1_Merge

AnfPrak

2022-03-30

```
#####################################
#### Package / Library setup
#####################################
# Specifies which packages are used and installs / loads all that are required
lib_need <- c("tidyverse")
lib_have <- lib_need %in% rownames(installed.packages())
if(any(!lib_have)) install.packages(lib_need[!lib_have])
invisible(lapply(lib_need, library, character.only = TRUE))
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
rm(lib_have, lib_need)
#####################################
```

Concept of this document

Read the documentation of 4_01_Readme.txt carefully. This script can be run for a first time to create the data set used for the presentation. This initial run requires that the duplicate files (see below) are already removed from the Meldedaten_RKI folder.

At the end there is the section "Filling in the missing days". Once this has been run the script can be rerun from the top and will create the entire data set including missing days which have either been added from further files or calculated from the difference of adjacent days.

Meta Data Analysis

First we examine the data files provided.

```
# The files must be in the subdirectory Meldedaten_RKI
# If unsure check with getwd()
# Get all filenames
daily_files <- list.files(file.path(".", "Meldedaten_RKI"))
# Create list of daily dates
daily_dates <- seq(as.Date("2020-03-27"), as.Date("2022-02-22"), by="days")
# Don't include ".rds" and compare only dates part
# of filename strings with daily dates as strings
```

```r
# Files do not show structure here:
issues <- which(str_sub(daily_files, -14, -5) %in%
                format(daily_dates, "%Y-%m-%d") != TRUE)
print(daily_files[issues])
```

```
## character(0)
```

```r
# Dates not covered by data
missing_days <- which(format(daily_dates, "%Y-%m-%d") %in%
                      str_sub(daily_files, -14, -5)  != TRUE)
print(daily_dates[missing_days])
```

```
## Date of length 0
```

```r
### Since missing days will be filled later we retain these in a file:
#saveRDS(missing_days, file = file.path(".", "missing_days.rds"))
missing_days <- readRDS(file = file.path(".", "missing_days.rds"))
```

Issues shows one file has a different name structure -> for one day there are two files. Verify the two files have the same data in the shared variables. (Will not be shown once renaming cases_GermanTemporal_2021-08-10.rds has occured.) Missing day is explained.

Recognize the remaining missing days as the ones removed and the one missing initially since they were duplicates.

Analyse the two data files for 2020-08-10

Note: A new directory Meldedaten_RKI_orig containing the original cases_GermanTemporal_2021-08-10.rds and cases_GermanTemporal_2021-08-10(1).rds was created for this verification. [Since the original cases_GermanTemporal_2021-08-10.rds was removed from the Meldedaten_RKI folder and the file with (1) renamed to replace it.]

```
cases_GermanTemporal_2021_08_10_v1 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2021-08-10.rds")
  )
cases_GermanTemporal_2021_08_10_v2 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2021-08-10(1).rds")
  )
# "state" <-> "land"
# "age_group" <-> "age"
# "new_case" <-> "newcase"
# "new_fatality" <-> "newdeath "
# "reference_date" <-> "date.desease"
# "new_recovered" <-> "cured"
# "count_recovered" <-> "newcure"
# "onset" <-> "is.beginning.desease"
# "age_group2" <-> "age2"
cases_GermanTemporal_2021_08_10_v1 <- as_tibble(cases_GermanTemporal_2021_08_10_v1) %>%
  mutate(across(where(is.factor), as.character)) %>%
  mutate(
    date = as.Date(date),
    reference_date = as.Date(reference_date)
  )
cases_GermanTemporal_2021_08_10_v2 <- as_tibble(cases_GermanTemporal_2021_08_10_v2) %>%
  mutate(
    date = as.Date(date),
    updated = as.Date(updated, format="%d.%m.%Y, %H:%M Uhr"),
    date.desease = as.Date(date.desease)
  )
compare <-
  subset(cases_GermanTemporal_2021_08_10_v1,
         select=names(cases_GermanTemporal_2021_08_10_v1)) ==
  subset(cases_GermanTemporal_2021_08_10_v2,
         select=names(cases_GermanTemporal_2021_08_10_v2)[c(3:9,11:18)])
print(all(compare))
```

```
## [1] TRUE
```

```
rm(cases_GermanTemporal_2021_08_10_v1)
rm(cases_GermanTemporal_2021_08_10_v2)
rm(compare)
gc()
```

```
##           used (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells 1105877 59.1    2196080  117.3         NA   1398167   74.7
## Vcells 1930181 14.8  164260076 1253.3      65536 185510742 1415.4
```

Hence we can remove the "cases_GermanTemporal_2021-08-10.rds" and use the more extensive "cases_GermanTemporal_2021-08-10(1).rds" by manually renaming the files.

Examine the remaining duplicate files (in R)

Note: A new directory Meldedaten_RKI_orig containing the following 6 pairs of files was created for this verification. [Note the files are identical even though they have different names - hence the duplicates are removed from the RKI_Meldedaten folder.]

```
cases_GermanTemporal_2020_04_06 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2020-04-06.rds")
  )
cases_GermanTemporal_2020_04_07 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2020-04-07.rds")
  )
cases_GermanTemporal_2020_04_29 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2020-04-29.rds")
  )
cases_GermanTemporal_2020_04_30 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2020-04-30.rds")
  )
cases_GermanTemporal_2020_05_18 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2020-05-18.rds")
  )
cases_GermanTemporal_2020_05_19 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2020-05-19.rds")
  )
cases_GermanTemporal_2020_07_13 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2020-07-13.rds")
  )
cases_GermanTemporal_2020_07_14 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2020-07-14.rds")
  )
cases_GermanTemporal_2021_09_08 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2021-09-08.rds")
  )
cases_GermanTemporal_2021_09_09 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2021-09-09.rds")
  )
cases_GermanTemporal_2021_10_25 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2021-10-25.rds")
  )
cases_GermanTemporal_2021_10_26 <- readRDS(
  file.path(".", "Meldedaten_RKI_orig", "cases_GermanTemporal_2021-10-26.rds")
  )
# Each pair has the same number of variables,
# so we can compare without standardizing
compare_2020_04_07 <-
  cases_GermanTemporal_2020_04_06 ==
  cases_GermanTemporal_2020_04_07
print(all(compare_2020_04_07))
```

```
## [1] TRUE
```

```
compare_2020_04_30 <-
  cases_GermanTemporal_2020_04_29 ==
  cases_GermanTemporal_2020_04_30
print(all(compare_2020_04_30))
```

```
## [1] TRUE
```

```
compare_2020_05_19 <-
  cases_GermanTemporal_2020_05_18 ==
  cases_GermanTemporal_2020_05_19
print(all(compare_2020_05_19))
```

```
## [1] TRUE
```

```
compare_2020_07_14 <-
  cases_GermanTemporal_2020_07_13 ==
  cases_GermanTemporal_2020_07_14
print(all(compare_2020_07_14))
```

```
## [1] TRUE
```

```
compare_2021_09_09 <-
  cases_GermanTemporal_2021_09_08 ==
  cases_GermanTemporal_2021_09_09
print(all(compare_2021_09_09))
```

```
## [1] TRUE
```

```
compare_2021_10_26 <-
  cases_GermanTemporal_2021_10_25 ==
  cases_GermanTemporal_2021_10_26
print(all(compare_2021_10_26))
```

```
## [1] TRUE
```

```
rm(cases_GermanTemporal_2020_04_06)
rm(cases_GermanTemporal_2020_04_07)
rm(cases_GermanTemporal_2020_04_29)
rm(cases_GermanTemporal_2020_04_30)
rm(cases_GermanTemporal_2020_05_18)
rm(cases_GermanTemporal_2020_05_19)
rm(cases_GermanTemporal_2020_07_13)
rm(cases_GermanTemporal_2020_07_14)
rm(cases_GermanTemporal_2021_09_08)
rm(cases_GermanTemporal_2021_09_09)
rm(cases_GermanTemporal_2021_10_25)
rm(cases_GermanTemporal_2021_10_26)
rm(compare_2020_04_07)
rm(compare_2020_04_30)
rm(compare_2020_05_19)
rm(compare_2020_07_14)
rm(compare_2021_09_09)
rm(compare_2021_10_26)
gc()
```

```
##           used (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells 1107644 59.2    2196080  117.3         NA   1398167   74.7
## Vcells 1935155 14.8  240321075 1833.6      65536 249226883 1901.5
```

Hence we can remove the duplicate files “cases_GermanTemporal_2020-04-07.rds”, “cases_GermanTemporal_2020-04-30.rds”, “cases_GermanTemporal_2020-05-19.rds”, “cases_GermanTemporal_2020-07-14.rds”, “cases_GermanTemporal_2021-09-09.rds” and “cases_GermanTemporal_2021-10-26.rds” from the Meldedaten_RKI folder.

Analyse the variable sets in the data files

Careful: This may take a long time - if possible just read in the resultant variable_info.rds file. Manually set run_switch = 1 to run.

```
# Now collect info about variables in files
variable_info <- list()
run_switch <- 0
if(run_switch == 1) {

  for(i in 1:length(daily_files)) {

    tmp_file <- readRDS(file.path(".", "Meldedaten_RKI", daily_files[i]))

    tmp_date_string <- str_sub(daily_files[i], -14, -5)

    # Show at what stage the loop is
    print(tmp_date_string)

    variable_info <- append(variable_info, list(c(tmp_date_string, names(tmp_file))))

    # Run garbage collector after every nth file (n=1)
    if(i %% 1 == 0) {
      rm(tmp_file)
      gc()
    }

  }

  saveRDS(variable_info, file = file.path(".", "variable_info.rds"))
  rm(tmp_file)
  gc()
}
variable_info <- readRDS(file.path(".", "variable_info.rds"))
```

Now analyze the data files based on this information. I.e. where do changes in variable set compositions occure?

```
# Now analyze the info about variables in files
var_analysis <- map_dfr(variable_info, ~as_tibble(t(.)))
```

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_repair` is
## Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
#var_analysis <- t(as.data.frame(var_analysis))
# Determine how many different sets of variables there are
var_sets <- unique(var_analysis[2:19])
print(var_sets)
```

```
## # A tibble: 6 x 18
##   V2    V3     V4    V5    V6    V7    V8    V9    V10   V11   V12   V13   V14
##   <chr> <chr>  <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
## 1 state distr~ age_~ gend~ cases deat~ date  new_~ new_~ <NA>  <NA>  <NA>  <NA>
## 2 state distr~ age_~ gend~ cases deat~ date  new_~ new_~ refe~ new_~ coun~ <NA>
## 3 state distr~ age_~ gend~ cases deat~ date  upda~ new_~ new_~ refe~ new_~ coun~
```

```
## 4 state distr~ age_~ gend~ cases deat~ date  upda~ new_~ new_~ refe~ new_~ coun~
## 5 id    landId land  dist~ age   gend~ cases deat~ date  dist~ upda~ newc~ newd~
## 6 state distr~ age_~ age_~ gend~ date  refe~ onset new_~ new_~ new_~ cases deat~
## # ... with 5 more variables: V15 <chr>, V16 <chr>, V17 <chr>, V18 <chr>,
## #   V19 <chr>
```

```
# Determine where variable sets change
var_change_pos <- vector(mode="list", length=length(variable_info))
for(i in 2:length(variable_info)) {
  var_change_pos[[i]] <- identical(variable_info[[i]][-1], variable_info[[i-1]][-1])
}
# Show results (note we compare i to i-1 starting with 2,
# hence need +1 to get the first item of new variable set)
change_sets <- which(unlist(var_change_pos) != TRUE) + 1
print(change_sets)
```

```
## [1]  12  16  34  35  36 497 541 671
```

```
print(var_analysis[change_sets,])
```

```
## # A tibble: 8 x 19
##   V1     V2    V3    V4    V5    V6    V7    V8    V9    V10   V11   V12   V13
##   <chr>  <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
## 1 2020-~ state dist~ age_~ gend~ cases deat~ date  new_~ new_~ refe~ new_~ coun~
## 2 2020-~ state dist~ age_~ gend~ cases deat~ date  upda~ new_~ new_~ refe~ new_~
## 3 2020-~ state dist~ age_~ gend~ cases deat~ date  upda~ new_~ new_~ refe~ new_~
## 4 2020-~ state dist~ age_~ gend~ cases deat~ date  upda~ new_~ new_~ refe~ new_~
## 5 2020-~ state dist~ age_~ gend~ cases deat~ date  upda~ new_~ new_~ refe~ new_~
## 6 2021-~ id    land~ land  dist~ age   gend~ cases deat~ date  dist~ upda~ newc~
## 7 2021-~ state dist~ age_~ gend~ cases deat~ date  upda~ new_~ new_~ refe~ new_~
## 8 2022-~ state dist~ age_~ age_~ gend~ date  refe~ onset new_~ new_~ new_~ cases
## # ... with 6 more variables: V14 <chr>, V15 <chr>, V16 <chr>, V17 <chr>,
## #   V18 <chr>, V19 <chr>
```

Harmonize Variable Codings

We do not want to lose any of the available information. Hence we have to rename the inconsistent variable names from 2021-09-23 to 2022-01-31 and fill rows not available with NA.

Further we must standardize the data and especially ensure that the dates are converted correctly.

```
# First we identify the renaming scheme:
#
# "state" <-> "land"
# "age_group" <-> "age"
# "new_case" <-> "newcase"
# "new_fatality" <-> "newdeath "
# "reference_date" <-> "date.desease"
# "new_recovered" <-> "cured"
# "count_recovered" <-> "newcure"
# "onset" <-> "is.beginning.desease"
# "age_group2" <-> "age2"
v1 <- c(unlist(na.omit(as_tibble(t(var_sets))[,1])))
v2 <- c(unlist(na.omit(as_tibble(t(var_sets))[,2])))
v3 <- c(unlist(na.omit(as_tibble(t(var_sets))[,3])))
v4 <- c(unlist(na.omit(as_tibble(t(var_sets))[,4])))
v5 <- c(unlist(na.omit(as_tibble(t(var_sets))[,5])))
v5_mod <- c(v5[1:2],
            "state",
            v5[4],
            "age_group",
            v5[6:11],
            "new_case",
            "new_fatality",
            "reference_date",
            "new_recovered",
            "count_recovered",
            "onset",
            "age_group2"
            )
v6 <- c(unlist(na.omit(as_tibble(t(var_sets))[,6])))
# Added later for filling missing dates:
v7_add <- c("landId",
            "state",
            "district",
            "age_group",
            "gender",
            "cases",
            "deaths",
            "id",
            "date",
            "districtId",
            "updated",
            "new_case",
            "new_fatality")
# Now we collect date formats across the data
identified_formats = c("%d.%m.%Y, %H:%M Uhr",
                       "%Y/%m/%d %H:%M:%S",
                       "%Y-%m-%d",
```

```
                        "%m/%d/%Y %H:%M:%S AM",
                        "%Y-%m-%d",
                        "%Y-%m-%d %H:%M:%S",
                        "%Y/%m/%d")
### Function standardize date format and column names
### in order to create proper tibble which can be combined with others.
###
### df_day: The data provided for the day
### Returns: Standardized tibble
###
### NOTE: The warnings are not problem here!
std_format_df <- function(df_day) {

  col_names <- names(df_day)

  if(all(col_names == v1)) {

    tmp_df <- df_day %>%
      mutate(
        date = as.Date(date, tryFormats=identified_formats)
      )

  }

  if(all(col_names == v2)) {

    tmp_df <- df_day %>%
      mutate(
        date = as.Date(date, tryFormats=identified_formats),
        reference_date = as.Date(reference_date, tryFormats=identified_formats)
      )

  }

  if(all(col_names == v3)) {

    tmp_df <- df_day %>%
      mutate(
        date = as.Date(date, tryFormats=identified_formats),
        updated = as.Date(updated, tryFormats=identified_formats),
        reference_date = as.Date(reference_date, tryFormats=identified_formats)
      )

  }

  if(all(col_names == v4)) {

    tmp_df <- df_day %>%
      mutate(
        date = as.Date(date, tryFormats=identified_formats),
        updated = as.Date(updated, tryFormats=identified_formats),
        reference_date = as.Date(reference_date, tryFormats=identified_formats)
      )
```

```
}

if(all(col_names == v5)) {

  tmp_df <- df_day

  names(tmp_df) <- v5_mod

  tmp_df <- tmp_df %>%
    mutate(
      date = as.Date(date, tryFormats=identified_formats),
      updated = as.Date(updated, tryFormats=identified_formats),
      reference_date = as.Date(reference_date, tryFormats=identified_formats)
    )

}

if(all(col_names == v5_mod)) {

  tmp_df <- df_day

  tmp_df <- tmp_df %>%
    mutate(
      date = as.Date(date, tryFormats=identified_formats),
      updated = as.Date(updated, tryFormats=identified_formats),
      reference_date = as.Date(reference_date, tryFormats=identified_formats)
    )

}

if(all(col_names == v6)) {

  tmp_df <- df_day %>%
    mutate(
      date = as.Date(date, tryFormats=identified_formats),
      updated = as.Date(updated, tryFormats=identified_formats),
      reference_date = as.Date(reference_date, tryFormats=identified_formats)
    )

}

if(all(col_names ==   v7_add)) {

  tmp_df <- df_day %>%
    mutate(
      date = as.Date(date, tryFormats=identified_formats),
      updated = as.Date(updated, tryFormats=identified_formats)
    )

}

return(tmp_df)
```

```
}
```

Create combined data table with publication_date

Now we use this function to standardize each day's data. Then an extra column called "publication_date" with the date from the filename will be added. Using this the delay in publication can be calculated. The individual days are combined by selecting the adequat rows and glueing them to the previous ones with bind_rows. This will fill non-existend columns with NA.

Manually activate run_switch since this may take very long to run.

```
# We start by taking the first file as is.
# Then we increment daywise and add the new data
# which is determined by new_case = 1, -1 and
# new_fatality = 1, -1
# Warning thrown here can be ignored.
# Possibly reread the files in the directory
#daily_files <- list.files(file.path(".", "Meldedaten_RKI"))
run_switch <- 0
time_start <- Sys.time()
if( run_switch ==1 ) {

  tmp_date_string <- str_sub(daily_files[1], -14, -5)

  initial_cases <- readRDS(file.path(".", "Meldedaten_RKI", daily_files[1]))

  cases_complete <- std_format_df(initial_cases) %>%
    mutate(
      publication_date = as.Date(tmp_date_string, tryFormats = "%Y-%m-%d")
    )

  rm(initial_cases)

  for(i in 2:length(daily_files)) {

    # Show at what stage the loop is
    tmp_date_string <- str_sub(daily_files[i], -14, -5)
    print(tmp_date_string)

    tmp_df <- readRDS(file.path(".", "Meldedaten_RKI", daily_files[i]))

    tmp_df <- std_format_df(tmp_df) %>%
      mutate(
        publication_date = as.Date(tmp_date_string, tryFormats = "%Y-%m-%d")
      )

    cases_complete <- bind_rows(cases_complete,
                                subset(tmp_df,
                                       new_case %in% c(-1,1) |
                                         new_fatality %in% c(-1,1)
                                       )
                                )

    gc()

  }
```

```
  rm(tmp_df)
  saveRDS(cases_complete, file = file.path(".", "cases_complete.rds"))
}
cases_complete <- readRDS(file.path(".", "cases_complete.rds"))
time_end <- Sys.time()
```

THIS IS THE PROGRESS USED FOR THE PRESENTATION. I.e. this is the combined data used for the presentation.

Filling in the missing days

Now we will fill the gaps with the additional data provided.

```
# First recall the missing days:
print(daily_dates[missing_days])
```

```
## [1] "2020-04-07" "2020-04-30" "2020-05-19" "2020-07-14" "2020-09-20"
## [6] "2021-09-09" "2021-10-26"
```

```
#Then the variable variant changes:
print(var_analysis[change_sets,])
```

```
## # A tibble: 8 x 19
##   V1     V2    V3    V4    V5    V6    V7    V8    V9    V10   V11   V12   V13
##   <chr>  <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
## 1 2020-~ state dist~ age_~ gend~ cases deat~ date  new_~ new_~ refe~ new_~ coun~
## 2 2020-~ state dist~ age_~ gend~ cases deat~ date  upda~ new_~ new_~ refe~ new_~
## 3 2020-~ state dist~ age_~ gend~ cases deat~ date  upda~ new_~ new_~ refe~ new_~
## 4 2020-~ state dist~ age_~ gend~ cases deat~ date  upda~ new_~ new_~ refe~ new_~
## 5 2020-~ state dist~ age_~ gend~ cases deat~ date  upda~ new_~ new_~ refe~ new_~
## 6 2021-~ id    land~ land  dist~ age   gend~ cases deat~ date  dist~ upda~ newc~
## 7 2021-~ state dist~ age_~ gend~ cases deat~ date  upda~ new_~ new_~ refe~ new_~
## 8 2022-~ state dist~ age_~ age_~ gend~ date  refe~ onset new_~ new_~ new_~ cases
## # ... with 6 more variables: V14 <chr>, V15 <chr>, V16 <chr>, V17 <chr>,
## #   V18 <chr>, V19 <chr>
```

So unfortunately we have changes in the variables sets adjacent to the gaps at 2020-04-07, 2020-04-30. We note this for potential later treatment.

Now we prepare the additional data provided - first we examine the column names manually. We note that for 2020-04-07 there is an additional variable set (which now is already included in the std_format_df - cf. v7_add). The remaining ones correspond to the already identified v5_mod. Hence we rename the columns accordingly.

```
# First recall the missing days:
print(daily_dates[missing_days])
```

```
## [1] "2020-04-07" "2020-04-30" "2020-05-19" "2020-07-14" "2020-09-20"
## [6] "2021-09-09" "2021-10-26"
```

```
# Now read in the original .csv provided:
cases_German_additional_2020_04_07 <- read.csv(
  file = file.path(".", "Ergaenzung_Originaldaten", "RKI_COVID19_2020-04-07.csv"),
  header=TRUE
  )
cases_German_additional_2020_04_30 <- read.csv(
  file = file.path(".", "Ergaenzung_Originaldaten", "RKI_COVID19_2020-04-30.csv"),
  header=TRUE
  )
cases_German_additional_2020_05_19 <- read.csv(
  file = file.path(".", "Ergaenzung_Originaldaten", "RKI_COVID19_2020-05-19.csv"),
  header=TRUE
  )
cases_German_additional_2020_07_14 <- read.csv(
  file = file.path(".", "Ergaenzung_Originaldaten", "RKI_COVID19_2020-07-14.csv"),
  header=TRUE
  )
```

```
cases_German_additional_2021_09_09 <- read.csv(
  file = file.path(".", "Ergaenzung_Originaldaten", "RKI_COVID19_2021-09-09.csv"),
  header=TRUE
  )
# Ensure compatible column names
names(cases_German_additional_2020_04_07) <- v7_add
names(cases_German_additional_2020_04_30) <- v5_mod
names(cases_German_additional_2020_05_19) <- v5_mod
names(cases_German_additional_2020_07_14) <- v5_mod
names(cases_German_additional_2021_09_09) <- v5_mod
```

We also note that there is no data provided for 2020-09-20 and 2021-10-26!

Analyse wether additional data files are consistent to fill gaps

We now examine whether the additional data can fill the gaps for cases and deaths. To that end we consider a three day period.

On day 1 the total number of cases (sum of new_case==0 and new_case==1) is calculated. Simliaryly for day 3 the total number of the previous day (sum of new_case==0 and new_case==-1) is calculated. Then for day 2 the sum of the previous day (sum new_case==0 and new_case==-1) and the current day 2 values (sum new_case==0 and new_case==1) are calculated. These are compared to day 1 and 3 respectively.

We proceed similarly for the deaths.

```
### Function standardize date format and column names
### in order to create proper tibble which can be combined with others.
###
### df_day: The data provided for the day
### Returns: Standardized tibble
###
### NOTE: The warnings are not problem here!
gap_fill_ok <- function(df_day_1, df_day_2, df_day_3) {

  # Note the use of abs() to count the negative cases correctly when new_case==-1
  d1_total_cases <- sum(subset(df_day_1, new_case!=-1)$cases)
  d3_prev_cases  <- sum(abs(subset(df_day_3, new_case!=1)$cases))

  d2_prev_cases  <- sum(abs(subset(df_day_2, new_case!=1)$cases))
  d2_total_cases <- sum(subset(df_day_2, new_case!=-1)$cases)

  check_cases    <- (d1_total_cases == d2_prev_cases) & (d3_prev_cases == d2_total_cases)


  # Note the use of abs() to count the negative cases correctly when new_case==-1
  d1_total_deaths <- sum(subset(df_day_1, new_fatality!=-1 & new_fatality!=-9)$deaths)
  d3_prev_deaths  <- sum(abs(subset(df_day_3, new_fatality!=1 & new_fatality!=-9)$deaths))

  d2_prev_deaths  <- sum(abs(subset(df_day_2, new_fatality!=1 & new_fatality!=-9)$deaths))
  d2_total_deaths <- sum(subset(df_day_2, new_fatality!=-1 & new_fatality!=-9)$deaths)

  check_deaths    <- (d1_total_deaths == d2_prev_deaths) & (d3_prev_deaths == d2_total_deaths)


  return(check_cases & check_deaths)
```

```
}
### 2020-04-07
cases_GermanTemporal_2020_04_06 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2020-04-06.rds")
  )
cases_GermanTemporal_2020_04_08 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2020-04-08.rds")
  )
print("2020-04-07 suitable:")
```

```
## [1] "2020-04-07 suitable:"
```

```
print(
  gap_fill_ok(cases_GermanTemporal_2020_04_06,
              cases_German_additional_2020_04_07,
              cases_GermanTemporal_2020_04_08)
)
```

```
## [1] TRUE
```

```
rm(cases_GermanTemporal_2020_04_06)
rm(cases_GermanTemporal_2020_04_08)
### 2020-04-30
cases_GermanTemporal_2020_04_29 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2020-04-29.rds")
  )
cases_GermanTemporal_2020_05_01 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2020-05-01.rds")
  )
print("2020-04-30 suitable:")
```

```
## [1] "2020-04-30 suitable:"
```

```
print(
  gap_fill_ok(cases_GermanTemporal_2020_04_29,
              cases_German_additional_2020_04_30,
              cases_GermanTemporal_2020_05_01)
)
```

```
## [1] TRUE
```

```
rm(cases_GermanTemporal_2020_04_29)
rm(cases_GermanTemporal_2020_05_01)
### 2020-05-19
cases_GermanTemporal_2020_05_18 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2020-05-18.rds")
  )
cases_GermanTemporal_2020_05_20 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2020-05-20.rds")
  )
print("2020-05-19 suitable:")
```

```
## [1] "2020-05-19 suitable:"
```

```
print(
  gap_fill_ok(cases_GermanTemporal_2020_05_18,
              cases_German_additional_2020_05_19,
```

```
              cases_GermanTemporal_2020_05_20)
)
```

```
## [1] TRUE
```

```
rm(cases_GermanTemporal_2020_05_18)
rm(cases_GermanTemporal_2020_05_20)
### 2020-07-14
cases_GermanTemporal_2020_07_13 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2020-07-13.rds")
  )
cases_GermanTemporal_2020_07_15 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2020-07-15.rds")
  )
print("2020-07-14 suitable:")
```

```
## [1] "2020-07-14 suitable:"
```

```
print(
  gap_fill_ok(cases_GermanTemporal_2020_07_13,
              cases_German_additional_2020_07_14,
              cases_GermanTemporal_2020_07_15)
)
```

```
## [1] TRUE
```

```
rm(cases_GermanTemporal_2020_07_13)
rm(cases_GermanTemporal_2020_07_15)
### 2021-09-09
cases_GermanTemporal_2021_09_08 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2021-09-08.rds")
  )
cases_GermanTemporal_2021_09_10 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2021-09-10.rds")
  )
print("2021-09-10 suitable:")
```

```
## [1] "2021-09-10 suitable:"
```

```
print(
  gap_fill_ok(std_format_df(cases_GermanTemporal_2021_09_08),
              cases_German_additional_2021_09_09,
              std_format_df(cases_GermanTemporal_2021_09_10))
)
```

```
## Warning in col_names == v2: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v3: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v4: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v6: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v7_add: longer object length is not a multiple of
## shorter object length
```

```
## Warning in col_names == v2: longer object length is not a multiple of shorter
## object length

## Warning in col_names == v3: longer object length is not a multiple of shorter
## object length

## Warning in col_names == v4: longer object length is not a multiple of shorter
## object length

## Warning in col_names == v6: longer object length is not a multiple of shorter
## object length

## Warning in col_names == v7_add: longer object length is not a multiple of
## shorter object length

## [1] TRUE
```

```
rm(cases_GermanTemporal_2021_09_08)
rm(cases_GermanTemporal_2021_09_10)
gc()
```

```
##             used  (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells   1159762  62.0    3647890  194.9         NA   3983796  212.8
## Vcells 110116130 840.2  346238348 2641.6      65536 346194911 2641.3
```

Hence all the additional data provided can be used to fill the gaps. We add it to the Meldedaten_RKI folder so we can rerun the " r combine data " loop above with these additional inputs.

```
saveRDS(cases_German_additional_2020_04_07,
        file = file.path(".",
                         "Meldedaten_RKI",
                         "cases_GermanTemporal_2020-04-07.rds"))
saveRDS(cases_German_additional_2020_04_30,
        file = file.path(".",
                         "Meldedaten_RKI",
                         "cases_GermanTemporal_2020-04-30.rds"))
saveRDS(cases_German_additional_2020_05_19,
        file = file.path(".",
                         "Meldedaten_RKI",
                         "cases_GermanTemporal_2020-05-19.rds"))
saveRDS(cases_German_additional_2020_07_14,
        file = file.path(".",
                         "Meldedaten_RKI",
                         "cases_GermanTemporal_2020-07-14.rds"))
saveRDS(cases_German_additional_2021_09_09,
        file = file.path(".",
                         "Meldedaten_RKI",
                         "cases_GermanTemporal_2021-09-09.rds"))
rm(cases_German_additional_2020_04_07)
rm(cases_German_additional_2020_04_30)
rm(cases_German_additional_2020_05_19)
rm(cases_German_additional_2020_07_14)
rm(cases_German_additional_2021_09_09)
gc()
```

```
##             used  (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells   1158853  61.9    3647890  194.9         NA   3983796  212.8
## Vcells 74471947 568.2  276990679 2113.3      65536 346194911 2641.3
```

Calculate remaining 2 days from differences of adjacent days

Now it only remains to treat the two missing days of 2020-09-20 and 2021-10-26. We only fill the gaps for cases an deaths.

```
cases_GermanTemporal_2020_09_19 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2020-09-19.rds")
  )
cases_GermanTemporal_2020_09_21 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2020-09-21.rds")
  )
# Verify column names compatability - they are the same
names(cases_GermanTemporal_2020_09_19)
```

```
##  [1] "state"           "district"        "age_group"       "gender"
##  [5] "cases"           "deaths"          "date"            "updated"
##  [9] "new_case"        "new_fatality"    "reference_date"  "new_recovered"
## [13] "count_recovered" "onset"           "age_group2"
```

```
names(cases_GermanTemporal_2020_09_21)
```

```
##  [1] "state"           "district"        "age_group"       "gender"
##  [5] "cases"           "deaths"          "date"            "updated"
##  [9] "new_case"        "new_fatality"    "reference_date"  "new_recovered"
## [13] "count_recovered" "onset"           "age_group2"
```

```
# Standardize
cases_GermanTemporal_2020_09_19 <- std_format_df(cases_GermanTemporal_2020_09_19)
```

```
## Warning in col_names == v1: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v2: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v3: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v5: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v5_mod: longer object length is not a multiple of
## shorter object length
```

```
## Warning in col_names == v7_add: longer object length is not a multiple of
## shorter object length
```

```
cases_GermanTemporal_2020_09_21 <- std_format_df(cases_GermanTemporal_2020_09_21)
```

```
## Warning in col_names == v1: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v2: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v3: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v5: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v5_mod: longer object length is not a multiple of
```

```
## shorter object length
```

```
## Warning in col_names == v7_add: longer object length is not a multiple of
## shorter object length
```

```
# Determine total number of cases and deaths by groups on 2020-09-19
df_day_1 <-cases_GermanTemporal_2020_09_19 %>%
  select(state, district, age_group, gender, cases, deaths, date, new_case, new_fatality) %>%
  group_by(state, district, age_group, gender, date) %>%
  mutate(
    helper_col_cases = case_when(
      new_case == -1 ~ 0,
      new_case ==  0 ~ 1,
      new_case ==  1 ~ 1
    ),
    helper_col_deaths = case_when(
      new_fatality == -9 ~ 0,
      new_fatality == -1 ~ 0,
      new_fatality ==  0 ~ 1,
      new_fatality ==  1 ~ 1
    )
  ) %>%
  select(-new_case, -new_fatality) %>%
  summarise(
    total_cases_1  = sum(cases*helper_col_cases),
    total_deaths_1 = sum(deaths*helper_col_deaths)
  ) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'state', 'district', 'age_group', 'gender'.
## You can override using the `.groups` argument.
```

```
# Verify the totals align with expectation
d1_total_cases_v1  <- sum(subset(cases_GermanTemporal_2020_09_19, new_case!=-1)$cases)
d1_total_deaths_v1 <- sum(
  subset(cases_GermanTemporal_2020_09_19, new_fatality!=-1 & new_fatality!=-9)$deaths
  )
d1_total_cases_v2  <- sum(df_day_1$total_cases_1)
d1_total_deaths_v2 <- sum(df_day_1$total_deaths_1)
# Compare - they match
print(d1_total_cases_v1 == d1_total_cases_v2)
```

```
## [1] TRUE
```

```
print(d1_total_deaths_v1 == d1_total_deaths_v2)
```

```
## [1] TRUE
```

```
# Determine the total number of cases and deaths by groups of the previous day for 2020-09-21
df_day_3 <-cases_GermanTemporal_2020_09_21 %>%
  select(state, district, age_group, gender, cases, deaths, date, new_case, new_fatality) %>%
  #select(-updated, -reference_date, -new_recovered, -count_recovered, - onset, -age_group2) %>%
  group_by(state, district, age_group, gender, date) %>%
  mutate(
    helper_col_cases = case_when(
      new_case == -1 ~ -1,
      new_case ==  0 ~  1,
```

```
      new_case ==  1 ~  0
    ),
    helper_col_deaths = case_when(
      new_fatality == -9 ~  0,
      new_fatality == -1 ~ -1,
      new_fatality ==  0 ~  1,
      new_fatality ==  1 ~  0
    )
  ) %>%
  select(-new_case, -new_fatality) %>%
  summarise(
    total_cases_3  = sum(cases*helper_col_cases),
    total_deaths_3 = sum(deaths*helper_col_deaths)
  ) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'state', 'district', 'age_group', 'gender'.
## You can override using the `.groups` argument.
```

```
# Verify the totals align with expectation
d3_prev_total_cases_v1  <- sum(
  abs(subset(cases_GermanTemporal_2020_09_21, new_case!=1)$cases)
  )
d3_prev_total_deaths_v1 <- sum(
  abs(subset(cases_GermanTemporal_2020_09_21, new_fatality!=1 & new_fatality!=-9)$deaths)
  )
d3_prev_total_cases_v2  <- sum(df_day_3$total_cases_3)
d3_prev_total_deaths_v2 <- sum(df_day_3$total_deaths_3)
# Compare - they match
print(d3_prev_total_cases_v1 == d3_prev_total_cases_v2)
```

```
## [1] TRUE
```

```
print(d3_prev_total_deaths_v1 == d3_prev_total_deaths_v2)
```

```
## [1] TRUE
```

```
# Now we recreate day 2
# We assume no cancellations took place on day 2, except when necessary
# (i.e. difference in cases is negative between day 3 and 1)
df_day_2 <- merge(x=df_day_3,
                  y=df_day_1,
                  by=c("state", "district", "age_group", "gender", "date"),
                  all.x=TRUE)
# If total_cases_1 or total_deaths_1 is NA then these are new entries on day 2.
# For the remaining entries determine the difference between these two.
df_day_2 <- df_day_2 %>%
  mutate(
    diff_cases  = total_cases_3 - total_cases_1,
    diff_deaths = total_deaths_3 - total_deaths_1
  ) %>%
  mutate(
    new_case = case_when(
      is.na(total_cases_1) & total_cases_3 >  0 ~  1,
      # Note this is a new case on 2020-09-21, hence 0 cases difference:
      is.na(total_cases_1) & total_cases_3 == 0 ~  -2,
```

```
      diff_cases >  0   ~  1,
      diff_cases == 0   ~  0,
      diff_cases <  0   ~ -1
    ),
    cases = case_when(
      is.na(total_cases_1) & total_cases_3 >  0 ~  total_cases_3 ,
      # Note this is a new case on day 3, hence 0 cases difference with NA on day 1:
      is.na(total_cases_1) & total_cases_3 == 0 ~  0,
      diff_cases >  0   ~  diff_cases,
      diff_cases == 0   ~  total_cases_1,
      diff_cases <  0   ~  diff_cases
    ),
    new_fatality = case_when(
      is.na(total_deaths_1) & total_deaths_3 >  0 ~  1,
      # Note this is a new fatality on 2020-09-21, hence 0 cases difference:
      is.na(total_deaths_1) & total_deaths_3 == 0 ~  -2,
      diff_deaths >  0   ~  1,
      diff_deaths == 0   ~  0,
      diff_deaths <  0   ~ -1
    ),
    deaths = case_when(
      is.na(total_deaths_1) & total_deaths_3 >  0 ~  total_deaths_3 ,
      # Note this is a new fatality on day 3, hence 0 cases difference with NA on day 1:
      is.na(total_deaths_1) & total_deaths_3 == 0 ~  0,
      diff_deaths >  0   ~  diff_deaths,
      diff_deaths == 0   ~  total_deaths_1,
      diff_deaths <  0   ~  diff_deaths
    )
  )
# Now remove the non-entries coded by -2
df_day_2 <- df_day_2[-which(df_day_2$new_case == -2 & df_day_2$new_fatality == -2),]
# Hamonize remaining -2s
df_day_2 <- df_day_2 %>%
  mutate(
    new_case = case_when(
      new_case == -2 & new_fatality > -2 ~ 0,
      TRUE ~ new_case
    ),
    new_fatality = case_when(
      new_case > -2 & new_fatality == -2 ~ -9,
      TRUE ~ new_fatality
    )
  )
# Now we select the desired columns which will comply with the v1 variable set
df_day_2 <- df_day_2 %>%
  select(state, district, age_group, gender, cases, deaths, date, new_case, new_fatality)
# So we save the reconstructed day:
saveRDS(df_day_2, file = file.path(".",
                                   "Meldedaten_RKI",
                                   "cases_GermanTemporal_2020-09-20.rds"))
rm(cases_GermanTemporal_2020_09_19)
rm(cases_GermanTemporal_2020_09_21)
rm(df_day_1)
```

```
rm(df_day_2)
rm(df_day_3)
gc()
```

```
##            used  (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells  1200828  64.2    3647890  194.9         NA   3983796  212.8
## Vcells 74564569 568.9  221592544 1690.7      65536 346194911 2641.3
```

We proceed similarly for the 2021-10-26

```
cases_GermanTemporal_2021_10_25 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2021-10-25.rds")
  )
cases_GermanTemporal_2021_10_27 <- readRDS(
  file.path(".", "Meldedaten_RKI", "cases_GermanTemporal_2021-10-27.rds")
  )
# Verify column names compatability - they are the same
names(cases_GermanTemporal_2021_10_25)
```

```
##  [1] "state"           "district"        "age_group"       "gender"
##  [5] "cases"           "deaths"          "date"            "updated"
##  [9] "new_case"        "new_fatality"    "reference_date"  "new_recovered"
## [13] "count_recovered" "onset"           "age_group2"
```

```
names(cases_GermanTemporal_2021_10_27)
```

```
##  [1] "state"           "district"        "age_group"       "gender"
##  [5] "cases"           "deaths"          "date"            "updated"
##  [9] "new_case"        "new_fatality"    "reference_date"  "new_recovered"
## [13] "count_recovered" "onset"           "age_group2"
```

```
# Standardize
cases_GermanTemporal_2021_10_25 <- std_format_df(cases_GermanTemporal_2021_10_25)
```

```
## Warning in col_names == v1: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v2: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v3: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v5: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v5_mod: longer object length is not a multiple of
## shorter object length
```

```
## Warning in col_names == v7_add: longer object length is not a multiple of
## shorter object length
```

```
cases_GermanTemporal_2021_10_27 <- std_format_df(cases_GermanTemporal_2021_10_27)
```

```
## Warning in col_names == v1: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v2: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v3: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v5: longer object length is not a multiple of shorter
## object length
```

```
## Warning in col_names == v5_mod: longer object length is not a multiple of
## shorter object length
```

```
## Warning in col_names == v7_add: longer object length is not a multiple of
## shorter object length
```

```
# Determine total number of cases and deaths by groups on 2021-10-25
df_day_1 <-cases_GermanTemporal_2021_10_25 %>%
  select(state, district, age_group, gender, cases, deaths, date, new_case, new_fatality) %>%
  group_by(state, district, age_group, gender, date) %>%
  mutate(
    helper_col_cases = case_when(
      new_case == -1 ~ 0,
      new_case ==  0 ~ 1,
      new_case ==  1 ~ 1
    ),
    helper_col_deaths = case_when(
      new_fatality == -9 ~ 0,
      new_fatality == -1 ~ 0,
      new_fatality ==  0 ~ 1,
      new_fatality ==  1 ~ 1
    )
  ) %>%
  select(-new_case, -new_fatality) %>%
  summarise(
    total_cases_1  = sum(cases*helper_col_cases),
    total_deaths_1 = sum(deaths*helper_col_deaths)
  ) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'state', 'district', 'age_group', 'gender'.
## You can override using the `.groups` argument.
```

```
# Verify the totals align with expectation
d1_total_cases_v1  <- sum(subset(cases_GermanTemporal_2021_10_25, new_case!=-1)$cases)
d1_total_deaths_v1 <- sum(
  subset(cases_GermanTemporal_2021_10_25, new_fatality!=-1 & new_fatality!=-9)$deaths
  )
d1_total_cases_v2  <- sum(df_day_1$total_cases_1)
d1_total_deaths_v2 <- sum(df_day_1$total_deaths_1)
# Compare - they match
print(d1_total_cases_v1 == d1_total_cases_v2)
```

```
## [1] TRUE
```

```
print(d1_total_deaths_v1 == d1_total_deaths_v2)
```

```
## [1] TRUE
```

```
# Determine the total number of cases and deaths by groups of the previous day for 2021-10-27
df_day_3 <-cases_GermanTemporal_2021_10_27 %>%
  select(state, district, age_group, gender, cases, deaths, date, new_case, new_fatality) %>%
```

```
  group_by(state, district, age_group, gender, date) %>%
  mutate(
    helper_col_cases = case_when(
      new_case == -1 ~ -1,
      new_case ==  0 ~  1,
      new_case ==  1 ~  0
    ),
    helper_col_deaths = case_when(
      new_fatality == -9 ~  0,
      new_fatality == -1 ~ -1,
      new_fatality ==  0 ~  1,
      new_fatality ==  1 ~  0
    )
  ) %>%
  select(-new_case, -new_fatality) %>%
  summarise(
    total_cases_3  = sum(cases*helper_col_cases),
    total_deaths_3 = sum(deaths*helper_col_deaths)
  ) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'state', 'district', 'age_group', 'gender'.
## You can override using the `.groups` argument.
```

```
# Verify the totals align with expectation
d3_prev_total_cases_v1  <- sum(
  abs(subset(cases_GermanTemporal_2021_10_27, new_case!=1)$cases)
  )
d3_prev_total_deaths_v1 <- sum(
  abs(subset(cases_GermanTemporal_2021_10_27, new_fatality!=1 & new_fatality!=-9)$deaths)
  )
d3_prev_total_cases_v2  <- sum(df_day_3$total_cases_3)
d3_prev_total_deaths_v2 <- sum(df_day_3$total_deaths_3)
# Compare - they match
print(d3_prev_total_cases_v1 == d3_prev_total_cases_v2)
```

```
## [1] TRUE
```

```
print(d3_prev_total_deaths_v1 == d3_prev_total_deaths_v2)
```

```
## [1] TRUE
```

```
# Now we recreate day 2
# We assume no cancellations took place on day 2, except when necessary
# (i.e. difference in cases is negative between day 3 and 1)
df_day_2 <- merge(x=df_day_3,
                  y=df_day_1,
                  by=c("state", "district", "age_group", "gender", "date"),
                  all.x=TRUE)
# If total_cases_1 or total_deaths_1 is NA then these are new entries on day 2.
# For the remaining entries determine the difference between these two.
df_day_2 <- df_day_2 %>%
  mutate(
    diff_cases  = total_cases_3 - total_cases_1,
    diff_deaths = total_deaths_3 - total_deaths_1
  ) %>%
```

```
  mutate(
    new_case = case_when(
      is.na(total_cases_1) & total_cases_3 >  0 ~  1,
      # Note this is a new case on 2020-09-21, hence 0 cases difference:
      is.na(total_cases_1) & total_cases_3 == 0 ~  -2,
      diff_cases >  0   ~  1,
      diff_cases == 0   ~  0,
      diff_cases <  0   ~ -1
    ),
    cases = case_when(
      is.na(total_cases_1) & total_cases_3 >  0 ~  total_cases_3 ,
      # Note this is a new case on day 3, hence 0 cases difference with NA on day 1:
      is.na(total_cases_1) & total_cases_3 == 0 ~  0,
      diff_cases >  0   ~  diff_cases,
      diff_cases == 0   ~  total_cases_1,
      diff_cases <  0   ~  diff_cases
    ),
    new_fatality = case_when(
      is.na(total_deaths_1) & total_deaths_3 >  0 ~  1,
      # Note this is a new fatality on 2020-09-21, hence 0 cases difference:
      is.na(total_deaths_1) & total_deaths_3 == 0 ~  -2,
      diff_deaths >  0   ~  1,
      diff_deaths == 0   ~  0,
      diff_deaths <  0   ~ -1
    ),
    deaths = case_when(
      is.na(total_deaths_1) & total_deaths_3 >  0 ~  total_deaths_3 ,
      # Note this is a new fatality on day 3, hence 0 cases difference with NA on day 1:
      is.na(total_deaths_1) & total_deaths_3 == 0 ~  0,
      diff_deaths >  0   ~  diff_deaths,
      diff_deaths == 0   ~  total_deaths_1,
      diff_deaths <  0   ~  diff_deaths
    )
  )
# Now remove the non-entries coded by -2
df_day_2 <- df_day_2[-which(df_day_2$new_case == -2 & df_day_2$new_fatality == -2),]
# Hamonize remaining -2s
df_day_2 <- df_day_2 %>%
  mutate(
    new_case = case_when(
      new_case == -2 & new_fatality > -2 ~ 0,
      TRUE ~ new_case
    ),
    new_fatality = case_when(
      new_case > -2 & new_fatality == -2 ~ -9,
      TRUE ~ new_fatality
    )
  )
# Now we select the desired columns which will comply with the v1 variable set
df_day_2 <- df_day_2 %>%
  select(state, district, age_group, gender, cases, deaths, date, new_case, new_fatality)
# So we save the reconstructed day:
saveRDS(df_day_2, file = file.path(".",
```

```
                                    "Meldedaten_RKI",
                                    "cases_GermanTemporal_2021-10-26.rds"))
rm(cases_GermanTemporal_2021_10_25)
rm(cases_GermanTemporal_2021_10_27)
rm(df_day_1)
rm(df_day_2)
rm(df_day_3)
gc()
```

```
##            used  (Mb) gc trigger   (Mb) limit (Mb)  max used   (Mb)
## Ncells  1200845  64.2    5684657  303.6         NA  11102845  593.0
## Vcells 77711488 592.9  212792842 1623.5      65536 346194911 2641.3
```

Now there are data files for every day from 2020-03-27 up to 2022-02-22 in the Meldedaten_RKI folder. Hence the script can be run from the top and create the full combined data.