Solutions: SimOb All Injects

# Output to R script:  
knitr::knit\_hooks$set(purl = knitr::hook\_purl)

# Inject 09

## 1. Install packages and load libraries

# Check if the 'pacman' package is installed, if not install it:  
if (!requireNamespace("pacman", quietly = TRUE)) install.packages("pacman")  
  
# Load the required libraries into the current R session:  
pacman::p\_load(rio,   
 here,   
 tidyverse,   
 skimr,  
 plyr,  
 janitor,  
 lubridate,  
 gtsummary,   
 flextable,  
 officer,  
 epikit,   
 apyramid,   
 scales,  
 broom,  
 EpiStats)

## 2. Import your data

# Import the raw data set:  
copdata <- rio::import(here::here("data", "Copenhagen\_raw.csv"))

## 3. Explore and clean your data

head(copdata)

id sex age group class diarrhoea bloody vomiting abdo nausea fever  
1 1 male 18 1 2 1 0 0 1 0 NA  
2 3 female 18 1 3 NA NA NA NA NA NA  
3 5 female 17 1 1 NA NA NA 1 1 NA  
4 6 male 17 1 2 NA NA NA NA NA NA  
5 7 female 18 1 3 1 0 0 1 1 0  
6 8 male 180 1 2 1 0 0 1 0 0  
 headache jointpain starthour meal tuna tunaD shrimps shrimpsD green greenD  
1 0 0 9 1 1 2 1 2 0 0  
2 NA NA NA 1 0 0 0 0 0 0  
3 1 NA NA 1 NA NA NA NA NA NA  
4 NA NA NA 0 0 0 0 0 0 0  
5 1 0 15 1 1 2 1 2 1 2  
6 0 0 15 1 1 2 1 2 1 2  
 veal vealD pasta pastaD rocket rocketD sauce sauceD bread breadD champagne  
1 1 2 1 3 1 1 1 2 1 2 1  
2 1 1 1 3 1 3 1 3 1 3 1  
3 1 0 1 1 NA NA NA NA 1 1 0  
4 1 0 0 0 0 0 0 0 0 0 1  
5 1 2 1 2 1 2 1 2 1 2 1  
6 1 2 1 2 1 2 1 2 1 2 1  
 champagneD beer beerD redwine redwineD whitewine whitewineD dayonset  
1 1 1 3 0 0 0 0 12nov2006  
2 1 0 0 1 3 0 0   
3 0 0 0 0 0 0 0   
4 3 1 3 1 3 1 3   
5 1 1 2 0 0 1 3 12nov2006  
6 1 1 3 0 0 1 2 13nov2006

dim(copdata)

[1] 397 40

str(copdata)

'data.frame': 397 obs. of 40 variables:  
 $ id : int 1 3 5 6 7 8 9 10 11 12 ...  
 $ sex : chr "male" "female" "female" "male" ...  
 $ age : int 18 18 17 17 18 180 16 15 43 16 ...  
 $ group : int 1 1 1 1 1 1 0 1 0 1 ...  
 $ class : int 2 3 1 2 3 2 NA 1 NA 1 ...  
 $ diarrhoea : int 1 NA NA NA 1 1 NA 0 1 NA ...  
 $ bloody : int 0 NA NA NA 0 0 NA 0 NA NA ...  
 $ vomiting : int 0 NA NA NA 0 0 NA 0 NA NA ...  
 $ abdo : int 1 NA 1 NA 1 1 NA 0 NA NA ...  
 $ nausea : int 0 NA 1 NA 1 0 NA 1 1 NA ...  
 $ fever : int NA NA NA NA 0 0 NA 0 NA NA ...  
 $ headache : int 0 NA 1 NA 1 0 NA 1 1 NA ...  
 $ jointpain : int 0 NA NA NA 0 0 NA 0 1 NA ...  
 $ starthour : int 9 NA NA NA 15 15 NA NA 3 NA ...  
 $ meal : int 1 1 1 0 1 1 1 1 1 1 ...  
 $ tuna : int 1 0 NA 0 1 1 1 1 1 1 ...  
 $ tunaD : int 2 0 NA 0 2 2 2 1 2 2 ...  
 $ shrimps : int 1 0 NA 0 1 1 1 0 1 1 ...  
 $ shrimpsD : int 2 0 NA 0 2 2 2 0 2 2 ...  
 $ green : int 0 0 NA 0 1 1 1 0 NA 1 ...  
 $ greenD : int 0 0 NA 0 2 2 2 0 NA 2 ...  
 $ veal : int 1 1 1 1 1 1 1 1 1 1 ...  
 $ vealD : int 2 1 0 0 2 2 1 3 2 2 ...  
 $ pasta : int 1 1 1 0 1 1 1 1 1 1 ...  
 $ pastaD : int 3 3 1 0 2 2 2 3 2 3 ...  
 $ rocket : int 1 1 NA 0 1 1 1 1 1 1 ...  
 $ rocketD : int 1 3 NA 0 2 2 2 2 2 2 ...  
 $ sauce : int 1 1 NA 0 1 1 0 1 NA NA ...  
 $ sauceD : int 2 3 NA 0 2 2 0 1 NA NA ...  
 $ bread : int 1 1 1 0 1 1 1 1 1 1 ...  
 $ breadD : int 2 3 1 0 2 2 2 2 2 2 ...  
 $ champagne : int 1 1 0 1 1 1 1 1 1 1 ...  
 $ champagneD: int 1 1 0 3 1 1 2 1 1 1 ...  
 $ beer : int 1 0 0 1 1 1 NA 1 NA 1 ...  
 $ beerD : int 3 0 0 3 2 3 NA 1 NA 2 ...  
 $ redwine : int 0 1 0 1 0 0 1 0 1 0 ...  
 $ redwineD : int 0 3 0 3 0 0 2 0 2 0 ...  
 $ whitewine : int 0 0 0 1 1 1 1 1 NA 0 ...  
 $ whitewineD: int 0 0 0 3 3 2 1 3 NA 0 ...  
 $ dayonset : chr "12nov2006" "" "" "" ...

skimr::skim(copdata)

Data summary

|  |  |
| --- | --- |
| Name | copdata |
| Number of rows | 397 |
| Number of columns | 40 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| character | 2 |
| numeric | 38 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: character**

| skim\_variable | n\_missing | complete\_rate | min | max | empty | n\_unique | whitespace |
| --- | --- | --- | --- | --- | --- | --- | --- |
| sex | 0 | 1 | 4 | 6 | 0 | 2 | 0 |
| dayonset | 0 | 1 | 0 | 9 | 175 | 4 | 0 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | 0 | 1.00 | 216.55 | 123.41 | 1 | 112 | 216 | 320 | 435 | ▇▇▇▇▇ |
| age | 0 | 1.00 | 18.57 | 9.92 | 8 | 16 | 17 | 18 | 180 | ▇▁▁▁▁ |
| group | 0 | 1.00 | 0.96 | 0.20 | 0 | 1 | 1 | 1 | 1 | ▁▁▁▁▇ |
| class | 36 | 0.91 | 1.94 | 0.84 | 1 | 1 | 2 | 3 | 3 | ▇▁▆▁▇ |
| diarrhoea | 141 | 0.64 | 0.82 | 0.39 | 0 | 1 | 1 | 1 | 1 | ▂▁▁▁▇ |
| bloody | 200 | 0.50 | 0.03 | 0.17 | 0 | 0 | 0 | 0 | 1 | ▇▁▁▁▁ |
| vomiting | 179 | 0.55 | 0.30 | 0.46 | 0 | 0 | 0 | 1 | 1 | ▇▁▁▁▃ |
| abdo | 152 | 0.62 | 0.85 | 0.35 | 0 | 1 | 1 | 1 | 1 | ▂▁▁▁▇ |
| nausea | 170 | 0.57 | 0.76 | 0.43 | 0 | 1 | 1 | 1 | 1 | ▂▁▁▁▇ |
| fever | 223 | 0.44 | 0.26 | 0.44 | 0 | 0 | 0 | 1 | 1 | ▇▁▁▁▃ |
| headache | 174 | 0.56 | 0.63 | 0.48 | 0 | 0 | 1 | 1 | 1 | ▅▁▁▁▇ |
| jointpain | 207 | 0.48 | 0.16 | 0.37 | 0 | 0 | 0 | 0 | 1 | ▇▁▁▁▂ |
| starthour | 177 | 0.55 | 12.49 | 4.92 | 3 | 9 | 9 | 15 | 21 | ▁▇▁▆▃ |
| meal | 9 | 0.98 | 0.97 | 0.17 | 0 | 1 | 1 | 1 | 1 | ▁▁▁▁▇ |
| tuna | 16 | 0.96 | 0.71 | 0.45 | 0 | 0 | 1 | 1 | 1 | ▃▁▁▁▇ |
| tunaD | 16 | 0.96 | 1.32 | 1.00 | 0 | 0 | 2 | 2 | 3 | ▆▅▁▇▂ |
| shrimps | 17 | 0.96 | 0.67 | 0.47 | 0 | 0 | 1 | 1 | 1 | ▃▁▁▁▇ |
| shrimpsD | 17 | 0.96 | 1.34 | 1.04 | 0 | 0 | 2 | 2 | 3 | ▆▂▁▇▂ |
| green | 30 | 0.92 | 0.59 | 0.49 | 0 | 0 | 1 | 1 | 1 | ▆▁▁▁▇ |
| greenD | 30 | 0.92 | 1.14 | 1.05 | 0 | 0 | 1 | 2 | 3 | ▇▂▁▇▂ |
| veal | 15 | 0.96 | 0.89 | 0.31 | 0 | 1 | 1 | 1 | 1 | ▁▁▁▁▇ |
| vealD | 14 | 0.96 | 1.83 | 0.91 | 0 | 1 | 2 | 2 | 3 | ▂▃▁▇▃ |
| pasta | 15 | 0.96 | 0.88 | 0.32 | 0 | 1 | 1 | 1 | 1 | ▁▁▁▁▇ |
| pastaD | 15 | 0.96 | 1.81 | 0.91 | 0 | 1 | 2 | 2 | 3 | ▂▃▁▇▃ |
| rocket | 24 | 0.94 | 0.57 | 0.50 | 0 | 0 | 1 | 1 | 1 | ▆▁▁▁▇ |
| rocketD | 24 | 0.94 | 1.08 | 1.06 | 0 | 0 | 1 | 2 | 3 | ▇▂▁▆▂ |
| sauce | 42 | 0.89 | 0.42 | 0.49 | 0 | 0 | 0 | 1 | 1 | ▇▁▁▁▆ |
| sauceD | 42 | 0.89 | 0.83 | 1.06 | 0 | 0 | 0 | 2 | 3 | ▇▁▁▃▁ |
| bread | 18 | 0.95 | 0.91 | 0.29 | 0 | 1 | 1 | 1 | 1 | ▁▁▁▁▇ |
| breadD | 18 | 0.95 | 1.75 | 0.71 | 0 | 2 | 2 | 2 | 3 | ▁▂▁▇▁ |
| champagne | 25 | 0.94 | 0.87 | 0.34 | 0 | 1 | 1 | 1 | 1 | ▁▁▁▁▇ |
| champagneD | 25 | 0.94 | 1.37 | 0.93 | 0 | 1 | 1 | 2 | 3 | ▂▇▁▂▃ |
| beer | 30 | 0.92 | 0.78 | 0.42 | 0 | 1 | 1 | 1 | 1 | ▂▁▁▁▇ |
| beerD | 35 | 0.91 | 1.94 | 1.23 | 0 | 1 | 3 | 3 | 3 | ▃▂▁▂▇ |
| redwine | 50 | 0.87 | 0.23 | 0.42 | 0 | 0 | 0 | 0 | 1 | ▇▁▁▁▂ |
| redwineD | 52 | 0.87 | 0.45 | 0.92 | 0 | 0 | 0 | 0 | 3 | ▇▁▁▁▁ |
| whitewine | 31 | 0.92 | 0.73 | 0.45 | 0 | 0 | 1 | 1 | 1 | ▃▁▁▁▇ |
| whitewineD | 36 | 0.91 | 1.58 | 1.21 | 0 | 0 | 2 | 3 | 3 | ▆▅▁▅▇ |

names(copdata)

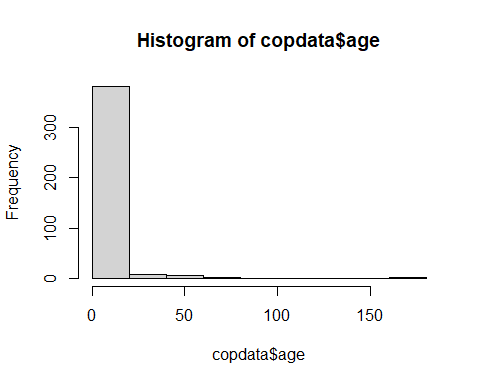
[1] "id" "sex" "age" "group" "class"   
 [6] "diarrhoea" "bloody" "vomiting" "abdo" "nausea"   
[11] "fever" "headache" "jointpain" "starthour" "meal"   
[16] "tuna" "tunaD" "shrimps" "shrimpsD" "green"   
[21] "greenD" "veal" "vealD" "pasta" "pastaD"   
[26] "rocket" "rocketD" "sauce" "sauceD" "bread"   
[31] "breadD" "champagne" "champagneD" "beer" "beerD"   
[36] "redwine" "redwineD" "whitewine" "whitewineD" "dayonset"

Let’s explore and manage some variables in detail.

#### Age

Through visual exploration of the age histogram we see that there is at least one very high value, likely implausible. You can then create a cross-tabulation of variables age and group to have a better idea of how your data looks like.

# Have a look at the histogram   
hist(copdata$age)



# Create cross-tab with the group variable:   
janitor::tabyl(dat = copdata,   
 var1 = age,   
 var2 = group)

age 0 1  
 8 0 1  
 15 0 11  
 16 1 99  
 17 0 115  
 18 0 112  
 19 0 39  
 20 0 3  
 26 1 0  
 29 1 0  
 30 1 0  
 31 1 0  
 32 1 0  
 33 1 0  
 34 1 0  
 39 1 0  
 43 1 0  
 54 1 0  
 56 1 0  
 58 2 0  
 59 1 0  
 65 1 0  
 180 0 1

Note that group is coded as 0 and 1, and these may be difficult to interpret when they mean something other than “no” and “yes”, respectively. From the codebook, you know that teachers are represented by 0, and students by 1. Let’s change this to make our lives easier:

# Convert group to a factor and label 0 as teacher, 1 as student:  
copdata <- copdata %>%   
 mutate(group = factor(group,   
 labels = c("teacher", "student")))

Now, have a look at your cross-tab again:

janitor::tabyl(dat = copdata,   
 var1 = age,   
 var2 = group)

age teacher student  
 8 0 1  
 15 0 11  
 16 1 99  
 17 0 115  
 18 0 112  
 19 0 39  
 20 0 3  
 26 1 0  
 29 1 0  
 30 1 0  
 31 1 0  
 32 1 0  
 33 1 0  
 34 1 0  
 39 1 0  
 43 1 0  
 54 1 0  
 56 1 0  
 58 2 0  
 59 1 0  
 65 1 0  
 180 0 1

With this table, we can more easily identify ages that are likely to be typographic errors. Specifically:

* There is one teacher aged 16 (likely digit reversal - should be 61)
* There is one student aged 8 (likely missing a digit - should be 18)
* There is one student aged 180 (likely has an extra digit - should be 18)

Assuming you have contacted the school to make sure your suspicions about the actual ages are correct, we can now correct them, using case\_when(). We create logical conditions to identify the incorrect ages, combining the values for age with the group they belong to:

# Update incorrect ages to the correct values with case\_when:   
copdata <- copdata %>%   
 mutate(age =   
 case\_when(   
 # Where respondent is 16 and a teacher, change age to 61:  
 age == 16 & group == "teacher" ~ 61,   
 # where respondent is 8 or 180 and a student, change age to 18:  
 age == 8 & group == "student" ~ 18,   
 age == 180 & group == "student" ~ 18,   
 # Keep remaining values as is:   
 .default = as.numeric(age)   
 # if .default is not working, try:  
 # TRUE ~ age  
 )   
 )

#### Dose response

Now let’s create a summary table of the dose response columns.

#| label: check\_dose\_cols  
#| tbl-cap: no caption  
  
# Create summary table for dose response columns:   
drtable <- copdata %>%   
 # Select all the columns with column names that end in upper case 'D':   
 select(ends\_with("D", ignore.case = FALSE)) %>%   
 # Create the summary table, excluding missing values:   
 gtsummary::tbl\_summary(missing = "no")   
  
 # Print the summary table:   
drtable

Table printed with {flextable}, not {gt}. Learn why at  
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html  
To suppress this message, include `message = FALSE` in the code chunk header.

| **Characteristic** | **N = 397**1 |
| --- | --- |
| tunaD |  |
| 0 | 110 (29%) |
| 1 | 78 (20%) |
| 2 | 155 (41%) |
| 3 | 38 (10.0%) |
| shrimpsD |  |
| 0 | 125 (33%) |
| 1 | 35 (9.2%) |
| 2 | 184 (48%) |
| 3 | 36 (9.5%) |
| greenD |  |
| 0 | 151 (41%) |
| 1 | 41 (11%) |
| 2 | 147 (40%) |
| 3 | 28 (7.6%) |
| vealD |  |
| 0 | 42 (11%) |
| 1 | 70 (18%) |
| 2 | 184 (48%) |
| 3 | 87 (23%) |
| pastaD |  |
| 0 | 44 (12%) |
| 1 | 70 (18%) |
| 2 | 183 (48%) |
| 3 | 85 (22%) |
| rocketD |  |
| 0 | 162 (43%) |
| 1 | 52 (14%) |
| 2 | 126 (34%) |
| 3 | 33 (8.8%) |
| sauceD |  |
| 0 | 206 (58%) |
| 1 | 33 (9.3%) |
| 2 | 87 (25%) |
| 3 | 29 (8.2%) |
| breadD |  |
| 0 | 35 (9.2%) |
| 1 | 51 (13%) |
| 2 | 267 (70%) |
| 3 | 26 (6.9%) |
| champagneD |  |
| 0 | 49 (13%) |
| 1 | 208 (56%) |
| 2 | 45 (12%) |
| 3 | 70 (19%) |
| beerD |  |
| 0 | 81 (22%) |
| 1 | 41 (11%) |
| 2 | 57 (16%) |
| 3 | 183 (51%) |
| redwineD |  |
| 0 | 266 (77%) |
| 1 | 31 (9.0%) |
| 2 | 21 (6.1%) |
| 3 | 27 (7.8%) |
| whitewineD |  |
| 0 | 100 (28%) |
| 1 | 73 (20%) |
| 2 | 67 (19%) |
| 3 | 121 (34%) |
| 1n (%) | |

## 4. Modify variables format

We want to modify the “variable format” or “column type” of many variables, so that we can do future calculations with them in the following injects (some functions will need a specific “type” of input).

Table 1\_Inject 09: Variable types to modify

| Variable name | Original | Desired | Hint |
| --- | --- | --- | --- |
| sex | character | factor | mutate(), as.factor() |
| class | integer | factor | mutate(), as.factor() |
| All the clinical symptom variables | integer | logical | mutate(across()), as.logical() |
| All the food variables representing the amount of specific foods eaten (those finishing with a capital “D”) | integer | factor | mutate(across()), as.factor() |
| dayonset | character | date | lubridate::dmy() |
| starthour and dayonset together | integer (starthour)  date (dayonset) | POSIXct, POSIXt | lubridate::ymd\_h() could have inside stringr::str\_glue() with dayonset and starthour |

#### Sex, group and class

Let’s start transforming one-by-one the first two variables in the table: sex, and class.

copdata <- copdata %>%   
 dplyr::mutate(  
 sex = as.factor(sex),  
 class = as.factor(class))

#### Symptoms and Food variables

For these variables, we are going to show you a couple of different ways to carry out the same variable type transformation in a *set* of variables, so you don’t need to do one variable at a time. We are showing you these ways so you see alternative ways to do the same thing.

1. For the variables that are clinical symptoms, we will list them one by one and show you the use of mutate(across( )).

copdata <- copdata %>%   
 dplyr::mutate(  
 # clinical symptoms  
 across(.cols = c(diarrhoea, bloody, vomiting,  
 abdo, nausea, fever,headache, jointpain),   
 .fns = ~ as.logical(.)  
 )  
 )

1. For the variables that are food doses, we will show you how to first create a vector of names, following by using mutate(across(all\_of( ))) on this vector.

# Create a vector with all the food variables representing the amount of specific foods items eaten (those finishing with a capital "D")  
# One way of doing it:  
food\_dose <- copdata %>%   
 dplyr::select(  
 ends\_with("D", ignore.case = FALSE)) %>%   
 names()  
  
# Another way of doing it:  
food\_dose <- c("tunaD", "shrimpsD", "greenD", "vealD",   
 "pastaD", "rocketD", "sauceD", "breadD",  
 "champagneD", "beerD", "redwineD", "whitewineD")  
  
  
copdata <- copdata %>%   
 dplyr::mutate(  
 # food dose variables  
 across(.cols = all\_of(food\_dose),   
 .fns = ~as.factor(.)))

***Note***: The tilde (~) bellow is used to apply the transformation as.logical(.) to each selected column, which in our case is either all columns included in food\_items and food\_dose.

#### Date and time variables

You can use lubridate::dmy() to mutate the dayonset variable into a date variable. Note that we are using the function dmy() because dates are formatted as day, then month (abbreviated character string), then year (i.e. “12nov2006”).

# Have a look at how the data is stored  
head(copdata$dayonset)

[1] "12nov2006" "" "" "" "12nov2006" "13nov2006"

class(copdata$dayonset)

[1] "character"

# Update copdata:  
copdata <- copdata %>%   
 # Change column to date class:  
 dplyr::mutate(  
 dayonset = lubridate::dmy(dayonset))  
  
# Check class of updated column:  
class(copdata$dayonset)

[1] "Date"

Having a variable that defines “time” in an outbreak investigation can be very useful when creating a case definition. An hour of the day, without a date associated with it doesn’t help you much, thus, you should merge together day and time of onset of symptoms into a single variable. Moreover, you will be using this combined variable later on to estimate an incubation period and create your epicurve. We can combine these two variables by using the lubridate::ymd\_h() function.

Before we proceed, it would be wise to check if any respondents have a value for dayonset but not starthour, or vice versa. The lubridate date-time conversion functions do not have an explicit argument for dealing with missing values, but the truncated = … argument can help prevent spurious date-times being derived from a date-time combination where one value is missing.

We can check if we have any missing values by cross-tabulating starthour with dayonset:

# Cross-tabulate dayonset with starthour:  
janitor::tabyl(dat = copdata,   
 var1 = starthour,   
 var2 = dayonset)

starthour 2006-11-11 2006-11-12 2006-11-13 NA\_  
 3 0 10 2 0  
 9 0 97 6 0  
 15 0 64 6 0  
 21 9 26 0 0  
 NA 2 0 0 175

This shows us that there are two respondents who had an onset date, but are missing onset time (starthour). Since starthour is represented by 1 - 2 digits, we can specify that we want lubridate to also parse date-time combinations that are truncated by up to two digits:

copdata <- copdata %>%   
 # Combine dayonset and starthour in a new date time variable:  
 mutate(onset\_datetime =   
 lubridate::ymd\_h(  
 str\_glue("{dayonset}, {starthour}"),   
 # Deal with missing starthour:  
 truncated = 2))

Warning: There was 1 warning in `mutate()`.  
ℹ In argument: `onset\_datetime = lubridate::ymd\_h(str\_glue("{dayonset},  
 {starthour}"), truncated = 2)`.  
Caused by warning:  
! 175 failed to parse.

Note that we needed to use str\_glue() to concatenate dayonset and starthour together before we could convert the variable to a date-time object. This is because the ymd\_h() function expects a single character string, containing both the date and the time, as input.

The argument truncated = 2 will result in dates with missing starthour still being converted to date-time, with the missing time being set to 00:00 (midnight). Whether you want to deal with missing starthour in this way or prefer to code these date-times as NA will depend on how you want them to be represented in your analysis.

Now we can check that everything in the new combined date-time variable has parsed correctly:

head(copdata$dayonset)

[1] "2006-11-12" NA NA NA "2006-11-12"  
[6] "2006-11-13"

head(copdata$starthour)

[1] 9 NA NA NA 15 15

head(copdata$onset\_datetime)

[1] "2006-11-12 09:00:00 UTC" NA   
[3] NA NA   
[5] "2006-11-12 15:00:00 UTC" "2006-11-13 15:00:00 UTC"

## 5. Export clean data

Save the cleaned data set before proceeding with using your case definition to identify cases in your dataset. Use the .rds format, as it preserves column classes. This ensures you will have only minimal cleaning to do after importing the data into R at the next inject.

rio::export(x = copdata,   
 file = here::here("data", "Copenhagen\_clean1.rds"))

# Inject 10

## 2. Import your data

# Import the clean data set:  
copdata <- rio::import(here::here("data", "Copenhagen\_clean1.rds"))

## 3. Identify cases

### a) Ate a meal at the school dinner

Keep in your dataset only those who ate a meal.

copdata <- copdata %>%   
 filter(meal == TRUE)

### b) Fell ill after the start of the meal

Define “fell ill” as any person having had diarrhoea with OR without blood, OR vomiting. Note that we the concept of having eaten a meal is already included as per one of the steps above:

copdata <- copdata %>%   
 mutate(gastrosymptoms = case\_when(  
 # Those had diarrhoea...  
 diarrhoea == TRUE |  
 #or bloody diarrhoea...  
 bloody == TRUE |  
 # or vomiting, are marked as TRUE (fell ill after the meal)  
 vomiting == TRUE ~ TRUE,  
 # The rest are FALSE. This includes those who ate a meal but had no symptoms (did not fell ill after the meal)  
 .default = FALSE)  
 )

### c) Fell ill within the time period of interest

Create a new meal\_datetime variable as per 11 Nov 2006, at 18:00h:

# Start with copdata:  
copdata <- copdata %>%   
 # Create new column for meal date and time:  
 mutate(meal\_datetime = lubridate::ymd\_hm("2006-11-11 18:00"))

Calculate incubation time and its median:

copdata <- copdata %>%   
 mutate(incubation = onset\_datetime - meal\_datetime,  
 incubation = as.numeric (incubation))  
  
  
median(as.numeric(copdata$incubation), na.rm = TRUE)

[1] 15

We see that the median incubation time is 15 hours. This is useful information, as incubation periods tend to be relatively pathogen-specific. We can now refine the case definition and limit the maximum incubation period from the meal time to 48 hours (2 days) after the meal, as the data points to a fast-acting bacterial toxin or a virus.

copdata <- copdata %>%   
 mutate(case = case\_when(  
 # Those who had symptoms <48h from the meal are cases (TRUE)  
 gastrosymptoms == TRUE &   
 onset\_datetime >= meal\_datetime &  
 onset\_datetime <= (meal\_datetime + days(2)) ~ TRUE,  
 # Those who had symptoms >48h from the meal are non-cases (FALSE)  
 gastrosymptoms == TRUE &   
 onset\_datetime > (meal\_datetime + days(2)) ~ FALSE,  
 # The rest are considered non-cases. Including, those who had no symptoms at all, who have missing data on the onset\_datetime variable, or who had symptoms before eating the meal   
 .default = FALSE)  
 )

Note that we may be incurring in misclassification bias with the code above. The last section indicates that if a person had clinical symptoms before eating the meal, they are considered as non-cases. However, it could be that a person had symptoms before the meal, and yet, still got infected by the pathogen when eating their meal (bad luck, we know…).

Moreover, if you remember from inject 9, there were a couple of people with an dayonset, but no starthour. The code we used (lubridate::ymd\_h with argument truncated = 2) results in dates with missing starthour being converted to date-time, with the missing time being set to 00:00 (midnight). This means that these two people don’t fulfill the case definition criteria because we marked their symptoms started early in the morning of Nov 11 (at 00:00), before the meal time (18:00), and thus, they did not “fell ill within the time period of interest”.

The two situations above are a reminder that you need to be both careful and aware of the implications of your data analysis decisions.

# Tabulate cases:  
janitor::tabyl(dat = copdata, case)

case n percent  
 FALSE 161 0.4270557  
 TRUE 216 0.5729443

Let’s have a look at how many people ate a meal, had symptoms, and were considered as cases after applying our case definition:

copdata %>%   
 summarise(atemeal = sum(meal == TRUE),  
 hadsympt = sum(gastrosymptoms == TRUE),  
 nb\_cases = sum(case == TRUE)  
 )

atemeal hadsympt nb\_cases  
1 377 216 216

# 4. Export clean data

Finally, we can save the cleaned data set before proceeding with descriptive analysis.

rio::export(x = copdata,   
 file = here::here("data", "Copenhagen\_clean2.rds"))

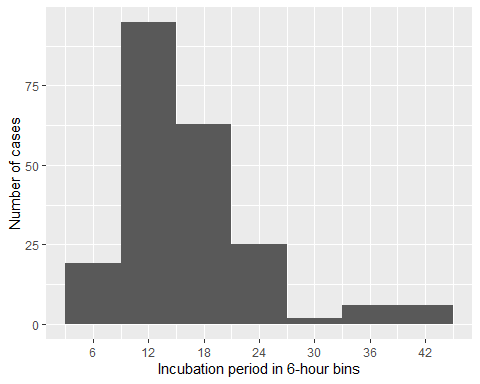
#Inject 11 ## 2. Import your data

# Import the clean data set:  
copdata <- rio::import(here::here("data", "Copenhagen\_clean2.rds"))

## 3. Time

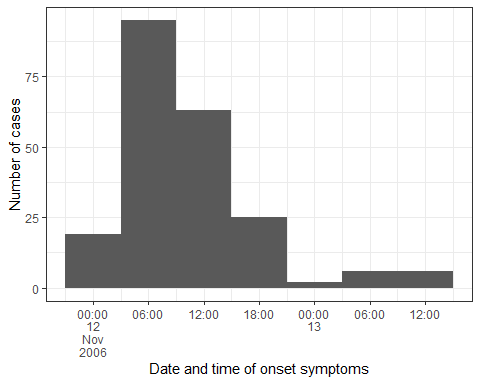
### a) Incubation period histogram

#| label: inc\_time  
  
# Create a dataset with only cases  
cases <- copdata %>%   
 filter(case == TRUE)  
  
incplot <- cases %>%   
 # Create an empty ggplot frame:  
 ggplot() +  
 # Add a histogram of incubation:  
 geom\_histogram(  
 mapping = aes(x = incubation),   
 # Set bin widths to 6 hours:  
 binwidth = 6) +  
 # Adapt scale to better fit data  
 scale\_x\_continuous(breaks = seq(0, 48, 6)) +   
 # Label x and y axes:  
 labs(x = "Incubation period in 6-hour bins",  
 y = "Number of cases")  
  
# Print plot:  
incplot

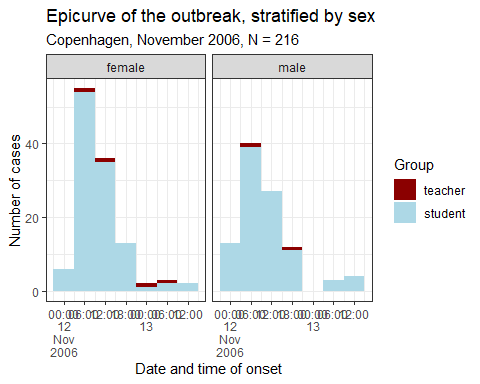


### b) Epicurve for date and time of onset

# Create a vector with sequences every 6h from the first to the last case  
breaks\_6h <- seq(from = min(cases$onset\_datetime, na.rm = TRUE),  
 to = max(cases$onset\_datetime, na.rm = TRUE),  
 by = "6 hours")  
  
# Fetch cases data:  
epicurve\_datetime <- cases %>%   
 # Add factor onset\_datetime to ggplot aesthetic:  
 ggplot(  
 mapping = aes(x = onset\_datetime)) +   
 # Add geom\_histogram:  
 geom\_histogram(  
 # Apply the vector of requences created above  
 breaks = breaks\_6h) +  
 # Adapt scale to data and adjust axis label angle:  
 scale\_x\_datetime(  
 date\_breaks = "6 hours",  
 labels = label\_date\_short()) +  
 # Update x and y axis labels:  
 labs(x = "Date and time of onset symptoms",   
 y = "Number of cases") +  
 # Remove unnecessary grid lines:  
 theme\_bw()  
  
# Print epicurve:  
epicurve\_datetime



epicurve\_strata <- cases %>%   
 # Add factor onset\_day to ggplot aesthetic:  
 ggplot(  
 mapping = aes(x = onset\_datetime, fill = group)) +   
 # Add nicer fill colours:  
 scale\_fill\_manual(values = c("darkred", "lightblue")) +  
 # Add geom\_histogram:  
 geom\_histogram(  
 # Apply the vector of requences created above  
 breaks = breaks\_6h) +  
 # Adjust x axis scales to a suitable unit:  
 scale\_x\_datetime(  
 date\_breaks = "6 hours",   
 labels = label\_date\_short()) +  
 # Update x and y axis labels:  
 labs(x = "Date and time of onset",   
 y = "Number of cases",   
 fill = "Group",   
 title = "Epicurve of the outbreak, stratified by sex",  
 subtitle = str\_glue("Copenhagen, November 2006, N = {sum(copdata$case)}")) +  
 # Stratify by sex:  
 facet\_wrap(facets = "sex",  
 ncol = 2) +  
 # Add theme:  
 theme\_bw()  
  
# Print epicurve:  
epicurve\_strata



## 4. Person

### a) Cross-tabulation of cases with group

copdata %>%   
 janitor::tabyl(case, group) %>%   
 adorn\_totals() %>%   
 adorn\_percentages() %>%   
 adorn\_pct\_formatting()

case teacher student  
 FALSE 5.6% 94.4%  
 TRUE 2.8% 97.2%  
 Total 4.0% 96.0%

### b) Cross-tabulation of cases with sex

copdata %>%   
 janitor::tabyl(case, sex) %>%   
 adorn\_totals() %>%   
 adorn\_percentages() %>%   
 adorn\_pct\_formatting()

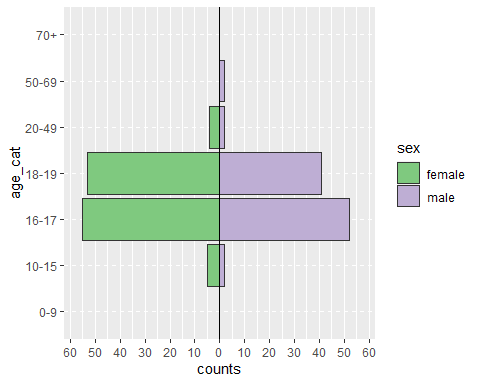
case female male  
 FALSE 59.6% 40.4%  
 TRUE 54.2% 45.8%  
 Total 56.5% 43.5%

### c) Extra - Age-sex pyramid of cases

copdata <- copdata %>%   
 # Create age categories:  
 mutate(age\_cat = epikit::age\_categories(  
 # Name of age column:  
 x = age,   
 # Define the age categories:  
 breakers = c(0, 10, 16, 18, 20, 50, 70)  
 )  
 )  
  
  
# Check age categories:  
janitor::tabyl(copdata, age\_cat)

age\_cat n percent  
 0-9 0 0.00000000  
 10-15 11 0.02917772  
 16-17 201 0.53315650  
 18-19 147 0.38992042  
 20-49 11 0.02917772  
 50-69 7 0.01856764  
 70+ 0 0.00000000

# Pipe copdata:  
agesex <- copdata %>%   
 # Filter for cases only:  
 filter(case == TRUE) %>%   
 # Create age sex pyramid:  
 apyramid::age\_pyramid(  
 # Specify column containing age categories:  
 age\_group = "age\_cat",  
 # Specify column containing sex:  
 split\_by = "sex",   
 # Don't show midpoint on the graph:  
 show\_midpoint = FALSE  
 )  
  
# Print plot:  
agesex



(Hint: change show\_midpoint = FALSE to TRUE to see skewedness in the data patterns more easily).

## 5. Symptoms

1. Summary table of symptoms, stratified by case definition

# Create summary table:  
tabsymptoms <- copdata %>%   
 # Select person characteristics to summarise:  
 select(case, diarrhoea, bloody, vomiting,  
 abdo, nausea, fever,headache, jointpain) %>%   
 # transform clinical symptoms to factors, so NA can be accounted properly in the table  
 dplyr::mutate(  
 across(.cols = c(diarrhoea, bloody, vomiting,  
 abdo, nausea, fever,headache, jointpain),   
 .fns = ~as.factor(.))) %>%  
 # Make NA a explicit level of factor variables  
 dplyr::mutate(  
 across(.cols = c(diarrhoea, bloody, vomiting,  
 abdo, nausea, fever,headache, jointpain),  
 .fns = ~forcats::fct\_na\_value\_to\_level(.))) %>%   
   
 # Create the summary table:  
 gtsummary::tbl\_summary(  
 # Stratify by case:  
 by = case,   
 # Calculate row percentages:  
 percent = "column",  
 # Create nice labels:  
 label = list(  
 diarrhoea ~ "Diarrhoea",   
 bloody ~ "Dysentary",  
 vomiting ~ "Vomiting",  
 abdo ~ "Abdominal pain",  
 nausea ~ "Nausea",   
 fever ~ "Fever",   
 headache ~ "Headache",   
 jointpain ~ "Joint pain")  
   
 ) %>%   
   
 # Add totals:  
 add\_overall() %>%   
 # Make variable names bold and italics:  
 bold\_labels() %>%   
 italicize\_labels() %>%   
 # Modify header:  
 modify\_header(  
 label = "\*\*Characteristic\*\*",  
 stat\_0 = "\*\*Overall\*\*\n \*\*N\*\* = {N}",  
 stat\_1 = "\*\*Non-case\*\*\n \*\*N\*\* = {n}",  
 stat\_2 = "\*\*Case\*\*\n \*\*N\*\* = {n}",   
 )  
  
# Print the table:  
tabsymptoms

Table printed with {flextable}, not {gt}. Learn why at  
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html  
To suppress this message, include `message = FALSE` in the code chunk header.

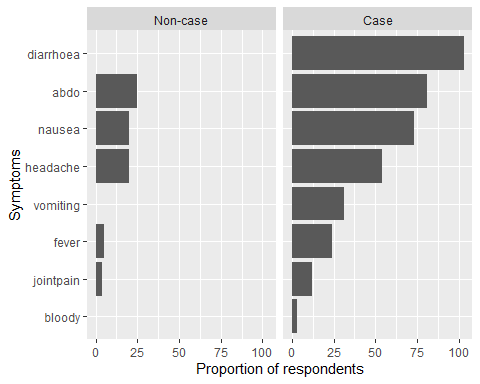
| **Characteristic** | **Overall** **N** = 3771 | **Non-case** **N** = 1611 | **Case** **N** = 2161 |
| --- | --- | --- | --- |
| ***Diarrhoea*** |  |  |  |
| FALSE | 46 (18%) | 40 (100%) | 6 (2.8%) |
| TRUE | 206 (82%) | 0 (0%) | 206 (97%) |
| ***Dysentary*** |  |  |  |
| FALSE | 189 (97%) | 42 (100%) | 147 (97%) |
| TRUE | 5 (2.6%) | 0 (0%) | 5 (3.3%) |
| ***Vomiting*** |  |  |  |
| FALSE | 149 (69%) | 42 (100%) | 107 (62%) |
| TRUE | 66 (31%) | 0 (0%) | 66 (38%) |
| ***Abdominal pain*** |  |  |  |
| FALSE | 35 (14%) | 6 (12%) | 29 (15%) |
| TRUE | 207 (86%) | 44 (88%) | 163 (85%) |
| ***Nausea*** |  |  |  |
| FALSE | 55 (25%) | 12 (26%) | 43 (24%) |
| TRUE | 169 (75%) | 34 (74%) | 135 (76%) |
| ***Fever*** |  |  |  |
| FALSE | 127 (74%) | 32 (80%) | 95 (73%) |
| TRUE | 44 (26%) | 8 (20%) | 36 (27%) |
| ***Headache*** |  |  |  |
| FALSE | 83 (38%) | 11 (25%) | 72 (41%) |
| TRUE | 137 (62%) | 33 (75%) | 104 (59%) |
| ***Joint pain*** |  |  |  |
| FALSE | 159 (85%) | 32 (84%) | 127 (85%) |
| TRUE | 29 (15%) | 6 (16%) | 23 (15%) |
| 1n (%) | | | |

1. Bar plot of symptoms stratified by case definition

# Create list of symptom variables:  
symptoms <- c("diarrhoea",   
 "bloody",   
 "vomiting",   
 "abdo",   
 "nausea",   
 "fever",   
 "headache",   
 "jointpain")  
  
# Create nice labels for case definition:  
caselabs <- ggplot2::as\_labeller(c(`FALSE` = "Non-case",   
 `TRUE` = "Case"))  
# Select variables and cases:  
symptom\_bar <- copdata %>%   
 # Select symptom columns:  
 select(case, c(all\_of(symptoms))) %>%  
 # Drop NAs:  
 drop\_na() %>%   
 # Reshape (pivot longer):  
 pivot\_longer(!case,   
 names\_to = "Symptoms",   
 values\_drop\_na = TRUE) %>%   
 # Keep only TRUE values:  
 filter(value == TRUE) %>%   
   
 # Group by symptoms and case:  
 group\_by(Symptoms, case) %>%   
 # Count for each symptom by case:  
 dplyr::summarise(count = n()) %>%   
 # Create plot:  
 ggplot(  
 mapping = aes(  
 # Order symptom bars so most common ones are ontop:  
 x = reorder(Symptoms, desc(count), decreasing = TRUE),   
 y = count)) +  
 # Display bars as proportions  
 geom\_bar(stat = "identity") +  
 # Update x axis label:  
 xlab("Symptoms") +  
 # Update y axis label:  
 ylab("Proportion of respondents") +  
 # Flip plot on its side so symptom labels are clear:  
 coord\_flip() +  
 # Facet the plot by (labelled) case:  
 facet\_wrap(facets = "case",  
 labeller = caselabs,  
 ncol = 2)

`summarise()` has grouped output by 'Symptoms'. You can override using the  
`.groups` argument.

# Print plot:  
symptom\_bar



## 6. Attack proportions

1. Overall attack proportion

# Create table of case status:  
total\_ap <- tabyl(copdata, case) %>%   
 # Add row totals:  
 adorn\_totals(where = "row") %>%   
 # Add percentages with 1 digit after the decimal point:  
 adorn\_pct\_formatting(digits = 1) %>%   
 # Filter to rows where case is TRUE:  
 filter(case == TRUE) %>%   
 # Select the column percent:  
 select(percent) %>%   
 # Extract (pull) the value from this cell:  
 pull()  
  
# Print result:  
total\_ap

[1] "57.3%"

1. Attack proportions for class, group and sex by case status

# Table to calculate attack proportions:  
attack\_prop <- copdata %>%   
 # Select columns:  
 select (case, class, group, sex) %>%   
   
 # Create table:  
 tbl\_summary(  
 # Stratified by case  
 by = case,  
 # with row percentages  
 percent = "row") %>%  
   
 # Add totals:  
 add\_overall() %>%  
   
 # Make variable names bold and italics:  
 bold\_labels() %>%   
 italicize\_labels() %>%   
   
 # Modify header:  
 modify\_header(  
 label = "\*\*Characteristic\*\*",  
 stat\_0 = "\*\*Overall\*\* \*\*N\*\* = {N}",  
 stat\_1 = "\*\*Non-case\*\* \*\*N\*\* = {n}",  
 stat\_2 = "\*\*Case\*\* \*\*N\*\* = {n}"  
 )  
  
  
# Print table:  
attack\_prop

Table printed with {flextable}, not {gt}. Learn why at  
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html  
To suppress this message, include `message = FALSE` in the code chunk header.

| **Characteristic** | **Overall** **N** = 3771 | **Non-case** **N** = 1611 | **Case** **N** = 2161 |
| --- | --- | --- | --- |
| ***class*** |  |  |  |
| 1 | 131 (100%) | 63 (48%) | 68 (52%) |
| 2 | 101 (100%) | 44 (44%) | 57 (56%) |
| 3 | 111 (100%) | 38 (34%) | 73 (66%) |
| Unknown | 34 | 16 | 18 |
| ***group*** |  |  |  |
| teacher | 15 (100%) | 9 (60%) | 6 (40%) |
| student | 362 (100%) | 152 (42%) | 210 (58%) |
| ***sex*** |  |  |  |
| female | 213 (100%) | 96 (45%) | 117 (55%) |
| male | 164 (100%) | 65 (40%) | 99 (60%) |
| 1n (%) | | | |

# Inject 14

## 2. Import your data

# Import the raw data set:  
copdata <- rio::import(here::here("data", "Copenhagen\_clean2.rds"))

## 3. Hypothesis tests for other variables

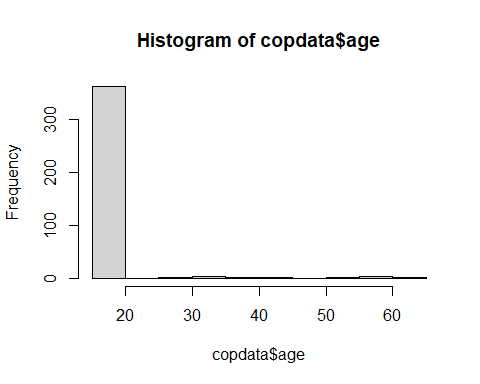
#### a) age

With the Shapiro-Wilk test we check if the variables are following the normal distribution. The null hypothesis is that the data follow a normal distribution, therefore, rejecting the null hypothesis means that the data do not follow the normal distribution. A p-value below the cutoff for rejecting the null hypothesis, e.g., a p-value<0.05 means that we reject the null hypothesis that the data follow the normal distribution. For age, the p-value is <0.05, therefore we reject the null hypothesis that the data are normally distributed. As we see in the graph most frequently reported age is <20 years.

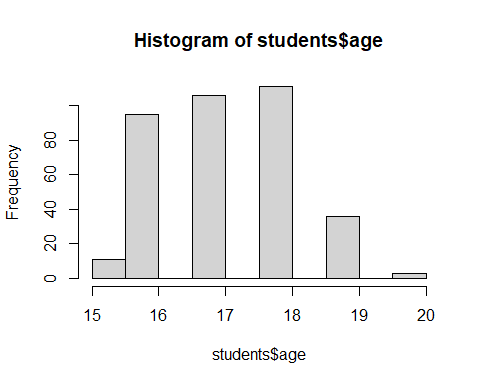
# Check if age overall follows a normal distribution:  
shapiro.test(copdata$age)

Shapiro-Wilk normality test  
  
data: copdata$age  
W = 0.31302, p-value < 2.2e-16

# Can simply have a look at  
hist(copdata$age)



# Looking only at the students:  
students <- copdata %>%   
 filter(group == "student")  
hist(students$age)



Age overall (nor within the students’ group) is not normally distributed.

We compare the age for cases and non-cases using the Wilcoxon test that is used when the data are not normally distributed. The null hypothesis is that there is no difference in the age between the two groups compared. Given that p-value>0.05 we do not reject the null hypothesis.

# Perform Wilcoxon rank sum test on age and sex:  
wilcox.test(age ~ case,   
 data = copdata)

Wilcoxon rank sum test with continuity correction  
  
data: age by case  
W = 15934, p-value = 0.1512  
alternative hypothesis: true location shift is not equal to 0

#### b) sex

copdata %>%   
 select(sex, case) %>%   
 tbl\_summary(by = case) %>%   
 add\_p()

Table printed with {flextable}, not {gt}. Learn why at  
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html  
To suppress this message, include `message = FALSE` in the code chunk header.

| **Characteristic** | **FALSE**, N = 1611 | **TRUE**, N = 2161 | **p-value**2 |
| --- | --- | --- | --- |
| sex |  |  | 0.3 |
| female | 96 (60%) | 117 (54%) |  |
| male | 65 (40%) | 99 (46%) |  |
| 1n (%) | | | |
| 2Pearson's Chi-squared test | | | |

#### c) class

copdata %>%   
 select(class, case) %>%   
 tbl\_summary(by = case) %>%   
 add\_p()

Table printed with {flextable}, not {gt}. Learn why at  
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html  
To suppress this message, include `message = FALSE` in the code chunk header.

| **Characteristic** | **FALSE**, N = 1611 | **TRUE**, N = 2161 | **p-value**2 |
| --- | --- | --- | --- |
| class |  |  | 0.090 |
| 1 | 63 (43%) | 68 (34%) |  |
| 2 | 44 (30%) | 57 (29%) |  |
| 3 | 38 (26%) | 73 (37%) |  |
| Unknown | 16 | 18 |  |
| 1n (%) | | | |
| 2Pearson's Chi-squared test | | | |

#### d) group

copdata %>%   
 select(group, case) %>%   
 tbl\_summary(by = case) %>%   
 add\_p()

Table printed with {flextable}, not {gt}. Learn why at  
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html  
To suppress this message, include `message = FALSE` in the code chunk header.

| **Characteristic** | **FALSE**, N = 1611 | **TRUE**, N = 2161 | **p-value**2 |
| --- | --- | --- | --- |
| group |  |  | 0.2 |
| teacher | 9 (5.6%) | 6 (2.8%) |  |
| student | 152 (94%) | 210 (97%) |  |
| 1n (%) | | | |
| 2Pearson's Chi-squared test | | | |

#### Let’s do all together

copdata %>%   
 select(sex, class, group, case) %>%   
 tbl\_summary(by = case) %>%   
 add\_p()

Table printed with {flextable}, not {gt}. Learn why at  
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html  
To suppress this message, include `message = FALSE` in the code chunk header.

| **Characteristic** | **FALSE**, N = 1611 | **TRUE**, N = 2161 | **p-value**2 |
| --- | --- | --- | --- |
| sex |  |  | 0.3 |
| female | 96 (60%) | 117 (54%) |  |
| male | 65 (40%) | 99 (46%) |  |
| class |  |  | 0.090 |
| 1 | 63 (43%) | 68 (34%) |  |
| 2 | 44 (30%) | 57 (29%) |  |
| 3 | 38 (26%) | 73 (37%) |  |
| Unknown | 16 | 18 |  |
| group |  |  | 0.2 |
| teacher | 9 (5.6%) | 6 (2.8%) |  |
| student | 152 (94%) | 210 (97%) |  |
| 1n (%) | | | |
| 2Pearson's Chi-squared test | | | |

## 4. Risk Ratios

The risk ratios of each food item (including the 2x2 table) are reported below. The output of the CS() command is two tables (one with the 2x2 table and one with the risk difference, the risk ratio and the attributable fraction among exposed as well as the attributable fraction among the population (and the confidence intervals for all the estimates). The Chi-square and the p-value are also reported. In the second part, a table with all the food items is printed including attack rates for exposed and unexposed as well as risk ratios and the 95% confidence intervals (CI ll and CI ul, for the lower and upper interval) and p-values.

### a) Calculate 95% CI Risk Ratios for food

# You could use the EpiStats package for each food item  
CS(copdata, "case", "tuna")

$df1  
 Cases Non Cases Total Risk  
Exposed 156 115 271 0.58  
Unexposed 60 42 102 0.59  
Total 216 157 373 0.58  
  
$df2  
 Point estimate 95%CI ll 95%CI ul  
Risk difference -0.01 -0.12 0.10  
Risk ratio 0.98 0.81 1.19  
Prev. frac. ex. 0.02 -0.19 0.19  
Prev. frac. pop 0.02 NA NA  
chi2(1) 0.05 NA NA  
Pr>chi2 0.826 NA NA

CS(copdata, "case", "shrimps")

$df1  
 Cases Non Cases Total Risk  
Exposed 150 105 255 0.59  
Unexposed 65 52 117 0.56  
Total 215 157 372 0.58  
  
$df2  
 Point estimate 95%CI ll 95%CI ul  
Risk difference 0.03 -0.08 0.14  
Risk ratio 1.06 0.87 1.28  
Attr. frac. ex. 0.06 -0.14 0.22  
Attr. frac. pop 0.04 NA NA  
chi2(1) 0.35 NA NA  
Pr>chi2 0.553 NA NA

CS(copdata, "case", "green")

$df1  
 Cases Non Cases Total Risk  
Exposed 123 93 216 0.57  
Unexposed 83 60 143 0.58  
Total 206 153 359 0.57  
  
$df2  
 Point estimate 95%CI ll 95%CI ul  
Risk difference -0.01 -0.12 0.09  
Risk ratio 0.98 0.82 1.18  
Prev. frac. ex. 0.02 -0.18 0.18  
Prev. frac. pop 0.01 NA NA  
chi2(1) 0.04 NA NA  
Pr>chi2 0.837 NA NA

CS(copdata, "case", "veal")

$df1  
 Cases Non Cases Total Risk  
Exposed 201 137 338 0.59  
Unexposed 14 22 36 0.39  
Total 215 159 374 0.57  
  
$df2  
 Point estimate 95%CI ll 95%CI ul  
Risk difference 0.21 0.04 0.37  
Risk ratio 1.53 1.01 2.32  
Attr. frac. ex. 0.35 0.01 0.57  
Attr. frac. pop 0.32 NA NA  
chi2(1) 5.64 NA NA  
Pr>chi2 0.018 NA NA

# And so one

# You can save time (and probably typos!) by creating a vector for food variables...  
food\_vars <- c("tuna", "shrimps", "green", "veal",   
 "pasta", "rocket", "sauce", "bread",  
 "champagne", "beer", "redwine", "whitewine")  
  
# ...and using EpiStats::CSTable() to run all variables together!  
CSTable(copdata, "case", food\_vars)

$df  
 Tot.Exp. Exp.Cases AR% Tot.Unex. Unex.Cases AR% RR CI ll CI ul  
pasta 338 202 59.76 36 13 36.11 1.65 1.06 2.58  
veal 338 201 59.47 36 14 38.89 1.53 1.01 2.32  
champagne 316 187 59.18 48 21 43.75 1.35 0.97 1.89  
rocket 211 114 54.03 154 95 61.69 0.88 0.73 1.04  
sauce 149 90 60.40 198 106 53.54 1.13 0.94 1.36  
beer 281 166 59.07 78 41 52.56 1.12 0.89 1.42  
redwine 80 42 52.50 259 150 57.92 0.91 0.72 1.14  
shrimps 255 150 58.82 117 65 55.56 1.06 0.87 1.28  
whitewine 260 150 57.69 98 54 55.10 1.05 0.85 1.29  
bread 342 196 57.31 29 16 55.17 1.04 0.74 1.46  
tuna 271 156 57.56 102 60 58.82 0.98 0.81 1.19  
green 216 123 56.94 143 83 58.04 0.98 0.82 1.18  
 p(Chi2)  
pasta 0.006  
veal 0.018  
champagne 0.044  
rocket 0.144  
sauce 0.202  
beer 0.303  
redwine 0.393  
shrimps 0.553  
whitewine 0.659  
bread 0.823  
tuna 0.826  
green 0.837

### b) Prepare the RR table for publication

rr\_tbl <- CSTable(copdata, "case", food\_vars) %>%   
 as.data.frame() %>%   
 rownames\_to\_column() %>%   
 flextable() %>%   
 set\_header\_labels(  
 values = c("Food Item",  
 "Total exposed",   
 "Cases exposed",   
 "AR among exposed",   
 "Total unexposed",  
 "Cases unexposed",  
 "AR among unexposed",  
 "RR",   
 "95% lower CI",   
 "95% upper CI",  
 "p-value"))

## 5. Dose Response

#### a) Pasta

# Binomial regression for RRs.   
# The outcome needs to be exponentiated so we can interpret it properly!  
binom\_pastaD <- glm(case ~ pastaD, data = copdata,   
 family = binomial(link = "log"))  
  
# To get exponentiated:  
binom\_pastaD\_exp <- glm(case ~ pastaD, data = copdata,   
 family = binomial(link = "log")) %>%   
 tidy(exponentiate = TRUE,   
 conf.int = TRUE)  
  
binom\_pastaD\_exp

# A tibble: 4 × 7  
 term estimate std.error statistic p.value conf.low conf.high  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 (Intercept) 0.361 0.222 -4.60 0.00000433 0.218 0.524  
2 pastaD1 1.50 0.247 1.65 0.0993 0.963 2.58   
3 pastaD2 1.62 0.230 2.09 0.0364 1.09 2.72   
4 pastaD3 1.86 0.234 2.64 0.00826 1.23 3.14

#### b) Veal

# Let's get the results directly exponentiated  
binom\_vealD\_exp <- glm(case ~ vealD, data = copdata,   
 family = binomial(link = "log")) %>%   
 tidy(exponentiate = TRUE,   
 conf.int = TRUE)  
  
binom\_vealD\_exp

# A tibble: 4 × 7  
 term estimate std.error statistic p.value conf.low conf.high  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 (Intercept) 0.441 0.193 -4.24 0.0000224 0.283 0.608  
2 vealD1 1.30 0.219 1.18 0.238 0.871 2.09   
3 vealD2 1.31 0.203 1.31 0.189 0.921 2.06   
4 vealD3 1.41 0.210 1.62 0.105 0.970 2.25

#### c) Champagne

# Let's get the results directly exponentiated  
binom\_champagneD\_exp <- glm(case ~ champagneD, data = copdata,   
 family = binomial(link = "log")) %>%   
 tidy(exponentiate = TRUE,   
 conf.int = TRUE)  
  
binom\_champagneD\_exp

# A tibble: 4 × 7  
 term estimate std.error statistic p.value conf.low conf.high  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 (Intercept) 0.438 0.164 -5.05 0.000000439 0.303 0.578  
2 champagneD1 1.24 0.176 1.24 0.215 0.911 1.83   
3 champagneD2 1.66 0.188 2.70 0.00684 1.17 2.48   
4 champagneD3 1.48 0.187 2.10 0.0359 1.05 2.21

# Inject 15

## 2. Import your data

# Import the raw data set:   
copdata <- rio::import(here::here("data", "Copenhagen\_clean2.rds"))

## 3. Risk Ratio

### a) Veal as exposure of interest, stratified by having eaten pasta

stratall <- copdata %>%   
 # Mutate across to convert cases to numeric:  
 mutate(across(.cols = case,   
 .fns = ~ as.numeric(.)))  
  
# Pass data to the csinter function:  
pastastrata <- csinter(x = stratall,   
 cases = "case",   
 exposure = "veal",   
 by = "pasta")  
  
pastastrata

$df1  
 CSInter case - veal by(pasta) Total Cases Risk % P.est. Stats  
1 pasta = 1 338 <NA> NA Risk difference 0.10  
2 Exposed 330 198 60.00 Risk Ratio 1.20  
3 Unexposed 8 4 50.00 Attrib.risk.exp 0.17  
4 NA <NA> NA Attrib.risk.pop 0.16  
5 pasta = 0 36 <NA> NA Risk difference 0.02  
6 Exposed 8 3 37.50 Risk Ratio 1.05  
7 Unexposed 28 10 35.71 Attrib.risk.exp 0.05  
8 NA <NA> NA Attrib.risk.pop 0.01  
9 Missing / Missing % 3 0.8% NA <NA> NA  
 95%CI-ll 95%CI-ul  
1 -0.25 0.45  
2 0.60 2.41  
3 -0.68 0.59  
4 NA NA  
5 -0.36 0.40  
6 0.38 2.92  
7 -1.65 0.66  
8 NA NA  
9 NA NA  
  
$df2  
 Point Estimate Chi2 p.value Stats 95%CI-ll 95%CI-ul  
1 Woolf test of homogeneity 0.04 0.833 NA NA NA  
2 Crude RR for veal NA NA 1.53 1.01 2.32  
3 MH RR veal adjusted for pasta NA NA 1.15 0.64 2.04  
4 Adjusted/crude relative change NA NA -25.08 NA NA

Let’s check if pasta is associated with veal (if we are thinking veal may be a confounder, we need to see if there is an association between the potential confounder (veal) and the exposure (pasta)):

# Perform Wilcoxon rank sum test on pasta and veal:  
wilcox.test(pasta ~ veal,   
 data = copdata)

Wilcoxon rank sum test with continuity correction  
  
data: pasta by veal  
W = 1496, p-value < 2.2e-16  
alternative hypothesis: true location shift is not equal to 0

### c) Champagne as exposure of interest, stratified by having eaten pasta

# Pass data to the csinter function:  
champstrata <- csinter(x = stratall,   
 cases = "case",   
 exposure = "champagne",   
 by = "pasta")