AMMnet meeting, Dakar
2025

Developing data visualization dashboards in R

Demba Kande
Sarja Jarjusey
Hayley Thompson

PATH

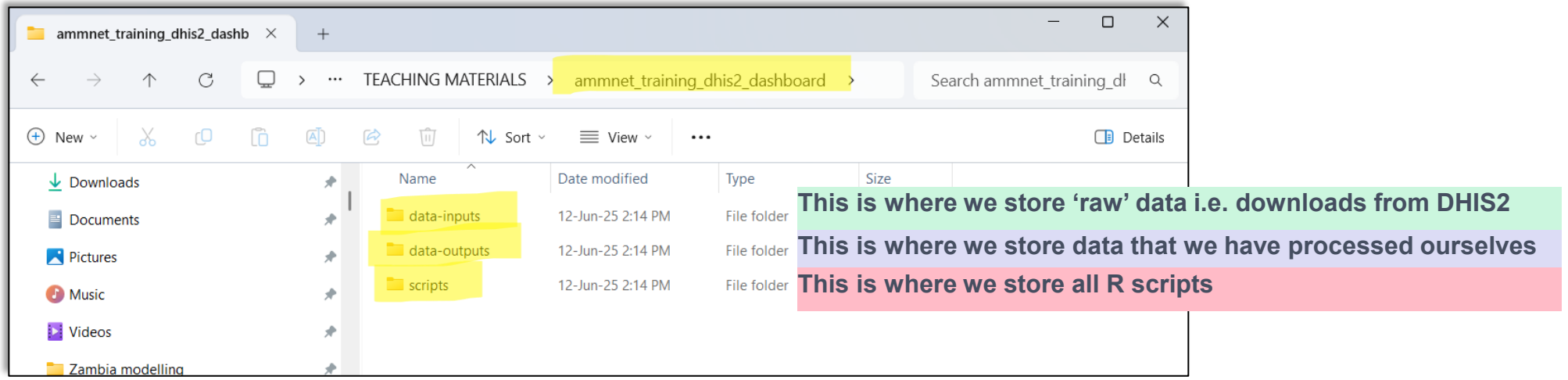
Preparation for the session while we wait to start

1) ENSURE YOU HAVE **INTERNET ACCESS**

2) ENSURE YOU HAVE **R** AND **RSTUDIO** DOWNLOADED

3) MAKE A FOLDER IN A SENSIBLE PLACE ON YOUR COMPUTER CALLED SOMETHING LIKE **'ammnet_training_dhis2_dashboard'**

4) Within this folder make a three subfolders



5) If you know how to create a **project** in R, do so in this same folder (ask a facilitator or neighbor for help if not)

PATH

Structure of the workshop

Session 1: How to manually pull data from DHIS2 and clean it ready for use in a dashboard

Session 2: Developing a simple Rshiny dashboard to visualize routine malaria data

PATH

Introduction to DHIS2

DHIS2 in an open-source, web-based **health management information system** (HMIS) platform developed by the University of Oslo. It supports data collection, analysis, visualization, and reporting.

It has been adopted by ministries of health in **over 70 countries**, especially across **Africa, Asia, and Latin America**. Commonly used at **national and sub-national levels** for routine health data reporting.

Key features of DHIS2 are:

Captures data from **health facilities and programs** (e.g., malaria, HIV, maternal health)

Enables **real-time dashboards, maps, and charts** for decision-making

Supports **integrated, decentralized data entry** across the health system

Facilitates **monitoring, evaluation, and planning** through tailored analytics

PATH

Plan for session 1

- Learn how to access DHIS2
- Explore some existing dashboards to see graphical functionality
- Learn how to make a data query to access the data elements, time period and geographical units of interest
- Download data as a csv
- Read data into R
- Clean data, including name cleaning, visual inspection of outliers/missing data, cross-referencing different variables

The outcome of this session is a cleaned malaria dataset that we'll use for part 2 (building a shiny dashboard)

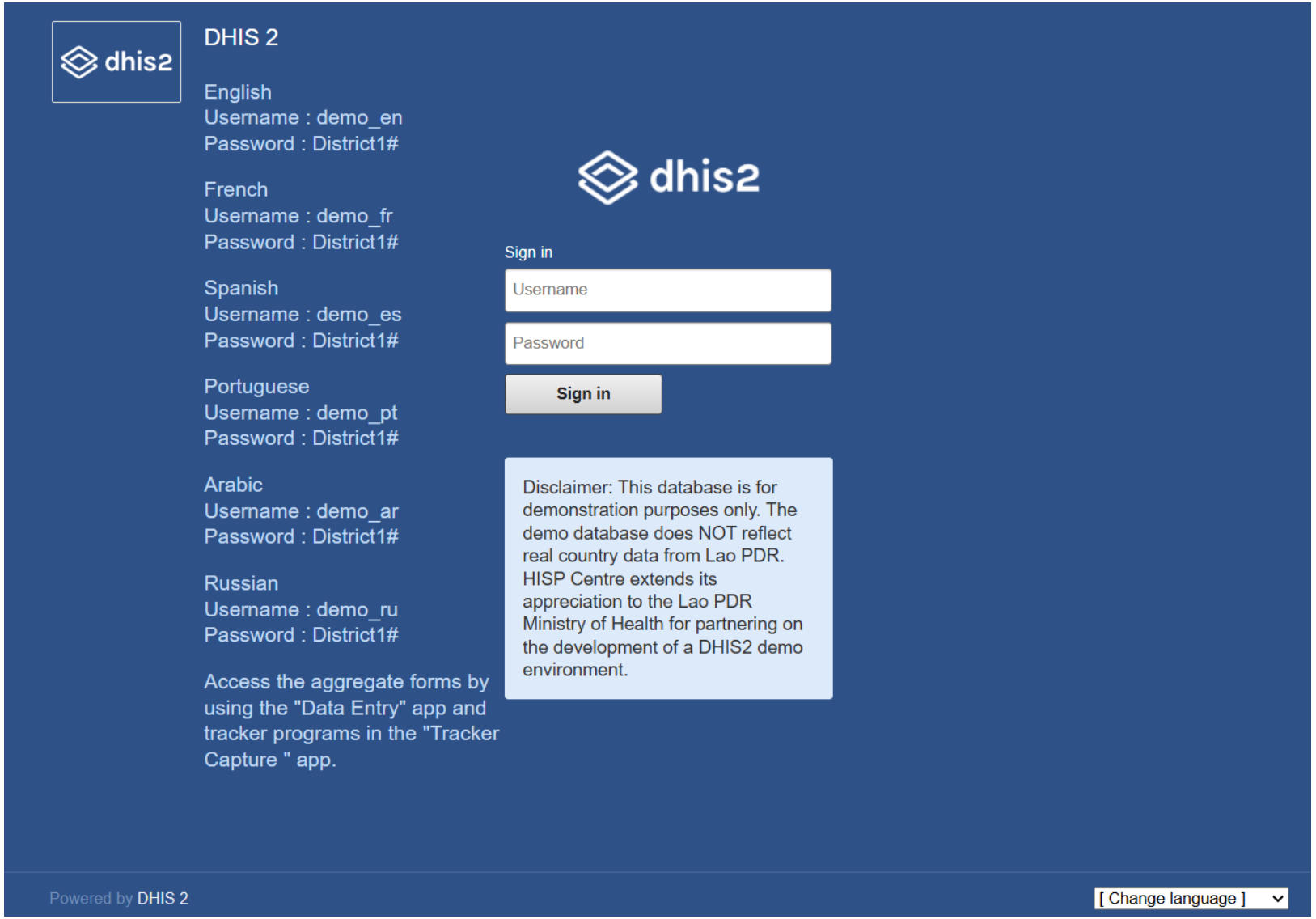
PATH

Navigating around DHIS2

DHIS2 has a **demonstration database** available based on Lao PDR but containing made-up data

This is what we'll be using in today's workshop

https://demos.dhis2.org/hmis/

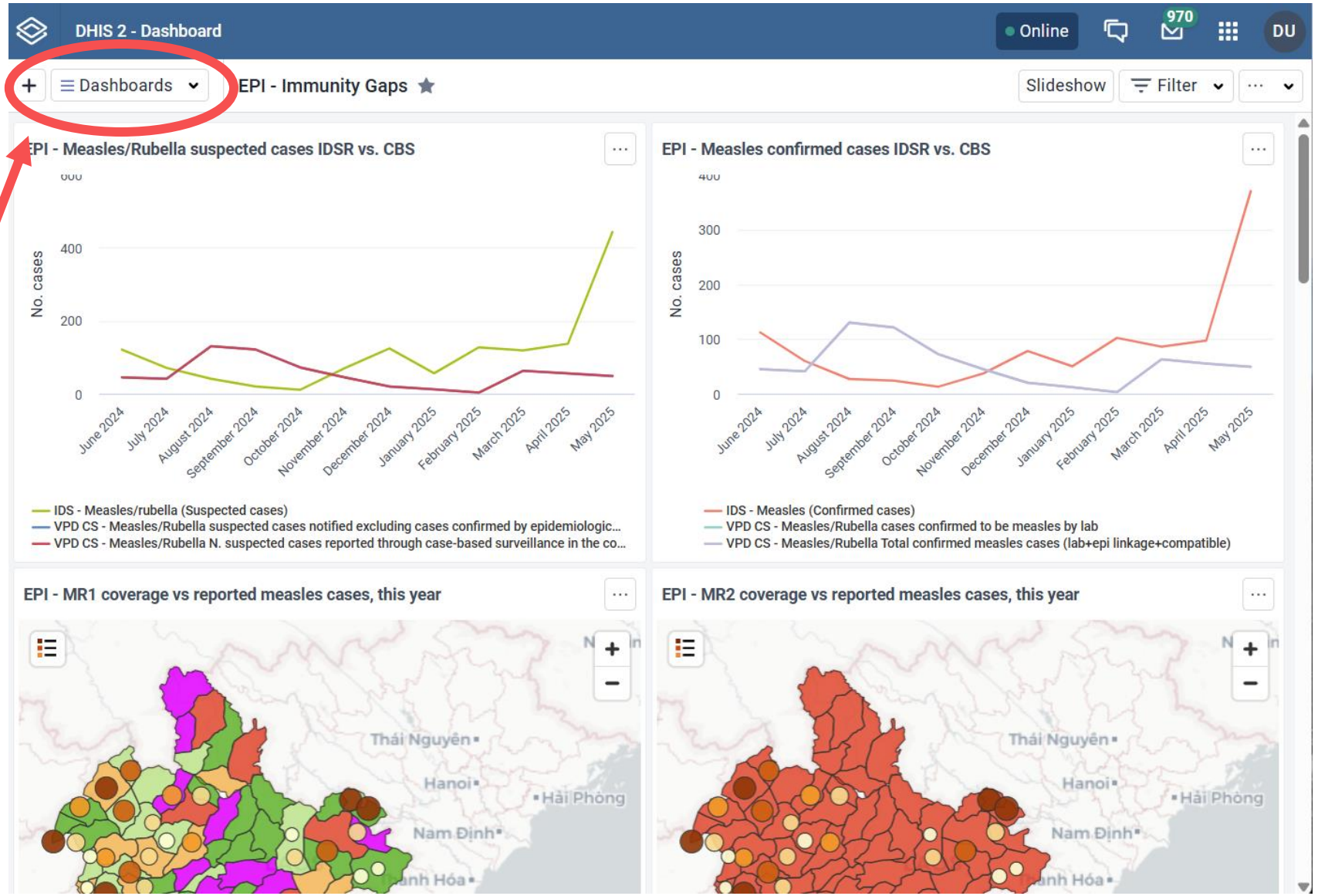


PATH

Existing dashboards

When you first open this instance of DHIS2 you will see some pre-made dashboards on vaccine-preventable diseases and vaccine coverage rates

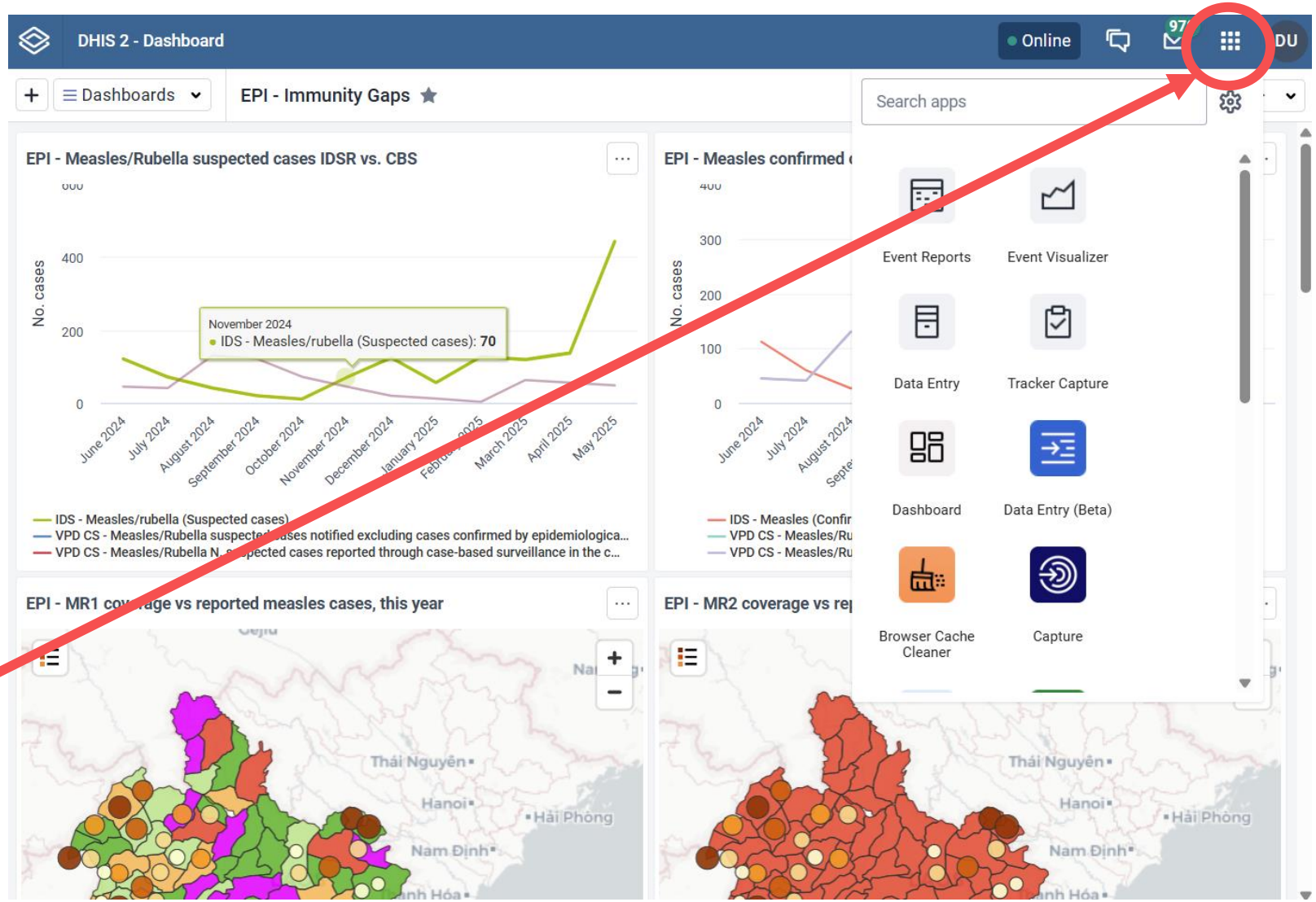
You can explore other pre-made dashboards to see what is possible within DHIS2



Accessing raw data

DHIS2 does have lots of visualization capabilities, however, we often want to extract data from DHIS2 to conduct our own **bespoke analyses** – i.e. a risk stratification or a statistical model looking at intervention impact

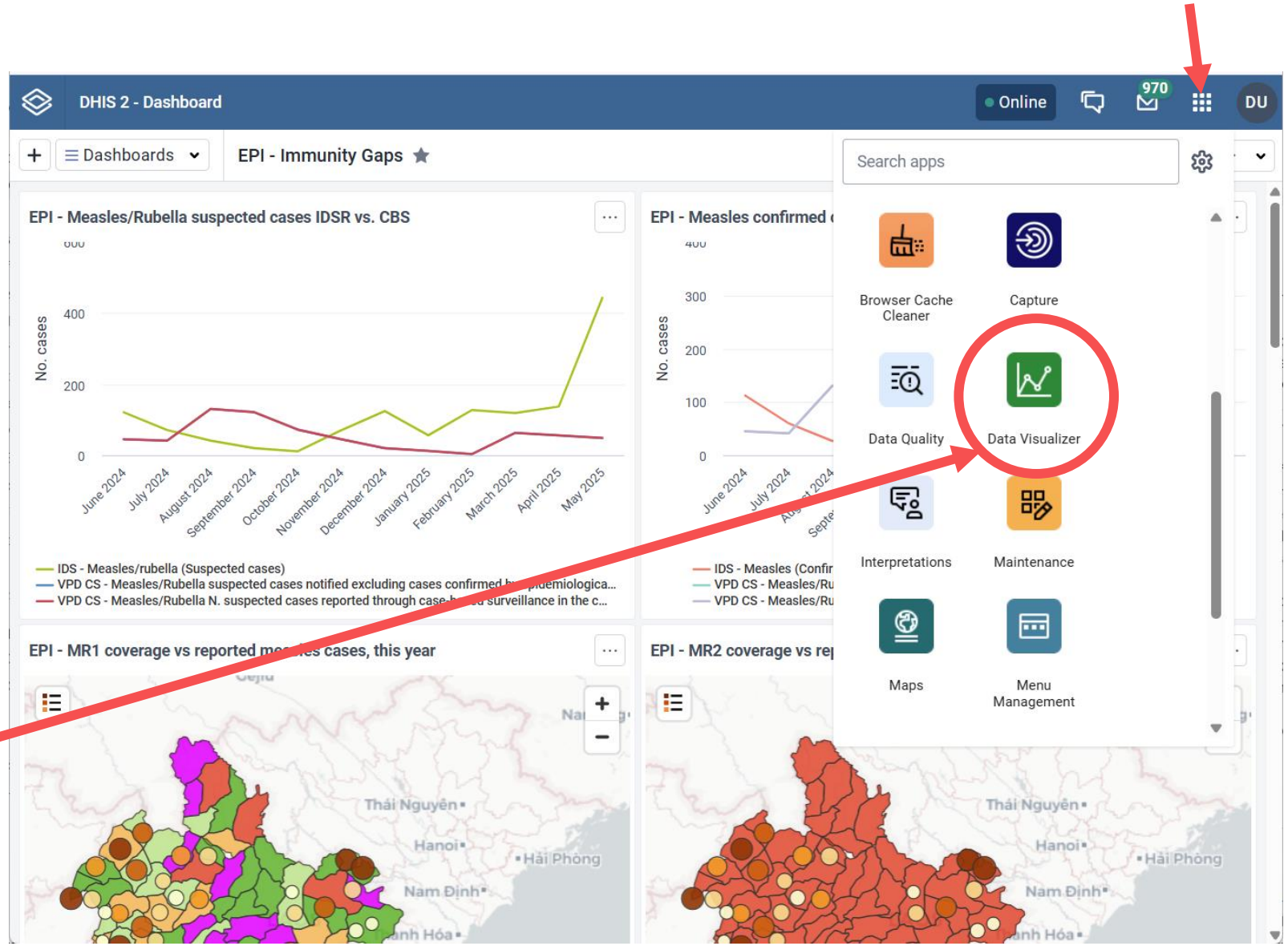
To access the raw data, you need to click this 3x3 grid



Finding data visualizer

DHIS2 does have lots of visualization capabilities, however, we often want to extract data from DHIS2 to conduct our own **bespoke analyses** – i.e. a risk stratification or a statistical model looking at intervention impact

Scroll down this menu and select **‘data visualizer’**



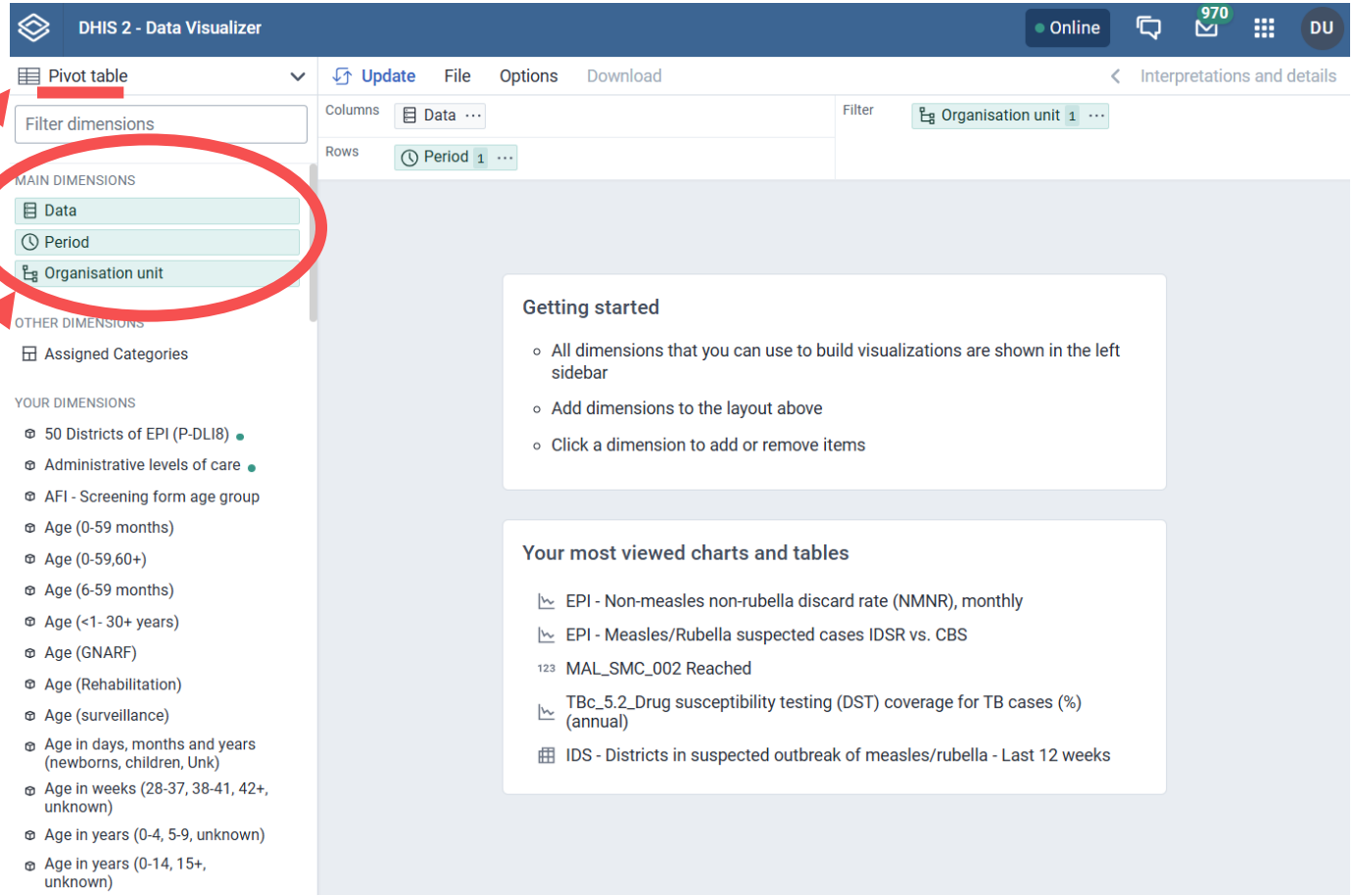
Navigating data visualizer

You should now see this screen – there are several important components of this page

For extracting raw data, we want to use the **pivot table**, but there are options here to present data in different graph formats

This is where you **select the data**, period and organisation unit (i.e. Health facility, ward, district etc.

We'll now look at these one by one on the next slides



Data dimension

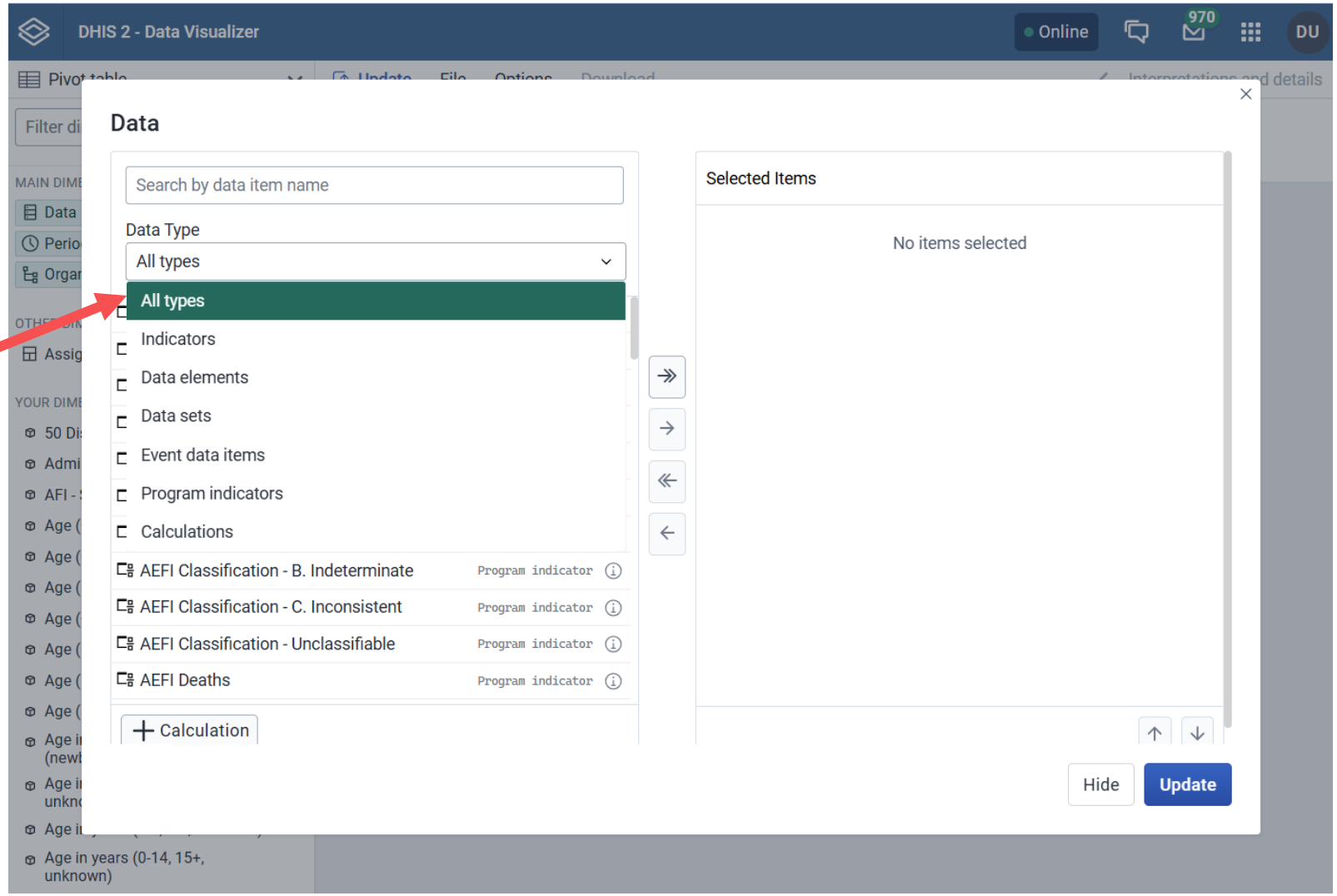
Click on the ‘data’ option to see this menu

Here you can see there are a range of ‘data types’

Today we’ll focus on

- **Data elements** – these are the building blocks of data in DHIS2 – its typically a ‘count’, i.e. malaria cases, or RDTs conducted

- **Indicators** – these are calculated based on a combination of the data elements i.e. Number of tests conducted = # RDT + # microscopy, incidence = cases/population



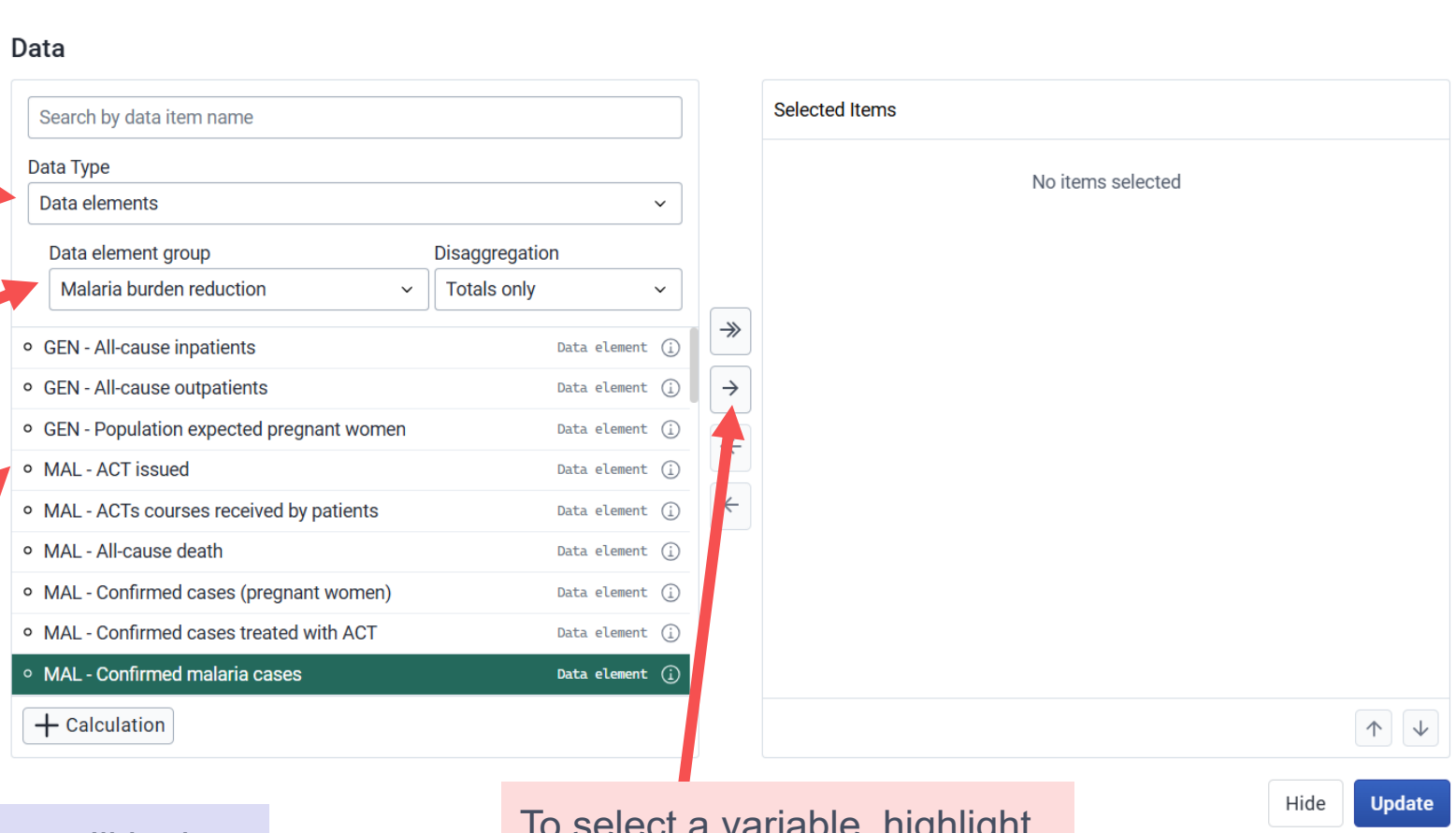
Finding malaria data (1)

Avoid the malaria indicators with a 'CH' prefix – these relate to cases detected via community case management – today we'll focus on malaria cases reported at health facilities

Firstly, select **data elements**

Next select **malaria burden reduction** in the data element group box

Here we see a wide range of malaria burden variables

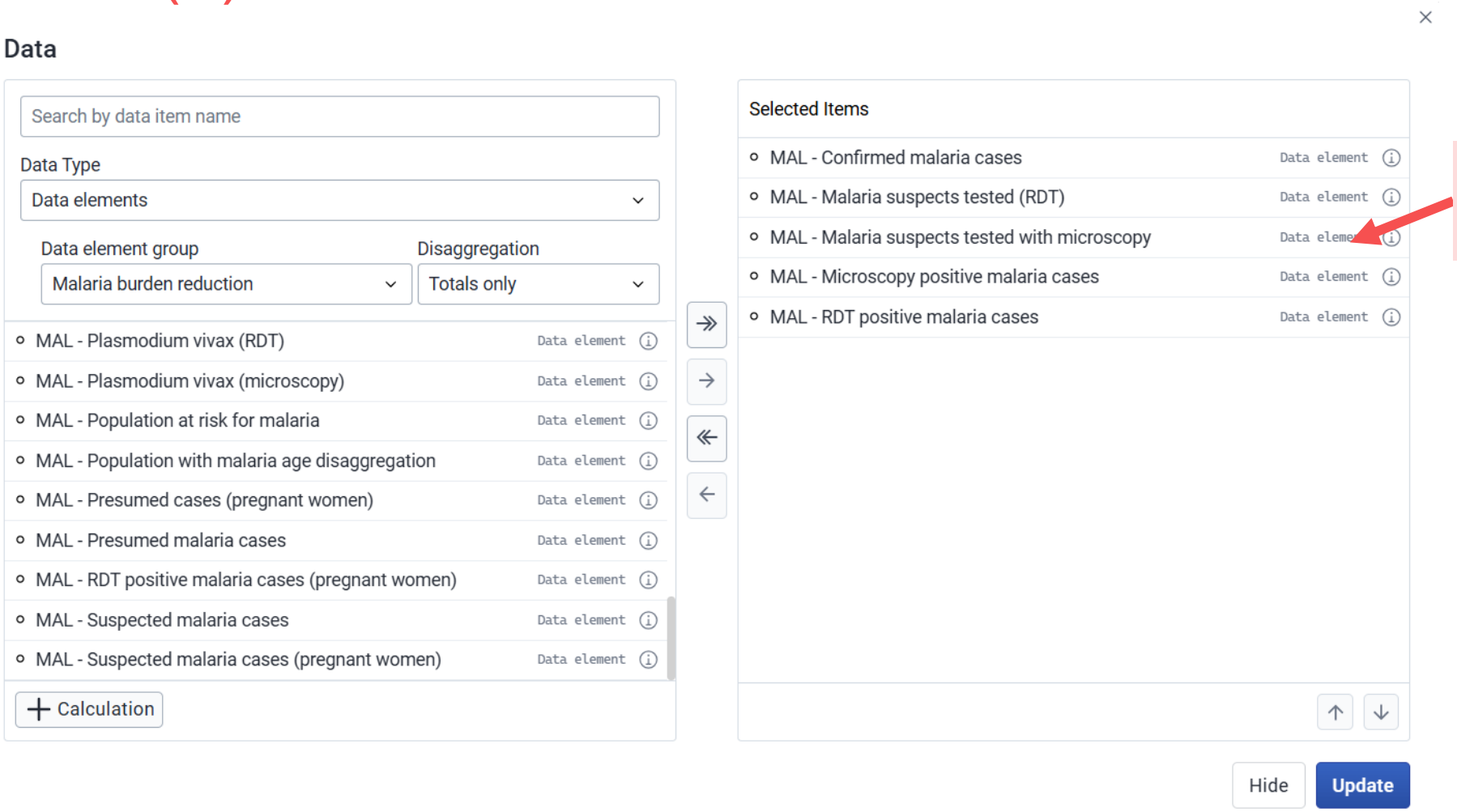


For todays example, select, we'll look at **confirmed malaria cases and malaria tests** (RDTs and microscopy) conducted

To select a variable, highlight it as shown here, then press the top 'across' arrow to move it into the **selected items** section

PATH

Finding malaria data (2)

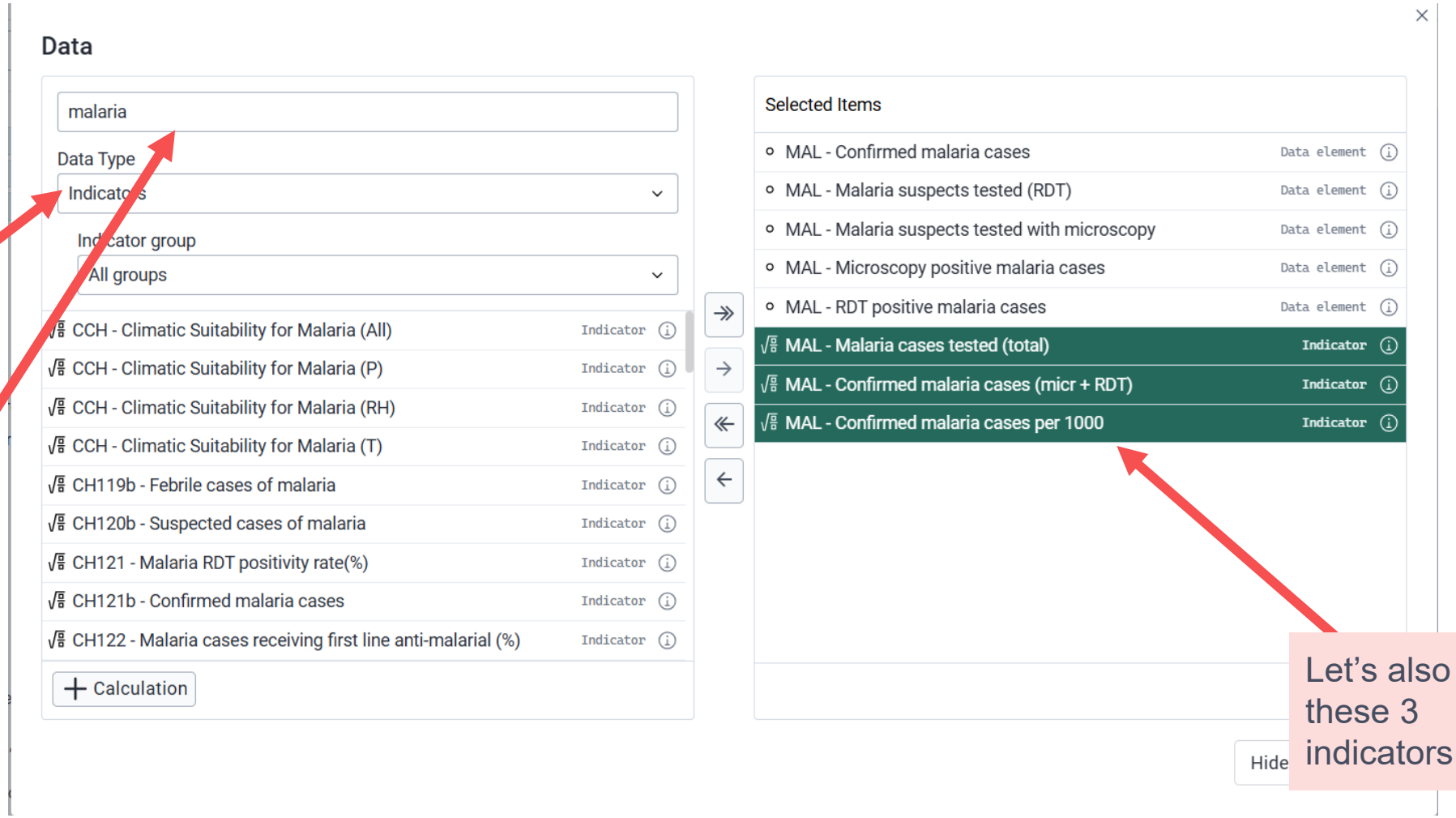


Select these 5 malaria data elements

Finding malaria data (3)

We'll also use some malaria indicators

Another way to find your indicators is simply to type a keywork in the top box



Let's also select these 3 indicators

A note on finding the ‘right’ variables

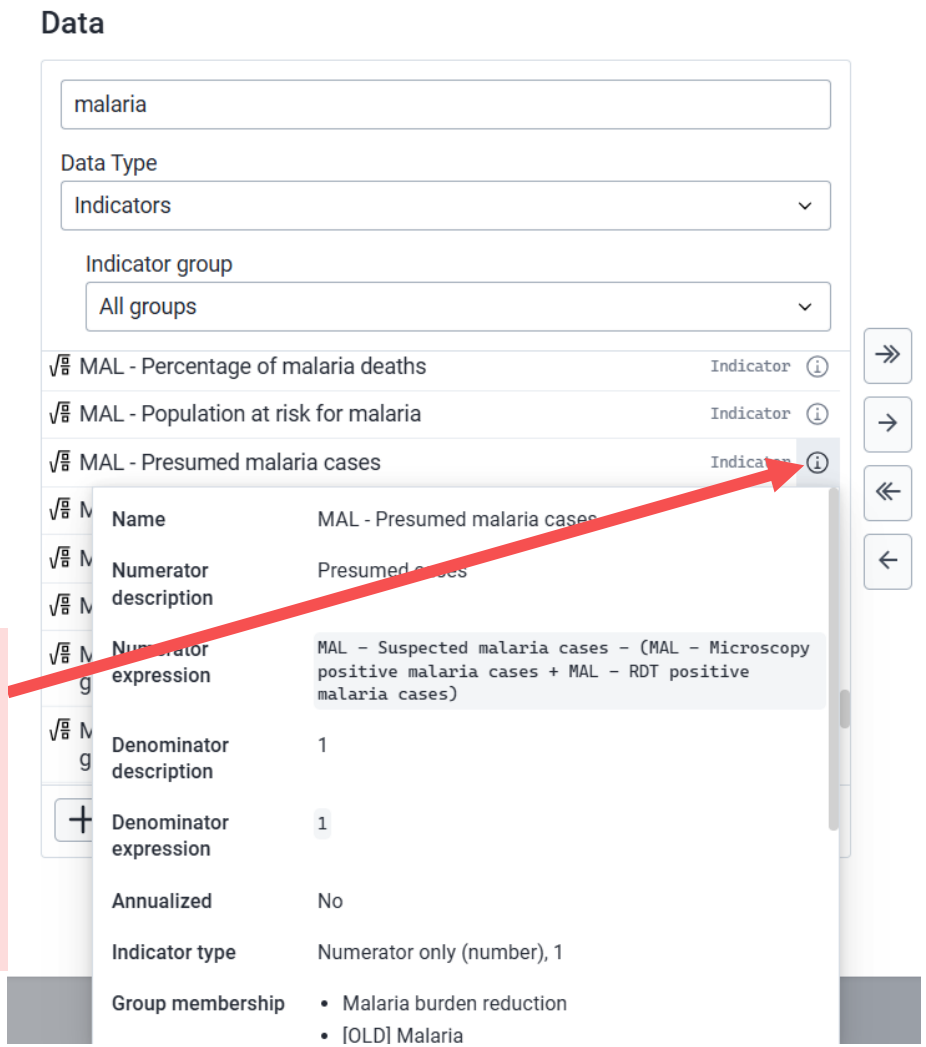
There are typically a lot of variables that relate to malaria burden, and not all countries regularly report all variables – we normally start by asking a local expert which elements/indicators to use

Failing that, we can manually interrogate the data to see which are reported

Feel free to select a much wider range of malaria variable to explore whether they’re all useful, and how they relate to each other

There are also sometimes duplicate variables, for example, this demo DHIS2 has 2 identical indicators for total malaria cases

You can click the circled ‘*i*’ Here to get more details on each variables

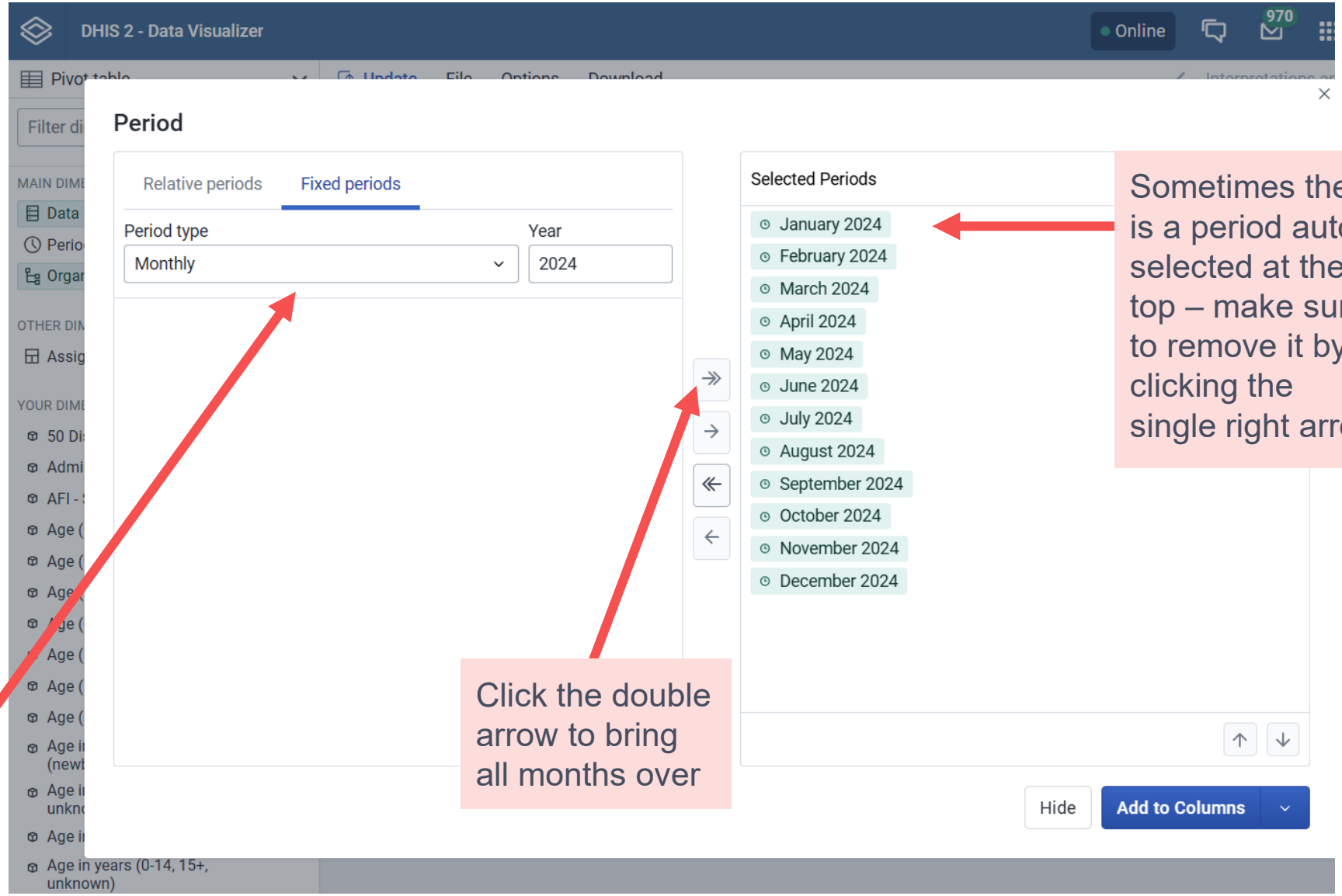


Selecting time periods

You can either select a period relative to today (i.e. last 3 months), or you can select fixed periods

In this example, we're going to look at monthly data from 2024 only

First select 'fixed periods' and then select 'monthly' and '2024'



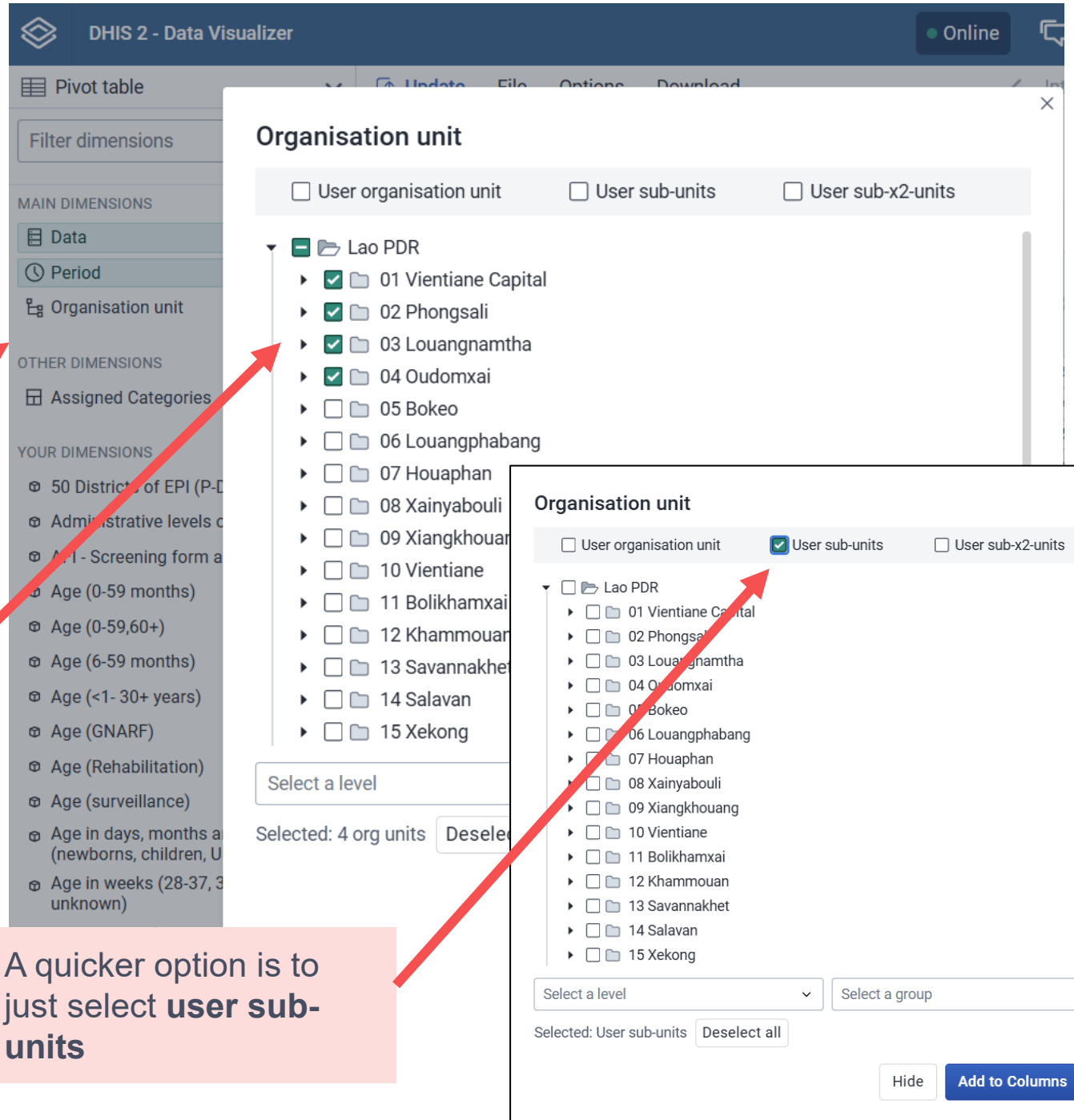
Selecting organization unit

At the moment we're looking at national totals, but we can also look at the data at smaller spatial units – down to the health facility level

Select **organization unit**

We can either select all the provinces manually by selecting the boxes – this method is best when we just want to look at a subset of provinces, districts etc.

A quicker option is to just select **user sub-units**



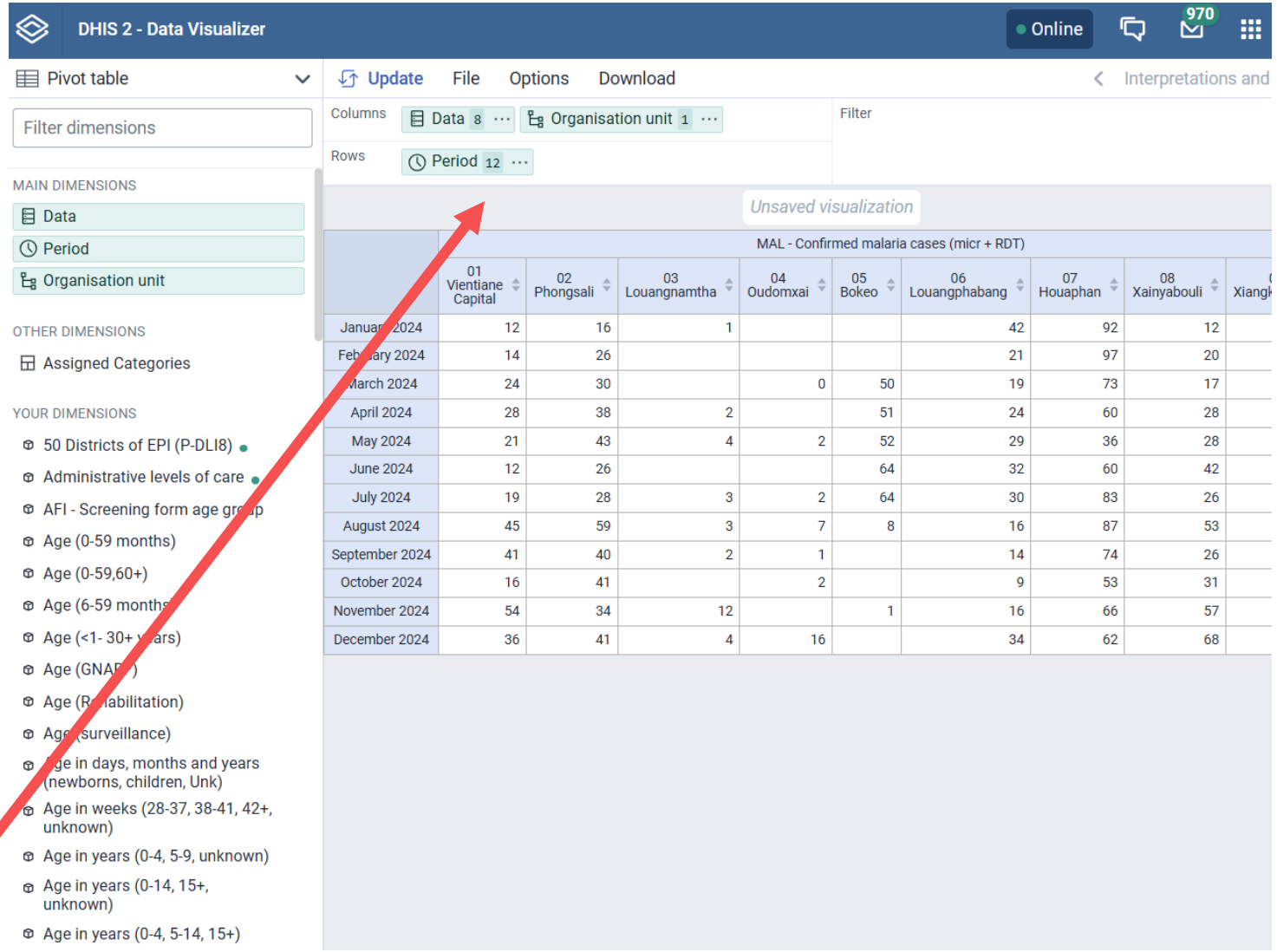
Reorienting and downloading data (1)

The data are returned in a wide format, with 18 columns (one per province) for each indicator

For future data manipulation in R, it's preferable to have the data in a **long format**

This is a data format where each row represents a single observation, and each column represents a variable or feature

To make data **long format** we can drag these green boxes all to the **rows' area** (see next slide)



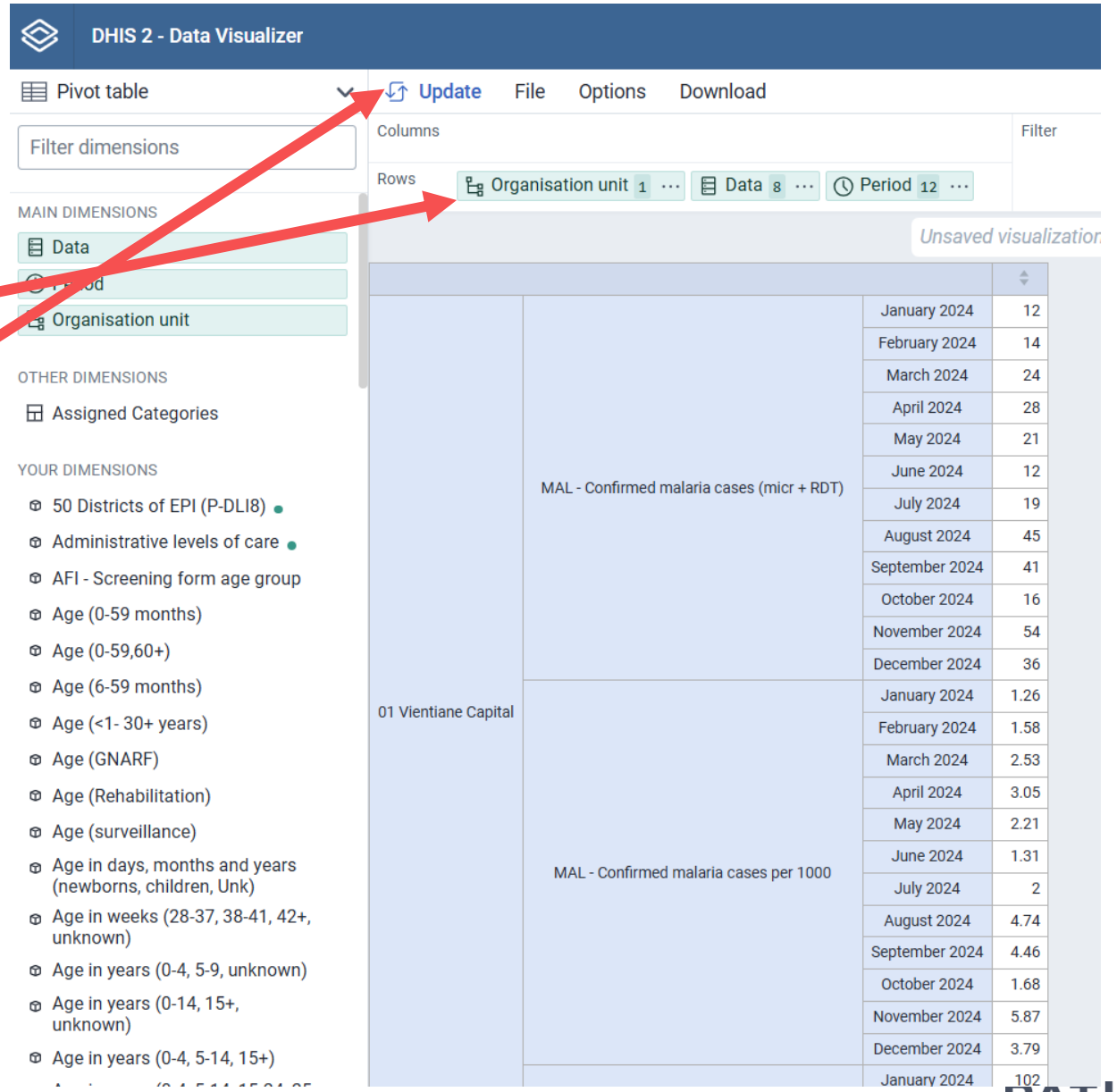
Useful resources for learning about **long** vs. **wide** data
https://tavareshugo.github.io/r-intro-tidyverse-gapminder/09-reshaping/index.html

PATH

Reorienting and downloading data (2)

You can change to order of these to see what is most logical for your data

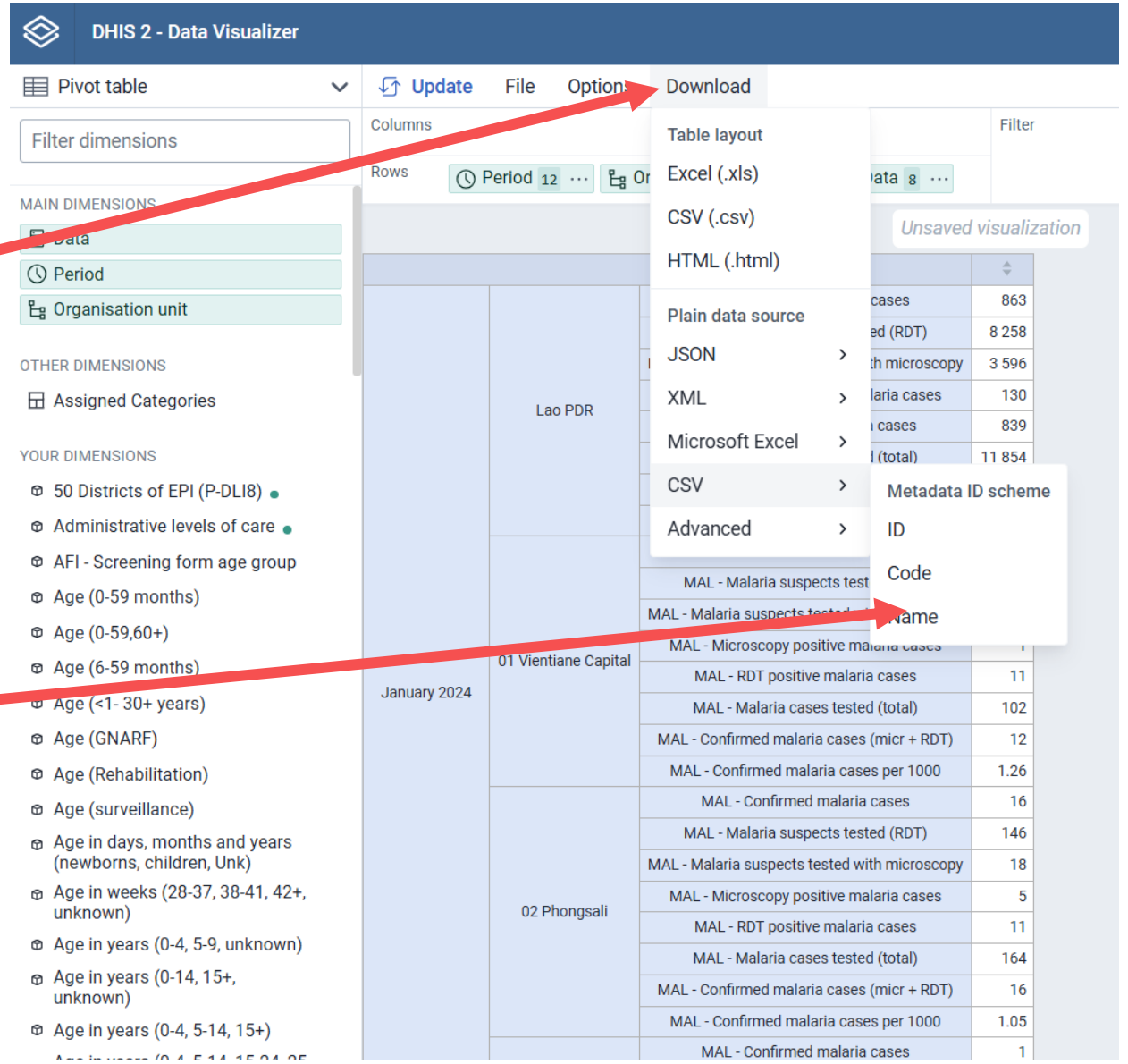
You'll need to **update** the table each time you modify it



Reorienting and downloading data (3)

We can now **download** our data as a csv

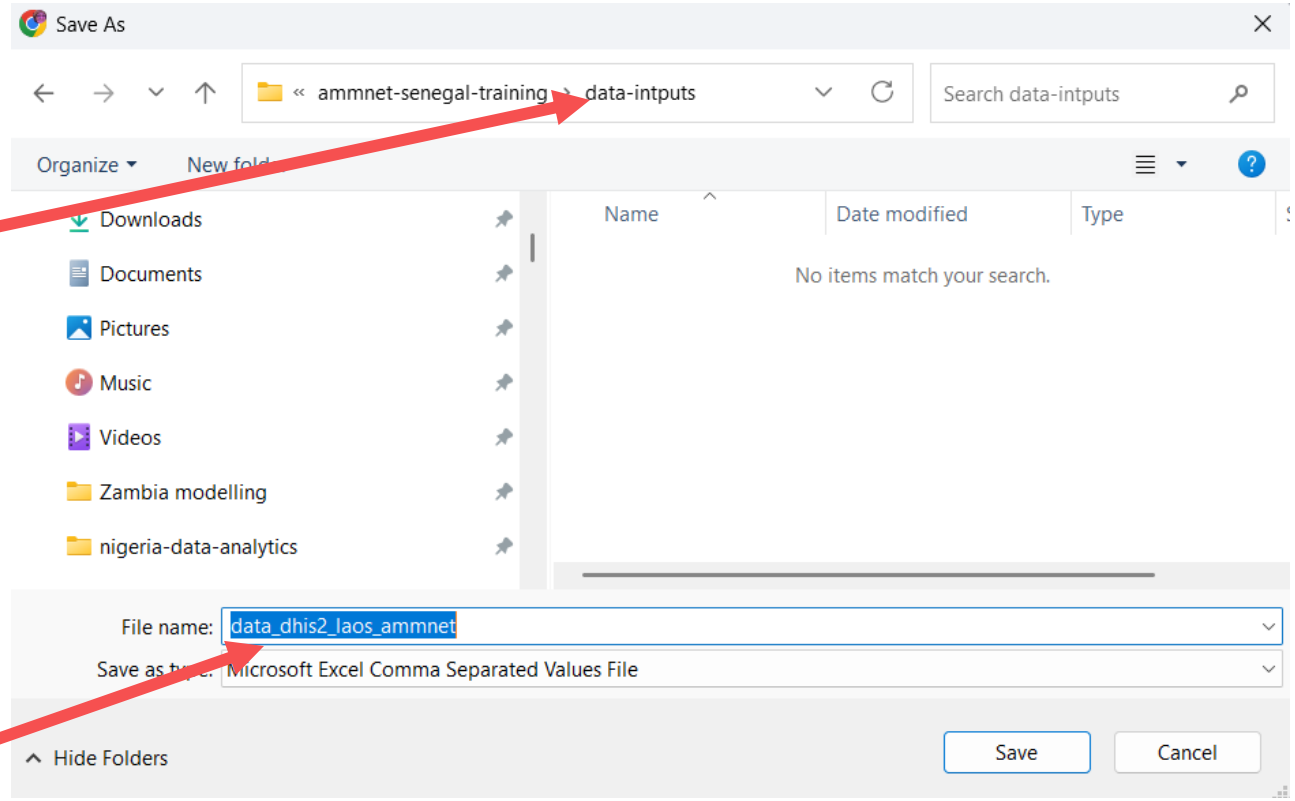
The easiest data format to work with is **plain data source -> CSV -> Name**



Reorienting and downloading data (4)

Save your data in your **data-inputs** folder that you created at the start of the session

Rename your file to something informative – I've chosen **data_dhis2_laos_ammnet**



Reading the data in R and cleaning

For this section we just need 2 packages tidyverse and janitor, if you don't already have them, you can install with the commands

```
install.packages("tidyverse")

install.packages("janitor")
```

Step 1: read in the data, clean the names and look at the data

```
library(tidyverse)

library(janitor)

laos_dat <- read_csv("data-inputs/data_dhis2_laos_ammnet.csv") %>%

      clean_names()

head(laos_dat)
```

Cleaning (1)

These data names look long and cumbersome – let's shorten them once we decided which ones to use

The download returned the national level data, which we don't want – let's filter this out

These columns don't look useful, so lets remove them from our output

```
# A tibble: 6 × 9
  data                  period organisation_unit value numerator denominator factor multiplier divisor
  <chr>                 <chr>  <chr>             <dbl>     <dbl>       <dbl>  <dbl>      <dbl>   <dbl>
1 MAL - Malaria cas… Janua… Lao PDR           11854     11854           1      1          1       1
2 MAL - Malaria cas… Janua… 01 Vientiane Cap…   102       102           1      1          1
3 MAL - Malaria cas… Janua… 02 Phongsali        164       164           1      1          1
4 MAL - Malaria cas… Janua… 03 Louangnamtha     123       123           1      1          1
5 MAL - Malaria cas… Janua… 04 Oudomxai         169       169           1      1          1
6 MAL - Malaria cas… Janua… 05 Bokeo             80        80           1      1          1
```

The date is in character format, but we need it in date format for R to know how it should be ordered

```
laos_dat_clean <- laos_dat %>%
    filter(organisation_unit != "Lao PDR") %>% # filter out national level data
    dplyr::select(-c(numerator, denominator, factor, multiplier, divisor)) %>% # remove unused columns
    mutate(data = str_replace_all(data, "MAL - ", ""), # remove starting MAL key
           data = str_to_lower(data),      # lower case
           data = str_replace_all(data, " ", "_"),      # spaces to underscores
           data = str_remove_all(data, "[()]"), # remove parenthesis
            data = str_remove_all(data, "[+]")) %>% # remove plus sign in one variable name
    mutate(month = ymd(parse_date_time(period, orders = "B Y"))) # make month a date variable
```

PATH

Cleaning (2)

It’s a good idea to look at the unique values in each text column to see if they all make sense

```
unique(laos_dat$data)
[1] "MAL - Malaria cases tested (total)" "MAL - Confirmed malaria cases (micr + RDT)"
[3] "MAL - Confirmed malaria cases per 1000" "MAL - Confirmed malaria cases"
[5] "MAL - Microscopy positive malaria cases" "MAL - Malaria suspects tested (RDT)"
[7] "MAL - Malaria suspects tested with microscopy" "MAL - RDT positive malaria cases"
```

We downloaded data that had similar names because we wanted to check for consistency – lets check these now – I’d expect

“MAL - Malaria cases tested (total)” = “MAL - Malaria suspects tested (RDT)” + “MAL - Malaria suspects tested with microscopy”

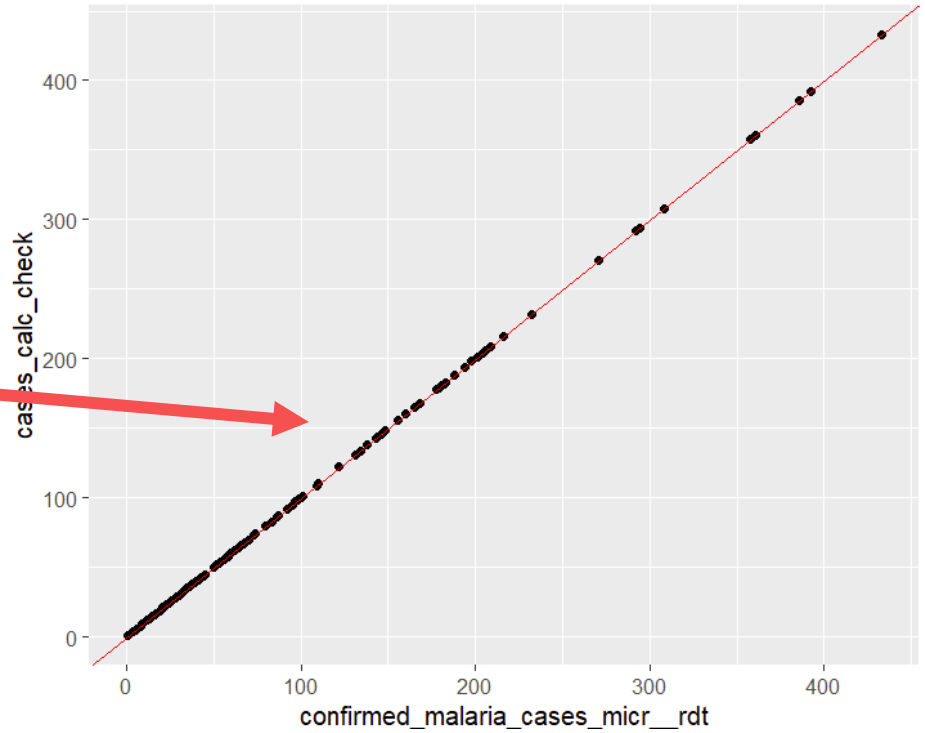
And

"MAL - Confirmed malaria cases (micr + RDT)" = “MAL - Microscopy positive malaria cases” + "MAL - RDT positive malaria cases" =

Cleaning (3)

```
laos_dat_wide <- laos_dat_clean %>%
    pivot_wider(names_from = data) %>%
    mutate(test_calc_check = rowSums(across(c(malaria_suspects_tested_rdt,
        malaria_suspects_tested_with_microscopy)), na.rm = TRUE),
    cases_calc_check = rowSums(across(c(rdt_positive_malaria_cases,
        microscopy_positive_malaria_cases))))

ggplot(laos_dat_wide) +
    geom_point(aes(x = malaria_cases_tested_total,
                   y = test_calc_check)) +
    geom_abline(intercept = 0, slope = 1, color = "red")
```

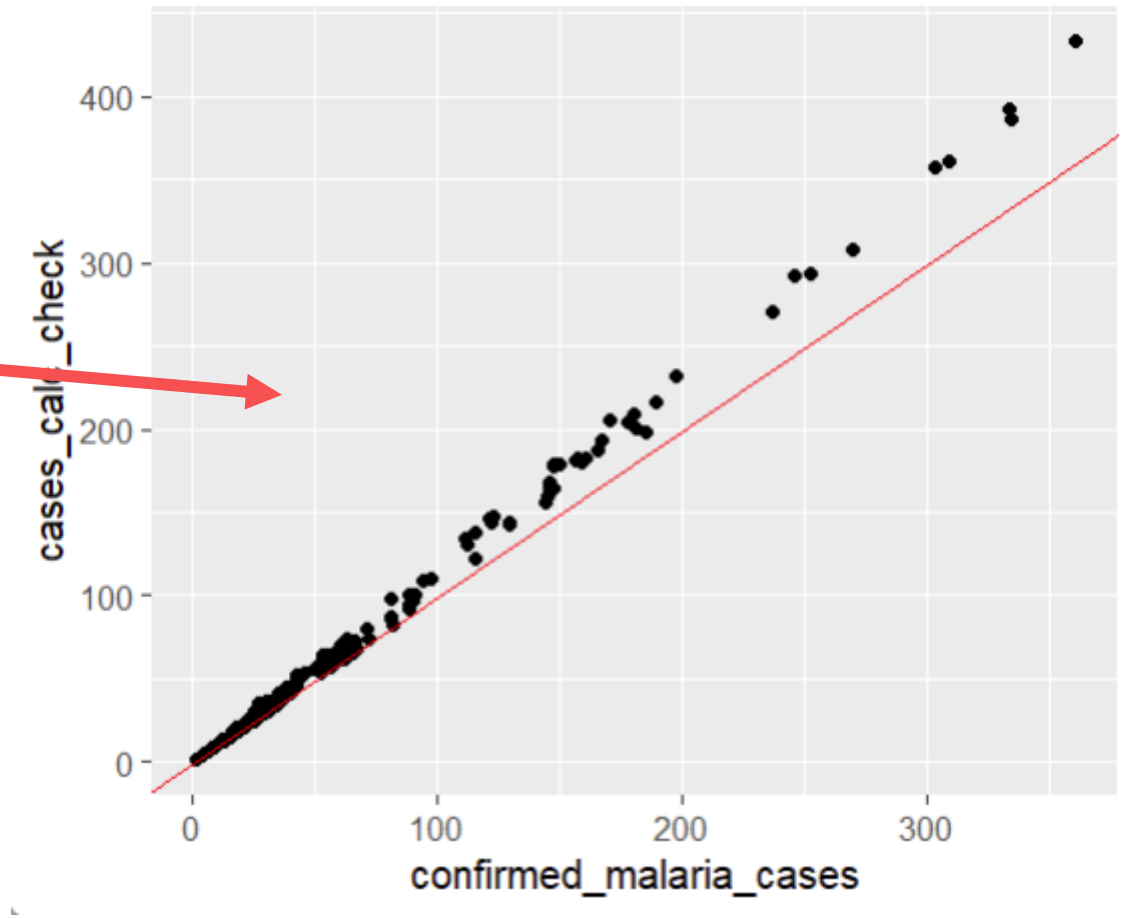
Here we see that these 2 variables are consistent, so we can continue just looking at one



Cleaning (4)

However the same is not true when we compare **cases_calc_check** with **confirmed_malaria** cases (we'd expect the points to fall on the red (x=y) line

Q: Can you write the code to generate this plot?



This means we, as the data analysts, need to **make a decision** about which variable to use.
Option 1: ask a local **expert**
Option 2: make an **educated decision** and **document** that decision so you can explain your results transparently in the future

For this analysis, we will use **confirmed malaria cases** (as we have spoken to a Laos surveillance officer who reported that in some setting individuals are tested with both RDT and microscopy which can lead to double counting of cases)

Cleaning (5)

Filter only the variable we want for our dashboard

Rename them so they're shorter and easier to work with

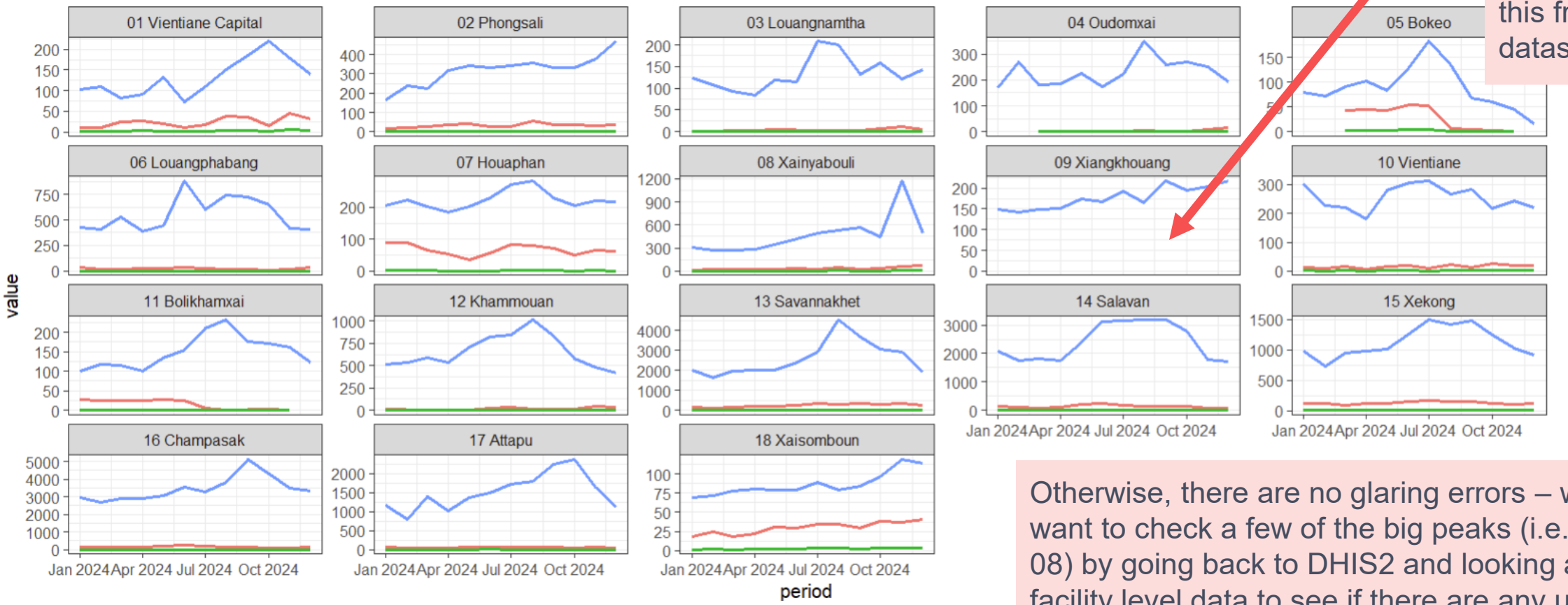
```
laos_dat_clean2 <- laos_dat_clean %>%
    filter(data %in% c("malaria_cases_tested_total",
                       "confirmed_malaria_cases",
                        "confirmed_malaria_cases_per_1000")) %>%

    mutate(var_name = case_when(data == "malaria_cases_tested_total" ~ "malaria_tests",
                                data == "confirmed_malaria_cases" ~ "malaria_cases",
                                data == "confirmed_malaria_cases_per_1000" ~ "malaria_cases_per_1000")) %>%

    dplyr::select(organisation_unit, period, var_name, value) %>%
    arrange(organisation_unit, period)
```

Select only the column we want to use and arrange in a more logical order

PATH

ALWAYS plot your data when cleaning – it's the easiest way to help spot anomalies/errors

```
ggplot(laos_dat_clean2) +
    geom_line(aes(x = period, y = value, color = var_name), linewidth = 0.9)+
    facet_wrap(vars(organisation_unit), scale = "free_y") +
    theme_bw()
```



Here we see that province 09 does not have any case or incidence data – lets just remove this from the dataset for now

Otherwise, there are no glaring errors – we might want to check a few of the big peaks (i.e. in province 08) by going back to DHIS2 and looking at the health facility level data to see if there are any unexpectedly large values

Filter out the province with missing variables

```
laos_dat_final <- laos_dat_clean2 %>%   filter(organisation_unit != "09 Xiangkhouang")
```

```
write_csv(laos_dat_final, "data-outputs/data_dhis2_laos_ammnet_cleaned.csv")
```

Save your output in the **data-outputs** folder, making note that this is now a cleaned dataset

If you haven't made it through all the data downloading and cleaning and please download **this dataset** so you will be ready for part 2: building a shiny dashboard

PATH