

# Package ‘sirfunctions’

March 11, 2025

**Title** Key Functions to Analyze Global Polio Surveillance Data

**Version** 1.3.1

**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)

**Suggests** blastula,  
forcats,  
ggspatial,  
git2r,  
officer,  
ggpubr,  
ggrepel,

```

prettyunits,
purrr,
httr,
janitor,
datasets,
readxl,
tibble,
tools,
knitr,
magick,
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

check_afp_guid_etry_data . . . . .	4
check_missing_rows . . . . .	5
clean_etry_data . . . . .	5
clean_es_data . . . . .	6
clean_iss_data . . . . .	7
clean_lab_data . . . . .	8
compress_png . . . . .	9
create_60_day_export . . . . .	10
create_afp_export . . . . .	10
create_emergence_group_gif . . . . .	11
create_npafp_export . . . . .	12
create_pop_check_export . . . . .	13
create_pot_comp_clust_export . . . . .	14
create_stool_adequacy_export . . . . .	15
etry_data_errors . . . . .	16
duplicate_check . . . . .	16
edav_io . . . . .	17
explore_edav . . . . .	18
extract_country_data . . . . .	19
f.color.schemes . . . . .	19
f.ev.rate.01 . . . . .	20
f.expand.bbox . . . . .	21
f.metadata.tag . . . . .	21
f.npafp.rate.01 . . . . .	22
f.plot.looks . . . . .	23

f.stool.ad.01 . . . . .	24
f.timely.detection.01 . . . . .	25
fix_ctype_data_missing_guids . . . . .	26
freeze_dr_data . . . . .	27
generate_60_day_tab . . . . .	28
generate_60_day_table_data . . . . .	29
generate_adhoc_map . . . . .	29
generate_ad_final_col . . . . .	31
generate_afp_by_month_summary . . . . .	32
generate_afp_case_map . . . . .	33
generate_afp_epicurve . . . . .	34
generate_afp_prov_year . . . . .	34
generate_case_num_dose_g . . . . .	35
generate_ctype_timeliness_graph . . . . .	36
generate_dist_pop_map . . . . .	37
generate_dr_ppt . . . . .	38
generate_dr_ppt2 . . . . .	40
generate_es_det_map . . . . .	41
generate_es_site_det . . . . .	42
generate_es_tab . . . . .	43
generate_es_timely . . . . .	44
generate_inad_tab . . . . .	45
generate_int_data . . . . .	46
generate_iss_barplot . . . . .	47
generate_iss_map . . . . .	48
generate_lab_timeliness . . . . .	49
generate_npafp_maps . . . . .	50
generate_npafp_maps_dist . . . . .	51
generate_pop_map . . . . .	52
generate_pop_tab . . . . .	53
generate_potentially_compatibles_cluster . . . . .	54
generate_prov_timeliness_graph . . . . .	55
generate_stool_ad_maps . . . . .	56
generate_stool_ad_maps_dist . . . . .	57
generate_stool_data . . . . .	58
generate_surv_ind_tab . . . . .	60
generate_timeliness_maps . . . . .	61
get_all_polio_data . . . . .	62
get_azure_storage_connection . . . . .	63
get_constant . . . . .	64
get_diff_cols . . . . .	65
get_lab_locs . . . . .	65
get_ppt_template . . . . .	66
get_region . . . . .	66
init_dr . . . . .	67
iss_data_errors . . . . .	68
lab_data_errors . . . . .	69
load_clean_ctype_sp . . . . .	70
load_clean_dist_sp . . . . .	71
load_clean_prov_sp . . . . .	72
load_iss_data . . . . .	73
load_lab_data . . . . .	74

send_outlook_email . . . . .	75
send_teams_message . . . . .	75
set_emergence_colors . . . . .	76
test_EDAV_connection . . . . .	77
upload_dr_to_github . . . . .	78
upload_to_sharepoint . . . . .	78

<b>Index</b>	<b>80</b>
--------------	-----------

---

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 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

```
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)
```

---

check_missing_rows	<i>Check for rows with NA values</i>
--------------------	--------------------------------------

---

**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**

```
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
missing <- check_missing_rows(raw.data$afp, "age.months", c("place.admin.0", "yronset"))
```

---

clean_ctype_data	<i>Cleans and adds additional age and dosage number columns to the AFP linelist</i>
------------------	---

---

**Description**

The function does additional cleaning of the ctype.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_ctype_data(ctype.data)
```

**Arguments**

ctype.data	list Large list containing polio country data. This is the output of <a href="#">extract_country_data()</a> or <a href="#">init_dr()</a> .
------------	--

**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	<i>Clean environmental surveillance data</i>
---------------	--

---

## 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	<b>[Deprecated]</b> 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

```
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)
```

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 <b>[Deprecated]</b> 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	<i>Clean lab data</i>
----------------	-----------------------

---

**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 <a href="#">get_all_polio_data()</a> .
dist	sf	Shapefile of all districts.
ctry	sf	Shapefile of all countries.
include_env	bool	To include environmental detections in analysis. Defaults to TRUE.
cumulative	bool	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	<i>Exports NPAFP indicator data summary tables</i>
---------------------	--

---

**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\\_compatibles\\_cluster\(\)](#).

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	<i>Check data quality errors from the country data</i>
------------------	--

---

### 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 <a href="#">extract_country_data()</a> or <a href="#">init_dr()</a> .
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	<i>Assess duplicates in the get_all_polio_data() output</i>
-----------------	---

---

### 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

```
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
raw.data <- duplicate_check(raw.data)
```

---

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

<code>io</code>	str The type of operation to perform in EDAV. <ul style="list-style-type: none"> <li>• "read" Read a file from EDAV, must be an rds, csv, or rda.</li> <li>• "write" Write a file from EDAV, must be an rds, csv or rda.</li> <li>• "exists.dir" Returns a boolean after checking to see if a folder exists.</li> <li>• "exists.file" Returns a boolean after checking to see if a file exists.</li> <li>• "create" Creates a folder and all preceding folders.</li> <li>• "list" Returns a tibble with all objects in a folder.</li> <li>• "upload" Moves a file of any type to EDAV.</li> </ul>
<code>default_dir</code>	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 <code>file_loc</code> .
<code>file_loc</code>	str Location to "read", "write", "exists.dir", "exists.file", "create" or "list", can include the information in <code>default_dir</code> if you set that parameter to NULL.
<code>obj</code>	rojb Object to be saved, needed for "write". Defaults to NULL.
<code>azcontainer</code>	Azure container object returned from <code>get_azure_storage_connection()</code> .
<code>force_delete</code>	bool Use delete io without verification in the command line.
<code>local_path</code>	str Local file pathway to upload a file to EDAV. Default is NULL. This parameter is only required when passing "upload" in the <code>io</code> parameter.
<code>...</code>	Optional parameters that work with <code>readr::read_delim()</code> .

**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
edav_io("write", file_loc = "Data/test", obj = df) # saves df to the test folder in EDAV
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"))
```

**Arguments**

path                      str Path to start at initially.

**Value**

tibble Data from the EDAV environment.

**Examples**

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

## End(Not run)
```

---

extract_country_data	<i>Extract country specific information from raw polio data</i>
----------------------	---

---

**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 <code>get_all_polio_data()</code> .

**Value**

Named list with country specific datasets.

**Examples**

```
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
ctry.data <- extract_country_data("nigeria", raw.data)
```

---

f.color.schemes	<i>Utility function for colors</i>
-----------------	------------------------------------

---

**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: <ul style="list-style-type: none"> <li>• "epicurve" Mapped to different cdc.classification.all2 values.</li> <li>• "para.case" A subset of "epicurve" representing paralytic cases.</li> <li>• "afp.prov" Mapped for case counts at the province level.</li> <li>• "afp.dist" Mapped for case counts at the province level.</li> <li>• "pot.comp" Colors for categories of compatibles and potentially compatibles.</li> <li>• "silence" Colors to use to map silent populations.</li> <li>• "silence.v2" Colors to use to map silent populations.</li> <li>• "cases" Values to map case type.</li> </ul>
------	--

- "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()`.

**Value**

Named list with color sets.

**Examples**

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

---

f.ev.rate.01	<i>Calculate EV detection rate function</i>
--------------	---

---

**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 enterovirus in an ES sample. This is ctry.data\$es of <code>extract_country_data()</code> or <code>init_dr()</code> , or raw.data\$es of <code>get_all_polio_data()</code> .
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**

```
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")
```

---

f.expand.bbox	<i>Expand bounding box</i>
---------------	----------------------------

---

### Description

Sourced from [https://rdr.io/github/Chrisjb/basemapR/src/R/expand\\_bbox.R](https://rdr.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 <code>sf::st_bbox()</code> .
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

```
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)
```

---

f.metadata.tag	<i>Function to add metadata tags to figures and tables</i>
----------------	--

---

### 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 <a href="#">get_all_polio_data()</a> or <a href="#">extract_country_data()</a> .
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**

```
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
```

---

f.npafp.rate.01	<i>Calculate non-polio AFP rate</i>
-----------------	-------------------------------------

---

**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 <a href="#">extract_country_data()</a> or <a href="#">init_dr()</a> or raw.data\$afp of <a href="#">get_all_polio_data()</a> .
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 <a href="#">get_all_polio_data()</a> and <a href="#">extract_country_data()</a> .

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. <ul style="list-style-type: none"> <li>• "prov" Province level.</li> <li>• "dist" District level.</li> <li>• "ctry" Country level.</li> </ul>
pending	bool Should cases classified as PENDING or LAB PENDING be included in calculations? Default TRUE.
missing_agemonths	bool Should cases with NA values for age.months be included? Default FALSE.
rolling	bool Should the analysis be performed on a rolling bases? Default FALSE.
sp_continuity_validation	bool 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.

---

f.plot.looks	<i>Set plot looks</i>
--------------	-----------------------

---

**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: <ul style="list-style-type: none"> <li>• "02"</li> <li>• "epicurve"</li> <li>• "geomtile"</li> <li>• "gpln_type1"</li> <li>• "gpln_type2"</li> </ul>
------	---

**Value**

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

## Examples

```
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
```

---

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. <ul style="list-style-type: none"> <li>• "prov" Province level.</li> <li>• "dist" District level.</li> <li>• "ctry" Country level.</li> </ul>
missing	str How to treat missing data. Valid values are: "good", "bad", "remove". Defaults to "good". When calculating the adequacy.final column: <ul style="list-style-type: none"> <li>• "good" uses adequacy.03</li> </ul>



- "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	bool Should data be analyzed on a rolling bases? Defaults to FALSE.
sp_continuity_validation	bool 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**

```
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
)
```

---

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	bool 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

```
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"
)
```

---

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 <a href="#">check_afp_guid_ctry_data</a>
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**

```
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"
)
```

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	<i>60-day follow up table</i>
---------------------	-------------------------------

---

**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 <a href="#">generate_stool_data()</a> .
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

```
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")
```

---

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 <a href="#">get_all_polio_data()</a> . 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	bool Whether to include VPDV in maps. Default TRUE.
new_detect	bool 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: <ul style="list-style-type: none"> <li>• "ALL": All regions</li> <li>• "YES": Recent Detections - &lt;13 months</li> </ul>
owner	str Who produced the map. Defaults to "CDC-GID-PEB".
new_detect_expand	bool 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 <a href="#">set_emergence_colors()</a> .

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: <ul style="list-style-type: none"> <li>• "full_slide"</li> <li>• "soco_slide"</li> <li>• "half_slide"</li> </ul> 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	<b>[Deprecated]</b> 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**

```
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
)
```



---

```
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

<code>afp.all</code>	sf AFP linelist containing point geometry. This is <code>ctry.data\$afp.all</code> , which is an output of either <a href="#">extract_country_data()</a> and <a href="#">init_dr()</a> .
<code>ctry.shape</code>	sf Country shapefile in long format.
<code>prov.shape</code>	sf Province shapefile in long format.
<code>start_date</code>	str Start date of analysis.
<code>end_date</code>	str End date of analysis. Default is today's date.
<code>output_path</code>	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 <a href="#">extract_country_data()</a> or <a href="#">init_dr()</a> .
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 <a href="#">generate_afp_by_month_summary()</a> .
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\_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**

etry.data	list A large list containing polio data of country. This is the output of <code>extract_country_data()</code> or <code>init_dr()</code> . Note that etry_data needs to be cleaned via <code>clean_etry_data()</code> 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:
etry.data <- init_dr("algeria")
etry.data <- clean_etry_data(etry.data)
generate_case_num_dose_g(etry.data, "2021-01-01", "2023-12-31")

## End(Not run)
```

---

generate\_etry\_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_etry_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 <b>[Deprecated]</b> 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 <a href="#">extract_country_data()</a> or <a href="#">init_dr()</a> .
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\\_ctype\\_sp\(\)](#), [load\\_clean\\_prov\\_sp\(\)](#), [load\\_clean\\_dist\\_sp\(\)](#)

**Examples**

```
## Not run:
ctype.data <- init_dr("algeria")
ctype.shape <- load_clean_ctype_sp(ctype_name = "ALGERIA", type = "long")
prov.shape <- load_clean_prov_sp(ctype_name = "ALGERIA", type = "long")
dist.shape <- load_clean_dist_sp(ctype_name = "ALGERIA", type = "long")
generate_pop_map(ctype.data, ctype.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,
  ctype.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 Path to the PowerPoint template.
ctry.data	list List containing polio data for a country. Either the output of <a href="#">extract_country_data()</a> or <a href="#">init_dr()</a> .
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 <a href="#">extract_country_data()</a> or <a href="#">init_dr()</a> .
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	<i>ES detection map</i>
---------------------	-------------------------

---

### 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 <code>ctry.data\$es</code> , which is part of the outputs of <code>extract_country_data()</code> and <code>init_dr()</code> .
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	<b>[Deprecated]</b> tibble Please pass the output of <code>clean_es_data()</code> into <code>es.data</code> instead. This parameter 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 <code>clean_es_data()</code> or a cleaned <code>ctry.data\$es</code> .
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	<b>[Deprecated]</b> Please pass the SIA data directly to <code>sia.data</code> instead of a list containing it.
es.data.long	<b>[Deprecated]</b> 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	<i>ES surveillance sites summary table</i>
-----------------	--

---

**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 time-liness 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)
```

---

<code>generate_es_timely</code>	<i>ES timeliness scatterplot</i>
---------------------------------	----------------------------------

---

**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 <a href="#">extract_country_data()</a> or <a href="#">init_dr()</a> .
cstool	tibble Stool adequacy at the country level. This is the output of <a href="#">f.stool.ad.01()</a> .
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	<i>Generate a summary table for sample timeliness intervals</i>
-------------------	---

---

## 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 <a href="#">generate_lab_timeliness()</a> .
ctry.data	list <b>[Deprecated]</b>
spatial.scale	str <b>[Deprecated]</b> Renamed in favor of spatial_scale.
lab.data	tibble <b>[Deprecated]</b> 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 <a href="#">clean_iss_data()</a> .
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 <a href="#">clean_iss_data()</a> .
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\_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)
```

## Arguments

lab_data	tibble Lab data. Ensure that this lab data is cleaned using <a href="#">clean_lab_data()</a> 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.

## 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	<i>Maps of NPAFP rates by province and year</i>
---------------------	---

---

## 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 <a href="#">f.npafp.rate.01()</a> 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 <code>f.npafp.rate.01()</code> 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 <a href="#">extract_country_data()</a> or <a href="#">init_dr()</a> .
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\\_ctype\\_sp\(\)](#), [load\\_clean\\_prov\\_sp\(\)](#)

**Examples**

```
## Not run:
ctry.data <- init_dr("algeria")
ctry.shape <- load_clean_ctype_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	<i>Summary table of indicators at the province level</i>
------------------	--

---

**Description**

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

**Usage**

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

**Arguments**

npafp	tibble NPAFP table. Output of <a href="#">f.npafp.rate.01()</a> at the province level.
pstool	tibble Stool adequacy at province level. Output of <a href="#">f.stool.ad.01()</a> at the province level.
start_date	str Start date of analysis.
end_date	str End date of analysis.
prov.case.ind	tibble <b>[Deprecated]</b> Deprecated in favor of the more informative npafp 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 bool Add column for clusters? Default to FALSE.

**Value**

tibble A summary table of cases.

**Examples**

```

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
)

```

---

```
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_etry_timeliness_graph(int.data.prov)

## 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 <a href="#">extract_country_data()</a> or <a href="#">init_dr()</a> .
pstool	tibble Stool adequacy table at province level. This is the output of <a href="#">f.stool.ad.01()</a> 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 <a href="#">extract_country_data()</a> or <a href="#">init_dr()</a> .
dstool	tibble District stool adequacy table. This is the output of <a href="#">f.stool.ad.01()</a> 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	<i>Generate stool adequacy columns in the AFP dataset</i>
---------------------	---

---

### 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 <code>ctry.data\$afp.all.2</code>
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 <code>adequacy.final</code> column: <ul style="list-style-type: none"> <li>• "good" uses <code>adequacy.03</code></li> <li>• "bad" uses <code>adequacy.01</code></li> <li>• "exclude" uses <code>adequacy.02</code></li> </ul>
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**

```
raw.data <- get_all_polio_data(attach.spatial.data = FALSE)
stool.data <- generate_stool_data(raw.data$afp, "2021-01-01", "2023-12-31")
```

---

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 <a href="#">f.npafp.rate.01()</a> calculated at the country level.
dist.extract	tibble District NPAFP rate. Output of <a href="#">f.npafp.rate.01()</a> calculated at the district level.
cstool	tibble Country stool adequacy. Output of <a href="#">f.stool.ad.01()</a> calculated at the country level.
dstool	tibble District stool adequacy. Output of <a href="#">f.stool.ad.01()</a> calculated at the district level.
afp.case	tibble AFP case counts. Output of <a href="#">generate_afp_by_month_summary()</a> 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 <a href="#">extract_country_data()</a> or <a href="#">init_dr()</a> .
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	bool 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)
```

---

get_all_polio_data	<i>Retrieve all pre-processed polio data</i>
--------------------	--

---

**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 from the data will recreate the "small" dataset when new information is available 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",
  folder = "GID/PEB/SIR/Data/",
  force.new.run = F,
  recreate.static.files = F,
  attach.spatial.data = T
)
```

**Arguments**

size	str Size of data to download. Defaults to "small".
	<ul style="list-style-type: none"> <li>• "small": Data from 2019-present.</li> <li>• "medium": Data from 2016-present.</li> <li>• "large": Data from 2001-present.</li> </ul>
folder	str Location of the CDC pre-processed endpoint, defaults to "GID/PEB/SIR/Data".
force.new.run	bool Default FALSE, if TRUE will run recent data and cache.
recreate.static.files	bool Default FALSE, if TRUE will run all data and cache.
attach.spatial.data	bool Default TRUE, adds spatial data to downloaded object.

**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

## End(Not run)
raw.data <- get_all_polio_data(size = "small", attach.spatial.data = FALSE) # exclude spatial data
```

---

```
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"
)
```

**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.

**Value**

Azure container verification

**Examples**

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

---

get_constant	<i>Obtain a constant variable used in sirfunctions</i>
--------------	--

---

**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)
```

**Arguments**

constant_name	str Name of the constant. Valid values include: <ul style="list-style-type: none"> <li>• "DEFAULT_EDAV_FOLDER"</li> <li>• "CTRY_RISK_CAT"</li> <li>• "LAB_LOCATIONS"</li> <li>• "DR_TEMPLATE"</li> <li>• "SIRFUNCTIONS_GITHUB_TREE"</li> <li>• "AFRO_LAB_DATA"</li> <li>• "EMRO_LAB_DATA"</li> </ul>
---------------	--

**Value**

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

**Examples**

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



---

get_diff_cols	<i>Get the columns where records differ in a group</i>
---------------	--

---

### 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_locs	<i>Table of information regarding testing labs in each country</i>
--------------	--

---

### 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**

```
ctry.seq <- get_lab_locs()
```

---

get_ppt_template	<i>Get path of the PowerPoint template</i>
------------------	--

---

**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	<i>Determines whether lab data is EMRO or AFRO</i>
------------	--

---

**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**

```
get_region("algeria")
```

---

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
)
```

**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	bool 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	bool Whether to source local functions or use sirfunctions. 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)
```

---

iss_data_errors	<i>Checks for errors in the ISS data</i>
-----------------	--

---

### 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 <code>init_dr()</code> . Otherwise, users should manually specify the error folder.
ctry.data	list <b>[Deprecated]</b> 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	<i>Generate a log of potential errors in the lab data</i>
-----------------	---

---

**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 <b>[Deprecated]</b>

**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"
ctype.data <- init_dr("algeria", lab_data_path = lab_path)
lab_data_errors(ctype.data$lab.data, ctype.data$afp.data)

## End(Not run)
```

---

load_clean_ctype_sp	<i>Download country geographic data</i>
---------------------	---

---

## Description

Pulls country shapefiles directly from the geodatabase.

## Usage

```
load_clean_ctype_sp(
  azcontainer = suppressMessages(get_azure_storage_connection()),
  fp = "GID/PEB/SIR/Data/spatial/global.ctype.rds",
  ctype_guid = NULL,
  ctype_name = NULL,
  end_year = lubridate::year(Sys.Date()),
  st_year = 2000,
  data_only = FALSE,
  type = NULL,
  version = "standard",
  end.year = lifecycle::deprecated(),
  st.year = lifecycle::deprecated(),
  data.only = lifecycle::deprecated()
)
```

## Arguments

azcontainer	Azure validated container object
fp	str Location of geodatabase.
ctype_guid	str array Array of all country GUIDS that you want to pull.
ctype_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	bool 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. <ul style="list-style-type: none"> <li>• "long" Return a dataset for every year group.</li> <li>• NULL Return a dataset only with unique GUIDs and when they were active.</li> </ul>
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.

end.year      int **[Deprecated]** Renamed in favor of end\_year.

st.year        int **[Deprecated]** Renamed in favor of st\_year.

data.only      bool **[Deprecated]** Renamed in favor of data\_only.

### Value

tibble or sf Dataframe containing spatial data.

### Examples

```
ctry <- load_clean_ctry_sp(ctry_name = "ALGERIA")
ctry.long <- load_clean_ctry_sp(ctry_name = "ALGERIA", type = "long")
```

---

load_clean_dist_sp	<i>Download district geographic data</i>
--------------------	--

---

### 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",
  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	bool 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. <ul style="list-style-type: none"> <li>• "long" Return a dataset for every year group.</li> <li>• NULL Return a dataset only with unique GUIDs and when they were active.</li> </ul>
version	str Specify whether to return standard shapefiles or new shapefiles still under evaluation/development. Default is "standard". <ul style="list-style-type: none"> <li>• "standard" Standard shapefiles.</li> <li>• "dev" New shapefiles still under evaluation/development.</li> </ul>
end.year	int <b>[Deprecated]</b> Renamed in favor of end_year.
st.year	int <b>[Deprecated]</b> Renamed in favor of st_year.
data.only	bool <b>[Deprecated]</b> Renamed in favor of data_only.

### Value

tibble or sf Dataframe containing spatial data.

### Examples

```
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")
```

---

load_clean_prov_sp	<i>Download province geographic data</i>
--------------------	--

---

### 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",
  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	bool 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. <ul style="list-style-type: none"> <li>• "long" Return a dataset for every year group.</li> <li>• NULL Return a dataset only with unique GUIDs and when they were active.</li> </ul>
version	str Specify whether to return standard shapefiles or new shapefiles still under evaluation/development. Default is "standard". <ul style="list-style-type: none"> <li>• "standard" Standard shapefiles.</li> <li>• "dev" New shapefiles still under evaluation/development.</li> </ul>
end.year	int <b>[Deprecated]</b> Renamed in favor of end_year.
st.year	int <b>[Deprecated]</b> Renamed in favor of st_year.
data.only	bool <b>[Deprecated]</b> 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	<i>Read ISS/eSURV data</i>
---------------	----------------------------

---

**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	<i>Function to load the raw lab data</i>
---------------	--

---

**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	<i>Send email through Outlook</i>
--------------------	-----------------------------------

---

**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	<i>Send a message on Microsoft Teams</i>
--------------------	--

---

**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 <a href="#">get_all_polio_data()</a> .
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	bool 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)
```

---

test_EDAV_connection	<i>Test network connection to the EDAV</i>
----------------------	--

---

**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 <a href="#">get_azure_storage_connection()</a> .
folder	str Location of folder in the EDAV environment that you want to download and upload data from.
return_list	bool 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**

```
test_EDAV_connection()
```

---

upload_dr_to_github	<i>Upload desk review script to the sg-desk-reviews GitHub repository</i>
---------------------	---

---

### 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	<i>Upload file to Sharepoint</i>
----------------------	----------------------------------

---

### 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

check\_afp\_guid\_ctry\_data, 4  
check\_afp\_guid\_ctry\_data(), 26  
check\_missing\_rows, 5  
clean\_ctry\_data, 5  
clean\_ctry\_data(), 36, 44, 46  
clean\_es\_data, 6  
clean\_es\_data(), 42, 43  
clean\_iss\_data, 7  
clean\_iss\_data(), 47, 48  
clean\_lab\_data, 8  
clean\_lab\_data(), 49  
compress\_png, 9  
create\_60\_day\_export, 10  
create\_afp\_export, 10  
create\_emergence\_group\_gif, 11  
create\_npafp\_export, 12  
create\_pop\_check\_export, 13  
create\_pot\_comp\_clust\_export, 14  
create\_stool\_adequacy\_export, 15  
ctry\_data\_errors, 16  
  
dplyr::distinct(), 65  
duplicate\_check, 16  
  
edav\_io, 17  
explore\_edav, 18  
extract\_country\_data, 19  
extract\_country\_data(), 4, 5, 14, 16, 20, 22, 31, 33, 34, 36, 37, 39, 40, 42, 44, 45, 52, 56, 57, 62  
  
f.color.schemes, 19  
f.ev.rate.01, 20  
f.expand.bbox, 21  
f.metadata.tag, 21  
f.npafp.rate.01, 22  
f.npafp.rate.01(), 12, 50, 51, 53, 60  
f.plot.looks, 23  
f.stool.ad.01, 24  
f.stool.ad.01(), 15, 31, 45, 53, 56–60  
f.timely.detection.01, 25  
fix\_ctry\_data\_missing\_guids, 26  
freeze\_dr\_data, 27  
  
generate\_60\_day\_tab, 28  
  
generate\_60\_day\_table\_data, 10, 29  
generate\_60\_day\_table\_data(), 28, 54, 59  
generate\_ad\_final\_col, 31  
generate\_adhoc\_map, 29  
generate\_adhoc\_map(), 20, 76  
generate\_afp\_by\_month\_summary, 32  
generate\_afp\_by\_month\_summary(), 35, 60  
generate\_afp\_case\_map, 33  
generate\_afp\_epicurve, 34  
generate\_afp\_prov\_year, 34  
generate\_case\_num\_dose\_g, 35  
generate\_ctry\_timeliness\_graph, 36  
generate\_dist\_pop\_map, 37  
generate\_dr\_ppt, 38  
generate\_dr\_ppt2, 40  
generate\_dr\_ppt2(), 38  
generate\_es\_det\_map, 41  
generate\_es\_site\_det, 42  
generate\_es\_site\_det(), 20  
generate\_es\_tab, 43  
generate\_es\_timely, 44  
generate\_inad\_tab, 45  
generate\_int\_data, 46  
generate\_int\_data(), 37  
generate\_iss\_barplot, 47  
generate\_iss\_map, 48  
generate\_lab\_timeliness, 49  
generate\_lab\_timeliness(), 46  
generate\_npafp\_maps, 50  
generate\_npafp\_maps\_dist, 51  
generate\_pop\_map, 52  
generate\_pop\_tab, 53  
generate\_potentially\_compatibles\_cluster, 54  
generate\_potentially\_compatibles\_cluster(), 14  
generate\_prov\_timeliness\_graph, 55  
generate\_stool\_ad\_maps, 56  
generate\_stool\_ad\_maps\_dist, 57  
generate\_stool\_data, 58  
generate\_stool\_data(), 10, 11, 29  
generate\_surv\_ind\_tab, 60  
generate\_timeliness\_maps, 61



get\_all\_polio\_data, [62](#)  
get\_all\_polio\_data(), [12](#), [17](#), [19](#), [20](#), [22](#),  
[30](#), [31](#), [76](#)  
get\_azure\_storage\_connection, [63](#)  
get\_azure\_storage\_connection(), [17](#), [77](#)  
get\_constant, [64](#)  
get\_diff\_cols, [65](#)  
get\_lab\_locs, [65](#)  
get\_lab\_locs(), [26](#)  
get\_ppt\_template, [66](#)  
get\_region, [66](#)  
  
init\_dr, [67](#)  
init\_dr(), [4](#), [5](#), [14](#), [16](#), [20](#), [22](#), [27](#), [33](#), [34](#), [36](#),  
[37](#), [39](#), [40](#), [42](#), [44](#), [45](#), [52](#), [56](#), [57](#), [62](#),  
[68](#), [73](#)  
iss\_data\_errors, [68](#)  
  
lab\_data\_errors, [69](#)  
load\_clean\_ctry\_sp, [70](#)  
load\_clean\_ctry\_sp(), [33](#), [38](#), [50](#), [53](#)  
load\_clean\_dist\_sp, [71](#)  
load\_clean\_dist\_sp(), [38](#)  
load\_clean\_prov\_sp, [72](#)  
load\_clean\_prov\_sp(), [33](#), [38](#), [50](#), [53](#)  
load\_iss\_data, [73](#)  
load\_lab\_data, [74](#)  
  
readr::read\_delim(), [17](#)  
  
send\_outlook\_email, [75](#)  
send\_teams\_message, [75](#)  
set\_emergence\_colors, [76](#)  
set\_emergence\_colors(), [30](#)  
sf::st\_as\_sfc(), [21](#)  
sf::st\_bbox(), [21](#)  
Sys.getenv(), [67](#)  
  
test\_EDAV\_connection, [77](#)  
  
upload\_dr\_to\_github, [78](#)  
upload\_to\_sharepoint, [78](#)