The MOISCRUST dataset: a spatio-temporal continuous soil moisture dataset from a Mediterranean semiarid dryland from 2006 to 2020

SUPPLEMENTARY MATERIAL

Joaquín Moreno[1,2] Sergio Asensio[2] Miguel Berdugo[2,3]
Beatriz Gozalo[2] Victoria Ochoa[2] Blas M. Benito[2]
Fernando T. Maestre[2,4]

Contents

[1] Corresponding author, e-mail: joaquin.moreno@ua.es
[2] Instituto Multidisciplinar para el Estudio del Medio "Ramon Margalef", Universidad de Alicante, Edificio Nuevos Institutos, Carretera de San Vicente del Raspeig s/n, 03690 San Vicente del Raspeig, Spain.
[3] Institut Department of Environmental Systems Science, ETH Zürich. Universitätstrasse 16, 8092 Zurich, Switzerland.
[4] Departamento de Ecología, Universidad de Alicante, Carretera de San Vicente del Raspeig s/n, 03690 San Vicente del Raspeig, Alicante, Spain.

1 Summary

XXX

2 Installing and loading the required libraries

The following libraries are required to run this workflow. These are already installed in the local renv folder, so there is no need to install them if you don't have them in your computer.

```
#automatic install of packages if they are not readily available
required_packages <- c(
 "data.table",
 "janitor",
 "tidyverse",
 "kableExtra",
 "foreach",
 "doParallel",
 "readr",
 "writexl",
 "RSQLite",
 "zip",
 "knitr"
)

#loading packages
invisible(lapply(required_packages, library, character.only = TRUE))

#removing objects
rm(required_packages)
```

```
#knitr options
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
knitr::knit_hooks$set(document = function(x) {sub('\\usepackage{xcolor}', '\\usepackage{xcolor}', x, fixed = TRUE)})
```

3 Data loading and preparation

3.1 Loading the data

```
#load, clean names, and rename time columns
moiscrust <- data.table::fread(file = "moiscrust_raw.csv") %>%
 janitor::clean_names() %>%
 dplyr::rename(
  date = dates,
  time = hour
  ) %>%
 as.data.frame()
```

3.2 Formatting dates and times

```
#date to Year-month-day
moiscrust$date <- format(
 as.POSIXct(
  strptime(
   moiscrust$date,
   "%d/%m/%Y",
   tz = ""
   )
  ),
 format = "%Y-%m-%d"
 )

#time to Hour-Minute
moiscrust$time <- format(
 as.POSIXct(
  strptime(
   moiscrust$time,
   "%H:%M:%S",
   tz = ""
   )
  ),
```

```
 format = "%H:%M"
 )

#joining date and time
moiscrust$date_time <- as.POSIXct(
 paste(
  moiscrust$date,
  moiscrust$time
  ),
 format="%Y-%m-%d %H:%M"
 )

#unique id for each observation
moiscrust$date_time_id <- 1:nrow(moiscrust)

#creating year, month, and day related columns
moiscrust$year <- lubridate::year(moiscrust$date)
moiscrust$year_day <- lubridate::yday(moiscrust$date)
moiscrust$month <- lubridate::month(moiscrust$date)
moiscrust$month_day <- lubridate::mday(moiscrust$date)
moiscrust$week <- lubridate::week(moiscrust$date)
moiscrust$week_day <- lubridate::wday(moiscrust$date)
```

3.3 Reordering columns

```
#names of the sensors
sensors <- c(
 "retama5094",
 "retama5062",
 "retama5063",
 "stipa5094",
 "stipa5062",
 "stipa5063",
 "bs_cl5094",
 "bs_cl5062",
 "bs_cl5063",
 "bs_cm5094",
 "bs_cm5062",
 "bs_cm5063",
 "bs_ch5094",
 "bs_ch5062",
```

```
  "bs_ch5063"
)

#reordering columns of moiscrust to have time in the left side
moiscrust <- moiscrust[, c(
  "date_time",
  "date_time_id",
  "date",
  "time",
  "year",
  "year_day",
  "month",
  "month_day",
  "week",
  "week_day",
  sensors
)]
```

3.4 To long format

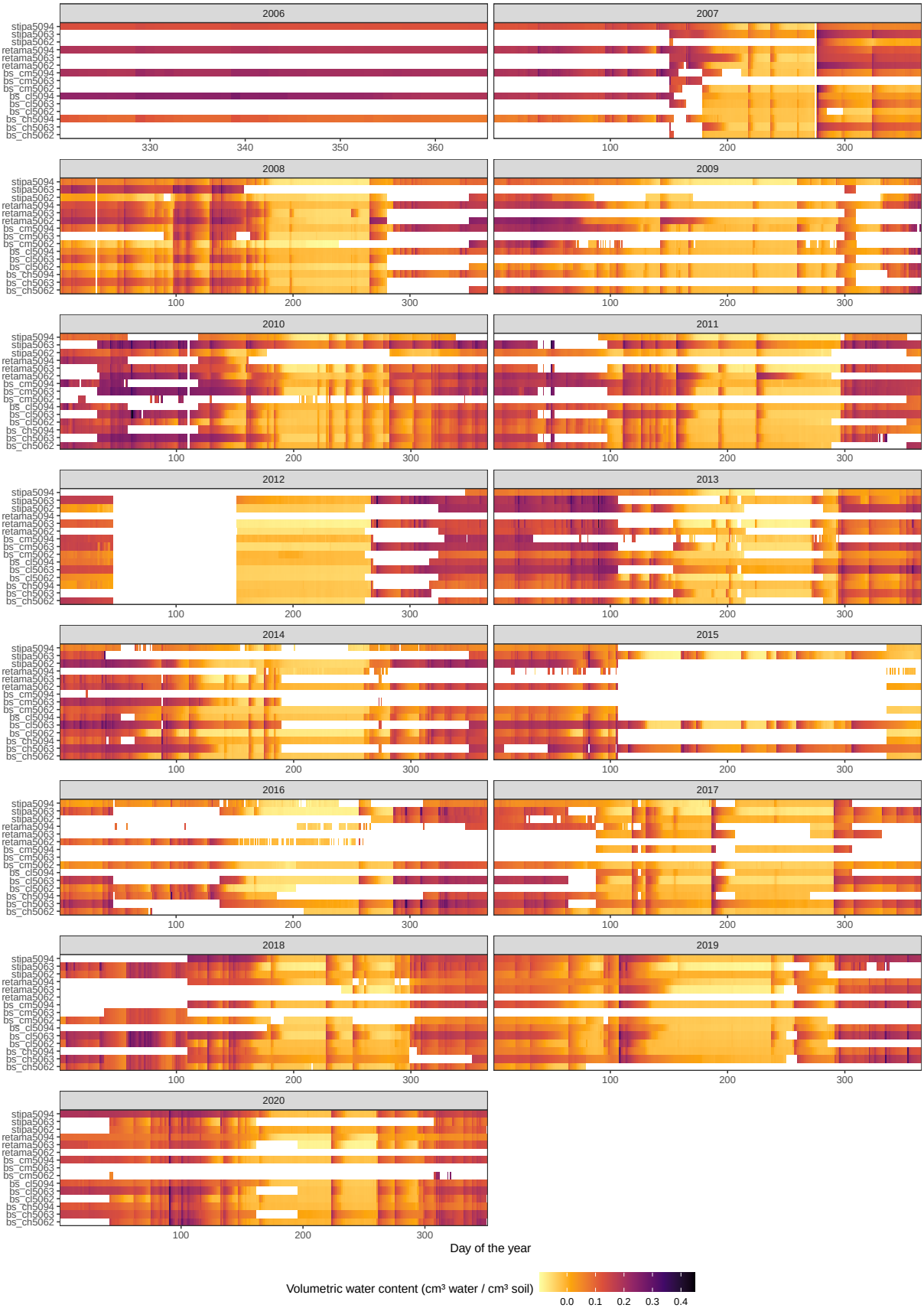
```
#to long format
moiscrust_long <- tidyr::pivot_longer(
  moiscrust,
  cols = all_of(sensors),
  names_to = "sensor",
  values_to = "soil_moisture"
)
```

3.5 Data visualization

```
ggplot(moiscrust_long) +
  facet_wrap("year", scales = "free_x", ncol = 2) +
  aes(x = year_day, y = sensor, fill = soil_moisture) +
  geom_tile() +
  coord_cartesian(expand = FALSE) +
  theme_bw() +
  scale_fill_viridis_c(direction = -1, na.value = "white", option = "B") +
  theme(legend.position = "bottom") +
  ylab("") +
  xlab("Day of the year") +
```

```
ggtitle("MOISCRUST database (raw data)") +
labs(fill = expression("Volumetric water content (cm³ water / cm³ soil)")) +
theme(legend.key.width = unit(1, "cm"))
```

MOISCRUST database (raw data)



```
ggsave(width = 12, height = 17, filename = "MOISCRUST_raw.pdf")
```

3.6 Number of NA per time-series

```
#counting NA values per sensor
moiscrust_NA <- moiscrust_long %>%
 group_by(sensor) %>%
 summarise(na_count = sum(is.na(soil_moisture))) %>%
 mutate(na_count_percent = round((na_count * 100) / nrow(moiscrust), 1))

#adding sensor sensor_group to the moiscrust_NA data frame
moiscrust_NA$sensor_group <- c(
 "biocrust_high",
 "biocrust_high",
 "biocrust_high",
 "biocrust_low",
 "biocrust_low",
 "biocrust_low",
 "biocrust_medium",
 "biocrust_medium",
 "biocrust_medium",
 "retama",
 "retama",
 "retama",
 "stipa",
 "stipa",
 "stipa"
)

#reordering columns and arranging by na_count
moiscrust_NA <- moiscrust_NA[, c(
 "sensor",
 "sensor_group",
 "na_count",
 "na_count_percent"
 )] %>%
 dplyr::arrange(na_count) %>%
 as.data.frame()

#showing the table
kableExtra::kbl(moiscrust_NA)
```

sensor	sensor_group	na_count	na_count_percent
bs_ch5094	biocrust_high	7674	16.5
stipa5094	stipa	10553	22.7
bs_cl5094	biocrust_low	10987	23.6
bs_cl5063	biocrust_low	11114	23.9
bs_cl5062	biocrust_low	11188	24.1
bs_ch5062	biocrust_high	11776	25.3
bs_ch5063	biocrust_high	14125	30.4
stipa5062	stipa	15363	33.0
stipa5063	stipa	15604	33.5
bs_cm5094	biocrust_medium	16934	36.4
bs_cm5062	biocrust_medium	18743	40.3
retama5063	retama	20505	44.1
retama5094	retama	22840	49.1
retama5062	retama	22999	49.4
bs_cm5063	biocrust_medium	31225	67.1

4 Developing criteria to find candidates for gap filling

```
#combining sensors in pairs x-y
sensors_pairs <- combn(
  x = sensors,
  m = 2
) %>%
  t() %>%
  as.data.frame()

#adding combinations y-x so all pairs have both directions
#removing repeated pairs
#joining with moiscrust_NA to get sensor groups
#add column same_sensor_group to check if x and y are or not in the same sensor group
#add empty columns to store % of shared data, model's R squared, and model ID
sensors_pairs <- sensors_pairs %>%
  rbind(
    data.frame(
      V1 = sensors_pairs$V2,
      V2 = sensors_pairs$V1
    )
  ) %>%
  distinct(
```

```
  V1,
  V2,
  .keep_all = TRUE
 ) %>%
 left_join(
  moiscrust_NA[, c("sensor", "sensor_group")],
  by = c("V1" = "sensor")
 ) %>%
 left_join(
  moiscrust_NA[, c("sensor", "sensor_group")],
  by = c("V2" = "sensor")
 ) %>%
 rename(
  y = V1,
  x = V2,
  sensor_group_y = sensor_group.x, #not a mistake
  sensor_group_x = sensor_group.y, #not a mistake
 ) %>%
 mutate(
  same_sensor_group = ifelse(
   sensor_group_x == sensor_group_y,
   TRUE,
   FALSE
   ),
  valid_cases_shared_percent = NA,
  sensors_r_squared = NA,
  model_id = row_number()
 )

#list to store models
sensors_pairs_models <- list()

#looping through sensors pairs to:
#fit lm model y ~ x and save it in sensors_pairs_models
#
for(i in 1:nrow(sensors_pairs)){

 #names of the sensors y and x
 y_i <- sensors_pairs[i, "y"]
 x_i <- sensors_pairs[i, "x"]
```

```
  #data of the sensor pair
  sensor_pair_i <- moiscrust[, c(y_i, x_i)]

  #complete cases of the sensor pair
  sensor_pair_i <- sensor_pair_i[complete.cases(sensor_pair_i), ]

  #common cases
  sensors_pairs[i, "valid_cases_shared_percent"] <-
    nrow(sensor_pair_i) / nrow(moiscrust) * 100

  #R squared of the sensor pair
  sensors_pairs[i, "sensors_r_squared"] <- cor(
    sensor_pair_i[, 1],
    sensor_pair_i[, 2]
    )

  #model formula y ~ x
  formula_i <- as.formula(paste(y_i, "~", x_i))

  #linear model
  sensors_pairs_models[[i]] <- lm(
    formula = formula_i,
    data = sensor_pair_i
  )

}

#selection score to find candidates during gap filling
#(sensors_r_squared * 100) +
#valid_cases_shared_percent +
#same_sensor_group (TRUE = 100, FALSE = 0)
sensors_pairs <- mutate(
  sensors_pairs,
  selection_score =
    (sensors_r_squared * 100) +
    valid_cases_shared_percent +
    ifelse(same_sensor_group == TRUE, 100, 0)
)

#removing objects we don't need
rm(
```

```
  sensor_pair_i,
  formula_i,
  i,
  x_i,
  y_i
)

#garbage collection
invisible(gc())
```

5 Imputation of missing data

```
#creating data frame of predictors
x <- moiscrust[, sensors]

#creating data frame to store model results
y <- matrix(
  data = NA,
  nrow = nrow(moiscrust),
  ncol = 12
  ) %>%
  as.data.frame()

#new colnames
colnames(y) <- c(
  "interpolated",
  "model_estimate",
  "model_ci_lower",
  "model_ci_upper",
  "model_predictor",
  "same_sensor_group",
  "sensors_r_squared",
  "valid_cases_shared_percent",
  "selection_score",
  "date_time_id",
  "sensor",
  "sensor_group"
  )

#transferring time id
```

```
y[, "date_time_id"] <- moiscrust[, "date_time_id"]
y[, "interpolated"] <- FALSE
```

5.1 Data imputation, step by step

The steps to fill the gaps in the MOISCRUST database go as follows:

1. A given sensor name is selected: "stipa5063"

```
sensor = "stipa5063"
```

2. The sensors pairs from the table sensors_pairs where the selected sensor is y (the response variable) are selected.

```
sensors_pair <- sensors_pairs %>%
  dplyr::filter(y == sensor)
```

y	x	sensor_group_y	sensor_group_x	same_sensor_group	valid_cases_share
stipa5063	bs_cl5094	stipa	biocrust_low	FALSE	
stipa5063	bs_cl5062	stipa	biocrust_low	FALSE	
stipa5063	bs_cl5063	stipa	biocrust_low	FALSE	
stipa5063	bs_cm5094	stipa	biocrust_medium	FALSE	
stipa5063	bs_cm5062	stipa	biocrust_medium	FALSE	
stipa5063	bs_cm5063	stipa	biocrust_medium	FALSE	
stipa5063	bs_ch5094	stipa	biocrust_high	FALSE	
stipa5063	bs_ch5062	stipa	biocrust_high	FALSE	
stipa5063	bs_ch5063	stipa	biocrust_high	FALSE	
stipa5063	retama5094	stipa	retama	FALSE	
stipa5063	retama5062	stipa	retama	FALSE	
stipa5063	retama5063	stipa	retama	FALSE	
stipa5063	stipa5094	stipa	stipa	TRUE	
stipa5063	stipa5062	stipa	stipa	TRUE	

3. The first row of the dataset x is selected.

```
x_row <- x[1, ]
```

retama5094	retama5062	retama5063	stipa5094	stipa5062	stipa5063	bs_cl5094	bs_cl5062
0.197	NA	NA	0.132	NA	NA	0.232	NA

3.1. If there is a recorded value of soil moisture for the sensor "stipa5063", goes to the next row, until there is a row with NA.

4. If there is an empty value, the potential candidate predictors are selected from the row by removing sensors with NA, and the target sensor.

```
predictor_candidates <- as.vector(x[1, ])
predictor_candidates <- predictor_candidates[which(
    !is.na(predictor_candidates) &
      names(predictor_candidates) != sensor
    )]
```

retama5094	stipa5094	bs_cl5094	bs_cm5094	bs_ch5094
0.197	0.132	0.232	0.205	0.121

5. From these predictors, the one with the highest selection score is selected from sensors_pair generated in the step **2.**.

```
best_predictor <- sensors_pair %>%
 dplyr::filter(x %in% names(predictor_candidates)) %>%
 dplyr::arrange(desc(selection_score)) %>%
 dplyr::slice(1)
```

y	x	sensor_group_y	sensor_group_x	same_sensor_group	valid_cases_shared_p
stipa5063	stipa5094	stipa	stipa	TRUE	48.

6. The model to use, stored in the list sensors_pairs_models, is selected from the value best_predictor$model_id, and used to predict a value for the empty cell.

```
predict(
 object = sensors_pairs_models[[best_predictor$model_id]],
 newdata = x_row,
 se.fit = TRUE,
 type = "response",
 interval = "confidence"
 )$fit
```

```
##       fit      lwr      upr
## 1 0.1435911 0.1419916 0.1451907
```

7. These values and others available in best_predictor are transferred to the same row in the matrix y.

8. Once all the sensors and rows have been processed this way, the matrix y is joined with moiscrust_long, and its interpolated values are transferred to the soil_moisture column, along with other columns indicating the quality of the interpolation.

5.2 Applying the algorithm to the complete dataset

The code applies the algorithm explained above to every sensor and row. Sensors are processed in parallel to speed up the gap-filling operation.

```
#setup for parallel execution
if(.Platform$OS.type == "windows"){
  temp_cluster <- parallel::makeCluster(
    parallel::detectCores() - 1,
    type = "PSOCK"
  )
} else {
  temp_cluster <- parallel::makeCluster(
    parallel::detectCores() - 1,
    type = "FORK"
  )
}
doParallel::registerDoParallel(cl = temp_cluster)

#parallelized loop (each sensor is processed in one separated thread)
moiscrust_interpolation <- foreach::foreach(
  sensor_i = sensors
) %dopar% {

  #subset sensors_pairs
  sensors_pair_i <- sensors_pairs %>%
    dplyr::filter(y == sensor_i)

  #scanning the rows of x one by one
  for(row_i in 1:nrow(x)){

    #if is not NA, next iteration
    if(!is.na(x[row_i, sensor_i])){next}

    #getting target row row
    x_row_i <- x[row_i, ]

    #getting predictor candidates available in x_row_i
    predictor_candidates_i <- as.vector(x_row_i)
    predictor_candidates_i <- predictor_candidates_i[which(
      !is.na(predictor_candidates_i) &
        names(predictor_candidates_i) != sensor_i
      )]

    #selecting the predictor candidate with the best selection_score score
    best_predictor_i <- sensors_pair_i %>%
```

```
    dplyr::filter(x %in% names(predictor_candidates_i)) %>%
    dplyr::arrange(desc(selection_score)) %>%
    dplyr::slice(1)

  #if there is no best candidate available, next iteration
  if(nrow(best_predictor_i) == 0){next}

  #compute estimates with the model of the best predictor
  y[row_i, c(
    "model_estimate",
    "model_ci_lower",
    "model_ci_upper"
    )] <- predict(
      object = sensors_pairs_models[[best_predictor_i$model_id]],
      newdata = x_row_i,
      se.fit = TRUE,
      type = "response",
      interval = "confidence"
      )$fit

  #adding interpolation flag
  y[row_i, "interpolated"] <- TRUE
  y[row_i, "model_predictor"] <- best_predictor_i$x
  y[row_i, "sensors_r_squared"] <- best_predictor_i$sensors_r_squared
  y[row_i, "selection_score"] <- best_predictor_i$selection_score
  y[row_i, "valid_cases_shared_percent"] <- best_predictor_i$valid_cases_shared_percent
  y[row_i, "sensor_group"] <- best_predictor_i$sensor_group_y
  y[row_i, "same_sensor_group"] <- best_predictor_i$same_sensor_group

 }

 #adding sensor_i name
 y[, "sensor"] <- sensor_i

 return(y)

}

#stop cluster
parallel::stopCluster(temp_cluster)
```

```
#removing loop objects
rm(
  x,
  y,
  temp_cluster
)
```

To data frame and joining with moiscrust_long

```
#naming the output
names(moiscrust_interpolation) <- sensors

#to data frame
moiscrust_interpolation_long <- do.call(
  "rbind",
  moiscrust_interpolation
)

#joining with moiscrust_long
moiscrust_long <- dplyr::left_join(
  moiscrust_long,
  moiscrust_interpolation_long,
  by = c("date_time_id", "sensor")
)

#transferring estimates to the soil_moisture column
moiscrust_long$soil_moisture <- ifelse(
  is.na(moiscrust_long$soil_moisture),
  moiscrust_long$model_estimate,
  moiscrust_long$soil_moisture
)

#adding a interpolation_quality flag following the criteria in the paper
moiscrust_long$interpolation_quality <- ifelse(
  moiscrust_long$sensors_r_squared > 0.85 &
  moiscrust_long$valid_cases_shared_percent > 20,
  "acceptable",
  "poor"
)

#filling NA with "observation"
moiscrust_long[
```

```
 is.na(moiscrust_long$interpolation_quality), "interpolation_quality"
 ] <- "observation"

#adding NA where there are no values
moiscrust_long[is.na(moiscrust_long$soil_moisture), "interpolation_quality"] <- NA

#computing number of NA cases again
moiscrust_NA <- moiscrust_long %>%
 group_by(sensor) %>%
 summarise(na_count_after = sum(is.na(soil_moisture))) %>%
 mutate(na_count_percent_after = round((na_count_after * 100) / nrow(moiscrust), 1)) %>%
 left_join(
  y = moiscrust_NA,
  by = "sensor"
 ) %>%
 transmute(
  sensor,
  na_count_before = na_count,
  na_count_after,
  na_count_percent_before = na_count_percent,
  na_count_percent_after
 )

#removing moiscrust_interpolation
rm(moiscrust_interpolation)
```

The interpolation has removed all gaps where there was a value to interpolate from.

```
kableExtra::kbl(
 moiscrust_NA,
 col.names = c(
  "Sensor",
  "NA before interpolation",
  "NA after interpolation",
  "NA % before interpolation",
  "NA % after interpolation"
  )
 )
```

Sensor	NA before interpolation	NA after interpolation	NA % before interpolation	NA % after
bs_ch5062	11776	989	25.3	
bs_ch5063	14125	989	30.4	
bs_ch5094	7674	989	16.5	
bs_cl5062	11188	989	24.1	
bs_cl5063	11114	989	23.9	
bs_cl5094	10987	989	23.6	
bs_cm5062	18743	989	40.3	
bs_cm5063	31225	989	67.1	
bs_cm5094	16934	989	36.4	
retama5062	22999	989	49.4	
retama5063	20505	989	44.1	
retama5094	22840	989	49.1	
stipa5062	15363	989	33.0	
stipa5063	15604	989	33.5	
stipa5094	10553	989	22.7	

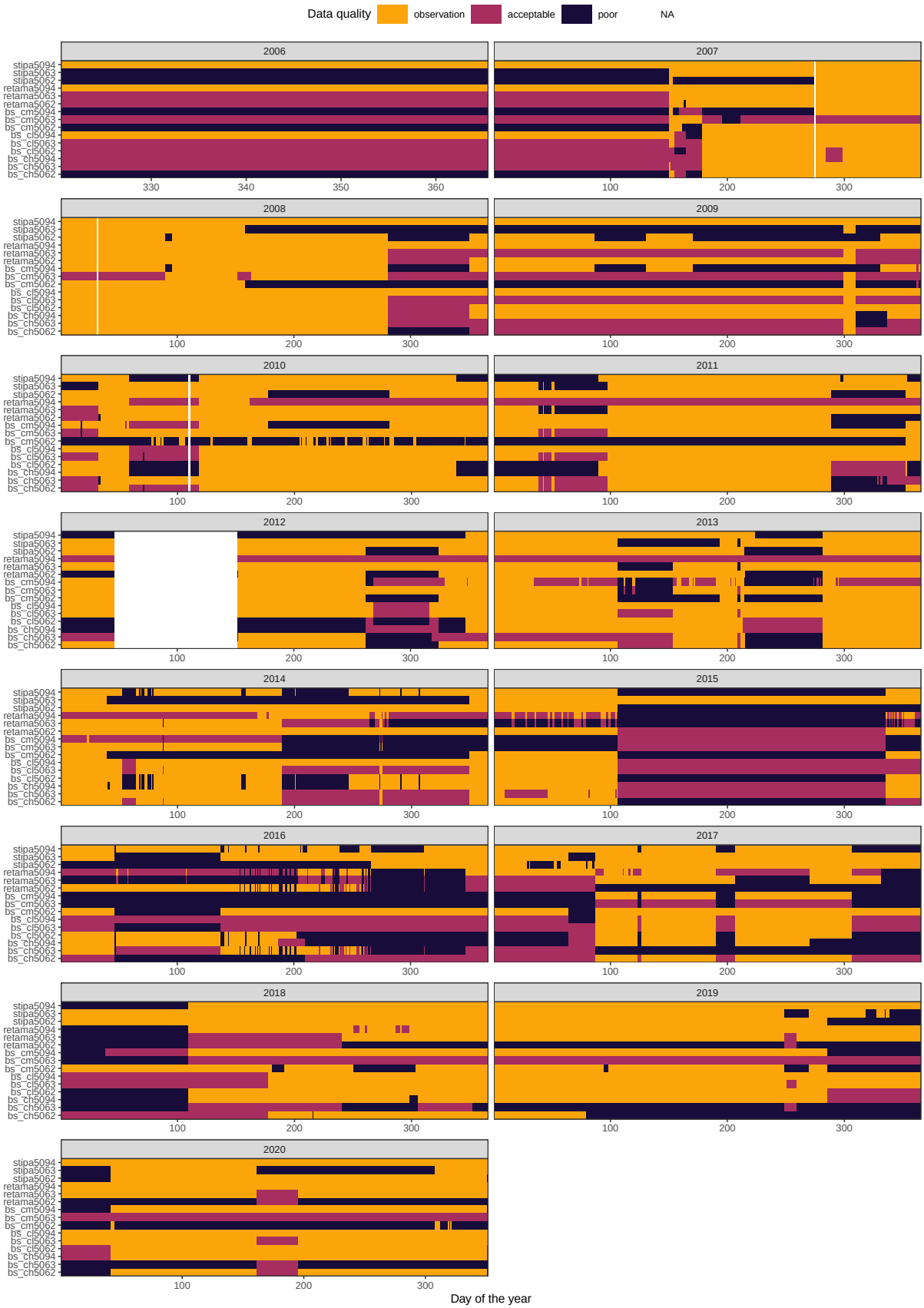
6 Visualizing the interpolated time series

Quality of the interpolation

```
ggplot(moiscrust_long) +
 facet_wrap(
  "year",
  scales = "free_x",
  ncol = 2
 ) +
 aes(
  x = year_day,
  y = sensor,
  fill = factor(
   interpolation_quality,
   levels = c(
    "observation",
    "acceptable",
    "poor"
    )
   )
 ) +
 geom_tile() +
 coord_cartesian(expand = FALSE) +
```

```
theme_bw() +
scale_fill_viridis_d(
  direction = -1,
  begin = 0.1,
  end = 0.8,
  na.value = "white",
  option = "B"
  ) +
theme(legend.position = "top") +
ylab("") +
xlab("Day of the year") +
ggtitle("MOISCRUST database (data quality)") +
labs(
  fill = expression("Data quality")) +
theme(legend.key.width = unit(1, "cm"))
```

MOISCRUST database (data quality)

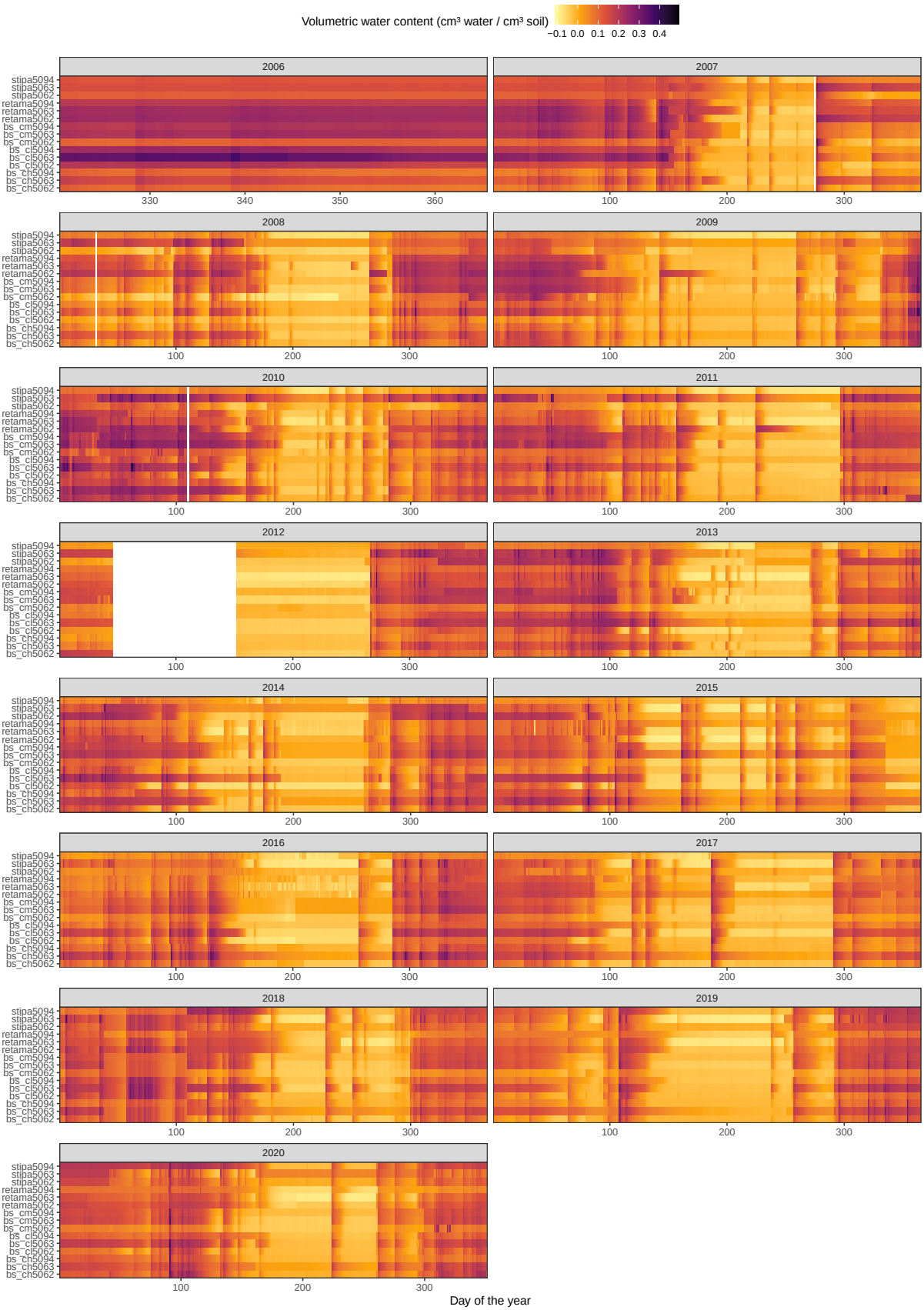


```
ggsave(
 width = 12,
 height = 17,
 filename = "MOISCRUST_interpolated_quality.pdf"
 )
```

The r squared between the predicted sensor values and the predictor are shown with transparency. Interpolated records with higher transparency may have lower quality.

```
ggplot(moiscrust_long) +
 facet_wrap(
  "year",
  scales = "free_x",
  ncol = 2
  ) +
 aes(
  x = year_day,
  y = sensor,
  fill = soil_moisture
  ) +
 geom_tile() +
 coord_cartesian(expand = FALSE) +
 theme_bw() +
 scale_fill_viridis_c(
  direction = -1,
  na.value = "white",
  option = "B"
  ) +
 theme(legend.position = "top") +
 ylab("") +
 xlab("Day of the year") +
 ggtitle("MOISCRUST database (observed and interpolated records)") +
 labs(fill = expression("Volumetric water content (cm³ water / cm³ soil)")) +
 theme(legend.key.width = unit(0.8, "cm"))
```

MOISCRUST database (observed and interpolated records)



```r
ggsave(width = 12, height = 17, filename = "MOISCRUST_interpolated.pdf")
```

7 Data base formatting

The dataset moiscrust_long is the database in long format already.

date_time	date_time_id	date	time	year	year_day	month	month_day	wee
2006-11-17 18:00:00	1	2006-11-17	18:00	2006	321	11	17	4
2006-11-17 18:00:00	1	2006-11-17	18:00	2006	321	11	17	4
2006-11-17 18:00:00	1	2006-11-17	18:00	2006	321	11	17	4
2006-11-17 18:00:00	1	2006-11-17	18:00	2006	321	11	17	4
2006-11-17 18:00:00	1	2006-11-17	18:00	2006	321	11	17	4
2006-11-17 18:00:00	1	2006-11-17	18:00	2006	321	11	17	4

Its columns are:

- date_time: date and time in POSIX format.
- date_time_id: integer, unique ID for each value of date_time.
- date: date in format year-month-day.
- time: time in format hour-minute.
- year: integer, year.
- year_day: integer, day of the year.
- month: integer, month number.
- week: integer, week of the year.
- week_day: integer, day of the week.
- sensor: character, sensor name.
- soil_moisture: numeric, soil moisture value in $m^3 water/m^3 soil$.
- interpolated: boolean, TRUE for interpolated records and FALSE for observations.
- model_estimate: numeric, prediction of the linear model.
- model_ci_lower: numeric, lower bound of the confidence interval of the estimate.
- model_ci_upper: numeric, upper bound of the confidence interval of the estimate.
- model_predictor: character, name of the sensor used as predictor in the linear model.
- same_sensor_group: boolean, TRUE if the sensor and its predictor are in the same group ("stipa", "retama", "biocrust_low", "biocrust_medium", "biocrust_high").
- sensors_r_squared: numeric, R squared between sensor and model_predictor.
- valid_cases_shared_percent: numeric, percentage of shared valid cases between sensor and model_predictor, taking the total number of values in date_time_id as reference.
- selection_score: numeric, value used to select the model_predictor, based on the sum of same_sensor_group (100 if TRUE and 0 if FALSE), sensors_r_squared multiplied by 100, and valid_cases_shared_percent.
- interpolation_quality: character, with the values "observation" for observations, "ac-

ceptable" for interpolated values where sensors_r_squared is higher than 0.85 and valid_cases_shared_percent is higher than 20, and "poor" for interpolated values below at least one of these thresholds.

8 Saving the data base in different formats

To expand its usability as much as possible, we provide the data in four different formats: .RData, .csv, .xlsx, and SQLite.

```
#create directory for the database
dir.create("database")

#rename dataset
moiscrust <- moiscrust_long

#save as RData
save(
  moiscrust,
  file = "database/moiscrust.RData"
  )

#save as csv
write_excel_csv(
  x = moiscrust,
  path = "database/moiscrust.csv"
)

#save as excel file
write_xlsx(
  x = moiscrust,
  path = "database/moiscrust.xlsx"
)

#save as SQLite
drv <- dbDriver("SQLite")
tfile <- "database/moiscrust.db"
con <- dbConnect(drv, dbname = "database/moiscrust.db")
dbWriteTable(con, "moiscrust", moiscrust, overwrite = TRUE)
dbDisconnect(con)

#compressing the file
```

```
zip(
  zipfile = ”database.zip”,
  files = ”database”
  )
```