Analysis of surveillance data : Analysing Cryptosporidium notification data from country X, 2004-2015

# Copyright and License

**Source:**

This case study was first designed by Esther Kissling, EpiConcept, 2016; it was then translated in to *R* by Alexander Spina in 2018. It is based on surveillance data from an anonymous country.

**Revisions:**

*If you modify this case study, please indicate below your name and changes you made*

**You are free:**

* **to Share** — to copy, distribute and transmit the work
* **to Remix** — to adapt the work

**Under the following conditions:**

* **Attribution** — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work). The best way to do this is to keep as it is the list of contributors: sources, authors and reviewers.
* **Share Alike** — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one. Your changes must be documented. Under that condition, you are allowed to add your name to the list of contributors.
* You cannot sell this work alone but you can use it as part of a teaching.

**With the understanding that:**

* **Waiver** — Any of the above conditions can be waived if you get permission from the copyright holder.
* **Public Domain** — Where the work or any of its elements is in the public domain under applicable law, that status is in no way affected by the license.
* **Other Rights** — In no way are any of the following rights affected by the license:
  + Your fair dealing or fair use rights, or other applicable copyright exceptions and limitations;
  + The author’s moral rights;
  + Rights other persons may have either in the work itself or in how the work is used, such as publicity or privacy rights.
* **Notice** — For any reuse or distribution, you must make clear to others the license terms of this work by keeping together this work and the current license.

This licence is based on <http://creativecommons.org/licenses/by-sa/3.0/>

# Objectives

At the end of the case study, participants should be able to analyse surveillance data using Stata, going through all steps including

* data checking,
* data cleaning/recoding,
* data description,
* appropriate statistical testing,
* merging datasets with denominators,
* calculating incidence rates and
* calculating incidence rate ratios.

# Guide to the case study

The case study is designed for use with *R* statistical programming software.  
All files necessary for completing a session are placed in the corresponding session folder. There should be no need to copy files from other session folders.

# Background

Cryptosporidium is a protozoal parasite that causes a diarrhoeal illness in humans known as cryptosporidiosis. It is transmitted by the faeco-oral route, with both animals and humans serving as potential reservoirs. Cryptosporidiosis is a notifiable disease in many countries across the world.

## Case definition

3.9 Cryptosporidiosis

**Clinical Criteria** Any person with at least one of the following two:

* Diarrhoea;
* Abdominal pain.

**Laboratory Criteria** At least one of the following four:

* Demonstration of *Cryptosporidium* oocysts in stool;
* Demonstration of *Cryptosporidium* in intestinal fluid or small-bowel biopsy specimens;
* Detection of *Cryptosporidium* nucleic acid in stool;
* Detection of *Cryptosporidium* antigen in stool.

**Epidemiological Criteria**

One of the following *five* epidemiological links:

* Human to human transmission
* Exposure to a common source
* Animal to human transmission
* Exposure to contaminated food/drinking water
* Environmental exposure

**Case Classification**

**A. Possible case** NA

**B. Probable case**

Any person meeting the clinical criteria with an epidemiological link

**C. Confirmed case**

Any person meeting the clinical and laboratory criteria

Note: If the national surveillance system is not capturing clinical symptoms, all laboratory-confirmed individuals should be reported as confirmed cases.

*From:* [Commission Implementing Decision on the communicable diseases and related special health issues to be covered by epidemiological surveillance – Annex 1](https://ec.europa.eu/health/sites/health/files/communicable_diseases/docs/2018_impldecision_annex1_en.pdf) (\*replacing Commission Decision No. 2000/96/EC).

## Datasets

The main dataset for the case study covers the years from 2004-2015. The dataset is available in Excel: “crypto.xls”.

Each record designates one case of Cryptosporidium. Information on species (e.g. *C. parvum*, *C. hominis*, etc.) is not available in this dataset in country X. Due to funding restrictions, routine speciation of samples was stopped in most laboratories in country X.

Below you can find a data dictionary of the variables and values included in the dataset:

|  |  |  |  |
| --- | --- | --- | --- |
| Variable name | Type | Code | Definition |
| ID | String |  | Identifies each patient with cryptosporidiosis |
| Notif\_Date | Date | dd/mm/yyyy | Date of notification |
| Week | Integer |  | Week of notification |
| Month | Integer |  | Month of notification |
| Quarter | Integer | 0=female, 1=male | Quarter of notification |
| Year | Integer |  | Year of notification |
| Region | Integer | 1, 2, 3, 4, 5, 6, 7 | Region of notification of case |
| OnsetDate | Date | dd/mm/yyyy | Date of symptom onset |
| AgeY | Integer |  | Age in years |
| Sex | Integer | 0=female, 1=male | Gender |
| PatientType | String |  | Type of patient, e.g. A&E patient, GP patient, Hospital inpatient, etc. |
| CountryofInfection | String |  | Suspected country of infection. |
| CaseClassification | String |  | Classification of case: confirmed, probable, not specified. |

You will also be using denominator data for the case study. Here we are using 2011 denominator data in the Excel spreadsheet: “denominators2011.xls”.

There are three tabs on the spreadsheet: Region, AgeGroup, AgeGroup by Region, which give the following information

* Region: provides total population numbers by region in 2011
* AgeGroup: provides total population numbers for each age in years from 0 to 100 in 2011 - AgeGroup by Region: provides total population numbers for each age in years from 0 to 100 by region in 2011

## Main task for the case study

Your main task for the case study is to analyse the surveillance data, with a focus on 2015 data. Your task is in particular to calculate incidence rates, and compare the 2015 data with 2014 data and other previous years. When using *R*, please use *R-scripts*, ensuring there is a “Master” *R-scripts* that serves as a table of contents. This “Master” file should link to other *R-scripts* using the *source* function.

## Q1: Create a plan of analysis

## Help Q1 (example):

**Data checking**

* Checks for completeness
* Checks for legal values (range, unexpected values): cross-tabulations
* Checking consistency of dates
* Histograms of continuous variables (age and date variables)

**Recoding the data**

* Recode continuous variables if needed (e.g. age into age groups)
* Recode string variables where appropriate
* Add labels if appropriate
* Possibly:
  + Recode PatientType variable into hospitalised/not hospitalised
  + Create a proxy for urban/rural
  + Create an imported (yes/no) variable
  + Create a “count” variable indicating that each line represents one case (which facilitates further analysis)

**Descriptive analysis (with a focus on 2015 data)**

* Describe the number of cases in 2015
* Describe age (age histogram, median and interquartile range)
* Describe sex, hospitalisation status of patient, region of notification, country of infection.

**Comparative analysis 2015 to 2014**

* Age histogram 2015 to 2014
* Comparison of age 2015 to 2014:
  + Comparison of means
  + Comparison of medians
* Comparison of proportion of male/female, hospitalised, urban/rural
* Choose the appropriate statistical tests and the appropriate level of confidence

**Calculating annual incidence rates**

* Calculate incidence rates and their 95% CI by year using Stata’s “ci” command
* Plot the annual incidence rates
* Calculate incidence rates and their 95% CI by year and region
* Calculate age group-specific incidence rates and their 95% CI by year

**Calculate incidence rate ratios (examples)**

* Calculate incidence rate ratios, 95% CI and p-values between years with 2015 as the reference
* Calculate incidence rate ratios, 95% CI and p-values between the average of 2012-14 and 2015
* Calculate incidence rate ratios, 95% CI and p-values between urban and rural areas for 2015.

## Q2: Data checking

Install and Load packages required for this case study. Ensure you are in your correct working directory. Import the worksheet “Crypto” in the Excel spreadsheet “crypto.xls”. Familiarise yourself with the data. Check the data: check completeness of the variables, cross-tabulate the data to check legal values, check consistency of dates, check continuous variables. NB: You can refer to the data dictionary in the background section.

Are there any issues in the data? Do you need to carry out any data cleaning?

Use *R-scripts*.

## Help Q2:

### 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](https://cran.r-project.org/)) and many more packages in development can be found on [GitHub](https://github.com/). They may be installed and updated over the Internet.

We will mainly use packages which come ready installed with R (base code), but where it makes things easier we will use add-on packages. All the R packages you need for the exercises can be installed over the Internet.

You will need to install these before starting. We will be using the following packages.

* *epiR*: for creating two by two tables and calculating incidence rates
* *broom*: for cleaning up the output of poisson regressions

Packages can be installed using the *install.packages* function, where you specify the name of the package in quotation marks and whether you also want to install other packages which are required to run the package of interest (where TRUE/FALSE means YES/NO); for example:

#install the epiR package and packages it depends on  
install.packages("epiR", dependencies = TRUE)

If you want to install multiple packages at once you can simply save (assign them to an object using the arrow, and “c” simply pulls strings together) the names of the packages you are interested in as a string object (which you can call whatever you like, in this case we called it required\_packages) and pass these through the *install.packages* function; for example:

# Installing multiple required packages for the case study  
required\_packages <- c("epiR", "broom")  
install.packages(required\_packages, dependencies = TRUE)

Alternatively, if you are unsure whether the packages are already installed you can run the following for-loop. It is not necessary to understand the code at this point, but simply appreciated that it is doing the same as above, while also checking whether the packages are installed.

# Installing required packages for this case study  
for (pkg in required\_packages) {  
 if (!pkg %in% rownames(installed.packages())) {  
 install.packages(pkg)  
 }  
}

Once you have successfully installed all your packages they are saved on your computer. This means that you only need to install them the first time.

After that, whenever you want to use a package in your *R* session, you need to load the package using the *library* function; the console will give you several messages in red, however these are most often just information and not errors. Note that here, you do not require quotation marks, for example:

library(epiR)

Here too, you can load multiple packages at the same time within a loop; again do not try to understand this just yet, just appreciate it is possible.

# Loading required packages for this case study  
required\_packages <- c("epiR", "broom")  
  
for (i in seq(along = required\_packages)) {  
 library(required\_packages[i], character.only = TRUE)  
}

### Setting your working directory

You can check the path for your current working directory using the *getwd* function.

#Check your current working directory  
getwd()

To set your working directory you can use the *setwd* function.

setwd("C:/Users/Username/Desktop/EpiconceptCrypto")

### Reading in files

Import the dataset from a comma separated value (.csv) file using the *read.csv* function, storing it as a data frame within *R* called crypto. For a CSV file the separator is normally a comma, however depending on the language of your operating system this can also be other values, for example a semi-colon. Here we also specify that we do not want to read in string (character or grouped variables as factors).

crypto <- read.csv("crypto.csv", sep = ";" ,   
 stringsAsFactors = FALSE )

### Familiarise yourself with data

You can examine the structure of your data set using the following functions. The *str* function will provide an overview of which variable types are in your dataset. The *summary* function will give minimum, maximum, first and third quartiles as well as medians and means for variables which are not strings (characters). Each of these commands can be run for individual variables also. You can refer to an individual variable of a data set by using the **$**, for example, if you wanted to obtain a summary of the a numeric age variable, then you would write **summary(crypto$age)**.

# str provides an overview of the number of observations and variable types  
str(crypto)  
  
# summary provides mean, median and max values of your variables  
summary(crypto)

You can also look at completeness of specific variables by combining the *table* function with the *is.na* function. It is also possible to combine multiple arguments using arguments such as “and” (using &) as well as “or” (using |). Note that *R* is case sensitive, so that there is a difference between “Not specified” and “Not Specified” in the PatientType and CountryofInfection variables This could also be achieved using a package as described in the appendix.

# Examine how many are missing or unknown in the AgeY variable   
table(is.na(crypto$AgeY) | crypto$AgeY == "Unknown")  
  
# missing, unknown or not specified in the PatientType variable  
table(is.na(crypto$PatientType) |   
 crypto$PatientType == "Unknown" |   
 crypto$PatientType == "Not Specified")  
  
# missing, unknown or not specified in the CountryofInfection variable  
table(is.na(crypto$CountryofInfection) |   
 crypto$CountryofInfection == "Unknown" |   
 crypto$CountryofInfection == "Not specified")

You can also check if the onset date is before or on the same day as the notification date, and then return the corresponding IDs:

# check number not missing with onset on or before notification date  
  
table(!is.na(crypto$Notif\_Date) &   
 crypto$OnsetDate <= crypto$Notif\_Date)

There are two ways to select rows which have onset after notification. One way is to use the *subset* function, which you specify the dataset in the x argument, then provide a rule for selecting rows in the subset argument and finally specify which columns to select. The second alternative involves using square brackets to subset the data frame; in this scenario what comes before the comma specifies rows and what comes after specifies columns; for example **dataset[rows, columns]**. Both options give the same outcome.

# return IDs, onset and notification dates for those with onset after notification  
subset(  
 x = crypto,  
 subset = !is.na(crypto$Notif\_Date) &  
 crypto$OnsetDate > crypto$Notif\_Date,  
 select = c("ID", "OnsetDate", "Notif\_Date")  
)  
  
  
# return IDs, onset and notification dates for those with onset after notification  
  
crypto[which(!is.na(crypto$Notif\_Date) &  
 crypto$OnsetDate > crypto$Notif\_Date), c("ID", "OnsetDate", "Notif\_Date")]

You can also check if all ages are within a reasonable age range. To do this first change “Unknown” to be NA and then create a new age variable which is AgeY in numeric form.

# replace unknown with NA   
crypto$AgeY[crypto$AgeY == "Unknown"] <- NA  
  
# create new age variable as numeric of AgeY  
crypto$age <- as.numeric(crypto$AgeY)

You can then use *summary* to get information about the age variable.

# summary provides mean, median and max values of age  
summary(crypto$age)

For numeric variables, such as age and dates, you can use histograms to check for unusual patterns or outliers. You specify your variable as well as axis labels. To save you can plot, then use *dev.copy* to choose a file type and name; *dev.off* closes the connection. For date variable you need to specify what time frame you would like to plot, such as “days”, “weeks”, “months”, “years”. Because there are so many points to plot, you need to specify you want the frequency, otherwise the density will be plotted.

#Plot a histogram of age  
hist(crypto$age,  
 xlab = "Age",  
 ylab = "Count"  
)  
  
#save histogram of age as a png file  
dev.copy(png,'age.png')  
dev.off()  
  
  
#plot histogram of notification date  
 #choose days and frequency  
hist(crypto$Notif\_Date,  
 breaks = "days",  
 freq = TRUE,  
 xlab = "Notification date",  
 ylab = "Count"  
)  
  
#save as a png  
dev.copy(png,'notificationdate.png')  
dev.off()  
  
  
  
#plot histogram of onset date  
 #choose days and frequency  
hist(crypto$OnsetDate,  
 breaks = "days",  
 freq = TRUE,  
 xlab = "Onset date",  
 ylab = "Count"  
)  
  
#save as a png  
dev.copy(png,'onsetdate.png')  
dev.off()

## Q3: Data recoding

Rename all variable names to lower case. Recode string variables to numeric where this is useful. Add labels to relevant variables. Create the age bands used for the annual report (0-4 5-9 10-14 15-19 20-24 25-34 35-44 45-54 55-64 65+). Create a variable called “count” signifying that each record has one disease count. Optional: Generate a new variable indicating if a patient is hospitalised or not. Create a variable for “urban/rural”, with Region 1 as a proxy for “urban”. Create a variable indicating if this is an imported case or not.

## Help Q3:

### Reading in files

Import the dataset from a comma separated value (.csv) file using the *read.csv* function, storing it as a data frame within *R* called crypto. For a CSV file the separator is normally a comma, however depending on the language of your operating system this can also be other values, for example a semi-colon. Here we also specify that we do not want to read in string (character or grouped variables as factors).

crypto <- read.csv("crypto.csv", sep = ";" ,   
 stringsAsFactors = FALSE )

### Rename all variable names to all lowercase letters:

You can check and change the variable names in your dataset using the *names* function. Then using the *tolower* function you can re-assign these in lower case letters.

names(crypto) <- tolower( names(crypto) )

### Recode string to numeric variables, where useful:

You have already done this in the previous section, but now the variable names are in lower case.

# replace unknown with NA   
crypto$agey[crypto$agey == "Unknown"] <- NA  
  
# create new age variable as numeric of AgeY  
crypto$age <- as.numeric(crypto$agey)

### Add labels where appropriate:

In order to add labels in *R* you have to change variables in to factors. This allows you to specify levels (the order in which categories appear in output) and then label these levels.

#re-write the sex variable as a factor defining levels and labels  
crypto$sex <- factor(crypto$sex,   
 levels = c(1, 0),   
 labels = c("male", "female")  
 )

#Check the outcome   
table(crypto$sex, useNA = "always")

### Create annual report age groups with labels:

There are several ways to do this. The simplest version is as below, for other options see the appendix.

#generate an empty variable called ar\_age  
crypto$ar\_age <- NA  
  
#where age is under 5, set ar\_age to 0  
crypto$ar\_age[crypto$age < 5] <- 0  
  
#set the rest of the groups   
crypto$ar\_age[crypto$age >= 5 &   
 crypto$age < 10] <- 1  
  
crypto$ar\_age[crypto$age >= 10 &   
 crypto$age < 15] <- 2  
  
crypto$ar\_age[crypto$age >= 15 &   
 crypto$age < 20] <- 3  
  
crypto$ar\_age[crypto$age >= 20 &   
 crypto$age < 25] <- 4  
  
crypto$ar\_age[crypto$age >= 25 &   
 crypto$age < 35] <- 5  
  
crypto$ar\_age[crypto$age >= 35 &   
 crypto$age < 45] <- 6  
  
crypto$ar\_age[crypto$age >= 45 &   
 crypto$age < 55] <- 7  
  
crypto$ar\_age[crypto$age >= 55 &   
 crypto$age < 65] <- 8  
  
crypto$ar\_age[crypto$age >= 65] <- 9  
  
  
  
#change to a factor and define labels   
  
crypto$ar\_age <- factor(crypto$ar\_age,   
 levels = 0:9,   
 labels = c("0-4",  
 "5-9",   
 "10-14",   
 "15-19",   
 "20-24",   
 "25-34",   
 "35-44",   
 "45-54",   
 "55-64",   
 "65+"  
 )  
 )

### Add a count variable that signifies one count of disease:

crypto$count <- 1

### Save the file:

You can save your cleaned dataset as an R datafile (.Rda) using the *save* function and re-load the same dataset using the *load* function.

#save your dataset  
save(crypto, file = "crypto.Rda")

### Optional

*NB.* If doing the optional recoding, please save the file at the end.

### Create a variable for “hospitalised”:

#If hospital inpatient then 1 else 0  
crypto$hospitalised <- ifelse(crypto$patienttype == "Hospital Inpatient",   
 1, 0)  
  
#Not specified and unknown set to missing  
crypto$hospitalised[crypto$patienttype == "Not Specified" |   
 crypto$patienttype == "Unknown"] <- NA

### Create a proxy for urban vs. rural:

#if region is 1 then urban else rural  
crypto$urban <- ifelse(crypto$region == 1, 1, 0)   
  
#add order and labels  
crypto$urban <- factor(crypto$urban,   
 levels = c(1, 0),   
 labels = c("urban", "rural")  
 )

### Create an imported variable:

#If country X then not imported, else imported  
crypto$imported <- ifelse(crypto$countryofinfection == "Country X", 0, 1)   
  
#Not specified and unknown set to missing  
crypto$imported[crypto$countryofinfection == "Not Specified" |   
 crypto$countryofinfection == "Unknown"] <- NA  
  
#add order and labels  
crypto$imported <- factor(crypto$imported,   
 levels = c(1, 0),   
 labels = c("Imported", "Country X")  
 )

## Q4: Descriptive analysis

Use the dataset “crypto recoded.dta”. Focus on the year 2015. Describe the variables in the dataset. Summarise the results.

## Help Q4:

Open your dataset using the load function.

#load your dataset   
load("crypto.Rda")

Restrict your data to 2015 using the subset function. In this situation you over-write your dataset with the subset

#assign your 2015 subset to crypto (over-write original crypto)  
crypto <- subset(  
 x = crypto,  
 subset = year == 2015  
 )

How many cases were notified?

#check number of rows in your dataset  
nrow(crypto)

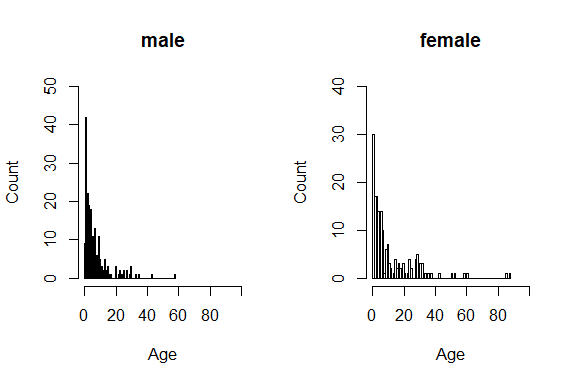
## [1] 390

Describe age.

#Plot a histogram of age  
 #you can specify a bar for each age with "breaks"  
 #you can set your x axis from 0-100 using "xlim"  
hist(crypto$age,   
 xlab = "Age",  
 ylab = "Count",   
 breaks = 100,  
 xlim = c(0, 100)  
)  
  
  
#Get a summary of age   
summary(crypto$age)

To plot side by side histograms you need to use the “par” function.

#specify you want one row of two histograms  
par(mfrow = c(1,2))  
  
#plot a histogram for males (use squarebrackets to subset)  
 #give a title using "main",   
 #set the y axis limits using ylim  
hist(crypto$age[crypto$sex == "male"],   
 main = "male",  
 xlab = "Age",   
 ylab = "Count",  
 breaks = 100,   
 xlim = c(0, 100),   
 ylim = c(0, 50) )  
  
#plot a histogram for females  
hist(crypto$age[crypto$sex == "female"],   
 main = "female",  
 xlab = "Age",   
 ylab = "Count",  
 breaks = 100,   
 xlim = c(0, 100),   
 ylim = c(0, 40) )



Describe sex. To see how to bind these together in to a single contingency table, see the appendix.

#get counts of sex   
 #save table as "counts"  
counts <- table(crypto$sex)   
  
#get proportions for counts table  
prop.table(counts)  
  
#you could also multiple by 100 and round to 2 digits  
round(prop.table(counts)\*100, digits = 2)

Describing hospitalised patients

#get counts of hospitalisations   
 #save table as "counts"  
counts <- table(crypto$hospitalised)   
  
#get rounded proportions of counts  
round(prop.table(counts)\*100, digits = 2)

Do the same for age groupings among hospitalised patients

# get counts of hospitalisations by agegroup  
# save table as "counts"  
counts <- table(crypto$ar\_age, crypto$hospitalised)  
  
# get rounded proportions of counts  
# specify that you want row proportions (margin = 1)  
round(  
 prop.table(counts, margin = 1) \* 100,  
 digits = 2  
)

Describe urban.

#get counts  
 #save table as "counts"  
counts <- table(crypto$urban)   
  
#get rounded proportions of counts  
round(prop.table(counts)\*100, digits = 2)

Describe imported.

#get counts  
 #save table as "counts"  
counts <- table(crypto$imported)   
  
#get rounded proportions of counts  
round(prop.table(counts)\*100, digits = 2)

## Q5: Comparative analysis 2015 vs. 2014

Use the dataset “crypto recoded.dta”. Focus on the year 2015 compared to 2014 data. Look for differences in age, gender, levels of hospitalisation and urban/rural distribution. Think of which statistical tests to use where appropriate.

## Help Q5:

Open your dataset using the load function.

#load your dataset   
load("crypto.Rda")

Drop data from years that are not relevant using the subset function. For this analysis we are only interested in 2015 and 2014 data. In this situation you over-write your dataset with the subset.

#assign your 2014-2015 subset to crypto (over-write original crypto)  
crypto <- subset(  
 x = crypto,  
 subset = year >= 2014  
 )  
  
#check cases per year   
table(crypto$year, useNA = "always")

##   
## 2014 2015 <NA>   
## 351 390 0

Compare age in 2015 to 2014:

# specify you want one row of two histograms  
par(mfrow = c(1,2))  
  
# plot a histogram for males (use squarebrackets to subset)  
 # give a title using "main",   
 # set the y axis limits using ylim  
hist(crypto$age[crypto$year == 2014],   
 main = "2014",  
 xlab = "Age",   
 ylab = "Count",  
 breaks = 100,   
 xlim = c(0, 100),   
 ylim = c(0, 100) )  
  
# plot a histogram for females  
hist(crypto$age[crypto$year == 2015],   
 main = "2015",  
 xlab = "Age",   
 ylab = "Count",  
 breaks = 100,   
 xlim = c(0, 100),   
 ylim = c(0, 100) )

Look at the median and the interquartile range and test for equality of distributions:

# use the aggregate function to group by year  
 # year must be as a list  
 # specify the function you would like to use (summary)  
aggregate(crypto$age, by = list(crypto$year), FUN = summary)  
  
# use the boxplot function to plot   
  
boxplot(age~year, data = crypto)

wilcox.test(crypto$age~crypto$year)

##   
## Wilcoxon rank sum test with continuity correction  
##   
## data: crypto$age by crypto$year  
## W = 65014, p-value = 0.2362  
## alternative hypothesis: true location shift is not equal to 0

Look at the means (and standard deviations) and compare means using the t-test:

# use the aggregate function to group by year   
 # year must be as a list   
 # specify the function you would like to use (summary)  
aggregate(crypto$age, by = list(crypto$year), FUN = summary)  
  
# use t.test function to compare means  
t.test(crypto$age ~ crypto$year)

Comparison of proportion of male/female, hospitalised, urban/rural, imported/not imported:

# For sex  
  
# get counts  
 # save table as "counts"  
counts <- table(crypto$sex, crypto$year)   
  
# get rounded proportions of counts  
 # margin = 2 for column proportions  
round(prop.table(counts, margin = 2) \* 100, digits = 2)  
  
# chisq.test function requires you to input a table  
chisq.test(counts)

# For hospitalised  
  
# get counts  
 # save table as "counts"  
counts <- table(crypto$hospitalised, crypto$year)   
  
# get rounded proportions of counts  
 # margin = 2 for column proportions  
round(prop.table(counts, margin = 2) \* 100, digits = 2)  
  
# chisq.test function requires you to input a table  
chisq.test(counts)

# For urban  
  
# get counts  
 # save table as "counts"  
counts <- table(crypto$urban, crypto$year)   
  
# get rounded proportions of counts  
 # margin = 2 for column proportions  
round(prop.table(counts, margin = 2) \* 100, digits = 2)  
  
# chisq.test function requires you to input a table  
chisq.test(counts)

# For imported  
  
# get counts  
 # save table as "counts"  
counts <- table(crypto$imported, crypto$year)   
  
# get rounded proportions of counts  
 # margin = 2 for column proportions  
round(prop.table(counts, margin = 2) \* 100, digits = 2)  
  
# chisq.test function requires you to input a table  
chisq.test(counts)

The proportion of urban cases is higher in 2015 compared to 2014 (12.6% vs. 6.6% respectively; p<0.01).

The proportion of imported cases is similar in 2015 compared to 2014 (46.2% vs. 48.4% respectively; p=0.6062). The median age in 2015 is 5 years (IQR: 2-10) compared to 4 years in 2014 (IQR: 1-10). The age distributions in 2015 and 2014 are not statistically significantly different from each other (p=0.236).

## Q6: Calculating incidence rates

Calculate annual incidence rates (per 100 000 population), overall, by region and by age group (as used for the annual report). Plot the incidence rates. Calculate 95% CI around the incidence rates. Discuss what the 95% CI around these incidence rates mean.

Store the incidence rates and their 95% CI in a data frame.

When calculating annual incidence rates, we need information on total populations (e.g. total population by year, by region or by age). For this case study, we will use the population information in the Excel spreadsheet: “denominators2011.xls” (see also page 4 for more information). Note that we will use 2011 denominator data for all years for this case study, assuming little population fluctuation.

## Help Q6:

There are different approaches you can take when merging denominator data and calculating rates. You could, for example, collapse first the case-based data and collapse the denominator data and then merge the two. Here we take the approach of merging the data first and then collapsing them.

Reading in denominator data from csv files.

# read in denominators by region  
region <- read.csv("region.csv", sep = ";" ,   
 stringsAsFactors = FALSE )  
  
  
# read in denominators by age  
agegroup <- read.csv("agegroup.csv", sep = ";" ,   
 stringsAsFactors = FALSE )  
  
# read in denominatros by age and region  
agegroupregion <- read.csv("agegroupregion.csv", sep = ";" ,   
 stringsAsFactors = FALSE )

### Merge the denominator data to the main dataset:

# load your dataset   
load("crypto.Rda")

Checking we have no missing values for the variable we would like to merge by:

# check region  
table(crypto$region, useNA = "always")

Merge first with region data:

# overwrite crypto with merged dataset  
 # all.x specifies crypto as the main dataset of interest  
crypto <- merge(crypto, region, by = "region", all.x = TRUE)  
  
# rename the total variable  
 # subset crypto variable names where equal to "total"  
 # overwrite with "den\_reg"  
names(crypto)[names(crypto) == "total"] <- "den\_reg"

Merge with age data:

# overwrite crypto with merged dataset  
 # all.x specifies crypto as the main dataset of interest  
crypto <- merge(crypto, agegroup, by = "age", all.x = TRUE)  
  
# rename the total variable  
 # subset crypto variable names where equal to "total"  
 # overwrite with "den\_reg"  
names(crypto)[names(crypto) == "total"] <- "den\_age"  
  
# drop those with missing denominator data  
crypto <- subset(  
 x = crypto,  
 subset = !is.na(crypto$den\_age)  
 )

Save the data:

# save your dataset  
save(crypto, file = "cryptorecodeddenom.Rda")

### Calculating annual incidence rates:

#load your cleaned dataset  
 # note that it is still called crypto  
load("cryptorecodeddenom.Rda")

First collapse the data by year (and region, as all our denominators are by region).

# sum the counts variable while grouping region, year and denom  
cryptoreg <- aggregate(count ~ region + year + den\_reg,   
 FUN = sum,   
 data = crypto)

We can now collapse again by year.

# cbind collapses count and den\_reg sepparately by year  
cryptoyear <- aggregate(cbind(count, den\_reg) ~ year,   
 FUN = sum,   
 data = cryptoreg)

Calculate the annual incidence rates with 95% CI using the *epi.conf* function from the *epiR* package.

# epiconf requires your counts and denoms to be in a matrix  
 # select columns from your cryptoyear dataset using squarebrackets  
 # change it to a matrix using as.matrix  
IR <- epi.conf(as.matrix(cryptoyear[, c("count", "den\_reg")]),  
 ctype = "inc.rate",  
 method = "exact") \* 100000

Add this to your collapsed data frame with counts using cbind

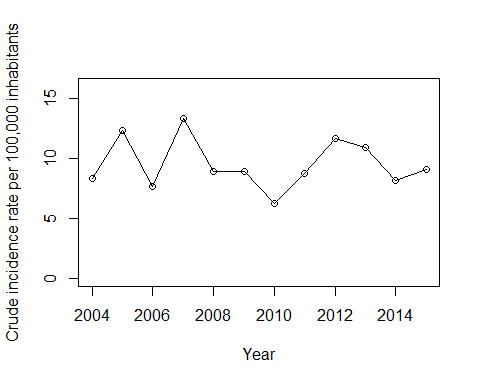
# add columns to cryptoyear from IR  
cryptoyear <- cbind(cryptoyear, IR)

Plot the incidence rates:

# plot the estimate from above  
 # type "o" specifies line graph  
plot(cryptoyear$year, cryptoyear$est, type = "o")

And you can make the plot nicer:

plot(cryptoyear$year, cryptoyear$est, type = "o",   
 ylim = c(0, 16),   
 ylab = "Crude incidence rate per 100,000 inhabitants",   
 xlab = "Year")



You can then also export your plot and save your dataset

# export plot as a png file  
dev.copy(png,'Crypto inc by year.png')  
dev.off()  
  
# save incidence rate dataset  
save(cryptoyear, file = "cryptoyear.Rda")

### Annual incidence rates by region

# load your cleaned dataset  
 # note that it is still called crypto  
load("cryptorecodeddenom.Rda")

First collapse the data by year (and region, as all our denominators are by region).

# sum the counts variable while grouping region, year and denom  
cryptoreg <- aggregate(count ~ region + year + den\_reg,   
 FUN = sum,   
 data = crypto)  
  
# sort your dataset by region   
cryptoreg <- cryptoreg[order(cryptoreg$region), ]

Calculate the annual incidence rates with 95% CI using the *epi.conf* function from the *epiR* package.

# epiconf requires your counts and denoms to be in a matrix  
 # select columns from your dataset using squarebrackets  
 # change it to a matrix using as.matrix  
IR <- epi.conf(as.matrix(cryptoreg[, c("count", "den\_reg")]),  
 ctype = "inc.rate",  
 method = "exact") \* 100000

Add this to your collapsed data frame with counts using *cbind*.

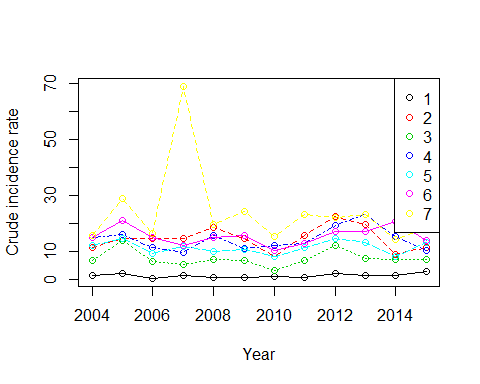
# add columns to dataset from IR  
cryptoreg <- cbind(cryptoreg, IR)

It would be possible to iteratively add one line per region at a time to a graph. However it would be better to reshape data to wide-format.

# First we need to get rid of unnecessary variables  
cryptoreg <- cryptoreg[ , c("region", "year", "est")]  
  
  
# then we can spread the data with reshape  
cryptoreg <- reshape(cryptoreg,   
 idvar = "year",   
 timevar = "region",   
 direction = "wide")

Then you can create graphs:

# use matplot to plot columns 2 to 8 of your dataset  
 # choose to plot lines with dots using pch = 1  
matplot(cryptoreg$year, cryptoreg[, 2:8],   
 type = "o",   
 col = 1:7 ,   
 pch = 1,   
 ylab = "Crude incidence rate",   
 xlab = "Year"  
 )  
  
# add a legend   
legend("topright",   
 legend = 1:7,   
 col = 1:7,   
 pch = 1)



### Annual incidence rates by age

#load your cleaned dataset  
 #note that it is still called crypto  
load("cryptorecodeddenom.Rda")

First collapse the data by year and age (remember to use the denominator for age). Remember that we do not need to collapse the denominator data:

# sum the counts variable while grouping age, agegroup, year and denom  
cryptoage <- aggregate(count ~ age + ar\_age + year + den\_age,   
 FUN = sum,   
 data = crypto)  
  
# sort your dataset by year   
cryptoage <- cryptoage[order(cryptoage$year), ]

Then in a second step collapse the data by age group and year (note that here we need to collapse the denominator data):

# cbind collapses count and den\_reg sepparately by year  
cryptoage <- aggregate(cbind(count, den\_age) ~ ar\_age + year,   
 FUN = sum,   
 data = cryptoage)

# epiconf requires your counts and denoms to be in a matrix  
 # select columns from your dataset using squarebrackets  
 # change it to a matrix using as.matrix  
IR <- epi.conf(as.matrix(cryptoage[, c("count", "den\_age")]),  
 ctype = "inc.rate",  
 method = "exact") \* 100000

Add this to your collapsed data frame with counts using *cbind*.

# add columns to dataset from IR  
cryptoage <- cbind(cryptoage, IR)

## Q7: Calculate incidence rate ratios

Calculate incidence rate ratios, 95% CI and p-values

* between years with 2015 as the reference.
* between the average of 2012-14 and 2015.
* between urban and rural areas for 2015.

Use Poisson regression.

Interpret your findings.

## Help Q7:

#load your cleaned dataset  
load("cryptoyear.Rda")

Calculate incidence rate ratios of all years compared to 2015 data:

In order to use the 2015 year as a reference, you need to use the *relevel* function.

# change 2015 to reference group using relevel function  
cryptoyear$year <- relevel(factor(cryptoyear$year), ref = "2015")

For poisson regression, we will use the glm function with a poisson family and log link.

# run poisson regression of counts by year  
model1 <- glm(count ~ year ,   
 family = poisson(link = "log"),   
 data = cryptoyear, offset = log(den\_reg))  
  
# use the tidy function from broom package to simplify the regression output  
model1clean <- tidy(model1, exponentiate = TRUE, conf.int = TRUE)

Example of interpretation: Compared to the incidence rate in 2015, the incidence in 2014 was around 10% lower, although this was not statistically significant. Compared to the incidence rate in 2015, the incidence rate in 2013 was around 21% higher (p=0.006).

### Incidence rate ratio of 2015 compared to 2012-14 data:

collapse your data producing mean counts for 2012-2014

# create a variable for current year   
cryptoyear$curryear <- NA  
cryptoyear$curryear[cryptoyear$year == 2015] <- 1  
cryptoyear$curryear[cryptoyear$year %in% c(2012:2014)] <- 0  
  
# aggregate using the mean function  
meanyear <- aggregate(count ~ curryear + den\_reg,   
 FUN = mean,   
 data = cryptoyear)  
  
#change the count variable to an integer   
 #required for poisson regression  
meanyear$count <- as.integer(meanyear$count)

# run poisson regression of counts by year  
model2 <- glm(count ~ curryear,   
 family = poisson(link = "log"),   
 data = meanyear, offset = log(den\_reg))  
  
# use the tidy function from broom package to simplify the regression output  
model2clean <- tidy(model2, exponentiate = TRUE, conf.int = TRUE)

There was a 12% reduction in incidence of Cryptosporidiosis in 2015 compared to the average of 2012-14 (p=0.077).

### Incidence rate ratio: urban vs rural areas:

# load your cleaned dataset  
 # note that it is still called crypto  
load("cryptorecodeddenom.Rda")

Subset data for 2015

# only keep 2015 counts  
crypto <- subset(crypto,   
 year == 2015)

# aggregate using the sum function  
cryptoag <- aggregate(count ~ urban + den\_reg,   
 FUN = sum,   
 data = crypto)  
  
# sum rural seperately   
 # aggregate doesnt work because only one urban row  
cryptorural <- colSums(cryptoag[1:6, 2:3])  
  
# bind sums together   
cryptosum <- rbind(cryptorural, cryptoag[7, 2:3])  
  
# change urban to binary  
cryptosum$urban <- c(0,1)

# run poisson regression of counts by year  
model3 <- glm(count ~ urban,   
 family = poisson(link = "log"),   
 data = cryptosum, offset = log(den\_reg))  
  
# use the tidy function from broom package to simplify the regression output  
model3clean <- tidy(model3, exponentiate = TRUE, conf.int = TRUE)

In 2015, there was a 76% lower incidence of Cryptosporidiosis in urban areas compared to rural areas.

## Q8: Incorporate output in to an R-markdown document (optional)

To make your work reproducible, consider combining your text, code and output in to a single document. Create an R-markdown with output to a word document.

* Open a new R-markdown document
* Add appropriate headers and text
* Incorporate code from question 7 analyses above
* Knit to a word document

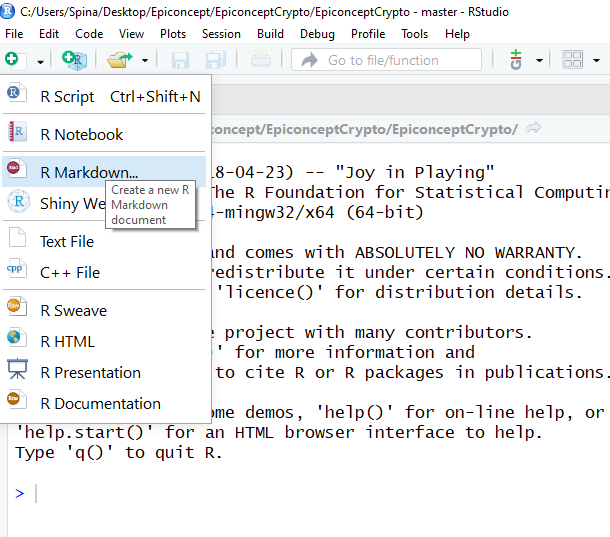
## Help Q8:

### Open a new R-markdown document

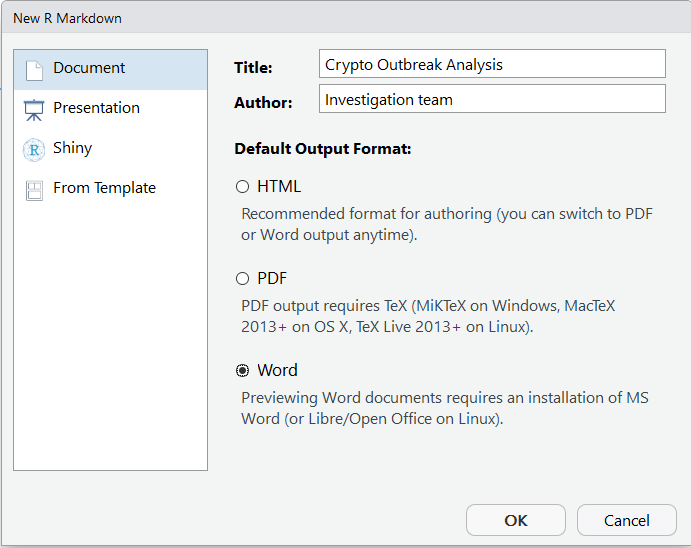
*R-markdown* documents look similar to *R-scripts*, however they are able to do more than just run R-code. Extensive documentation is available on the [RStudio website](https://rmarkdown.rstudio.com/index.html).

In order to produce output documents you will need to install [Pandoc](https://pandoc.org/installing.html) and also a LaTeX processor such as [MiKTeX](https://miktex.org/download).

Once you have done this you can create a new *R-markdown* document. Do this by clicking on the **+** drop-down menu on the top right (as in the figure below).



You will then be prompted to enter a document name, author and choose the output type. In this case we choose word.



Once you push OK, the *R-markdown* document opens and looks similar to an *R-script*. At the top is the so called YAML header, where the information you entered in the pop-up window appears. Below this is an *R-code chunk*, which specifies that all the following *R-code chunks* in this *R-markdown* document should be shown (or echoed) in the output word document. If you do not want your code to appear in your final word document, then set echo = FALSE. Below this is some example text and figures, which you can delete.



### Add headers and text

To add a header to your document, put a hashtag (#) in front of your text. As in the figure above where it says “## R Markdown”, the text will turn blue. A single hashtag will give you a header, with subsequent hashtags producing subheadings. See this [R-studio cheeatsheet](https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf) for details and other useful syntax.

Try to add the following text to your *R-markdown* document with appropriate headings, subheadings and italics:

# Background

Cryptosporidium is a protozoal parasite that causes a diarrhoeal illness in humans known as cryptosporidiosis. It is transmitted by the faeco-oral route, with both animals and humans serving as potential reservoirs. Cryptosporidiosis is a notifiable disease in many countries across the world. This report investigated cryptosporidium incidence in 2015 from surveillance data in Country X.

# Methods

## Data aquisition and structure

Routinely reported national surveillance data was used for analysis. This covered the years from 2004-2015 for the whole of Country X.

The data was case-based and thus one record designated one case of Cryptosporidium. Information on species (e.g. *C. parvum*, *C. hominis*, etc.) was not available in this dataset in country X. Due to funding restrictions, routine speciation of samples was stopped in most laboratories in country X.

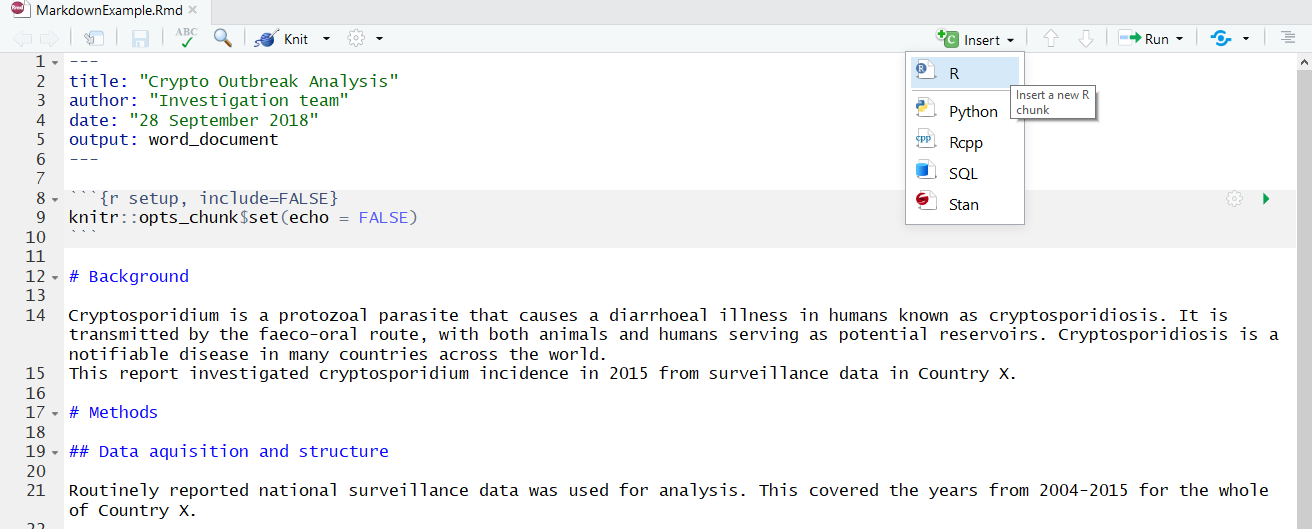
Denominator data was sourced from the office of national statistics in Country X. These population counts were available from the year 2011 and broken down by region and age group.

## Data analysis

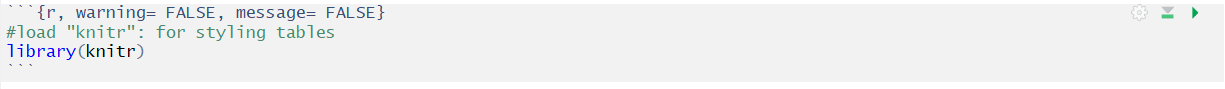
Using poisson regression, we calculated incidence rate ratios with corresponding 95% confidence intervals and p-values for three analyses. First comparing all years available using 2015 as a reference. Second, between the average of 2012-2014 and 2015. Finally, between urban and rural areas for 2015.

### Adding code and output

To insert a chunk of code to your *R-markdown* document, click on the insert button and choose *R* (as in the figure below). Alternatively you can type Ctrl+Alt+I as a keyboard shortcut.



You might at this point want to load the *knitr* package, which allows you to output nice tables from a data frame. You can specify at the top of the chunk that you do not want to echo the code in your output as well as surpressing warnings and messages from loading the package. Within the *R* code chunk, you can comment and code exactly the same way as in *R-scripts*. You also need to load packages that you require for analysis (e.g. “broom” for this section).



In between code chunks you can continue to add appropriate headers and text (e.g. add results and describe your tables). You can now add a coad chunk with the code from question 7.

#### Reading in datasets ####  
  
#load your cleaned dataset  
load("cryptoyear.Rda")  
  
  
#### Incidence rate ratios of all years compared to 2015 data ###   
  
# change 2015 to reference group using relevel function  
cryptoyear$year <- relevel(factor(cryptoyear$year), ref = "2015")  
  
# run poisson regression of counts by year  
model1 <- glm(count ~ year ,   
 family = poisson(link = "log"),   
 data = cryptoyear, offset = log(den\_reg))  
  
# use the tidy function from broom package to simplify the regression output  
model1clean <- tidy(model1, exponentiate = TRUE, conf.int = TRUE)

Then once you have your clean data frame with results, you can pass this through the *kable* function from the *knitr* package to get a nice table in your word document.

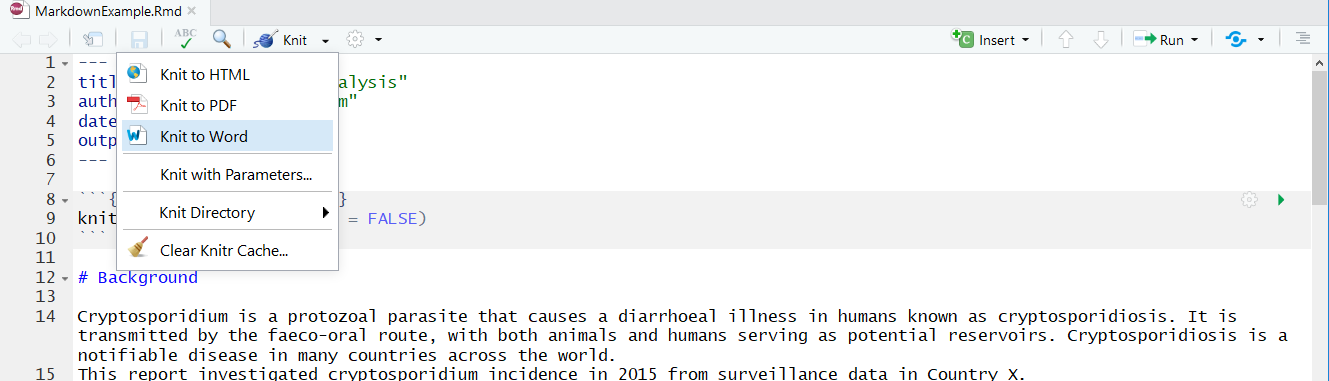
#clean output table of model  
kable(model1clean)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| (Intercept) | 0.0000906 | 0.0506370 | -183.8446688 | 0.0000000 | 0.0000819 | 0.0000999 |
| year2004 | 0.9205128 | 0.0731411 | -1.1323917 | 0.2574698 | 0.7974134 | 1.0623136 |
| year2005 | 1.3589744 | 0.0667150 | 4.5976188 | 0.0000043 | 1.1928657 | 1.5495599 |
| year2006 | 0.8461538 | 0.0747958 | -2.2334700 | 0.0255180 | 0.7305071 | 0.9795225 |
| year2007 | 1.4666667 | 0.0656685 | 5.8322049 | 0.0000000 | 1.2901552 | 1.6690736 |
| year2008 | 0.9794872 | 0.0719854 | -0.2879212 | 0.7734071 | 0.8505192 | 1.1279334 |
| year2009 | 0.9871795 | 0.0718436 | -0.1796041 | 0.8574634 | 0.8574485 | 1.1364900 |
| year2010 | 0.6923077 | 0.0791695 | -4.6447798 | 0.0000034 | 0.5923223 | 0.8079884 |
| year2011 | 0.9666667 | 0.0722262 | -0.4693803 | 0.6387978 | 0.8389716 | 1.1136711 |
| year2012 | 1.2897436 | 0.0674698 | 3.7712195 | 0.0001625 | 1.1303484 | 1.4727038 |
| year2013 | 1.2051282 | 0.0684964 | 2.7240242 | 0.0064492 | 1.0539706 | 1.3787362 |
| year2014 | 0.9000000 | 0.0735738 | -1.4320384 | 0.1521328 | 0.7789502 | 1.0394810 |

Continue to do this for the following parts of the analysis; adding text and code where appropriate.

### Knit to word document

Once you have finished with your document, you can click on the Knit button to create your word document output.



# Appendix

### Reading in Excel files

Import the dataset from excel using the *read.xlsx* function from the *xlsx* package, storing it as a data frame within *R* called crypto. Here we also specify that we do not want to read in string (character or grouped variables as factors).

#read in the dataset, specifying sheet name  
crypto <- read.xlsx("crypto.xls", sheetName = "Crypto",   
 stringsAsFactors = FALSE )

### Get summary information of your dataset

#Install the summarytools package  
install.packages("summarytools")  
  
#load the package to this   
library(summarytools)  
  
#call the dfsummary function and view its output  
 #note that view is not capatilised (summarytools function)  
view(dfSummary(crypto))

### Create annual report age groups with labels:

It is possible to create nice age categories with labels using a self-made function. This makes use of the *cut* function in combination with others such as *seq*, *upper* and *lower*.

#create an age grouping function   
  
age.cat <- function(x, lower = 0, upper, by = 10,  
 sep = "-", above.char = "+") {  
  
 labs <- c(paste(seq(lower, upper - by, by = by),  
 seq(lower + by - 1, upper - 1, by = by),  
 sep = sep),  
 paste(upper, above.char, sep = ""))  
  
 cut(floor(x), breaks = c(seq(lower, upper, by = by), Inf),  
 right = FALSE, labels = labs)  
}

Once it has been saved as a function you can use it just as any other.

#Create a variable with 5 year age bands  
crypto$agegp5 <- age.cat(crypto$age,   
 upper = 80,   
 by = 5)  
  
#create a variable with 10 year age bands   
crypto$agegp10 <- age.cat(crypto$age,   
 upper = 90,   
 by = 10)

You can alternatively make your own groupings with the following code.

# Create alternative age grouping var  
crypto$agegpalt <- as.character(cut(  
 crypto$age,  
 breaks = c(-1, 4, 49, 200),  
 labels = c("0-4", "5-49", "50+")  
 ))

### Get a contingency table with proportions

You can bind different data frames together using the *rbind* (for rows) and *cbind* (for columns) functions. These have been combined in the following *big.table* function to give the desired outputs. You can run this code below which saves the *big.table* function in your environment; then you can use it the same way any other function works.

#load the function  
big.table <- function(data, useNA = "no") {  
 count <- table(data, useNA = useNA)  
 prop <- round(prop.table(count)\*100, digits = 2)  
 cumulative <- cumsum(prop)  
 rbind(count,  
 prop,  
 cumulative)   
}

#use the big.table function for sex  
big.table(crypto$sex)