ThermStats : an R package for quantifying surface thermal heterogeneity in assessments of microclimates

Supplementary material

Rebecca A. Senior[1]*, Jane K. Hill[2] and David P. Edwards[1]

[1]Department of Animal and Plant Sciences, Alfred Denny Building, University of Sheffield, Western Bank, Sheffield, S10 2TN, UK

[2]Department of Biology, University of York, Wentworth Way, York, YO10 5DD, UK

***Corresponding author:** rebecca.a.senior@gmail.com, +44(0)114 2220123 (R.A. Senior)

ORCID iDs: orcid.org/0000-0002-8208-736X (R.A. Senior); orcid.org/0000-0003-1871-7715 (J.K. Hill)

Contents

1 Text S1: Package vignette

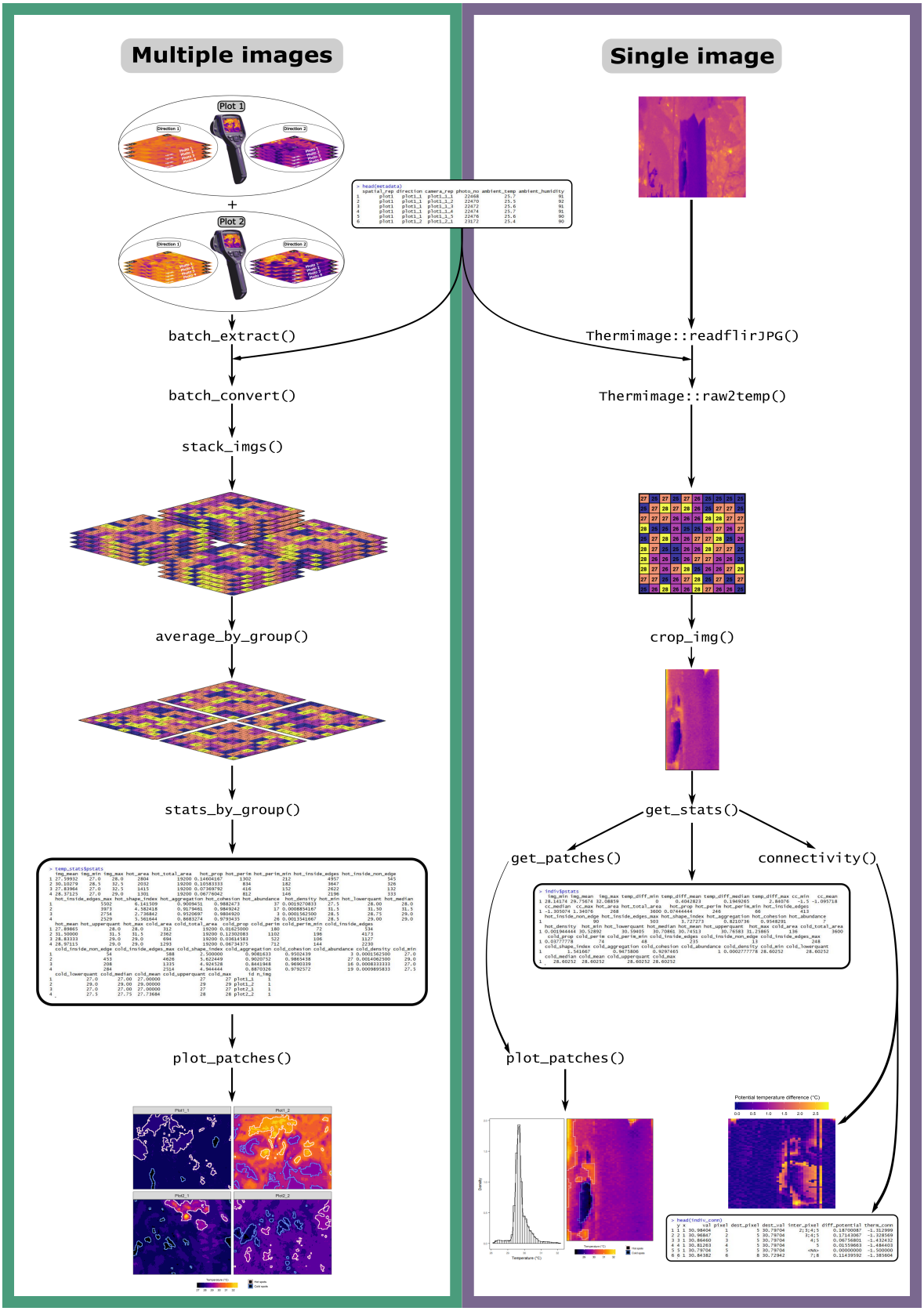

Figure 1: Schematic summarising the key functions for processing groups of images (left) or a single image (right).

1.1 Summary

ThermStats is designed for biologists using thermography to quantify thermal heterogeneity. It uses the Thermimage package (Tattersall, 2017) to batch process data from FLIR thermal cameras, and takes inspiration from FRAGSTATS (McGarigal, Cushman, & Ene, 2012), SDMTools (VanDerWal, Falconi, Januchowski, Shoo, & Storlie, 2014), Faye, Rebaudo, Yánez-Cajo, Cauvy-Fraunié, & Dangles (2016) and Shi, Wen, Paull, & Guo (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.

1.2 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 thermal images included in ThermStats installation
flir_raw <-
    batch_extract(in_dir = system.file("extdata",
                                       package = "ThermStats"),
                  write_results = FALSE)
```

1.3 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 the calibration constants extracted in `batch_extract` and environmental parameters defined by the user:

- Emissivity = the amount of radiation emitted by a particular object, for a given temperature.
- Object distance = the distance between the camera and the object of interest.
- Reflected apparent temperature = thermal radiation that originates from other objects and is reflected by the object of interest.
- 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 metadata
photo_index <- match(names(raw_dat),
                     metadata$photo_no)
# Batch convert
flir_converted <-
    batch_convert(
        raw_dat = raw_dat,
        # Emissivity = mean of range in Scheffers et al. 2017
        E = mean(c(0.982,0.99)),
        # Object distance = hypotenuse of right triangle where
        # vertical side is 1.3 m (breast height) & angle down is 45°
        OD = (sqrt(2))*1.3,
        # Apparent reflected temperature & atmospheric temperature =
        # atmospheric temperature measured in the field
```

```
        RTemp = metadata$atm_temp[photo_index],
        ATemp = metadata$atm_temp[photo_index],
        # Relative humidity = relative humidity measured in the field
        RH = metadata$rel_humidity[photo_index],
        # Calibration constants from 'batch_extract'
        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)
```

1.4 Calculating thermal statistics

Statistics can be calculated for individual thermal images (in a matrix or raster format), or across multiple images 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 can include summary statistics specified by the user – for example, mean, minimum and maximum – as well as thermal connectivity (based on the climate connectivity measure of McGuire, Lawler, McRae, & Theobald, 2016) and spatial statistics for hot and cold spots, identified using the G* variant of the Getis-Ord local statistic (Getis & Ord, 1996).

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

```
flir_stats <-
    get_stats(
        # The temperature dataset
        img = flir_converted$`8565`,
```

```
        # The ID of the dataset
        id = "8565",
        # Whether or not to calculate thermal connectivity
        calc_connectivity = FALSE,
        # Whether or not to identify hot and cold spots
        patches = TRUE,
        # The image projection (only relevant for geographic data)
        img_proj = NULL,
        # The image extent (only relevant for geographic data)
        img_extent = NULL,
        # The data to return
        return_vals = c("df", # Temperature data as dataframe
                        "patches", # Patch outlines
                        "pstats"), # Patch statistics dataframe
        # The summary statistics of interest
        sum_stats = c("median", "SHDI",
                      "perc_5", "perc_95"))
```

For grouped images, `stats_by_group` requires the user to supply a list of matrices or a raster stack, and (optionally) the metadata and the name of the variable in the metadata that defines the 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.

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

Table 1: Example metadata denoting the grouping ('rep_id') of different thermal images. Statistics can be calculated over multiple images 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

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

img_median	img_perc_5	img_perc_95	img_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

1.5 Plotting

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

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

```
    patches = flir_stats$patches,
    # Add hatching
    hatching = TRUE)
```

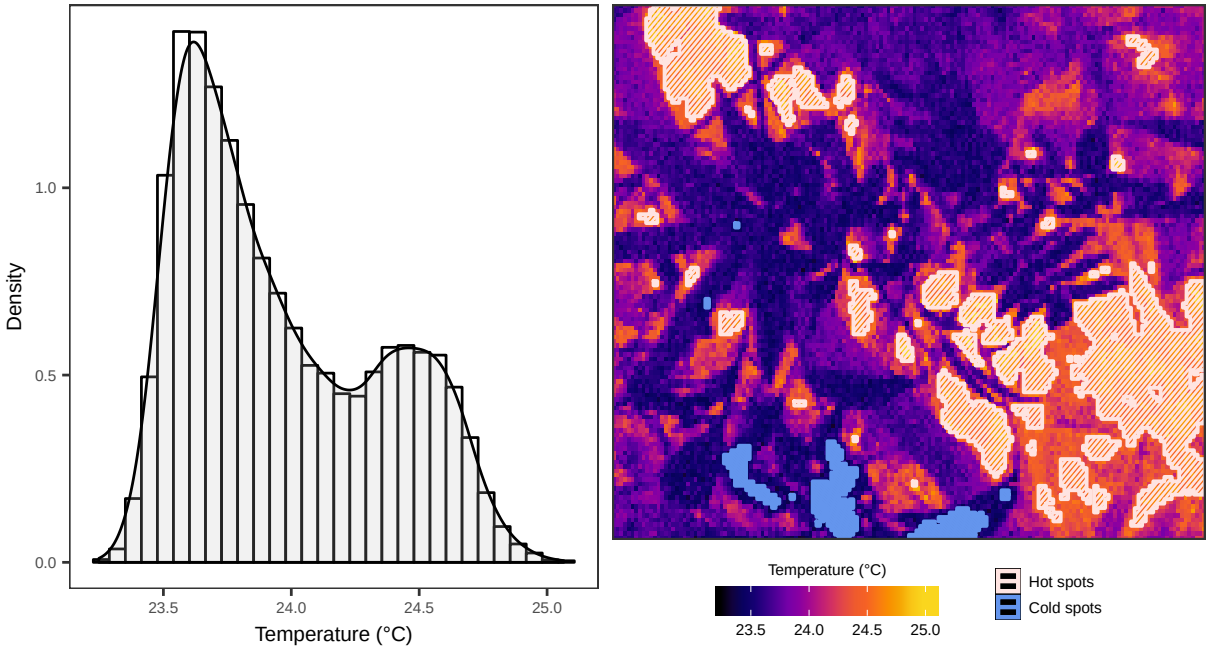

Figure 2: The output of `plot_patches` includes a histogram and the original temperature data overlaid with outlines of hot and cold spots, identified using the G* variant of the Getis-Ord local statistic.

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*(4), 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, K., Cushman, S. A., & Ene, E. (2012). FRAGSTATS v4: Spatial pattern analysis program for categorical and continuous maps. Retrieved from http://www.umass.edu/landeco/research/fragstats/fragstats.html

McGuire, J. L., Lawler, J. J., McRae, B. H., & Theobald, D. M. (2016). Achieving climate connectivity in a fragmented landscape. *Proceedings of the National Academy of Sciences*, *113*(26), 7195–7200. doi:10.1073/pnas.1602817113

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. Retrieved from https://CRAN.R-project.org/package=Thermimage

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. Available at: Https://cran.r-project.org/package=SDMTools. Retrieved from https://CRAN.R-project.org/package=SDMTools