Compare the OMI and TROPOMI NO2 products

Tze-Li Liu

2021-05-06

```
library(stars) ; library(sf) ; library(raster)
library(dplyr) ; library(tidyr)
library(ggplot2)
library(lubridate) ; library(stringr)
```

This document shows the comparison between the OMI- and TROPOMI-NO2 products, with regards to:

- overall data availability
- spatial resolution (grid cells)
- pixel-wise comparisons
  - scatter plot and simple linear regression
  - cross table of missing-value pixels between TROPOMI and OMI
  - disagreement between the missing-value pixels
  - spatial and temporal distribution of missing-value pixels

---

Loading the datasets

Switzerland boundary

```
CH <- st_read("1_data/raw/Switzerland_shapefile/CHE_adm0.shp") %>%
  st_geometry() %>%
  st_simplify(preserveTopology = TRUE , dTolerance = 0.05)
```

The polygon is simplified (reduce endpoints) to facilitate visualization.

Load OMI data

The 365 .tif files are read simultaneously as a stacked raster array.

```
# file paths
files_OMI <- list.files("1_data/raw/OMI-NO2" , "_AOI.tif$" , full.names = TRUE)
# timestamps
ts_OMI_raw <- basename(files_OMI) %>%
  str_extract("\\d{8}") %>%
  as_date()
# read as a stacked raster
#OMI_raw <- read_stars(files_OMI) # but every image of each day as one separate attribute; needs to mer
```

```
OMI_raw <- read_stars(files_OMI , along = list(date = ts_OMI_raw))  %>%
  setNames("value")
# print stars object
OMI_raw
```

```
## stars object with 3 dimensions and 1 attribute
## attribute(s):
##      value
##  Min.   :-9.982e+14
##  1st Qu.: 4.562e+14
##  Median : 1.389e+15
##  Mean   : 1.968e+15
##  3rd Qu.: 2.728e+15
##  Max.   : 5.517e+16
##  NA's   :83574
## dimension(s):
##      from  to     offset  delta                       refsys point values x/y
## x       1  24       5.25   0.25 GEOGCS["unnamed ellipse",... FALSE   NULL [x]
## y       1  13      45.25   0.25 GEOGCS["unnamed ellipse",... FALSE   NULL [y]
## date    1 365 2019-01-01 1 days                         Date    NA   NULL
```

A spatial-temporal array with three dimensions: `x`, `y`, `date`.

Load TROPOMI

The 365 .tif files are read simultaneously as a stacked raster array. Each .tif file comes with two bands for the TROPOMI data: `nitorgendioxide_tropospheric_column` and `QA_value`. The `nitorgendioxide_tropospheric_column` is preliminarily screened with `QA_value > 0.5` while downloading the data.

```
# file paths
files_TROPOMI <- list.files("1_data/raw/TROPOMI-NO2/preprocessed_resampled" , "_AOI_rs.tif$" , full.nam
# timestamps
ts_TROPOMI_raw <- basename(files_TROPOMI) %>%
  str_extract("\\d{8}") %>%
  as_date()
# read as a stacked raster
TROPOMI_raw <- read_stars(files_TROPOMI ,
                          along = list(date = ts_TROPOMI_raw) ,
                          proxy = FALSE) %>%
  st_set_dimensions(values = rep(c("NO2" , "QA")) , "band") %>%
  setNames("value")
# pring stars object
TROPOMI_raw
```

```
## stars object with 4 dimensions and 1 attribute
## attribute(s), summary of first 1e+05 cells:
##      value
##  Min.   :-7.848e+15
##  1st Qu.: 1.000e+00
##  Median : 1.000e+00
##  Mean   : 1.909e+15
##  3rd Qu.: 1.902e+15
##  Max.   : 1.620e+17
##  NA's   :27728
```

```
## dimension(s):
##      from  to     offset      delta                        refsys point   values
## x       1  44    5.30956   0.132697 GEOGCS["WGS 84",DATUM["WG... FALSE     NULL
## y       1  32    48.2725 -0.0907649 GEOGCS["WGS 84",DATUM["WG... FALSE     NULL
## band    1   2         NA         NA                            NA    NA NO2, QA
## date    1 365 2019-01-01     1 days                          Date    NA     NULL
##      x/y
## x    [x]
## y    [y]
## band
## date
```

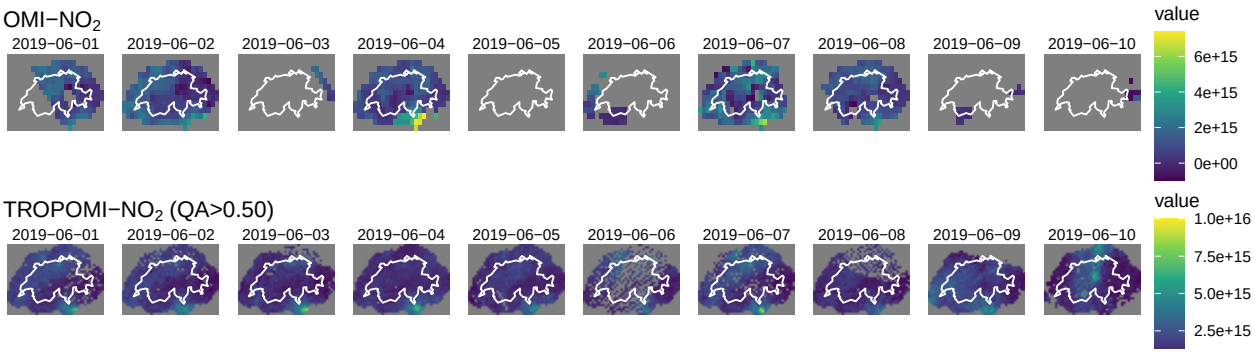
A spatial-temporal array with four dimensions: `x`, `y`, `band` and `date`.

Overview

Compare the original products: coverage

A time-series of seven days with the raw OMI- and TROPOMI-NO2 images.

```
# assign the dates to visualize and compare
selected_dates <- interval("2019-06-01" , "2019-06-10")
# subset the image arrays based on the assigned date
OMI_selected_dates <- OMI_raw %>%
  filter(date %within% selected_dates)
TROPOMI_selected_dates <- TROPOMI_raw %>%
  slice(band , 1) %>%
  filter(date %within% selected_dates)
# visualize
cowplot::plot_grid(
  ggplot() +
    geom_stars(data = OMI_selected_dates) +
    geom_sf(data = CH , fill = NA , color = "white") +
    facet_grid(~date) +
    scale_fill_viridis_c() +
    labs(title = expression("OMI-NO"[2])) +
    theme_void() ,
  ggplot() +
    geom_stars(data = TROPOMI_selected_dates) +
    geom_sf(data = CH , fill = NA , color = "white") +
    facet_grid(~date) +
    scale_fill_viridis_c(limits = c(0,1e16)) +
    labs(title = expression("TROPOMI-NO"[2] *" (QA>0.50)")) +
    theme_void() ,
  align = "v" , axis = "lr" , ncol = 1
)
```



The canvas above shows the raw OMI-NO2 images for ten randomly selected days (2019-06-01~2019-06-10), and the one below shows the raw TROPOMI-NO2 images of the same days. Note the difference between the OMI and TROPOMI products: cloud-interrupted pixels are already screened in the `ColumnAmountNO2TropCloudScreened` of OMI but not in the `nitorgendioxide_tropospheric_column` of TROPOMI. Instead, a `QA_value` band comes with the TROPOMI data product. During data downloading and preprocessing, the criteria `QA_value > 0.5` is used to prelimenarily screen the cloud-interrupted pixels. It can be seen in the figure that the coverage of the TROPOMI data is wider compared to OMI, under the `QA_value > 0.5` screen criteria. This might not be realistic. Additionally, some outlier extreme values exist in this TROPOMI product (not shown). Therefore the TROPOMI `NO2` band is screened again using the criteria `QA_value > 0.75`.

More details of the `QA_value` can be found in the *S5P Mission Performance Centre Nitrogen Dioxide [L2_NO2___] Readme* document:

> The quality of the individual observations depends on many factors, including cloud cover, surface albedo, presence of snow-ice, saturation, geometry etc. These aspects are taken into account in the definition of the "quality assurance value" (`qa_value`), available for each individual observation, which provides the users of the data with an easy filter to remove less accurate observations. The qa_value is a continuous variable, ranging from 0 (error) to 1 (all is well). The main flag for data usage is as follows:
> For variables `nitrogendioxide_tropospheric_column`, `nitrogendioxide_total_column`, `nitrogendioxide_summed_total_column`:
> - `qa_value > 0.75`
> This is the recommended pixel filter. It removes cloud-covered scenes (cloud radiance fraction 0.5), partially snow/ice covered scenes, errors, and problematic retrievals.
> - `qa_value > 0.50`
> Compared to the stricter filter, this adds the good quality retrievals over clouds and over scenes covered by snow/ice. Errors and problematic retrievals are still filtered out. In particular, this filter may be useful for assimilation and model comparison studies.

Re-screen the TROPOMI data with `QA_value > 0.75`

```
TROPOMI_QA75 <- TROPOMI_raw %>%
  split("band") %>%  # split dimension "band" into attributes: NO2 and QA
  mutate(NO2_QA75 = ifelse(QA>=0.75 , NO2 , NA_real_)) %>% # filter NO2 values based on QA values
  merge() %>%        # merge the attributes (bands) back as a dimension
  setNames("value") %>% # rename attribute
  st_set_dimensions(4 , names = "band") # clean dimension name
# print the stars object
TROPOMI_QA75
```

```
## stars object with 4 dimensions and 1 attribute
## attribute(s), summary of first 1e+05 cells:
##      value
```

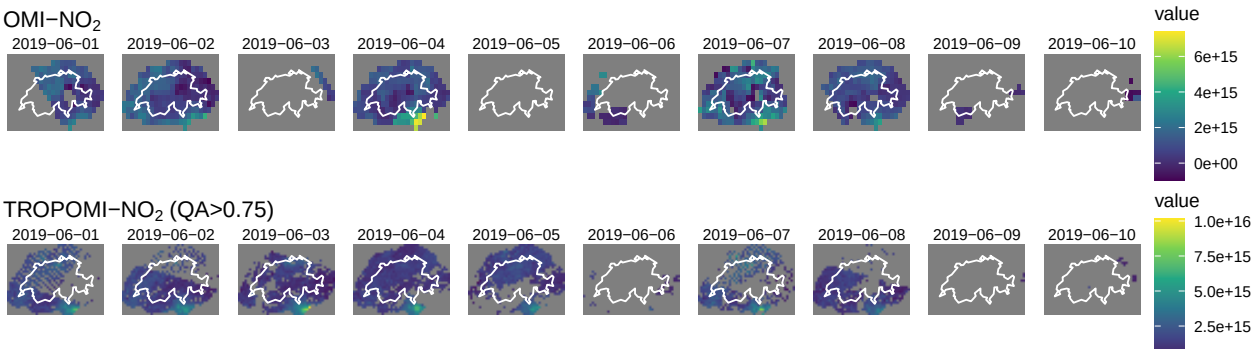
```
##  Min.   :-7.848e+15
##  1st Qu.: 8.389e+14
##  Median : 1.958e+15
##  Mean   : 3.447e+15
##  3rd Qu.: 4.011e+15
##  Max.   : 1.620e+17
##  NA's   :28839
## dimension(s):
##      from  to     offset      delta                          refsys point
## x       1  44    5.30956   0.132697 GEOGCS["WGS 84",DATUM["WG... FALSE
## y       1  32    48.2725 -0.0907649 GEOGCS["WGS 84",DATUM["WG... FALSE
## date    1 365 2019-01-01     1 days                            Date    NA
## band    1   3         NA         NA                              NA    NA
##                             values x/y
## x                             NULL [x]
## y                             NULL [y]
## date                          NULL
## band NO2     , QA      , NO2_QA75
```

The TROPOMI NO2 band is now re-screened as `stars` object `TROPOMI_QA75` with three bands: `NO2`, `QA_value`, `NO2_QA75`.

Compare the products: coverage

Here we use the re-screened TROPOMI product to visualize the 10-day time series of the same dates again and compare between the data availability of OMI and TROPOMI.

```
# assign the dates to visualize and compare
selected_dates <- interval("2019-06-01" , "2019-06-10")
# subset the image arrays based on the assigned date
OMI_selected_dates <- OMI_raw %>%
  filter(date %within% selected_dates)
TROPOMI_selected_dates_QA75 <- TROPOMI_QA75 %>%
  slice(band , 3) %>%  # band 3: NO2_QA75
  filter(date %within% selected_dates)
# visualization
cowplot::plot_grid(
  ggplot() +
    geom_stars(data = OMI_selected_dates) +
    geom_sf(data = CH , fill = NA , color = "white") +
    facet_grid(~date) +
    scale_fill_viridis_c() +
    labs(title = expression("OMI-NO"[2])) +
    theme_void() ,
  ggplot() +
    geom_stars(data = TROPOMI_selected_dates_QA75) +
    geom_sf(data = CH , fill = NA , color = "white") +
    facet_grid(~date) +
    scale_fill_viridis_c() +
    labs(title = expression("TROPOMI-NO"[2] *" (QA>0.75)")) +
    theme_void() ,
  align = "v" , axis = "lr" , ncol = 1
)
```



It can be seen that:

- The overall pattern of NO2 values looks similar: both showing higher values on bottom-right corner.
- The pattern of data availability is closer between the two products after using `QA_value > 0.75` for TROPOMI. Therefore it's advisable to use this criteria for screening in further analyses.

```
# use NO2_QA75 for TROPOMI
TROPOMI_fine <- TROPOMI_QA75 %>%
  filter(band == "NO2_QA75") %>%
  split("band") %>%
  setNames("TROPOMI")
```

OMI- and TROPOMI- grid cells

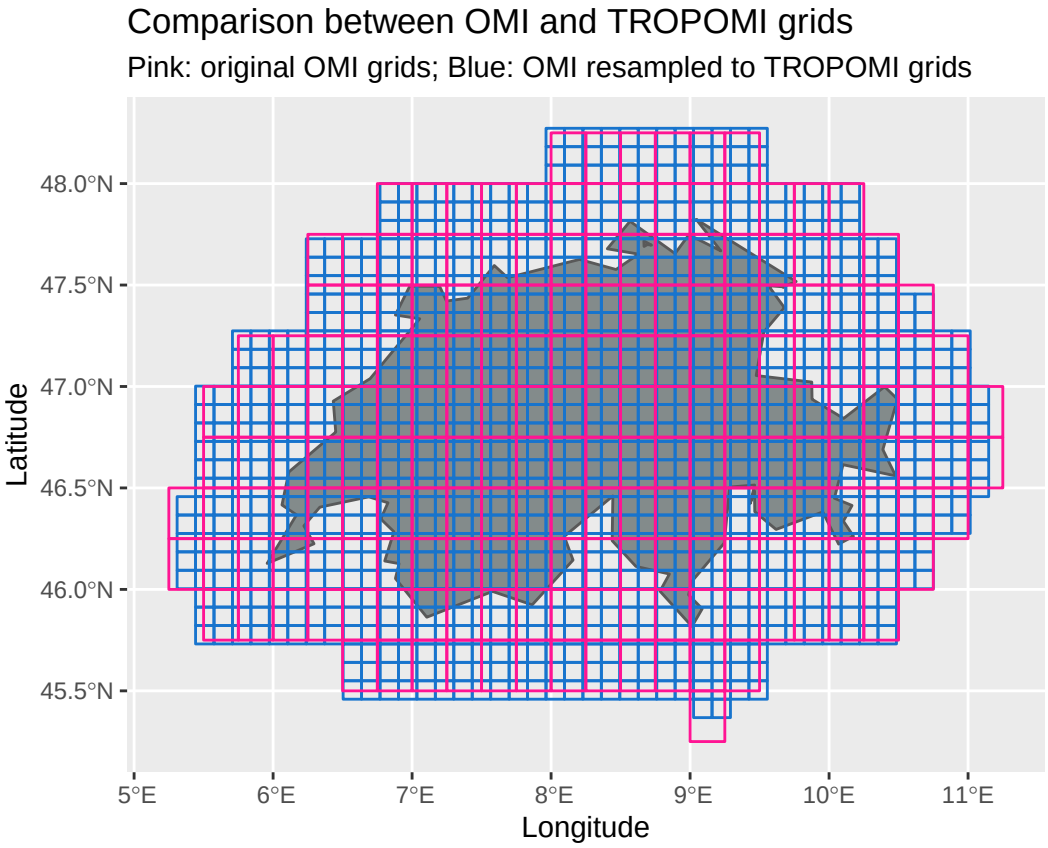
One biggest difference between the OMI- and TROPOMI products is the spatial resolution of the grid cells. Here we take a look at the grid cells. First the OMI raster array is resampled to the TROPOMI grids.

```
OMI_rs <- OMI_raw %>%
  st_warp(dest = TROPOMI_fine) %>%
  setNames("OMI")
# print stars object
OMI_rs
```

```
## stars object with 3 dimensions and 1 attribute
## attribute(s):
##       OMI
##  Min.   :-9.982e+14
##  1st Qu.: 4.634e+14
##  Median : 1.404e+15
##  Mean   : 1.987e+15
##  3rd Qu.: 2.749e+15
##  Max.   : 5.517e+16
##  NA's   :354970
## dimension(s):
##      from  to     offset      delta                       refsys point values
## x       1  44    5.30956   0.132697 GEOGCS["WGS 84",DATUM["WG... FALSE   NULL
## y       1  32    48.2725 -0.0907649 GEOGCS["WGS 84",DATUM["WG... FALSE   NULL
## date    1 365 2019-01-01     1 days                         Date    NA   NULL
##      x/y
## x    [x]
## y    [y]
## date
```

Now the spatial resolution and extent of `OMI_rs` is the same as `TROPOMI_raw` (shown above) (look at the dimensions). We can compare the size and shape of the grid cells before and after resampling.

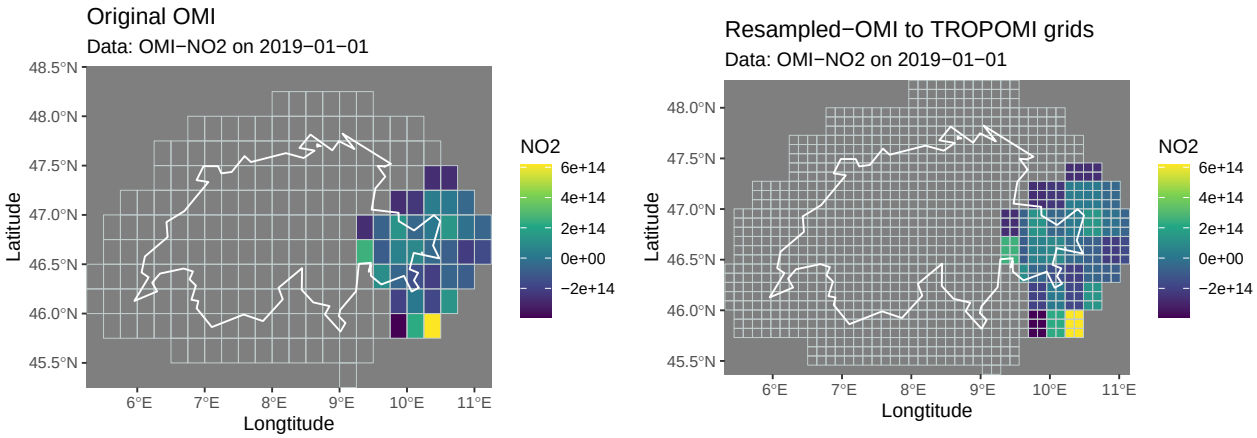
```
ggplot() +
  geom_sf(data = CH , fill = "azure4") +
  geom_sf(data = OMI_rs[,,,50] %>%
            st_as_sf(as_points = FALSE) ,
          color = "dodgerblue3" , fill = NA) +
  geom_sf(data = OMI_raw[,,,50] %>%
            st_as_sf(as_points = FALSE) ,
          color = "deeppink" , fill = NA) +
  labs(x = "Longitude" , y = "Latitude" ,
       title = "Comparison between OMI and TROPOMI grids" ,
       subtitle = "Pink: original OMI grids; Blue: OMI resampled to TROPOMI grids")
```



The pink grids are the original OMI grids, and the blue ones are from TROPOMI. The OMI cells are coarser compared to TROPOMI. We can also take a look at the before/after of resampling the OMI images. The resampling method is the default nearest neighbor method (see `help(st_warp)`) .

```
cowplot::plot_grid(
  # plot: OMI with original grids
  ggplot() +
    # first image (2019-01-01)
    geom_stars(data = OMI_raw[,,,1]) +
    # reference grid cells
    geom_sf(data = OMI_raw[,,,50] %>%
              st_as_sf(as_points = FALSE) ,
            color = "azure3" , fill = NA , size = 0.1) +
```

```
    # Switzerland
    geom_sf(data = CH , fill = NA , color = "white") +
    scale_fill_viridis_c() +
    coord_sf(expand = FALSE) +
    labs(x = "Longtitude" , y = "Latitude" , fill = "NO2" ,
         title = "Original OMI" ,
         subtitle = "Data: OMI-NO2 on 2019-01-01") ,
  # plot: OMI resampled to TROPOMI grids
  ggplot() +
    # first image (2019-01-01)
    geom_stars(data = OMI_rs[,,,1]) +
    # reference grid cells
    geom_sf(data = OMI_rs[,,,50] %>%
              st_as_sf(as_points = FALSE) ,
            color = "azure3" , fill = NA , size = 0.1) +
    # Switzerland
    geom_sf(data = CH , fill = NA , color = "white") +
    scale_fill_viridis_c() +
    coord_sf(expand = FALSE) +
    labs(x = "Longtitude" , y = "Latitude" , fill = "NO2" ,
         title = "Resampled-OMI to TROPOMI grids" ,
         subtitle = "Data: OMI-NO2 on 2019-01-01")
)
```



Now the OMI raster images are resampled and downscaled to the finer TROPOMI grids, and **the pixel-wise comparison will be based on TROPOMI and the resampled-OMI data (the figure on the right)**.

Pixel-wise comparison

Data preparation

```
FULL <- c(TROPOMI_fine , OMI_rs)
```

Stack the TROPOMI and the resampled-OMI raster arrays

```
## stars object with 3 dimensions and 2 attributes
## attribute(s):
##     TROPOMI                OMI
```

```
##  Min.   :-2.146e+15   Min.   :-9.982e+14
##  1st Qu.: 9.162e+14   1st Qu.: 4.634e+14
##  Median : 1.469e+15   Median : 1.404e+15
##  Mean   : 1.962e+15   Mean   : 1.987e+15
##  3rd Qu.: 2.299e+15   3rd Qu.: 2.749e+15
##  Max.   : 7.484e+16   Max.   : 5.517e+16
##  NA's   :350395       NA's   :354970
## dimension(s):
##      from  to     offset      delta                        refsys point values
## x       1  44    5.30956   0.132697 GEOGCS["WGS 84",DATUM["WG... FALSE   NULL
## y       1  32    48.2725 -0.0907649 GEOGCS["WGS 84",DATUM["WG... FALSE   NULL
## date    1 365 2019-01-01     1 days                         Date    NA   NULL
##      x/y
## x    [x]
## y    [y]
## date
```

- 3 dimensions: `x`, `y`, `date`
- 2 attributes: `TROPOMI`, `OMI`

AOI mask Some pixels are outside of the AOI and are always NA, and should be removed from the `data.frame` so that it wouldn't be counted.

```
AOI_stars <- st_read("1_data/raw/Switzerland_shapefile/AOI_4326.shp") %>%
  st_rasterize(template = TROPOMI_fine) %>%
  slice("band" , 1) %>%
  mutate(AOI = ifelse(is.na(FID) , FALSE , TRUE))
```

```
## Reading layer `AOI_4326' from data source `/Masterarbeit/analysis/1_data/raw/Switzerland_shapefile/A
## Simple feature collection with 1 feature and 1 field
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 5.309117 ymin: 45.36734 xmax: 11.14935 ymax: 48.27325
## CRS:            4326
```

The `AOI` band in `AOI_stars` has `TRUE/FALSE` values to filter pixels that are outside the AOI. It will be used later in the summary statistics.

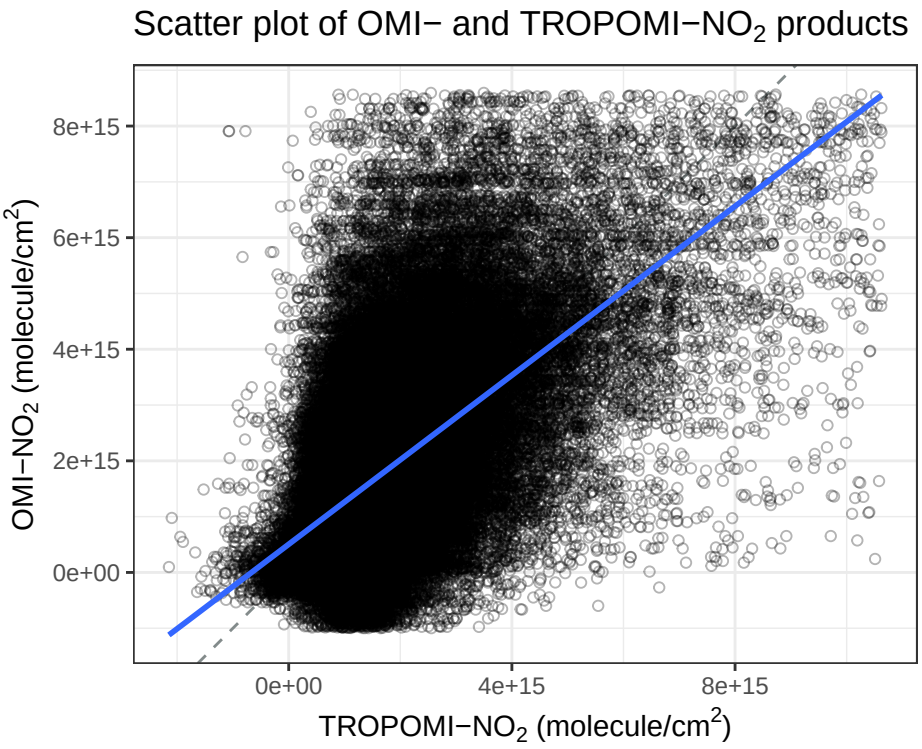
Scatter plot, simple linear regression

The scatter plot of OMI- and TROPOMI-measured NO2 values, with the extreme values (higher than the 99th quantile) excluded.

```
FULL %>%
  as.data.frame() %>%
  # exclude values > Q99
  filter(TROPOMI < quantile(TROPOMI , 0.99 , na.rm = TRUE)) %>%
  filter(OMI < quantile(OMI , 0.99 , na.rm = TRUE)) %>%
  # visualization
  ggplot(aes(x = TROPOMI , y = OMI)) +
  geom_abline(intercept = 0 , slope = 1 , color = "azure4" , linetype = 2) +
  geom_point(shape = 1 , alpha = 0.3) +
  geom_smooth(method = lm) +
  coord_fixed(1) +
  labs(title = expression("Scatter plot of OMI- and TROPOMI-NO"[2]*" products") ,
       x = expression("TROPOMI-NO"[2] * " (molecule/cm"^2*")") ,
```

```
       y = expression("OMI-NO"[2] * " (molecule/cm"^2*")")) +
  theme_bw()
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Fit a simple linear reression to the data to have a preliminary exploration of the relationship between the column NO2 values observed by OMI and TROPOMI.

```
lm_FULL <- FULL %>%
  as.data.frame() %>%
  # exclude values > Q99
  filter(TROPOMI < quantile(TROPOMI , 0.99 , na.rm = TRUE)) %>%
  filter(OMI < quantile(OMI , 0.99 , na.rm = TRUE)) %>%
  # scaling
  mutate(TROPOMI_scaled = scale(TROPOMI) ,
         OMI_scaled = scale(OMI)) %>%
  # fit linear regression
  lm(OMI_scaled ~ TROPOMI_scaled , data = .)
summary(lm_FULL)
```

```
##
## Call:
## lm(formula = OMI_scaled ~ TROPOMI_scaled, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8082 -0.4905 -0.1438  0.4048  4.8061
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -1.415e-15  2.521e-03     0.0        1
```

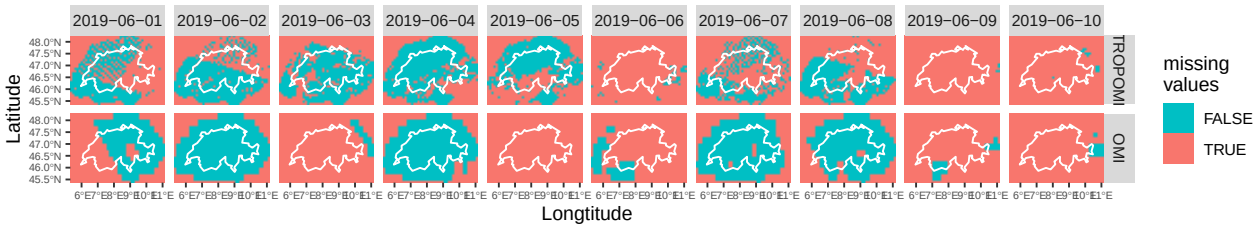
```
## TROPOMI_scaled  6.020e-01  2.521e-03   238.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7985 on 100326 degrees of freedom
## Multiple R-squared:  0.3623, Adjusted R-squared:  0.3623
## F-statistic: 5.701e+04 on 1 and 100326 DF,  p-value: < 2.2e-16
```

The linear relationship is not very strong between the pixel values of OMI and TROPOMI (R^2=0.36). The estimated intercept is close to 0, while the estimated slope is lower than 1 (0.602+-0.002), suggesting that the TROPOMI observes higher values compared to the co-locating OMI pixel cells. This makes sense, since the spatial resolution of TROPOMI cells are higher, and therefore local high values are more likely to be captured in the TROPOMI measurement and, on the other hand, might be "diluted" in the coarser OMI cells.

Cross table of missing-value pixels between TROPOMI and OMI

We also look at the distribution of missing-value pixels in the TROPOMI and OMI products. One might be interested in: when one pixel is missing value in one product, what is the proportion which the pixel is also missing value in the other product. For example in the following graph, where the two colors indicate the binary case of missing/non-missing value, we can see that the occurance of missing value is slightly different.

```
ggplot() +
  geom_stars(
    data = FULL %>%
      # counting NAs as two TRUE.FALSE bands
      mutate(TROPOMI = is.na(TROPOMI) ,
             OMI = is.na(OMI)) %>%
      filter(date %within% selected_dates) %>%
      merge()
  ) +
  geom_sf(data = CH , fill = NA , color = "white") +
  coord_sf(expand = FALSE) +
  scale_fill_discrete(direction = -1) +
  facet_grid(attributes ~ date) +
  theme(axis.text = element_text(size = 6)) +
  labs(x = "Longtitude" , y = "Latitude" , fill = "missing \nvalues")
```



For this question, a cross table is calculated to see the percentage of agreement between the two products.

```
# cross table
xtabs_NA <- FULL %>%
  # counting NAs as two TRUE.FALSE bands
  transmute(NA_TROPOMI = is.na(TROPOMI) ,
            NA_OMI = is.na(OMI)) %>%
  as.data.frame() %>%
  xtabs(~NA_TROPOMI + NA_OMI , data = .)
```

```
##            NA_OMI
## NA_TROPOMI   FALSE   TRUE
```

```
##       FALSE 102327  61198
##       TRUE   56623 293772
```

```
# percentage
(xtabs_NA/sum(xtabs_NA)) %>%
  round(3)
```

```
##           NA_OMI
## NA_TROPOMI FALSE  TRUE
##      FALSE 0.199 0.119
##      TRUE  0.110 0.572
```

The probability that a pixel has missing value in both products on the same day is 57.2%. On the other hand, it's 19.9% that a pixel has non-missing value in both products on the same day. The proportions of "one pixel being missing in one product yet not missing in the other product" are roughly the same for both products: around 11%. Overall, the *percent agreement* is 77.0%, *Kappa* is 0.54.

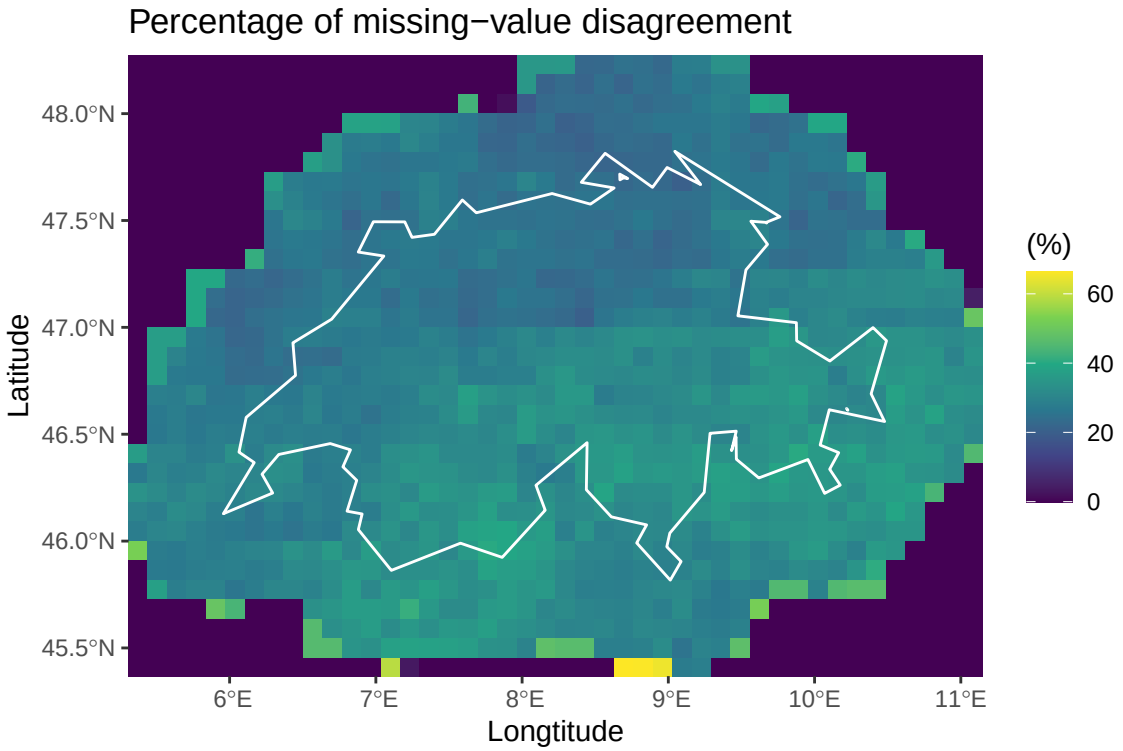
Disagreement per pixel

It's also interesting to look at the spatial distribution of the pixels where the missing value occurance disagrees more between the two products.

```
agree <- FULL %>%
  # counting NAs as two TRUE.FALSE bands
  mutate(NA_TROPOMI = is.na(TROPOMI) ,
         NA_OMI = is.na(OMI)) %>%
  # calculating disagreement between NA_TROPOMI and NA_OMI
  mutate(agree = ifelse(NA_TROPOMI , -1 , 1) * ifelse(NA_OMI , -1 , 1)) %>%  # 1 as agree; -1 as disagr
  select(agree)

# aggregate the whole year
agree_sum <- agree %>%
  mutate(agree_binary = ifelse(agree == 1 , 1 , 0) ,
         disagree_binary = ifelse(agree == -1 , 1 , 0)) %>%
  select(-agree) %>%
  st_apply(c("x","y") , sum) %>%  # sum value for each pixel (across dates)
  merge() %>%
  st_set_dimensions(3 , values = c("agree" , "disagree") , names = c("var"))
```

```
# percentage of missing-value disagreement
ggplot() +
  geom_stars(data = agree_sum[,,,2]/365*100) + # only plot disagree; calculate percentage
  geom_sf(data = CH , fill = NA , color = "white") +
  scale_fill_viridis_c() +
  coord_sf(expand = FALSE) +
  labs(x = "Longtitude" , y = "Latitude" , fill = "(%)" ,
       title = "Percentage of missing-value disagreement")
```



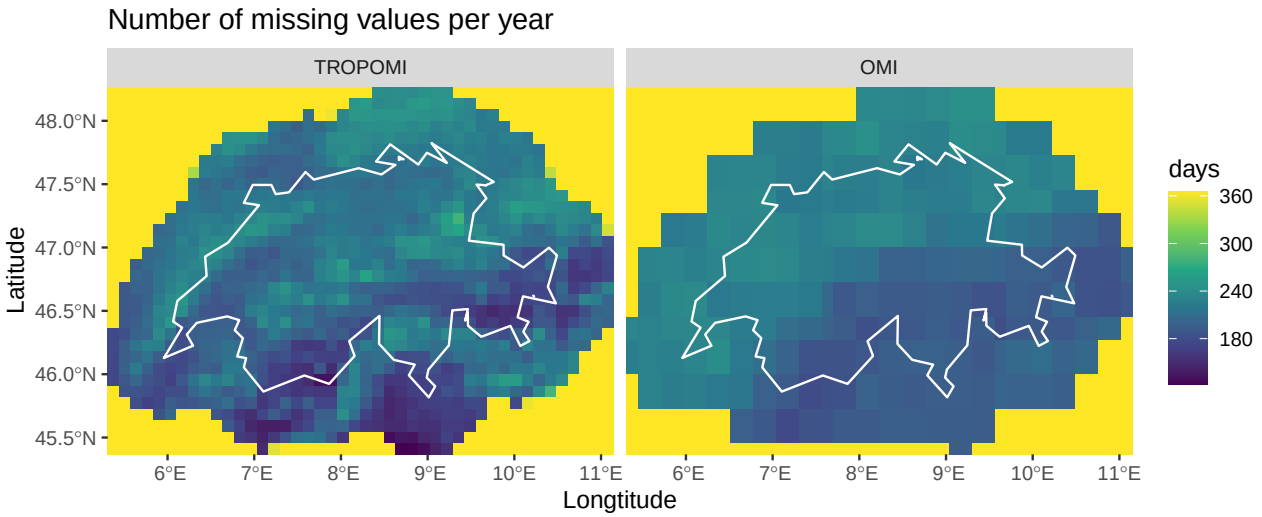
The figure above shows the **disagreeement** of missing-value occurance between the two products for each pixel of the whole year 2019: the lighter the color, the higher the disagreement (missing in one product while not-missing). Generally there's not much difference, with the disagreement being slightly lower in the northern Switzerland.

Spatial pattern of missing-value pixels

After comparing the consistency of the missing-value pattern between the two products, let's look at the distribution of the missing values itself. We summarized for each pixel the total number of days where the value is missing. It's shown in the graph below.

```
NA_sum <- FULL %>%
  # pixel=1 if value is NA, else 0
  transmute(NA_TROPOMI = ifelse(is.na(TROPOMI) , 1 , 0) ,
            NA_OMI = ifelse(is.na(OMI) , 1 , 0)) %>%
  # aggregate the whole year for each pixel: sum
  st_apply(c("x","y") , sum)
```

```
ggplot() +
  geom_stars(
    data = NA_sum %>%
      merge() %>%
      st_set_dimensions(3 , values = c("TROPOMI" , "OMI") , names = "source")
  ) +
  geom_sf(data = CH , fill = NA , color = "white") +
  scale_fill_viridis_c(breaks = seq(0,360,60)) +
  coord_sf(expand = FALSE) +
  facet_grid(~source) +
  labs(x = "Longtitude" , y = "Latitude" , fill = "days" ,
       title = "Number of missing values per year")
```



In general, areas with higher percentage of missing values (lighter color) can be observed mostly in central-south-eastern Switzerland.

Temporal pattern of missing-value pixels (per season)

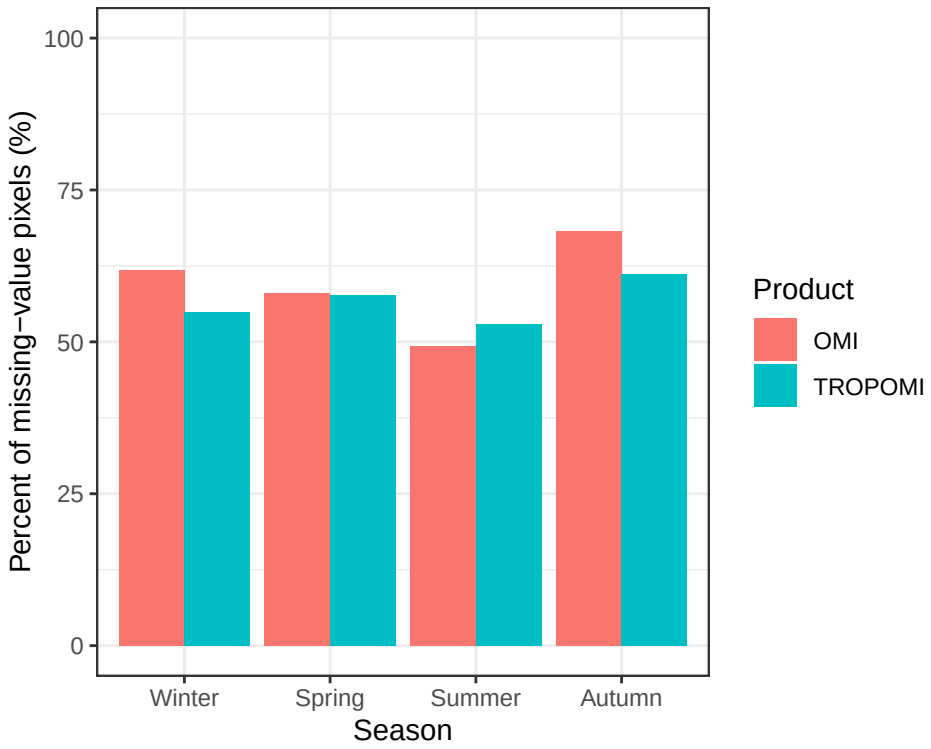
We can also calcuate the total number of pixel-day of missing values for each season to see the temporal variability of data availability.

```
NA_season <- FULL %>%
  # pixel=1 if value is NA, else 0
  transmute(NA_TROPOMI = ifelse(is.na(TROPOMI) , 1 , 0) ,
            NA_OMI = ifelse(is.na(OMI) , 1 , 0)) %>%
  # calculate season from the date dimension
  # and use season as the third dimension
  st_set_dimensions(3 ,
                    values = as.character(quarter(st_get_dimension_values(. , 3) , fiscal_start = 12)) ,
                    names = "season")
# summary statistics: proportion of missing values per season
NA_season_df <- NA_season %>%
  as.data.frame() %>%
  # remove the pixels that are outside AOI
  full_join(as.data.frame(AOI_stars) , by = c("x" , "y")) %>%
  filter(AOI) %>%
  select(-FID , -AOI) %>%
  # calculate the total missing-value pixels for each season
  pivot_longer(cols = starts_with("NA_") ,
               names_to = "product" , names_prefix = "NA_" , values_to = "is_NA") %>%
  group_by(season , product , is_NA) %>%
  summarize(count = n()) %>%
  ungroup() %>%
  # calculate proportion
  pivot_wider(names_from = is_NA , values_from = count) %>%
  rename(valid = `0` , missing = `1`) %>%
  mutate(missing_proportion = missing/(valid + missing))
```

```
## `summarise()` has grouped output by 'season', 'product'. You can override using the `.groups` argume
```

```
# visualization: bar chart
NA_season_df %>%
```

```
  ggplot(aes(x = season , y = missing_proportion , fill = product)) +
  geom_bar(stat = "identity" , position = "dodge") +
  scale_x_discrete(labels = c("Winter" , "Spring" , "Summer" , "Autumn")) +
  scale_y_continuous(breaks = seq(0,1,0.25) , labels = seq(0,1,0.25)*100 , limits = c(0,1)) +
  labs(x = "Season" , y = "Percent of missing-value pixels (%)" , fill = "Product") +
  theme_bw()
```



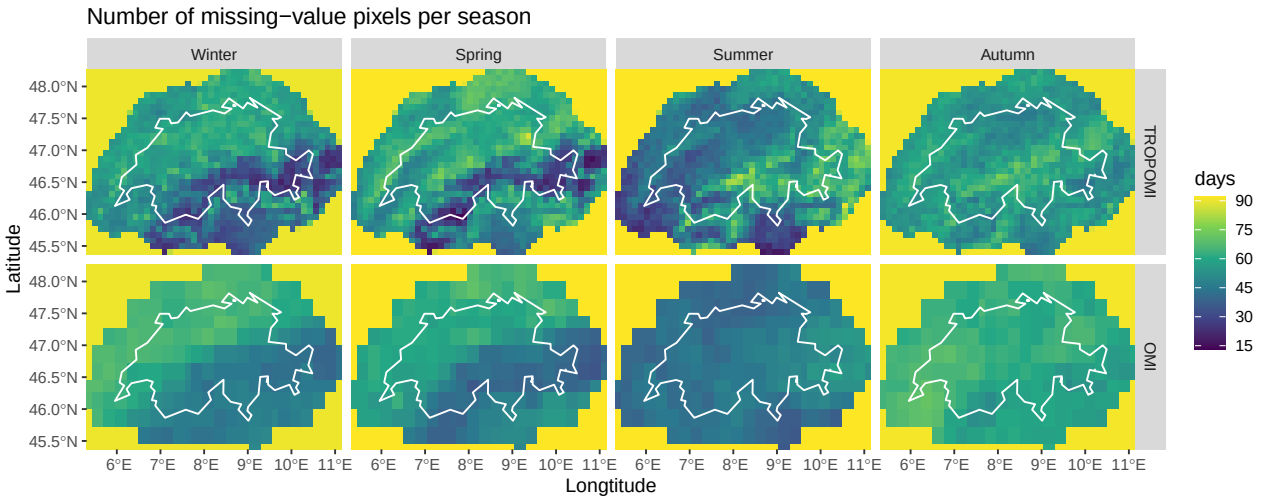
We can see that there are more missing values in autumn, and fewer in summer. However in general, more than 50% of pixel-day are missing, therefore imputation of the satellite-derived NO2 product is necessary before using it as input variables in the further modeling of ground-level concentrations.

Spatial-temporal distribution of missing values

Combining the previous two sessions, this graph summarizes both the spatial and temporal variability of data availability of the two satellite-derived NO2 products. The total number of days with unavailable measurement of each pixel in each season is calculated and demonstrated.

```
NA_season_sum <- c(
  # calculate the total number of missing-values per pixel for each season
  NA_season %>%
    filter(season == "1") %>%
    st_apply(c("x" , "y") , sum) %>%
    merge() %>%
    st_set_dimensions(3 , values = c("TROPOMI" , "OMI") , names = "source") ,
  NA_season %>%
    filter(season == "2") %>%
    st_apply(c("x" , "y") , sum) %>%
    merge() %>%
    st_set_dimensions(3 , values = c("TROPOMI" , "OMI") , names = "source") ,
  NA_season %>%
    filter(season == "3") %>%
```

```
    st_apply(c("x" , "y") , sum) %>%
    merge() %>%
    st_set_dimensions(3 , values = c("TROPOMI" , "OMI") , names = "source") ,
  NA_season %>%
    filter(season =="4") %>%
    st_apply(c("x" , "y") , sum) %>%
    merge() %>%
    st_set_dimensions(3 , values = c("TROPOMI" , "OMI") , names = "source") ,
  along = list("season" = c("Winter" , "Spring" , "Summer" , "Autumn") )
)
ggplot() +
  geom_stars(data = NA_season_sum) +
  geom_sf(data = CH , fill = NA , color = "white") +
  scale_fill_viridis_c(breaks = seq(0,90,15)) +
  coord_sf(expand = FALSE) +
  facet_grid(source~season) +
  labs(x = "Longtitude" , y = "Latitude" , fill = "days" ,
       title = "Number of missing-value pixels per season")
```



Elevation and data availability

Additionally we look at the relationship between altitude and data availability. The DEM data is loaded, resampled to TROPOMI grids, and classified into 4 levels (0-999m, 1000-1999m, 2000-2999m, 3000-3999m). The data availability (days of missing values per pixel per year) of each altitude level is summarized below as box-plots.

```
# load DEM
file_DEM <-"1_data/raw/EU-DEM-1.1/eu_dem_v11_E40N20_AOI.tif"
DEM_rs <- read_stars(file_DEM) %>%
  st_warp(dest = NA_sum) %>%
  setNames("altitude")
```

```
# altitude class
DEM_rs %>%
  # cut altitude into 4 classes
  mutate(altitude_class = cut(altitude ,
```

```
                              seq(0,4000,1000) ,
                              labels = as.character(0:3))) %>%
c(. , NA_sum) %>%
as.data.frame() %>%
filter(!is.na(altitude)) %>%
# NA pixel-day for each altitude class
pivot_longer(cols = starts_with("NA_") ,
             names_to = "product" , names_prefix = "NA_" , values_to = "nNA") %>%
ggplot(aes(x = altitude_class , y = nNA)) +
geom_boxplot() +
facet_grid(~product) +
scale_x_discrete(labels = c("0-999" , "1,000-1,999" , "2,000-2,999" , "3,000-3,999")) +
scale_y_continuous(limits = c(0,365)) +
coord_flip() +
labs(x = "Altitude (m)" , y = "Days" ,
     title = "Days of missing values per pixel per year") +
theme_bw()
```