An outbreak of Campylobacter jejuni in Greece

2021 version

**Contributors to *R* code:**  
Daniel Gardiner (PHE) and Lukas Richter (AGES)

**revisions by:** Ashley Sharp (PHE), Hikaru Bolt (PHE), Patrick Keating, Alexander Spina

and Achim Dörre (RKI)

The following code has been adapted to *R* for learning purposes. The initial contributors are listed below. All copyrights and licenses of the original document apply here as well.

**Authors:**  
Ioannis Karagiannis

**Revisions**  
**December 2011:** Major expansion of background and rationale; addition of preliminary questions; addition of explanation of variables; added help for tasks of descriptive analysis; expansion of the help in the univariable analysis; major expansion of the help provided for the stratified analysis  
**November 2012:** Breakdown of background to more questions to facilitate learning; addition of a table and a map for attack rates by municipality and by age group; renamed variable “gender” to “sex” to indicate biological sex; added IDs to dataset; creation and addition of variable “well” to teach confounding; minor changes in the phrasing of the tasks throughout; addition of two-by-two tables for univariate analysis.  
**November 2013:** Minor clarifications in the background; change of the wording from “case-cohort” to “case-control” throughout; clarifications in the help provided throughout.  
**December 2015:** Minor clarifications in the background; addition of expected learning outcomes; addition of loops in Stata; addition of an information bubble on user-written commands; minor stylistic improvements throughout.  
**December 2016:** Removal of three two-by-two tables; addition of answers on the presence of effect modification/confounding in Table 6; correction of typos.

**December 2021:** minor revision (typos, explanations, remarks)

# An introduction to the R companion

“To understand computations in R, two slogans are helpful:

* Everything that exists is an object.
* Everything that happens is a function call.

John Chambers

If you look at the Global Environment panel (by default in the upper right of the screen) you will see a list of objects stored in that environment. When you load your data in R you create an object. This is completely separate from the data file itself (the excel file, or csv file etc). You can create as many objects as you like, for example you could store a few variables from your original data as a new object, or create a summary table and store that.

Functions in R are equivalent to commands in STATA. All functions in R take the form of a name followed by brackets, e.g. functionname(). Inside the brackets go various arguments. You can access the help file for a function by calling ?functionname. The help file will show which arguments the function takes and what the function does. Arguments have a default order, as specified in the help file, though you can override this by specifying which argument you are entering using the equals sign “=”, e.g. my\_function(x=5, y=matrix(0, 2, 2)).

A good reference for R users is the book R for Data Science by Garrett Grolemund and Hadley Wickham, which is available free online at <https://r4ds.had.co.nz/>.

Another great reference is The Epidemiologist R Handbook, which is available for free at <https://epirhandbook.com/en/>.

### RStudio projects

The easiest way to work with R is using RStudio ‘projects’. RStudio is a graphical user interface that runs R in the background. A ‘project’ is an RStudio file that saves your workspace so you can easily pick up from where you left off. Put all the files that you will need for this case study in a folder called ‘Copenhagen’ and create a project in the same folder by clicking file -> new project -> existing directory, and choosing the folder. For simplicity, make sure there are no subfolders in this folder, and put all data and scripts in the main Copenhagen folder.

### Setting your working directory

You can set a folder to be your working directory (using the setwd() command). Open the project that you’ve created and you will see that the working directory is the same as folder itself: you can check this by calling getwd().You can see what’s in your working directory by looking at the **Files tab** (by default in the bottom right area of the screen). If you want to set your working directory you use the function setwd(“C:/Users/yourname/Desktop/Campy”), but make sure that you put the correct folder path inside this function and use quotation marks. Note that R paths use forward slashes “/”, while windows paths use back slashes “\”, so if you copy a path from windows you have to change them manually.

getwd()

### Installing packages and functions

R packages are bundles of functions which extend the capability of R. Thousands of add-on packages are available in the main online repository (known as CRAN) and many more packages in development can be found on GitHub. They may be installed and updated using RStudio and via the internet.

Many packages come ready installed with R (base code), and so for many steps additional packages are not necessary. However, in many situations life becomes easier when using certain packages (such as *ggplot2* or *epiR*). In addition, we have included a few extra functions to simplify the code required. Furthermore, note that in the following, mainly base code is used including *for* loops. They are a means to automate and simplify steps in your script. An alternative is the syntax provided by the *tidyverse* package (e.g. >%> *pipes* and the *map* function, as considered during the OIM week). Feel free to think of tidyverse solutions to complement the following suggested solution.

Run the following code at the beginning to make sure that you have made available all the packages and functions that you need. Be sure to include it in any script, too.

# Loading required packages for the week  
library(ggplot2)  
library(foreign)  
library(Hmisc)  
library(epiR)  
library(knitr)

Note: This code assumes the packages are already installed. You only need to install a package once (and then load it whenever the package is needed). If this code does not work, try installing the packages using the code below.

install.packages("package name in quotation marks")

R and Stata have minor differences in default settings and methods. In this document we will follow the Stata analysis as closely as possible, but small and usually unimportant differences may be noted between the statistical findings in R and those in Stata. At some points additional steps (which would usually be optional in R) will be taken to produce output which is comparable to that of Stata.

# Getting started

## Functions required in this session

There are two functions you will need to use in this session. You can use them once you have set the working directory. The **epicurve** function allows creation of easily formatted epicurves. To find out more about the function, first load it as above and then click on function in the **Global Environment** tab on the right of the R Studio window. The **single variable analysis** function allows convenient calculation of attack rates of multiple variables at one time and provides similar output to the cctable and cstable commands in Stata. Please note that everything that is done by the **single variable analysis** function, could also be done directly. The advantage of using this function is primarily that the individual steps are stored in that script and whenever we want to perform an analysis of this type, we just need to type sva(…) with the respective arguments/inputs.

#These scripts need to be present in your working directory  
  
# Adds a function to create epicurves  
source("epicurve.v.1.8.R")   
  
# Adds a function to create output similar to cctable or cstable in Stata  
source("single.variable.analysis.v0.2.R")

## Reading in your dataset

You will work with Stata.dta data sets, which can be loaded into R with the “foreign” or “readstata13” packages. You can read in the Stata dataset to R using the foreign package and its read.dta function.

campy <- read.dta("campy.dta", convert.factors = FALSE)

## Browsing your dataset

*RStudio* has the nice feature that everything is in one window, so you can browse your dataset and your code without having to switch between windows.

# to browse your data, use the View command  
View(campy)

Alternatively, you can also view your dataset by clicking on **campy** in the top right **global environment** panel in *RStudio*. Your global environment is where you can see all the datasets, functions and other things you have loaded in the current session.

# Analytical epidemiology

## Task 2. How many observations does your dataset have? How many cases and how many controls does it contain?

### Help, Task 2

You can browse the dataset in order to see what it looks like and how many variables it includes. An indirect way of looking how many observations your dataset contains is the **table** command. You can use it for a single variable along with the option **useNA = “always”** to make sure that the missing values are also displayed in the output.

# View data set  
View(campy)  
  
# Assess the distribution of a single variable using the table function  
table(campy$datesym, useNA = "always")  
  
# Compute number of controls and cases  
table(campy$case, useNA = "always")

## Task 3. Explore each of the variables. What information do they contain? Are they labelled? Are they categorical or continuous variables?

### Help, Task 3

### Describing your dataset

You can view the structure of your data set using the following commands. Each of these commands can be run for individual variables, too. You can refer to an individual variable of a data set by using the **$**, for example, if you wanted to obtain a summary of the age variable, then you would write **summary(campy$age)**.

# str provides an overview of the number of observations and variable types  
str(campy)  
  
# summary provides mean, median and max values of your variables  
summary(campy)  
  
# summary of age  
summary(campy$age)  
  
# describe (from Hmisc package) provides no. of observations, missing values, unique levels of each variable  
describe(campy)

## Task 4. Can you think of any variables you could generate based on the ones you already have?

### Help, Task 4

Both powder milk and concentrated milk are types of milk that need to be diluted with water. For this reason, it may be of interest to combine the two to a single variable. We can use the logic of if statements to create this new variable. The **|** below stands for logical **or**. For example:

campy$diluted <- ifelse(campy$concentrated == 1 | campy$powder == 1, 1, 0)

## Task 5. Perform a descriptive analysis:

### Help, Task 5

To see if the age distribution between cases and controls differs (which you might not expect, since you have frequency-matched for age), you can use either the t-test or the Wilcoxon’s ranksum test (also called the Mann-Whitney test). The first one can be used only when the distribution of age in both groups is normal. The latter one can be used otherwise.

The Shapiro-Wilk test is a normality test. Its null hypothesis is that the normal distribution is followed. Hence, a p-value below your alpha (usually 0.05) means that the normal distribution is *not* followed and so a p-value above your alpha is preferable. To test whether the age distribution is normal among both cases and controls, you can run:

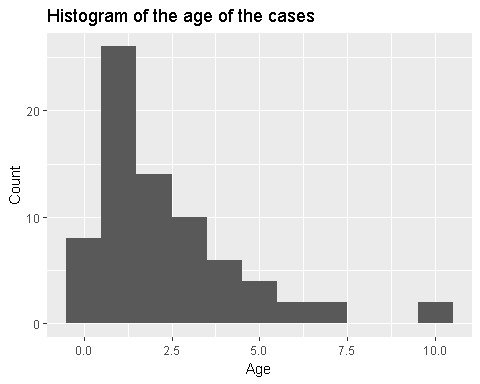
shapiro.test(campy$age)

##   
## Shapiro-Wilk normality test  
##   
## data: campy$age  
## W = 0.83527, p-value = 6.73e-15

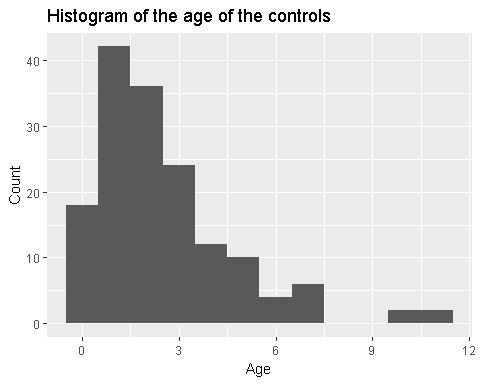
You can also visualise the age distribution among cases and controls.

Below we use the qplot function from the package ggplot2 and create histograms of the age of cases and controls. We can specify labels for the x-axis and y-axis as well as titles.

age\_hist\_cases <- qplot(campy$age[campy$case == 1],  
 xlab = "Age",  
 ylab = "Count",  
 main = "Histogram of the age of the cases ",  
 binwidth = 1)  
age\_hist\_cases



age\_hist\_controls <- qplot(campy$age[campy$case == 0],  
 xlab = "Age",  
 ylab = "Count",  
 main = "Histogram of the age of the controls",  
 binwidth = 1)  
age\_hist\_controls



Age does not appear to be normally distributed.

Now that you are absolutely sure that the hypothesis of normality in the variable age is not really the case, you choose to run Wilcoxon’s ranksum test:

wilcox.test(age ~ case, data = campy)

##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: age by case  
## W = 6024, p-value = 0.592  
## alternative hypothesis: true location shift is not equal to 0

If you had gone for the t-test, the command would have been:

t.test(age~case, var.equal = TRUE, data = campy)

Note: The null hypothesis in both t-test and Wilcoxon’s ranksum test is that the mean (or median, respectively) of the continuous variable (age) does not differ between the two groups of your dichotomous variable case (cases and controls). P-values lower than your alpha suggest you should consider that age differs between cases and controls.

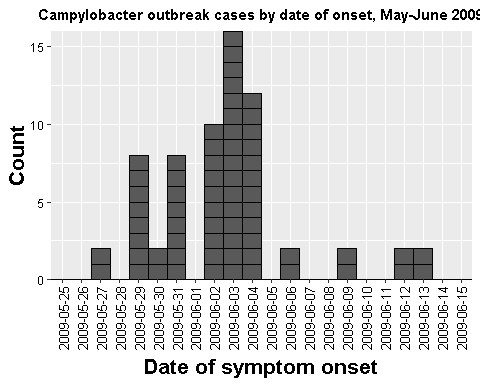
We would also like to construct an **epidemic curve**. We can use the previously loaded epicurve function (created by Daniel Gardiner FETP fellow from C2015). This function is very flexible and can be adapted to a variety of different data formats. You can read about all the different elements of this function by clicking on the funcion in the Global environment.

You can now format the epicurve in terms of the time period (day, week, month, quarter etc), the start and stop date, labels and more.

epicurve\_campy <- epicurve(campy, date.col = "datesym", time.period = "day",  
 start.at = "2009-05-25", stop.at = "2009-06-15",  
 xlab = "Date of symptom onset", ylab = "Count",  
 col.pal = 4, label.breaks = 0, epi.squares = TRUE, na.rm = TRUE)

## 166 rows have missing dates OR dates outside of the start/stop period

# As epicurve\_campy is a ggplot object, it is possible to tailor it as desired  
epicurve\_campy <- epicurve\_campy +  
 # rotating the x axis label by 90   
 theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +  
 # adding a title  
 ggtitle("Campylobacter outbreak cases by date of onset, May-June 2009") +  
 # centring the title and reducing its size   
 theme(plot.title = element\_text(hjust = 0.5, size = 11))   
  
epicurve\_campy



# You can save the epicurve as follows  
ggsave(filename = "epicurve.png")

## Task 6. Conduct the univariate analysis

### Help, Task6

The appropriate measure of impact for a case-control study is the odds ratio (OR). The **epi.2by2** function calculates the OR, 95% CI and the attributable fraction among the exposed in the population.

In order to use the epi.2by2 function, we first need to convert the outcome and exposure variables into factor/categorical variables to facilitate interpretation.

# We list the outcome/exposure variables  
vars <- c("case", "sex", "supply", "tap", "bottled", "filter", "well", "pacifier1", "pacifier2", "dishwasher", "microwave1", "microwave2", "breastfeeding", "concentrated", "powder", "freshmilk", "dilutetap", "diluted")  
  
  
# Convert all of those variables to factor variables and reorder the levels to aid interpretation  
for (var in vars) {  
 campy[,var] <- factor(campy[,var],levels = c(1,0))   
}

The for (…) notation defines a loop in R and it loosely be read as ‘For every element of the vector *vars* (which we temporarily call *var*), use the function *factor* on the respective column, which is named *var*.’ We could have done the same by writing out the command of the factor function for each and every variable. By using the *for* loop, we have just saved time and code length [and this is very desirable].

The epi.2by2 function can be used to calculate both RRs and ORs and you can find out more information on the function by writing **?epi.2by2** in the console (or by using the Help function in the bottom right panel in RStudio). The epi.2by2 function requires data to be in a table format and we specify that we want to calculate ORs by adding **method = “case.control”** as below. You can do that first for the variables *sex*, *supply* and *well*.

# Create a table with exposure and outcome variables  
sex <- table(campy$sex, campy$case)  
  
# Apply epi.2by2 function to the table  
uni\_sex <- epi.2by2(sex, method = "case.control")  
uni\_sex

## Outcome + Outcome - Total Prevalence \*  
## Exposed + 40 82 122 32.8  
## Exposed - 34 76 110 30.9  
## Total 74 158 232 31.9  
## Odds  
## Exposed + 0.488  
## Exposed - 0.447  
## Total 0.468  
##   
## Point estimates and 95 % CIs:  
## -------------------------------------------------------------------  
## Odds ratio (W) 1.09 (0.63, 1.90)  
## Attrib prevalence \* 1.88 (-10.12, 13.88)  
## Attrib prevalence in population \* 0.99 (-9.53, 11.50)  
## Attrib fraction (est) in exposed (%) 8.26 (-65.47, 49.30)  
## Attrib fraction (est) in population (%) 4.48 (-28.36, 28.92)  
## -------------------------------------------------------------------  
## X2 test statistic: 0.094 p-value: 0.759  
## Wald confidence limits  
## \* Outcomes per 100 population units

supply <- table(campy$supply, campy$case)  
uni\_supply <- epi.2by2(supply, method = "case.control")  
uni\_supply

## Outcome + Outcome - Total Prevalence \*  
## Exposed + 66 104 170 38.8  
## Exposed - 8 54 62 12.9  
## Total 74 158 232 31.9  
## Odds  
## Exposed + 0.635  
## Exposed - 0.148  
## Total 0.468  
##   
## Point estimates and 95 % CIs:  
## -------------------------------------------------------------------  
## Odds ratio (W) 4.28 (1.92, 9.57)  
## Attrib prevalence \* 25.92 (14.82, 37.02)  
## Attrib prevalence in population \* 18.99 (8.72, 29.27)  
## Attrib fraction (est) in exposed (%) 76.53 (46.21, 90.94)  
## Attrib fraction (est) in population (%) 68.37 (36.98, 84.12)  
## -------------------------------------------------------------------  
## X2 test statistic: 14.051 p-value: < 0.001  
## Wald confidence limits  
## \* Outcomes per 100 population units

well <- table(campy$well, campy$case)  
uni\_well <- epi.2by2(well, method = "case.control")  
uni\_well

## Outcome + Outcome - Total Prevalence \*  
## Exposed + 66 108 174 37.9  
## Exposed - 8 50 58 13.8  
## Total 74 158 232 31.9  
## Odds  
## Exposed + 0.611  
## Exposed - 0.160  
## Total 0.468  
##   
## Point estimates and 95 % CIs:  
## -------------------------------------------------------------------  
## Odds ratio (W) 3.82 (1.70, 8.56)  
## Attrib prevalence \* 24.14 (12.70, 35.57)  
## Attrib prevalence in population \* 18.10 (7.39, 28.81)  
## Attrib fraction (est) in exposed (%) 73.69 (39.47, 89.87)  
## Attrib fraction (est) in population (%) 65.84 (31.66, 82.92)  
## -------------------------------------------------------------------  
## X2 test statistic: 11.667 p-value: < 0.001  
## Wald confidence limits  
## \* Outcomes per 100 population units

Instead of looking at each variable one by one, we can also add the **exposure variables** in a loop and apply the epi.2by2 function to each variable of interest at one time and save the resulting outputs to a list of dataframes.

vars2 <- c("sex", "supply", "tap", "bottled", "filter", "well", "pacifier1", "pacifier2", "dishwasher", "microwave1", "microwave2", "breastfeeding", "concentrated", "powder", "freshmilk", "dilutetap", "diluted")  
  
# Create an empty list to store the output of the loop  
output <- list()  
  
for (var in vars2) {  
 # We make a table with each exposure variable and the case variable  
 table <- table(campy[,var], campy$case)   
 # apply epi.2by2 function to each table  
 uni\_table <- epi.2by2(table, method = "case.control")  
 # Save the results in the output list  
 output[[var]] <- uni\_table  
}  
  
output

The next step would involve extracting the relevant data from the output to make our final table of interest, which could take some time. This process can be sped up through the use of the **single variable analysis** function created by Daniel Gardiner (FETP fellow from C2015). This function gives similar output to **cctable** in Stata.

In order for this function to give similar output to the cctable command, the exposure and outcome variables must be converted to numeric variables as below:

**Note**: It is not possible to directly convert a factor variable to a numeric variable. You must first convert the factor variable to a character and then convert the character to a numeric variable.

vars <- c("case", "sex", "supply", "tap", "bottled", "filter", "well", "pacifier1", "pacifier2", "dishwasher", "microwave1", "microwave2", "breastfeeding", "concentrated", "powder", "freshmilk", "dilutetap", "diluted")  
  
# Convert factor to character to numeric  
for (var in vars) {  
 campy[,var] <- as.numeric(as.character(campy[,var]))   
}

The variables are now in a format compatible with the single variable analysis (sva) function. You can learn more about this function either by clicking on **sva** in the functions section of the global environment or typing **View(sva)** in the console.

The sva function requires definition of:

* the data set
* the outcome of interest
* the exposure variable(s)
* the measure (OR or RR) and
* verbose (FALSE gives restricted output)

vars2 <- c("sex", "supply", "tap", "bottled", "filter", "well", "pacifier1", "pacifier2", "dishwasher", "microwave1", "microwave2", "breastfeeding", "concentrated", "powder", "freshmilk", "dilutetap", "diluted")  
  
  
# Use the sva function, specifying each element of the function  
a <- sva(campy, outcome = "case", exposures = c(vars2), measure = "or", verbose = TRUE)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| exposure | cases | cases.exp | %cases.exp | controls | controls.exp | %controls.exp | or | lower | upper | p.value |
| sex | 74 | 40 | 54.1 | 158 | 82 | 51.9 | 1.09 | 0.63 | 1.90 | 0.78 |
| supply | 74 | 66 | 89.2 | 158 | 104 | 65.8 | 4.28 | 1.92 | 9.57 | 0.00 |
| tap | 72 | 65 | 90.3 | 158 | 121 | 76.6 | 2.84 | 1.20 | 6.72 | 0.02 |
| bottled | 72 | 11 | 15.3 | 158 | 70 | 44.3 | 0.23 | 0.11 | 0.46 | 0.00 |
| filter | 70 | 0 | 0.0 | 156 | 30 | 19.2 | 0.00 | 0.00 | NaN | 0.00 |
| well | 74 | 66 | 89.2 | 158 | 108 | 68.4 | 3.82 | 1.70 | 8.56 | 0.00 |
| pacifier1 | 72 | 36 | 50.0 | 156 | 84 | 53.8 | 0.86 | 0.49 | 1.50 | 0.67 |
| pacifier2 | 36 | 27 | 75.0 | 84 | 42 | 50.0 | 3.00 | 1.26 | 7.14 | 0.02 |
| dishwasher | 40 | 18 | 45.0 | 82 | 69 | 84.1 | 0.15 | 0.06 | 0.36 | 0.00 |
| microwave1 | 28 | 22 | 78.6 | 61 | 41 | 67.2 | 1.79 | 0.63 | 5.11 | 0.32 |
| microwave2 | 22 | 18 | 81.8 | 56 | 48 | 85.7 | 0.75 | 0.20 | 2.80 | 0.73 |
| breastfeeding | 72 | 4 | 5.6 | 158 | 16 | 10.1 | 0.52 | 0.17 | 1.62 | 0.32 |
| concentrated | 72 | 42 | 58.3 | 156 | 66 | 42.3 | 1.91 | 1.08 | 3.36 | 0.03 |
| powder | 72 | 18 | 25.0 | 154 | 38 | 24.7 | 1.02 | 0.53 | 1.94 | 1.00 |
| freshmilk | 72 | 14 | 19.4 | 154 | 48 | 31.2 | 0.53 | 0.27 | 1.05 | 0.08 |
| dilutetap | 60 | 24 | 40.0 | 99 | 31 | 31.3 | 1.46 | 0.75 | 2.85 | 0.30 |
| diluted | 72 | 60 | 83.3 | 156 | 104 | 66.7 | 2.50 | 1.24 | 5.05 | 0.01 |

## Stratified analysis

We have seen so far that some exposures appear to be statistically significantly associated with being a case. Some other exposures do not appear to be associated with disease outcome. Being a field epidemiologist, you decide not to stop your analysis yet. You think your results are very interesting and, after a discussion with your colleagues, you go further and perform a stratified analysis.

## Task 7. Stratify by water supply zone and identify effect modification or confounding

### Help, Task7

Stratifying essentially means to run the same analysis as in the univariate analysis, but restricting the analysis to the two separate strata we are interested in each time (in this case, the rural area and the town area).

We will illustrate the effect of stratification using tap as the exposure variable and supply as stratifiying variable. As we will use the **epi.2by2** function to perform the stratification, we first need to reconvert the outcome and exposure variables to factor variables. Note: the sva function doesn’t currently have a stratifying function.

# The outcome and exposure variables were defined above as vars  
  
# Convert all of those variables to factor variables and re-order the levels to aid interpretation  
for (var in vars) {  
 campy[,var] <- factor(campy[,var], levels = c(1,0))   
}

First, we conduct the univariate analysis with **tap** as the exposure variable:

tap <- table(campy$tap, campy$case)  
uni\_tap <- epi.2by2(tap, method = "case.control")  
uni\_tap

## Outcome + Outcome - Total Prevalence \*  
## Exposed + 65 121 186 34.9  
## Exposed - 7 37 44 15.9  
## Total 72 158 230 31.3  
## Odds  
## Exposed + 0.537  
## Exposed - 0.189  
## Total 0.456  
##   
## Point estimates and 95 % CIs:  
## -------------------------------------------------------------------  
## Odds ratio (W) 2.84 (1.20, 6.73)  
## Attrib prevalence \* 19.04 (6.24, 31.83)  
## Attrib prevalence in population \* 15.40 (3.04, 27.75)  
## Attrib fraction (est) in exposed (%) 64.64 (13.69, 87.41)  
## Attrib fraction (est) in population (%) 58.48 (11.38, 80.55)  
## -------------------------------------------------------------------  
## X2 test statistic: 5.997 p-value: 0.014  
## Wald confidence limits  
## \* Outcomes per 100 population units

Now, we will repeat the above analysis while stratifying by water supply, where supply = 1 is for rural areas and supply = 0 is for urban areas.

# Based on supply = 1  
tap1 <- table(campy$tap[campy$supply == 1], campy$case[campy$supply == 1])  
tap\_supp1 <- epi.2by2(tap1, method = "case.control")  
tap\_supp1

## Outcome + Outcome - Total Prevalence \*  
## Exposed + 58 72 130 44.6  
## Exposed - 6 32 38 15.8  
## Total 64 104 168 38.1  
## Odds  
## Exposed + 0.806  
## Exposed - 0.188  
## Total 0.615  
##   
## Point estimates and 95 % CIs:  
## -------------------------------------------------------------------  
## Odds ratio (W) 4.30 (1.68, 10.98)  
## Attrib prevalence \* 28.83 (14.42, 43.23)  
## Attrib prevalence in population \* 22.31 (8.58, 36.03)  
## Attrib fraction (est) in exposed (%) 76.54 (37.93, 92.50)  
## Attrib fraction (est) in population (%) 69.53 (31.20, 86.51)  
## -------------------------------------------------------------------  
## X2 test statistic: 10.361 p-value: 0.001  
## Wald confidence limits  
## \* Outcomes per 100 population units

# Based on supply = 0  
tap0 <- table(campy$tap[campy$supply == 0], campy$case[campy$supply == 0])  
tap\_supp0 <- epi.2by2(tap0, method = "case.control")  
tap\_supp0

## Outcome + Outcome - Total Prevalence \*  
## Exposed + 7 49 56 12.5  
## Exposed - 1 5 6 16.7  
## Total 8 54 62 12.9  
## Odds  
## Exposed + 0.143  
## Exposed - 0.200  
## Total 0.148  
##   
## Point estimates and 95 % CIs:  
## -------------------------------------------------------------------  
## Odds ratio (W) 0.71 (0.07, 7.04)  
## Attrib prevalence \* -4.17 (-35.22, 26.89)  
## Attrib prevalence in population \* -3.76 (-34.73, 27.20)  
## Attrib fraction (est) in exposed (%) -39.16 (-1454.10, 97.41)  
## Attrib fraction (est) in population (%) -35.00 (-912.15, 81.99)  
## -------------------------------------------------------------------  
## X2 test statistic: 0.084 p-value: 0.772  
## Wald confidence limits  
## \* Outcomes per 100 population units

The above approach provides the stratum-specific ORs, which we can compare to the crude odds ratio from the univariate analysis above. However, it does not provide the Mantel-Haenszel (M-H) odds ratio. We can obtain the M-H odds ratio and loop over the exposure variables of interest by doing the following:

# Define the exposure variables to be included in the analysis  
vars4 <- c("tap", "bottled", "filter", "well", "pacifier1", "pacifier2", "dishwasher", "microwave1", "breastfeeding", "concentrated", "powder", "freshmilk", "dilutetap", "diluted")  
  
# The variable "microwave 2" was excluded from the list of variables as it blocks the loop  
  
# Create a list to store the output  
output2 <- list()  
  
# create a 3 way table for each exposure variable of interest, the outcome and stratifiying variable in that order  
for (var in vars4) {  
 a <- table(campy[,var], campy$case, campy$supply)  
 # Use the epi.2by2 function to calculate OR   
 mh <- epi.2by2(a, method = "case.control")  
 # Identify the elements of interest from the mh object and append together   
 resultstable <- round(rbind(mh$massoc$OR.crude.wald,   
 mh$massoc$OR.strata.wald,   
 mh$massoc$OR.mh.wald),2)  
 # Create labels for each row of the results table  
 rownames(resultstable) <- c("Crude", "Strata 1", "Strata 0", "MH")  
 output2[[var]] <- resultstable  
}

## Warning in qf(1 - N., 2 \* a, 2 \* c + 2): NaNs produced

## Warning in qf(1 - N., 2 \* sa, 2 \* sc + 2): NaNs produced

## Warning in qf(1 - N., 2 \* a, 2 \* c + 2): NaNs produced

output2

## $tap  
## est lower upper  
## Crude 2.84 1.20 6.73  
## Strata 1 4.30 1.68 10.98  
## Strata 0 0.71 0.07 7.04  
## MH 3.45 1.46 8.20  
##   
## $bottled  
## est lower upper  
## Crude 0.23 0.11 0.46  
## Strata 1 0.15 0.07 0.36  
## Strata 0 1.02 0.22 4.73  
## MH 0.23 0.11 0.47  
##   
## $filter  
## est lower upper  
## Crude 0 0 NaN  
## Strata 1 0 0 NaN  
## Strata 0 0 0 NaN  
## MH 0 NaN NaN  
##   
## $well  
## est lower upper  
## Crude 3.82 1.70 8.56  
## Strata 1 1.28 0.23 7.19  
## Strata 0 1.92 0.33 11.23  
## MH 1.53 0.44 5.39  
##   
## $pacifier1  
## est lower upper  
## Crude 0.86 0.49 1.50  
## Strata 1 0.72 0.39 1.36  
## Strata 0 2.79 0.52 15.05  
## MH 0.88 0.49 1.56  
##   
## $pacifier2  
## est lower upper  
## Crude 3.00 1.26 7.14  
## Strata 1 3.23 1.14 9.11  
## Strata 0 1.55 0.26 9.08  
## MH 2.71 1.12 6.55  
##   
## $dishwasher  
## est lower upper  
## Crude 0.15 0.07 0.36  
## Strata 1 0.09 0.03 0.28  
## Strata 0 0.37 0.07 1.89  
## MH 0.14 0.06 0.35  
##   
## $microwave1  
## est lower upper  
## Crude 1.79 0.63 5.11  
## Strata 1 5.67 1.16 27.56  
## Strata 0 0.21 0.03 1.38  
## MH 1.71 0.62 4.68  
##   
## $breastfeeding  
## est lower upper  
## Crude 0.52 0.17 1.62  
## Strata 1 0.43 0.13 1.36  
## Strata 0 0.00 0.00 NaN  
## MH 0.41 0.13 1.29  
##   
## $concentrated  
## est lower upper  
## Crude 1.91 1.08 3.36  
## Strata 1 2.68 1.41 5.10  
## Strata 0 0.80 0.18 3.54  
## MH 2.20 1.23 3.95  
##   
## $powder  
## est lower upper  
## Crude 1.02 0.53 1.94  
## Strata 1 0.95 0.46 1.95  
## Strata 0 1.17 0.21 6.54  
## MH 0.98 0.50 1.90  
##   
## $freshmilk  
## est lower upper  
## Crude 0.53 0.27 1.05  
## Strata 1 0.54 0.25 1.15  
## Strata 0 0.67 0.12 3.64  
## MH 0.56 0.28 1.12  
##   
## $dilutetap  
## est lower upper  
## Crude 1.46 0.75 2.85  
## Strata 1 3.44 1.44 8.20  
## Strata 0 0.43 0.07 2.62  
## MH 2.24 1.06 4.71  
##   
## $diluted  
## est lower upper  
## Crude 2.50 1.24 5.05  
## Strata 1 3.48 1.59 7.62  
## Strata 0 0.86 0.15 4.81  
## MH 2.78 1.37 5.66