Scenario 2 - MPX in the EU

Dawson Coblin and Jose Lazo

2022-11-07

Monkeypox (MPX) Infection Rates in the European Union

This report aims to prepare data to analyze and assess the impact of MPV within
each country of the EU, and understand how case rates may differ by region and other various demographic factors. These findings will aid the implementation and appropriate response to the epidemic to local health departments.

Legend of Tasks and Milestones

Milestone	Description of Section
Milestone 1	Group Agreement
Milestone 2	Data-set Imported into R; Identify Key Data Elements and Types
Milestone 3	Clean Data-set Create Descriptive Statistics and Data Dictionary

Milestone 3 Sub-sections:
1.Sub-setting Rows and Columns as Needed- Clean individual Data Frames
2.Summarizing Data and Creating New Vectors, Organize Vectors for Use
3.Clean Variables and Combined Data-sets
4.Descriptive Stats of EU Data and Data Dictionary

1) Milestone 1: Group Agreement

Group R: Team Logistics Agreement
9/19/22

1. Project option selection

Scenario 2: Monkeypox (MPX) infection rates in the European Union

2. Create git repository

https://github.com/dcoblin/251.git

3. Roles and responsibilities

a. Team's preferred communication method

WhatsApp

b. Team's preferred meeting times and frequency

Weekends/Post 6 PM on weekdays

c. Team's preferred method for tracking progress

Google docs

d. Point person for contacting course facilitators with questions

Jose/Dawson

e. Point person for submitting all milestones

Dawson/Jose

2) Milestone 2: Data-set Imported into R; Identify Key Data Elements and Types

```
euro_mpx_cases <- read_csv("files/euro_mpx_cases.csv")
```

```
## Rows: 2987 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): CountryExp, CountryCode, Source
## dbl  (1): ConfCases
## date (1): DateRep
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
euro_census_stats <- read_csv("files/euro_census_stats.csv")
```

```
## Rows: 152534 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (8): COUNTRY_CODE, SEX, AGE, CAS, EDU, FLAGS, FOOTNOTES, RES_POP
## dbl (2): TIME, pop
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
euro_pop_denominators <- read_csv("files/euro_pop_denominators.csv")
```

```
## Rows: 603 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): DATAFLOW, LAST UPDATE, freq, indic_de, geo, OBS_FLAG
## dbl (2): TIME_PERIOD, OBS_VALUE
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
world_country_regions <- read_csv("files/world_country_regions.csv")
```

```
## Rows: 249 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (7): name, alpha-2, alpha-3, iso_3166-2, region, sub-region, intermediat...
## dbl (4): country-code, region-code, sub-region-code, intermediate-region-code
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Milestone 3: 1. Sub-setting Rows and Columns as Needed- Clean individual Data Frames

Data Frame Cleaning

```
#Cleaning the cases df
euro_mpx_cases <- euro_mpx_cases %>%
  select( DateRep, CountryExp, CountryCode  ,ConfCases)


#Creating a vector to use for country code/country name and regions in EU
euro_country_vector <- unique(euro_mpx_cases$CountryCode)

#Cleaning the denominators df
euro_pop_denominators <- euro_pop_denominators %>%
  filter( TIME_PERIOD == 2022) %>%
  rename( CountryCode = geo , country_pop_2022 = 'OBS_VALUE') %>%
  select( CountryCode, country_pop_2022)

euro_pop_denominators <- euro_pop_denominators[euro_pop_denominators$CountryCode
                                               %in% euro_country_vector, ]

head(euro_pop_denominators,3)
```

```
## # A tibble: 3 x 2
##   CountryCode country_pop_2022
##   <chr>                  <dbl>
## 1 AT                   8978929
## 2 BE                  11631136
## 3 BG                   6838937
```

```
#Clean census data to include only country code, res_pop. The date for the census
#dataframe is 2011, so the date is removed and not relevant.

euro_census_stats <- euro_census_stats %>%
  group_by(COUNTRY_CODE, RES_POP) %>%
  summarise(census_pop = sum(pop)) %>%
  rename(CountryCode = COUNTRY_CODE)
```

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

```
#Widen census info to allow each country row to have pop information.
euro_census_stats <-
  pivot_wider(euro_census_stats, names_from = RES_POP, values_from = census_pop)

euro_census_stats$`0-1000`[euro_census_stats$`0-1000` == 0] <- NA
euro_census_stats$`1000-9999`[euro_census_stats$`1000-9999` == 0] <- NA
euro_census_stats$`10000-99999`[euro_census_stats$`10000-99999` == 0] <- NA
euro_census_stats$`100000-199999`[euro_census_stats$`100000-199999` == 0] <- NA
euro_census_stats$`200000-499999`[euro_census_stats$`200000-499999` == 0] <- NA
```

```r
euro_census_stats$`500000-999999`[euro_census_stats$`500000-999999` == 0] <- NA
euro_census_stats$GE1000000[euro_census_stats$GE1000000 == 0] <- NA

head(euro_census_stats, 3)
```

```
## # A tibble: 3 x 8
## # Groups:   CountryCode [3]
##   CountryCode `0-1000` `1000-9999` `10000-99999` `100000-199999` `200000-499999`
##   <chr>          <dbl>       <dbl>         <dbl>           <dbl>           <dbl>
## 1 AT           3129102     3261222       1850964          687206          841610
## 2 BE           1053628     1819567       1931716          441535         1173542
## 3 BG           1425409     2360889       3149748          599457         1318768
## # ... with 2 more variables: `500000-999999` <dbl>, GE1000000 <dbl>
```

Milestone 3: 2. Summarizing Data and Creating New Vectors, Organize Vectors for Use

```
#We then assign counties with a region code in a new column a list "EU_region,"
#and include the country name and code
sr_northern_e<- c(
  "Denmark" , "Estonia", "Finland", "Iceland", "Ireland", "Latvia", "Lithuania",
  "Norway", "Sweden" , "DK" , "EE", "FI", "IE", "IS", "LT", "LU", "NO", "SE"
)
sr_western_e<- c(
  "Austria", "Belgium", "France", "Germany", "Luxembourg", "Netherlands" ,
  "DE", "AT" , "BE" , "FR", "LV", "NL"
)
sr_eastern_e<- c(
  "Bulgaria", "Czechia", "Hungary", "Poland", "Romania", "Slovakia", "BG", "CZ",
  "HU", "PL" , "RO", "SK"
)
sr_southern_e<- c(
  "Croatia", "Greece", "Italy", "Malta", "Portugal", "Slovenia", "Spain", "EL",
  "ES", "HR", "IT", "MT", "PT", "SI"
)
sr_western_a<- c(
  "Cyprus" , "CY"
)

#Create a floor month for the date range to aggregate by month
euro_mpx_cases <- euro_mpx_cases %>%
  mutate(floor_month = floor_date( DateRep , "month"))

#Review the aggregate totals as a value of confirmed cases in Europe and list as
#"cases_grouped_monthly"

euro_mpx_cases <- euro_mpx_cases %>%
  group_by(floor_month, CountryCode) %>%
  summarize(cases_grouped_monthly = sum(ConfCases))
```

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

```
#We now add the EU_Region vector based on the country name.
euro_mpx_cases <- euro_mpx_cases %>%
  mutate(EU_region = case_when(
    CountryCode %in% sr_eastern_e ~ "Eastern Europe",
    CountryCode %in% sr_northern_e ~ "Northern Europe",
    CountryCode %in% sr_southern_e ~ "Southern Europe",
    CountryCode %in% sr_western_e ~ "Western Europe",
    CountryCode %in% sr_western_a ~ "Western Asia"
    ))

head(euro_mpx_cases,3)
```

```
## # A tibble: 3 x 4
```

```
## # Groups:   floor_month [1]
##   floor_month CountryCode cases_grouped_monthly EU_region
##   <date>      <chr>                       <dbl> <chr>
## 1 2022-05-01  AT                              2 Western Europe
## 2 2022-05-01  BE                             16 Western Europe
## 3 2022-05-01  BG                              0 Eastern Europe
```

Milestone 3: 3. Clean Variables and Combined Data-sets

```
euro_mpx_cases <-  inner_join( euro_pop_denominators, euro_mpx_cases,
                               by= "CountryCode")
#Join census data.
euro_mpx_cases <- inner_join(euro_census_stats, euro_mpx_cases,
                             by= "CountryCode")
#Rearrange column order.
euro_mpx_cases <- relocate(euro_mpx_cases, EU_region, floor_month,
                           country_pop_2022 , cases_grouped_monthly,
                           .before = CountryCode)

#Combined the denominations data for 2022 pop with euro_mpx_cases dataset.
sum_region_cases <- euro_mpx_cases %>% group_by(EU_region) %>%
  summarise(sum(cases_grouped_monthly))

#Join the numeric sums into the main dataframe.
euro_mpx_cases <- inner_join(sum_region_cases, euro_mpx_cases,
                             by= "EU_region")

#Relocate the columns to a adjust fluidity of the data.
euro_mpx_cases<- relocate(euro_mpx_cases, floor_month, CountryCode,
                           country_pop_2022, cases_grouped_monthly,
                           .before = EU_region)
#Rename the sum of region cases that are confirmed.
euro_mpx_cases<- euro_mpx_cases %>%
   rename( sum_region_cases = `sum(cases_grouped_monthly)`)

head(euro_mpx_cases,3)
```

```
## # A tibble: 3 x 13
##   floor_month CountryCode country_pop_2022 cases_grouped_monthly EU_region
##   <date>      <chr>                  <dbl>                  <dbl> <chr>
## 1 2022-05-01  BG                   6838937                      0 Eastern Europe
## 2 2022-06-01  BG                   6838937                      3 Eastern Europe
## 3 2022-07-01  BG                   6838937                      1 Eastern Europe
## # ... with 8 more variables: sum_region_cases <dbl>, `0-1000` <dbl>,
## #   `1000-9999` <dbl>, `10000-99999` <dbl>, `100000-199999` <dbl>,
## #   `200000-499999` <dbl>, `500000-999999` <dbl>, GE1000000 <dbl>
```

Milestone 3: 4. Descriptive Stats of EU Data and Data Dictionary

Descriptive stats:

We find that the census information for 7 of the 29 countries in the EU are not available. These "0" values were replaced with "NA" to prevent graphing issues.

Region Data:

We see that the regions in EU include various numbers of countries, thus having unequal populations. A useful metric to create a graph will be the calculation of cases per 100,000. This will allow us to use 1 categorical variable (EU Region) to a quantitative variable (number of confirmed cases)

We compiled the relevant confirmed case information per region in the EU.

The pertinent information per EU region is listed below:

Table 2: Monkeypox Cases Per Region

EU Region	Total Cases Per Region	Total Region Population	Countries Per Region
Eastern Europe	277	89171711	6
Northern Europe	619	37518701	9
Southern Europe	7958	133879004	7
Western Asia	4	904705	1
Western Europe	8214	191156200	6

Data Dictionary

Table 3: Data Dictionary

	Data Type	Description
floor_month	Date	Reporting time period with the floor month
CountryCode	character	Abbreviation of Country
country_pop_2022	numeric	The current population of the country for 2022
cases_grouped_monthly	numeric	Total confirmed MPX cases grouped by month
EU_region	character	The designated region the country is located
sum_region_cases	numeric	Sum of the four months per country code
0-1000	numeric	The Sum of population 0-1000
1000-9999	numeric	The Sum of population 1000-9999
10000-99999	numeric	The Sum of population 10000-99999
100000-199999	numeric	The Sum of population 100000-199999
200000-499999	numeric	The Sum of population 200000-499999
500000-999999	numeric	The Sum of population 500000-999999
GE1000000	numeric	The Sum of population GE1000000