Package 'OKplan'

December 12, 2024

Title Tools to facilitate the Planning of the annual Surveillance Programmes

Version 0.7.1.9001

Date 2024-##-##

Description Provide tools to facilitate the planning of the annual surveillance programmes. The main focus is tools for generating standardized lists for NFSA.

URL https://github.com/NorwegianVeterinaryInstitute/OKplan

BugReports https://github.com/NorwegianVeterinaryInstitute/OKplan/issues

Depends R (>= 4.1.0)

License BSD_3_clause + file LICENSE

Encoding UTF-8

LazyData true

Imports checkmate,
dplyr,
openxlsx,
rlang,
stats,
NVIbatch (>= 0.4.0),
NVIcheckmate (>= 0.7.3),
NVIdb,
NVIpjsr,
NVIpretty (>= 0.4.0),
OKcheck

Suggests covr,
desc,
devtools,
findInFiles,
knitr,
purrr,
rmarkdown,
testthat,
usethis,
utils,
NVIpackager,
NVIrpackages

Remotes NorwegianVeterinaryInstitute/NVIbatch,
NorwegianVeterinaryInstitute/NVIcheckmate,
NorwegianVeterinaryInstitute/NVIdb,
NorwegianVeterinaryInstitute/NVIpackager,
NorwegianVeterinaryInstitute/NVIpjsr,
NorwegianVeterinaryInstitute/NVIpretty,
NorwegianVeterinaryInstitute/NVIrpackages,
NorwegianVeterinaryInstitute/OKcheck

RoxygenNote 7.3.1

VignetteBuilder knitr

Language en-GB

Contents

`adjust_samples_to_budget` *Adjust the sample size per selected unit*

Description

Adds a new column with an adjusted sample number per selected unit. The total sample size is adjusted to be in accord with the total budgeted sample number.

Usage

```
adjust_samples_to_budget(
  data,
  group = NULL,
  budget,
  sample_to_adjust,
  adjusted_sample = "justert_ant_prover",
  adjust_by = 1
)
```

Arguments

`data`	[`data.frame`] Data including a column with the sample number that should be adjusted.
`group`	[`character`] Vector with group variables. Defaults to `NULL`.
`budget`	[`numeric(1)`	`character(1)`] The total budgeted sample number or a column in data with the budget number of samples (per group).
`sample_to_adjust`	[`character(1)`] The name of the column with the sample number per unit that should be adjusted.
`adjusted_sample`	[`character(1)`] The name of the new column with the adjusted sample number per unit. Defaults to "justert_ant_prover".
`adjust_by`	[`numeric(1)`] The maximum number of samples that a sample can be adjust by. Defaults to 1.

Details

The sample number per unit should first have been estimated, for example can the sample number per abattoir be the total number of samples distributed on the abattoirs in accord with the slaughter volume at each abattoir. Often will rounding errors lead to a difference between the total budgeted sample number and the total estimated sample number. Therefore, the estimated sample number need to be adjusted.

The estimated sample number is first adjusted for the unit with the largest sample number. Thereafter, for the unit with the next largest sample number and so on.

The sample number will often be estimated so that it is a multiplicand multiplied by of a given number (multiplier). For example, if equal number of samples should be taken every month the multiplier can be 12, if the samples are pooled five and five, the multiplier can be 5. If the argument `adjust_by` is given the multiplier, the sample number will be adjusted by the multiplier unless the difference that should be adjusted is less than the multiplier. In that case, the sample number will be adjusted by a number less than the `adjust_by`.

Value

A data frame with a new column with an adjusted sample number.

Author(s)

Petter Hopp Petter.Hopp@vetinst.no

Examples

```
library(OKplan)
# Add data frame with sample number to adjust
x <- as.data.frame(cbind(c(1:10),
                         c(24, 30, 36, 12, 6, 18, 6, 0, 0, 0)))
colnames(x) <- c("id", "sample")
```

```
# Adjust total sample number to budget
x <- adjust_samples_to_budget(data = x,
                              budget = 150,
                              sample_to_adjust = "sample",
                              adjusted_sample = "new_sample",
                              adjust_by = 4)

# Adjust total sample number to budget per group
total_budget <- 60 # same budget for all groups

# Add data frame with sample number to adjust
x <- as.data.frame(cbind(c(1:10),
                         c(rep("x", 5), rep("y", 5)),
                         c(24, 18, 6, 0, 30, 36, 12, 6, 0, 0)))
colnames(x) <- c("id", "xy", "sample")

x2 <- adjust_samples_to_budget(data = x,
                               group = "xy",
                               budget = total_budget,
                               sample_to_adjust = "sample",
                               adjusted_sample = "new_sample",
                               adjust_by = 6)
```

`append_date_generated_line` *Add new last row with the generated date*

Description

Function to add new row with the generated date. Before the date a pretext can be set.

Usage

```
append_date_generated_line(
  data,
  pretext = "Datauttrekket er gjort",
  date = format(Sys.Date(), "%d/%m/%Y")
)
```

Arguments

`data`	[`data.frame`] The data to which an additional row with the generate date should be added.
`pretext`	[`character(1)`] The explaining text before the date value. Defaults to "Datauttrekket ble gjort".
`date`	[`character(1)`] Date for generating the data. Defaults to `base::Sys.Date`.

Details

Two rows are added to the data frame, the first is empty, the second has the generated date in the first column.

Value

data.frame with two additional rows, one empty and one with the generated date in the first column.

Author(s)

Petter Hopp Petter.Hopp@vetinst.no

Examples

```
## Not run:
# Add row with generated date using standard values
gris_virus_slaktegris_utvalg <- append_date_generated_line(gris_virus_slaktegris_utvalg)

## End(Not run)
```

---

append_sum_line *Append row with column sums*

Description

Appends a new row with column sums for selected columns. A pretext can be placed on the row.

Usage

```
append_sum_line(data, column, pretext = "Sum", position = "left")
```

Arguments

data	[data.frame] Data to which a row should be appended.
column	[character] The column names of columns to sum.
pretext	[character(1)] The explaining text before the sum. Defaults to "Sum".
position	[character(1)] The position for the pretext, one of c("first", left", "none"). Defaults to "left".

Details

One row is appended to the data frame. The sum is calculated with na.rm = TRUE.

If a tibble, it is transformed to a data frame to avoid errors if the pretext is to be placed in a numeric variable.

Value

data.frame with an appended row with sums.

Author(s)

Petter Hopp Petter.Hopp@vetinst.no

Examples

```
## Not run:
# Append row with sum
gris_blodprover_slakteri <- append_sum_line(data = gris_blodprover_slakteri,
                                            column = c("ant_prover"),
                                            pretext = "Sum",
                                            position = "first")

## End(Not run)
```

`check_OK_selection` *check_OK_selection is Deprecated*

Description

`check_OK_selection` was deprecated 2022-12-15 to replace it with check_ok_selection with a standardised function name with lower case letters for OK. Use check_ok_selection with the additional parameters `purpose =` and `plan_aar =`.

Usage

```
check_OK_selection(
  data,
  purpose = deparse(substitute(data)),
  plan_aar = as.numeric(format(Sys.Date(), "%Y")) + 1
)
```

Arguments

`data`	The table with data describing the selection for a OK programme.
`purpose`	String with descriptive text to be used in file name and heading of the report. Defaults to name of input data.
`plan_aar`	The year for which the selection is planned. Defaults to next year.

Details

The old help pages can be found at `help("check_OK_selection-deprecated")`. Information on deprecated function can be found at `help("OKplan-deprecated")`.

check_ok_selection *Checks the standard output data frame with the OK selection*

Description

Standard checks by performing descriptive statistics of variables in the standard output data frame with OK selection. `check_ok_selection` is a wrapper for `NVIbatch::output_rendered`.

Usage

```
check_ok_selection(
  input = system.file("templates", "check_ok_selection.Rmd", package = "OKplan"),
  output_file = paste0("Kontroll av okplan for ", purpose, " ", format(Sys.Date(),
    "%Y%m%d"), ".html"),
  output_dir = NULL,
  data = NULL,
  purpose = NULL,
  plan_aar = as.numeric(format(Sys.Date(), "%Y")) + 1,
  display = "browser",
  ...
)
```

Arguments

`input`	[`character(1)`] The path to the rmarkdown document with the checks. Defaults to "check_ok_selection.Rmd" in the `OKplan`.
`output_file`	[`character(1)`] The name of the output file.
`output_dir`	[`character(1)`] The directory to save the output file. Defaults to `NULL`.
`data`	[`data.frame`] The table with data describing the selection for a OK programme.
`purpose`	[`character(1)`] String with descriptive text to be used in file name and heading of the report.
`plan_aar`	[`numeric(1)`] The year for which the selection is planned. Defaults to next year.
`display`	[`logical(1)` | `character(1)`] Set "browser" for the default browser or "viewer" for the R studio viewer. 'TRUE' equals "browser". If 'FALSE', don't display the results file. Defaults to "browser".
`...`	Other arguments to be passed to `NVIbatch::output_rendered`.

Details

Gives descriptive statistics of the OK selection. This should used to check if the number of selected units per category are in accord with the design of the surveillance programme. If any mistakes are found, one must correct in the script that generates the selection.

The check must be performed on a data frame with standardised column names. This is ensured by using column names as defined for "okplan" in `OK_column_standards`.

The default behavior is to display the resulting html-file in the browser. To save the result in a permanent file, use a permanent directory as input to `output_dir`. The resulting file can also be sent by email by using additional arguments, see `NVIbatch::output_rendered`.

If checks are missing, are unnecessary or the headings are too cryptic, please contribute to improve the rmarkdown file "check_ok_selection.Rmd", see `vignette("Contribute_to_OKplan", package = "OKplan")`.

Value

Generates an html-file with the results of the checks to be displayed in the browser.

Author(s)

Petter Hopp Petter.Hopp@vetinst.no

Examples

```
## Not run:
# Checking OK selection data

purpose = "ok_virus_svin"
plan_aar = 2023

# Check
check_OK_selection(data = okplan_svin,
                   purpose = purpose,
                   plan_aar = plan_aar)

## End(Not run)
```

---

get_holiday *Get the non-workdays or workdays*

Description

Get the non-workdays or workdays within one year. The function is intended for use when planning sampling to excluded days or weeks from the sampling plan.

Usage

```
get_holiday(
  year,
  type = "workday",
  exclude_trapped_days = FALSE,
  output = "selected"
)
```

Arguments

year	[integer(1)] Year.
type	[character(1)] The type of non_workday or workday, see details. Defaults to "workday".
exclude_trapped_days	[character|logical(1)] Should trapped days and common days off be excluded from workday?, see details. Defaults to FALSE.
output	[character(1)] The output format of the data frame, see details. Defaults to "selected".

Details

type is used to select the type of non-workday or workday. Valid input are one of c("non_workday", "sat_to_sun", "public_holiday", "workday"). public_holiday are the non-moveable holidays, Easter and Pentacost; sat_to_sun are Saturdays and Sundays; and non_workday are public_holiday and sat_to_sun combined. workday is the opposite of non_workday when exclude_trapped_days = FALSE.

exclude_trapped_days is used to exclude trapped days and other days that many often takes a day off, i.e. the Easter week and the Christmas week. It is only Valid for workday and has no effect on the other types. Input "trapped" or TRUE will exclude trapped days, "easter" will exclude Monday to Wednesday before Thursday and "xmas" will exclude the days in the week of Christmas eve until New years eve.

The output is a data frame with the selected dates and the day_of_week (integer) when output = "selected". When output = "raw" the data frame includes all dates and the additional columns c("non_workday", "sat_to_sun", "public_holiday", "workday", "trapped" and "public"), see below for description.

The output data frame for output = "raw":

Column name	Format	Description
date	date	Date.
day_of_week	integer	Week day number, Monday = 1, Sunday = 7.
sat_to_sun	integer	Saturday and Sunday = 1, otherwise 0.
public_holiday	integer	Public holidays = 1 otherwise = 0.
non_workday	integer	Saturday, Sunday and public holidays = 1, otherwise = 0.
workday	integer	Workday, the opposite of non-workday when exclude_trapped_days = FALSE.
public	character	Easter = "e", Pentacost = "p", non-moveable = "n", otherwise NA.
trapped	character	trapped days (t), Easter week days (e) and/or Xmas week days (x) otherwise NA.

When output %in% c("fhi", "cstime") the data frame is formatted as the table cstime::nor_workdays_by_date created by National Public Health Institute (FHI).

The function is limited to years from 1968, as before 1968 Saturday was a normal workday in Norway. Be aware that Saturday was a normal school day in Norway until and including 1972.

Value

data frame with the selected dates.

Author(s)

Petter Hopp Petter.Hopp@vetinst.no

Examples

```
# Selects the public holidays
 public_holidays <- get_holiday(year = 2024,
                                type = "public_holiday")

# Selects workdays except the trapped days
 workdays <- get_holiday(year = 2024,
                         type = "workday",
                         exclude_trapped_days = TRUE)

# Selects workdays except days in Easter and Christmas week
 workdays <- get_holiday(year = 2024,
                         type = "workday",
                         exclude_trapped_days = c("easter", "xmas"))
```

---

get_tested_herds *Gets herds tested within an surveillance programme*

Description

Gets herds that have been sampled or tested within a surveillance programme for the selected years. You can choose between herds that have submitted samples or herds for which a certain number of samples have been examined for a specific disease.

Usage

```
get_tested_herds(
  eos_table,
  year = as.numeric(format(Sys.Date(), "%Y")) - 1,
  species = NULL,
  production = NULL,
  disease = NULL,
  min_prover = -1,
  tested = FALSE
)
```

Arguments

eos_table	[character(1)] EOS table name.
year	[numeric] One or more years that should be selected. Defaults to previous year.
species	[character] The species that should be selected. Defaults to NULL.
production	[character] The production type(s) for which number of tested samples should be calculated. Defaults to NULL.

disease [character(1)]
The disease for which number of tested samples should be calculated. Defaults to NULL.

min_prover [numeric(1)]
Minimum number of samples that should have been received or examined for the herd to be counted as sampled or tested. No check is performed if equal -1. Defaults to -1.

tested [logical(1)]
If TRUE, the number of tested samples, If FALSE, the number of received samples. Defaults to FALSE.

Details

For programmes having several surveillance streams, it is possible to select surveillance streams based on species and production type. The species and/or production type must be written as in the eos-table. Be aware that species and production type may be missing and for production type it may often be wrong. Therefore, selection by production type and/or species may remove saker that you would want to keep. No selection is performed when the species or production type is missing from the eos-table.

It is possible to define a minimum requirement of number of samples received or tested. For programmes covering several infections, it is necessary to input the disease for which the samples should have been tested. The disease name must be given as it is written in the column name for the number of examined samples.

The eos_table name is the same name as the table name in the EOS data base.

Value

data.frame with tested or sampled locations.

Author(s)

Petter Hopp Petter.Hopp@vetinst.no

---

make_random *Add new column with random numbers*

Description

Adds new column with random numbers. The function is built to be able to use it in piping.

Usage

```
make_random(data, colname = "random", seed = -1, init_seed = FALSE)
```

Arguments

data [data.frame]
Data to which a column with random number should be added.

colname [character(1)]
The name of the new column with the random number. Defaults to "random".

seed	[numeric(1)] The initializing seed. Defaults to -1, see base::set.seed.
init_seed	[logical(1)] Should the seed be initialized. Defaults to FALSE.

Details

To make reproducible random numbers the seed can be initialized with a specific value. The first time the seed is used, set init_seed = TRUE. Thereafter, use init_seed = FALSE if more random numbers are generated in the session to avoid overlapping random numbers.

Value

data.frame with a new column with random numbers.

Author(s)

Petter Hopp Petter.Hopp@vetinst.no

Examples

```
## Not run:
# Add column with random variables
x <- as.data.frame(c(1:10))
seed <- 12345

# Initialize with seed first time used
x <- make_random(x, seed = seed, init_seed = TRUE)

# Do not initialize the seed thereafter to avoid overlapping
x <- make_random(x, seed = seed, init_seed = FALSE)

# If you initialize again you get overlapping seeds
x <- make_random(x, seed = seed, init_seed = TRUE)

## End(Not run)
```

OK_column_standards *Data: Column standards for OK sampling plans.*

Description

A data frame with the column standards for data frames and Excel sheets produced when planning the sampling schemes for the Norwegian surveillance programmes. The raw data for the column standards can be edited in the original Excel table. The code for preparing of the data frame is written in "./data-raw/generate_OK_column_standards". The OK_column_standards is used as input for NVIdb::standardize_columns.

Usage

```
OK_column_standards
```

Format

A data frame with 14 variables:

db the database, either OK_planlegging or OK_kontroll

table_db name of source table / data frame

colname_db name of variable in source table

colname name of variable in working table, usually the same as in source table

label_1_no label (column name) used when writing to Excel, one line header. Generated from label_no and spec_no. For OK-planning usually the same as label_no

label_no label in short form

spec_no specification of label

label_1_en English label (column name) used when writing to Excel, one line header. Generated from label_no and spec_no. For OK-planning usually the same as label_no

label_en English label in short form

spec_en specification of label

colwidth_Excel column width used in Excel tables given in Excel units

colwidth_DT column width used in DT tables, currently not relevant for OK-planning

colclasses column class usedto import character strings correctly

colorder column order when saving standard data and reporting

Source

"colnames_translation_table.xlsx" at NVI's internal net.

Examples

```
standards <- data(OK_column_standards, package = "OKplan")
```

`save_okplan` *Writes the surveillance selection to a standardised file.*

Description

The surveillance selection is written to a standardised file. The file is standardised to include a standard set of columns and have the records in a standard order.

Usage

```
save_okplan(
  data,
  filename,
  filepath,
  sortvar = c("ok_hensiktkode", "ok_driftsformkode", "statuskode",
    "prioritet_av_reserve", "eier_lokalitetnr"),
  ...
)
```

Arguments

data	[data.frame] The sampling plan with the units to be reported.
filename	[character(1)] The name of the csv file including extension.
filepath	[character(1)] The path to the csv file.
sortvar	[character] The sort order for the records in the csv-file. Defaults to c("ok_hensiktkode", "ok_driftsformkode", "statuskode", "prioritet_av_reserve", "eier_lokalitetnr")
...	Other arguments to be passed to utils::write.csv2.

Details

The data is saved as an "okplan" file that will be used the source file when the selection list is generated.

The function uses NVIdb::standardize_columns. to select and order the columns. The formatting information is taken from OK_column_standards.

Value

None. Saves a data frame with the selection in a standard csv file.

Author(s)

Petter Hopp Petter.Hopp@vetinst.no

Examples

```
## Not run:
library(OKplan)
td <- tempdir()
okplan <- as.data.frame(list("ok_hensiktkode" = c("01002", "01002"),
                             "ok_driftsformkode" = c("010202", "010202"),
                             "statuskode" = c(1, 1),
                             "prioritet_av_reserve" = c(NA, NA),
                             "eier_lokalitetnr" = c("1101123456", "1102123456")))
write_okplan(data = okplan,
             filename = "okplan_species_disease",
             filepath = td)

## End(Not run)
```

---

style_sum_line *Style row with "Sum" in an Excel sheet*

Description

Style the font to bold for the row with the text "Sum" in one cell. It is possible to use other text decoration, see openxlsx::createStyle. A line with the text "Sum" or another text as given as input to text will be styled.

Usage

```
style_sum_line(
  workbook = workbook,
  sheet = sheet,
  data,
  text = "Sum",
  text_decoration = "bold",
  ...
)
```

Arguments

workbook	[Workbook] A workbook object containing a worksheet.
sheet	[character(1)] The Excel sheet name.
data	[data.frame] The data that have been exported to the Excel sheet. Used to find column number and row number for the pretext for which the row should be styled.
text	[character(1)] The text in the cell for which the row should be styled. Defaults to "Sum".
text_decoration	[character(1)] The text decoration style that should be used, see openxlsx::createStyle. Defaults to "bold".
...	Other arguments to be passed to openxlsx::addStyle.

Details

The whole line will be styled.

Value

None. One row in the workbook object is styled.

Author(s)

Petter Hopp Petter.Hopp@vetinst.no

---

write_ok_selection_list

Writes the sampling plan with the selection list to an Excel file.

Description

The sampling plan is output to an Excel sheet. The list with selected units is standardised and formatted in order to be submitted without further formatting.

Usage

```
write_ok_selection_list(
  data,
  sheet,
  filename,
  filepath,
  column_standards = OKplan::OK_column_standards,
  calculate_sum = TRUE,
  footnote = NULL,
  footnote_heights = NULL,
  dbsource,
  add_worksheet = FALSE,
  ...
)
```

Arguments

data	[data.frame] The sampling plan with the units to be reported.
sheet	[character(1)] The name of the Excel sheet.
filename	[character(1)] The name of the Excel file.
filepath	[character(1)] The path to the Excel file.
column_standards	[data.frame | list] The column standards to be used as input for NVIdb::standardize_columns when formatting the sampling plan for output, see details. Defaults to OK_column_standards.
calculate_sum	[logical(1)] Should a line with the sum be appended? Defaults to TRUE.
footnote	[character(1)] Footnote to appended? Defaults to NULL.
footnote_heights	[integer(1)] Manually set row height for the footnote. Defaults to NULL.
dbsource	[character(1)] The name of the dbtable in OK_column_standards that should be used for standardising and formatting the sampling plan output.
add_worksheet	[logical(1)] Should a worksheet be added to an existing workbook? Defaults to FALSE.
...	Other arguments to be passed to append_sum_line.

Details

The data must originate from an "okplan" file and the function uses NVIdb::standardize_columns. to select, order, format and style the columns. The formatting information is either taken from OK_column_standards or can be input as a list.

When using OK_column_standards, the formatting information is taken in accord with the argument dbsource. If the formatting needs to be edited, it must be edited in the general source file

for column standards and thereafter, build it into a new version of OKplan. As this can be a tedious process, there is a possibility to input the formatting information as a list.

The list input to column_standards must follow a specific format. It is a list with at least three named vectors:

- colname: a vector of all columns in in the source file that should be included in the Excel report with the selection list.
- collabel: A vector with the labels that should be used in the Excel report.
- colwidth: A vector with the column width that should be used in the Excel report.

In addition one may input:

- colorder: the order of the columns to be used in the Excel report. The default is to use the same order as they are entered in the vectors.
- column_db: input added as a possibility to keep on the same format as OK_column_standards. Not necessary to input.
- table_db: input added as a possibility to keep on the same format as OK_column_standards. Must be the same as dbsource. Not necessary to input.

All vectors must have the same order and the same length.

When calculate_sum is TRUE, a line with the sum will be appended. The default is to calculate the sum of the column "ant_prover". If the sum should be calculated for one or more other columns, you may give the column names as input to the argument column that will be passed to append_sum_line. The sum will only be appended for columns that exist in the data.

When more than one worksheet should be added to a single workbook, use add_worksheet = FALSE for the first worksheet and add_worksheet = TRUE for the consecutive worksheet(s).

Index