Package 'sirfunctions'

June 24, 2025

Title Key Functions to Analyze Global Polio Surveillance Data

Version 2.0.0

Description The sirfunctions package contains key functions used by the Surveillance, Innovation, and Research (SIR) team within the Polio Eradication Branch (PEB) at the Centers for Disease Control and Prevention (CDC). It includes functions to download cleaned global polio data from the World Health Organization (WHO) Polio Information System (POLIS) and geographic files. In addition, the package contains functions to visualize important epidemiological trends and perform key performance indicators such as NPAFP rates, EV rates, and stool adequacy. Apart from getting data and calculating key indicators, there are several functions that also facilitate communication, visualize trends, and perform data quality checks.

License MIT + file LICENSE

URL https://github.com/nish-kishore/sirfunctions

Imports AzureAuth (>= 1.3.0),
AzureStor (>= 3.7.0),
cli (>= 3.6.3),
dplyr (>= 1.1.0),
flextable (>= 0.9.0),
ggplot2 (>= 3.5.0),
glue (>= 1.8.0),
lifecycle (>= 1.0.0),
lubridate (>= 1.9.0),
readr (>= 2.1.0),
rlang (>= 1.1.0),
scales (>= 1.3.0),
sf (>= 1.0-19),
stringr (>= 1.5.0),
utils (>= 4.4.0),
tidyr (>= 1.3.0),
httr,
jsonlite,
purrr,
withr,
yaml,
stats

Suggests arrow,

qs2,
blastula,
forcats,
ggh4x,
ggspatial,
git2r,
officer,
ggpubr,
ggrepel,
prettyunits,
janitor,
datasets,
readxl,
tibble,
tools,
knitr,
magick,
openxlsx,
vctrs,
rmarkdown,
Microsoft365R,
zoo,
writexl,
rvg,
testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Contents

add_rolling_years 5
check_afp_geographies 6
check_afp_guid_ctry_data 6
check_missing_rows 7
clean_ctry_data 8
clean_es_data 8
clean_iss_data 9
clean_lab_data 10
compress_png 11
create_60_day_export 12
create_afp_export 13
create_emergence_group_gif 13
create_npafp_export 15
create_pop_check_export 16
create_pot_comp_clust_export 16
create_stool_adequacy_export 17
ctry_data_errors 18

duplicate_check . . . 19
edav_io . . . 19
explore_edav . . . 21
export_kpi_table . . . 21
extract_country_data . . . 23
f.color.schemes . . . 23
f.ev.rate.01 . . . 24
f.expand.bbox . . . 25
f.metadata.tag . . . 26
f.npafp.rate.01 . . . 26
f.plot.looks . . . 28
f.stool.ad.01 . . . 28
f.timely.detection.01 . . . 30
fix_ctry_data_missing_guids . . . 31
freeze_dr_data . . . 32
generate_60_day_tab . . . 33
generate_60_day_table_data . . . 33
generate_adhoc_map . . . 34
generate_ad_final_col . . . 36
generate_afp_by_month_summary . . . 37
generate_afp_case_map . . . 38
generate_afp_epicurve . . . 39
generate_afp_prov_year . . . 39
generate_c1_rollup . . . 40
generate_c1_table . . . 41
generate_c2_table . . . 42
generate_c3_rollup . . . 43
generate_c3_table . . . 44
generate_c4_table . . . 45
generate_case_num_dose_g . . . 46
generate_ctry_timeliness_graph . . . 47
generate_dist_pop_map . . . 48
generate_dr_ppt . . . 49
generate_dr_ppt2 . . . 51
generate_es_det_map . . . 52
generate_es_site_det . . . 53
generate_es_tab . . . 54
generate_es_timely . . . 55
generate_inad_tab . . . 56
generate_int_data . . . 57
generate_iss_barplot . . . 58
generate_iss_map . . . 59
generate_kpi_barchart . . . 60
generate_kpi_evdetect_bar . . . 60
generate_kpi_ev_map . . . 61
generate_kpi_map . . . 62
generate_kpi_npafp_bar . . . 63
generate_kpi_npafp_map . . . 64
generate_kpi_stoolad_bar . . . 65
generate_kpi_stool_map . . . 66
generate_kpi_tile . . . 67
generate_kpi_violin . . . 68

generate_lab_culture_violin . . . 68
generate_lab_itdres_seqres_violin . . . 69
generate_lab_itd_violin . . . 70
generate_lab_seqres_violin . . . 71
generate_lab_seqship_violin . . . 72
generate_lab_timeliness . . . 73
generate_npafp_maps . . . 74
generate_npafp_maps_dist . . . 75
generate_pop_map . . . 77
generate_pop_tab . . . 78
generate_potentially_compatibles_cluster . . . 79
generate_prov_timeliness_graph . . . 80
generate_sg_priority_map . . . 81
generate_stool_ad_maps . . . 81
generate_stool_ad_maps_dist . . . 83
generate_stool_data . . . 84
generate_surv_ind_tab . . . 85
generate_timeliness_maps . . . 87
generate_timely_det_violin . . . 88
generate_timely_ship_violin . . . 89
get_all_polio_data . . . 90
get_azure_storage_connection . . . 91
get_cdc_childvaxview_data . . . 92
get_constant . . . 93
get_ctry_abbrev . . . 94
get_diff_cols . . . 94
get_lab_date_col_missingness . . . 95
get_lab_locs . . . 95
get_ppt_template . . . 96
get_region . . . 97
get_vpd_data . . . 97
get_vpd_missingness . . . 98
get_vpd_vars . . . 99
init_dr . . . 99
init_kpi . . . 101
iss_data_errors . . . 101
lab_data_errors . . . 102
load_clean_ctry_sp . . . 103
load_clean_dist_sp . . . 104
load_clean_prov_sp . . . 106
load_iss_data . . . 107
load_lab_data . . . 108
send_outlook_email . . . 108
send_teams_message . . . 109
set_emergence_colors . . . 110
sirfunctions_io . . . 111
test_EDAV_connection . . . 112
update_polio_data . . . 113
upload_dr_to_github . . . 113
upload_to_sharepoint . . . 114

Index **116**

`add_rolling_years` *Label rolling year periods*

Description

[Experimental]

The function labels and categorizes dates based on the rolling period specified. The start year will always be Year 1 and the rolling period is defined by the start date and the number of periods to account for in a given rolling year. For example, if the start date is defined as Jan 1, 2021 and we would like to calculate a 12-month rolling period, the end date would be Dec 31, 2021.

Usage

```
add_rolling_years(
  df,
  start_date,
  end_date,
  date_col,
  period = months(12, FALSE)
)
```

Arguments

`df`	`tibble` A dataset containing at least one date column.
`start_date`	`str` Start date of Year 1. All years are classified in reference to this date.
`end_date`	`str` End date to filter to.
`date_col`	`str` The name of the date column.
`period`	`period` A `lubridate::period()` object. Defaults to `months(12, FALSE)`.

Details

The function will filter data using the column specified by `date_col` up to the end date specified.

Value

`tibble` A tibble with rolling year information.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
afp_data <- add_rolling_years(raw_data$afp, "2021-01-01", "2024-05-02", "dateonset")

## End(Not run)
```

check_afp_geographies *Check whether the AFP geography matches those of the population dataset*

Description

[Experimental]

In rare cases, the GUIDs assigned for a case may be incorrect. For example, it may have a GUID that is incorrect for a specific year. This function checks each AFP record for such instances.

Usage

```
check_afp_geographies(afp_data, pop_data, spatial_scale, fix_afp = FALSE)
```

Arguments

afp_data	tibble AFP dataset
pop_data	tibble Population dataset
spatial_scale	str Any of the following: "ctry", "prov", "dist.
fix_afp	logical Whether to update the results to show corrected GUIDs based on the population dataset.

Value

tibble Tibble with a column used for checking accuracy.

Examples

```
## Not run:
raw_data <- get_all_polio_data(attach.spatial.data = FALSE)
check_afp <- check_afp_geographies(raw_data$afp, raw_data$ctry.pop, "ctry")

## End(Not run)
```

check_afp_guid_ctry_data *Check GUIDs present in the AFP linelist but not in the pop files*

Description

The function will run a check in the AFP linelist for GUIDs that are not part of the spatial files. In these instances, typically, unknown GUIDs are part of the new geodatabase from WHO that get released in the next updated geodatabase. Therefore, this function should be used only if necessary. For example, in instances where mapping an AFP case into a a district is critical and the shapefile from `extract_country_data()` is not yet updated.

Usage

```
check_afp_guid_ctry_data(ctry.data)
```

Arguments

ctry.data	list Country polio data, with spatial data attached. Output of extract_country_data() or init_dr().

Value

list A list containing errors in province and district GUIDs.

Examples

```
## Not run:
raw.data <- get_all_polio_data() # must contain spatial data to run the function
ctry.data <- extract_country_data("algeria", raw.data)
error.list <- check_afp_guid_ctry_data(ctry.data)

## End(Not run)
```

---

check_missing_rows *Check for rows with NA values*

Description

A general function that checks the number of NA rows for a particular column.

Usage

```
check_missing_rows(df, .col_name, .group_by)
```

Arguments

df	tibble Dataset to check.
.col_name	str Name of the target column.
.group_by	str or list A string or a list of strings to group the check by.

Value

tibble A summary of the number of rows missing for the target variable.

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
missing <- check_missing_rows(raw.data$afp, "age.months", c("place.admin.0", "yronset"))

## End(Not run)
```

clean_ctry_data *Cleans and adds additional age and dosage number columns to the AFP linelist*

Description

The function does additional cleaning of the `ctry.data` list. It fills in missing districts, convert character date columns to a date data type, calculates age group, add columns for the number of doses per case, and cleans the environmental surveillance data.

Usage

```
clean_ctry_data(ctry.data)
```

Arguments

`ctry.data`	`list` Large list containing polio country data. This is the output of `extract_country_data()` or `init_dr()`.

Value

`list` Cleaned country data list.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
ctry.data <- clean_ctry_data(ctry.data)

## End(Not run)
```

clean_es_data *Clean environmental surveillance data*

Description

The cleaning step will attempt to impute missing site coordinates and create standardized columns used in the desk review.

Usage

```
clean_es_data(es.data, dist.shape, ctry.data = lifecycle::deprecated())
```

Arguments

`es.data`	`tibble` Environmental surveillance data.
`dist.shape`	`sf` District shapefile.
`ctry.data`	**[Deprecated]** `list` This parameter has been deprecated in favor of explicitly passing dataframes into the function. This allows for greater flexibility in the function.

Value

tibble Cleaned environmental surveillance data.

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
ctry.data <- extract_country_data("algeria", raw.data)
ctry.data$es <- clean_es_data(ctry.data$es, ctry.data$dist)

## End(Not run)
```

---

clean_iss_data *Perform common cleaning tasks for ISS/eSURV data*

Description

ISS/eSURV data often needs to be cleaned and standardized before analysis. Because these datasets may vary from country to country, reviewing the data first and its columns is the first step. In general, there are eight required columns. These are the parameters with a suffix _col. Modify the passed arguments as necessary so the function can successfully run. Priority levels are set to automatically detect high, medium, low, and not a focal site. Ensure that priority level column categories have these specification:

- High: begins with "h".
- Medium: begins with "m".
- Low: begins with "l".
- Not Focal Site: begins with "n" or "x".

Usage

```
clean_iss_data(
  iss_data,
  start_date,
  end_date,
  priority_col = "priority_level",
  start_time_col = "starttime",
  unreported_cases_col = "num_unreportedcases",
  prov_col = "states",
  dist_col = "districts",
  hf_col = "name_of_facility_visited",
  today_col = "today",
  date_of_visit_col = "date_of_visit",
  ctry.data = lifecycle::deprecated()
)
```

Arguments

iss_data	tibble ISS data.
start_date	str Start date of desk review.
end_date	str End date of desk review.
priority_col	str Column representing priority level.
start_time_col	str Column representing start time.
unreported_cases_col	str Column representing unreported cases.
prov_col	str Column representing province.
dist_col	str Column representing district.
hf_col	str Column representing the health facility name.
today_col	str Column representing when info was recorded.
date_of_visit_col	str Column representing date of visit.
ctry.data	list **[Deprecated]** Please pass the ISS data directly to the iss.data parameter.

Value

tibble Cleaned eSurv/ISS data.

Examples

```
## Not run:
iss_path <- "C:/Users/ABC1/Desktop/iss_data.csv"
ctry.data <- init_dr("somalia", iss_data_path = iss_path)
ctry.data$iss.data <- clean_iss_data(ctry.data$iss.data, start_date, end_date)

## End(Not run)
```

---

clean_lab_data *Clean lab data*

Description

Main lab data cleaning function. Automatically detects whether the dataset came from WHO or the regional office.

Usage

```
clean_lab_data(
  lab_data,
  start_date,
  end_date,
  afp_data = NULL,
  ctry_name = NULL,
  lab_locs_path = NULL
)
```

Arguments

lab_data	tibble Lab dataset.
start_date	str Start date of analysis.
end_date	str End date of analysis.
afp_data	tibble AFP linelist. Either ctry.data$afp.all.2 or raw.data$afp.
ctry_name	str or list Name or a list of countries. Defaults to NULL.
lab_locs_path	str Location of testing lab locations. Default is NULL. Will download from EDAV, if necessary.

Value

tibble Cleaned lab data.

Examples

```
## Not run:
lab_path <- "C:/Users/XRG9/lab_data_who.csv"
ctry.data <- init_dr("algeria", lab_data_path = lab_path)
ctry.data$lab_data <- clean_lab_data(ctry.data, "2021-01-01", "2023-12-31")

# Not using the desk review pipeline
raw.data <- get_all_polio_data()
ctry.data <- extract_country_data("algeria", raw.data)
ctry.data$lab_data <- read_csv(lab_path)
ctry.data$lab_data <- clean_lab_data(
  ctry.data$lab.data, "2021-01-01", "2023-12-31",
  ctry.data$afp.all.2, "algeria"
)

## End(Not run)
```

---

compress_png *Compress PNG files using pngquant*

Description

Compress PNG files. The software pngquant is required to use this function. It attempts to reduce the file size of images without major loss in image quality. Files sizes can be reduced from 30-60% using this function. The compressed file will be outputted to the same folder as the original image.

Usage

```
compress_png(img, pngquant_path = NULL, suffix = "")
```

Arguments

img	str File path to the png file.
pngquant_path	str File path to pngquant executable file (pngquant.exe).
suffix	str Optional parameter to add a suffix to the compressed image.

Value

None. Will output compressed image to the local folder.

Examples

```
## Not run:
img_path <- "C:/Users/ABC1/Desktop/pic1.png"
pngquant_path <- "C:/Users/ABC1/Downloads/pngquant.exe"
compress_png(img_path, pngquant_path, "_compressed")

## End(Not run)
```

---

create_60_day_export *Export 60-day follow up table*

Description

Exports the output of generate_60_day_table_data into an Excel file.

Usage

```
create_60_day_export(
  cases.need60day,
  country = Sys.getenv("DR_COUNTRY"),
  excel_output_path = Sys.getenv("DR_TABLE_PATH")
)
```

Arguments

cases.need60day	tibble Summary table for 60-day follow-up.
country	str Name of the country.
excel_output_path	str Output path of the Excel file.

Value

None.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
stool.data <- generate_stool_data(
  ctry.data$afp.all.2, "good", "inadequate",
  "2021-01-01", "2023-12-31"
)
cases.need60day <- generate_60_day_table_data(stool.data, start_date, end_date)
create_60_day_export(cases.need60day)

## End(Not run)
```

create_afp_export *Export the AFP linelist*

Description

Export the AFP linelist with `adequacy.final2` column. The `adequacy.final2` column describes the status of a stool sample, such as if a stool sample is adequate or inadequate. Specifically, it is created from `generate_stool_data()` which takes parameters on how to deal with missing or inadequate stool samples.

Usage

```
create_afp_export(
  stool.data,
  country = Sys.getenv("DR_COUNTRY"),
  excel_output_path = Sys.getenv("DR_TABLE_PATH")
)
```

Arguments

`stool.data`	`tibble` AFP data with final adequacy columns. This is the output of `generate_stool_data()`.
`country`	`str` Name of the country.
`excel_output_path`	`str` Output path of the Excel file.

Value

None.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
stool.data <- generate_stool_data(
  ctry.data$afp.all.2, "good", "inadequate",
  "2021-01-01", "2023-12-31"
)
create_afp_export(stool.data)

## End(Not run)
```

create_emergence_group_gif *Generate Emergence Group Movement Gifs*

Description

Generate the figures and stitch together a GIF to evaluate emergence group movement over time, generally aggregated as cumulative per month

Usage

```
create_emergence_group_gif(
  emergence_group,
  pos,
  dist,
  ctry,
  include_env = T,
  cumulative = T,
  out_gif
)
```

Arguments

emergence_group	str Designation of the emergence group to review.
pos	tibble Positives data set. This is raw.data$pos, which is part of the output of get_all_polio_data().
dist	sf Shapefile of all districts.
ctry	sf Shapefile of all countries.
include_env	logical To include environmental detections in analysis. Defaults to TRUE.
cumulative	logical To display cases as cumulative. Defaults to TRUE.
out_gif	str Location where gif should be saved.

Value

GIF written out to location of out_gif.

Examples

```
## Not run:

data <- get_all_polio_data(size = "medium")
pos <- data$pos
emergence_group <- "NIE-JIS-1"
dist <- data$global.dist
ctry <- data$global.ctry
include_env <- T
cumulative <- F
out_gif <- getwd()

create_emergence_group_gif(
  emergence_group, pos, dist, ctry, include_env,
  cumulative, out_gif
)

## End(Not run)
```

`create_npafp_export` *Exports NPAFP indicator data summary tables*

Description

The function combines the NPAFP rate summary tables from `f.npafp.rate.01()` and exports to an Excel file, with each geographic level on its own tab.

Usage

```
create_npafp_export(
  ctry.case.ind,
  prov.case.ind,
  dist.case.ind,
  excel_output_path = Sys.getenv("DR_TABLE_PATH")
)
```

Arguments

`ctry.case.ind`	`tibble` Country NPAFP indicator summary table.
`prov.case.ind`	`tibble` Province NPAFP indicator summary table.
`dist.case.ind`	`tibble` District NPAFP indicator summary table.
`excel_output_path`	`str` Output path of the Excel file.

Value

None.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
ctry.case.ind <- f.npafp.rate.01(
  ctry.data$afp.all.2, ctry.data$ctry.pop,
  "2021-01-01", "2023-01-01", "ctry"
)
prov.case.ind <- f.npafp.rate.01(
  ctry.data$afp.all.2, ctry.data$prov.pop,
  "2021-01-01", "2023-01-01", "prov"
)
dist.case.ind <- f.npafp.rate.01(
  ctry.data$afp.all.2, ctry.data$dist.pop,
  "2021-01-01", "2023-01-01", "dist"
)
create_npafp_export(ctry.case.ind, prov.case.ind, dist.case.ind)

## End(Not run)
```

create_pop_check_export

Exports file for checking population roll-ups

Description

Export the population roll-ups and determine differences between each population counts.

Usage

```
create_pop_check_export(
  ctry.data,
  country = Sys.getenv("DR_COUNTRY"),
  excel_output_path = Sys.getenv("DR_TABLE_PATH")
)
```

Arguments

ctry.data	list A large list containing polio data for a country. This is the output of either init_dr() or extract_country_data().
country	str Name of the country.
excel_output_path	str Output path of the Excel file.

Value

None.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
create_pop_check_export(ctry.data)

## End(Not run)
```

create_pot_comp_clust_export

Export potentially compatible and compatible summary table

Description

Exports the output of generate_potentially_compatibles_cluster() as an Excel file.

Usage

```
create_pot_comp_clust_export(
  pot.c.clust,
  country = Sys.getenv("DR_COUNTRY"),
  excel_output_path = Sys.getenv("DR_TABLE_PATH")
)
```

Arguments

pot.c.clust	tibble Potentially compatible cluster summary. The output of generate_potentially_compatible
country	str Name of the country.
excel_output_path	str Output path of where to store the Excel file.

Value

None.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
stool.data <- generate_stool_data(
  ctry.data$afp.all.2, "good", "inadequate",
  "2021-01-01", "2023-12-31"
)
cases.need60day <- generate_60_day_table_data(stool.data, start_date, end_date)
pot.c.clust <- generate_potentially_compatibles_cluster(cases.need60day)
create_pot_comp_clust_export(pot.c.clust)

## End(Not run)
```

---

create_stool_adequacy_export

Export stool adequacy data

Description

The function combines the stool adequacy summary tables from f.stool.ad.01() and exports to an Excel file, with each geographic level on its own tab.

Usage

```
create_stool_adequacy_export(
  cstool,
  pstool,
  dstool,
  excel_output_path = Sys.getenv("DR_TABLE_PATH")
)
```

Arguments

cstool	tibble Stool adequacy at country level.
pstool	tibble Stool adequacy at province level.
dstool	tibble Stool adequacy at district level.
excel_output_path	str Output path.

Value

None.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
cstool <- f.stool.ad.01(
  ctry.data$afp.all.2, ctry.data$ctry.pop,
  "2021-01-01", "2023-01-01", "ctry"
)
pstool <- f.stool.ad.01(
  ctry.data$afp.all.2, ctry.data$prov.pop,
  "2021-01-01", "2023-01-01", "prov"
)
dstool <- f.stool.ad.01(
  ctry.data$afp.all.2, ctry.data$dist.pop,
  "2021-01-01", "2023-01-01", "dist"
)
create_stool_adequacy_export(cstool, pstool, dstool)

## End(Not run)
```

---

ctry_data_errors *Check data quality errors from the country data*

Description

Performs a check for different errors in the AFP linelist and population files. It also alerts the users for GUIDs that have changed.

Usage

```
ctry_data_errors(ctry.data, error_path = Sys.getenv("DR_ERROR_PATH"))
```

Arguments

ctry.data	list Large list containing polio country data. This is the output of extract_country_data() or init_dr().
error_path	str Path where to store checks in ctry.data.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
ctry_data_errors(ctry.data)

## End(Not run)
```

duplicate_check *Assess duplicates in the get_all_polio_data() output*

Description

Checks for duplicate records in AFP, other, SIA, and Virus datasets.

Usage

```
duplicate_check(.raw.data = raw.data)
```

Arguments

.raw.data	Named list output of get_all_polio_data()

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
raw.data <- duplicate_check(raw.data)

## End(Not run)
```

edav_io *Helper function to read and write key data to the EDAV environment*

Description

The function serves as the primary way to interact with the EDAV system from R. It can read, write, create folders, check whether a file or a folder exists, upload files, and list all files in a folder.

Usage

```
edav_io(
  io,
  default_dir = "GID/PEB/SIR",
  file_loc = NULL,
  obj = NULL,
  azcontainer = suppressMessages(get_azure_storage_connection()),
  force_delete = F,
  local_path = NULL,
  ...
)
```

Arguments

io	str The type of operation to perform in EDAV. • "read" Read a file from EDAV, must be an rds, csv, rda, or xls/xlsx file. • "write" Write a file to EDAV, must be an rds, csv, rda, or xls/xlsx file. To write an Excel file with multiple sheets, pass a named list containing the tibbles of interest. See examples. • "exists.dir" Returns a boolean after checking to see if a folder exists. • "exists.file"Returns a boolean after checking to see if a file exists. • "create" Creates a folder and all preceding folders. • "list" Returns a tibble with all objects in a folder. • "upload" Moves a file of any type to EDAV. • "delete" Deletes a file. • "delete.dir" Deletes a folder.
default_dir	str The default directory in EDAV. "GID/PEB/SIR" is the default directory for all SIR data in EDAV. Can be set to NULL if you provide the full directory path in file_loc.
file_loc	str Location to "read", "write", "exists.dir", "exists.file", "create" or "list", can include the information in default_dir if you set that parameter to NULL.
obj	robj Object to be saved, needed for "write". Defaults to NULL.
azcontainer	Azure container object returned from get_azure_storage_connection().
force_delete	logical Use delete io without verification in the command line.
local_path	str Local file pathway to upload a file to EDAV. Default is NULL. This parameter is only required when passing "upload" in the io parameter.
...	Optional parameters that work with readr::read_delim() or readxl::read_excel().

Value

Output dependent on argument passed in the io parameter.

Examples

```
## Not run:
df <- edav_io("read", file_loc = "df1.csv") # read file from EDAV
# Passing parameters that work with read_csv or read_excel, like sheet or skip.
df2 <- edav_io("read", file_loc = "df2.xlsx", sheet = 1, skip = 2)
list_of_df <- list(df_1 = df, df_2 = df)
# Saves df to the test folder in EDAV
edav_io("write", file_loc = "Data/test/df.csv", obj = df)
# Saves list_of_df as an Excel file with multiple sheets.
edav_io("write", file_loc = "Data/test/df.xlsx", obj = list_of_df)
edav_io("exists.dir", "Data/nonexistentfolder") # returns FALSE
edav_io("exists.file", file_loc = "Data/test/df1.csv") # returns TRUE
edav_io("create", "Data/nonexistentfolder") # creates a folder called nonexistentfolder
edav_io("list") # list all files from the default directory
edav_io("upload", file_loc = "Data/test", local_path = "C:/Users/ABC1/Desktop/df2.csv")

## End(Not run)
```

`explore_edav` *Interactive loading of EDAV data*

Description

[Experimental]

This function is a way to interactively work with files in the EDAV environment, which is convenient as we don't have to search for files within Azure Storage Explorer.

Usage

```
explore_edav(
  path = get_constant("DEFAULT_EDAV_FOLDER"),
  azcontainer = suppressMessages(get_azure_storage_connection())
)
```

Arguments

`path`	`str` Path to start at initially.
`azcontainer`	Azure storage container provided by `get_azure_storage_connection()`.

Details

There are Excel files that may need additional formatting before it can be read properly into an R object. For example, skipping columns or rows. For complicated Excel files, it would be best to directly call `edav_io()` in "read" mode, and pass additional parameters via `....` See `edav_io()` examples for details.

Value

`tibble` Data from the EDAV environment.

Examples

```
## Not run:
test <- explore_edav()

## End(Not run)
```

`export_kpi_table` *Export KPI tables*

Description

[Experimental]

Performs formatting and export of the C1-C4 KPI tables.

Usage

```
export_kpi_table(
  c1 = NULL,
  c2 = NULL,
  c3 = NULL,
  c4 = NULL,
  output_path = Sys.getenv("KPI_TABLES"),
  drop_label_cols = FALSE,
  sc_targets = FALSE,
  pos_data = NULL,
  risk_table = NULL
)
```

Arguments

c1	tibble Output of generate_c1_table(). Defaults to NULL.
c2	tibble Output of generate_c2_table(). Defaults to NULL.
c3	tibble Output of generate_c3_table(). Defaults to NULL.
c4	tibble Output of generate_c4_table(). Defaults to NULL.
output_path	str Path to output the table to. Defaults to the path initiated after running init_kpi().
drop_label_cols	logical Keep or discard label columns. Defaults to TRUE.
sc_targets	logical Whether to use SC targets when exporting the table. Defaults to FALSE.
pos_data	tibble Positives dataset.
risk_table	tibble The risk table. Required if using sc_targets and outside of CDC.

Value

None.

Examples

```
## Not run:
init_kpi()
c1 <- generate_c1_table(raw_data, "2021-01-01", "2023-12-31")
c2 <- generate_c2_table(raw_data$afp, raw_data$ctry.pop, "2021-01-01", "2023-12-31", "ctry")
c3 <- generate_c3_table(raw_data$es, "2021-01-01", "2023-12-31")
c4 <- generate_c4_table(lab_data, raw_data$afp, "2021-01-01", "2024-12-31")
export_kpi_table(c1, c2, c3, c4)

## End(Not run)
```

extract_country_data *Extract country specific information from raw polio data*

Description

Filters country specific data from the CDC generated `raw.data` object from `get_all_polio_data()`.

Usage

```
extract_country_data(.country, .raw.data = raw.data)
```

Arguments

`.country`	`str` Country name of interest. Case insensitive.
`.raw.data`	`list` Output of `get_all_polio_data()`.

Value

Named `list` with country specific datasets.

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
ctry.data <- extract_country_data("nigeria", raw.data)

## End(Not run)
```

f.color.schemes *Utility function for colors*

Description

Utility function to return SIR color schemes used in various graphs and visualizations.

Usage

```
f.color.schemes(type)
```

Arguments

`type`	`str` Type of colors we can return. Accepted values include: • `"epicurve"` Mapped to different `cdc.classification.all2` values. • `"para.case"` A subset of `"epicurve"` representing paralytic cases. • `"afp.prov"` Mapped for case counts at the province level. • `"afp.dist"` Mapped for case counts at the province level. • `"pot.comp"` Colors for categories of compatibles and potentially compatibles.

- "silence" Colors to use to map silent populations.
- "silence.v2" Colors to use to map silent populations.
- "cases" Values to map case type.
- "es" Values used in ES data.
- "mapval" Values used for creating maps with percentages.
- "timeliness.col.vars" Mapping intervals used for lab timeliness intervals graphs.
- "emergence.groups" Standard emergence group colors. Used primarily with generate_adhoc_map().
- "es.vaccine.types" Default vaccine types. Used primarily with generate_es_site_det().
- "es.detections" Default detection types. Used primarily with generate_es_site_det().
- "vpd.critical.ctry": Priority VPD countries
- "vpd.binary.ctry": Priority and non-priority VPD countries binary designation.
- "vpd.colors": VPD colors.

Value

Named list with color sets.

Examples

```
color_list <- f.color.schemes("epicurve")
```

---

f.ev.rate.01 *Calculate EV detection rate function*

Description

Function to calculate the EV detection rate in sites from POLIS.

Usage

```
f.ev.rate.01(es.data, start.date, end.date)
```

Arguments

es.data	tibble ES data which includes site name (site.name), country (ADM0_NAME), date of collection (collect.date), and a binary ev detection variable (ev.detect) that indicates absence/presence (0, 1) of enterovius in an ES sample. This is ctry.data$es of extract_country_data() or init_dr(), or raw.data$es of get_all_polio_data().
start.date	str Date in the format of "YYYY-MM-DD".
end.date	str Date in the format of "YYYY-MM-DD".

Value

tibble Long format dataframe including site specific EV detection rates.

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
ctry.data <- extract_country_data("algeria", raw.data)
ev_rates <- f.ev.rate.01(ctry.data$es, "2021-01-01", "2023-12-31")

## End(Not run)
```

`f.expand.bbox` *Expand bounding box*

Description

Sourced from https://rdrr.io/github/Chrisjb/basemapR/src/R/expand_bbox.R. A function to take a bounding box (generated using `sf::st_bbox()`) and expand it by x meters in the X direction and y meters in the Y direction.

Usage

```
f.expand.bbox(bbox, X, Y, X2 = X, Y2 = Y, crs_out = 4326)
```

Arguments

bbox	bbox A bounding box generated by `sf::st_bbox()`.
X	`numeric` The distance in meters that we want to expand the bounding box by in the X direction.
Y	`numeric` The distance in meters that we want to expand the bounding box by in the Y direction.
X2	`numeric` If specified, the meters in the Easterly direction and meters X becomes meters in the Westerly direction.
Y2	`numeric` If specified, the meters to the South. meters Y becomes meters to the North.
crs_out	`int` EPSG coordinate system to return the bounding box in. Defaults to 4326 (lat/lng).

Value

A bbox object. This can be converted into an `sf` object using `sf::st_as_sfc()`.

Examples

```
## Not run:
ctry.shape <- load_clean_ctry_sp(ctry_name = "ALGERIA", st.year = 2019)
ctry.bbox <- sf::st_bbox(ctry.shape)
bbox_2 <- f.expand.bbox(ctry.bbox, 4, 4)

## End(Not run)
```

f.metadata.tag *Function to add metadata tags to figures and tables*

Description

Add metadata tags to figures and tables. These include the download date of the dataset. The function will return an error if both raw_data and time_tag parameters are NULL.

Usage

```
f.metadata.tag(object, raw_data = NULL, time_tag = NULL)
```

Arguments

object	ggplot or flextable The figure or table to add metadata to.
raw_data	list outputs of get_all_polio_data() or extract_country_data().
time_tag	str A date and time string. Defaults to raw.data$metadata$download_time.

Value

A ggplot or flextable object with metadata added.

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
df <- datasets::iris
p1 <- ggplot2::ggplot() +
  ggplot2::geom_col(data = df, ggplot2::aes(x = Sepal.Length, y = Sepal.Width))
p2 <- f.metadata.tag(p1, raw.data) # use raw.data download time
p3 <- f.metadata.tag(p1, time_tag = "2021-01-01") # use custom time tag

## End(Not run)
```

f.npafp.rate.01 *Calculate non-polio AFP rate*

Description

[Stable]

Calculate the NPAFP rate from POLIS data. Can either pass raw.data to calculate NPAFP rates on the global dataset, or a ctry.data dataset.

Usage

```
f.npafp.rate.01(
  afp.data,
  pop.data,
  start.date,
  end.date,
  spatial.scale,
  pending = T,
  missing_agemonths = F,
  rolling = F,
  sp_continuity_validation = T
)
```

Arguments

afp.data	tibble AFP data which includes GUID at a given spatial scale formatted as adm(0,1,2)guid, onset date as date and cdc.classification.all2 which includes "NPAFP", "PENDING", "LAB PENDING". This is either ctry.data$afp.all.2 of extract_country_data() or init_dr() or raw.data$afp of get_all_polio_data().
pop.data	tibble Under 15 population data by a given spatial scale including year, adm(0,1,2)guid, u15pop, and ctry/prov/dist as appropriate. This is part of the output of get_all_polio_data() and extract_country_data().
start.date	str Start date with the format "YYYY-MM-DD".
end.date	str Start date with the format "YYYY-MM-DD".
spatial.scale	str Spatial scale for analysis. • "prov" Province level. • "dist" District level. • "ctry" Country level.
pending	logical Should cases classified as PENDING or LAB PENDING be included in calculations? Default TRUE.
missing_agemonths	logical Should cases with NA values for age.months be included? Default FALSE.
rolling	logical Should the analysis be performed on a rolling bases? Default FALSE.
sp_continuity_validation	logical Should we filter places that are not present for the entirety of the analysis dates? Default TRUE.

Value

tibble A table containing NPAFP rates as well as additional information relevant to each location analyzed.

Examples

```
## Not run:
raw.data <- get_all_polio_data()
npafp_ctry <- f.npafp.rate.01(raw.data$afp, raw.data$ctry.pop, "2022-01-01", "2024-12-31", "ctry")

## End(Not run)
```

f.plot.looks *Set plot looks*

Description

The function serves to collate and return plot looks. Depending on the parameter, specific values in a ggplot2 theme object will be returned.

Usage

```
f.plot.looks(type)
```

Arguments

type	str	Type of graph format. Accepted values include: • "02" • "epicurve" • "geomtile" • "gpln_type1" • "gpln_type2"

Value

ggplot2 theme obj A theme object that can be added into an existing plot.

Examples

```
## Not run:
epicurve_looks <- f.plot.looks("epicurve")
df <- datasets::iris
p1 <- ggplot2::ggplot() +
  ggplot2::geom_col(data = df, ggplot2::aes(x = Sepal.Length, y = Sepal.Width))
p2 <- p1 + epicurve_looks

## End(Not run)
```

f.stool.ad.01 *Calculate percent stool adequacy*

Description

Creates a summary table of stool adequacy. The missing parameter defines how missing data is treated. "good" classifies missing data as good quality (POLIS method). "bad" classifies all missing as bad quality. "missing" excludes missing from the calculations.

Usage

```
f.stool.ad.01(
  afp.data,
  pop.data,
  start.date,
  end.date,
  spatial.scale,
  missing = "good",
  bad.data = "inadequate",
  rolling = F,
  sp_continuity_validation = T,
  admin.data = lifecycle::deprecated()
)
```

Arguments

afp.data	tibble AFP data which includes GUID at a given spatial scale formatted as adm(0,1,2)guid, onset date as date and cdc.classification.all2 which includes "NOT-AFP".
pop.data	tibble Full list of country administrative units by a given spatial scale including year, adm(0,1,2)guid, and ctry/prov/dist (as appropriate).
start.date	str Starting date for analysis formatted as "YYYY-MM-DD".
end.date	str Ending date for analysis as "YYYY-MM-DD".
spatial.scale	str Geographic level to group analysis on. • "prov" Province level. • "dist" District level. • "ctry" Country level.
missing	str How to treat missing data. Valid values are: "good", "bad", "remove". Defaults to "good". When calculating the adequacy.final column: • "good" uses adequacy.03 • "bad" uses adequacy.01 • "exclude" uses adequacy.02
bad.data	str How to treat bad data. Valid values are:"remove", "inadequate". Defaults to "inadequate". "inadequate" treats samples with bad data as inadequate.
rolling	logical Should data be analyzed on a rolling bases? Defaults to FALSE.
sp_continuity_validation	logical Should GUIDs not present in all years of the dataset be excluded? Default TRUE.
admin.data	tibble Population data. Renamed in favor of pop.data.

Value

tibble Long format stool adequacy evaluations.

Examples

```
## Not run:
raw.data <- get_all_polio_data()
stool.ads <- f.stool.ad.01(raw.data$afp, raw.data$ctry.pop,
  "2021-01-01", "2023-12-31",
  "ctry",
  sp_continuity_validation = FALSE
)

## End(Not run)
```

`f.timely.detection.01` *Function to calculate timeliness of detection*

Description

Calculates the overall timeliness of detection in AFP & ES POLIS data.

Usage

```
f.timely.detection.01(
  afp.data,
  es.data,
  ctryseq.data,
  start.date,
  end.date,
  rolling = F
)
```

Arguments

`afp.data`	`tibble` AFP data which includes classification of AFP cases with onset date and date of notification to HQ.
`es.data`	`tibble` ES data which includes classification of samples with collection date and date of notification to HQ.
`ctryseq.data`	`tibble` A table consisting of the following columns for each country: • With sequencing capacity within or outside of the country • Country (`ADM0_NAME`) • Classification of AFP cases & ES samples • Onset date of AFP cases and collection date of ES samples • Date of notification to HQ (`date.notification.to.hq`) This table is the output of `get_lab_locs()`.
`start.date`	`str` Start date for evaluation with format `"YYYY-MM-DD"`.
`end.date`	`str` End date for evaluation with format `"YYYY-MM-DD"`.
`rolling`	`logical` Should timeliness be calculated in a rolling basis? Default `FALSE`.

Value

`list` A list with two `tibbles` with global and sub-global AFP / ES detection timeliness evaluation.

Examples

```
## Not run:

raw.data <- get_all_polio_data()
ctry.data <- extract_country_data("algeria", raw.data)
ctry.seq <- get_lab_locs()
global.summary <- f.timely.detection.01(
  raw.data$afp, raw.data$es, ctry.seq,
  "2021-01-01", "2023-12-31"
)
ctry.summary <- f.timely.detection.01(
  ctry.data$afp.all.2, ctry.data$es, ctry.seq,
  "2021-01-01", "2023-12-31"
)

## End(Not run)
```

`fix_ctry_data_missing_guids`

Fix unknown GUIDs in the AFP linelist

Description

Fix unknown GUIDs in the AFP linelist by obtaining GUIDs found in the pop files. It attempts to replace the unknown GUIDs from the AFP linelist by using geographic info for a specific year that coincides with the case date and uses the GUIDs contained in the current spatial data instead.

Usage

```
fix_ctry_data_missing_guids(afp.data, pop.data, guid_list, spatial_scale)
```

Arguments

`afp.data`	`tibble` AFP linelist (afp.all.2).
`pop.data`	`tibble` Population file (prov.pop or dist.pop).
`guid_list`	`str list` Unknown GUIDs from the AFP linelist. This is the output of `check_afp_guid_ctry_data`
`spatial_scale`	`str` The spatial scale to impute data. Either `"prov"` or `"dist"`.

Value

`tibble` AFP data with corrected GUIDs based on the population files.

Examples

```
## Not run:
raw.data <- get_all_polio_data()
ctry.data <- extract_country_data("algeria", raw.data)
error.list <- check_afp_guid_ctry_data(ctry.data)
ctry.data$afp.all.2 <- fix_ctry_data_missing_guids(
  ctry.data$afp.all.2,
  ctry.data$dist.pop,
```

```
  error.list$dist_mismatches_pop,
  "dist"
)

## End(Not run)
```

`freeze_dr_data` *Freeze desk review data to the desk review folder in EDAV*

Description

Data from the desk review can be stored in EDAV so there's an exact copy of the dataset used in the desk review. This ensures that even after years, the desk reviews can be ran exactly as it was.

Usage

```
freeze_dr_data(
  rds_obj,
  file_name,
  country = Sys.getenv("DR_COUNTRY"),
  year = as.numeric(format(Sys.Date(), "%Y"))
)
```

Arguments

`rds_obj`	`Robj` Object loaded in R. This would be `ctry.data`, for example.
`file_name`	`str` Name given to the Rds object, do not append `.rds`. This is what gets stored in EDAV.
`country`	`str` Country as a string.
`year`	`int` It is recommended to set this to the year when the desk review was ran.

Value

A status message.

See Also

`init_dr()`

Examples

```
## Not run:
raw.data <- get_all_polio_data()
ctry.data <- init_dr("algeria")
freeze_dr_data(ctry.data, "algeria_ctry_data")

## End(Not run)
```

generate_60_day_tab *60-day follow up table*

Description

Generates a table summarizing the number of inadequate cases that need follow up.

Usage

```
generate_60_day_tab(cases.need60day)
```

Arguments

cases.need60day	tibble Summary table containing those that need 60 day follow-up. Output of generate_60_day_table_data().

Value

flextable A summary of cases requiring 60-day followups per year.

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
ctry.data <- extract_country_data("algeria", raw.data)
stool.data <- generate_stool_data(
  ctry.data$afp.all.2, "good", "inadequate",
  "2021-01-01", "2023-12-31"
)
cases.need60day <- generate_60_day_table_data(
  stool.data,
  "2021-01-01", "2023-12-31"
)
generate_60_day_tab(cases.need60day)

## End(Not run)
```

generate_60_day_table_data *Generate summary table for those requiring 60-day follow-up*

Description

The 60-day table highlights the number of cases per year that need 60-day follow-up. It summarizes the number of cases due for follow up, those with recorded follow ups, number missing follow ups, and compatible cases.

Usage

```
generate_60_day_table_data(stool.data, start_date, end_date)
```

Arguments

stool.data	tibble AFP data with stool adequacy columns. This is the output of generate_stool_data().
start_date	str Start date of analysis.
end_date	str End date of analysis.

Value

tibble A summary table for those requiring 60-day follow-up.

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
ctry.data <- extract_country_data("algeria", raw.data)
stool.data <- generate_stool_data(
  ctry.data$afp.all.2,
  "2021-01-01", "2023-12-31",
  "good", "inadequate"
)
table60.days <- generate_60_day_table_data(stool.data, "2021-01-01", "2023-12-31")

## End(Not run)
```

---

generate_adhoc_map *Create adhoc maps for emergences*

Description

Creates a map of recent emergences. The default will display outbreaks from the past 13 months.

Usage

```
generate_adhoc_map(
  raw.data,
  country,
  virus_type = "cVDPV 2",
  vdpv = T,
  new_detect = T,
  surv = c("AFP", "ES", "OTHER"),
  labels = "YES",
  owner = "CDC-GID-PEB",
  new_detect_expand = F,
  start_date = NULL,
  end_date = NULL,
  emg_cols = NULL,
  output = NULL,
  image_size = NULL,
  height = 6.2,
  width = 4.5,
  scale = 1.25,
  dpi = 300
)
```

Arguments

raw.data	list Global polio data. The output of get_all_polio_data(). Make sure the spatial data is attached, otherwise, it will not work.
country	str or list Country name or a list of country names.
virus_type	str or list. Virus type to include. Valid values are: "cVDPV 1", "cVDPV 2", "cVDPV 3", "WILD 1". Can pass as a list.
vdpv	logical Whether to include VPDV in maps. Default TRUE.
new_detect	logical Whether to highlight new detections based on WHO HQ report date. Default TRUE.
surv	str or list Surveillance options. Valid values are: "AFP", "ES", "OTHER" "OTHER" includes Case Contact, Community, Healthy Children Sampling. Can pass as a list.
labels	str Include labels for regions with virus detections. Options: • "ALL": All regions • "YES": Recent Detections - <13 months
owner	str Who produced the map. Defaults to "CDC-GID-PEB".
new_detect_expand	logical Whether to expand the reporting window. Defaults to FALSE.
start_date	str Start date. If not specified, defaults to 13 months prior to the download date of raw.data.
end_date	str End date. If not specified, defaults to the download date of raw.data.
emg_cols	list A named list with all of the emergence colors. Defaults to NULL, which will download using set_emergence_colors().
output	str Either a path to a local folder to save the map to, "sharepoint", or NULL. Defaults to NULL.
image_size	str Standard sizes of the map outputs. Options are: • "full_slide" • "soco_slide" • "half_slide" Defaults to NULL.
height	numeric Height of the map. Defaults to 6.2.
width	numeric Width of the map. Defaults to 4.5.
scale	numeric Scale of the map. Defaults to 1.25.
dpi	numeric DPI of the map. Defaults to 300.

Value

ggplot A map of outbreaks.

Examples

```
## Not run:
raw.data <- get_all_polio_data()
p1 <- generate_adhoc_map(raw.data, "algeria")
# Put colors in emergences that don't have a mapped color
emg_cols <- set_emergence_colors(raw.data, c("nigeria", "chad"))
emg_cols["NIE-BOS-1"] <- "yellow"
emg_cols["NIE-YBS-1"] <- "green"
p2 <- generate_adhoc_map(raw.data, c("nigeria", "chad"), emg_cols = emg_cols)

## End(Not run)
```

---

generate_ad_final_col *Helper function to add the* adequacy.final *column*

Description

The function is meant to be used for f.stool.ad.01(). This function will classify the adequacy of a stool sample based on timeliness and condition.

Usage

```
generate_ad_final_col(afp.data)
```

Arguments

afp.data	tibble AFP dataset. Either raw.data$afp from get_all_polio_data() or ctry.data$afp.all.2 from extract_country_data().

Value

tibble AFP dataset with adequacy.final column

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
stool.data <- generate_ad_final_col(raw.data$afp)

## End(Not run)
```

`generate_afp_by_month_summary` *Generate AFP case count summary*

Description

[Stable]

Summarize AFP case counts by month and another grouping variable.

Usage

```
generate_afp_by_month_summary(
  afp_data,
  start_date,
  end_date,
  by,
  pop_data = NULL,
  ctry.data = lifecycle::deprecated()
)
```

Arguments

`afp_data`	`tibble` AFP dataset.
`start_date`	`str` Start date of analysis.
`end_date`	`str` End date of analysis.
`by`	`str` How to group the data by. Either `"prov"`, `"dist"`, or `"year"`.
`pop_data`	`tibble` Population dataset.
`ctry.data`	**[Deprecated]** `ctry.data` is no longer supported; the function will explicitly ask for the AFP dataset instead of accessing it from a list.

Value

`tibble` Summary table of AFP cases by month and another grouping variable.

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
ctry.data <- extract_country_data("algeria", raw.data)
afp.by.month <- generate_afp_by_month_summary(
  raw.data$afp, "2021-01-01", "2023-12-31", "ctry",
  raw.data$ctry.pop
)

## End(Not run)
```

`generate_afp_case_map` *AFP case map*

Description

Generates a map of AFP cases, excluding any with pending classification.

Usage

```
generate_afp_case_map(
  afp.all,
  ctry.shape,
  prov.shape,
  start_date,
  end_date = lubridate::today(),
  output_path = Sys.getenv("DR_FIGURE_PATH")
)
```

Arguments

`afp.all`	`sf` AFP linelist containing point geometry. This is `ctry.data$afp.all`, which is an output of either `extract_country_data()` and `init_dr()`.
`ctry.shape`	`sf` Country shapefile in long format.
`prov.shape`	`sf` Province shapefile in long format.
`start_date`	`str` Start date of analysis.
`end_date`	`str` End date of analysis. Default is today's date.
`output_path`	`str` Local path where to save the figure to.

Value

`ggplot` Map of AFP cases.

See Also

`load_clean_ctry_sp()`, `load_clean_prov_sp()`

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
ctry.shape <- load_clean_ctry_sp(ctry_name = "ALGERIA", type = "long")
prov.shape <- load_clean_prov_sp(ctry_name = "ALGERIA", type = "long")
generate_afp_case_map(ctry.data, ctry.shape, prov.shape, "2023-12-31")

## End(Not run)
```

generate_afp_epicurve *Epicurve of AFP cases by year*

Description

Generates an epicurve line graph of AFP cases by year.

Usage

```
generate_afp_epicurve(
  ctry.data,
  start_date,
  end_date = lubridate::today(),
  output_path = Sys.getenv("DR_FIGURE_PATH")
)
```

Arguments

ctry.data	list Large list containing country polio data. This is the output of either extract_country_data() or init_dr().
start_date	str Start date of analysis.
end_date	str End date of analysis. By default, it is up to the current date.
output_path	str Local path location to save the figure.

Value

ggplot A line graph of AFP cases faceted by year.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
generate_afp_epicurve(ctry.data, start_date)

## End(Not run)
```

generate_afp_prov_year *AFP cases by province and year*

Description

Generates a tile plot for the number of AFP cases per month by province.

Usage

```
generate_afp_prov_year(
  afp.by.month.prov,
  start_date,
  end_date = lubridate::today(),
  output_path = Sys.getenv("DR_FIGURE_PATH")
)
```

Arguments

afp.by.month.prov	tibble Table summarizing AFP cases by month and province. This is the output of generate_afp_by_month_summary().
start_date	str Start date of the analysis.
end_date	str End date of the analysis. By default, it displays the most recent date.
output_path	str Local path to output the figure.

Value

ggplot A tile plot displaying the number of AFP cases by month and province.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
afp.by.month <- generate_afp_by_month(ctry.data$afp.all.2, start_date, end_date)
afp.by.month.prov <- generate_afp_by_month_summary(
  afp.by.month, ctry.data,
  start_date, end_date, "prov"
)
generate_afp_prov_year(afp.by.month.prov, start_date, end_date)

## End(Not run)
```

---

generate_c1_rollup *Generate C1 rollup for high-priority countries*

Description

[Experimental]

Generates a summary of how many of the high priority countries have met their AFP and ES indicators.

Usage

```
generate_c1_rollup(
  c1,
  priority_level = "HIGH",
  who_region = NULL,
  .group_by = "rolling_period",
```

```
  npafp_target = 80,
  stool_target = 80,
  ev_target = 80,
  timely_wpv_vdpv_target = 80
)
```

Arguments

c1	tibble The output of generate_c1_table().
priority_level	str or list Priority level. Defaults to "HIGH". Valid values are "LOW", "LOW (WATCHLIST)", "MED
who_region	str WHO region to summarize the data to.
.group_by	str How the rollup should be grouped. Defaults to the column "rolling_period".
npafp_target	num Target used when calculating the proportion of districts in a country that meets NPAFP rate.
stool_target	num Target used when calculating the proportion of districts in a country that meets stool adequacy rate.
ev_target	num Target used when calculating the proportion of ES sites in a country that meets EV detection rate.
timely_wpv_vdpv_target	num Target used when calculating the proportion of ES sites in a country that meets timeliness of detection of WPV and VDPV cases.

Value

tibble A summary rollup

Examples

```
## Not run:
raw_data <- get_all_polio_data()
c1 <- generate_c1_table(raw_data, "2022-01-01", "2024-12-31")
c1_rollup <- generate_c1_rollup(c1)

## End(Not run)
```

---

generate_c1_table *GPEI Strategy surveillance KPIs*

Description

[Experimental]

Monitoring surveillance KPIs for Certification of Poliomyelitis Eradication at country regional and global levels.

Usage

```
generate_c1_table(
  raw_data,
  start_date,
  end_date,
  risk_category = NULL,
  risk_table = NULL,
  lab_locs = NULL
)
```

Arguments

raw_data	list Global polio surveillance dataset. Output of get_all_polio_data().
start_date	str Start date of the analysis in YYYY-MM-DD format.
end_date	str End date of the analysis in YYYY-MM-DD format.
risk_category	str Risk category or a list of categories. Defaults to NULL. Valid values are: "LOW, LOW (WATCHLIST), MEDIUM, HIGH.
risk_table	tibble Priority level of each country. Defaults to NULL, which will download the information directly from EDAV.
lab_locs	tibble Summary of the sequencing capacities of labs. Output of get_lab_locs(). Defaults to NULL, which will download the information directly from EDAV. .

Value

tibble Summary table of GPSAP KPIs.

Examples

```
## Not run:
raw_data <- get_all_polio_data(attach.spatial.data = FALSE)
c1 <- generate_c1_table(raw_data, "2021-01-01", "2023-12-31")

## End(Not run)
```

---

generate_c2_table *AFP surveillance KPI summary*

Description

[Experimental]

This function creates a summary table of AFP surveillance KPIs.

Usage

```
generate_c2_table(
  afp_data,
  pop_data,
  start_date,
  end_date,
  spatial_scale,
```

```
  risk_category = NULL,
  lab_locs = NULL,
  risk_table = NULL
)
```

Arguments

afp_data	tibble AFP linelist data.
pop_data	tibble Population data.
start_date	str Start date of the analysis in YYYY-MM-DD format.
end_date	str End date of the analysis in YYYY-MM-DD format.
spatial_scale	str Either "ctry", "prov", "dist".
risk_category	str Risk category or a list of categories. Defaults to NULL. Valid values are: "LOW, LOW (WATCHLIST), MEDIUM, HIGH.
lab_locs	tibble Summary of the sequencing capacities of labs. Output of get_lab_locs(). Defaults to NULL, which will download the information directly from EDAV. .
risk_table	tibble Priority level of each country. Defaults to NULL, which will download the information directly from EDAV.

Value

tibble Summary table containing AFP KPIs.

Examples

```
## Not run:
raw_data <- get_all_polio_data(attach.spatial.data = FALSE)
c2 <- generate_c2_table(raw_data$afp, raw_data$ctry.pop, "2021-01-01", "2023-12-31", "ctry")

## End(Not run)
```

---

generate_c3_rollup *Create a country level rollup of the C3 label*

Description

[Experimental]

Create a country level summary of ES site performance with respect to meeting established targets for EV detection rates, good samples. Note, this country roll up will only consider sites that have at least 10 collections and open for at least 12 months, consistent with guidelines in the 2025-2026 GPSAP indicators.

Usage

```
generate_c3_rollup(
  c3,
  include_labels = TRUE,
  min_sample = 10,
  timely_wpv_vdpv_target = 80
)
```

Arguments

c3	tibble Output of generate_c3_table().
include_labels	logical Include columns for the labels? Default TRUE.
min_sample	num Only consider sites with at least this number of ES samples. Default is 10.
timely_wpv_vdpv_target	Target used when determining whether a country meets EV detection target.

Value

tibble A summary of the c3 table at the country level

Examples

```
## Not run:
raw_data <- get_all_polio_data()
c3 <- generate_c3_table(raw_data$es, "2021-01-01", "2023-12-31")
c3_rollup <- generate_c3_rollup(c3)

## End(Not run)
```

---

generate_c3_table *Environmental surveillance KPIs by site*

Description

[Experimental]

Environmental surveillance KPIs summarized by site.

Usage

```
generate_c3_table(
  es_data,
  start_date,
  end_date,
  risk_category = NULL,
  lab_locs = NULL,
  risk_table = NULL
)
```

Arguments

es_data	tibble Environmental surveillance data.
start_date	str Start date of the analysis in YYYY-MM-DD format.
end_date	str End date of the analysis in YYYY-MM-DD format.
risk_category	str Risk category or a list of categories. Defaults to NULL. Valid values are: "LOW, LOW (WATCHLIST), MEDIUM, HIGH.
lab_locs	tibble Summary of the sequencing capacities of labs. Output of get_lab_locs(). Defaults to NULL, which will download the information directly from EDAV. .
risk_table	tibble Priority level of each country. Defaults to NULL, which will download the information directly from EDAV.

Value

tibble A summary table of environmental surveillance KPIs.

Examples

```
## Not run:
raw_data <- get_all_polio_data(attach.spatial.data = FALSE)
c3 <- generate_c3_table(raw_data$es, "2021-01-01", "2023-12-31")

## End(Not run)
```

---

generate_c4_table *Laboratory surveillance KPIs*

Description

[Experimental]

Summarizes the timeliness of samples as it arrives in the lab and to sequencing results. Samples may come from both AFP and ES samples.

Usage

```
generate_c4_table(lab_data, afp_data, start_date, end_date)
```

Arguments

lab_data	tibble Lab data containing information of ES or AFP samples.
afp_data	tibble AFP surveillance data.
start_date	str Start date of the analysis in YYYY-MM-DD format.
end_date	str End date of the analysis in YYYY-MM-DD format.

Value

list A summary of timeliness KPIs for lab data.

Examples

```
## Not run:
raw_data <- get_all_polio_data(attach.spatial.data = FALSE)
lab_data <- readr::read_csv("C:/Users/ABC1/Desktop/lab_data.csv")
c4 <- generate_c4_table(lab_data, raw_data$afp, "2021-01-01", "2024-12-31")

## End(Not run)
```

`generate_case_num_dose_g` *Immunization rates per year*

Description

Generates a stacked percent bar plot displaying immunization rates per year for the country. Note that this function only graphs immunization rates for children aged 6-59 months that have the classification of NPAFP.

Usage

```
generate_case_num_dose_g(
  ctry.data,
  start_date,
  end_date,
  output_path = Sys.getenv("DR_FIGURE_PATH")
)
```

Arguments

`ctry.data`	`list` A large list containing polio data of country. This is the output of `extract_country_data()` or `init_dr()`. Note that `ctry_data` needs to be cleaned via `clean_ctry_data()` prior to running the function.
`start_date`	`str` Start date of analysis.
`end_date`	`str` End date of analysis.
`output_path`	`str` Local path of where to save the figure to.

Value

`ggplot` A percent bar plot displaying immunization rates per year by immunization status.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
ctry.data <- clean_ctry_data(ctry.data)
generate_case_num_dose_g(ctry.data, "2021-01-01", "2023-12-31")

## End(Not run)
```

`generate_ctry_timeliness_graph` *Timeliness intervals of samples at the country level*

Description

A stacked horizontal bar graph for timeliness intervals of samples at the country level. To get the full intervals from field to lab, the lab data needs to be attached. Otherwise, only the timeliness intervals from the field up to when it was sent to lab will be displayed.

Usage

```
generate_ctry_timeliness_graph(
  int.data,
  output_path = Sys.getenv("DR_FIGURE_PATH"),
  afp.year.lab = lifecycle::deprecated()
)
```

Arguments

`int.data`	`tibble` Summary table with timeliness intervals at the country level.
`output_path`	`str` Path where to output the figure.
`afp.year.lab`	`tibble` **[Deprecated]** Deprecated since it is not used anymore.

Value

`ggplot` Plot of timeliness intervals at the country level.

See Also

`generate_int_data()`

Examples

```
## Not run:
# Attaching lab data
lab_path <- "C:/Users/ABC1/Desktop/algeria_lab_data.csv"
ctry.data <- init_dr("algeria", lab_data_path = lab_path)
lab.timeliness.ctry <- generate_lab_timeliness(ctry.data$lab.data, "ctry", start_date, end_date)
int.data.ctry <- generate_int_data(ctry.data, start_date, end_date,
  spatial.scale = "ctry",
  lab.timeliness.ctry
)
generate_ctry_timeliness_graph(int.data.ctry)

## End(Not run)
```

`generate_dist_pop_map` *Map district U15 populations*

Description

Generates a map of U15 district populations, with population centers and roads.

Usage

```
generate_dist_pop_map(
  ctry.data,
  ctry.shape,
  prov.shape,
  dist.shape,
  end_date,
  output_path = Sys.getenv("DR_FIGURE_PATH"),
  caption_size = 11
)
```

Arguments

`ctry.data`	`list` Large list of polio country data. This is the output of either `extract_country_data()` or `init_dr()`.
`ctry.shape`	`sf` Shapefile of country in long format.
`prov.shape`	`sf` Shapefile of province in long format.
`dist.shape`	`sf` Shapefile of district in long format.
`end_date`	`str` End date of the analysis.
`output_path`	`str` Local path of where to save the figure.
`caption_size`	`numeric` Size of the caption. Default is 11.

Value

`ggplot` A map of district level populations and population centers.

See Also

`load_clean_ctry_sp()`, `load_clean_prov_sp()`, `load_clean_dist_sp()`

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
ctry.shape <- load_clean_ctry_sp(ctry_name = "ALGERIA", type = "long")
prov.shape <- load_clean_prov_sp(ctry_name = "ALGERIA", type = "long")
dist.shape <- load_clean_dist_sp(ctry_name = "ALGERIA", type = "long")
generate_pop_map(ctry.data, ctry.shape, prov.shape, dist.shape, "2023-12-31")

## End(Not run)
```

`generate_dr_ppt` *Generate the desk review slide deck*

Description

The original function to build the desk review PowerPoint. This function has been superseded by `generate_dr_ppt2()`. The function outputs images to the PowerPoint directly from objects, unlike `generate_dr_ppt2()` which uses images saved in a folder.

Usage

```
generate_dr_ppt(
  ppt_template_path,
  ctry.data,
  start_date,
  end_date,
  pop.map,
  pop.map.prov,
  afp.case.map,
  afp.epi.curve,
  surv.ind.tab,
  afp.dets.prov.year,
  pop.tab,
  npafp.map,
  npafp.map.dist,
  stool.ad.maps,
  stool.ad.maps.dist,
  inad.tab.flex,
  tab.60d,
  case.num.dose.g,
  timely_nation,
  timely_prov,
  mapt_all,
  es.site.det,
  es.det.map,
  es.timely,
  es.table,
  country = Sys.getenv("DR_COUNTRY"),
  ppt_output_path = Sys.getenv("DR_POWERPOINT_PATH")
)
```

Arguments

`ppt_template_path`	`str` Sath to the PowerPoint template.
`ctry.data`	`list` List containing polio data for a country. Either the output of `extract_country_data()` or `init_dr()`.
`start_date`	`str` Start date of desk review.
`end_date`	`str` End date of desk review.
`pop.map`	`ggplot` Country pop map.

pop.map.prov ggplot Prov pop map.

afp.case.map ggplot Map of afp cases.

afp.epi.curve ggplot AFP epicurve.

surv.ind.tab flextable Surveillance indicator table.

afp.dets.prov.year ggplot AFP detections for province.

pop.tab flextable Table of population.

npafp.map ggplot NPAFP map for country level.

npafp.map.dist ggplot NPAFP map for district level.

stool.ad.maps ggplot Stool adequacy maps at province.

stool.ad.maps.dist ggplot Stool adequacy maps at district.

inad.tab.flex flextable Inadequate table.

tab.60d flextable 60-day follow-up table.

case.num.dose.g ggplot Immunization rates per year.

timely_nation ggplot Timeliness at country level.

timely_prov ggplot Timeliness at province level.

mapt_all ggplot Map with all indicators.

es.site.det ggplot ES site viral detection.

es.det.map ggplot ES site detection maps.

es.timely ggplot ES timeliness.

es.table flextable ES table.

country str Name of the country.

ppt_output_path str Path where the PowerPoint should be outputted.

Value

None.

Examples

```
## Not run:
# Assume all figures and tables are assigned to the appropriate variable.
template_path <- "C:/Users/ABC1/Desktop/deskreview_template.pptx"
generate_dr_ppt(
  template_path, ctry.data, start_date, end_date, pop.map,
  pop.map, pop.map.prov, afp.case.map, afp.epi.curve,
  surv.ind.tab, afp.dets.prov.year, pop.tab, npafp.map,
  npafp.map.dist, stool.ad.maps, stool.ad.maps.dist,
  inad.tab.flex, tab.60d, case.num.dose.g,
  timely_nation, timely_prov,
  mapt_all, es.site.det, es.det.map, es.timely,
  es.table
)

## End(Not run)
```

`generate_dr_ppt2` *Generate the desk review slide deck from the figures folder*

Description

Generating the PowerPoint from the figures folder is generally faster and allows figures to remain consistent. Tables remain as PowerPoint tables.

Usage

```
generate_dr_ppt2(
  ctry.data,
  start_date,
  end_date,
  surv.ind.tab,
  inad.tab.flex,
  tab.60d,
  pop.tab,
  es.table,
  ppt_template_path = NULL,
  fig.path = Sys.getenv("DR_FIGURE_PATH"),
  country = Sys.getenv("DR_COUNTRY"),
  ppt_output_path = Sys.getenv("DR_POWERPOINT_PATH")
)
```

Arguments

`ctry.data`	`list` Country polio data. Either the output of `extract_country_data()` or `init_dr()`.
`start_date`	`str` Start date of desk review.
`end_date`	`str` End date of desk review.
`surv.ind.tab`	`flextable` Surveillance indicator table
`inad.tab.flex`	`flextable` Inadequates table.
`tab.60d`	`flextable` 60-day follow-up table.
`pop.tab`	`flextable` Population table.
`es.table`	`flextable` ES table.
`ppt_template_path`	`str` Path to the PowerPoint template.
`fig.path`	`str` File path to the figures folder.
`country`	`str` Name of the country.
`ppt_output_path`	`str` Path where the PowerPoint should be outputted.

Value

None.

Examples

```
## Not run:
# Assume all figures and tables are assigned to the appropriate variable.
ppt_template <- "C:/Users/ABC1/Desktop/deskreview_template.pptx"
generate_dr_ppt2(ctry.data, start_date, end_date,
  surv.ind.tab, inad.tab.flex, tab.60d, es.table,
  ppt_template_path = ppt_template
)

## End(Not run)
```

---

generate_es_det_map *ES detection map*

Description

Generates a map showing the detection rate of each ES sites on a rolling period as defined by the start and end dates of the analysis.

Usage

```
generate_es_det_map(
  es.data,
  ctry.shape,
  prov.shape,
  es_start_date = (lubridate::as_date(es_end_date) - lubridate::years(1)),
  es_end_date = end_date,
  output_path = Sys.getenv("DR_FIGURE_PATH"),
  es.data.long = lifecycle::badge("deprecated")
)
```

Arguments

es.data	tibble ES data for a country. This is ctry.data$es, which is part of the outputs of extract_country_data() and init_dr().
ctry.shape	sf Country shapefile in long format.
prov.shape	sf Province shapefile in long format.
es_start_date	str Start date of analysis. Default is one year from the end date.
es_end_date	str End date of analysis.
output_path	str Local path where to save the figure to.
es.data.long	**[Deprecated]** tibble Please pass the output of clean_es_data() into es.data instead. This paramater is not being used in the function.

Value

ggplot Map of EV detection rates for the environmental surveillance sites.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
es.data.long <- generate_es_data_long(ctry.data$es)
ctry.shape <- load_clean_ctry_sp(ctry_name = "ALGERIA", type = "long")
prov.shape <- load_clean_prov_sp(ctry_name = "ALGERIA", type = "long")
generate_es_det_map(ctry.data$es, ctry.shape, prov.shape,
  es_end_date = "2023-01-01"
)

## End(Not run)
```

---

generate_es_site_det *Virus detection in ES sites*

Description

Generates a dot plot for viral detections across ES sites, with SIA dates overlaid.

Usage

```
generate_es_site_det(
  sia.data,
  es.data,
  es_start_date = (lubridate::as_date(es_end_date) - lubridate::years(1)),
  es_end_date = end_date,
  output_path = Sys.getenv("DR_FIGURE_PATH"),
  vaccine_types = NULL,
  detection_types = NULL,
  ctry.data = lifecycle::deprecated(),
  es.data.long = lifecycle::deprecated()
)
```

Arguments

`sia.data`	`tibble` SIA surveillance data.
`es.data`	Environmental surveillance data, cleaned using `clean_es_data()` or a cleaned `ctry.data$es`.
`es_start_date`	`str` Start date of analysis. By default, it is one year from the end date.
`es_end_date`	`str` End date of analysis.
`output_path`	`str` Local path to output the figure to.
`vaccine_types`	`list` A named list with colors assigned names corresponding to vaccine types. By default, it will use a prefilled list inside the function. However, the function will alert for missing vaccine types and the user must pass another list appended by that vaccine type.
`detection_types`	`list` A named list with colors assigned names corresponding to viral detection type. By default, it will use a prefilled list inside the function. However, the function will alert for missing detection types and the user must pass another list appended by that vaccine type.

ctry.data	**[Deprecated]** Please pass the SIA data directly to sia.data instead of a list containing it.
es.data.long	**[Deprecated]** Please pass cleaned ES data instead.

Value

ggplot A dot plot of viral detections per ES sites and SIA campaigns.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
es.data <- clean_es_data(ctry.data$es)
generate_es_site_det(ctry.data, es.data)

## End(Not run)
```

---

generate_es_tab *ES surveillance sites summary table*

Description

Generates a summary table on the performance of surveillance sites over a rolling basis as indicated by the start and end dates. Includes information on the EV detection rate, number of samples collected, percentage of samples with good condition, and percentage of samples meeting the timeliness target of arriving to lab within 3 days.

Usage

```
generate_es_tab(
  es.data,
  es_start_date = (lubridate::as_date(es_end_date) - lubridate::years(1)),
  es_end_date = end_date
)
```

Arguments

es.data	tibble ES data. This is ctry.data$es, which is part of the output of either extract_country_data() or init_dr(). Ensure that the ctry.data object has been cleaned with clean_ctry_data() first. Otherwise, there will be an error.
es_start_date	str Start date of analysis. Defaults to a year before the end date.
es_end_date	str End date of analysis.

Value

flextable Summary table of ES surveillance site performance.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
ctry.data <- clean_ctry_data(ctry.data)
generate_es_tab(ctry.data$es, es_end_date = "2023-12-31")

## End(Not run)
```

`generate_es_timely` *ES timeliness scatterplot*

Description

Generates a scatterplot of the time it takes for each environmental samples to arrive in lab.

Usage

```
generate_es_timely(
  es.data,
  es_start_date = (lubridate::as_date(es_end_date) - lubridate::years(1)),
  es_end_date = end_date,
  output_path = Sys.getenv("DR_FIGURE_PATH")
)
```

Arguments

`es.data`	`tibble` ES data.
`es_start_date`	`str` Start date of analysis. By default, this is one year from the end date.
`es_end_date`	`str` End date of analysis.
`output_path`	`str` Local path for where to save the figure to.

Value

`ggplot` A scatterplot for timeliness of ES samples.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
generate_es_timely(ctry.data$es)

## End(Not run)
```

generate_inad_tab *Issues with stool adequacy at the country level*

Description

Generates a summary table at the country level highlighting issues around stool adequacy.

Usage

```
generate_inad_tab(ctry.data, cstool, start_date, end_date)
```

Arguments

ctry.data	list large list containing polio data for a country. This is the output of extract_country_data() or init_dr().
cstool	tibble Stool adequacy at the country level. This is the output of f.stool.ad.01().
start_date	str Start date of analysis.
end_date	str End date of analysis.

Value

flextable Summary table containing stool adequacy issues at the country level.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
start_date <- "2021-01-01"
end_date <- "2023-12-31"
cstool <- f.stool.ad.01(
  afp.data = ctry.data$afp.all.2,
  admin.data = ctry.data$ctry.pop,
  start.date = start_date,
  end.date = end_date,
  spatial.scale = "ctry",
  missing = "good",
  bad.data = "inadequate",
  rolling = F,
  sp_continuity_validation = F
)
generate_inad_tab(ctry.data, cstool, start_date, end_date)

## End(Not run)
```

`generate_int_data` *Generate a summary table for sample timeliness intervals*

Description

The summary table will output timeliness intervals of samples from collection to lab testing. Lab timeliness will only be calculated if the lab data is attached. Otherwise, by default, the function will return only the timeliness intervals up to when the samples were sent to lab.

Usage

```
generate_int_data(
  afp_data,
  pop_data,
  start_date,
  end_date,
  spatial_scale,
  lab_data_summary = NULL,
  ctry.data = lifecycle::deprecated(),
  spatial.scale = lifecycle::deprecated(),
  lab.data = lifecycle::deprecated()
)
```

Arguments

`afp_data`	`tibble` AFP dataset.
`pop_data`	`tibble` Population dataset that matches the spatial scale.
`start_date`	`str` Start date of analysis.
`end_date`	`str` End date of analysis.
`spatial_scale`	`str` Scale to summarize to. Valid values are: `"ctry"` or `"prov"`. `"dist"` not available currently.
`lab_data_summary`	`tibble` Summarized lab data, if available. This parameter will calculate timeliness intervals in the lab. Otherwise, only the field component will be presented. This is the output of `generate_lab_timeliness()`.
`ctry.data`	`list` **[Deprecated]**
`spatial.scale`	`str` **[Deprecated]** Renamed in favor of `spatial_scale`.
`lab.data`	`tibble` **[Deprecated]** Renamed in favor of `lab_data_summary`. Passing ctry.data has been deprecated in favor of independently assigning the AFP dataset to afp.data and the population dataset to pop.data. This allows the function to run either on raw.data or ctry.data.

Value

`tibble` A table summarizing median days for different timeliness intervals.

See Also

`clean_ctry_data()`

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
ctry.data <- extract_country_data("algeria", raw.data)
# lab data not attached
int.data <- generate_int_data(
  raw.data$afp, raw.data$ctry.pop,
  "2021-01-01", "2023-12-31", "ctry"
)

# If lab data is available. Assume ctry.data is loaded.
lab_path <- "C:/Users/ABC1/Desktop/algeria_lab.csv"
lab.data <- readr::read_csv(lab_path)
lab.data.summary <- generate_lab_timeliness(
  lab.data, "ctry",
  "2021-01-01", "2023-12-31"
)
int.data <- generate_int_data(
  ctry.data$afp.all.2, ctry.data$ctry.pop,
  "2021-01-01", "2023-12-31", "ctry",
  lab.data.summary
)

## End(Not run)
```

---

generate_iss_barplot *Visits to health clinics per year*

Description

Generates a bar plot showing the number of visits to health clinics per year using the ISS/eSURV data.

Usage

```
generate_iss_barplot(
  iss.data = NULL,
  start_date,
  end_date,
  output_path = Sys.getenv("DR_FIGURE_PATH")
)
```

Arguments

iss.data	tibble ISS/eSURV data that has been cleaned via clean_iss_data().
start_date	str Start date of the analysis.
end_date	str End date of the analysis.
output_path	str Local path where the figure is saved to.

Value

ggplot Bar plot of health clinic visits.

Examples

```
## Not run:
iss_path <- "C:/Users/ABC1/Desktop/iss_data.csv"
ctry.data <- init_dr("algeria", iss_data_path = iss_path)
ctry.data$iss.data <- clean_iss_data(ctry.data)
generate_iss_barplot(ctry.data$iss.data)

## End(Not run)
```

`generate_iss_map` *Map of high priority health facilities*

Description

Generates a map of high priority health facilities across years based on ISS/eSURV data.

Usage

```
generate_iss_map(
  iss.data,
  ctry.shape,
  prov.shape,
  start_date,
  end_date,
  output_path = Sys.getenv("DR_FIGURE_PATH")
)
```

Arguments

`iss.data`	`tibble` ISS/eSurv data. Ensure that the `iss.data` is part of `ctry.data` and has been cleaned by `clean_iss_data()`.
`ctry.shape`	`sf` Country shapefile in long format.
`prov.shape`	`sf` Province shapefile in long format.
`start_date`	`str` Start date of analysis.
`end_date`	`str` End date of analysis.
`output_path`	`str` Local path where to save the figure to.

Value

`ggplot` Map of where the high priority health facilities are across years.

Examples

```
## Not run:
iss_path <- "C:/Users/ABC1/Desktop/iss_data.csv"
ctry.data <- init_dr("algeria", iss_data_path = iss_path)
ctry.data$iss.data <- clean_iss_data(ctry.data)
ctry.shape <- load_clean_ctry_sp(ctry_name = "ALGERIA", type = "long")
prov.shape <- load_clean_prov_sp(ctry_name = "ALGERIA", type = "long")
generate_iss_map(
```

```
  ctry.data$iss.data, ctry.shape, prov.shape,
  "2021-01-01", "2023-12-31"
)

## End(Not run)
```

`generate_kpi_barchart` *Generate KPI indicator bar charts*

Description

[Experimental]

A generalized function to create bar charts using the KPI tables.

Usage

```
generate_kpi_barchart(df, indicator, target, label, faceting, y.axis.title)
```

Arguments

`df`	`tibble` A KPI table, namely C1-C4.
`indicator`	`str` Name of the indicator within the KPI table.
`target`	`num` A numeric target.
`label`	`str` Name of the column containing labels.
`faceting`	`ggplot2` A ggplot2 faceting object. Either using `ggplot2::facet_grid()` or `ggplot2::facet_wrap()`.
`y.axis.title`	`Str` Title of the y axis.

Value

`ggplot2` A bar chart.

`generate_kpi_evdetect_bar`
Generate KPI EV rate bar chart summary

Description

[Experimental]

Generates a bar chart highlighting percentage of priority countries achieving 80% of ES sites meeting sensitivity threshold of at least 50% samples positive for enterovirus. Sites included are sites with at least 10 collections in the last 12 months.

Usage

```
generate_kpi_evdetect_bar(
  c1,
  afp_data,
  output_path = Sys.getenv("KPI_FIGURES"),
  who_region = NULL
)
```

Arguments

c1	tibble Output of generate_c1_table().
afp_data	tibble AFP dataset. List item of the output of get_all_polio_data().
output_path	str Folder location to output the image to.
who_region	str A WHO region or a list of regions. Valid values are: • "AFRO": African Region • "AMRO": Region of the Americas • "EMRO": Eastern Mediterranean Region • "EURO": European Region • "SEARO":South-East Asia Region • "WPRO":Western Pacific Region

Value

ggplot2 A barplot.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
c1 <- generate_c1_table(raw_data, "2021-01-01", "2023-12-31", rolling = TRUE)
plot <- generate_kpi_evdetect_bar(c1, raw_data$afp)

## End(Not run)
```

---

generate_kpi_ev_map *EV detection rate map*

Description

[Experimental]

Generates a map of EV detection rates for each environmental surveillance site.

Usage

```
generate_kpi_ev_map(
  c3,
  .year_label,
  who_region = NULL,
  output_path = Sys.getenv("KPI_FIGURES"),
  dot_size = 2.3,
  ctry_sf = NULL
)
```

Arguments

c3	tibble Output of generate_c3_table(). This must be summarized at the site level (i.e., use the default .group_by) param of the generate_c3_table().
.year_label	str Roll up year (i.e., "Year 1", "Year 2", ...).
who_region	str Name of the region or a list of regions.
output_path	str Where to output the figure to. Defaults to the figure path assigned after running init_kpi().
dot_size	num Point size.
ctry_sf	sf Country shapefile in long format. Output of load_clean_ctry_sp() with type = "long". Defaults to NULL, which will download the required country shapefile when the function is ran.

Value

ggplot A map showing EV detection rate by site.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
c3 <- generate_c3_table(raw_data$es, "2021-01-01", "2023-12-31")
map <- generate_kpi_ev_map(c3, "Year 1", "AFRO", getwd())

## End(Not run)
```

---

generate_kpi_map *Generate KPI maps*

Description

[Experimental]

Generalized function to build KPI maps.

Usage

```
generate_kpi_map(
  c2,
  who_region,
  indicator,
  .year_label,
  risk_category,
  color_scheme,
  legend_title,
  .ctry_sf,
  .dist_sf
)
```

Arguments

c2	tibble Output of generate_c2_table().
who_region	str A WHO region or a list of regions. Valid values are: • "AFRO": African Region • "AMRO": Region of the Americas • "EMRO": Eastern Mediterranean Region • "EURO": European Region • "SEARO":South-East Asia Region • "WPRO":Western Pacific Region
indicator	quoted var Indicator variable.
.year_label	str Year of the rollup.
risk_category	str A string or a list of strings with priority categories. Valid values are: "LOW", "LOW (WATCHLIST)", "MEDIUM", "HIGH".
color_scheme	list Named list with color mappings.
legend_title	str Title of the legend.
.ctry_sf	sf Country shapefile.
.dist_sf	sf District shapefile.

Value

ggplot A ggplot object.

---

generate_kpi_npafp_bar

Generate KPI NPAFP bar chart summary

Description

[Experimental]

Generates a bar chart highlighting percentage of priority countries with 80% of 100,000+ under 15y pop districts achieving npAFP rate of >= 2/100,000.

Usage

```
generate_kpi_npafp_bar(c1, afp_data, output_path = Sys.getenv("KPI_FIGURES"))
```

Arguments

c1	tibble Output of generate_c1_table().
afp_data	tibble AFP dataset. List item of the output of get_all_polio_data().
output_path	str Folder location to output the image to.

Value

ggplot2 A barplot.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
c1 <- generate_c1_table(raw_data, "2021-01-01", "2023-12-31", rolling = TRUE)
plot <- generate_kpi_npafp_bar(c1, raw_data$afp)

## End(Not run)
```

`generate_kpi_npafp_map`

Generate district level NPAFP maps by region

Description

[Experimental]

Generates a map of district NPAFP rates.

Usage

```
generate_kpi_npafp_map(
  c2,
  year_label,
  who_region = NULL,
  risk_category = NULL,
  output_path = Sys.getenv("KPI_FIGURES"),
  ctry_sf = NULL,
  dist_sf = NULL
)
```

Arguments

`c2`	`tibble` Output of `generate_c2_table()`.
`year_label`	`str` Roll up year (i.e., `"Year 1"`, `"Year 2"`, `...`).
`who_region`	`str` A WHO region or a list of regions. Valid values are: • `"AFRO"`: African Region • `"AMRO"`: Region of the Americas • `"EMRO"`: Eastern Mediterranean Region • `"EURO"`: European Region • `"SEARO"`:South-East Asia Region • `"WPRO"`:Western Pacific Region
`risk_category`	`str` A string or a list of strings with priority categories. Valid values are: "LOW", "LOW (WATCHLIST)", "MEDIUM", "HIGH".
`output_path`	`str` Where to output the figure to. Defaults to the path initialized when `init_kpi()` was ran.
`ctry_sf`	`sf` Country shapefile in long format. Output of `load_clean_ctry_sp()` with `type = "long"`. Defaults to `NULL`, which will download the required country shapefile when the function is ran.
`dist_sf`	`sf` District shapefile in long format. Output of `load_clean_dist_sp()` with `type = "long"`. Defaults to `NULL`, which will download the required district shapefile when the function is ran.

Value

ggplot A district NPAFP map.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
c2 <- generate_c2_table(raw_data$afp, raw_data$dist, "2021-01-01",
"2023-12-31", c("ctry", "dist", "adm2guid", "year"))
map <- generate_kpi_npafp_map(c2, "Year 1", "AFRO", output_path = getwd())

## End(Not run)
```

---

generate_kpi_stoolad_bar

Generate KPI Stool adequacy bar chart summary

Description

[Experimental]

Generates a bar chart highlighting percentage of priority countries achieving stool adequacy targets.

Usage

```
generate_kpi_stoolad_bar(c1, afp_data, output_path = Sys.getenv("KPI_FIGURES"))
```

Arguments

c1	tibble Output of generate_c1_table().
afp_data	tibble AFP dataset. List item of the output of get_all_polio_data().
output_path	str Folder location to output the image to.

Value

ggplot2 A barplot.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
c1 <- generate_c1_table(raw_data, "2021-01-01", "2023-12-31", rolling = TRUE)
plot <- generate_kpi_stoolad_bar(c1, raw_data$afp)

## End(Not run)
```

`generate_kpi_stool_map` *Generate district level stool adequacy maps*

Description

[Experimental]

Generates a map of district level stool adequacy maps

Usage

```
generate_kpi_stool_map(
  c2,
  year_label,
  who_region = NULL,
  risk_category = NULL,
  output_path = Sys.getenv("KPI_FIGURES"),
  ctry_sf = NULL,
  dist_sf = NULL
)
```

Arguments

`c2`	`tibble` Output of `generate_c2_table()`.
`year_label`	`str` Roll up year (i.e., `"Year 1"`, `"Year 2"`, `...`).
`who_region`	`str` A WHO region or a list of regions. Valid values are: • `"AFRO"`: African Region • `"AMRO"`: Region of the Americas • `"EMRO"`: Eastern Mediterranean Region • `"EURO"`: European Region • `"SEARO"`:South-East Asia Region • `"WPRO"`:Western Pacific Region
`risk_category`	`str` A string or a list of strings with priority categories. Valid values are: "LOW", "LOW (WATCHLIST)", "MEDIUM", "HIGH".
`output_path`	`str` Where to output the figure to. Defaults to the path initialized when `init_kpi()` was ran.
`ctry_sf`	`sf` Country shapefile in long format. Output of `load_clean_ctry_sp()` with `type = "long"`. Defaults to `NULL`, which will download the required country shapefile when the function is ran.
`dist_sf`	`sf` District shapefile in long format. Output of `load_clean_dist_sp()` with `type = "long"`. Defaults to `NULL`, which will download the required district shapefile when the function is ran.

Value

`ggplot` A stool adequacy map.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
c2 <- generate_c2_table(raw_data$afp, raw_data$dist, "2021-01-01",
"2023-12-31", c("ctry", "dist", "adm2guid", "year"))
map <- generate_kpi_stool_map(c2, "Year 1", "AFRO", output_path = getwd())

## End(Not run)
```

---

generate_kpi_tile *Generate tile plots for indicators*

Description

Generates tile plots for indicators present in c1-c4, showing changes over multiple rolling periods.

Usage

```
generate_kpi_tile(
  c_table,
  priority_category = "HIGH",
  output_path = Sys.getenv("KPI_FIGURES")
)
```

Arguments

c_table	tibble Either C1, C2, C3, C4
priority_category	str A string or a list of priority category. Valid values are: "LOW", "LOW (WATCHLIST)", "MEDIUM", "HIGH".
output_path	str Where to output the figure to.

Details

The function automatically detects the geographic scale to present in each plot. If passing a table that is grouped lower than the country level, we recommend that only one country is present or a subset of districts are presented so that the output plot is legible.

Value

ggplot2 A tile plot for each indicator for each geography

Examples

```
## Not run:
raw_data <- get_all_polio_data()
c1 <- generate_c1_table(raw_data, "2021-01-01", "2023-12-31")
generate_kpi_tile(c1)

## End(Not run)
```

generate_kpi_violin *Generates KPI violin plot*

Description

[Experimental]

This function is generalized to produce violin plots used in the KPI code.

Usage

```
generate_kpi_violin(
  df,
  country.label,
  interval,
  priority_level,
  faceting,
  target,
  y.min = 0,
  y.max
)
```

Arguments

df	tibble Data to be used.
country.label	quoted var Country label.
interval	quoted var Interval to use.
priority_level	quoted var Priority level column.
faceting	ggplot::facet A faceting object.
target	num Numeric target.
y.min	num Minimum used in the y-axis.
y.max	num Maximum used in the y-axis.

Value

ggplot A plot object.

generate_lab_culture_violin *Timeliness of lab samples from collection to virus isolation results*

Description

[Experimental]

Shows the timeliness of lab culture results.

Usage

```
generate_lab_culture_violin(
  lab_data,
  afp_data,
  start_date,
  end_date,
  priority_level = c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW"),
  who_region = NULL,
  rolling = TRUE,
  output_path = Sys.getenv("KPI_FIGURES"),
  y_max = 60
)
```

Arguments

lab_data	tibble Global lab dataset.
afp_data	tibble AFP dataset.
start_date	str Start date of the analysis formatted as "YYYY-MM-DD".
end_date	str End date of the analysis formatted as "YYYY-MM-DD".
priority_level	list Priority levels to display. Defaults to c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW").
who_region	list Regions to display. Defaults to NULL, which shows all of the regions.
rolling	logical Using rolling periods or year-to-year? Defaults to TRUE.
output_path	str Where to output the figure to.
y_max	num Maximum value in the y-axis.

Value

ggplot A violin plot showing timeliness of lab culture.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
lab_data <- edav_io("read", file_loc = get_constant("CLEANED_LAB_DATA"))
generate_lab_culture_violin(lab_data, raw_data$afp,
                            "2021-01-01", "2023-12-31", getwd())

## End(Not run)
```

---

generate_lab_itdres_seqres_violin

Timeliness of final ITD results to sequencing results

Description

[Experimental]

Shows the timeliness from date of ITD results to sequencing results. The target is 7 days for cases not shipped for sequencing and 14 days for samples shipped for sequencing.

Usage

```
generate_lab_itdres_seqres_violin(
  lab_data,
  afp_data,
  start_date,
  end_date,
  priority_level = c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW"),
  who_region = NULL,
  rolling = TRUE,
  output_path = Sys.getenv("KPI_FIGURES"),
  y_max = 60
)
```

Arguments

lab_data	tibble Global lab dataset.
afp_data	tibble AFP dataset.
start_date	str Start date of the analysis formatted as "YYYY-MM-DD".
end_date	str End date of the analysis formatted as "YYYY-MM-DD".
priority_level	list Priority levels to display. Defaults to c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW").
who_region	list Regions to display. Defaults to NULL, which shows all of the regions.
rolling	logical Using rolling periods or year-to-year? Defaults to TRUE.
output_path	str Where to output the figure to.
y_max	num Maximum value in the y-axis.

Value

ggplot A violin plot showing the timeliness of sequencing results.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
lab_data <- edav_io("read", file_loc = get_constant("CLEANED_LAB_DATA"))
generate_lab_itdres_seqres_violin(lab_data, raw_data$afp,
                            "2021-01-01", "2023-12-31", getwd())

## End(Not run)
```

---

generate_lab_itd_violin

Timeliness of virus isolation to ITD results

Description

[Experimental]

Shows the timeliness of ITD results of specimens that require ITD. The target is 7 days or less.

Usage

```
generate_lab_itd_violin(
  lab_data,
  afp_data,
  start_date,
  end_date,
  priority_level = c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW"),
  who_region = NULL,
  rolling = TRUE,
  output_path = Sys.getenv("KPI_FIGURES"),
  y_max = 30
)
```

Arguments

lab_data	tibble Global lab dataset.
afp_data	tibble AFP dataset.
start_date	str Start date of the analysis formatted as "YYYY-MM-DD".
end_date	str End date of the analysis formatted as "YYYY-MM-DD".
priority_level	list Priority levels to display. Defaults to c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW").
who_region	list Regions to display. Defaults to NULL, which shows all of the regions.
rolling	logical Using rolling periods or year-to-year? Defaults to TRUE.
output_path	str Where to output the figure to.
y_max	num Maximum value in the y-axis.

Value

ggplot A violin plot showing timeliness of ITD results from lab culture.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
lab_data <- edav_io("read", file_loc = get_constant("CLEANED_LAB_DATA"))
generate_lab_itd_violin(lab_data, raw_data$afp,
                        "2021-01-01", "2023-12-31", getwd())

## End(Not run)
```

---

generate_lab_seqres_violin

Timeliness of arrival at sequencing lab to sequencing results

---

Description

[Experimental]

Shows the timeliness of arrival in the sequencing lab to sequencing results. The target is 7 days for cases shipped for sequencing.

Usage

```
generate_lab_seqres_violin(
  lab_data,
  afp_data,
  start_date,
  end_date,
  priority_level = c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW"),
  who_region = NULL,
  rolling = TRUE,
  output_path = Sys.getenv("KPI_FIGURES"),
  y_max = 60
)
```

Arguments

lab_data	tibble Global lab dataset.
afp_data	tibble AFP dataset.
start_date	str Start date of the analysis formatted as "YYYY-MM-DD".
end_date	str End date of the analysis formatted as "YYYY-MM-DD".
priority_level	list Priority levels to display. Defaults to c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW").
who_region	list Regions to display. Defaults to NULL, which shows all of the regions.
rolling	logical Using rolling periods or year-to-year? Defaults to TRUE.
output_path	str Where to output the figure to.
y_max	num Maximum value in the y-axis.

Value

ggplot A violin plot showing the timeliness of sequencing results.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
lab_data <- edav_io("read", file_loc = get_constant("CLEANED_LAB_DATA"))
generate_lab_seqres_violin(lab_data, raw_data$afp,
                           "2021-01-01", "2023-12-31", getwd())

## End(Not run)
```

---

generate_lab_seqship_violin

Timeliness of shipment to sequencing results

Description

[Experimental]

Shows the timeliness of shipment to sequencing results for specimens that require sequencing. The target is 7 days or less.

Usage

```
generate_lab_seqship_violin(
  lab_data,
  afp_data,
  start_date,
  end_date,
  priority_level = c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW"),
  who_region = NULL,
  rolling = TRUE,
  output_path = Sys.getenv("KPI_FIGURES"),
  y_max = 50
)
```

Arguments

lab_data	tibble Global lab dataset.
afp_data	tibble AFP dataset.
start_date	str Start date of the analysis formatted as "YYYY-MM-DD".
end_date	str End date of the analysis formatted as "YYYY-MM-DD".
priority_level	list Priority levels to display. Defaults to c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW").
who_region	list Regions to display. Defaults to NULL, which shows all of the regions.
rolling	logical Using rolling periods or year-to-year? Defaults to TRUE.
output_path	str Where to output the figure to.
y_max	num Maximum value in the y-axis.

Value

ggplot A violin plot showing the timeliness of shipment to sequencing results.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
lab_data <- edav_io("read", file_loc = get_constant("CLEANED_LAB_DATA"))
generate_lab_seqship_violin(lab_data, raw_data$afp,
                            "2021-01-01", "2023-12-31", getwd())

## End(Not run)
```

---

generate_lab_timeliness

Summary of lab sample timeliness

Description

Generates a summary of the timeliness of samples for specific intervals.

Usage

```
generate_lab_timeliness(
  lab_data,
  spatial.scale,
  start_date,
  end_date,
  start.date = lifecycle::deprecated(),
  end.date = lifecycle::deprecated()
)
```

Arguments

lab_data	tibble Lab data. Ensure that this lab data is cleaned using clean_lab_data() before running the function.
spatial.scale	str Spatial scale to analyze the data. Valid values are "ctry", "prov", "dist".
start_date	str Start date of analysis.
end_date	str End date of analysis.
start.date	str **[Deprecated]** renamed in favor of start_date.
end.date	str **[Deprecated]** renamed in favor of end_date.

Value

tibble A table with timeliness data summary.

Examples

```
## Not run:
lab_path <- "C:/Users/XRG9/lab_data_who.csv"
ctry.data <- init_dr("algeria", lab_data_path = lab_path)
ctry.data$lab_data <- clean_lab_data(ctry.data, "2021-01-01", "2023-12-31")
lab.timeliness.ctry <- generate_lab_timeliness(ctry.data$lab_data, "ctry", start_date, end_date)

## End(Not run)
```

---

generate_npafp_maps *Maps of NPAFP rates by province and year*

Description

Generates a map of NPAFP rates for each province per year.

Usage

```
generate_npafp_maps(
  prov.extract,
  ctry.shape,
  prov.shape,
  start_date,
  end_date,
  output_path = Sys.getenv("DR_FIGURE_PATH"),
  caption_size = 2
)
```

Arguments

prov.extract	tibble Province NPAFP rate table. This is the output of f.npafp.rate.01() calculated at the province level.
ctry.shape	sf Country shape in long format.
prov.shape	sf Province shape in long format.
start_date	str Start date of analysis.
end_date	str End date of analysis.
output_path	str Local path where the figure is saved to.
caption_size	numeric Size of the caption. Default is 2.

Value

ggplot Map of NPAFP rates by province.

See Also

load_clean_ctry_sp(), load_clean_prov_sp()

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
prov.extract <- f.npafp.rate.01(
  afp.data = ctry.data$afp.all.2,
  pop.data = ctry.data$prov.pop,
  start.date = start_date,
  end.date = end_date,
  spatial.scale = "prov",
  pending = T,
  rolling = F,
  sp_continuity_validation = F
)
ctry.shape <- load_clean_ctry_sp(ctry_name = "ALGERIA", type = "long")
prov.shape <- load_clean_prov_sp(ctry_name = "ALGERIA", type = "long")
generate_npafp_maps(prov.extract, ctry.shape, prov.shape, "2021-01-01", "2023-12-31")

## End(Not run)
```

---

generate_npafp_maps_dist

Maps of NPAFP rates by district and year

Description

Generates maps of the NPAFP rates for each district per year.

Usage

```
generate_npafp_maps_dist(
  dist.extract,
  ctry.shape,
  prov.shape,
  dist.shape,
  start_date,
  end_date,
  output_path = Sys.getenv("DR_FIGURE_PATH"),
  caption_size = 2
)
```

Arguments

dist.extract	tibble Province NPAFP rate table. This is the output of f.npafp.rate.01() calculated at the province level.
ctry.shape	sf Country shapefile in long format.
prov.shape	sf Province shapefile in long format.
dist.shape	sf District shapefile in long format.
start_date	str Start date of analysis.
end_date	str End date of analysis.
output_path	str Local path Where the figure is saved to.
caption_size	numeric Size of the caption. Default is 2.

Value

ggplot A map of districts with their NPAFP rates.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
dist.extract <- f.npafp.rate.01(
  afp.data = ctry.data$afp.all.2,
  pop.data = ctry.data$prov.pop,
  start.date = start_date,
  end.date = end_date,
  spatial.scale = "dist",
  pending = T,
  rolling = F,
  sp_continuity_validation = F
)
ctry.shape <- load_clean_ctry_sp(ctry_name = "ALGERIA", type = "long")
prov.shape <- load_clean_prov_sp(ctry_name = "ALGERIA", type = "long")
dist.shape <- load_clean_dist_sp(ctry_name = "ALGERIA", type = "long")
generate_npafp_maps_dist(
  dist.extract, ctry.shape, prov.shape, dist.shape,
  "2021-01-01", "2023-12-31"
)

## End(Not run)
```

generate_pop_map *Country map with province populations*

Description

The map displays the U15 population for each province for a country.

Usage

```
generate_pop_map(
  ctry.data,
  ctry.shape,
  prov.shape,
  end_date,
  output_path = Sys.getenv("DR_FIGURE_PATH"),
  caption_size = 11
)
```

Arguments

ctry.data	list Large list containing country polio data. This is the output of extract_country_data() or init_dr().
ctry.shape	sf Country shape file in long format.
prov.shape	sf Province shape file in long format.
end_date	str End date of the analysis.
output_path	str Local path where to save the figure.
caption_size	numeric Size of the caption. Default is 11.

Value

ggplot A map of U15 province populations and population centers.

See Also

load_clean_ctry_sp(), load_clean_prov_sp()

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
ctry.shape <- load_clean_ctry_sp(ctry_name = "ALGERIA", type = "long")
prov.shape <- load_clean_prov_sp(ctry_name = "ALGERIA", type = "long")
generate_pop_map(ctry.data, ctry.shape, prov.shape, "2023-12-31")

## End(Not run)
```

generate_pop_tab *Summary table of indicators at the province level*

Description

Generates a table summarizing both NPAFP and stool adequacy rates at the province level and by year.

Usage

```
generate_pop_tab(
  pnpafp,
  pstool,
  start_date,
  end_date,
  prov.case.ind = lifecycle::deprecated()
)
```

Arguments

pnpafp	tibble NPAFP table. Output of f.npafp.rate.01() at the province level.
pstool	tibble Stool adequacy at province level. Output of f.stool.ad.01() at the province level.
start_date	str Start date of analysis.
end_date	str End date of analysis.
prov.case.ind	tibble **[Deprecated]** Deprecated in favor of the more informative pnpafp param name.

Value

flextable Summary table of province NPAFP and stool adequacy rates per year.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
start_date <- "2021-01-01"
end_date <- "2023-12-31"
prov.extract <- f.npafp.rate.01(
  afp.data = ctry.data$afp.all.2,
  pop.data = ctry.data$prov.pop,
  start.date = start_date,
  end.date = end_date,
  spatial.scale = "prov",
  pending = T,
  rolling = F,
  sp_continuity_validation = F
)
pstool <- f.stool.ad.01(
  afp.data = ctry.data$afp.all.2,
  admin.data = ctry.data$prov.pop,
  start.date = start_date,
```

```
  end.date = end_date,
  spatial.scale = "prov",
  missing = "good",
  bad.data = "inadequate",
  rolling = F,
  sp_continuity_validation = F
)
generate_pop_tab(prov.extract, pstool, start_date, end_date)

## End(Not run)
```

`generate_potentially_compatibles_cluster`
Creating a table of compatible and potentially compatible cases

---

Description

Creates a table of compatible and potentially compatible cases, with an optional parameter to run a clustering algorithm.

Usage

```
generate_potentially_compatibles_cluster(cases.need60day, create_cluster = F)
```

Arguments

`cases.need60day`	`tibble` Summary table of cases that need 60-day follow-up. This is the output of `generate_60_day_table_data()`.
`create_cluster`	`logical` Add column for clusters? Default to `FALSE`.

Value

`tibble` A summary table of cases.

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
ctry.data <- extract_country_data("algeria", raw.data)
stool.data <- generate_stool_data(
  ctry.data$afp.all.2, "2021-01-01", "2023-12-31",
  "good", "inadequate"
)
table60.days <- generate_60_day_table_data(stool.data, "2021-01-01", "2023-12-31")
pot.c.clust <- generate_potentially_compatibles_cluster(table60.days,
  create_cluster = TRUE
)

## End(Not run)
```

generate_prov_timeliness_graph
Timeliness intervals of samples at the province level

Description

A stacked horizontal bar graph for timeliness intervals of samples at the province level. To get the full intervals from field to lab, the lab data needs to be attached. Otherwise, only the timeliness intervals from the field up to when it was sent to lab will be displayed.

Usage

```
generate_prov_timeliness_graph(
  int.data,
  output_path = Sys.getenv("DR_FIGURE_PATH"),
  afp.prov.year.lab = lifecycle::deprecated()
)
```

Arguments

int.data	tibble Summary table with timeliness intervals at the province level.
output_path	str Path where to output the figure.
afp.prov.year.lab	tibble **[Deprecated]** Deprecated since it is not used anymore.

Value

ggplot Plot of timeliness intervals at the country level.

Examples

```
## Not run:
# Attaching lab data
lab_path <- "C:/Users/ABC1/Desktop/algeria_lab_data.csv"
ctry.data <- init_dr("algeria", lab_data_path = lab_path)
lab.timeliness.prov <- generate_lab_timeliness(ctry.data$lab.data, "prov", start_date, end_date)
int.data.prov <- generate_int_data(ctry.data, start_date, end_date,
  spatial.scale = "prov",
  lab.timeliness.prov
)
generate_ctry_timeliness_graph(int.data.prov)

## End(Not run)
```

generate_sg_priority_map
SG Prioritization Map

Description

[Experimental]

Creates a map showing the SG prioritization map.

Usage

```
generate_sg_priority_map(
  ctry_risk_cat = NULL,
  year = lubridate::year(Sys.Date()),
  ctry_sf = NULL,
  output_path = Sys.getenv("KPI_FIGURES")
)
```

Arguments

ctry_risk_cat	tibble Risk category for each country. Defaults to NULL, which downloads the SG risk category data set from EDAV.
year	tibble Active year for the country shape files.
ctry_sf	sf Country shapefile in long format.
output_path	str Where to output the figure to. Defaults to the path to the figures folder set when running init_kpi().

Value

ggplot A map.

Examples

```
## Not run:
sg_priority_map(output_path = getwd())

## End(Not run)
```

generate_stool_ad_maps
Stool adequacy maps by province

Description

Generates maps that contain the stool adequacy rate for each province per year.

Usage

```
generate_stool_ad_maps(
  ctry.data,
  pstool,
  ctry.shape,
  prov.shape,
  start_date,
  end_date,
  output_path = Sys.getenv("DR_FIGURE_PATH"),
  caption_size = 3
)
```

Arguments

ctry.data	list Large list containing polio data of a country. This is the output of either extract_country_data() or init_dr().
pstool	tibble Stool adequacy table at province level. This is the output of f.stool.ad.01() calculated at the province level.
ctry.shape	sf Country shapefile in long format.
prov.shape	sf Province shapefile in long format.
start_date	str Start date of analysis.
end_date	str End date of analysis.
output_path	str Where to save the figure to.
caption_size	numeric Size of the caption. Defaults to 3.

Value

ggplot A map of stool adequacy rates for each province by year.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
pstool <- f.stool.ad.01(
  afp.data = ctry.data$afp.all.2,
  admin.data = ctry.data$prov.pop,
  start.date = start_date,
  end.date = end_date,
  spatial.scale = "prov",
  missing = "good",
  bad.data = "inadequate",
  rolling = F,
  sp_continuity_validation = F
)
ctry.shape <- load_clean_ctry_sp(ctry_name = "ALGERIA", type = "long")
prov.shape <- load_clean_prov_sp(ctry_name = "ALGERIA", type = "long")
generate_stool_ad_maps(ctry.data, pstool, ctry.shape, prov.shape, "2021-01-01", "2023-12-31")

## End(Not run)
```

generate_stool_ad_maps_dist *Maps of stool adequacy by district and year*

Description

Generates maps of stool adequacy map by district and year.

Usage

```
generate_stool_ad_maps_dist(
  ctry.data,
  dstool,
  ctry.shape,
  prov.shape,
  dist.shape,
  start_date,
  end_date,
  output_path = Sys.getenv("DR_FIGURE_PATH"),
  caption_size = 3
)
```

Arguments

ctry.data	list Large list containing polio data for a country. This is the output of extract_country_data() or init_dr().
dstool	tibble District stool adequacy table. This is the output of f.stool.ad.01() calculated at the district level.
ctry.shape	sf Country shapefile in long format.
prov.shape	sf Province shapefile in long format.
dist.shape	sf District shapefile in long format.
start_date	str Start date of analysis.
end_date	str End date of analysis.
output_path	str Local path where to save the figure to.
caption_size	numeric Size of the caption. Defaults to 3.

Value

ggplot Maps of stool adequacy rates for each district by year.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
dstool <- f.stool.ad.01(
  afp.data = ctry.data$afp.all.2,
  admin.data = ctry.data$dist.pop,
  start.date = start_date,
  end.date = end_date,
```

```
  spatial.scale = "dist",
  missing = "good",
  bad.data = "inadequate",
  rolling = F,
  sp_continuity_validation = F
)
ctry.shape <- load_clean_ctry_sp(ctry_name = "ALGERIA", type = "long")
prov.shape <- load_clean_prov_sp(ctry_name = "ALGERIA", type = "long")
dist.shape <- load_clean_dist_sp(ctry_name = "ALGERIA", type = "long")
generate_stool_ad_maps_dist(
  ctry.data, dstool,
  ctry.shape, prov.shape, dist.shape,
  "2021-01-01", "2023-12-31"
)

## End(Not run)
```

`generate_stool_data` *Generate stool adequacy columns in the AFP dataset*

Description

The function adds the adequacy final column called `adequacy.final` and `adequacy.final2` into the AFP linelist. The function borrows in part from `f.stool.ad.01()`, so that the adequacy final column generated can match with how the stool adequacy function treats bad or missing data and classify the adequacy final column. `adequacy.final` contains the original classification of the sample and `adequacy.final2` contains the final classification according to how missing and bad data are treated.

Usage

```
generate_stool_data(
  afp.data,
  start_date,
  end_date,
  missing = "good",
  bad.data = "inadequate"
)
```

Arguments

`afp.data`	`tibble` AFP linelist. Either `ctry.data$afp.all.2`
`start_date`	`str` Start date of the analysis.
`end_date`	`str` End date of the analysis.
`missing`	`str` How to treat missing data. Valid values are: `"good"`, `"bad"`, `"remove"`. Defaults to `"good"`. When calculating the `adequacy.final` column: • `"good"` uses `adequacy.03` • `"bad"` uses `adequacy.01` • `"exclude"` uses `adequacy.02`
`bad.data`	`str` How to treat bad data. Valid values are:`"remove"`, `"inadequate"`. Defaults to `"inadequate"`. `"inadequate"` treats samples with bad data as inadequate.

Details

Unlike the stool adequacy function, this will not filter out NOT-AFP cases, as it is expected for other functions that use the output of this function to do the filtering. For example, generate_60_day_table_data().

Value

tibble AFP linelist with stool adequacy columns.

See Also

f.stool.ad.01()

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
stool.data <- generate_stool_data(raw.data$afp, "2021-01-01", "2023-12-31")

## End(Not run)
```

---

generate_surv_ind_tab *Surveillance indicator table*

Description

Generates the surveillance indicator table for each year. Outputs the number of AFP cases, national NPAFP rate and stool adequacy,percentage of population living in districts with greater than or equal to 100,000 U15 meeting both indicators.

Usage

```
generate_surv_ind_tab(
  ctry.data,
  ctry.extract,
  dist.extract,
  cstool,
  dstool,
  afp.case,
  country_name = Sys.getenv("DR_COUNTRY")
)
```

Arguments

ctry.data	list Large list containing polio data of a country.
ctry.extract	tibble Country NPAFP rate. Output of f.npafp.rate.01() calculated at the country level.
dist.extract	tibble District NPAFP rate. Output of f.npafp.rate.01() calculated at the district level.
cstool	tibble Country stool adequacy. Output of f.stool.ad.01() calculated at the country level.

dstool	tibble District stool adequacy. Output of f.stool.ad.01() calculated at the district level.
afp.case	tibble AFP case counts. Output of generate_afp_by_month_summary() with by="year".
country_name	str Name of the country.

Value

flextable Table summarizing yearly trends in NPAFP and stool adequacy at the national level.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
ctry.extract <- f.npafp.rate.01(
  afp.data = ctry.data$afp.all.2,
  pop.data = ctry.data$ctry.pop,
  start.date = start_date,
  end.date = end_date,
  spatial.scale = "ctry",
  pending = T,
  rolling = F,
  sp_continuity_validation = F
)
dist.extract <- f.npafp.rate.01(
  afp.data = ctry.data$afp.all.2,
  pop.data = ctry.data$ctry.pop,
  start.date = start_date,
  end.date = end_date,
  spatial.scale = "dist",
  pending = T,
  rolling = F,
  sp_continuity_validation = F
)
cstool <- f.stool.ad.01(
  afp.data = ctry.data$afp.all.2,
  admin.data = ctry.data$ctry.pop,
  start.date = start_date,
  end.date = end_date,
  spatial.scale = "ctry",
  missing = "good",
  bad.data = "inadequate",
  rolling = F,
  sp_continuity_validation = F
)
dstool <- f.stool.ad.01(
  afp.data = ctry.data$afp.all.2,
  admin.data = ctry.data$dist.pop,
  start.date = start_date,
  end.date = end_date,
  spatial.scale = "dist",
  missing = "good",
  bad.data = "inadequate",
  rolling = F,
  sp_continuity_validation = F
)
```

```
afp.by.month <- generate_afp_by_month(ctry.data$afp.all.2, "2021-01-01", "2023-12-31")
afp.case <- generate_afp_by_month_summary(afp.by.month, ctry.data, start_date, end_date, "year")
generate_surv_ind_tab(ctry.data, ctry.extract, dist.extract, cstool, dstool, afp.case)

## End(Not run)
```

`generate_timeliness_maps` *Maps evaluating timeliness of samples against timeliness targets.*

Description

Generates a map at the provincial level summarizing the timeliness of samples across different timeliness targets. The figure is faceted by the type of timeliness target, with each facet containing the percentage of samples from each province that met the targets over the years.

Usage

```
generate_timeliness_maps(
  ctry.data,
  ctry.shape,
  prov.shape,
  start_date,
  end_date,
  mark_x = T,
  pt_size = 4,
  output_path = Sys.getenv("DR_FIGURE_PATH")
)
```

Arguments

`ctry.data`	`list` Large list containing polio data for a country. This is the output of `extract_country_data()` or `init_dr()`.
`ctry.shape`	`sf` Country shapefile in long format.
`prov.shape`	`sf` Province shapefile in long format.
`start_date`	`str` Start date of analysis.
`end_date`	`str` End date of analysis.
`mark_x`	`logical` Mark where there are less than 5 AFP cases? Defaults to `TRUE`.
`pt_size`	`numeric` Size of the marks.
`output_path`	`str` Local path where to save the figure to.

Value

`ggplot` Faceted map of each province evaluated against timeliness targets across years.

Examples

```
## Not run:
ctry.data <- init_dr("algeria")
ctry.shape <- load_clean_ctry_sp(ctry_name = "ALGERIA", type = "long")
prov.shape <- load_clean_prov_sp(ctry_name = "ALGERIA", type = "long")
generate_timeliness_maps(ctry.data, ctry.shape, prov.shape, "2021-01-01", "2023-12-31")

## End(Not run)
```

`generate_timely_det_violin` *Timely detection of AFP/ES samples*

Description

[Experimental]

Generates a violin plot highlighting the median detection time of samples.

Usage

```
generate_timely_det_violin(
  raw_data,
  start_date,
  end_date,
  priority_level = c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW"),
  who_region = NULL,
  rolling = TRUE,
  output_path = Sys.getenv("KPI_FIGURES"),
  y_max = 250
)
```

Arguments

`raw_data`	`list` Global polio data. Output of `get_all_polio_data()`
`start_date`	`str` Analysis start date formatted as "YYYY-MM-DD".
`end_date`	`str` Analysis end date formatted as "YYYY-MM-DD".
`priority_level`	`list` Priority levels to display. Defaults to `c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW")`.
`who_region`	`list` Regions to display. Defaults to `NULL`, which shows all of the regions.
`rolling`	`logical` Using rolling periods or year-to-year? Defaults to `TRUE`.
`output_path`	`str` Where to output the figure to.
`y_max`	`num` The maximum y-axis value.

Value

`ggplot` A violin plot showing timeliness of detection.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
generate_timely_det_violin(raw_data, "2021-01-01", "2023-12-31")

## End(Not run)
```

`generate_timely_ship_violin`
Timely shipment of AFP stool samples

Description

[Experimental]

Generates a violin plot highlighting the median detection time of stool shipment to lab.

Usage

```
generate_timely_ship_violin(
  afp_data,
  start_date,
  end_date,
  priority_level = c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW"),
  who_region = NULL,
  rolling = TRUE,
  output_path = Sys.getenv("KPI_FIGURES"),
  y_max = 60
)
```

Arguments

`afp_data`	`list` AFP data.
`start_date`	`str` Analysis start date formatted as "YYYY-MM-DD".
`end_date`	`str` Analysis end date formatted as "YYYY-MM-DD".
`priority_level`	`list` Priority levels to display. Defaults to `c("HIGH", "MEDIUM", "LOW (WATCHLIST)", "LOW")`.
`who_region`	`list` Regions to display. Defaults to `NULL`, which shows all of the regions.
`rolling`	`logical` Using rolling periods or year-to-year? Defaults to `TRUE`.
`output_path`	`str` Where to output the figure to.
`y_max`	`num` The maximum y-axis value.

Value

`ggplot` A violin plot showing timeliness of detection.

Examples

```
## Not run:
raw_data <- get_all_polio_data()
generate_timely_ship_violin(raw_data$afp, "2021-01-01", "2023-12-31")

## End(Not run)
```

`get_all_polio_data` *Retrieve all pre-processed polio data*

Description

Download POLIS data from the CDC pre-processed endpoint. By default this function will return a "small" or recent dataset. This is primarily for data that is from 2019 onwards. You can specify a "medium" sized dataset for data that is from 2016 onwards. Finally the "large" sized dataset will provide information from 2001 onwards. Regular pulls form the data will recreate the "small" dataset when new information is availble and the Data Management Team can force the creation of the "medium" and "large" static datasets as necessary.

Usage

```
get_all_polio_data(
  size = "small",
  data_folder = "GID/PEB/SIR/Data",
  polis_folder = "GID/PEB/SIR/POLIS",
  core_ready_folder = "Core_Ready_Files",
  force.new.run = FALSE,
  recreate.static.files = FALSE,
  attach.spatial.data = TRUE,
  use_edav = TRUE,
  use_archived_data = FALSE,
  archive = TRUE,
  keep_n_archives = Inf,
  output_format = "rds"
)
```

Arguments

`size`	`str` Size of data to download. Defaults to `"small"`. • `"small"`: Data from 2019-present. • `"medium"`: Data from 2016-present. • `"large"`: Data from 2001-present.
`data_folder`	`str` Location of the data folder containing pre-processed POLIS data, spatial files, coverage data, and population data. Defaults to `"GID/PEB/SIR/Data"`.
`polis_folder`	`str` Location of the POLIS folder. Defaults to `"GID/PEB/SIR/POLIS"`.
`core_ready_folder`	`str` Which core ready folder to use. Defaults to `"Core_Ready_Files"`.
`force.new.run`	`logical` Default `FALSE`, if `TRUE` will run recent data and cache.

recreate.static.files
: logical Default FALSE, if TRUE will run all data and cache.

attach.spatial.data
: logical Default TRUE, adds spatial data to downloaded object.

use_edav
: logical Build raw data list using EDAV files. Defaults to TRUE.

use_archived_data
: logical Allows the ability to recreate the raw data file using previous preprocessed data. If

archive
: Logical. Whether to archive previous output directories before overwriting. Default is TRUE.

keep_n_archives
: Numeric. Number of archive folders to retain. Defaults to Inf, which keeps all archives. Set to a finite number (e.g., 3) to automatically delete older archives beyond the N most recent.

output_format
: str: output_format to save files as. Available formats include 'rds' 'rda' 'csv' 'qs2' and 'parquet', Defaults is #' 'rds'.

Value

Named list containing polio data that is relevant to CDC.

Examples

```
## Not run:
raw.data <- get_all_polio_data() # downloads data since 2019, including spatial files
raw.data <- get_all_polio_data(size = "small", attach.spatial.data = FALSE) # exclude spatial data

## End(Not run)
```

---

get_azure_storage_connection

Validate connection to EDAV

Description

Generate token which connects to CDC EDAV resources and validates that the individual still has access. The current tenant ID is hard coded for CDC resources.

Usage

```
get_azure_storage_connection(
  app_id = "04b07795-8ddb-461a-bbee-02f9e1bf7b46",
  auth = "authorization_code",
  posit_yaml_path = NULL,
  ...
)
```

Arguments

app_id	str Application ID defaults to "04b07795-8ddb-461a-bbee-02f9e1bf7b46", this can be changed if you have a service principal.
auth	str Authorization type defaults to "authorization_code", this can be changed if you have a service principal. Valid values are:"authorization_code", "device_code", "client_credentials", "resource_owner", "on_behalf_of". See **Details** of AzureAuth::get_azure_token() for further details.
posit_yaml_path	str Path to the YAML file in Posit workbench. If NULL, the path will be in "~/credentials/posit_workbench_creds.yaml".
...	additional parameters passed to AzureAuth::get_azure_token().

Value

Azure container verification

Examples

```
azcontainer <- get_azure_storage_connection()
```

---

get_cdc_childvaxview_data

Pull CDC NCIRD childvaxview immunization coverage data

Description

Pull coverage data from API and filter by desired geographic level and vaccines.

Usage

```
get_cdc_childvaxview_data(
  geo_level = NULL,
  vaccines = NULL,
  limit = 1000,
  base_url = "https://data.cdc.gov/resource/fhky-rtsk.json"
)
```

Arguments

geo_level	str Geographic categories of coverage data. Choose from: 'national', 'regional', 'state', or 'substate'.
vaccines	str A string or vector of strings of vaccines for which to provide coverage data. Choose from: 'DTaP', 'Polio', 'Hep B', 'PCV', 'Varicella', 'MMR', 'Hib', 'Hep A', 'Influenza', 'Rotavirus', 'Combined 7 series'.
limit	int Number of rows to download, defaults to max allowed (1000).
base_url	str URL to US CDC NCIRD API endpoint. Defaults to "https://data.cdc.gov/resource/fhky-rtsk.json".

Value

tibble Dataframe of vaccine coverage estimates for all VPDs.

Examples

```
## Not run:
cdc_data <- get_cdc_childvaxview_data(geo_level="substate")
cdc_data <- get_cdc_childvaxview_data(geo_level="national", vaccines=c("Polio","MMR"))

## End(Not run)
```

---

get_constant *Obtain a constant variable used in sirfunctions*

Description

Some links used in certain functions are hardcoded, but may change in the future. For ease of link maintenance, constants will be added to this function.

Usage

```
get_constant(constant_name = NULL)
```

Arguments

constant_name	str Name of the constant. Valid values include:

- "DEFAULT_EDAV_FOLDER"
- "CTRY_RISK_CAT"
- "LAB_LOCATIONS"
- "DR_TEMPLATE"
- "SIRFUNCTIONS_GITHUB_TREE"
- "AFRO_LAB_DATA"
- "EMRO_LAB_DATA"
- "CLEANED_LAB_DATA"

Value

str A string, typically a file path or a URL.

Examples

```
get_constant("DEFAULT_EDAV_FOLDER")
```

get_ctry_abbrev *Get country abbreviations*

Description

[Experimental]

Gets the country abbreviation from the AFP dataset.

Usage

```
get_ctry_abbrev(afp_data)
```

Arguments

afp_data	tibble AFP dataset

Value

tibble A tibble with the country, abbreviation, and WHO region

Examples

```
## Not run:
raw_data <- get_all_polio_data(attach.spatial.data = FALSE)
ctry_abbrev <- get_ctry_abbrev(raw_data$afp)

## End(Not run)
```

get_diff_cols *Get the columns where records differ in a group*

Description

Get the columns where duplicates differ after performing a dplyr::distinct() operation. In some instances, two records might exist with the same unique identifier. In datasets with lots of columns, it is difficult to figure out which columns these potential duplicates differ. The function outputs the columns where records with the same unique identifier differ.

Usage

```
get_diff_cols(df, id_col)
```

Arguments

df	df or tibble Dataframe with at least one column containing unique identifiers and other columns.
id_col	str Column used as a unique identifier for records.

Value

tibble A tibble showing the columns where duplicates differ.

Examples

```
df1 <- dplyr::tibble(col1 = c(1, 1, 2), col2 = c("a", "b", "c"), col3 = c(1, 1, 3))
diff_cols <- get_diff_cols(df1, "col1")
```

---

get_lab_date_col_missingness *Get missingness of date variables in the lab dataset*

Description

Obtains the percentage of missingness in date variables within the lab dataset

Usage

```
get_lab_date_col_missingness(lab_data, group_by = NULL)
```

Arguments

lab_data	tibble Lab data.
group_by	str A column or a vector of columns to group results by.

Value

tibble Summary of missingness of date variables

Examples

```
## Not run:
get_lab_date_col_missingness(lab_data)

## End(Not run)
```

---

get_lab_locs *Table of information regarding testing labs in each country*

Description

Imports information on testing labs for each country, either from a CSV file or downloaded from EDAV. If no argument is passed, the function will download the table from EDAV.

Usage

```
get_lab_locs(path = NULL)
```

Arguments

path	str Path to the lab location file. Defaults to NULL.

Value

tibble A table containing the test lab location information.

Examples

```
## Not run:
ctry.seq <- get_lab_locs()

## End(Not run)
```

---

get_ppt_template *Get path of the PowerPoint template*

Description

The desk review PowerPoint template is used to build the desk review slide deck. The function will either download the template from the sg-desk-reviews GitHub page or get it locally.

Usage

```
get_ppt_template(path = NULL)
```

Arguments

path	str Path to the PowerPoint template. If NULL, will prompt user to download from the sg-desk-review GitHub repository

Value

str Local path of the PowerPoint template.

Examples

```
## Not run:
get_ppt_template()

# If present locally
template_path <- "C:/Users/ABC1/Desktop/deskreview_template.pptx"
ppt_template <- get_ppt_template(template_path)

## End(Not run)
```

get_region *Determines whether lab data is EMRO or AFRO*

Description

Outputs the name of the region which a country belongs to.

Usage

```
get_region(country_name = Sys.getenv("DR_COUNTRY"))
```

Arguments

country_name	str Name of the country.

Value

str A string, either "EMRO" or "AFRO".

Examples

```
## Not run:
get_region("algeria")

## End(Not run)
```

get_vpd_data *Get vaccine preventable diseases dataset*

Description

[Experimental]

Gets the VPD dataset from EDAV.

Usage

```
get_vpd_data(
  vpd_name = NULL,
  variable_name = NULL,
  years = NULL,
  ctry_name = NULL,
  iso3_codes = NULL,
  add_ctry_sf = TRUE,
  add_ctry_pop = TRUE
)
```

Arguments

vpd_name	str A VPD or a list of VPDs. Defaults to NULL, which returns the full VPD dataset.
variable_name	str A variable or a list of variables. Defaults to NULL, which returns a dataset containing all the variable names.
years	int A year or a list of years. Defaults to NULL, which returns a dataset containing all the years available.
ctry_name	str A country or a list country names. Defaults to NULL, which returns all the countries in the dataset.
iso3_codes	str An ISO3 code or a list of ISO3 codes. Defaults to NULL, which returns all the ISO3 codes in the dataset.
add_ctry_sf	logical Attach the country shapefile? Defaults to TRUE.
add_ctry_pop	logical Attach the country population data? Defaults to TRUE.

Value

list A list containing the VPD data, and optionally the shapefile and population data.

Examples

```
## Not run:
vpd_data <- get_vpd_data()

## End(Not run)
```

---

get_vpd_missingness *Get years with missing data based on variable name and VPD*

Description

[Experimental]

For each pair of VPD and variable name, returns the years with missing data for each country.

Usage

```
get_vpd_missingness(
  vpd_name = NULL,
  variable_name = NULL,
  min_year = 1980,
  max_year = lubridate::year(Sys.Date())
)
```

Arguments

vpd_name	str A VPD name or a list of names.
variable_name	str A variable name or a list of names.
min_year	int Minimum year to analyze. Defaults to 1980.
max_year	int Maximum year to analyze. Defaults to the current year.

Value

tibble A summary table of years with missing data for each country for a particular VPD and variable name.

Examples

```
## Not run:
missing_years <- get_vpd_missingness("Cholera", "cases")

## End(Not run)
```

---

get_vpd_vars *Obtain available variables for VPD data*

Description

lifecycle::badge("experimental")

Obtains the variable names available for each vaccine preventable disease (VPD) in the variable column.

Usage

```
get_vpd_vars()
```

Value

tibble A tibble with the available variable choices.

Examples

```
## Not run:
vpd_vars <- get_vpd_vars()

## End(Not run)
```

---

init_dr *Set up the folders and load polio data for the desk review*

Description

Prepares the folders and files required for the desk review. The function primarily serves to organize the files used for the desk review and set standardized environmental variables (i.e., Sys.getenv(), where values for x related to the desk review is prefixed with "DR") . The function only supports running one country at a time.

Usage

```
init_dr(
  country_name,
  start_date = NULL,
  end_date = NULL,
  local_dr_folder = getwd(),
  sg_dr_folder = NULL,
  lab_data_path = NULL,
  iss_data_path = NULL,
  attach_spatial_data = T,
  branch = "main",
  source = T,
  data_folder = "GID/PEB/SIR/Data",
  polis_folder = "GID/PEB/SIR/POLIS",
  use_edav = TRUE
)
```

Arguments

country_name	str Name of the country.
start_date	str Start date of the desk review. If NULL, defaults to four years from when the function was ran on January 1st.
end_date	str End date of the desk review. If NULL, defaults to six weeks from when the function is ran.
local_dr_folder	str Folder where the desk review code is located. Defaults to the current working directory.
sg_dr_folder	str Folder where the local git repository is located. Defaults to NULL.
lab_data_path	str Location of the lab data. Defaults to NULL.
iss_data_path	str Location of the ISS data. Defaults to NULL.
attach_spatial_data	logical Whether to include spatial data. Defaults to TRUE.
branch	str What branch to download the DR functions from GitHub. "main" is the default, which contains the official version of the package. Other branches, like "dev" may contain experimental features not yet available in the "main" branch.
source	logical Whether to source local functions or use sirfunctions. Defaults to TRUE.
data_folder	str Location of the data folder containing pre-processed POLIS data, spatial files, coverage data, and population data. Defaults to "GID/PEB/SIR/Data".
polis_folder	str Location of the POLIS folder. Defaults to "GID/PEB/SIR/POLIS".
use_edav	logical Build raw data list using EDAV files. Defaults to TRUE.

Value

list A list containing all dataframe for all polio data.

Examples

```
## Not run:
ctry.data <- init_dr("algeria", source = F) # Sets up folder in the current working directory
```

```
ctry.data <- init_dr("algeria", branch = "dev") # Use functions from the dev branch

## End(Not run)
```

`init_kpi` *Initialize the KPI analysis pipeline*

Description

[Experimental]

Sets up folder structures and environmental variables, as well as download global polio data.

Usage

```
init_kpi(path = getwd(), name = NULL, edav = TRUE)
```

Arguments

`path`	`str` Path to the folder containing the KPI analysis folders. Defaults to current working directory.
`name`	`str` Name of the KPI analysis folder. If not given any names, the folder will be named the date the function is ran.
`edav`	`logical` Whether to use EDAV to load the raw_data and lab_data files. Defaults to `TRUE`.

Value

Does not return anything

Examples

```
## Not run:
init_kpi(name = "kpi_jan_2024")

## End(Not run)
```

`iss_data_errors` *Checks for errors in the ISS data*

Description

Currently, the function reports the number of missing priority levels.

Usage

```
iss_data_errors(
  iss_data,
  error_path = Sys.getenv("DR_ERROR_PATH"),
  ctry.data = lifecycle::deprecated()
)
```

Arguments

iss_data	tibble ISS data.
error_path	str Path to error folder. The function defaults to a global environment variable called DR_ERROR_PATH, as it is assumed ISS data error checking is done as part of the desk review template. The setting of desk review environmental variables is automatically handled by init_dr(). Otherwise, users should manually specify the error folder.
ctry.data	list **[Deprecated]** Please pass the ISS data directly to the iss.data parameter.

Value

Status messages on the checks completed and results.

Examples

```
## Not run:
iss_path <- "C:/Users/ABC1/Desktop/iss_data.csv"
ctry.data <- init_dr("somalia", iss_data_path = iss_path)
iss_data_errors(ctry.data$iss.data)

## End(Not run)
```

---

lab_data_errors *Generate a log of potential errors in the lab data*

Description

Checks the loaded lab data for potential issues. The function will detect whether the lab data loaded either came from the regional office or from global.

Usage

```
lab_data_errors(
  lab.data,
  afp.data,
  start.date = start_date,
  end.date = end_date,
  ctry_name = Sys.getenv("DR_COUNTRY"),
  error_path = Sys.getenv("DR_ERROR_PATH"),
  ctry.data = lifecycle::deprecated()
)
```

Arguments

lab.data	tibble Polio lab data.
afp.data	tibble AFP linelist.
start.date	str Start date of the analysis.
end.date	str End date of the analysis.

ctry_name	list or str A name of a country or a list of countries. Please pass lab data directly into lab.data parameter instead.
error_path	str File path to store the error log.
ctry.data	list **[Deprecated]**

Value

None. It outputs locally an Excel file containing the error log.

Examples

```
## Not run:
lab_path <- "C:/Users/ABC1/Desktop/lab_data.xlsx"
start_date <- "2021-01-01"
end_date <- "2023-12-31"
ctry.data <- init_dr("algeria", lab_data_path = lab_path)
lab_data_errors(ctry.data$lab.data, ctry.data$afp.data)

## End(Not run)
```

---

load_clean_ctry_sp *Download country georgraphic data*

Description

Pulls country shapefiles directly from the geodatabase.

Usage

```
load_clean_ctry_sp(
  azcontainer = suppressMessages(get_azure_storage_connection()),
  fp = "GID/PEB/SIR/Data/spatial/global.ctry.rds",
  ctry_guid = NULL,
  ctry_name = NULL,
  end_year = lubridate::year(Sys.Date()),
  st_year = 2000,
  data_only = FALSE,
  type = NULL,
  version = "standard",
  edav = TRUE,
  end.year = lifecycle::deprecated(),
  st.year = lifecycle::deprecated(),
  data.only = lifecycle::deprecated()
)
```

Arguments

azcontainer	Azure validated container object
fp	str Location of geodatabase.
ctry_guid	str array Array of all country GUIDS that you want to pull.
ctry_name	str array Array of all country names that you want to pull.

end_year	int Last year you want to pull information for. Default is current year.
st_year	int Earlier year of spatial data you want to pull. Default is 2000.
data_only	logical Whether to return a tibble with shapefiles or not. Defaults to FALSE.
type	str Whether to return a spatial object for every year group. Defaults to NULL. • "long" Return a dataset for every year group. • NULL Return a dataset only with unique GUIDs and when they were active.
version	str Specify whether to return standard shapefiles or new shapefiles still under evaluation/development. Default is "standard". • "standard" Standard shapefiles. • "dev" New shapefiles still under evaluation/development.
edav	logical Load data from EDAV? Defaults to TRUE.
end.year	int **[Deprecated]** Renamed in favor of end_year.
st.year	int **[Deprecated]** Renamed in favor of st_year.
data.only	logical **[Deprecated]** Renamed in favor of data_only.

Value

tibble or sf Dataframe containing spatial data.

Examples

```
## Not run:
ctry <- load_clean_ctry_sp(ctry_name = "ALGERIA")
ctry.long <- load_clean_ctry_sp(ctry_name = "ALGERIA", type = "long")

## End(Not run)
```

---

load_clean_dist_sp *Download district geographic data*

Description

Pulls district shapefiles directly from the geodatabase.

Usage

```
load_clean_dist_sp(
  azcontainer = suppressMessages(get_azure_storage_connection()),
  fp = "GID/PEB/SIR/Data/spatial/global.dist.rds",
  dist_guid = NULL,
  dist_name = NULL,
  prov_name = NULL,
  ctry_name = NULL,
  end_year = lubridate::year(Sys.Date()),
  st_year = 2000,
  data_only = FALSE,
  type = NULL,
  version = "standard",
```

```
  edav = TRUE,
  end.year = lifecycle::deprecated(),
  st.year = lifecycle::deprecated(),
  data.only = lifecycle::deprecated()
)
```

Arguments

azcontainer	Azure validated container object.
fp	str Location of geodatabase.
dist_guid	str array Array of all district GUIDS that you want to pull.
dist_name	str array Array of all dist names that you want to pull.
prov_name	str array Array of all province names that you want to pull.
ctry_name	str array Array of all country names that you want to pull.
end_year	int Last year you want to pull information for. Default is current year.
st_year	int Earlier year of spatial data you want to pull. Default is 2000.
data_only	logical Whether to return a tibble with shapefiles or not. Defaults to FALSE.
type	str Whether to return a spatial object for every year group. Defaults to NULL. • "long" Return a dataset for every year group. • NULL Return a dataset only with unique GUIDs and when they were active.
version	str Specify whether to return standard shapefiles or new shapefiles still under evaluation/development. Default is "standard". • "standard" Standard shapefiles. • "dev" New shapefiles still under evaluation/development.
edav	logical Load from EDAV? Defaults to TRUE.
end.year	int **[Deprecated]** Renamed in favor of end_year.
st.year	int **[Deprecated]** Renamed in favor of st_year.
data.only	logical **[Deprecated]** Renamed in favor of data_only.

Value

tibble or sf Dataframe containing spatial data.

Examples

```
## Not run:
dist <- load_clean_dist_sp(ctry_name = c("ALGERIA", "NIGERIA"), st.year = 2019)
dist.long <- load_clean_dist_sp(ctry_name = "ALGERIA", st.year = 2019, type = "long")

## End(Not run)
```

load_clean_prov_sp *Download province geographic data*

Description

Pulls province shapefiles directly from the geodatabase

Usage

```
load_clean_prov_sp(
  azcontainer = suppressMessages(get_azure_storage_connection()),
  fp = "GID/PEB/SIR/Data/spatial/global.prov.rds",
  prov_guid = NULL,
  prov_name = NULL,
  ctry_name = NULL,
  end_year = lubridate::year(Sys.Date()),
  st_year = 2000,
  data_only = FALSE,
  type = NULL,
  version = "standard",
  edav = TRUE,
  end.year = lifecycle::deprecated(),
  st.year = lifecycle::deprecated(),
  data.only = lifecycle::deprecated()
)
```

Arguments

azcontainer	Azure validated container object
fp	str Location of geodatabase.
prov_guid	str array Array of all province GUIDS that you want to pull.
prov_name	str array Array of all province names that you want to pull.
ctry_name	str array Array of all country names that you want to pull.
end_year	int Last year you want to pull information for. Default is current year.
st_year	int Earlier year of spatial data you want to pull. Default is 2000.
data_only	logical Whether to return a tibble with shapefiles or not. Defaults to FALSE.
type	str Whether to return a spatial object for every year group. Defaults to NULL. • "long" Return a dataset for every year group. • NULL Return a dataset only with unique GUIDs and when they were active.
version	str Specify whether to return standard shapefiles or new shapefiles still under evaluation/development. Default is "standard". • "standard" Standard shapefiles. • "dev" New shapefiles still under evaluation/development.
edav	logical Load from EDAV? Defaults to TRUE.
end.year	int **[Deprecated]** Renamed in favor of end_year.
st.year	int **[Deprecated]** Renamed in favor of st_year.
data.only	logical **[Deprecated]** Renamed in favor of data_only.

Value

tibble or sf Dataframe containing spatial data.

Examples

```
## Not run:
prov <- load_clean_prov_sp(ctry_name = c("ALGERIA", "NIGERIA"), st_year = 2019)
prov.long <- load_clean_prov_sp(ctry_name = "ALGERIA", st_year = 2019, type = "long")

## End(Not run)
```

---

load_iss_data *Read ISS/eSURV data*

Description

The function is written to assist in load the ISS data from a path specified by the user during init_dr(). This function is not meant to be exported.

Usage

```
load_iss_data(iss_path, sheet_name = NULL)
```

Arguments

iss_path	str Path to the excel or csv file.
sheet_name	str Optional name of the ISS data. This is mainly used if the path is to an Excel file and that Excel file has multiple tabs.

Value

tibble ISS/eSURV data loaded into a tibble.

Examples

```
## Not run:
iss_path <- "C:/Users/ABC1/Desktop/iss_data.csv"
iss_data <- load_iss_data(iss_path)

## End(Not run)
```

load_lab_data *Function to load the raw lab data*

Description

This a function to load lab data that are either CSVs or Excel files.

Usage

```
load_lab_data(lab_data_path, sheet_name = NULL)
```

Arguments

lab_data_path	str File path as a string to the lab data.
sheet_name	str Name of the sheet to load. This is optional in cases of an Excel sheet with multiple tabs.

Value

tibble Lab data loaded from the CSV or Excel file path.

Examples

```
## Not run:
lab_data_path <- "C:/Users/ABC1/Desktop/lab_data.csv"
lab_data <- load_lab_data(lab_data_path)

## End(Not run)
```

send_outlook_email *Send email through Outlook*

Description

Function to send an email through Outlook from R.

Usage

```
send_outlook_email(title, body, recipient, attachment = NULL)
```

Arguments

title	str Subject of message to be sent.
body	str Long string of body of message to be sent.
recipient	str A semicolon separated list of recipients.
attachment	str Path to local document to be attached to email. Defaults to NULL.

Value

Status message whether the operation was a success or an error message.

Examples

```
## Not run:
title_message <- "Test"
body_message <- "this is a test"
recipient_list <- c("ab123@email.com")
send_outlook_email(title_message, body_message, recipient_list)

## End(Not run)
```

---

send_teams_message *Send a message on Microsoft Teams*

Description

Helper function to send message to validated MS Teams interface.

Usage

```
send_teams_message(
  msg,
  team_id = "CGH-GID-PEB-SIR",
  channel = "CORE 2.0",
  attach = NULL,
  type = "text"
)
```

Arguments

msg	str Message to be sent.
team_id	str Teams ID. Defaults to "CGH-GID-PEB-SIR".
channel	str Channel where message should be sent.
attach	str Local path of files to be attached in message.
type	str Type of message to be sent. Either "text" or "html".

Value

Status message whether the operation was a success or an error message.

Examples

```
## Not run:
message <- "this is a test"
send_teams_message(message)

## End(Not run)
```

`set_emergence_colors` *Set the emergence colors*

Description

Used in conjunction to `generate_adhoc_map()`. The function returns a named list with emergence names mapped to a color.

Usage

```
set_emergence_colors(
  raw.data,
  country,
  start_date = NULL,
  end_date = NULL,
  get_unassigned = FALSE
)
```

Arguments

`raw.data`	`list` Global polio data output of `get_all_polio_data()`.
`country`	`str` or `list` Countries of interest.
`start_date`	`str` Start date of the time span to look for emergences. Defaults to 13 months from the end date.
`end_date`	`str` End date of the time span to look for emergences Defaults to download date of `raw.data`.
`get_unassigned`	`logical` Get a list of emergence without a color mapped. This parameter is useful for ensuring that emergences are all accounted for when making a map.

Value

`list` A named list containing the mapping of emergence and corresponding colors.

Examples

```
## Not run:
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
emg.cols <- set_emergence_colors(raw.data, "algeria")

## End(Not run)
```

`sirfunctions_io` *sirfunctions i/o handler*

Description

[Experimental]

Manages read/write/list/create/delete functions for sirfunctions. This function is adapted from tidypolis_io.

Usage

```
sirfunctions_io(
  io,
  default_folder = "GID/PEB/SIR",
  file_loc,
  obj = NULL,
  edav = TRUE,
  azcontainer = suppressMessages(sirfunctions::get_azure_storage_connection()),
  full_names = T,
  ...
)
```

Arguments

`io`	`str` The type of operation to use. Valid values include: • `"read"`: reads data from the specified `file_path`. • `"write"`: writes data to the specified `file_path`. • `"list"`: lists the files in the specified `file_path`. • `"exists.dir"`: determines whether a directory is present. • `"exists.file"`: determines whether a file is present. • `"create.dir"`: creates a directory to the specified `file_path`. • `"delete"`: deletes a file or folder in the specified `file_path`.
`default_folder`	`str` The default folder to use. Defaults to `"GID/PEB/SIR`.
`file_loc`	`str` Path of file relative to the `default_folder`.
`obj`	`str` Object to be loaded into EDAV
`edav`	`logical` Whether the function should interact with the EDAV environment. Defaults to `TRUE`, otherwise, interacts with files locally.
`azcontainer`	`Azure container` A container object returned by `get_azure_storage_connection()`.
`full_names`	`logical` If `io="list"`, include the full reference path. Default `TRUE`.
`...`	Optional parameters that work with `readr::read_delim()` or `readxl::read_excel()`.

Value

Conditional on `io`. If `io` is `"read"`, then it will return a tibble. If `io` is `"list"`, it will return a list of file names. Otherwise, the function will return `NULL`. `exists.dir` and `exists.file` will return a `logical`.

Examples

```
## Not run:
df <- sirfunctions_io("read", file_loc = "df1.csv") # read file from EDAV
# Passing parameters that work with read_csv or read_excel, like sheet or skip.
df2 <- sirfunctions_io("read", file_loc = "df2.xlsx", sheet = 1, skip = 2)
list_of_df <- list(df_1 = df, df_2 = df)
# Saves df to the test folder in EDAV
sirfunctions_io("write", file_loc = "Data/test/df.csv", obj = df)
# Saves list_of_df as an Excel file with multiple sheets.
sirfunctions_io("write", file_loc = "Data/test/df.xlsx", obj = list_of_df)
sirfunctions_io("exists.dir", "Data/nonexistentfolder") # returns FALSE
sirfunctions_io("exists.file", file_loc = "Data/test/df1.csv") # returns TRUE
sirfunctions_io("create", "Data/nonexistentfolder") # creates a folder called nonexistentfolder
sirfunctions_io("list") # list all files from the default directory

## End(Not run)
```

---

test_EDAV_connection *Test network connection to the EDAV*

Description

Tests upload and download from EDAV by creating a temporary file of a given size and testing the time it takes to upload and download the file.

Usage

```
test_EDAV_connection(
  azcontainer = suppressMessages(get_azure_storage_connection()),
  folder = "GID/PEB/SIR/Data",
  return_list = F,
  test_size = 1e+07
)
```

Arguments

azcontainer	Azure storage container provided by get_azure_storage_connection().
folder	str Location of folder in the EDAV environment that you want to download and upload data from.
return_list	logical return a list of download time estimates. Defaults to FALSE.
test_size	int byte size of a theoretical file to be uploaded or downloaded.

Value

System message with download and update time, potentially a list.

Examples

```
## Not run:
test_EDAV_connection()

## End(Not run)
```

`update_polio_data` *Update a local dataset with new data*

Description

[Experimental]

Update a local global polio data (raw.data) with new data.

Usage

```
update_polio_data(local_dataset, overwrite = T)
```

Arguments

`local_dataset`	`str` File path to the global polio data RDS file.
`overwrite`	`logical` Should the file be overwritten? Default `TRUE`.

Value

None.

Examples

```
## Not run:
local_raw_data <- "C:/Users/ABC1/Desktop/raw.data.rds"
update_polio_data(local_raw_data, overwrite = FALSE)

## End(Not run)
```

`upload_dr_to_github` *Upload desk review script to the sg-desk-reviews GitHub repository*

Description

Upload the desk review template script to the sg-desk-reviews repository, which houses the code for the desk reviews. This function can be used in a general sense to upload files to a github repository. Note that the function will only commit, and that the user must push themselves.

Usage

```
upload_dr_to_github(file_path, repo_path, message = "updating file")
```

Arguments

file_path	str Location of the file to upload to the sg-desk-reviews repo.
repo_path	str Local path of the sg-desk-review repo.
message	str Message to include in the commit.

Value

A status message.

Examples

```
## Not run:
dr_template_path <- "C:/Users/ABC1/Desktop/local_dr/algeria/2024/algeria_template.Rmd"
repo_path <- "C:/Users/ABC1/Desktop/github/sg-desk-reviews"
str_message <- "Added algeria to the SG folder"
upload_dr_to_github(dr_template_path, repo_path, str_message)

## End(Not run)
```

---

upload_to_sharepoint *Upload file to Sharepoint*

Description

Helper function to upload file to MS SharePoint

Usage

```
upload_to_sharepoint(
  file_to_upload,
  sharepoint_file_loc,
  site = "https://cdc.sharepoint.com/teams/CGH-GID-PEB-SIR283",
  drive = "Documents"
)
```

Arguments

file_to_upload	str Local path of files to be uploaded.
sharepoint_file_loc	str Location in SharePoint to upload file. Must include the file name and extension (i.e., folder/file_name.csv).
site	str SharePoint site location. Defaults to "CGH-GID-PEB" or the site URL: "https://cdc.sharepoint.com/teams/CGH-GID-PEB-SIR283".
drive	str SharePoint drive to upload data to.

Value

Status message whether the operation was a success or an error message.

Examples

```
## Not run:
file_path <- "C:/Users/ABC1/df1.csv"
sp_path <- "test_folder/df1.csv"
upload_to_sharepoint(file_path, sp_path)

## End(Not run)
```

Index

add_rolling_years, 5
AzureAuth::get_azure_token(), *92*

check_afp_geographies, 6
check_afp_guid_ctry_data, 6
check_afp_guid_ctry_data(), *31*
check_missing_rows, 7
clean_ctry_data, 8
clean_ctry_data(), *46*, *54*, *57*
clean_es_data, 8
clean_es_data(), *52, 53*
clean_iss_data, 9
clean_iss_data(), *58, 59*
clean_lab_data, 10
clean_lab_data(), *74*
compress_png, 11
create_60_day_export, 12
create_afp_export, 13
create_emergence_group_gif, 13
create_npafp_export, 15
create_pop_check_export, 16
create_pot_comp_clust_export, 16
create_stool_adequacy_export, 17
ctry_data_errors, 18

dplyr::distinct(), *94*
duplicate_check, 19

edav_io, 19
edav_io(), *21*
explore_edav, 21
export_kpi_table, 21
extract_country_data, 23
extract_country_data(), *6–8*, *16*, *18*, *24*, *26, 27*, *36*, *38, 39*, *46*, *48, 49*, *51, 52*, *54*, *56*, *77*, *82, 83*, *87*

f.color.schemes, 23
f.ev.rate.01, 24
f.expand.bbox, 25
f.metadata.tag, 26
f.npafp.rate.01, 26
f.npafp.rate.01(), *15*, *75, 76*, *78*, *85*
f.plot.looks, 28
f.stool.ad.01, 28
f.stool.ad.01(), *17*, *36*, *56*, *78*, *82–86*
f.timely.detection.01, 30
fix_ctry_data_missing_guids, 31
freeze_dr_data, 32

generate_60_day_tab, 33
generate_60_day_table_data, *12*, 33
generate_60_day_table_data(), *33*, *79*, *85*
generate_ad_final_col, 36
generate_adhoc_map, 34
generate_adhoc_map(), *24*, *110*
generate_afp_by_month_summary, 37
generate_afp_by_month_summary(), *40*, *86*
generate_afp_case_map, 38
generate_afp_epicurve, 39
generate_afp_prov_year, 39
generate_c1_rollup, 40
generate_c1_table, 41
generate_c1_table(), *22*, *41*, *61*, *63*, *65*
generate_c2_table, 42
generate_c2_table(), *22*, *63, 64*, *66*
generate_c3_rollup, 43
generate_c3_table, 44
generate_c3_table(), *22*, *44*, *62*
generate_c4_table, 45
generate_c4_table(), *22*
generate_case_num_dose_g, 46
generate_ctry_timeliness_graph, 47
generate_dist_pop_map, 48
generate_dr_ppt, 49
generate_dr_ppt2, 51
generate_dr_ppt2(), *49*
generate_es_det_map, 52
generate_es_site_det, 53
generate_es_site_det(), *24*
generate_es_tab, 54
generate_es_timely, 55
generate_inad_tab, 56
generate_int_data, 57
generate_int_data(), *47*
generate_iss_barplot, 58
generate_iss_map, 59
generate_kpi_barchart, 60

generate_kpi_ev_map, 61
generate_kpi_evdetect_bar, 60
generate_kpi_map, 62
generate_kpi_npafp_bar, 63
generate_kpi_npafp_map, 64
generate_kpi_stool_map, 66
generate_kpi_stoolad_bar, 65
generate_kpi_tile, 67
generate_kpi_violin, 68
generate_lab_culture_violin, 68
generate_lab_itd_violin, 70
generate_lab_itdres_seqres_violin, 69
generate_lab_seqres_violin, 71
generate_lab_seqship_violin, 72
generate_lab_timeliness, 73
generate_lab_timeliness(), *57*
generate_npafp_maps, 74
generate_npafp_maps_dist, 75
generate_pop_map, 77
generate_pop_tab, 78
generate_potentially_compatibles_cluster, 79
generate_potentially_compatibles_cluster(), *16, 17*
generate_prov_timeliness_graph, 80
generate_sg_priority_map, 81
generate_stool_ad_maps, 81
generate_stool_ad_maps_dist, 83
generate_stool_data, 84
generate_stool_data(), *13*, *34*
generate_surv_ind_tab, 85
generate_timeliness_maps, 87
generate_timely_det_violin, 88
generate_timely_ship_violin, 89
get_all_polio_data, 90
get_all_polio_data(), *14*, *19*, *23, 24*, *26, 27*, *35, 36*, *42*, *61*, *63*, *65*, *88*, *110*
get_azure_storage_connection, 91
get_azure_storage_connection(), *20, 21*, *111, 112*
get_cdc_childvaxview_data, 92
get_constant, 93
get_ctry_abbrev, 94
get_diff_cols, 94
get_lab_date_col_missingness, 95
get_lab_locs, 95
get_lab_locs(), *30*, *42–44*
get_ppt_template, 96
get_region, 97
get_vpd_data, 97
get_vpd_missingness, 98
get_vpd_vars, 99

ggplot2::facet_grid(), *60*
ggplot2::facet_wrap(), *60*

init_dr, 99
init_dr(), *7, 8*, *16*, *18*, *24*, *27*, *32*, *38, 39*, *46*, *48, 49*, *51, 52*, *54*, *56*, *77*, *82, 83*, *87*, *102*, *107*
init_kpi, 101
init_kpi(), *22*, *62*, *64*, *66*, *81*
iss_data_errors, 101

lab_data_errors, 102
load_clean_ctry_sp, 103
load_clean_ctry_sp(), *38*, *48*, *62*, *64*, *66*, *75*, *77*
load_clean_dist_sp, 104
load_clean_dist_sp(), *48*, *64*, *66*
load_clean_prov_sp, 106
load_clean_prov_sp(), *38*, *48*, *75*, *77*
load_iss_data, 107
load_lab_data, 108
lubridate::period(), *5*

readr::read_delim(), *20*, *111*
readxl::read_excel(), *20*, *111*

send_outlook_email, 108
send_teams_message, 109
set_emergence_colors, 110
set_emergence_colors(), *35*
sf::st_as_sfc(), *25*
sf::st_bbox(), *25*
sirfunctions_io, 111
Sys.getenv(), *99*

test_EDAV_connection, 112

update_polio_data, 113
upload_dr_to_github, 113
upload_to_sharepoint, 114