POLSCI 9590: Methods I

Probabilities

Dave Armstrong

Videos

We covered a few different things in the videos:

1. What are probabilities
2. Calculating probabilities of discrete events.
    - probabilities of unrelated events.
    - probabilities of related events.
    - probabilities of mutually exclusive events.
3. Continuous probabilities (didn't calculate these, yet).

Videos

We covered a few different things in the videos:

1. What are probabilities
2. Calculating probabilities of discrete events.
    - probabilities of unrelated events.
    - probabilities of related events.
    - probabilities of mutually exclusive events.
3. Continuous probabilities (didn't calculate these, yet).

Questions?

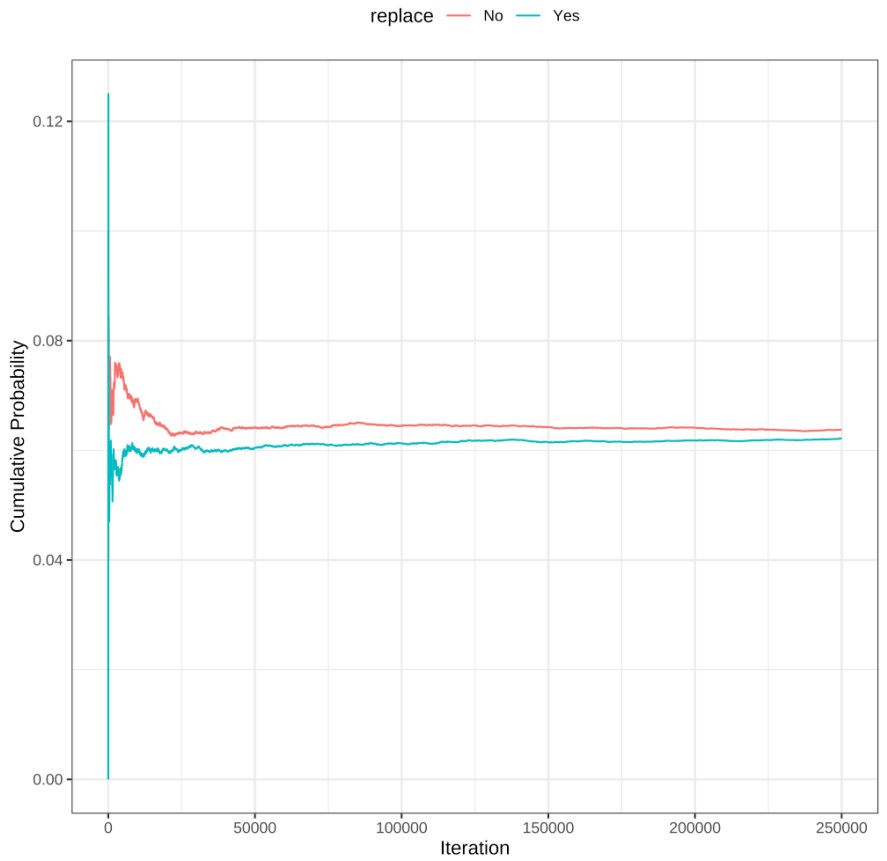
Probabilities

```
marbles <- function(draws,
                    colors=c("red" = 10, "blue" = 10, "yellow" = 10, "green" = 10),
                    replace=FALSE){
  # draws should be a character vector of required draws
  # colors defines the contents of the marble bag
  # replace indicates whether a drawn marble should be replaced in the bag
  ## make the bag
  inbag <- rep(names(colors), colors)
  ## initialize the draws
  out_draw <- NULL
  ## loop over the values in draws
  for(i in 1:length(draws)){
    ## sample 1 observation from the bag
    tmp_s <- sample(inbag, 1)
    ## record the sampled item in the output object
    out_draw <- c(out_draw, tmp_s)
    if(!replace){
      ## if no replacement, then remove the drawn value
      ## from the bag
      w <- min(which(inbag == tmp_s))
      inbag <- inbag[-w]
    }
  }
  ## return a vector indicating whether the
  ## draw matched all the conditions
  all(out_draw == draws)
}
```

Figure

```
reps1 <- replicate(250000,
                   marbles(c("red", "green"),
                      replace=TRUE)
                  )
reps2 <- replicate(250000,
                   marbles(c("red", "green"),
                           replace=FALSE)
                   )
tmp <- tibble(
  reps = c(reps1, reps2),
  it = rep(1:250000, 2),
  replace=rep(c("Yes", "No"), each=250000))
tmp <- tmp %>%
  group_by(replace) %>%
  mutate(cm = cummean(reps)) %>%
  filter(it %% 10 == 0)
```

```
ggplot(tmp, aes(x=it,
                y=cm,
                colour=replace))+
  geom_line() +
  theme_bw() +
  labs(x="Iteration",
       y="Cumulative Probability") +
  theme(legend.position="top")
```



Data Management

There isn't really a theoretical equivalent to the data management tools that will consume a lot of your time.

- **Recoding:** changing the values of your variable (often times collapsing or creating groups)
- **filtering:** finding only observations that meet some pre-defined condition.

There will be other tasks, too, that we will talk about later.

Setup

R | Python | Stata

```r
library(ggplot2)
library(dplyr)
library(rio)
library(scales)
```

Recoding

R | Python | Stata

Recoding can be done lots of ways, but the one we'll use is the `case_when()` function.

- This fits nicely into the `dplyr` world.
- It follows all of R's conventions.

The `case_when()` function takes a bunch of arguments of the form:

```
case_when(x,
  condition1 ~ value1,
  condition2 ~ value2,
  TRUE ~ value3)
```

Every observation that matches `condition#` will be given the value `value#`. All other observations that don't meet `condition1` or `condition2` are assigned `value3`.

Logical Operators

R | Python | Stata

- `==` equailty: `x==1` will be `TRUE` if and only if `x` is equal to 1.
- `&` conjunction (and): `x==1 & y==2` will be `TRUE` if and only if both conditions hold.
- `|` disjunction (or): `x==1 | y==2` will be `TRUE` if either or both of the conditions are met.
- `!` - negation: turns `TRUE` into `FALSE` and vice versa. (e.g., `x != 4` is `FALSE` when `x` is 4)
- `%in%` element: `x %in% c(1,4,6)` will be `TRUE` if `x` is any of 1, 4 or 6.
- `<` less than: `x < 1` will be `TRUE` for all numbers up to, but not including 1. `x <= 1` will be `TRUE` for all numbers up to *and including* 1. `>` works similarly.

Examples of Recoding

R Python Stata

```
dat = data.frame(x = c(1,2,3,4))
dat <- dat%>%
  mutate(y = case_when(x <=3 ~ "yes",
                       TRUE ~ "no"))
dat
```

```
##   x   y
## 1 1 yes
## 2 2 yes
## 3 3 yes
## 4 4  no
```

```
dat <- dat %>%
  mutate(y = case_when(x %in% c(1,2,3) ~ "yes",
                       TRUE ~ "no"))
dat
```

```
##   x   y
## 1 1 yes
## 2 2 yes
## 3 3 yes
## 4 4  no
```

Missing Values

R | Python | Stata

The `TRUE ~` arguments turns *everything else* including `NA` to the indicated value.

```
dat <- data.frame(x = c(1,2,3,4, NA))
dat <- dat %>%
  mutate(y = case_when(x <=3 ~ "yes",
                TRUE ~ "no"))
dat
```

```
##    x   y
## 1  1 yes
## 2  2 yes
## 3  3 yes
## 4  4  no
## 5 NA  no
```

```
dat <- dat %>%
  mutate(y = case_when(x <=3 ~ "yes",
                is.na(x) ~ NA_character_,
                TRUE ~ "no"))
dat
```

```
##    x    y
## 1  1  yes
## 2  2  yes
## 3  3  yes
## 4  4   no
## 5 NA <NA>
```

If the variable you're making (`y` in this case) isn't a character string, but a number, you could use `NA_real_` in place of `NA_character_`.

Multiple Rules

R Python Stata

It's worth seeing what happens when we have multiple rules for a single value:

```
dat <- data.frame(x=1:4)
dat <- dat %>%
  mutate(y = case_when(x <= 2 ~ "l",
                       x >= 2 ~ "g"),
         z = case_when(x >= 2 ~ "g",
               x <= 2 ~ "l"))
dat
```

```
##   x y z
## 1 1 l l
## 2 2 l g
## 3 3 g g
## 4 4 g g
```

The first rule takes precedence.

CES example

Let's use the skills we learned last week to make a new variable `market_01` which is 0 if `market` is lower than its 40^{th} percentile and 1 otherwise (missings should stay missing).

R | Python | Stata

```
ces <- import("ces19.dta")
q40 <- quantile(ces$market, .4, na.rm=TRUE)
ces <- ces %>%
  mutate(market_01 = case_when(
    market < q40 ~ 0,
    is.na(market) ~ NA_real_,
    TRUE ~ 1
  ))
table(ces$market_01, useNA="ifany")
```

```
##
##    0    1 <NA>
## 1049 1707   43
```

Recode Example

Let's find people who like the NDP leader more than the other two.

R | Python | Stata

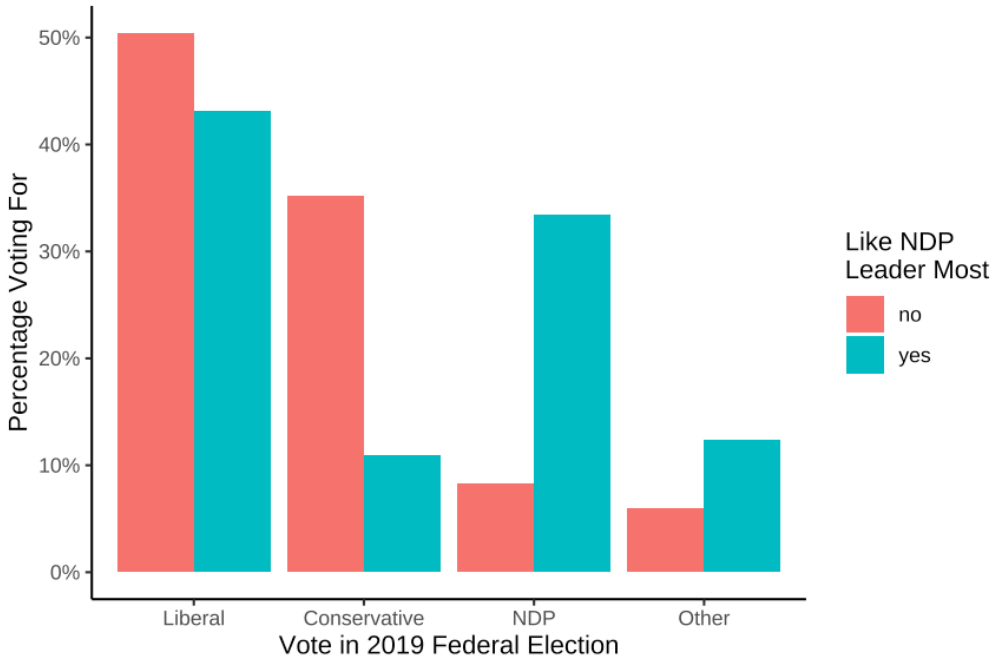
```
ces <- ces %>%
  mutate(heart_ndp = case_when(
    leader_lib < leader_ndp & leader_con < leader_ndp ~ "yes",
    is.na(leader_lib) | is.na(leader_con) | is.na(leader_ndp) ~ NA_character_,
    TRUE ~ "no"))
table(ces$heart_ndp, useNA="ifany")
```

```
##
##   no  yes <NA>
## 2030  744   25
```

Vote by heart_ndp

R Python Stata

```
ces %>%
  mutate(vote = factorize(vote)) %>%
  group_by(vote, heart_ndp) %>%
  summarise(n = n()) %>%
  na.omit() %>%
  group_by(heart_ndp) %>%
  mutate(pct = n/sum(n)) %>%
  ggplot(aes(x=vote, y=pct, fill=heart_ndp)) +
    geom_bar(stat="identity",
             position=position_dodge()) +
    theme_classic() +
  scale_y_continuous(label=percent) +
  labs(x="Vote in 2019 Federal Election",
       y="Percentage Voting For",
       fill="Like NDP\nLeader Most")
```



Exercise 1

Question: What is the difference between people who love and hate Trudeau?

1. Make a new variable that is coded `"love"` for observations where `leader_lib` is greater than or equal to 90 and `"hate"` for observations where `leader_lib` is less than or equal to 10. All other observations should be missing.
2. Create the distribution of `educ`, `agegrp`, `market` and `relig` for these two groups.

Exercise 2

Question: What does the distribution of mental health look like for three groups of resilience.

1. Import the `gss16_can.dta` data set.
2. Use the `case_when()` function to create three groups of resilience using the $33rd$ and 67^{th} percentiles as the cutoffs.
3. Make a graph of `SRH_115` using this new resilience measure as the `facet` variable.