EpiR

grienne

February 22, 2019

Welcome to RMarkdown Documents!

RMarkdown documents are how many people produce files that are shareable for efficient collaborative work!

Please note the annotations before the text by the *'#'* symbol.

Quick Note: Running Code

1. Highlight the code then click ctrl+enter
2. There is a green forward arrow in the upper right of these highlighted areas with code in them (these are called chunks btw!). Click that arrow and it will run all the code in sequential order!

Packages

The code below checks your R to make sure you have the packages needed and if you don't it downloads and installs them!

IMPORTANT!

In this template I am using a data set in the *MASS* package. However, you will **NOT** need this package. When you are using this code, **DELETE** that package!

Your packages statement should look like this:

packages = c("epiR", "tidyverse", "survival", "readr")

```
# specify the packages of interest
packages = c("epiR", "tidyverse", "survival", "readr", "MASS")

# use this function to check if each package is on the local
# machine if a package is installed, it will be loaded if any
# are not, the missing package(s) will be installed and
# loaded
package.check <- lapply(packages, FUN = function(x) {
    if (!require(x, character.only = TRUE)) {
        install.packages(x, dependencies = TRUE)
        library(x, character.only = TRUE)
    }
})
```

```
## Warning: package 'epiR' was built under R version 3.5.2
```

Loading Data

Loading Data in R can be intensive, but luckily RStudio makes this fairly easy!

The link below is a website that will guide you through that process in RStudio.

Loading Data: [link]https://support.rstudio.com/hc/en-us/articles/218611977-Importing-Data-with-RStudio

This is a sample dataset that is located in the MASS Package. You will be working with ARIC. Some additional code will be provided for you at a later date

When you load a data file, I encourage you to rename it to something clear and short! The code below does 2 things: 1. I take the loaded dataset (named birthwt) and rename it to dat1 'dat1 <- birthwt' *Note: You can rename it however you want* 2. I take a quick look at the dataset to make sure it looks okay 'head(dat1)'

```
dat1 <- birthwt
head(dat1)
```

```
##    low age lwt race smoke ptl ht ui ftv  bwt
## 85   0  19 182    2     0   0  0  1   0 2523
## 86   0  33 155    3     0   0  0  0   3 2551
## 87   0  20 105    1     1   0  0  0   1 2557
## 88   0  21 108    1     1   0  0  1   2 2594
## 89   0  18 107    1     1   0  0  1   0 2600
## 91   0  21 124    3     0   0  0  0   0 2622
```

```
# R is case sensitive, typically it is good practice to make
# everything lowercase so you don't get errors because of
# case heterogeneity.
names(dat1) <- tolower(names(dat1))
```

—DATA MANAGEMENT Part 1 — Subsetting Data

Subsetting Data

There are a lot of ways to subset data!

*The **detach** command removes the MASS package so I can use the **select** command. You will not need the detach command code line as you will not have MASS loaded for your work. However, if you want to mess around with this template using the birthweight data set then leave this code as is*

```
# IMPORTANT! Delete the detach code line if you do not need
# it!

detach("package:MASS", unload = TRUE)


# Subsetting Tips I suggest creating a dataset when
# subsetting, sometimes you make unfixable or annoying
# mistakes, it is easier at times to go back to your original
# dataset and try again.

#------------------------------------------------------------

### This code works by taking the original data (dat1) then
### selecting the columns you want instead to create another
### dataset. In the example below I wanted the following
### variables: low, lwt, and race

dat2 <- dat1 %>% select(low, lwt, race)
```

—DATA MANAGEMENT PART 2— Variable Management

Variable Attribute Control

We need to talk about variable attributes. Every statistical software and database program manages data by controlling managing variable "attributes."

Think of using the Excel software. If you want to change the type of data or format of the data you can 'format cells' to change the column data type to dates, integers, text, etc.

R controls data by identifying variable attributes. Attribute control is crucial to effective code-writing.

For this class you need to know that categorical variables should be *factors* & continuous data should be *integers*

```
# The str command produces an output that tells you the
# attribute of a variable. 'Int' means integer, Facotr means
# factor When you import data, you may see 'str' or 'chr',
# this means string and character respectively.

str(dat1)
```

```
## 'data.frame':    189 obs. of  10 variables:
##  $ low  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ age  : int  19 33 20 21 18 21 22 17 29 26 ...
##  $ lwt  : int  182 155 105 108 107 124 118 103 123 113 ...
##  $ race : int  2 3 1 1 1 3 1 3 1 1 ...
##  $ smoke: int  0 0 1 1 1 0 0 0 1 1 ...
##  $ ptl  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ht   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ui   : int  1 0 0 1 1 0 0 0 0 0 ...
##  $ ftv  : int  0 3 1 2 0 0 1 1 1 0 ...
##  $ bwt  : int  2523 2551 2557 2594 2600 2622 2637 2637 2663 2665 ...
```

```
# Turn the variables for 2x2 contingency tables into factors

dat1$low <- as.factor(dat1$low)
dat1$smoke <- as.factor(dat1$smoke)
dat1$race <- as.factor(dat1$race)

# check to make sure you changed it!
str(dat1)
```

```
## 'data.frame':    189 obs. of  10 variables:
##  $ low  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ age  : int  19 33 20 21 18 21 22 17 29 26 ...
##  $ lwt  : int  182 155 105 108 107 124 118 103 123 113 ...
##  $ race : Factor w/ 3 levels "1","2","3": 2 3 1 1 1 3 1 3 1 1 ...
##  $ smoke: Factor w/ 2 levels "0","1": 1 1 2 2 2 1 1 1 2 2 ...
##  $ ptl  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ht   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ui   : int  1 0 0 1 1 0 0 0 0 0 ...
##  $ ftv  : int  0 3 1 2 0 0 1 1 1 0 ...
##  $ bwt  : int  2523 2551 2557 2594 2600 2622 2637 2637 2663 2665 ...
```

Renaming Data

I don't like working with 1's and 0's if the data is categorical. R doesn't really care as long as the variable attribute is correct, but often to label my data to keep my code output simple and easily readable!

This might look intimidating but I promise it's fairly simple!

This uses an *'ifelse'* statement to recode the numbers (now factors) into words. We will break this code down to the components

Note: *R really cares about the parentheses so make sure every parentheses has it's partner!*

```
# 1.'dat1$race <- tells R which dataset and which column
# (race) you are working on.  with(dat1, --- this is the
# beginning of the code for changing, it must be ordered like
# this!

# 2. if_else(column name == outcome, 'What you want it to be
# if true', 'what you want it to be if false') Now, I am
# changing multiple outcome possibilities so it says: If
# column 'race' outcome equals 1, that means 'black', but if
# it equals 2....etc The last one 'Other' is the 'what...if
# false' statement.  YOU ALWAYS NEED something here, so just
# keep in mind what you want your column to say if the other
# values arent there, here I knew that if it wasn't 1-3 the
# data should be coded as 'other'

dat1$race <- with(dat1, if_else(race == 1, "Black", ifelse(race ==
    2, "White", ifelse(race == 3, "Non-White Hispanic", "Other"))))
```

Filtering Data

Often times you will want to filter your data! Data filtering can be easy or complex, it really depends on your needs, for this course it should be pretty easy!

I am going to show you how to filter categorical data & continuous data; there are lots of different ways to do this. R uses basic LOGICAL OPERATORS just like excel for the most part!

I am going to take the dat2 we created above and use that for my example!

There are many logical operators, I present a few below, but you may need others [link]https://www.datamentor.io/r-programming/operator/

Note: *As a personal preference, when filtering I usually don't create new datasets, but feel free to create new ones if it is easier for you, just change the name on the left side of the code*

```
# Same if_else pattern as above I am turning the race
# category so that it is easier for me to filter. This is
# really just preference, I do not like working with 1's and
# 0's.
dat2$race <- with(dat2, if_else(race == 1, "Black", ifelse(race ==
    2, "White", ifelse(race == 3, "Non-White Hispanic", "Other"))))

#-----------------------------------

dat2 <- dat2 %>% # categorical variables must be bounded by the quotation
# marks
```

```
filter(race == "Black") %>%
# Continuous variables do not need 'quotation' marks, just
# basic mathematical operators Below I said all lwt greater
# than 107

filter(lwt > 107) %>%
# Here I tell R to select lwt output that is greater
# than/equal to 130, but less/equal to 200

filter(lwt >= 130 & lwt <= 200)

# Note How I did 2 filters at once! This is done by PIPING! I
# won't go into it, but basically at the end of the first
# filter, I did another %>%, that lets R know to run another
# process. To end a pipe just don't add a %>% at the end!

# Tip: I could actually have did the selection and filtering
# all at the same time using this process when I am working,
# I tend to do that, but I encourage you instead to split
# these things up.  Until you're used to it, it makes it
# easier to identify possible problems and remember what you
# were doing
```

Descriptive Statistics

```
# The quick descriptive statistics analysis
summary(dat1)
```

```
##  low          age              lwt             race           smoke
##  0:130   Min.   :14.00   Min.   : 80.0   Length:189         0:115
##  1: 59   1st Qu.:19.00   1st Qu.:110.0   Class :character   1: 74
##          Median :23.00   Median :121.0   Mode  :character
##          Mean   :23.24   Mean   :129.8
##          3rd Qu.:26.00   3rd Qu.:140.0
##          Max.   :45.00   Max.   :250.0
##       ptl               ht                ui              ftv
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.0000   Median :0.00000   Median :0.0000   Median :0.0000
##  Mean   :0.1958   Mean   :0.06349   Mean   :0.1481   Mean   :0.7937
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:1.0000
##  Max.   :3.0000   Max.   :1.00000   Max.   :1.0000   Max.   :6.0000
##       bwt
##  Min.   : 709
##  1st Qu.:2414
##  Median :2977
##  Mean   :2945
##  3rd Qu.:3487
##  Max.   :4990
```

```
# If you want to do a summary for only one variable
```

```r
summary(dat1$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   14.00   19.00   23.00   23.24   26.00   45.00
```

```r
# When you use the '$' sign in front of the named data set, a
# box will appear with your variables, you can select it
# there or just continue typing the variable name
```

Generate Contingency Tables

Order of the code input matters! So I suggest identifying your outcome/exposure variables first. This way you know exactly where to put them in the code below!

Note: *Some of the output from these tables won't be used until later in the course! But it all generates so just ignore what you don't need for now*

—Ratios—

Calculations! Finally, it's time to do some actual stats!

Note: *The command:* ***conf.level =****, The default in R is 95% and this is what you will assume for this course. However, this can be changed to whatever you want*

IMPORTANT!: When you finish the epi.2by2 command, you will see output in the console box. However, under the chunk RMarkdown will produce 2 selectable frames. A data.frame box and an R Console Box. The data.frame has the incidence risk, odds ratios, and the incidence rate (when appropriate)

```r
## Generate the 2 by 2 table. Exposure (rows) = smoke. Outcome
## (columns) = low.

## Note order of variables; DNN command places titles IN THE
## FINAL OUTPUT
tab1 <- table(dat1$smoke, dat1$low, dnn = c("Smoke", "Low BW"))

# Print, tells R to actually show the table it made!
print(tab1)
```

```
##       Low BW
## Smoke  0  1
##     0 86 29
##     1 44 30
```

```r
#-------------------------------------------

# Units = 100, tells R in what unit count, so this is per 100
# of whatever your primary unit is
t1r <- epi.2by2(dat = tab1, method = "cohort.count", conf.level = 0.95,
    units = 100, homogeneity = "breslow.day", outcome = "as.columns")


print(t1r$res$OR.strata.cfield)
```

```
##        est    lower    upper
## 1 2.021944 1.073694 3.794586
```

```
# The outcomes from this table with the 95% confidence
# intervals Incident Risk Ratio Odds Ratio Attributable Risk
# Attr Risk in Pop Attrib. Fraction in exposed Attrib.
# Fraction in Pop.

#-----------------------------------------------------------------

# Sample Interpretation Odds ratio: The odds of having a low
# birth weight child for smokers is 2.02 (95% CI 1.08 to
# 3.78) times greater than the odds of having a low birth
# weight child for non-smokers.
```

—Stratification—

The code below stratifies by race.

Note: *Note the output order of* print(tab2)* when you generate the OR per strata the output will be in the same order*

```
## Now stratify by race: Note added a factor in the code. This
## is telling R to stratify by Race Remember, order matters!!


tab2 <- table(dat1$smoke, dat1$low, dat1$race, dnn = c("Smoke",
    "Low BW", "Race"))
print(tab2)
```

```
## , , Race = Black
##
##      Low BW
## Smoke  0  1
##     0 40  4
##     1 33 19
##
## , , Race = Non-White Hispanic
##
##      Low BW
## Smoke  0  1
##     0 35 20
##     1  7  5
##
## , , Race = White
##
##      Low BW
## Smoke  0  1
##     0 11  5
##     1  4  6
```

```
## Compute the crude odds ratio, the Mantel-Haenszel adjusted
## odds ratio and other measures of association:
rval <- epi.2by2(dat = tab2, method = "case.control", conf.level = 0.95,
    units = 100, homogeneity = "breslow.day", outcome = "as.columns")
```

```
print(rval)
```

```
##              Outcome +    Outcome -      Total        Prevalence *
## Exposed +           86           29        115                74.8
## Exposed -           44           30         74                59.5
## Total              130           59        189                68.8
##                  Odds
## Exposed +        2.97
## Exposed -        1.47
## Total            2.20
##
## Point estimates and 95 % CIs:
## -------------------------------------------------------------------
## Odds ratio (crude)                            2.02 (1.08, 3.78)
## Odds ratio (M-H)                              3.09 (1.49, 6.39)
## Odds ratio (crude:M-H)                        0.66
## Attrib prevalence (crude) *                   15.32 (1.61, 29.04)
## Attrib prevalence (M-H) *                     22.17 (2.55, 41.79)
## Attrib prevalence (crude:M-H)                 0.69
## -------------------------------------------------------------------
##  Test of homogeneity of  OR: X2 test statistic: 3.126 p-value: 0.21
##  Wald confidence limits
##  M-H: Mantel-Haenszel
##  * Outcomes per 100 population units
```

```
# Generate The OR per strata with confidence intervals
rval$res$OR.strata.wald
```

```
##        est     lower     upper
## 1 5.757576 1.7823209 18.599164
## 2 1.250000 0.3502120  4.461584
## 3 3.300000 0.6346062 17.160249
```

```
## Sample Interpretation: After accounting for the confounding
## effect of race, the odds of having a low birth weight child
## for smokers is 3.09 (95% CI 1.49 to 6.39) times that of
## non-smokers.
```

—Rates—

```
# Now we will compute INCIDENCE RATES


# Essentially you need 2 columns; cases and the person-time
# (c7_futime) In the ARIC data set both of these columns are
# provided As you continue on, you will learn to identify
# those columns!

dat <- as.table(matrix(c(136, 22050, 1709, 127650), nrow = 2,
    byrow = TRUE))
```

```r
# note structure: first column is cases, second is person
# time While this data set is created, remember as above, you
# will create a tab, and order matters!
print(dat)
```

```
##        A      B
## A    136  22050
## B   1709 127650
```

```r
rval <- epi.2by2(dat = dat, method = "cohort.time", conf.level = 0.95,
    units = 1000, homogeneity = "breslow.day", outcome = "as.columns")

# The outcomes from this table with the 95% confidence
# intervals Incident Rate Ratio Attrib Rate Attrib Rate in
# Population Attr Frac in Exposed(%) Attrib. Fraction in Pop.

print(rval)
```

```
##              Outcome +    Time at risk        Inc rate *
## Exposed +          136           22050             6.17
## Exposed -         1709          127650            13.39
## Total             1845          149700            12.32
##
## Point estimates and 95 % CIs:
## -------------------------------------------------------------------
## Inc rate ratio                                0.46 (0.38, 0.55)
## Attrib rate *                                 -7.22 (-8.44, -6.00)
## Attrib rate in population *                   -1.06 (-1.91, -0.22)
## Attrib fraction in exposed (%)                -117.07 (-160.42, -82.20)
## Attrib fraction in population (%)             -8.63 (-8.82, -8.44)
## -------------------------------------------------------------------
##  X2 test statistic: 78.963 p-value: < 0.001
##  Wald confidence limits
##  * Outcomes per 1000 units of population time at risk
```

```r
# Incidence Rate The code above generates all of the
# information needed The next code line pulls out the
# incidence rate specifically
summary(rval)$ARate.strata.wald
```

```
##        est     lower     upper
## 1 -7.22037 -8.435865 -6.004875
```

```r
## Interpretation The incidence rate of cancer was 7.22 cases
## per 1000 person-years less in the blind, compared with
## those who were not blind but had severe visual impairment
## (90% CI 6.00 to 8.43 cases per 1000 person-years).


#--------------------


# The code below does a rate ratio of the two groups.
round(summary(rval)$IRR.strata.wald, digits = 2)
```

```
##    est lower upper
```

```
## 1 0.46  0.38  0.55
```

```
# Interpretation The incidence rate of cancer in the blind
# group was less than half that of the comparison group
# (incidence rate ratio 0.46, 90% CI 0.38 to 0.55).
```

—Complete Rows—

Sometimes you need to evaluate if your data is complete. R has a useful way of telling you how many of your observations are missing values (NA)

```
# I create an object that evaluates the number of complete
# cases in a data set
ok <- complete.cases(dat1)

# I sum the number of incomplete observations
sum(!ok)
```

```
## [1] 0
```