Untitled

Debora

2026-01-19

According to the recommendation of IPCC, we refer to local climatology and define heat wave as the period of at least five consecutive days with a daily maximum temperature over 5 °C higher than that of the local summertime climatology. The detailed calculation steps are as follows:

Step 1: Calculate the local temperature threshold: the mean daily T max of the summer months (i.e., June, July, and August) during the simulation period (i.e., 2071–2100) is recorded as the local temperature threshold.

Step 2: Determine the warm day: if the daily T max is 5 °C higher than the local temperature threshold, it is classified as a warm day.

Step 3: Calculate the frequency and duration of heat wave: a period of at least five consecutive warm days is called as a heat wave, and the total number of consecutive warm days is the duration of the heat wave. https://link.springer.com/article/10.1007/s00704-017-2123-8

Historical data to obtain average Tmax (climatology)

Define years and station

```
years <- 1961:1990
station <- "CYYZ"
```

Collect temperature data from all years (1961:1990)

```
##         date month_day temp_c  diel_sd
## 1 1961-06-01     06-01   21.3 2.661317
## 2 1961-06-02     06-02   24.1 3.392308
## 3 1961-06-03     06-03   23.0 4.263903
## 4 1961-06-04     06-04   26.3 5.235574
## 5 1961-06-05     06-05   28.5 5.103322
## 6 1961-06-06     06-06   26.3 3.876041
```

```
years <- 1961:1990
station <- "CYYZ"
```

Overall mean of historical daily maxima

```
summer_climatology_mean <- round(mean(all_data$temp_c, na.rm = TRUE), 2)
summer_climatology_mean
```

```
## [1] 25.38
```

Define station and date range

Download observations

```
obs <- riem_measures(
  station = station,
  date_start = start_date,
  date_end = end_date
)

obs$temp_c <- (obs$tmpf - 32) * 5 / 9
obs$date <- as.Date(obs$valid)
obs$month <- format(obs$date, "%m")
```

Keep only summer months: June, July, August

```
obs <- obs[obs$month %in% c("06", "07", "08"), ]
```

Daily max temperatures

```
daily_max <- aggregate(temp_c ~ date, data = obs, FUN = max)
```

Identify 5+ consecutive days 5°C above historical average tmax

Keep only summer months: June, July, August

```
over <- daily_max$temp_c > summer_climatology_mean + 5
```

Run-length encoding: groups consecutive TRUEs and FALSEs together

```
runs <- rle(over)
```

Ending positions of each run

```
ends <- cumsum(runs$lengths)
```

Calculate the starting positions of each run

```
starts <- ends - runs$lengths + 1
```

Find runs where condition was TRUE for 5 or more consecutive days

```
heatwave_indices <- which(runs$values & runs$lengths >= 5)
```

Get start and end dates of heatwaves

```
heatwave_periods <- data.frame(
  start_date = daily_max$date[starts[heatwave_indices]],
  end_date   = daily_max$date[ends[heatwave_indices]],
  duration   = runs$lengths[heatwave_indices]
)
```

View result

```
print(heatwave_periods)
```

```
##   start_date   end_date duration
## 1 2021-08-22 2021-08-26        5
## 2 2022-07-19 2022-07-24        6
```

Function to get mean, sd, and autocorrelation for 15-day window

```
get_stats_around_heatwave <- function(mid_date, daily_max) {
  start_window <- mid_date - 7
  end_window <- mid_date + 7

  # Subset data
  window_data <- daily_max[daily_max$date >= start_window & daily_max$date <= end_window, ]

  # Calculate stats
  mean_temp <- mean(window_data$temp_c, na.rm = TRUE)
  sd_temp <- sd(window_data$temp_c, na.rm = TRUE)
  acf_temp <- acf(window_data$temp_c, lag.max = 1, plot = FALSE)$acf[2]  # lag-1 autocorrelation

  # Return with start and end date
  return(data.frame(
    start_date = start_window,
    end_date = end_window,
    mean = round(mean_temp, 2),
    sd = round(sd_temp, 2),
    autocorrelation = round(acf_temp, 2)
  ))
}
```

Calculate midpoint of each heatwave

```
mid_dates <- as.Date(heatwave_periods$start_date + floor((heatwave_periods$end_date - heatwave_periods$s
```

Apply to each midpoint

```
results <- do.call(rbind, lapply(mid_dates, get_stats_around_heatwave, daily_max = daily_max))
```

View result

```
print(results)
```

```
##   start_date   end_date  mean   sd autocorrelation
## 1 2021-08-17 2021-08-31 29.47 2.29            0.28
## 2 2022-07-14 2022-07-28 29.33 3.06            0.53
```

Get stats around historical data with the same timeframe of heatwaves

Reference heatwave midpoints

```
mid_dates_reference <- as.Date(c("2021-08-24", "2022-07-21"))
```

Generate 15-day windows for each midpoint across all years

```
get_windows <- function(mid_date, years) {
  md <- format(mid_date, "%m-%d")
  lapply(years, function(yr) {
    ref <- as.Date(paste0(yr, "-", md))
    seq(ref - 7, ref + 7, by = "1 day")
  })
}
```

Apply for both midpoints and combine

```
windows1 <- get_windows(mid_dates_reference[1], years)
windows2 <- get_windows(mid_dates_reference[2], years)
windows_all <- unique(do.call(c, c(windows1, windows2)))
```

Filter for those dates

```
selected_data <- all_data[all_data$date %in% windows_all, ]
```

Ensure date column is in Date format

```
selected_data$date <- as.Date(selected_data$date)
```

Extract year, month, and day

```
selected_data$year <- format(selected_data$date, "%Y")
selected_data$month_day <- format(selected_data$date, "%m-%d")
```

Define periods

```
periods <- list(
  c("07-14", "07-28"),
  c("08-17", "08-30")
)
```

Get all unique years

```
years <- unique(format(selected_data$date, "%Y"))
```

Loop over each year and period

```
for (yr in years) {
  for (p in periods) {
    start_date <- as.Date(paste0(yr, "-", p[1]))
    end_date <- as.Date(paste0(yr, "-", p[2]))

    # Subset the data for this year and period
    subset_data <- selected_data[selected_data$date >= start_date &
                                 selected_data$date <= end_date, ]

    temps <- subset_data$temp_c
    if (length(temps) >= 2) {
      all_means <- c(all_means, mean(temps, na.rm = TRUE))
      all_sds <- c(all_sds, sd(temps, na.rm = TRUE))
      all_acfs <- c(all_acfs, acf(temps, lag.max = 1, plot = FALSE)$acf[2])
    }
  }
}
```

Final summary

```
final_mean <- round(mean(all_means, na.rm = TRUE), 2)
final_sd <- round(mean(all_sds, na.rm = TRUE), 2)
final_acf <- round(mean(all_acfs, na.rm = TRUE), 2)
```

Print

```
cat("Mean of means:", final_mean, "\n")
```

```
## Mean of means: 25.82
```

```
cat("Mean of SDs:", final_sd, "\n")
```

```
## Mean of SDs: 2.96
```

```
cat("Mean of autocorrelations:", final_acf, "\n")
```

```
## Mean of autocorrelations: 0.41
```

MIMICRY FUNCTION

```
mimicry <- function(x, y) {
  xs <- sort(x, index.return = TRUE)
  ys <- sort(y, index.return = TRUE)
  zs <- sort(ys$ix, index.return = TRUE)
  z <- xs$x[zs$ix]
  return(z)
}
```

MIMICRY SIMULATION FUNCTION

```
simulate_temp_mimicry <- function(n_days, mean_temp, target_autocorr, target_sd,
                                  lower_thresh = min(selected_data$temp_c),
                                  upper_thresh = max(selected_data$temp_c), tol = 100) {
  # Step 1: White noise
  x <- rnorm(n_days, mean = mean_temp, sd = target_sd)

  # Step 2: Truncate out-of-bounds values
  icount <- 0
  while ((any(x > upper_thresh) || any(x < lower_thresh)) && icount < tol) {
    icount <- icount + 1
    for (i in 1:length(x)) {
      if (x[i] > upper_thresh || x[i] < lower_thresh) {
        x[i] <- rnorm(1, mean = mean_temp, sd = target_sd)
      }
    }
  }
  if (icount >= tol) {
    x[x > upper_thresh] <- upper_thresh
    x[x < lower_thresh] <- lower_thresh
  }

  # Step 3: Standardize and rescale
  x <- scale(x)[, 1]
  noise <- x * target_sd + mean_temp

  # Step 4: Generate AR(1) process
  max_attempts <- 1000
  attempt <- 0
  best_diff <- Inf
  best_gs <- NULL

  while (attempt < max_attempts) {
    attempt <- attempt + 1
    gs <- numeric(n_days)
    gs[1] <- rnorm(1)
    crand <- rnorm(n_days)
    for (i in 2:n_days) {
      gs[i] <- target_autocorr * gs[i - 1] + crand[i] * sqrt(target_sd) * sqrt(1 - target_autocorr^2)
    }
    ac_try <- acf(gs, lag.max = 1, plot = FALSE)$acf[2]
```

```r
    diff <- abs(ac_try - target_autocorr)
    if (diff < 0.01) break
    if (diff < best_diff) {
      best_diff <- diff
      best_gs <- gs
    }
  }

  if (attempt >= max_attempts) {
    warning("Max attempts reached. Using best approximation.")
    gs <- best_gs
  }

  # Step 5: Apply mimicry
  z <- mimicry(noise, gs)
  return(round(z, 1))
}
```

RUN SIMULATIONS

```r
n_days <- 15
set.seed(123)
```

Function to count clusters of 5+ consecutive days > threshold

```r
count_clusters <- function(temp, threshold) {
  exceed <- temp > threshold
  rle_res <- rle(exceed)
  sum(rle_res$values & rle_res$lengths >= 5)
}
```

Number of simulations and days

```r
n_sim <- 1000
```

Threshold = Ambient mean of max + 5BC

```r
temp_threshold <- summer_climatology_mean + 5
```

Prepare list to store results

```r
cluster_counts <- list()
simulation_data <- list()
```

New values after 40% and 60% increase in autocorrelation

```r
auto_40 <- sqrt(0.6^2 * 1.40)
auto_60 <- sqrt(0.6^2 * 1.60)
```

Create the simulation_scenarios data frame

```r
simulation_scenarios <- data.frame(
  Scenario = c("Ambient", "Ambient+", "Ambient+Autocorrel+40", "Ambient+Autocorrel+60"),
  Mean = c(summer_climatology_mean, summer_climatology_mean+3.5, summer_climatology_mean+3.5, summer_cli
  #historical mean of max daily temps or mean+climate change warming
  SD = c(final_sd), #historical standard deviation
  Autocorrelation = c(0.6, 0.6, auto_40, auto_60),
  #historical autocorrelation of max daily temps + projected increase in autocorrelation for eastern Ca
  #fig. 3 https://doi.org/10.1038/s41586-021-03943-z
  stringsAsFactors = FALSE
)
```

Loop through each scenario

```r
for (i in 1:nrow(simulation_scenarios)) {
  scenario <- simulation_scenarios$Scenario[i]
  mean_temp <- simulation_scenarios$Mean[i]
  sd_temp <- simulation_scenarios$SD[i]
  autocorr <- simulation_scenarios$Autocorrelation[i]

  # Run simulations
  sims <- replicate(n_sim, simulate_temp_mimicry(n_days, mean_temp, autocorr, sd_temp), simplify = FALSE

  # Store first 6 simulations for plotting
  for (j in 1:6) {
    sim_df <- data.frame(
      Day = 1:n_days,
      Temp = sims[[j]],
      Scenario = scenario,
      Simulation = paste("Sim", j)
    )
    simulation_data[[length(simulation_data) + 1]] <- sim_df
  }

  # Count clusters of 5+ days > threshold
  cluster_counts[[scenario]] <- sapply(sims, count_clusters, threshold = temp_threshold)
}
```

Summary table

```r
cluster_summary <- data.frame(
  Scenario = names(cluster_counts),
  Simulations_with_clusters = sapply(cluster_counts, function(x) sum(x > 0))
)
```

Print result

```
##                                          Scenario Simulations_with_clusters
## Ambient                                   Ambient                         0
## Ambient+                                 Ambient+                       169
## Ambient+Autocorrel+40 Ambient+Autocorrel+40                             310
## Ambient+Autocorrel+60 Ambient+Autocorrel+60                             358
```

Calculate probability

```
probability <- cluster_summary$Simulations_with_clusters / 1000
```

Plot