Using ThermStats to calculate statistics for thermal images

Rebecca A. Senior

26 July, 2018

Contents

1 Summary

`ThermStats` is designed for biologists using thermography to quantify thermal heteorgeneity. It uses the `Thermimage` package (Tattersall, 2017) to batch process data from FLIR thermal cameras, and takes inspiration from FRAGSTATS (McGarigal and Ene, 2012), `SDMTools` (VanDerWal et al., 2014), Faye et al. (2016) and Shi et al. (2016) to facilitate the calculation of various metrics of thermal heterogeneity for any gridded temperature data.

The package is available to download from GitHub using `devtools`:

```
## devtools::install_github("rasenior/ThermStats")
library(ThermStats)
```

Once loaded, the code below can be followed step-by-step. Please note that some steps may not work if there are spaces in the path to where the package is installed.

2 Extracting raw data

Data are extracted from FLIR images using `batch_extract`. This is a batch implementation of the `readflirJPG` function from `Thermimage`. It requires only the path to the directory of FLIR thermal images, and the freely available external software 'Exiftool'. Besides raw data, this step also retrieves camera-specific calibration parameters which are required later to convert raw data to temperature values.

```
# Batch extract four FLIR thermal images included
# in the ThermStats package installation
flir_raw <-
   batch_extract(in_dir =
                     system.file("extdata",
                                 package =
                                     "ThermStats"),
                 write_results = FALSE)
```

3 Converting raw data to temperature

Raw data are encoded in each thermal image as a 16 bit analog-to-digital signal, which represents the radiance received by the infrared sensor. The function `batch_convert` converts these raw data to temperature values using equations from infrared thermography, via a batch implementation of the function `raw2temp` in `Thermimage`. It uses environmental parameters defined by the user, and the

calibration constants extracted in `batch_extract`. See Chapter 3: Methods for a full discussion of the different environmental parameters. In brief:

- Emissivity = the amount of radiation emmitted by a particular object, for a given temperature.
- Object distance = the distance between the camera and the object of interest.
- Reflected apparent temperature = the temperature resulting from radiation that originates from the atmosphere and is reflected by the object.
- Atmospheric temperature = the temperature of the atmosphere.
- Relative humidity = the relative humidity of the atmosphere.

```
# Define raw data
raw_dat <- flir_raw$raw_dat
# Define camera calibration constants dataframe
camera_params <- flir_raw$camera_params
# Define metadata
metadata <- flir_metadata

# Create vector denoting the position of each
# photo within the metadata dataframe
photo_index <- match(names(raw_dat),
                     metadata$photo_no)

# Batch convert --------------------------------
flir_converted <-
    batch_convert(
        raw_dat = raw_dat,
        # Emissivity = mean of the range in
        # Scheffers et al. 2017
        E = mean(c(0.982,0.99)),
        # Object distance = hypotenuse of a right
        # triangle where the vertical side is
```

```
        # 1.3 m (breast height) & the angle down
        # is 45 degrees
        OD = (sqrt(2))*1.3,
        # Apparent reflected temperature,
        # atmospheric temperature and infrared
        # window temperature set as the
        # atmospheric temperature measured in
        # the field
        RTemp = metadata$atm_temp[photo_index],
        ATemp = metadata$atm_temp[photo_index],
        IRWTemp = metadata$atm_temp[photo_index],
        # Infrared Window transmission kept at
        # default value of 1
        IRT = 1,
        # Relative humidity is set as the
        # relative humidity measured in the field
        RH = metadata$rel_humidity[photo_index],
        # Calibration constants from
        # batch_extract, constant for each camera
        PR1 = camera_params[,"PlanckR1"],
        PB = camera_params[,"PlanckB"],
        PF = camera_params[,"PlanckF"],
        PO = camera_params[,"PlanckO"],
        PR2 = camera_params[,"PlanckR2"],
        # Whether to write results or just return
        write_results = FALSE)
```

4 Calculating thermal statistics

Statistics can be calculated for individual temperature matrices, or across multiple matrices within a specified grouping. The latter is useful for sampling designs where multiple images are collected at each sampling event to capture temperature across a wider sampling unit, such as a plot. In either case, statistics include summary statistics specified by the user – for example, mean, minimum and maximum – as well as spatial statistics for hot and cold spots, identified using the Getis-Ord local statistic (Getis and Ord, 1996).

For an individual matrix, `get_stats` requires the user to specify the matrix and the desired statistics. Statistics can be calculated for geographic temperature data (in a matrix or raster format), in which case the user should also define the extent and projection of the data.

```
flir_stats <-
    get_stats(
        # The temperature matrix
        val_mat = flir_converted$`8565`,
        # The ID of the matrix
        matrix_id = "8565",
        # Whether or not to identify hot and
        # cold spots
        get_patches = TRUE,
        # The size of the neighourhood
        # (for calculating local G statistic)
        k = 8,
        # The neghbour weighting style
        # (for calculating local G statistic)
        style = "W",
        # The matrix projection
        # (only relevant for geographic data)
        mat_proj = NULL,
```

```
        # The matrix extent
        # (only relevant for geographic data)
        mat_extent = NULL,
        # The data to return
        return_vals = c("df",
                        "patches",
                        "pstats"),
        # The names of the statistics functions
        pixel_fns = NULL,
        # The summary statistics
        median, perc_5, perc_95, SHDI
    )
```

For grouped matrices, `stats_by_group` requires the user to supply a list of matrices along with metadata and the name of the variable in the metadata that defines the matrix grouping. Table 1 shows the metadata used in the code snippet, where photo number ('photo_no') defines individual temperature matrices, and the replicate identity ('rep_id') defines the grouping of photos. There are two replicates, 'T7P1' and 'T7P2', and each has two associated photos.

Table 1: Example metadata denoting the grouping ('rep_id') of different temperature matrices. Statistics can be calculated over multiple matrices within a group, using the function `stats_by_group`.

photo_no	rep_id	atm_temp	rel_humidity
8565	T7P1	24.00	96
8583	T7P1	24.00	96
8589	T7P2	23.25	98
8613	T7P2	23.50	96

By default, both `get_stats` and `stats_by_group` return a dataframe with patch statistics (Table 2) for each matrix or matrix group, respectively.

Table 2: A snippet of hot spot patch statistics returned by `stats_by_group`, which implements `get_stats` within groups.

median	perc_5	perc_95	SHDI	hot_shape_index	hot_aggregation
23.5	23	24.5	1.16	7.54	0.895
24.0	23	25.0	1.68	7.80	0.855

5 Plotting

In addition to patch statistics, `get_stats` can return (1) the temperature matrix in a dataframe format, and (2) a `SpatialPolygonsDataFrame` of its hot and cold spots. The function `plot_patches` can then plot the temperature distribution (if `plot_distribution = TRUE`), and recreates the original thermal image overlaid with outlines of hot and cold spots.

```
plot_patches(
    # The raw temperature data
    df = flir_stats$df,
    # The patch outlines
    patches = flir_stats$patches
        )
```

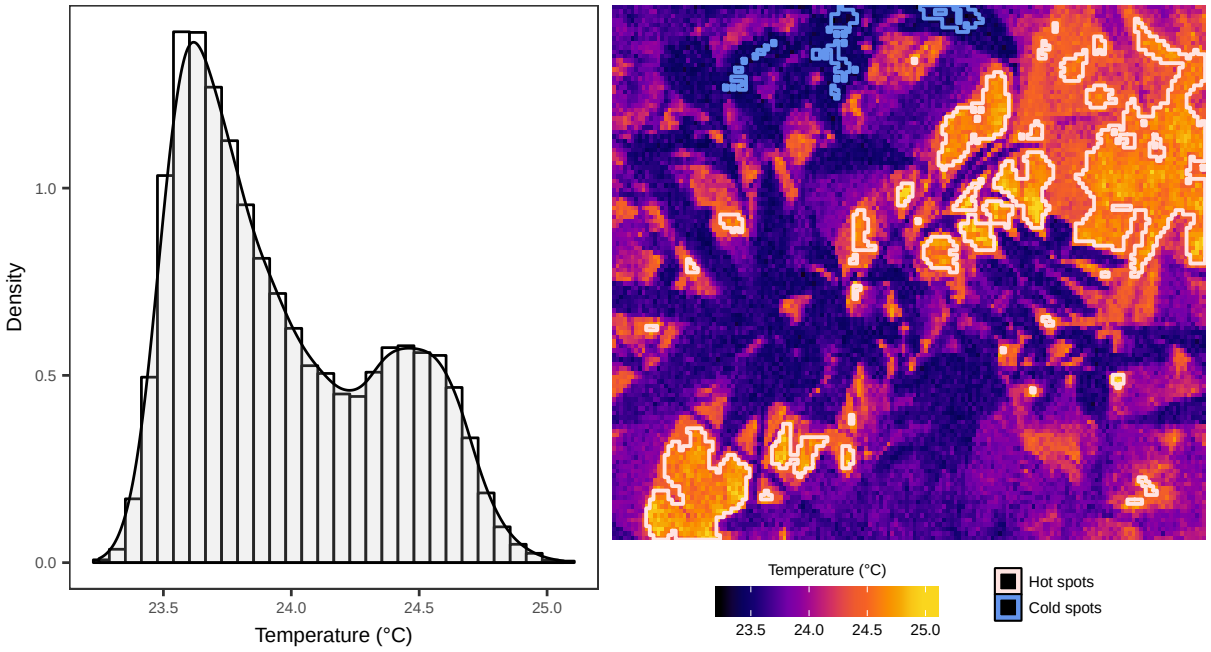

Figure 1: The output of `plot_patches` includes a histogram and the original temperature data overlaid with outlines of hot and cold spots, identified using the Getis-Ord local statistic.

References

Faye, E., Rebaudo, F., Yánez-Cajo, D., Cauvy-Fraunié, S., Dangles, O., 2016. A toolbox for studying thermal heterogeneity across spatial scales: From unmanned aerial vehicle imagery to landscape metrics. Methods in Ecology and Evolution 7, 437–446. doi:10.1111/2041-210X.12488

Getis, A., Ord, J.K., 1996. Local spatial statistics: An overview. Spatial analysis: modelling in a GIS environment 374, 261–277.

McGarigal, S.C., K., Ene, E., 2012. FRAGSTATS v4: Spatial pattern analysis program for categorical and continuous maps.

Shi, H., Wen, Z., Paull, D., Guo, M., 2016. A framework for quantifying the thermal buffering effect of microhabitats. Biological Conservation 204, 175–180. doi:10.1016/j.biocon.2016.11.006

Tattersall, G.J., 2017. Thermimage: Thermal Image Analysis.

VanDerWal, J., Falconi, L., Januchowski, S., Shoo, L., Storlie, C., 2014. SDMTools: Species distribution modelling tools: Tools for processing data associated with species distribution modelling exercises.