Supplementary material for Buerkert et al. 2020

Eduardo Fernandez[a], Andreas Buerkert[b], Beke Tietjen[b], and Eike Luedeling[a]

[a] Department of Horticultural Sciences, Institute of Crop Science and Resource Conservation (INRES), Auf dem Hügel 6, D-53121 Bonn, Germany
[b] Organic Plant Production and Agroecosystems Research in the Tropics and Subtropics (OPATS), University of Kassel, Steinstrasse 19, D-37213 Witzenhausen, Germany

Forecasting future temperatures from past observations

Climate change is expected to affect many agricultural systems and to change their structure. Deciduous fruit tree orchards, which require cold temperatures during winter to overcome the trees' dormancy stage, are very sensitive to global warming, because increasing temperatures may reduce the amount of winter chill they receive. In this document, we show a reproducible example of a methodology to simulate chill for future scenarios based on past temperature records and climate model projections for oasis systems in the western Hajar Mountains of Oman. We used this methodology to produce a scientific manuscript published in the journal **Climatic Change**. The work titled **Revisiting climate change effects on winter chill in mountain oases of northern Oman** was authored by Andreas Buerkert, Eduardo Fernandez, Beke Tietjen, and Eike Luedeling.

As a primary source of data, we used temperature observations recorded with data loggers located in three mountain oases in the Al Muyadin watershed of Al Jabal Al Akhdar, spanning an elevation gradient between 1030 and 1900 m above sea level. In this example, we use data from Qasha', an oasis in the middle of this gradient, at 1640 m a.s.l..

First steps

First, we install and/or load the required libraries. To install a library from CRAN use the command `install.packages()` and specify the library name between quotes. This step is avoided if the libraries are already installed.

```r
install.packages(c("chillR", "dplyr", "tidyr","devtools"))
```

After installation, the specific libraries must be loaded into R for further use. This is done via the `library()` command using the library name without quotes.

```r
library(chillR)
library(dplyr)
library(tidyr)
```

Import the data from Qasha'

We can use the `read.csv()` function to import the data recorded with the data loggers at this site. The path of the file will be different for each user, so make sure you download the data we provided and save the file in an easy-to-find location.

```r
qasha_raw <- read.csv("temp_dataloggers/qasha_2004_2019.csv", sep = ";")
```

The complete period of measurements in this data set extended from 30.10.2004 to 15.02.2019. However, some periods are missing due to battery failures or other issues. We therefore need to compute the percentage of the period that was effectively recorded.

Since we know the time step used to record temperatures (i.e. 30 minutes) and the beginning and end of the intended period, we can compute the targeted number of records in this time frame.

First, we can compute the number of complete days covered between the installation of the data logger and the time of the final read-out.

```
dates_00 <- qasha_raw[which(qasha_raw$Time == "00:00:00"), "Date"]

min_day <- min(as.Date(dates_00, format = "%d.%m.%Y"))

max_day <- max(as.Date(dates_00, format = "%d.%m.%Y")) - 1

days <- max_day - min_day
```

The code above determined an intended period of 5,219 days for temperature recording. Knowing the total number of days and the frequency of recording, we can compute the intended number of measurements. This can be obtained multiplying 5,219 days by 48 measurements per day (i.e. 2 measurements each hour).

```
measurements <- as.numeric(days) * 48
```

This resulted in a total of 250,512 intended measurement points on complete days during the measurement period. To this number, we need to add measurements taken on the day of installation and the read-out day, for both of which daily records are not complete.

```
measurements_before <- nrow(qasha_raw[which(as.Date(qasha_raw$Date,
                                                    format = "%d.%m.%Y") < min_day), ])

measurements_after <- nrow(qasha_raw[which(as.Date(qasha_raw$Date,
                                                   format = "%d.%m.%Y") > max_day - 1), ])

measurements <- measurements + measurements_after + measurements_before
```

Since the data logger only recorded data for the period when it was active, the number of rows in `qasha_raw` can be assumed as the effectively recorded data. The percentage of complete data can be computed as follow:

```
percentage <- (nrow(qasha_raw) / measurements) * 100
```

This resulted in 62.3 % of data complete at this site between 30.10.2004 and 15.02.2019.

Weather data processing

Data logger formats can be hard to process and cumbersome to handle during further analyses. We therefore reformatted the weather data into `chillR` format for compatibility with `chillR` functions to be applied later. This format requires specific columns for the variables `Year`, `Month`, `Day`, `Hour`, and `Temp`.

```
qasha_all_data <- data.frame(Year = as.numeric(substr(qasha_raw$Date, 7, 10)),
                             Month = as.numeric(substr(qasha_raw$Date, 4, 5)),
                             Day = as.numeric(substr(qasha_raw$Date, 1, 2)),
                             Hour = as.numeric(substr(qasha_raw$Time, 1, 2)),
                             Time = qasha_raw$Time,
                             Temp = qasha_raw$Temperature...C.)
```

After formatting, we summarized the weather data into daily extreme records using functions from the `dplyr` library. This allowed further comparison with the secondary source of weather data (i.e. the Saiq weather station).

```
qasha <- qasha_all_data %>% group_by(Year, Month, Day) %>% summarise(Tmin = min(Temp),
                                                                    Tmax = max(Temp))
```

Since the data for the winter season 2018/2019 finished on 15.02.2019 we decided to only consider data up to 2018 considering the complete winter season of 2017/2018 as the last one.

```
qasha <- as.data.frame(filter(qasha, Year <= 2018))
```

Climate-related assessments usually require weather records of a considerable length to allow meaningful analysis. Our primary *in-situ* weather data set only consisted of about 14 years of records. We considered this time frame insufficient for forecasting future temperatures but adequate for calibrating a long-term weather data set from a nearby weather station. We therefore looked for such a weather station with long-term records. We used the function `handle_gsod()` from `chillR` to list the 25 closest weather stations to the location of interest. This function uses information from the Global Summary of the Day data base.

```
stat_list <- handle_gsod(action = "list_stations", location = c(x = 57.7, y = 23))
```

The function identified the weather station SAIQ located 10.11 km from the position specified as target location as the closest source of auxiliary temperature data. This weather station started recording maximum and minimum temperatures in 1983 and recorded the most recent data in 2019. We downloaded the weather data via the same `handle_gsod()` function using a different setting for the `action` parameter (i.e. “download_weather”). Note that this process may take a few minutes.

```
gw <- handle_gsod(action = "download_weather", location = "412540_99999",
                  time_interval = c(1983, 2019),
                  station_list = stat_list)
```

The output produced by this function consists of a list of 2 elements that are difficult to process. To convert the data into a useful format, we used the function `handle_gsod()` in its third `action` mode.

```
weather <- handle_gsod(gw)$weather
```

The data frame above contains a number of missing rows due to missing observations. These rows are filled with `NA` values. To generate a weather data frame containing all values for the `Year`, `Month` and `Day` columns we used the `make_all_day_table()` function from `chillR`. This function fills in the values for the columns mentioned above but adds `NA` values in the columns for minimum and maximum records.

```
weather <- make_all_day_table(weather)
```

All the steps described above resulted in two incomplete weather data frames. The first data frame (i.e. `qasha_raw`) consisted of *in-situ* temperature records from 30.10.2004 to 15.02.2019 with 62.3 % of data complete. The second data frame (i.e. `weather`) contains incomplete weather records from the nearby weather station at SAIQ from 1983 to 2019.

Patch extreme missing records with data from an auxiliary weather station

Based on the auxiliary weather data recorded at the SAIQ station and to generate a longer local weather record, we extended the *in-situ* data set back to 1983. We filled these cells with `NA` values for Tmin and Tmax variables via the `make_all_day_table()` function, producing a data frame as long as the one downloaded from the auxiliary source of weather data.

```
start_line <- qasha[1, ]

start_line[, c("Year", "Month", "Day", "Tmin", "Tmax")] <- c(1983, 1, 1, NA, NA)

qasha_complete <- rbind(start_line, qasha)

qasha_complete <- make_all_day_table(qasha_complete, add.DATE = F)
```

Table 1: Summary of the filling procedure performed on the in-situ data frame with data from the auxiliary weather station

	mean_bias	stdev_bias	filled	gaps_remain
Tmin	1.835	1.475	8516	1421
Tmax	4.743	2.045	8522	1415

To fill the gaps in the *in-situ* data frame, we used data from the auxiliary data set. We used the function `patch_daily_temperatures()` from `chillR`. This function computes the mean difference between main and auxiliary data for Tmin and Tmax variables and adjusts for it in filling the gaps. To make this function work, both data frames need to share a number of days with data to compute the correction factor.

```r
qasha_patched <- patch_daily_temperatures(qasha_complete, list(weather))
```

The function returns a list of 2 elements. The first element (weather), is a data frame containing the weather data for the *in-situ* data set with gaps filled using data from the auxiliary data set after correction for the mean difference for each variable. The second element (statistics) is a list of data frames containing information on the filling procedure (Table 1). It includes the mean bias for each variable, the standard deviation bias, the number of gaps filled and the number of gaps remaining.

The procedure above resulted in about 1,420 missing days out of 13,149 possible records. By using the function `fix_weather()`, which fills gaps by linear interpolation and produces some summary statistics, we determined that most of the missing days were observed before 1990.

```r
qasha_fixed <- fix_weather(qasha_patched)
```

`fix_weather()` returns a list of 2 elements. The first element is a data frame containing weather data with `NA` values filled by linear interpolation. The second element is a data frame containing information on the interpolation procedure. This data set showed that 1,421 and 1,415 missing values for minimum and maximum temperatures, respectively were filled through this procedure. A total of 1188 and 1186 days were missing before 1991 for Tmin and Tmax records, respectively. Based on this result, we removed from the analysis all records before this year.

```r
qasha_patched$weather <- subset(qasha_patched$weather, qasha_patched$weather$Year > 1990)

qasha_fixed <- fix_weather(qasha_patched)
```

After removing the rows before 1991, we applied the linear interpolation procedure again. We interpolated 233 and 229 out of 10,227 days for minimum and maximum records, respectively (about 4.52 %). This allowed us to obtain a complete data frame of weather records for 28 years.

Compute the observed chill from the weather data frame

We defined the chilling accumulation period as the time frame between November 1 and April 30 for each winter season. For further use, we need to define these dates as Julian dates, i.e. the day of the year.

```r
Start_JDay <- 305
End_JDay <- 120
```

We used the function `tempResponse_daily_list()` from `chillR` to compute climate-related metrics. This function allows the estimation, by default, of metrics such as chilling (in units of the Chilling Hours, Dynamic and Utah models) as well as heat (in units of the Growing Degree Hours model) using extreme daily records as main input. However, these models strictly require hourly temperature data as input. In past assessments, hourly temperatures have often been approximated based on an 'idealized' daily temperature curve, based on sunset and sunrise times derived from the location's latitude. For the mountain oases investigated in the present study, we deemed this procedure unreliable, because the rough topography of the area strongly influences daily temperature dynamics, with some locations being shaded by surrounding mountains for much of the day. Fortunately, the high-resolution data we recorded provided an opportunity for adjusting for this effect. We added to the `chillR` package the option to derive hourly temperature records from empirically determined daily temperature curves. These curves can be obtained with the `Empirical_daily_temperature_curve()` function from `chillR`. In brief, this function determines the `Prediction_coefficient`, which indicates by what fraction of the daily temperature range the temperature during the specified hour is above the daily minimum temperature. This is done separately for each month of the year. The output from `Empirical_daily_temperature_curve()` is a data frame that can be used as input to the `tempResponse_daily_list()` function.

```
empi_temps <- Empirical_daily_temperature_curve(qasha_all_data)
```

After determining the empirical temperature coefficients we can estimate the chilling and heat for the observed period.

```
chill_observed <- tempResponse_daily_list(qasha_fixed$weather, latitude = 23.07,
                                          Start_JDay = Start_JDay, End_JDay = End_JDay,
                                          misstolerance = 20, empirical = empi_temps)
```

`chill_observed` is a list containing a summary data frame for climate-related metrics computed with the observed weather data using an empirical temperature curve to derive hourly records from daily extreme observations. For comparison, we also computed the observed chill accumulation by deriving hourly records from daily extreme temperatures using the above-mentioned idealized daily temperature curve based on the latitude of the place. This option is included in `tempResponse_daily_list()` by setting the `empirical` parameter of the function to `NULL`.

```
idealized_chill_observed <- tempResponse_daily_list(qasha_fixed$weather, latitude = 23.07,
                                                    Start_JDay = Start_JDay,
                                                    End_JDay = End_JDay,
                                                    misstolerance = 20, empirical = NULL)
```

By comparing the results from both the empirical and idealized procedure to derive hourly temperatures from daily extremes, we determined that in low-elevation oases (i.e. Qasha' and Masayrat ar Ruwajah) the idealized temperature curve under-estimated chill accumulation compared to the empirical curve (Figure 1). We believe this is explained by the ability of the empirical temperature curve to capture specific micro-climatic conditions such as the shadow by surrounding mountains which may reduce the temperature and increase the frequency of chilling periods.

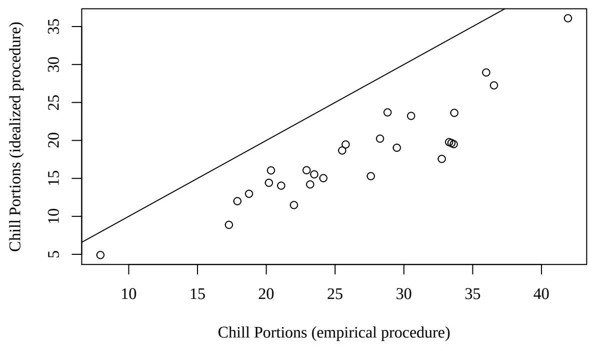

Figure 1: Idealized versus empirical computation of chill accumulation in Qasha using observed daily temperature extreme records. The diagonal black line represents the y = x equation

We decided to adopt the empirical approach in this and further analyses in this study, since it may better reflect the conditions experienced by the trees at each of the sites we evaluated.

Past weather scenarios

Although we analyzed the actual chill accumulation for 27 past seasons, analyzing these single-season estimates may obscure long-term trends due to year-to-year random variation. To better identify long-term trends in previous years, we generated past weather scenarios (i.e. simulated past temperatures) to characterize typical agroclimatic conditions for a number of points in time. We selected eight years spanning the period of interest and computed typical mean daily minimum and maximum temperatures for each month by applying a running mean function across all recorded monthly extreme temperatures for the respective month for all years on records (i.e. 28). This was achieved by using the function `temperature_scenario_from_records()` from the `chillR` library.

```r
past_temp_scenarios <- temperature_scenario_from_records(weather = qasha_fixed$weather,
                                                          year = c(1991, 1995, 1999,
                                                                   2003, 2007, 2011,
                                                                   2015, 2018))
```

The output of `temperature_scenario_from_records()` is a list containing 8 lists, each corresponding to one of the years used in the call of the function. These lists contain a data frame with the typical minimum and maximum temperature for each month of the year. The past scenarios generated with `temperature_scenario_from_records()` are used as input to fed a weather generator that produces 101 replicates of plausible weather data for each scenario year.

```r
past_weather_scenarios <- temperature_generation(weather = qasha_fixed$weather,
                                                 years = c(1991, 1995, 1999, 2003,
                                                           2007, 2011, 2015, 2018),
                                                   sim_years = c(2000, 2100),
                                             temperature_scenario = past_temp_scenarios)
```

`past_weather_scenarios` is a list of data frames containing the simulated weather, with columns DATE, Year, Month, Day, Tmin, and Tmax. We computed climate-related metrics for each synthetic season of the past simulated scenarios. As mentioned for observed records, we derived hourly temperature from simulated extreme daily temperatures using an empirical temperature curve.

```r
simu_past_response <- tempResponse_daily_list(past_weather_scenarios,
                                              latitude = 23.07,
                                              Start_JDay = Start_JDay,
                                              End_JDay = End_JDay,
                                              misstolerance = 20,
                                              empirical = empi_temps)
```

`simu_past_response` contains a number of data frames summarizing the chilling and heat responses computed for each season of the reference years of the past scenarios.

Future temperature scenarios

We downloaded future temperature projections from the Climate Wizard database which is maintained by the International Center of Tropical Agriculture (CIAT). We used an Application Programming Interface - API accessed by functions in the `chillR` package. The Climate Wizard database contains projections for two Representative Concentration Pathway scenarios (RCP4.5 and RCP8.5) produced by fifteen climate models. We downloaded temperature projections for the period 2035-2065 and 2070-2100 for both RCP scenarios using the `getClimateWizardData()` function from `chillR`. This function retrieves a list of variable length depending on the number of climate models used. Each list element is a climate scenario containing a data frame that shows the difference in mean minimum and maximum temperature expected for each month of the year relative to a baseline period defined by the `baseline` parameter.

In a simple call the function looks like this:

```r
getClimateWizardData(coordinates = c(longitude = 57.66, latitude = 23.07),
                     scenario = "rcp45",
                     start_year = 2035,
                     end_year = 2065,
                     temperature_generation_scenarios = TRUE,
                     baseline = c(1950, 2005))
```

These weather scenarios can be used to feed the weather generator and produce a number of synthetic seasons for the place of interest from past temperature records. This is done based on the expected change relative to the baseline of the Climate Wizard data. However, in most cases the baseline of the past weather data of the place of interest is not the same as the one used in the Climate Wizard function. To make both baselines comparable, we used the `temperature_scenario_baseline_adjustment()` function from `chillR`. Using the `temperature_scenario_from_records()` function we defined a scenario using the central year of the past weather record (for the weather baseline) and the central year of the Climate Wizard baseline. For the Climate Wizard baseline, we used the period between 1973 and 2017.

```r
weather_baseline_scen <- temperature_scenario_from_records(weather = qasha_fixed$weather,
                                                           year = c(2004.5))

clim_wiz_baseline_scen <- temperature_scenario_from_records(weather = qasha_fixed$weather,
                                                            year = c(1995))
```

After defining each baseline scenario, we adjusted the baseline using the respective function.

```r
baseline_adjustment <- temperature_scenario_baseline_adjustment(weather_baseline_scen,
                                                                clim_wiz_baseline_scen)
```

We implemented a `for` loop to automate the generation of future temperatures according to RCP scenarios and reference years and saved the outputs in the list `future_temps`. Note that running the next code block takes quite a while.

```r
RCPs <- c("rcp45", "rcp85")

Times <- c(2050, 2085)

future_temps <- list()

for(RCP in RCPs)
  for(Time in Times){

    start_year <- Time - 15
    end_year <- Time + 15

    clim_scen <- getClimateWizardData(c(longitude = 57.66,
                                        latitude = 23.07),
                                      RCP,
                                      start_year,
                                      end_year,
                                      temperature_generation_scenarios = TRUE,
                                      baseline = c(1973, 2017))

    clim_scen_adj <- temperature_scenario_baseline_adjustment(baseline_adjustment,
                                                              clim_scen)

    temps <- temperature_generation(weather = qasha_fixed$weather,
                                    years = c(1991, 2018),
                                    sim_years = c(2000, 2100),
                                    temperature_scenario = clim_scen_adj)

    future_temp <- list(temps)

    names(future_temp) <- paste0(RCP, "-", Time)

    future_temps <- c(future_temps, future_temp)
  }
```

`future_temps` contains 4 elements (i.e. of type list) for each of the combinations for RCP and reference year. Each element within these sub-lists is a data frame containing simulated weather data (minimum and maximum temperatures) according to the climate models for which we downloaded data from the Climate Wizard database. Using these projected temperature records, we computed chill and heat occurrence for each future scenario.

```
chills <- list()

chills[1] <- make_climate_scenario(simu_past_response,
                                   caption = "Historic",
                                   historic_data = chill_observed,
                                   time_series = TRUE)
```

We applied the function `make_climate_scenario()` from `chillR` to prepare the data in a useful format for plotting. This function organizes the data according to the different scenarios used in the study. The code chunk above takes the responses computed on the simulated past weather scenarios to generate a 'Historic' climate scenario. The `historic_data` parameter in this function helps to include the actual observations for chilling accumulation. The `for` loop below automates this process over the `future_temps` list adding each scenario to the `chills` object.

```
for (i in 1 : length(future_temps)){

  temps <- future_temps[[i]]

  chill <- tempResponse_daily_list(temps, latitude = 23.07,
                                   Start_JDay = Start_JDay,
                                   End_JDay = End_JDay,
                                   misstolerance = 20,
                                   empirical = empi_temps)

  if(substr(names(future_temps)[i], 1, 5) == "rcp45") RCPcaption <- "RCP4.5"
  if(substr(names(future_temps)[i], 1, 5) == "rcp85") RCPcaption <- "RCP8.5"

  if(substr(names(future_temps)[i], 7, 10) == "2050") Time_caption <- "2050"
  if(substr(names(future_temps)[i], 7, 10) == "2085") Time_caption <- "2085"

  chills <- make_climate_scenario(chill, caption = c(RCPcaption, Time_caption),
                                  add_to = chills)}
```

Finally, we generated the plots with an updated version of the `plot_climate_scenarios()` function of `chillR`. The updated version can be found in the `dormancyR`[1] library, available in a public repository.

[1] `dormancyR` is not a CRAN package. This library is constantly under development and it is possible that some functions will have been integrated into `chillR` or simply removed by the time you read this vignette. For compatibility of this document, we suggest downloading the `tar.gz` file in the releases section (link) of github and install the package from file (see example).

dormancyR can be directly installed from the github repository with the command install_github() from the devtools library.

```
devtools::install_github("https://github.com/EduardoFernandezC/dormancyR")
library(dormancyR)
```

Once installed, we can use plot_scenarios() from dormancyR to visualize the different scenarios for different metrics.

```
dormancyR::plot_scenarios(scenario_list = chills,
                          metric = "Chill_Portions",
                          add_historic = TRUE,
                          outlier_shape = NA,
                          color = "red",
                          size = 0.8,
                          historic_color = "skyblue")
```

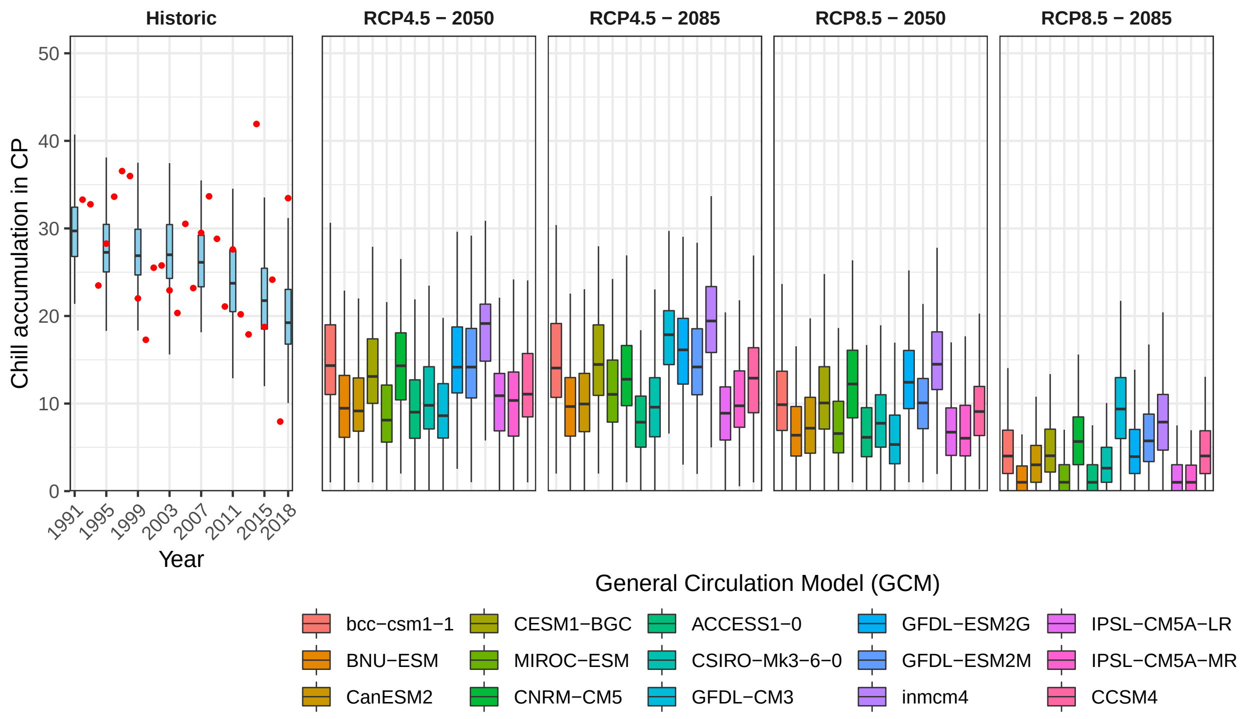

Figure 2: Chill distributions according to the Dynamic Model (expressed in Chill Portions) for different climate scenarios according to fifteen climate models. In 'Historic' panel, skyblue boxplots represent the chill distributions according to simulated past scenarios. Red dots represent the actual chill occurrence computed from the observed weather data. In the four rigth-hand panels, boxplots represent the chill distributions according to different climate models (colors).

Conclusions

In this reproducible example we have demonstrated the procedure we adopted in the main manuscript to generate future weather projections based on past observations. We recorded hourly records at each of three sites in Oman for a period of 14 years. To extend the period covered by these data, we retrieved weather information from a nearby weather station. We estimated mean differences for minimum and maximum records between the *in-situ* and auxiliary source of information and corrected the temperature for specific location bias. We used the resulted data frame to simulate past and future weather scenarios under two RCP scenarios by two time horizons. We computed chill-related metrics to estimate changes relative to the past baseline. We hope this vignette will be useful for the applying this procedure in different environments to assess the impacts of climate change on temperature-dependent systems. Our example allows the reader to reproduce the same figures we used in the main manuscript from the same raw data.