

# One Health data and website Sykdomspulsen infrastructure

MATRIX WP6 meeting

25.03.2021

# Agenda

- MATRIX
  - Datasets
  - Website
  
- Sykdomspulsen infrastructure
  - R packages
  - Databases
  - Analysis tasks
  - Website

# Sykdomspulsen team

## Sykdomspulsen core:



Gry



Richard



Beatriz



Chi

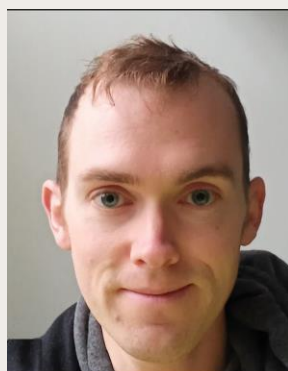


Calvin

## Sykdomspulsen H2020:



Clemence



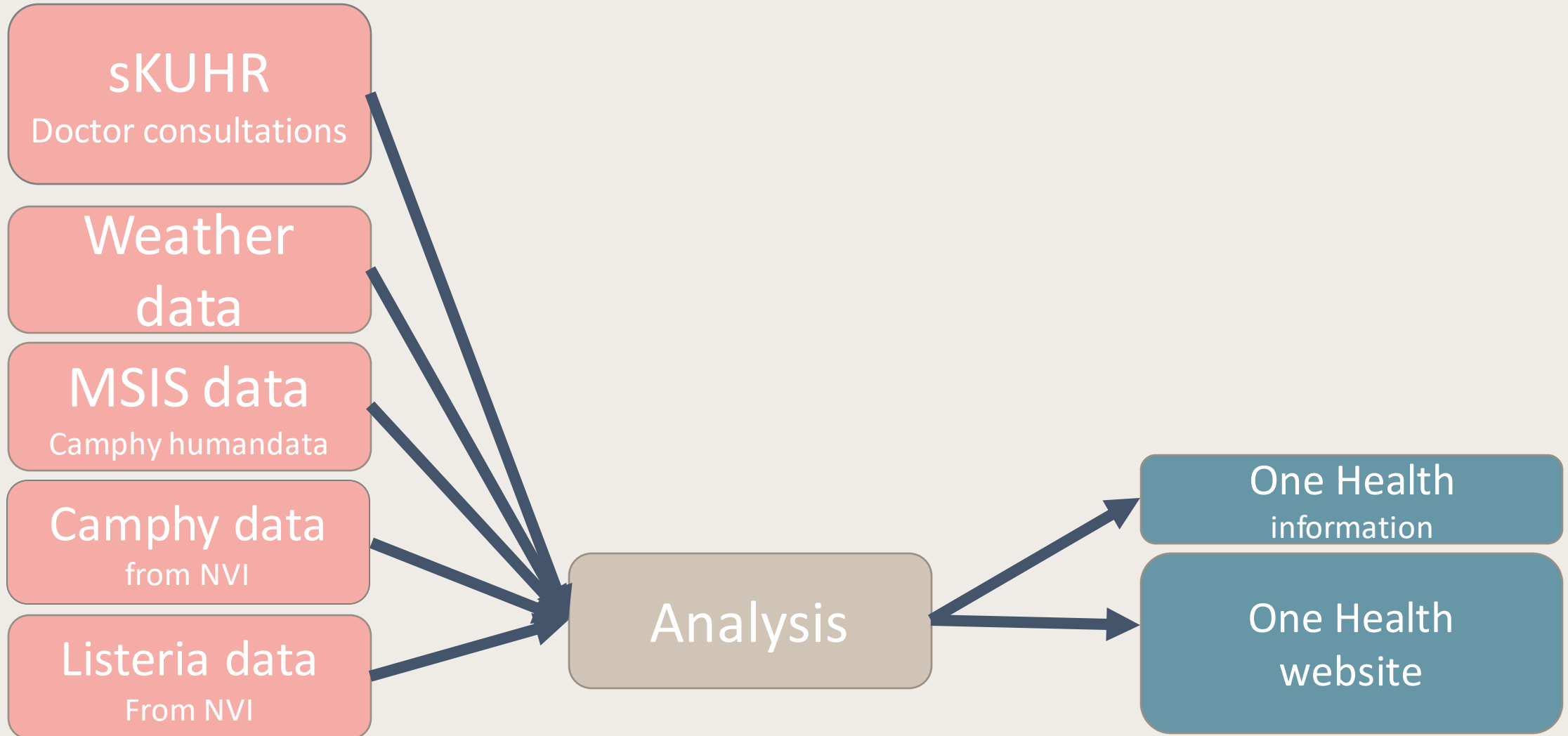
David

## Dødelighet:

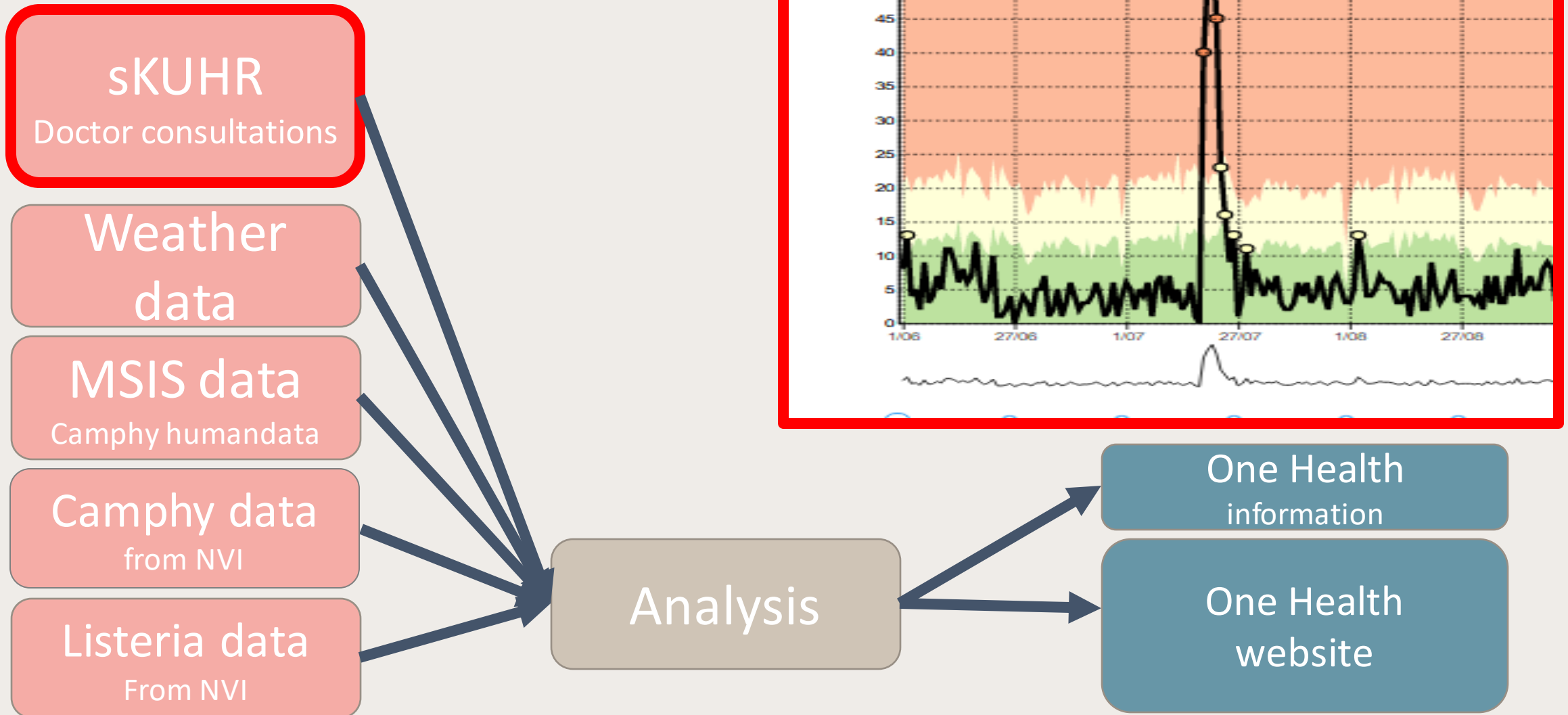


Aurora

# H2020 NOVA and MATRIX



# H2020 NOVA and MAT



# H2020 NOVA and MATRIX

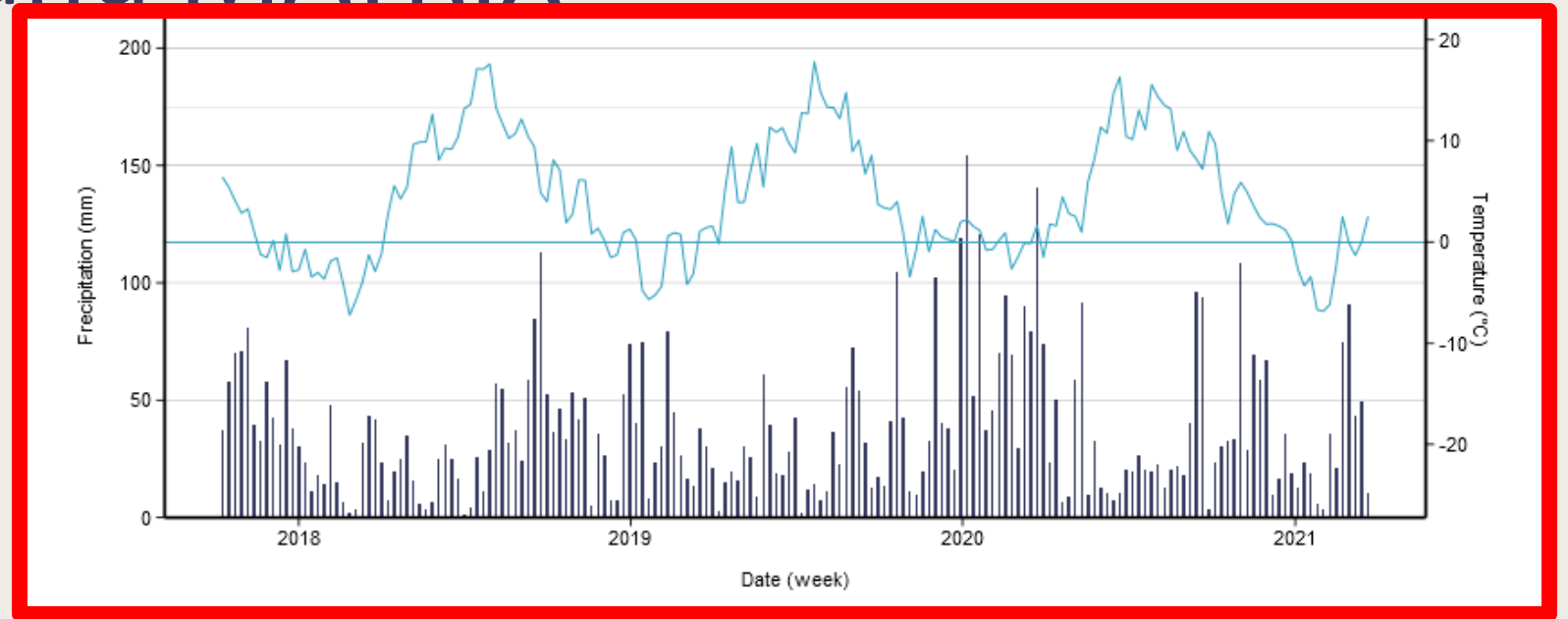
sKUHR  
Doctor consultations

Weather  
data

MSIS data  
Camphy humandata

Camphy data  
from NVI

Listeria data  
From NVI



Analysis

One Health  
information

One Health  
website

# H2020 NOVA and MATRIX

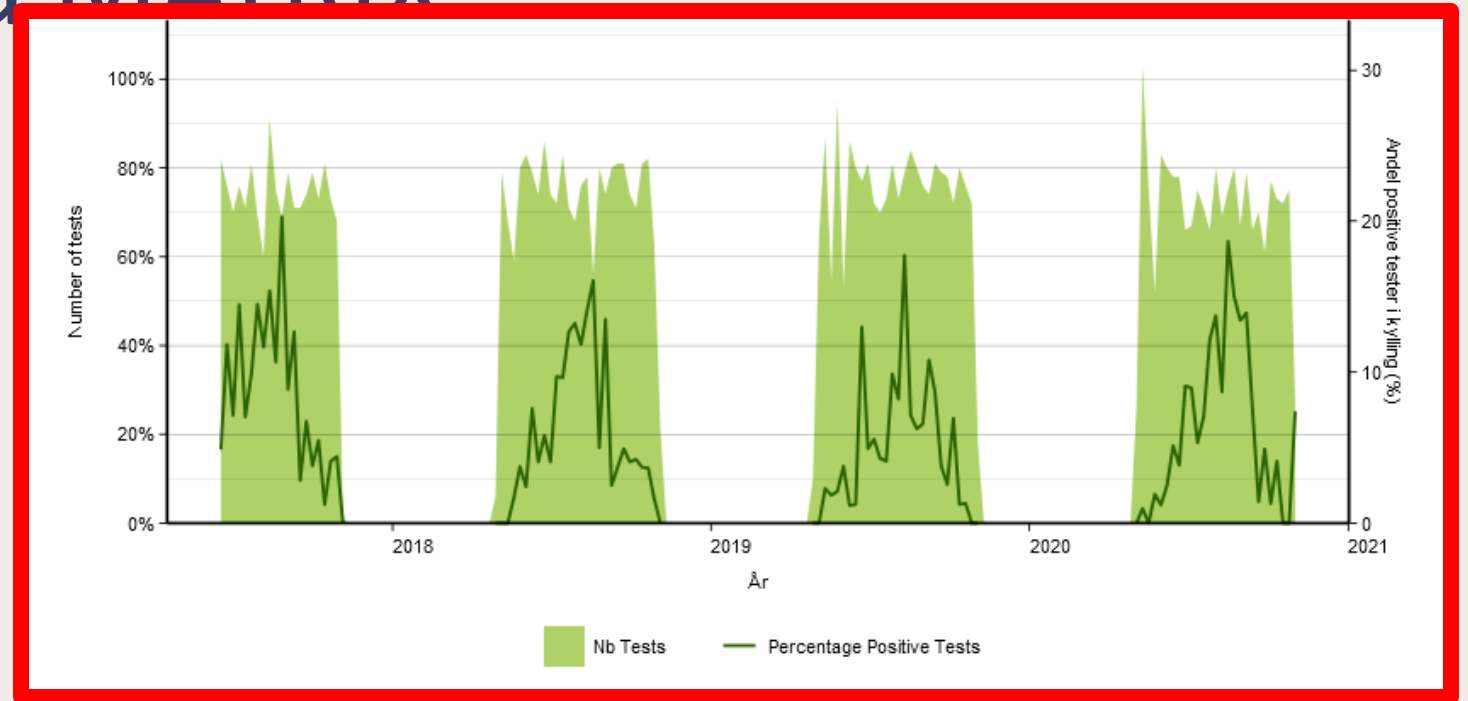
sKUHR  
Doctor consultations

Weather  
data

MSIS data  
Camphy humandata

Camphy data  
from NVI

Listeria data  
From NVI

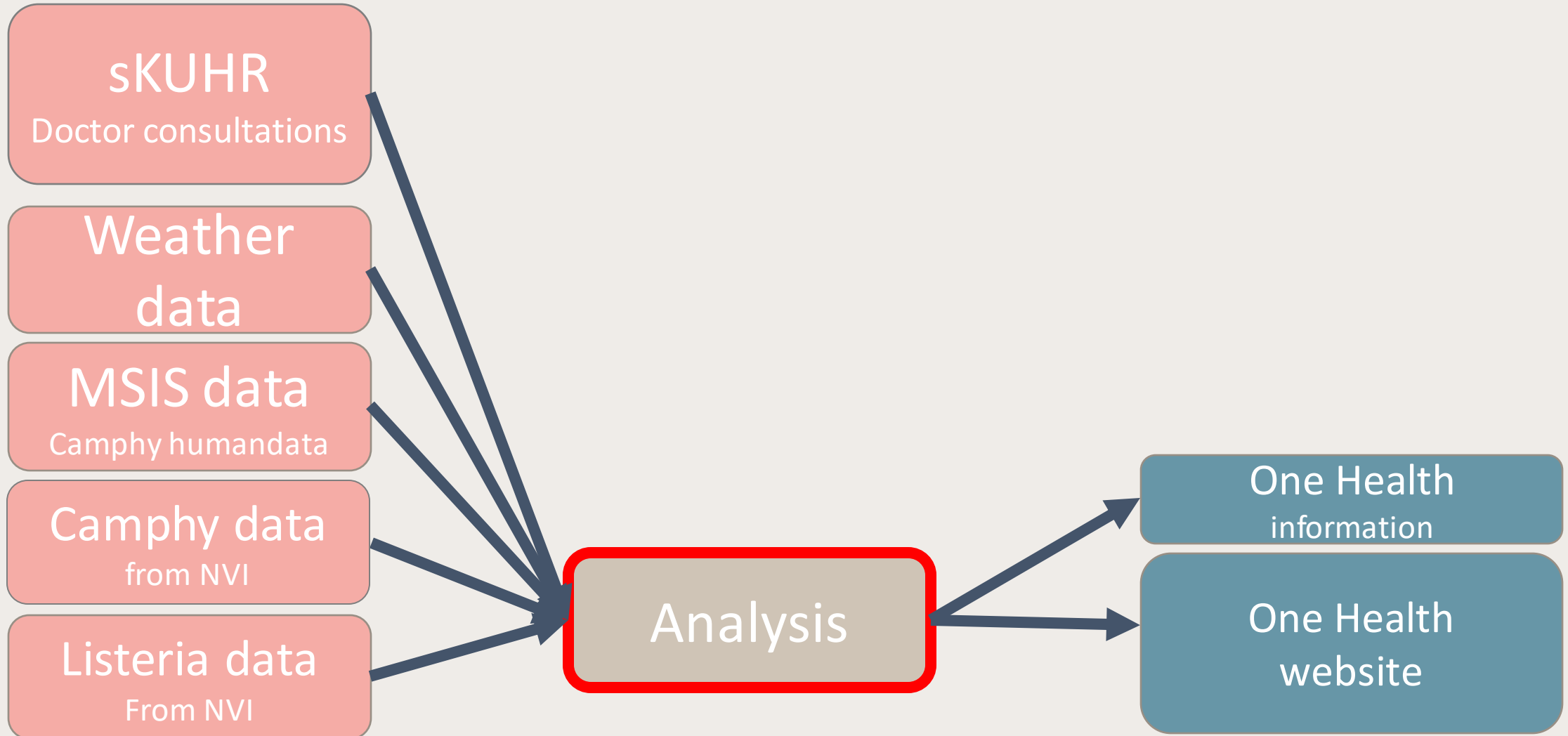


Analysis

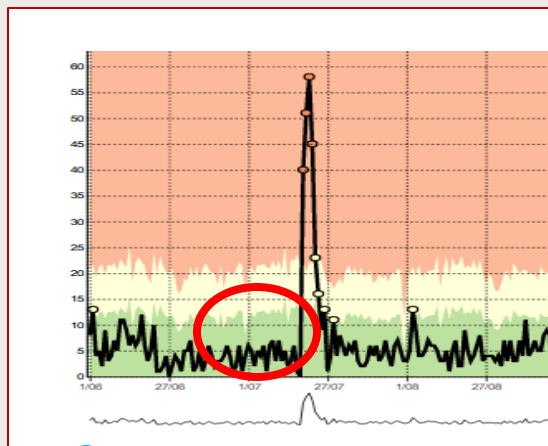
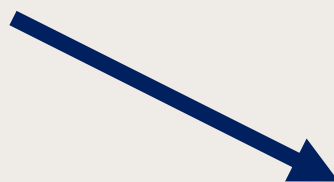
One Health  
information

One Health  
website

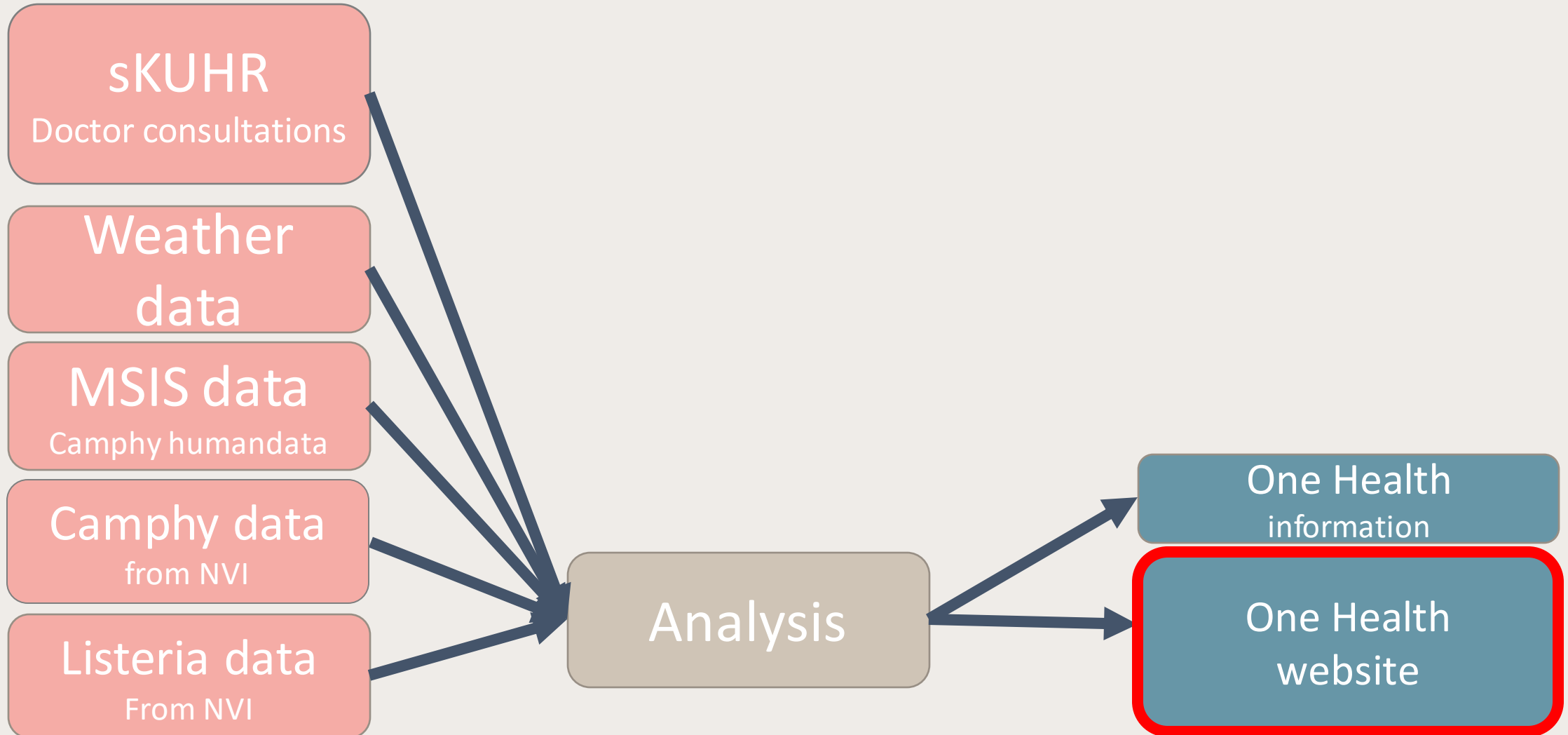
# H2020 NOVA and MATRIX





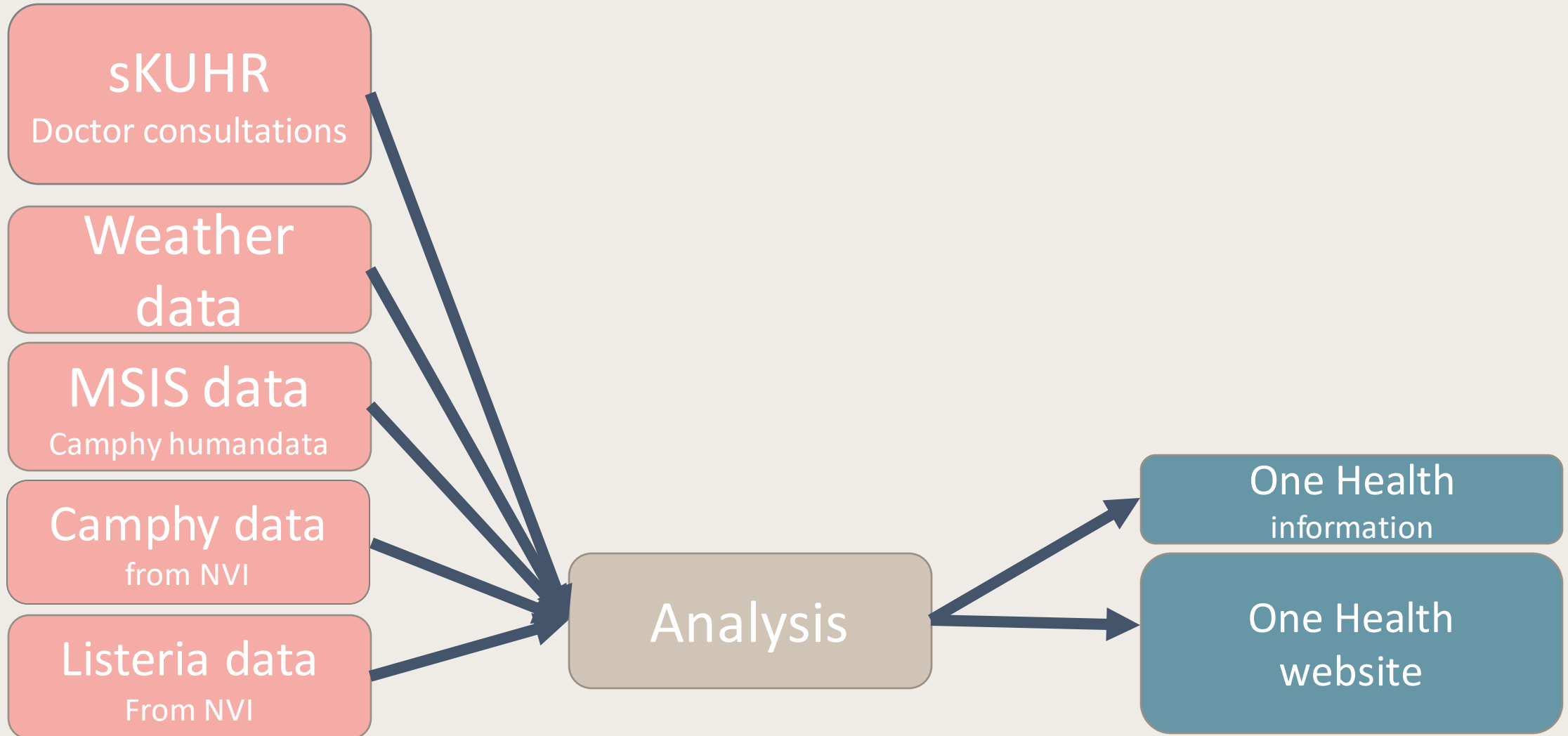


# H2020 NOVA and MATRIX



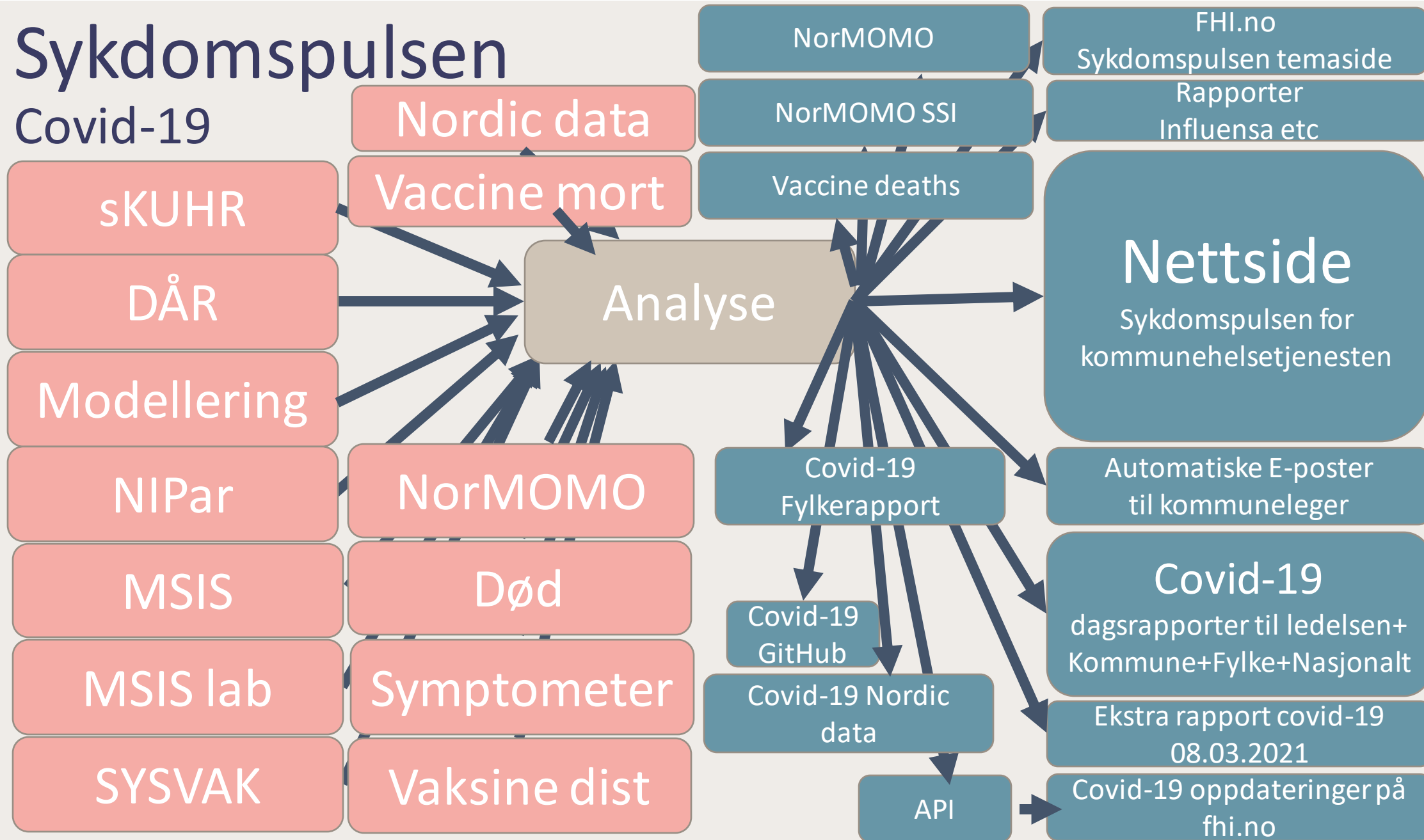
# Sykdomspulsen infrastructure

# H2020 NOVA and MATRIX

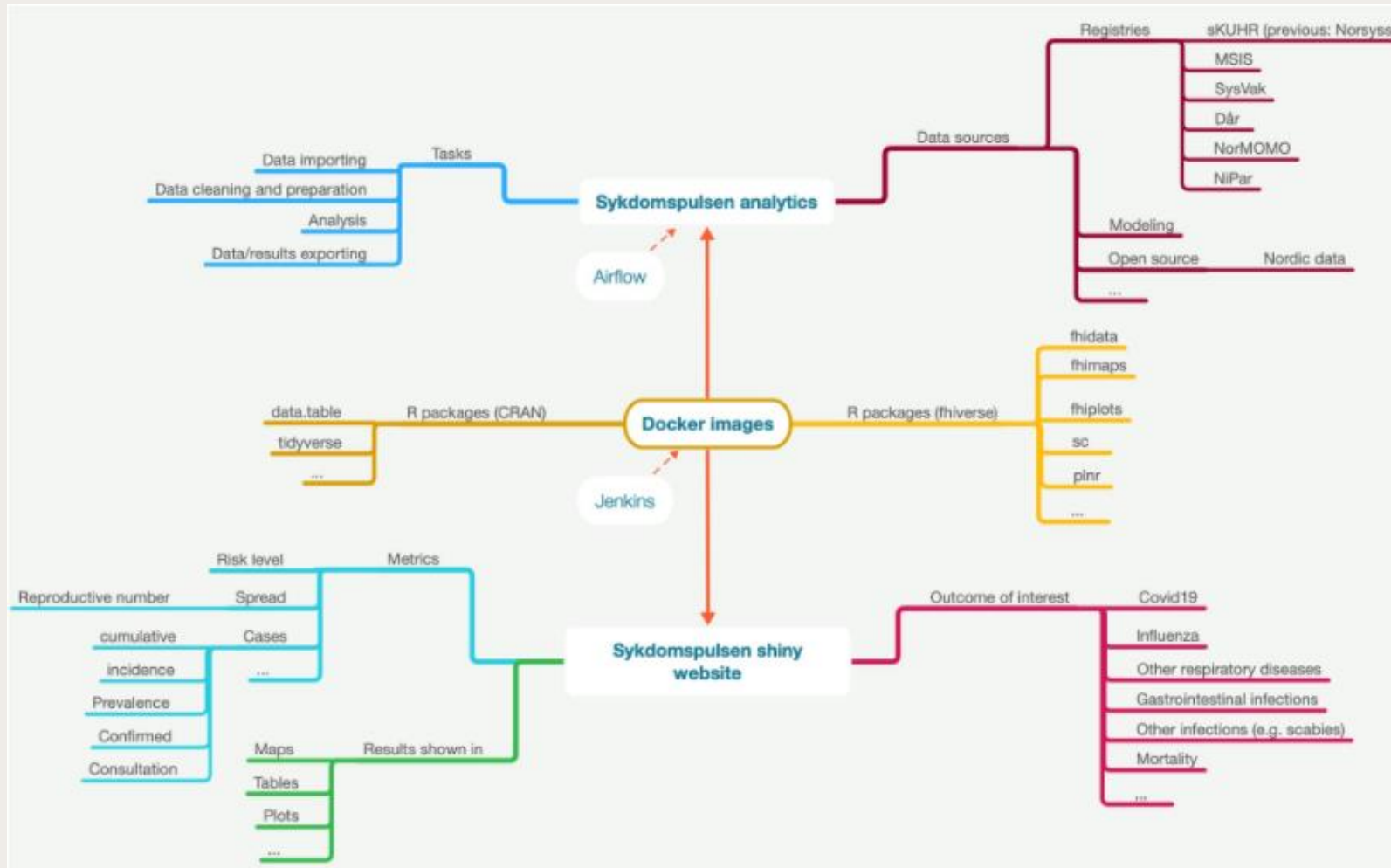


# Sykdomspulsen

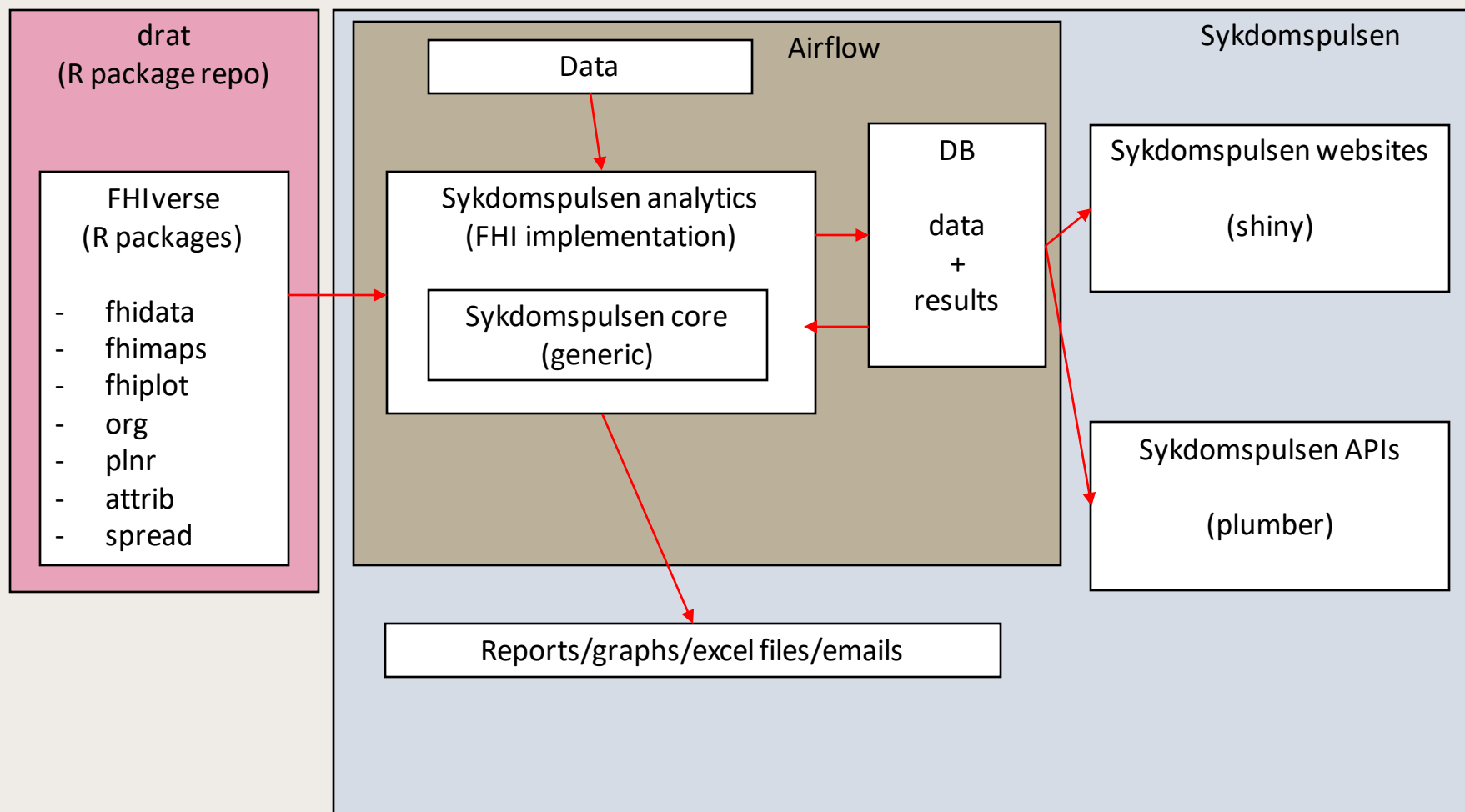
Covid-19



# Sykdomspulsen



# Sykdomspulsen



# fhiverse

A set of R packages designed to solve analytical problems that FHI sees every day while doing real-time surveillance of infectious diseases



# fhiverse

## fhidata (structural data)

```
> fhidata::norway_locations_names()
  location_code location_name location_name_description_nb
1:         norge         Norge
2:   county03         Oslo
3:   county11   Rogaland
4:   county15 Møre og Romsdal
5:   county18   Nordland
---
581: faregion1         Øst
582: faregion2   Stor-Oslo
583: faregion3   Sør og vest
584: faregion4         Midt
585: faregion5         Nord
> |
```

```
> fhidata::norway_population_by_age_cats(cats = list(c(0:10),c(11:20)))
  calyear location_code granularity_geo age sex pop imputed
1:   1846         norge         nation 00-10 total 338914 FALSE
2:   1846         norge         nation 11-20 total 266726 FALSE
3:   1847         norge         nation 00-10 total 340678 FALSE
4:   1847         norge         nation 11-20 total 270134 FALSE
5:   1848         norge         nation 00-10 total 343429 FALSE
---
21778: 2023 wardtrondheim500102 wardtrondheim 11-20 total 6021 TRUE
21779: 2023 wardtrondheim500103 wardtrondheim 00-10 total 6723 TRUE
21780: 2023 wardtrondheim500103 wardtrondheim 11-20 total 6332 TRUE
21781: 2023 wardtrondheim500104 wardtrondheim 00-10 total 5794 TRUE
21782: 2023 wardtrondheim500104 wardtrondheim 11-20 total 5571 TRUE
>
```

- Redistricting of municipalities/counties (happens extremely frequently in our current political climate)
- Names of locations
  - “Oslo”, “Oslo (city)”, “Oslo city”, “Oslo municipality”, “Oslo (m)”, “Oslo county”, “Oslo County”????????
  - Consistent ordering of locations in tables!
- granularity\_geo
  - nation, county, notmainlandcounty, missingcounty, municip, notmainlandmunicip, missingmunicip, wardoslo, extrawardoslo, missingwardoslo, wardbergen, missingwardbergen, wardstavanger, missingwardstavanger, wardtrondheim, missingwardtrondheim, baregion, region, faregion
  - We create ‘data skeletons’ for all granularity\_geo’s, so that our statisticians never need to think about “How do I deal with merging datasets that are missing rows?”
- Location hierarchy
  - Nation <-> Trondelag (county) <-> Trondheim (municipality) <-> Østbyen (ward)
- Population files
  - Also convenience functions that allow the user to aggregate to any age categories they want

# fhiverse

- **fhimaps** (maps that don't require geolibraries and are consistent with fhidata naming)
  - Kommunesammenslåing 2017, 2019, 2020
  - Fylkekart, kommunekart, delt fylkekart, delt kommunekart, fylkekart med Oslo innlegg, kommunekart med Oslo innlegg
  - Labelling points for all fylke/kommuner
  - Can create a map in 5 lines of code with zero geolibrary dependencies
- **fhiplot** (lets people easily create graphs according to FHI style guidelines)
- **spread** (infectious disease spread model that is already set up to run with Norwegian structural data thanks to fhidata)
- **org** (run the same code on different computers with different folder names, fundamental to the covid daily/weekly reports)
- **plnr** (enables the creation of reports with hundreds of different data sources and hundreds of different/independent outputs – fundamental to the covid daily/weekly reports)
- All researchers can use FHIverse packages, which increases the efficiency of the institute, and makes all of our analyses/outputs more consistent across different teams!

# Sykdomspulsen analytics/core

- **Infrastructure that allows for real time:**
  - Data extraction (e.g. weather data)
  - Data cleaning/harmonization (100s of data sources, 100 million+ rows of data)
  - Analysis (100 000+ analyses per day)
  - Graph/table/report/email creation
- **Focus on:**
  - **Only using statistical languages!** The user will never write C#, SQL, JS, or anything besides R!
  - Easy and sustainable harmonization
  - Speed/high performance
  - Reliability
  - Easy to debug/develop with
  - **Sustainability when increasing number of data sources/analyses/outputs**
- **Why do we need it?**
  - We didn't have a good way to handle the increasing number of data sources, analyses, and tasks
  - Our old code (Versions 1-6) meant that «more data/analyses/tasks» -> more complex code
  - **Sykdomspulsen's (V7) level of complexity stays the same, regardless of the number of tasks!**

# Sykdomspulsen analytics/core

## sKUHR

Receive daily updates for all of Norway

Aggregate -> clean -> harmonize -> upsert -> analyse

DB table long format = 200 000 000 rows, wide format = 20 000 000 rows

Analysis combination = (age group) \* (ICPC-2 code) \* (fylke/kommune) \* (years) = 100 000+ analyses = 24 minutes (runs in parallel) -> 4 500 000 rows of results

## Excess mortality monitoring

Receive weekly updates for all of Norway

Aggregate -> clean -> harmonize -> upsert -> analyse

DB table = 2 100 000 rows

Analysis combination = (age group) \* (county) \* (years) = 1000 analyses = 2 minutes (runs in parallel) -> 65 000 rows of results

## Attributable mortality

Analyses a combination of weather + KUHR + covid-19 + mortality

Orchestration/dependency of tasks is handled externally, using Airflow

# Sykdomspulsen analytics/core

## Database manager

Very explicit variable names  
Iso.... Seasonweek...

Redundant variables to make  
it easy for analysis

Consistent variable naming,  
which is extremely explicit

You identify unique rows of  
data. Sykdomspulsen core  
handles the rest.

```
# **** covid19 risk levels **** ----  
# covid19_risk_levels_countyreport ----  
sc::add_schema(  
  schema = sc::Schema$new(  
    db_table = "covid19_risk_levels_countyreport",  
    db_config = sc::config$db_config,  
    db_field_types = c(  
      "granularity_time" = "TEXT",  
      "granularity_geo" = "TEXT",  
      "location_code" = "TEXT",  
      "border" = "INTEGER",  
      "age" = "TEXT",  
      "sex" = "TEXT",  
      "isoyear" = "INTEGER",  
      "isoweek" = "INTEGER",  
      "isoyearweek" = "TEXT",  
      "season" = "TEXT",  
      "seasonweek" = "DOUBLE",  
      "date" = "DATE",  
  
      "msis_cases_n_sum0_6" = "INTEGER",  
      "msis_cases_pr100000_sum0_13" = "DOUBLE",  
      "lab_testevents_pr1000_sum0_6" = "DOUBLE",  
      "lab_testevents_pos_pr1000_sum0_6" = "DOUBLE",  
      "msis_areas_over_limit_numerator_n_sum0_13" = "INTEGER",  
      "msis_areas_over_limit_denom_n_sum0_13" = "INTEGER"  
    ),  
    db_load_folder = tempdir(),  
    keys = c(  
      "granularity_time",  
      "location_code",  
      "date",  
      "age",  
      "sex"  
    ),  
    validator_field_types = sc::validator_field_types_sykdomspulsen,  
    validator_field_contents = sc::validator_field_contents_sykdomspulsen,  
    info = "This db table is used for..."  
  )  
)
```

Mandatory  
common fields  
for structural  
variables

Only English is allowed,  
except for names of things  
(registries: msi, sysvak,  
daar)

# Sykdomspulsen analytics/core

## Database tables

- We create 'data skeletons' for all granularity\_geo's, so that our statisticians never need to think about "How do I deal with merging datasets that are missing rows?"
- Very easy to duplicate database tables. All db tables that will be used in a website have particular prefixes (e.g. "webkht\_\*", "weboh\_\*", "fhino\_api\_\*"). We copy from our data/analysis db tables into webkht\_\* db tables.
  - Very important, because vaccine data must be used in the morning report at 0630, but it's not allowed to be displayed on the website before 1300. Hence, we have two db tables: vaccine\_data (updated at 0630) and webkht\_vaccine\_data (a duplicate of vaccine\_data at 1300).
- All database table updates are automatically logged in a central db table that makes it very easy to display on our website "graph/table last updated..."

# Sykdomspulsen analytics/core

## Tasks

- How literally everything is done 😊 Data cleaning/importing, analyses, report creation, graphs, emails...
- Designed to work exactly like a single interactive independent script on your computer (line 1, line 2, line 3...)
- No loops, minimal amount of data, the “core of the problem”
  - “I have been given data for 0-4 year old males in Oslo in 2012, what analysis will I run on this data?”
- Does not need high R levels to work on editing the task

# Sykdomspulsen analytics/core

## Task manager

The only permitted loops are defined in the task manager, not in the task (allows for very easy parallelization)

Arguments

What will I do with my data?

How do I get my data?

What database tables can I access?

Immediate parallelization

```
# analysis_norsyss_respiratory ----
sc::add_task(
  sc::task_from_config_v3(
    name = "analysis_norsyss_respiratory",
    cores = 1,
    for_each_plan = plnr::expand_list(
      # x=1
      location_code = norway_locations_long()[granularity_geo %in% c("nation","county","municip")]$location_code
    ),
    for_each_argset = NULL,
    universal_argset = list(year_start = 2016),
    upsert_at_end_of_each_plan = FALSE,
    insert_at_end_of_each_plan = FALSE,
    action_fn_name = "sykdomspulsen::analysis_norsyss_respiratory_action",
    data_selector_fn_name = "sykdomspulsen::analysis_norsyss_respiratory_data_selector",
    schema = list(
      "data_norsyss_wide" = sc::config$schemas$data_norsyss_wide,
      "results_norsyss_respiratory" = sc::config$schemas$results_norsyss_respiratory
    ),
    info = "This task analyses NorSySS respiratory (shiny)"
  )
)
```



# Sykdomspulsen analytics/core

## Task data selector

Allows us to jump directly into the function, like it's a normal script

Extracting the data that is required for the analysis

Restrict on location\_code (comes from the plan!). This is basically “the loop”.

```
5 #' analysis_norsyss_respiratory (data selector)
6 #' @param argset Argset
7 #' @param schema DB Schema
8 #' @export
9 analysis_norsyss_respiratory_data_selector = function(argset, schema){
10   if(plnr::is_run_directly()){
11     # inside here is just for testing/development
12     argset <- sc::tm_get_argset("analysis_norsyss_respiratory", index_plan=1)
13     schema <- sc::tm_get_schema("analysis_norsyss_respiratory")
14   }
15 }
16
17
18 # The database schemas can be accessed here
19 # identical to sc::tbl("data_norsyss_wide")
20 d <- schema$data_norsyss_wide$dplyr_tbl() %>%
21   mandatory_db_filter(
22     granularity_time = "week",
23     granularity_geo = NULL,
24     age = "total",
25     sex = "total"
26   ) %>%
27   dplyr::filter(location_code == !! argset$location_code) %>%
28   dplyr::filter(year >= !!argset$year_start) %>%
29   dplyr::collect() %>%
30   as.data.table()
31
32
33 # The variable returned must be a named list
34 retval <- list(
35   "data_norsyss_wide" = d
36 )
37 }
```

# Sykdomspulsen analytics/core

## Task action

Can test entire task

Allows us to jump directly into the function, like it's a normal script

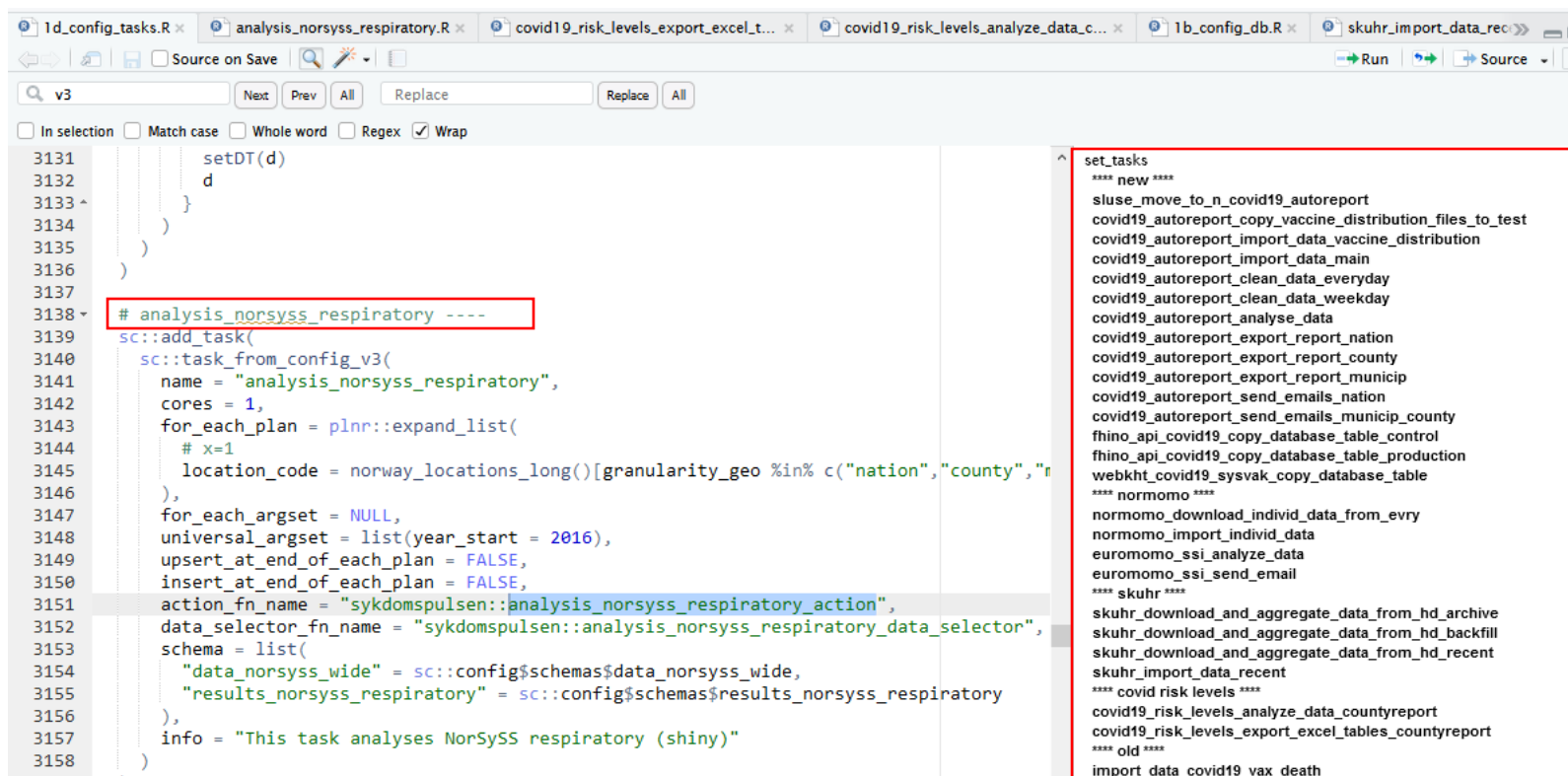
Code goes here. No loops!!

Upsert to database. So easy!

```
1 #' analysis_norsyss_respiratory (action)
2 #' @param data Data
3 #' @param argset Argset
4 #' @param schema DB Schema
5 #' @export
6 analysis_norsyss_respiratory_action <- function(data, argset, schema) {
7   # tm_run_task("analysis_norsyss_respiratory")
8
9   if(plnr::is_run_directly()){
10     index_plan <- 1
11     index_argset <- 1 # do not change
12
13     data <- sc::tm_get_data("analysis_norsyss_respiratory", index_plan = index_plan)
14     argset <- sc::tm_get_argset("analysis_norsyss_respiratory", index_plan = index_plan, index_argset = index_argset)
15     schema <- sc::tm_get_schema("analysis_norsyss_respiratory")
16   }
17
18   # data$data_norsyss_wide$location_code %>% unique
19   # sc::tm_get_plans_argsets_as_dt("analysis_norsyss_respiratory")
20   # code goes here
21
22   resp_codes <- stringr::str_subset(config$def$norsyss$diag_single_with_tariff$tag_output, "^r")
23   cols_n <- paste0('n_', resp_codes)
24   cols_denom <- paste0('denom_', resp_codes)
25
26   d <- data$data_norsyss_wide
27
28   # select the resp codes: count and denom
29   # compute percentage for selected columns
30   num <- dplyr::select(d, cols_n)
31   denom <- dplyr::select(d, cols_denom)
32
33   # if name match, then directly divide
34   num_diagcode <- stringr::str_sub(colnames(num), start = 3, end = 100000L)
35   denom_diagcode <- stringr::str_sub(colnames(denom), start = 7, end = 100000L)
36   stopifnot(all.equal(num_diagcode, denom_diagcode))
37
38   percentage <- num/denom *100
39   colnames(percentage) <- paste0('pr100_', resp_codes)
40
41   # put together:
42   # abandon codes such as influenza, keep only necessary cols
43   dd <- cbind(d[, 1:12], percentage)
44   setorder(dd, yrwk)
45
46   # schema$results_norsyss_respiratory$db_upsert_load_data_infile(retval)
69   schema$results_norsyss_respiratory$db_upsert_load_data_infile(retval)
70   # schema$results_norsyss_respiratory$db_add_constraint()
```

# Sykdomspulsen analytics/core

How do I navigate the tasks?

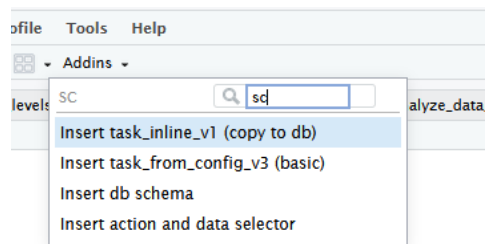


```
1d_config_tasks.R x analysis_norsyss_respiratory.R x covid19_risk_levels_export_excel_t... x covid19_risk_levels_analyze_data.c... x 1b_config_db.R x skuhr_import_data_rec...  
v3  
In selection Match case Whole word Regex Wrap  
3131     setDT(d)  
3132     d  
3133   }  
3134 )  
3135 )  
3136 )  
3137 )  
3138 # analysis_norsyss_respiratory ----  
3139 sc::add_task(  
3140   sc::task_from_config_v3(  
3141     name = "analysis_norsyss_respiratory",  
3142     cores = 1,  
3143     for_each_plan = plnr::expand_list(  
3144       # x=1  
3145       location_code = norway_locations_long()[granularity_geo %in% c("nation", "county", "m  
3146     ),  
3147     for_each_argset = NULL,  
3148     universal_argset = list(year_start = 2016),  
3149     upsert_at_end_of_each_plan = FALSE,  
3150     insert_at_end_of_each_plan = FALSE,  
3151     action_fn_name = "sykdomspulsen::analysis_norsyss_respiratory_action",  
3152     data_selector_fn_name = "sykdomspulsen::analysis_norsyss_respiratory_data_selector",  
3153     schema = list(  
3154       "data_norsyss_wide" = sc::config$schemas$data_norsyss_wide,  
3155       "results_norsyss_respiratory" = sc::config$schemas$results_norsyss_respiratory  
3156     ),  
3157     info = "This task analyses NorSySS respiratory (shiny)"  
3158   )  
)
```

```
set_tasks  
**** new ****  
sluse_move_to_n_covid19_autoreport  
covid19_autoreport_copy_vaccine_distribution_files_to_test  
covid19_autoreport_import_data_vaccine_distribution  
covid19_autoreport_import_data_main  
covid19_autoreport_clean_data_everyday  
covid19_autoreport_clean_data_weekday  
covid19_autoreport_analyse_data  
covid19_autoreport_export_report_nation  
covid19_autoreport_export_report_county  
covid19_autoreport_export_report_municip  
covid19_autoreport_send_emails_nation  
covid19_autoreport_send_emails_municip_county  
fhino_api_covid19_copy_database_table_control  
fhino_api_covid19_copy_database_table_production  
webkht_covid19_sysvak_copy_database_table  
**** normomo ****  
normomo_download_individ_data_from_evry  
normomo_import_individ_data  
euromomo_ssi_analyze_data  
euromomo_ssi_send_email  
**** skuhr ****  
skuhr_download_and_aggregate_data_from_hd_archive  
skuhr_download_and_aggregate_data_from_hd_backfill  
skuhr_download_and_aggregate_data_from_hd_recent  
skuhr_import_data_recent  
**** covid risk levels ****  
covid19_risk_levels_analyze_data_countyreport  
covid19_risk_levels_export_excel_tables_countyreport  
**** old ****  
import data covid19 vax death
```

# Sykdomspulsen analytics/core

How do I remember everything?

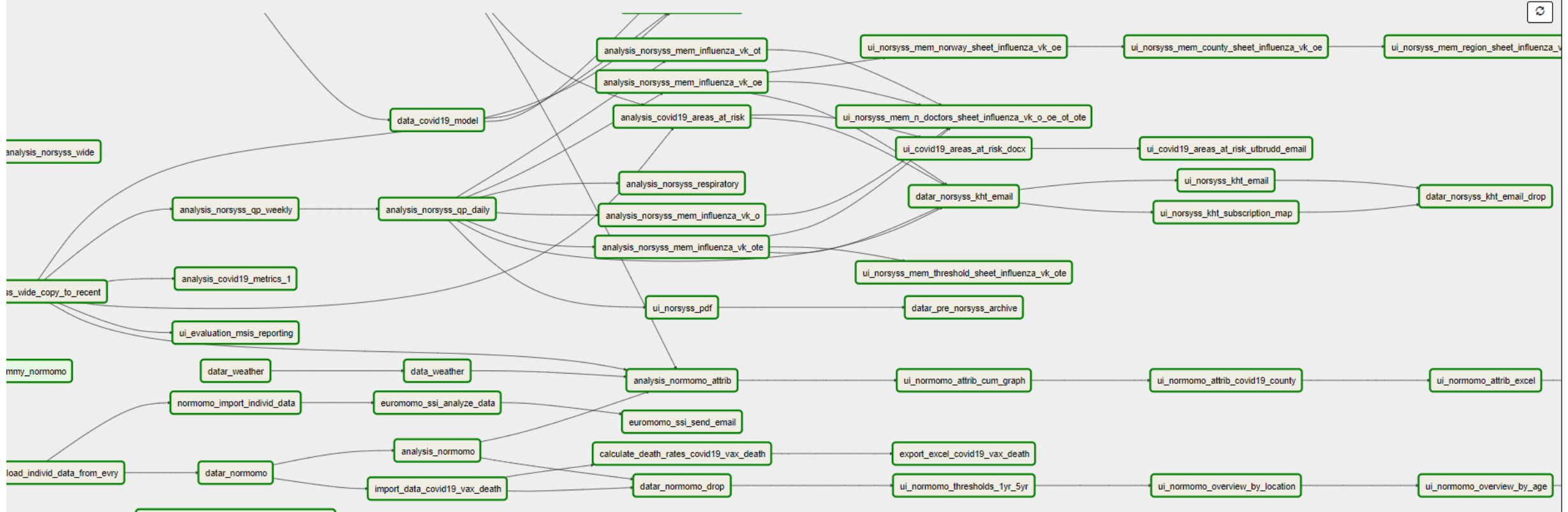


```
# TASK_NAME ----
# tm_run_task("TASK_NAME")
sc::add_task(
  sc::task_from_config_v3(
    name_grouping = "TASK_GROUPING",
    name_action = "TASK_ACTION",
    name_variant = "TASK_VARIANT",
    cores = 1,
    plan_argset_fn_name = NULL, # "PACKAGE::TASK_NAME_plan_argset"
    for_each_plan = plnr::expand_list(
      x = 1
    ),
    for_each_argset = NULL,
    universal_argset = NULL,
    upsert_at_end_of_each_plan = FALSE,
    insert_at_end_of_each_plan = FALSE,
    action_fn_name = "PACKAGE::TASK_NAME_action",
    data_selector_fn_name = "PACKAGE::TASK_NAME_data_selector",
    schema = list(
      "SCHEMA_NAME" = sc::config$schemas$SCHEMA_NAME
    ),
    info = "This task does..."
  )
)
```

```
# XGROUPX_XVARIANTX ----
sc::add_schema(
  schema = sc::Schema$new(
    db_table = "XGROUPX_XVARIANTX",
    db_config = sc::config$db_config,
    db_field_types = c(
      "granularity_time" = "TEXT",
      "granularity_geo" = "TEXT",
      "location_code" = "TEXT",
      "border" = "INTEGER",
      "age" = "TEXT",
      "sex" = "TEXT",
      "isoyear" = "INTEGER",
      "isoweek" = "INTEGER",
      "isoyearweek" = "TEXT",
      "season" = "TEXT",
      "seasonweek" = "DOUBLE",
      "date" = "DATE",
      "XXXX" = "DOUBLE"
    ),
    db_load_folder = tempdir(),
    keys = c(
      "granularity_time",
      "location_code",
      "date",
      "age",
      "sex"
    ),
    validator_field_types = sc::validator,
    validator_field_contents = sc::validator,
    info = "This db table is used for..."
  )
)
```

```
1
2 # **** action **** ----
3 #' TASK_NAME (action)
4 #' @param data Data
5 #' @param argset Argset
6 #' @param schema DB Schema
7 #' @export
8 TASK_NAME_action <- function(data, argset, schema) {
9   # tm_run_task("TASK_NAME")
10
11   if(plnr::is_run_directly()){
12     # sc::tm_get_plans_argsets_as_dt("TASK_NAME")
13
14     index_plan <- 1
15     index_argset <- 1
16
17     data <- sc::tm_get_data("TASK_NAME", index_plan = index_plan)
18     argset <- sc::tm_get_argset("TASK_NAME", index_plan = index_plan, index_argset = index_argset)
19     schema <- sc::tm_get_schema("TASK_NAME")
20   }
21
22   # code goes here
23 }
24
25 # **** data_selector **** ----
26 #' TASK_NAME (data selector)
27 #' @param argset Argset
28 #' @param schema DB Schema
29 #' @export
30 TASK_NAME_data_selector = function(argset, schema){
31   if(plnr::is_run_directly()){
32     # sc::tm_get_plans_argsets_as_dt("TASK_NAME")
33
34     index_plan <- 1
35
36     argset <- sc::tm_get_argset("TASK_NAME", index_plan = index_plan)
37     schema <- sc::tm_get_schema("TASK_NAME")
38   }
39
40   # The database schemas can be accessed here
41   d <- schema$SCHEMA_NAME$dplyr_tbl() %>%
42     dplyr::collect() %>%
43     as.data.table()
44
45   # The variable returned must be a named list
46   retval <- list(
47     "NAME" = d
48   )
49 }
77:1 **** functions ****
```

# Task orchestration (airflow)



# Sykdomspulsen analytics

## How do we implement it?

- We have two parallel systems
  - Sykdomspulsen analytics automatic
  - Sykdomspulsen analytics interactive
- Both have their own databases, airflow implementations, and folders
- Sykdomspulsen analytics interactive also has an active Rstudio Server installation
  - Extremely fast, low latency, close to the data
  - Only text is transmitted to the user
  - User just needs to use an internet browser
  - Can work for 8 hours remotely, using datasets that are 200 000 000 rows, and only use 50 MB of internet data

# Sykdomspulsen analytics

How do we implement it?

- We use docker containers to:
  - Freeze R packages
  - Ensure that all statisticians have exactly the same environment as the production environment
  - Ensure that we can update everything instantaneously, without having any issues or forgetting anything/anyone

# Sykdomspulsen websites

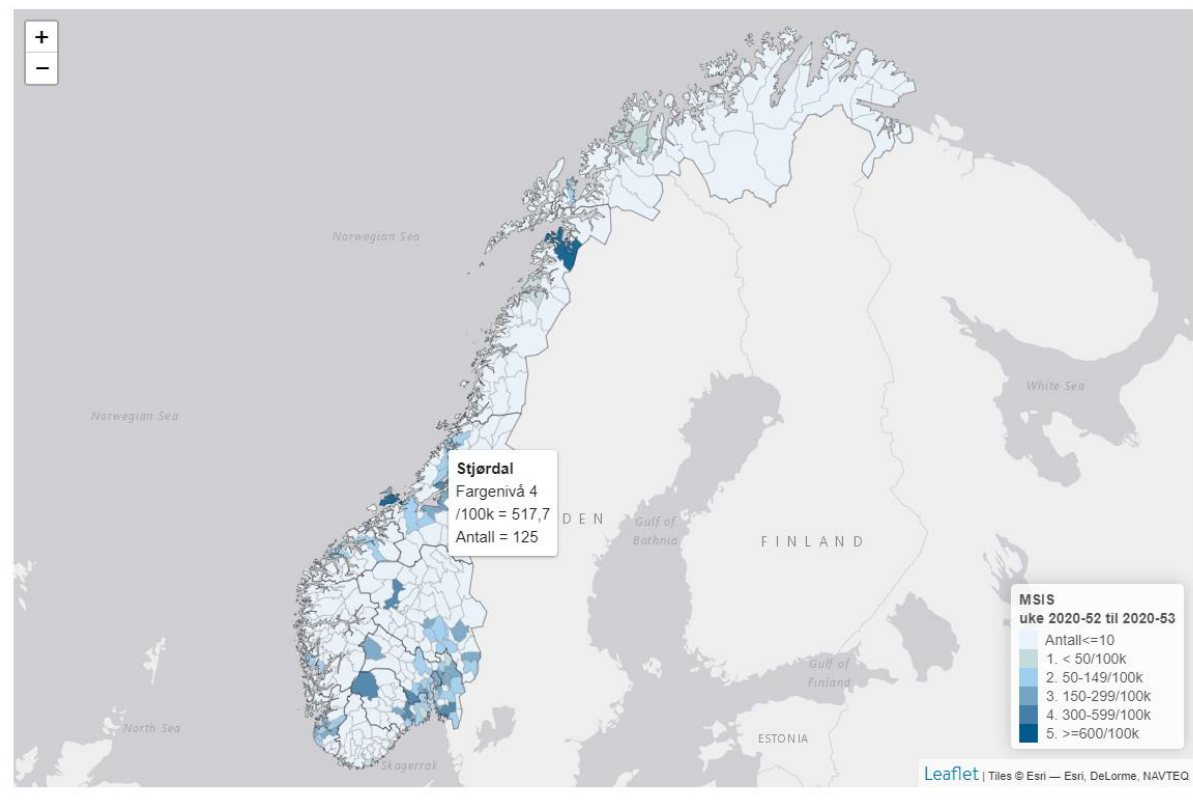
- Implemented using the «shiny» R package
- Production/test websites are implemented in Docker containers
- Interactive development:
  - Was done via Rstudio locally installed on a windows computer
  - Is now done via Rstudio Server
- Code is very similar to code used to produce the daily/weekly report, so there are efficiencies there as well
- Uses same «fhiverse» R-packages as Sykdomspulsen Analytics
- Fetches the data/results directly from the SQL databases (using R wrappers)
- Manipulates the data using the gold-standard R packages designed for these tasks
- Produces graphs/tables using the gold-standard R packages designed for these tasks



# Sykdomspulsen websites

## Efficient development

**Figur 1b.** Kommunekart med nye covid-19 tilfeller i løpet av de to siste fulle ukene pr. 100.000 innbyggere. Dataene kan endres hver dag da de ofte ikke er komplette for forrige uke i begynnelsen av en ny uke, men blir mer og mer komplett utover uka. Kommuner med under 10 tilfeller de siste 14 dagene er farget veldig lys blå/hvit.

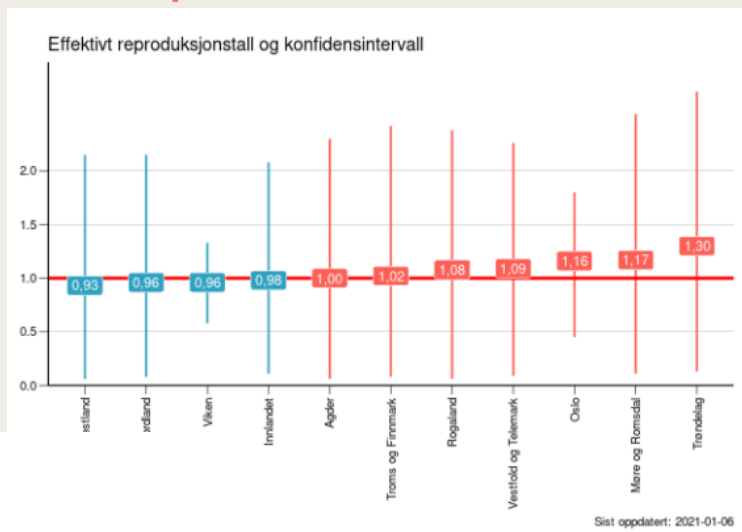


- Interactive map comes from «fhimaps»
- 40 lines of code

```
410 ui_a <- leaflet::leaflet(  
411   pd_county,  
412   options = leaflet::leafletOptions(preferCanvas = F)  
413 ) %>%  
414 leaflet::addProviderTiles(leaflet::providers$Esri.WorldGrayCanvas) %>%  
415 leaflet::addPolygons(  
416   fillColor = ~ pal_not_censored(n_0_13_fixed_status_with_numbers),  
417   weight = 0.3,  
418   opacity = 1,  
419   color = "white",  
420   fillOpacity = 0.9,  
421   highlight = leaflet::highlightOptions(  
422     weight = 5,  
423     color = "#00",  
424     fillOpacity = 0.7,  
425     bringToFront = F  
426   ),  
427   label = lab_a,  
428   labelOptions = leaflet::labelOptions(  
429     textSize = "15px"  
430   )  
431 ) %>%  
432 leaflet::addLegend(  
433   pal = pal_not_censored,  
434   values = labels_0_13_with_numbers,  
435   opacity = 1,  
436   title = glue("MSIS<br/>uke {d$description_0_13_fixed[1]}"),  
437   position = "bottomright"  
438 )  
439
```

# Sykdomspulsen websites

## Efficient development



```
591 # plot
592 q <- ggplot(d_county, aes(x = location_name_fct,
593   y = r_est,
594   color = rabove1,
595   label = fhiplot::format_nor(r_est, digits=2)))
596 q <- q + geom_pointrange(aes(ymin = r_thresholdl0,
597   ymax = r_thresholdu0),
598   size = 0.8)
599 q <- q + geom_hline(aes(yintercept = 1),
600   color = 'red', size = 1.2)
601 q <- q + geom_label(aes(fill = factor(rabove1)),
602   colour = 'white')
603 q <- q + scale_y_continuous(breaks = c(0, 0.5, 1, 1.5, 2),
604   expand = expansion(mult = c(0.0, 0.1)))
605 q <- q + expand_limits(y=0)
606 q <- q + fhiplot::scale_color_fhi(palette = 'posneg')
607 q <- q + fhiplot::scale_fill_fhi(palette = 'posneg')
608 q <- q + fhiplot::theme_fhi_lines_horizontal(panel_on_top = F)
609 q <- q + fhiplot::set_x_axis_vertical()
610 q <- q + theme(axis.title.x = element_blank(),
611   axis.title.y = element_blank(),
612   legend.position = 'none') +
613   labs(title = 'Effektivt reproduksjonstall og konfidensintervall',
614   caption = paste0('Sist oppdatert: ', date_model_was_run))
615
```

- 25 lines of code
- Can iterate and test a new version of the graph in under 10 seconds
- Incredibly fast development
- Incredibly flexible
- Perfect for the pandemic!!!
- Uses «fhiplot» so that the styles/colors/themes are the same on the website and in all reports

# Sykdomspulsen websites

## Key features for success

- Having a very strong analytics platform producing extremely standardized database tables
- Ability to duplicate database tables very easily at given time points
- Fhiverse R-packages and tutorials ensuring consistent output throughout the institute, and ensuring that all “structural data” is 100% identical in all analyses/infrastructure and easily updated
- Using statistical languages (R) everywhere
- Docker images ensuring 100% consistent environments for all users on all servers