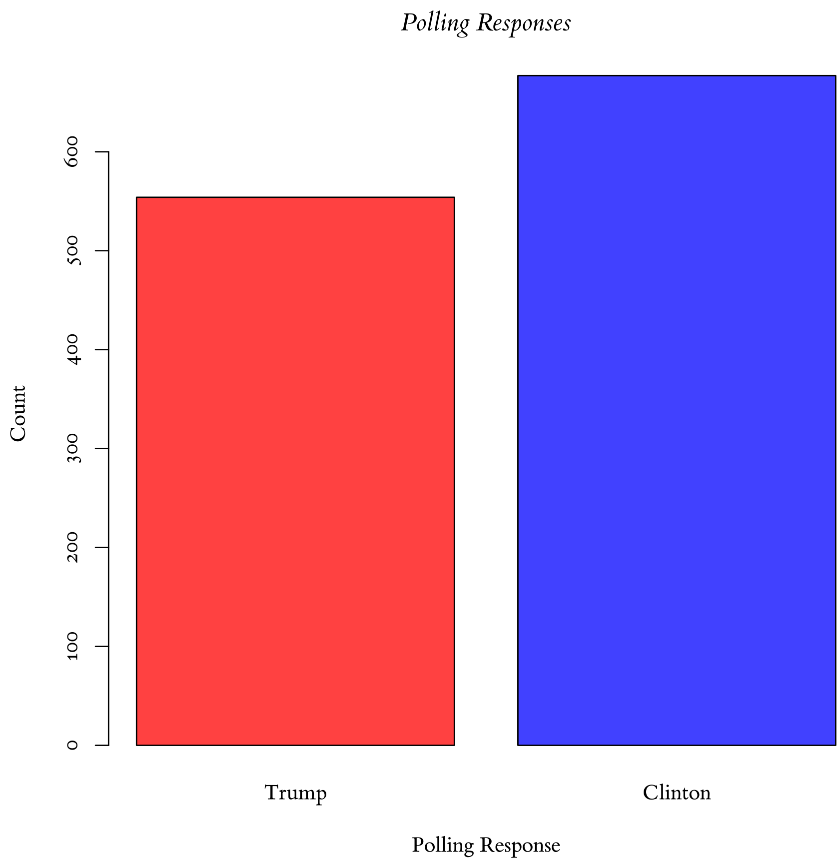
BAR PLOT *of polling data (using R's Base Graphics)*

```
poll.t <- table(dat$response)

kable(as.data.frame(poll.t), caption = "Frequency Table of Polling Data",
      col.names = c("Response", "Frequency"))
```

Table 4: Frequency Table of Polling Data

Response	Frequency
Trump	554
Clinton	677

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

palette(electpal)
barplot(poll.t, ylab = "Count",
        xlab = "Polling Response",
        family = "ETBembo",
        col = electpal, main = "Polling Responses")
```

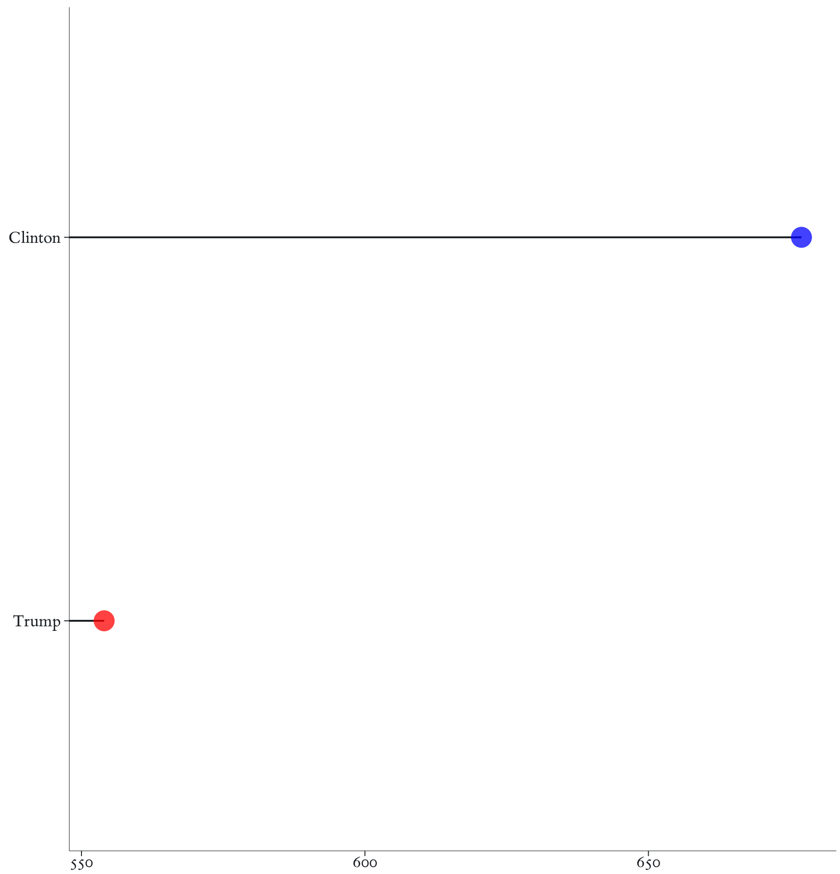


Dot Plot *of polling data using the `ggplot2` package.*

This would actually be interesting to do with repeated-measures data with the candidates on the `Y-axis` and time (in months/weeks) on the `X-axis`.

```
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]
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)
bpoll
```

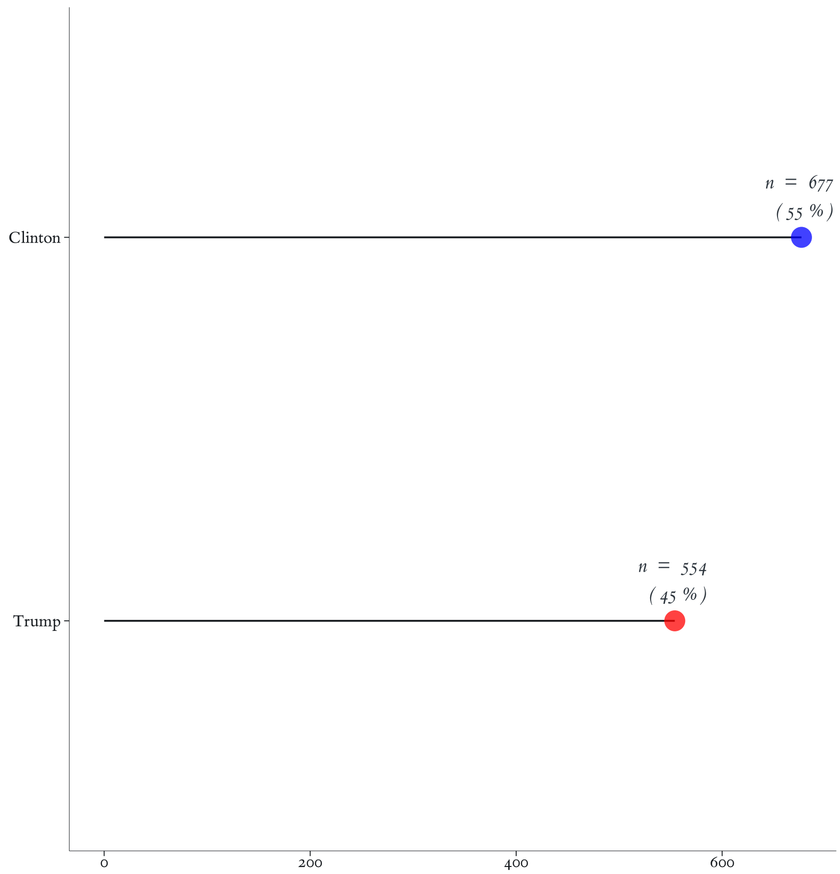


Don't forget to set the x-and-y-limits! Otherwise, you could be presenting a potentially misleading visualization of the data. Since these are polling data, there is a true "zero" such that `0` would reflect `0` votes for a given candidate in a given poll.[1] The data should thus be represented according to its appropriate scale limits.

[1] \textit{This has happened for one of the two current major party presidential candidates in the very recent past - I will not say who.}

```
bpoll + xlim(0, max( poll.df$Frequency)) +
    geom_text(vjust = -0.5, hjust = 0.5, stat = 'identity',
              position = 'identity', colour = mypal[19],
              size = rel(4), aes(family = "ETBembo", fontface = "italic",
                                 label = paste("n  = ", n,
                                               "\n", "(",
                                               round(n/ poll.df$N*100),
```

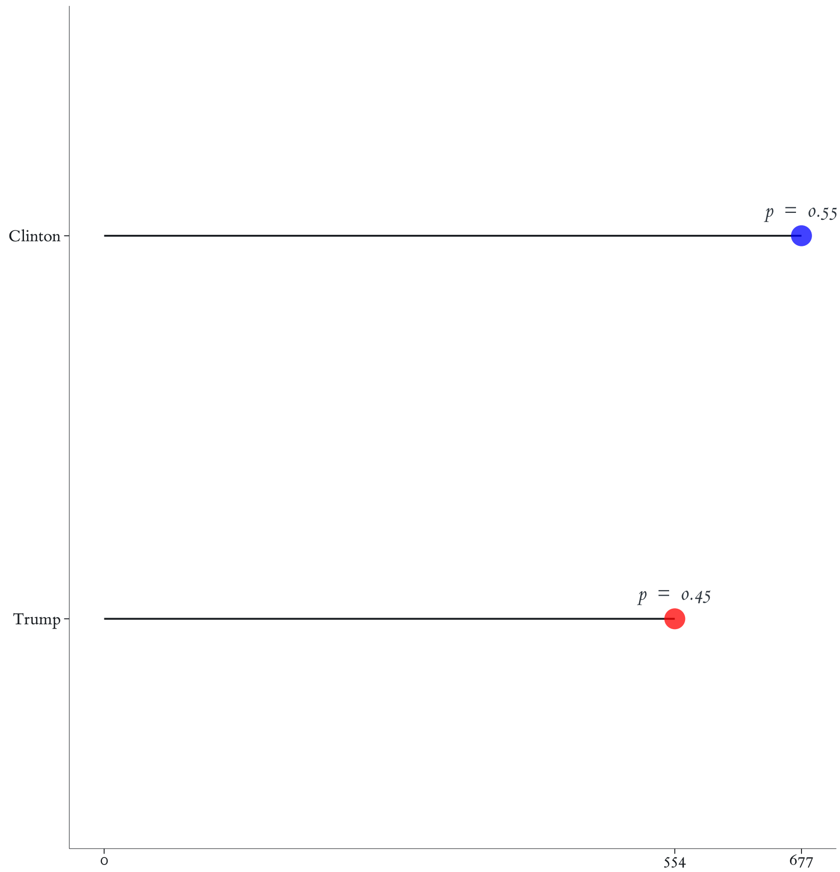
```
"%",")")))
```



Further, in plots like these, where discrete data with a relatively small number of categories[2] are juxtaposed with a continuous scale, setting the continuous axis' limits (in this case the `x-axis`) can help to further disambiguate the information.

[2] My persona rule of thumb for what constitutes a "relatively small category" is $N_{categories} \leq 5$

```r
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))))
```

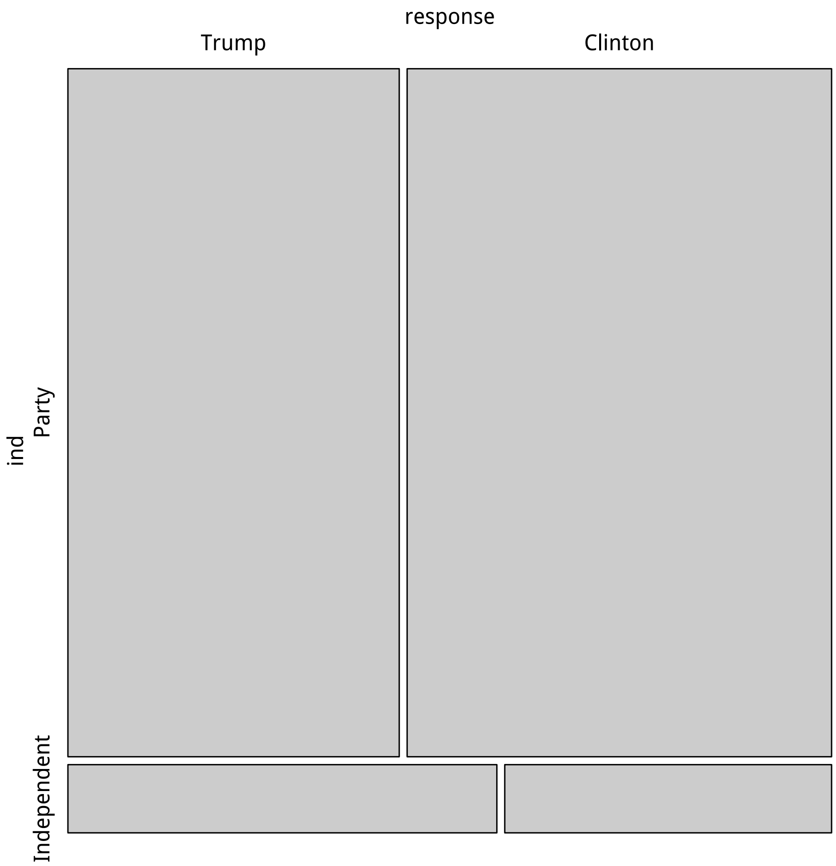


Mosaic Plots *for polling data*

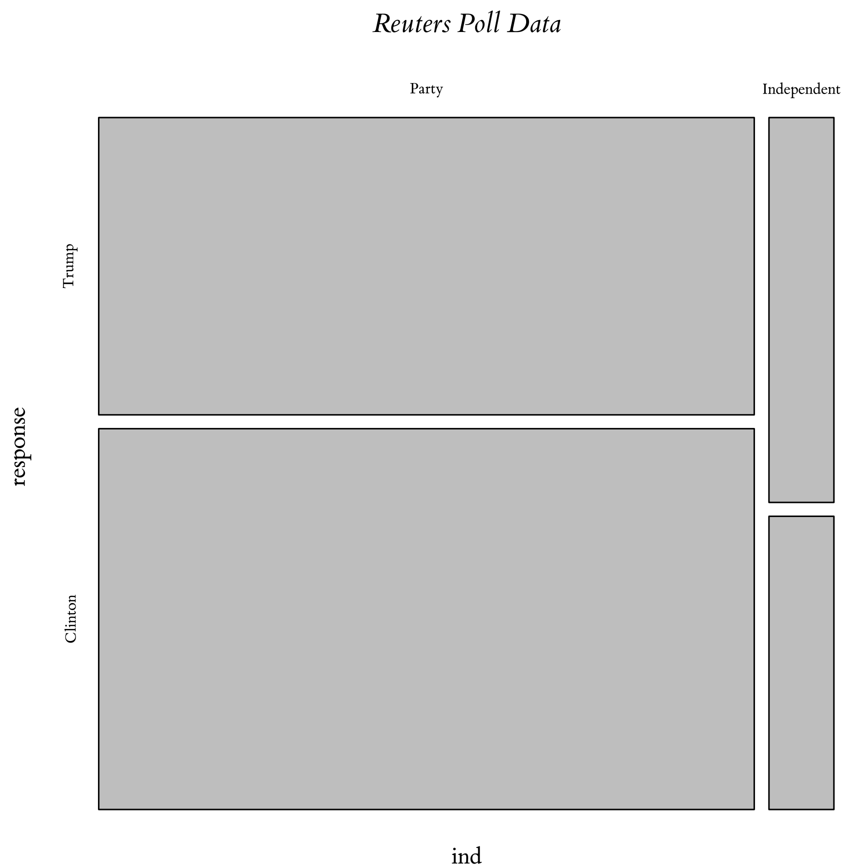
```
dat <- within(dat, {
    resp.F <- factor(response)
    ind.F <- factor(ind)
})

tbl <- table(dat$ind.F, dat$resp.F)
tbl
```

	Trump	Clinton
0	491	629
1	63	48

```
library(vcd)

dimnames(tbl) <- list(ind = c("Party", "Independent"),
                      response = c("Trump","Clinton"))
mosaic(tbl)
```




```
mosaicplot(tbl,main = "Reuters Poll Data")
```



Three-Way Mosaic Plot

```
dat <- R.rspss("data/cnnpoll.sav", vlabs = T)
ft <- with(dat, {
    ftable(dat, row.vars = 1:2, col.vars = 3)
})
ft
```

		"ind"	"party affiliate"	"independent"
"response"	"sex"			
"CLINTON"	"MALE"		100	106
	"FEMALE"		157	89
"TRUMP"	"MALE"		139	128
	"FEMALE"		140	77

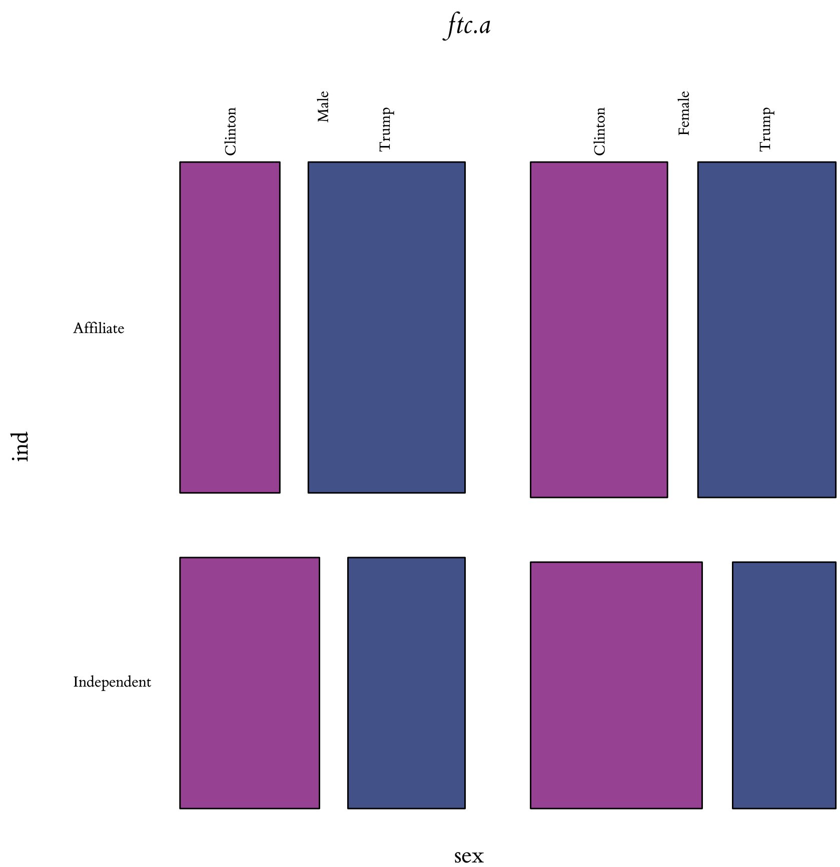
```
ftc <- matrix(ft, nrow = 4, byrow = T)
ftc
```

100	157
139	140
106	89
128	77

```
ftc.a <- array(ftc, dim = c(2, 2, 2), dimnames = list(
    sex = c("Male", "Female"),
    ind = c("Affiliate", "Independent"),
    response = c("Clinton", "Trump")))
ftc.a
```

100, 139, 106, 128, 157, 140, 89 and *77*

```
mosaicplot(ftc.a, type = "deviance", las = 2, color = mypal.a75[c(5, 16)])
```



Azen and Walker[3]'s (Chapter 3) Proficiency Data

[3] *Categorical Data Analysis for the Behavioral and Social Sciences*.

"See 'Chapter3, Figure4'"

> "suppose that federal guidelines state that 80%$ of students should demonstrate proficiency in mathematics, and in a randomly selected sample of 10 students only 70% of students were found to be proficient in mathematics. In such a case, we may wish to test whether the proportion of students who are proficient in mathematics in the population is significantly different than the federal guideline of 80%. In other words, we would like to know whether our obtained sample proportion of 0.7 is significantly lower than 0.8, so we would test the null hypothesis $H_0 : \pi = 0.8$ against the (one-sided, in this case) alternative $H_1 : \pi < 0.8$" (p. 23).
>
> > "Two variables are entered: the *proficiency level (`prof`)* and the *frequency (`count`)*" (p. 32).
> >
> > > "Note that if raw data for 10 individuals (i.e., 10 `rows of data`) were analyzed, where the variable called `prof` consisted of 7*yes responses* and 3*no responses*, the counts would be computed by the program and would not be needed as input" (p. 32).
>
> Azen and Walker,[4] (pp. 23 & 32)

[4] *Categorical Data Analysis for the Behavioral and Social Sciences*.

The `prof` variable should be read by `R` as a `string variable`.

```
library(foreign) ## read.spss() ##
dat <- read.spss("data/proficient.sav", to.data.frame = TRUE)
```

Data Inspection & Cleaning

```
str(dat)
```

'data.frame': 103978 obs. of 3 variables:
$ id : num 1 2 3 4 5 6 7 8 9 10 ...
$ response: num 1 1 1 1 1 1 1 1 1 1 ...
$ level : Factor w/ 4 levels "minimal","basic",..: 4 4 4 4 4 4 4 4 4 4 ...
- attr(, *"variable.labels")= Named chr*
..- *attr(*, "names")= chr
- attr(*, "codepage")= int 1252

```
    ## The proficient.sav dataset is HUGE!
dat <- tbl_df(dat)
    ## see the {dplyr} package - tbl_df's are just better
    ## (than dataframes), particularly when it comes to
    ## very large dataframes, such as these proficiency data.

glimpse(dat) ## {pkg:dplyr} ##
```

Observations: 103,978
Variables: 3
$ id 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...
$ response 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ level advanced, advanced, advanced, advanced, advance...

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

sapply(dat, Risna)
```

id	response	level
0	0	32269

It appears that there are several instances of `NA` in the `level` column. However, since I do not know these data too well just yet, I am apprehensive, to say the least, to just outright remove those rows containing `NA`s. Instead, I'm going to assign an arbituary, but meaningful to me, value to all instances of `NA` throughout the dataset.

The output from `sapply()` above indicates that the `level` column is the only one with `NA`s. So, in the next set of analyses, we should only see changes to that column in the end.

To ensure that I do not accidentally override an existing value for the `level` variable by assigning an arbituary value to `NA's`, I first call `unique(dat$level)` below to get a list of the numeric values currently assigned to the levels of `dat$level`. Then, I create a quick function

(`R.na()`) to replace `NA` values within a given `R-object`.

```
unique(dat$level)
```

advanced, *proficient*, *basic*, *minimal* and _ _Levels: minimal basic proficient advanced

```
levels(dat$level)
```

minimal, *basic*, *proficient* and *advanced*

```
lv <- c(levels(dat$level), "Unknown")
    ## I'm going to have to re-assign the factor levels to dat$level, so
    ## I am saving the labels, with an added level for the NAs, to use later

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)
}

dat$level <- sapply(dat$level, R.na, v = 0)
    ## 0 does not currently mean anything else in this data, so I am going
    ## to use to represent it's acutal meaning, which is simply a NULL value

dat$level <- factor(dat$level, levels = c(4, 3, 2, 1, 0), labels = lv)
unique(dat$level)
```

minimal, *basic*, *proficient*, *advanced* and _Unknown_Levels: minimal basic proficient advanced Unknown

```
levels(dat$level)
```

minimal, *basic*, *proficient*, *advanced* and *Unknown*

```
glimpse(dat)
```

Observations: 103,978
Variables: 3
$ id 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1. . .
$ response 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, . . .
$ level minimal, minimal, minimal, minimal, minimal, mi. . .

```
sapply(dat, R.isna)
```

id	response	level
0	0	0

Setting up for binomial analysis

Based on their description (and the fact that the example pertains primarily to the Bernoulli Distribution), it appears that Azen and Walker[5] used a dichotomized version of the `level` variable in their analysis example. However, I am curious about the level of extant intformation available from this variable's original 4-categories[6], as well as the difference, if any, between the two categorical data structuring methods.

[5] *Categorical Data Analysis for the Behavioral and Social Sciences*.

[6] i.e., minimal, basic, proficient, advanced, Unknown

```r
rec <- "c('proficient', 'advanced') = 1; c('Unknown', 'minimal', 'basic') = 0"
dat$lev.D <- car::recode(dat$level, rec)
    ## "D" = "Dichotomous" ##
summary(dat$lev.D)
```

0	1
83182	20796

Quick aside: Instead of the above dichotomization, I could have done the following:

```r
rec1 <- c("'minimal'=1; 'basic'=2; 'proficient'=3; 'advanced'=4; 'Unknown'=0")
dat$lev.D1 <- car:::recode(dat$level, rec1)
dat$lev.D1 <- as.integer(dat$lev.D1)
dat$lev.D1 <- cut(dat$lev.D1, breaks = 2, right = FALSE)
D1labs <- levels(dat$lev.D1)
levels(dat$lev.D1) <- c("0", "1")
summary(dat$lev.D1)
```

0	1
50913	53065

However, as you can see by the differences in the summary outputs above and the barplots below resulting from these different dichotomizing procedures, when I simply "cut()" the factor levels, I get a nicely split (i.e., "cut") factor. Unfortunately, the proportions of the newly dichotomized factor's levels do not necessarily reflect the actual proportions in the data.

Moral of the story - be careful when cutting and never assume that R knows what you are trying to do (R's purpose is not to Harry Huidini your data by reading your mind (or your prof's/advisor's mind), but rather to provide you with a set (to put it mildly) of tools to help you do the work ... but you still have to do the actual thinking work).

```
dat$level <- relevel(dat$level, ref = "Unknown")
    ## re-ordering levels for presentation ##
```

Below are the exact same plots, except with a random sample of $n = 10$ cases from the proficiency data per the example in Azen and Walker[7] (Chapter 3).

[7] *Categorical Data Analysis for the Behavioral and Social Sciences*.

```
dat.s <- sample_n(dat, 10)
```

References[8]

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`.

Azen, Razia, and Cindy M. Walker. *Categorical Data Analysis for the Behavioral and Social Sciences*. Kindle Edition. New York, NY, US: Routledge, 2011.

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`.

Meyer, David, Achim Zeileis, and Kurt Hornik. "Residual-Based Shadings for Visualizing (Conditional) Independence"." *Journal of Statistical Software* 17, no. 3 (2006): 1–48. `http://www.jstatsoft.org/v17/i03/`.

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."

[8] **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*, *vcd-v1.4.* Meyer, Zeileis, and Hornik, "Residual-Based Shadings for Visualizing (Conditional) Independence".", *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*

Journal of Statistical Software 40, no. 1 (2011): 1–29. http://www.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.