Shade Treatment Effects on Cattle Heat Stress

Rajan Shankar[a]

[a]School of Mathematics and Statistics, University of Sydney, NSW, 2006

This version was compiled on June 7, 2021

Minimising cattle heat stress is of importance to livestock farmers, both ethically and economically. In this project, we analysed the effects of different shade structures on the rumen temperature and drinking patterns of cattle.
We first developed a method to detect drinking events based on drops in cattle rumen temperature. We then used this method to determine that there was a significant difference in the number of drinks between unshaded cattle and shaded/sheltered cattle.
Next, we saw that there was a significant difference in the night-average rumen temperature between unshaded cattle and sheltered cattle. The differences in day-average rumen temperature between the shade treatments were only significant after incorporating an interaction term between shade and the accumulated heat load (AHL). Furthermore, we found that separating days into cloudy days and clear days based on the vapour-pressure deficit (VPD) allowed us to better identify heat stress in cattle.
Finally, we concluded that decreasing the heat load index (HLI) upper threshold in the calculation of the AHL does not increase the accuracy of predicting day-average rumen temperature.

linear mixed models | time series | outlier detection | drinking patterns | rumen temperature | interactive viz | data processing

1. Introduction

Environmental conditions are known to influence the health, welfare and productivity of feedlot cattle. It has been established that shade structures are beneficial for the health of these cattle. However, such studies have predominantly been conducted in northern, sub-tropical regions. This project considers cattle in unshaded, shaded (shade cloth) and sheltered (waterproof) pens that are located in the University of New England's research feedlot Tullimba, a southern Australian environment.

Experimental design. At Tullimba, there were 90 feedlot cattle divided equally across 9 pens. Each group of 3 pens consisted of one of 3 shade treatments—unshaded, shade cloth or waterproof. Hence, there were 30 animals under each of the shade treatments. The experiment ran for 105 days from the 23 December 2020 to the 7 April 2021.

Each animal was equipped with a bolus that measured its internal rumen temperature in 10-minute intervals. An instrument that collected weather data and an instrument that collected black globe temperature data at 10-minute intervals were also set up at Tullimba.

The University of New England's animal ethics committee approved the experimental procedures in November 2020 (authority number AEC20-091).

Device details. The instrument that collected weather data was a MCC Hub weather station, supplied by ICT International. It measured variables such as wind speed, solar radiation, air temperature, relative humidity, precipitation, etc. The rumen temperature boluses as well as the custom web interface for visualising the rumen temperature data were developed and supplied by smaXtec.

2. Data Processing and Cleaning

A major component of this project involved the rigorous processing and cleaning of the collected data. Due to the many sources of data—temperature boluses and weather instruments—and the imperfectness of these collection devices, it was a very difficult task to curate a single, tidy data set for analysis.

All data cleaning and analysis was performed in the R language (R Core Team, 2020). A multitude of packages from the tidyverse suite (Wickham *et al.*, 2019) were used to read, transform, merge and plot our data. We used janitor (Firke, 2020) for miscellaneous cleaning and tabling, visdat (Tierney, 2017) for visualising missing values, and tsibble (Wang *et al.*, 2020) for working with time series data and record keeping. We also used Python (Python Core Team, 2020) along with the package PyAutoGUI (Sweigart, 2020) to scrape the rumen temperature data from a web interface.

Minimal manual editing was done before processing the weather data in R; we briefly explain these edits in Appendix A.1. The code pipelines for all of the processing done in R is available in Appendix A.2.

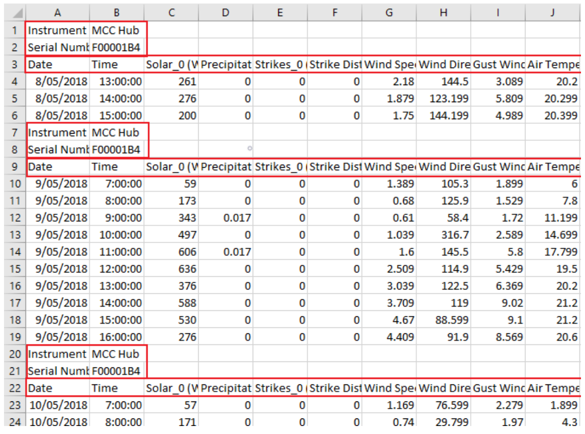
Weather data. We first discuss some key steps of the cleaning process for the weather data.

Initial read. A major problem with the weather data was that the variable names and the details of the data collection instrument were dispersed throughout the .csv file in arbitrary rows. To illustrate this, the first few rows of the file are shown in Figure 1.

Fortunately, this problem was dealt with fairly easily by pre-specifying variable types when the data file was read in.

Filling gaps. There was a bug in the weather data collection instrument in the sense that it would not record any observation at midnight, i.e. while most of the time observations were recorded in 10-minute intervals, there was a 20-minute gap between an observation at 23:50:00 and at 00:10:00 the next day. To make sure that such records existed in the data, we used the tsibble package (Wang *et al.*, 2020) to add in blank rows at these missing timestamps. Figure 2 illustrates this procedure.

Missing values. Figure 3 shows that the missingness was scattered throughout the experiment time series and that there



	A	B	C	D	E	F	G	H	I	J
1	Instrument	MCC Hub								
2	Serial Numb	F00001B4								
3	Date	Time	Solar_0 (W	Precipitat	Strikes_0 (	Strike Dist	Wind Spe	Wind Dire	Gust Winc	Air Tempe
4	8/05/2018	13:00:00	261	0	0	0	2.18	144.5	3.089	20.2
5	8/05/2018	14:00:00	276	0	0	0	1.879	123.199	5.809	20.299
6	8/05/2018	15:00:00	200	0	0	0	1.75	144.199	4.989	20.399
7	Instrument	MCC Hub								
8	Serial Numb	F00001B4								
9	Date	Time	Solar_0 (W	Precipitat	Strikes_0 (	Strike Dist	Wind Spe	Wind Dire	Gust Winc	Air Tempe
10	9/05/2018	7:00:00	59	0	0	0	1.389	105.3	1.899	6
11	9/05/2018	8:00:00	173	0	0	0	0.68	125.9	1.529	7.8
12	9/05/2018	9:00:00	343	0.017	0	0	0.61	58.4	1.72	11.199
13	9/05/2018	10:00:00	497	0	0	0	1.039	316.7	2.589	14.699
14	9/05/2018	11:00:00	606	0.017	0	0	1.6	145.5	5.8	17.799
15	9/05/2018	12:00:00	636	0	0	0	2.509	114.9	5.429	19.5
16	9/05/2018	13:00:00	376	0	0	0	3.039	122.5	6.369	20.2
17	9/05/2018	14:00:00	588	0	0	0	3.709	119	9.02	21.2
18	9/05/2018	15:00:00	530	0	0	0	4.67	88.599	9.1	21.2
19	9/05/2018	16:00:00	276	0	0	0	4.409	91.9	8.569	20.6
20	Instrument	MCC Hub								
21	Serial Numb	F00001B4								
22	Date	Time	Solar_0 (W	Precipitat	Strikes_0 (	Strike Dist	Wind Spe	Wind Dire	Gust Winc	Air Tempe
23	10/05/2018	7:00:00	57	0	0	0	1.169	76.599	2.279	1.899
24	10/05/2018	8:00:00	171	0	0	0	0.74	29.799	1.97	4.3

Fig. 1. First 24 rows of the weather data file with troublesome rows outlined in red.

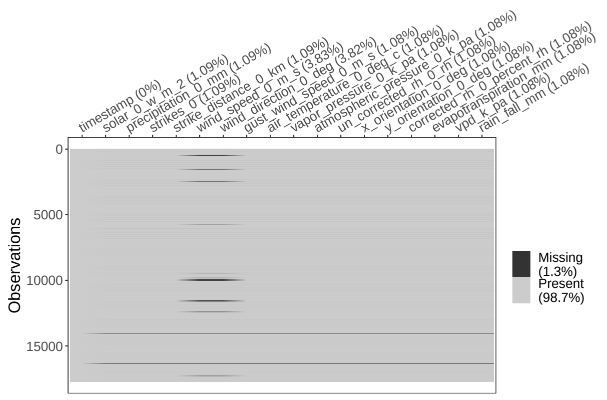
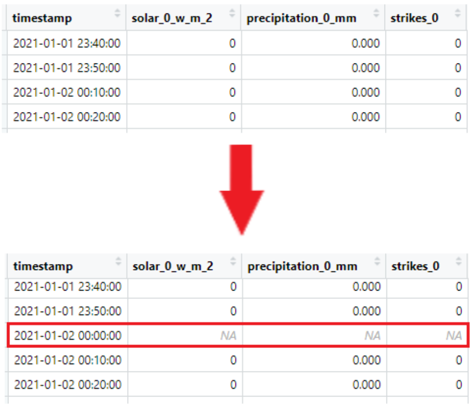


Fig. 3. Locations of missing values in the tidy weather data.



timestamp	solar_0_w_m_2	precipitation_0_mm	strikes_0
2021-01-01 23:40:00	0	0.000	0
2021-01-01 23:50:00	0	0.000	0
2021-01-02 00:10:00	0	0.000	0
2021-01-02 00:20:00	0	0.000	0

timestamp	solar_0_w_m_2	precipitation_0_mm	strikes_0
2021-01-01 23:40:00	0	0.000	0
2021-01-01 23:50:00	0	0.000	0
2021-01-02 00:00:00	NA	NA	NA
2021-01-02 00:10:00	0	0.000	0
2021-01-02 00:20:00	0	0.000	0

Fig. 2. Demonstration of the tsibble's fill_gaps function adding in blank rows at missing timestamps.

were no long, continuous chunks of missing values in any variable. We decided to use a linear interpolation strategy to impute missing values for each variable. This was implemented in R as a custom-written function (Appendix A.3).

Linear interpolation works by considering the last non-missing value x_n before a chunk of $k-1$ missing values and the next non-missing value x_{n+k}, and imputing $x_{n+i} = x_n + \frac{i}{k}(x_{n+k} - x_n)$ for all $1 \leq i < k$.

Black globe temperature data. The instrument collecting black globe temperature data was different to the one that collected the weather data, so we had a separate file that recorded the black globe temperature data. However, we discovered that these recordings were faulty and thus were not used in the project, so we do not go over the cleaning process of this data. We elaborate on how the recordings were faulty and what was done instead to estimate black globe temperature data in section 3.

Rumen temperature data. We next discuss the cleaning process for the rumen temperature data.

Semi-automatic web scraping. The rumen temperature data collected via an internal bolus within each animal was available to download from smaXtec's web interface. We did not have direct access to the database that stored the data, so we used a Python package called PyAutoGUI (Sweigart, 2020) to automate the downloading of 90 rumen temperature data files from the web interface.

PyAutoGUI allows users to automate mouse and keyboard actions such as clicking, scrolling and typing. We wrote a Python script using PyAutoGUI to loop a series of actions so that we could efficiently download all 90 rumen temperature data files. The process was not completely automatic as someone was needed to supervise the program to stop and restart it if errors occurred. This script will be used again in future work, as there are still 180 animals that have similar data that needs to be collected. The script code is available in Appendix A.4.

Merging rumen temperature data files. We created a custom function to read in a rumen temperature data file given its file name. As an overview, it

1. obtained the timestamp and rumen temperature from a specific animal's data file,
2. obtained that animal's ID from the beginning of the file, and
3. combined the time stamp, animal ID and rumen temperature into a single data frame.

We used our custom function in conjunction with the purrr package from the tidyverse (Wickham *et al.*, 2019) to produce a single, stacked data frame of all rumen temperature data given a list of the rumen temperature data file names. The final data frame consisted of a little more than a million observations.

To ensure that we had downloaded all of the required rumen temperature data, the animal IDs in the resulting data frame was checked against a file consisting of animal details that was recorded prior to the beginning of the experiment.

Daylight savings and missing records. On the 4 April 2021, Sydney changed timezone from AEST to AEDT. This resulted in the temperature boluses recording two different entries for each animal between the hours of 2am and 3am. We dealt with this by simply removing the earlier of the two entries for each animal. This issue only occurred with the temperature boluses and not with the weather data collection instrument.

Solving this issue enabled us to convert the data frame into a tsibble object because tsibble requires that the timestamps across all rows are unique (per animal). In a similar fashion as with the weather data, we ensured that there were no missing records in the data by adding in blank rows at any missing timestamps.

Pen IDs and shade treatments. The file consisting of animal details mentioned earlier includes the pen ID and shade treatment corresponding to each animal. We joined our data frame of rumen temperature data to the data in this file using animal ID as the key.

Animal oddities. We noticed that although the boluses were equipped on 23 December 2020, data for most animals only started being reliably captured from 2 January 2021, so we discarded all observations that were recorded before 2 January 2021.

Additionally, animal 244 died before the experiment ended, so we removed it from the data frame.

Missing values. The rumen temperature occasionally exhibited contiguous sequences of missing values; some that were short (less than 2 hours missing) and some that were very long (a few days missing). We used a linear interpolation strategy for sequences that were less than 2 hours in length, implemented via the same custom-written interpolation function used for missing weather data (Appendix A.3).

With missing value sequences that were greater than 2 hours in length, we could not simply use linear interpolation because that would have erased time-of-day effects and temperature drops that occurred during drinking bouts. Instead, we decided to impute these sequences with synthetic data sourced from nearby past or future days of the same animal that began at the same hour and minute. For example, to fill in a chunk that began at 2021-01-05 10:00:00 and ended at 2021-01-05 23:30:00, we used that same animal's data beginning at 2021-01-06 10:00:00 and ending at 2021-01-06 23:30:00. Figure 4 illustrates an example of how this procedure works.

This was implemented via a custom function (Appendix A.3) so that we could repeat it for many animals. One animal (animal 321) however, had 19 days of consecutive missing values. This chunk was far too long to sensibly apply synthetic imputation, so we decided to leave these values as missing values and deal with them as needed during later analysis.

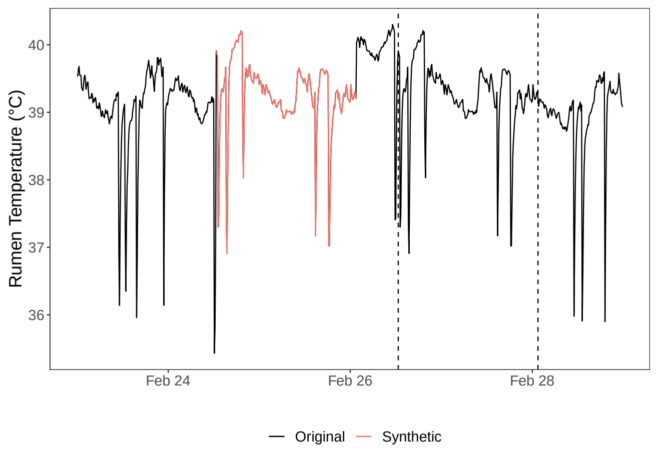


Fig. 4. Synthetic data being used to impute missing values. The portion of the time series between the dotted lines represents where the synthetic data was sourced from.

3. Methodology

The temperature boluses were located inside the rumen of each animal in order to provide accurate estimates of each animal's true rumen temperature every 10 minutes. An issue was that, when an animal began a drinking bout, its bolus reported drastically low temperatures due to the rush of cold water into the animal's rumen. We took advantage of this by improving upon an existing algorithm (Vázquez-Diosdado *et al.*, 2019) to detect drinking events based on drops in the rumen temperature data. When it came to modelling the rumen temperature based on weather variables, we used a variation of the rumen temperature variable where the drops in temperature had been smoothed out via a custom algorithm by smaXtec.

Detecting drinking events. Vázquez-Diosdado *et al.* (2019) suggested using an animal-day specific threshold to detect drinking events. Let $x_{i,n}$ be the rumen temperature measurement for animal i at time point n. Time point n is classified as a drinking event for animal i if $x_{i,n} < \bar{x}_{i,j_n} - s_{i,j_n}$, where j_n corresponds to the specific day that timestamp n occurs in. In other words, the algorithm

1. calculates the mean and standard deviation of the rumen temperature of each animal on each day, and then
2. classifies a point as a drinking event if it is more than one animal-day specific standard deviation s_{i,j_n} below its animal-day specific mean $\bar{x}_{i,j_n}$.

We developed and deployed an interactive web app (https://rajan-shankar.shinyapps.io/drinking_events) using the shiny R package (Chang *et al.*, 2020) to investigate whether we could improve on this existing approach by adding extra functionality. We incorporated tune-able hyperparameters into the web app based on the following:

- robust measures of centre and spread, i.e. the median and the median absolute deviation
- considering the first order differences of the rumen temperature time series
- using moving aggregation functions with varying window sizes as opposed to day-specific thresholds.

Users can control these hyperparameters to visualise how different drinking event detection methods fair on different animals' rumen temperature time series. We discuss selecting the optimal combination of hyperparameters as well as the mathematical formulation of the detection method in section 4.

Heat load indices. Certain weather variables are more influential than others when it comes to modelling the rumen temperature of cattle. Gaughan *et al.* (2008) constructed two indices based on weather variables—the heat load index (HLI) and the accumulated heat load (AHL)—that we used in combination with the shade treatment factor to analyse rumen temperature.

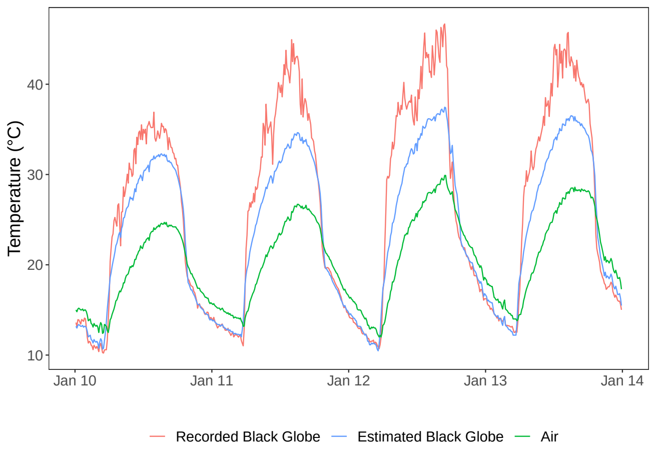


Fig. 5. Comparison of recorded and estimated BGT. The recorded BGT spikes up past the estimated BGT whenever the reading exceeds 20 °C.

One of the weather variables involved in the calculation of the HLI is the black globe temperature (BGT). We found that the instrument collecting BGT data was problematic, as it tended to exaggerate the BGT whenever it exceeded 20°C (Figure 5). We decided to instead estimate the BGT using air temperature and solar radiation (Hahn *et al.*, 2009) as

$$\widehat{T}_{\mathrm{BG}} = 1.33\ T_{\mathrm{air}} - 2.65\sqrt{T_{\mathrm{air}}} + 3.21\log_{10}(R_{\mathrm{solar}} + 1) + 3.5.$$

The HLI is then calculated as a function of the estimated BGT, relative humidity H_{rel} and wind speed W (Gaughan *et al.*, 2008) as

$$\mathrm{HLI} = \begin{cases} 10.66 + 0.28\ H_{\mathrm{rel}} + 1.3\ \widehat{T}_{\mathrm{BG}} - W & \text{if } \widehat{T}_{\mathrm{BG}} < 25 \\ 8.62 + 0.38\ H_{\mathrm{rel}} + 1.55\ \widehat{T}_{\mathrm{BG}} - 0.5\ W + e^{2.4-W} & \text{if } \widehat{T}_{\mathrm{BG}} \geq 25. \end{cases}$$

The AHL calculation involves the hyperparameters L, U and M, and is indexed by time t because it relies on previous HLI values (Gaughan *et al.*, 2008):

$$\mathrm{AHL}_t = \begin{cases} \max\left\{\left(\mathrm{AHL}_{t-1} + \frac{\mathrm{HLI}_{t-1}-L}{M}\right), 0\right\} & \text{if } \mathrm{HLI}_{t-1} < L \\ \mathrm{AHL}_{t-1} & \text{if } L \leq \mathrm{HLI}_{t-1} \leq U \\ \mathrm{AHL}_{t-1} + \frac{\mathrm{HLI}_{t-1}-U}{M} & \text{if } \mathrm{HLI}_{t-1} > U. \end{cases}$$

L and U are the lower and upper HLI thresholds, and are set depending on the breed of cattle. For our cattle, these were set to 77 and 86 respectively. M represents how often the HLI is collected every hour; our data was collected in 10-minute intervals, so this was set to 6.

Modelling. There were a few things that we needed to consider before we could build models using our data.

Smoothed rumen temperature. The main response variable for modelling purposes was the smoothed rumen temperature. The raw-recorded rumen temperature of an animal was not a good proxy for its true rumen temperature due to the presence of drops from drinking events. The smoothed rumen temperature was produced by the smaXtec web interface by:

1. calculating moving measures of centre and spread on the raw rumen temperature,
2. calculating a minimum threshold based on these measures, and then
3. for each temperature x_t below the minimum threshold, replacing it with $x_{t-1} - 0.05$.

Number of drinking events. Besides the smoothed rumen temperature, we also tried modelling the number of drinking events based on shade treatment. We used our best-performing drinking events detection method to count the number of drinking events per animal per day and used that as the response variable.

Random effects and packages. For both response variables, we used the lme4 package (Bates *et al.*, 2015) to implement linear mixed models. The package allowed us to incorporate animal ID as a random effect to account for animal-specific variation that was not of direct interest. We also used the emmeans package (Lenth, 2020), the car package (Fox and Weisberg, 2019) and the lmerTest package (Kuznetsova *et al.*, 2017) to test hypotheses for statistical significance.

Day/night aggregating. We aggregated our 10-minutely data into 6-hourly day summaries (over the hours of 10am – 4pm) and 6-hourly night summaries (over the hours of 12am – 6am). According to our industry expert Dr. Lees, summarising the data in this way allowed us to disentangle cooler night-time patterns and heat-of-day effects. The code that we used to produce this summarised version of our data is located in Appendix B.1.

4. Results

Drinking events. We first focus on results pertaining to when the response variable is the daily number of drinks.

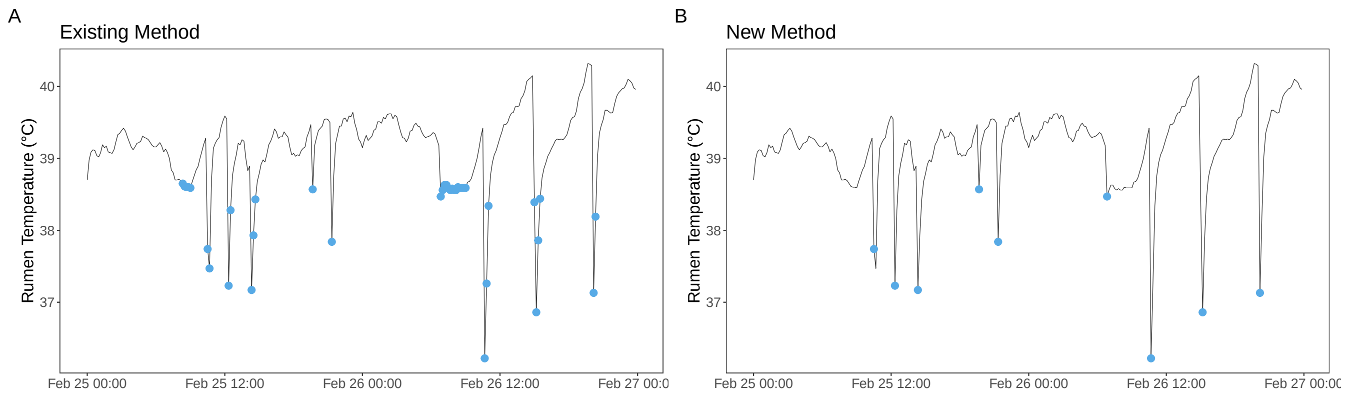


Fig. 6. Comparison of the existing method (plot A) and our proposed method (plot B) for detecting drinking events. The blue dots represent time points flagged as drinking events. Each sharp drop corresponding to a drinking event is only highlighted once in our method (plot B).

Detection method. Plot A in Figure 6 showcases the method developed by Vázquez-Diosdado *et al.* (2019) for detecting drinking events. There are two problems with this method; it picks up too many false positives (i.e. highlights time points that do not represent a drop in rumen temperature), and highlights multiple nearby time points that represent the same drinking event.

We alleviate the first problem by using robust measures of centre and spread—the median and the median absolute deviation (MAD)—instead of the mean and standard deviation to detect drops. The second problem arises from the fact that drinking events sometimes last longer than 10 minutes, which results in consecutive values that are similar to each other but relatively extreme to the rest of the temperature series. We deal with this problem by considering the differenced temperature series instead of the raw temperature series. Appendix C.1 illustrates how drinking events are detected on a differenced temperature series, using robust measures.

As a final step, if there are still multiple consecutive time points flagged as drinking events, we only identify the latest one as a drinking event. This sometimes occurs when the drop in rumen temperature is not instant; rather it happens in two or three decrements. Plot B in Figure 6 showcases our final method for detecting drinking events.

We now walk through the mathematical formulation of our drinking event detection method. Given a time series of 10-minutely recorded rumen temperatures $x_1, \dots, x_n$, the first order differences are

$$\begin{aligned} d_t &= x_t - x_{t-1}, \quad 1 < t \le n, \\ d_1 &= 0. \end{aligned}$$

The 24-hour rolling medians and median absolute differences (MAD) of the d_t are then

$$\begin{aligned} \text{med}_d(t) &= \text{median}\{d_{t-i} : 0 \le i < \min\{t, 144\}\} \\ \text{MAD}_d(t) &= \text{median}\{|d_{t-i} - \text{med}_d(t)| : 0 \le i < \min\{t, 144\}\}, \end{aligned}$$

for $1 \le t \le n$. We say that a time point t is a *candidate drinking event* if d_t is at least five 24-hour rolling MADs below its 24-hour rolling median. We denote the set of such time points t as

$$\mathscr{D}_{\text{cand}} = \{t : d_t \le \text{med}_d(t) - 5 \times \text{MAD}_d(t)\}.$$

We now calculate the final set of drinking events $\mathscr{D}$. If there is a *run* of two or more adjacent time points in $\mathscr{D}_{\text{cand}}$, then we add all time points except for the latest (highest) one in that run to a discard set denoted $\mathscr{X}$. We do this for every run in $\mathscr{D}_{\text{cand}}$. Thus, our final set of drinking events $\mathscr{D}$ is given by

$$\mathscr{D} = \mathscr{D}_{\text{cand}} \setminus \mathscr{X}.$$

Differences across shade treatments. Using our newly-developed method for detecting drinking events, a one-way ANOVA reveals that there is a significant difference in the mean daily number of drinks across the different shade treatments. We thus conduct a post-hoc test to identify exactly which pairs of shade treatments have different means. Figure 7 displays the 95% confidence intervals for each treatment pair combination. We see that it is the unshaded treatment that has a different mean compared to the other two treatments.

Rumen temperature. We next focus on results pertaining to when the response variable is the average rumen temperature.

Differences across shade treatments and time of day. Similar to the drinking events, we want to see if there is a significant difference in the mean daily average rumen temperature across the different shade treatments. This time however, the response is split into a day-average and a night-average, as plotted in Figure 8. One-way ANOVA results show that the means across the different shade treatments are only significantly different for the night-average rumen temperature. We thus conduct a post-hoc test to identify exactly

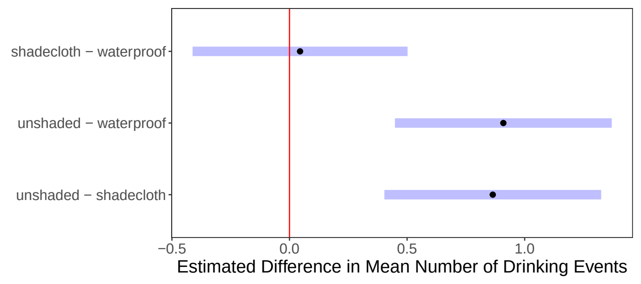


Fig. 7. Pair-wise contrasts of the estimated difference in the mean number of drinking events between the three different shade treatments. The lengths of the 95% confidence bars have been adjusted for multiple comparison issues via Tukey's HSD method.

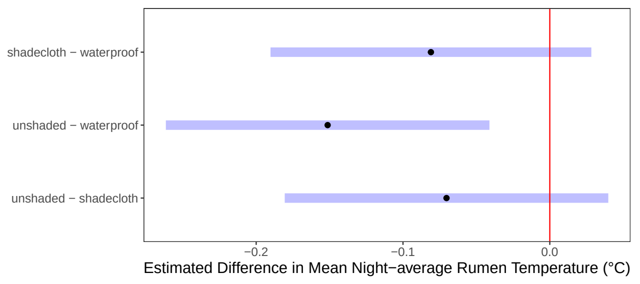


Fig. 9. Pair-wise contrasts of the estimated difference in the mean night-average rumen temperature between the three different shade treatments. The lengths of the 95% confidence bars have been adjusted for multiple comparison issues via Tukey's HSD method.

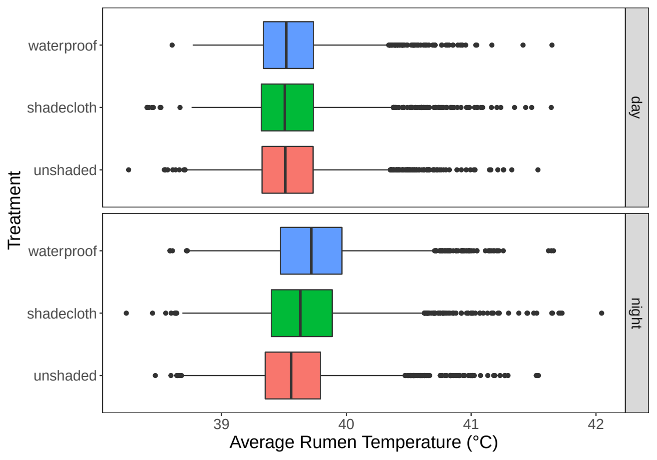


Fig. 8. Box plots of day-average rumen temperature faceted by shade treatment and time-of-day. There is no visible difference in the medians of the boxes in the day-time panel, but there is a pattern of decreasing medians in the night-time panel as such: waterproof > shade cloth > unshaded.

which pairs of shade treatments have different means. As shown in Figure 9, there is a significant difference in mean night-average rumen temperature between the unshaded and waterproof treatments.

Investigating interaction effects. In this subsection, we explore how the shade treatment factor can be combined with the AHL index to explain the day-average rumen temperature. We do not use the night-average because the AHL is always 0 during the night time. We further use the day-maximum AHL instead of the day-average due to the nature of the AHL—it is an *accumulated* measure, so it makes more sense to focus on its peaks.

Plot A in Figure 10 reveals that the slope relating day-maximum AHL to day-average rumen temperature appears to differ depending on the shade treatment. This suggests that there is a potential interaction effect between the shade treatment and the day-maximum AHL. ANOVA results confirm that the interaction term is indeed significant ($p < 0.001$).

Plot A in Figure 10 also reveals that there are only a handful of dates where the day-maximum AHL is substantially higher than 0. Out of these dates, only two of them (2021-01-27 and 2021-03-01) appear to follow the increasing trend in day-average rumen temperature. There are in fact, three weather variables—air temperature, relative humidity and vapour-pressure deficit—that have histograms of day-time 10-minutely observations that are distinctly different for those two aforementioned dates. Figure 11 depicts this difference using the vapour-pressure deficit (VPD) variable.

Taking note that air temperature and relative humidity are already indirectly involved in the calculation of the AHL, we would like to further investigate VPD. We want to produce a variable that separates days into those with low VPD and those with high VPD so that we can better explain the trends in day-average rumen temperature. Figure 12 reveals that the distribution of day-time third-quartile VPD is bimodal, so it makes sense to use a value close to the inflection point (day-time third-quartile VPD $= 2$) as a separation threshold.

Upon separating days using a day-time third-quartile VPD threshold of 2, we would like to see how the slopes differ between both treatment and VPD level. Plot B in Figure 10 reveals that the slopes are indeed quite different between the two VPD levels; however, we need to be careful to note that there are very few days with high AHL values in both VPD levels. We construct a three-way interaction model that incorporates interactions between the day-maximum AHL, shade treatment and VPD level, and its ANOVA results are shown in Table 1. Importantly, the three way interaction between treatment, max AHL and VPD threshold is statistically significant ($p = 0.004$).

Table 1. Three-way interaction model ANOVA

Source of variation	p value
treatment	0.944
max_ahl	< 0.001
vpd_threshold	< 0.001
treatment:max_ahl	< 0.001
treatment:vpd_threshold	0.003
max_ahl:vpd_threshold	< 0.001
treatment:max_ahl:vpd_threshold	0.004

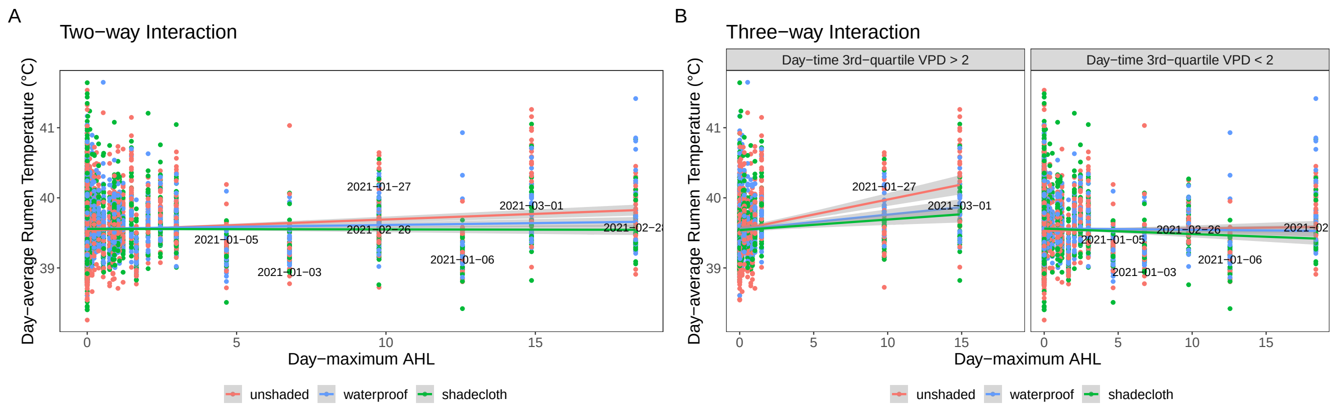


Fig. 10. Plots depicting interactions between shade treatment and AHL, and shade treatment, VPD level and AHL. Each column of data points represents a date, and these dates have been labelled for columns with high AHL.

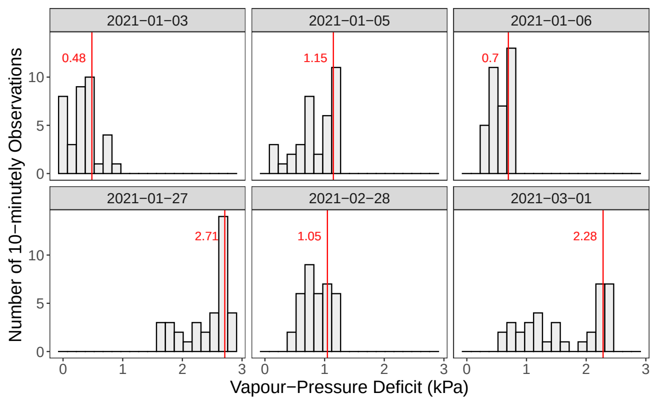


Fig. 11. Day-time 10-minutely VPD histograms of six high-AHL dates. The histograms for 2021-01-27 and 2021-03-01 are located much more to the right than the histograms of the other four dates. The red markers represent the third-quartile, and are relevant when considering Figure 12.

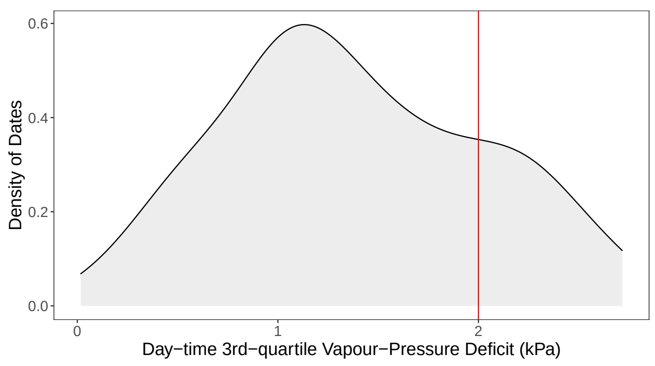


Fig. 12. Density plot of the day-time third-quartile VPD of all dates of the experiment. The inflection point at 2 is an ideal threshold for separating dates into cloudy days and clear days. We discuss the connection between VPD and cloud coverage in section 5. Using the third-quartile instead of the mean or median exaggerates the inflexion point and thus allows for better separation.

Tuning the HLI upper threshold in the AHL calculation via cross–validation. According to our industry expert Dr. Lees, decreasing the HLI upper threshold from its current value of 86 in the calculation of the AHL for our animals may increase the accuracy in predicting day-average rumen temperature. We test this theory using a cross-validation strategy with the two models that we constructed before—the three-way interaction model involving day-maximum AHL, shade treatment and VPD level, and the simpler two-way interaction model involving day-maximum AHL and shade treatment—trained on the following values of the HLI upper threshold: 86, 85, 84, 83, 82 and 81.

To assess the performance of competing models on unseen data, we perform a 10-times repeated 5-fold stratified cross-validation for each model, where we construct our folds by sampling across our 89 animals instead of sampling from 8000+ individual observations. Figure 13 shows the results of this procedure, along with 95% confidence error bars (constructed using the 10 repeats). There is substantial overlap across the error bars in both plots indicating that there is no significant difference in predictive performance between the different HLI upper thresholds considered.

5. Discussion

Drinking events. Our drinking event detection method was a nice, self-contained outcome for the project, as it managed to solve the problems that existed with the method developed by Vázquez-Diosdado *et al.* (2019). Looking at different time periods across different animals showed that our method was able to detect drops almost perfectly. However, it is important to note that we did not test our method in a supervised context; we did not have a variable in our data that contained information about when the animals were truly drinking.

We then saw that the mean daily number of drinks was significantly different across the different shade treatments; specifically, that it was higher for unshaded animals than

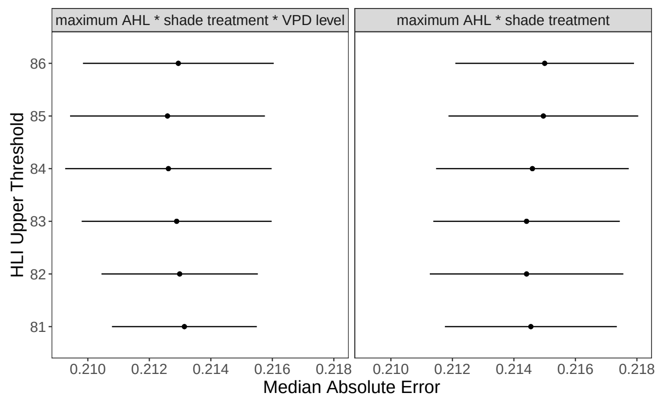


Fig. 13. 10-times repeated 5-fold stratified cross-validation results of our two models using different values of the HLI upper threshold hyperparameter. The predictive performance—measured by the median absolute error (MAE) between fitted values and true values—is not significantly different between the different hyperparameter values for either model.

for those under shade cloth or waterproof shelter. This increased frequency of drinks suggests that unshaded animals experience heat stress more readily. However, we do not know if these animals drank more water in total; they may have just taken smaller, more frequent drinks.

Rumen Temperature. We saw that the animals under the waterproof shelter were significantly warmer during the night than unshaded animals. This suggests that the waterproof shelter provides some form of thermal insulation during cooler hours. Alternatively, it may be that the waterproof shelter simply protected the animals against rainfall, as animals under the regular shade cloth were not significantly warmer than unshaded animals.

While the shade treatment alone did not have a significant effect on the mean day-average rumen temperature, there was a significant interaction effect when it was combined with the day-maximum AHL. This meant that when the day-maximum AHL was high (say, more than 10), then the shade treatment *did* have a significant effect on the mean day-average rumen temperature. Not surprisingly, it were the unshaded animals that had a higher mean day-average rumen temperature. In other words, unshaded animals were more prone to heat stress on high AHL days.

We also saw in Figure 10 that some dates did not support the increasing trend in day-average rumen temperature as the day-maximum AHL increased, and suspected that the vapour-pressure deficit (VPD) variable was the cause of this phenomenon. The VPD is connected to the relative humidity in that it measures the difference between the amount of moisture in the air and maximum moisture the air can hold when it is saturated. The VPD can be thought of as a measure of inverse cloud coverage; a high VPD means that there is not much moisture in the air, which leads to less cloud formation. Then, categorising dates into those with low VPD and those with high VPD can be interpreted as separating dates into cloudy days and clear days. The significant interaction terms in the model incorporating this binary separation variable in Table 1 support the utility of the VPD in explaining which dates follow the increasing trend in day-average rumen temperature and which dates do not. We must be careful to note that there are only two clear days with high day-maximum AHL values (2021-01-27 and 2021-03-01) in our data—primarily due to the relatively mild, cloudy summer experienced in 2021—and that it would be interesting to see if our three-way interaction model remains significant in future experiments with more high AHL and high VPD days.

Finally, we saw that decreasing the HLI upper threshold in the calculation of the AHL did not substantially improve the cross-validated prediction performance of either model. The current industry-standard upper threshold of 86 was constructed based on studies conducted in northern, subtropical regions, so it was worth checking to see if the same threshold was viable for cattle in a southern Australian climate.

6. Conclusion

In this project, we analysed the effects of different shade structures on the rumen temperature and drinking patterns of cattle. Our results provide important insights into when the meat industry should use shade structures to alleviate heat stress. As a concluding remark, Lean and Moate (2021) discuss the impact of climate change on the sustainability of the Australian cattle herd. They note that if we continue to see more extreme weather events as a result of climate change, more Australian cattle will need some form of shelter.

As part of our project, we also experimented with heatmap clustering methods using the superheat package (Barter and Yu, 2017) in an attempt to identify days where many cattle experienced higher-than-average heat stress (Appendix C.2). Due to time constraints, we were not able to form any conclusions about the usefulness of these methods, and so such methods should be considered in further research.

Other further research should consider using generalised additive models (GAMs) as a technique to model rumen temperature. Additionally, one could look into linking shade treatment effects to the eating quality of beef.

Acknowledgments. This research was funded by Meat and Livestock Australia grant B.FLT.4009. Guidance and advice was provided by Dr Garth Tarr (University of Sydney) and Dr Angela Lees (University of New England).

References

Barter R, Yu B (2017). *superheat: A Graphical Tool for Exploring Complex Datasets Using Heatmaps*. R package version 0.1.0, URL https://CRAN.R-project.org/package=superheat.

Bates D, Mächler M, Bolker B, Walker S (2015). "Fitting Linear Mixed-Effects Models Usinglme4." *Journal of Statistical Software*, **67**(1). doi:10.18637/jss.v067.i01. URL http://dx.doi.org/10.18637/jss.v067.i01.

Chang W, Cheng J, Allaire J, Xie Y, McPherson J (2020). *shiny: Web Application Framework for R*. R package version 1.5.0, URL https://CRAN.R-project.org/package=shiny.

Firke S (2020). *janitor: Simple Tools for Examining and Cleaning Dirty Data*. R package version 2.0.1, URL https://CRAN.R-project.org/package=janitor.

Fox J, Weisberg S (2019). *An R Companion to Applied Regression*. Third edition. Sage, Thousand Oaks CA. URL https://socialsciences.mcmaster.ca/jfox/Books/Companion/.

Gaughan J, Mader T, Holt S, Lisle A (2008). "A new heat load index for feedlot cattle1." *Journal of Animal Science*, **86**(1), 226–234. doi:10.2527/jas.2007-0305. URL http://dx.doi.org/10.2527/jas.2007-0305.

Hahn GL, Gaughan JB, Mader TL, Eigenberg RA (2009). *Chapter 5: Thermal Indices and Their Applications for Livestock Environments*, pp. 113–130. American Society of Agricultural and Biological Engineers. doi:10.13031/2013.28298. URL http://dx.doi.org/10.13031/2013.28298.

Kuznetsova A, Brockhoff P, Christensen R (2017). "lmerTest Package: Tests in Linear Mixed Effects Models." *Journal of Statistical Software*, **82**(13). doi:10.18637/jss.v082.i13. URL http://dx.doi.org/10.18637/jss.v082.i13.

Lean I, Moate P (2021). "Cattle, climate and complexity: food security, quality and sustainability of the Australian cattle industries." *Australian Veterinary Journal*. doi:10.1111/avj.13072. URL http://dx.doi.org/10.1111/avj.13072.

Lenth R (2020). *emmeans: Estimated Marginal Means, aka Least-Squares Means*. R package version 1.5.2-1, URL https://CRAN.R-project.org/package=emmeans.

Python Core Team (2020). *Python: A dynamic, open source programming language*. Python Software Foundation. URL https://www.python.org/.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Sweigart A (2020). *PyAutoGUI: A cross-platform GUI automation Python module for human beings*. URL https://readthedocs.org/projects/pyautogui/downloads/pdf/latest/.

Tierney N (2017). "visdat: Visualising Whole Data Frames." *The Journal of Open Source Software*, **2**(16), 355. doi:10.21105/joss.00355. URL http://dx.doi.org/10.21105/joss.00355.

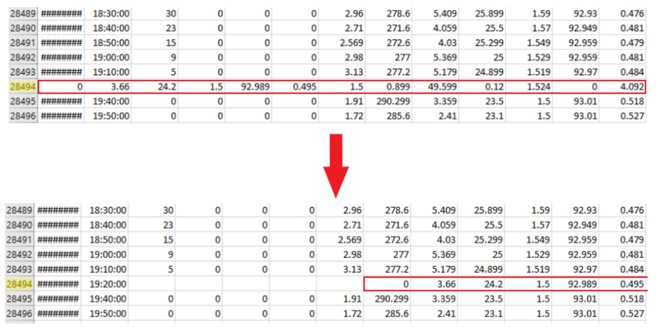
Vázquez-Diosdado J, Miguel-Pacheco G, Plant B, Dottorini T, Green M, Kaler J (2019). "Developing and evaluating threshold-based algorithms to detect drinking behavior in dairy cows using reticulorumen temperature." *Journal of Dairy Science*, **102**(11), 10471–10482. doi:10.3168/jds.2019-16442. URL http://dx.doi.org/10.3168/jds.2019-16442.

Wang E, Cook D, Hyndman R (2020). "A New Tidy Data Structure to Support Exploration and Modeling of Temporal Data." *Journal of Computational and Graphical Statistics*, **29**(3), 466–478. doi:10.1080/10618600.2019.1695624. URL http://dx.doi.org/10.1080/10618600.2019.1695624.

Wickham H, Averick M, Bryan J, Chang W, McGowan L, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen T, Miller E, Bache S, Müller K, Ooms J, Robinson D, Seidel D, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the Tidyverse." *Journal of Open Source Software*, **4**(43), 1686. doi:10.21105/joss.01686. URL http://dx.doi.org/10.21105/joss.01686.

7. Appendix A: Data Processing and Cleaning

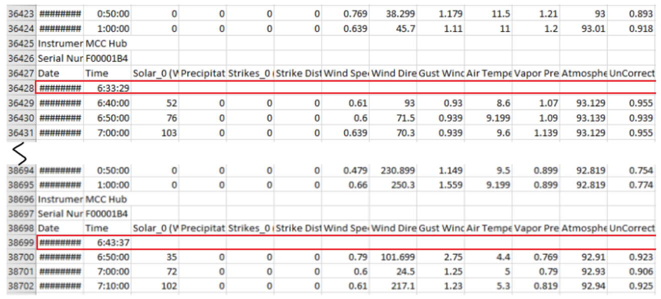
A.1. *Manual edits.* Figures 14 and 15 show the manual edits we made to the weather data in Microsoft Excel.



28489	########	18:30:00	30	0	0	0	2.96	278.6	5.409	25.899	1.59	92.93	0.476
28490	########	18:40:00	23	0	0	0	2.71	271.6	4.059	25.5	1.57	92.949	0.481
28491	########	18:50:00	15	0	0	0	2.569	272.6	4.03	25.299	1.549	92.959	0.479
28492	########	19:00:00	9	0	0	0	2.98	277	5.369	25	1.529	92.959	0.481
28493	########	19:10:00	5	0	0	0	3.13	277.2	5.179	24.899	1.519	92.97	0.484
28494	0	3.66	24.2	1.5	92.989	0.495	1.5	0.899	49.599	0.12	1.524	0	4.092
28495	########	19:40:00	0	0	0	0	1.91	290.299	3.359	23.5	1.5	93.01	0.518
28496	########	19:50:00	0	0	0	0	1.72	285.6	2.41	23.1	1.5	93.01	0.527

28489	########	18:30:00	30	0	0	0	2.96	278.6	5.409	25.899	1.59	92.93	0.476
28490	########	18:40:00	23	0	0	0	2.71	271.6	4.059	25.5	1.57	92.949	0.481
28491	########	18:50:00	15	0	0	0	2.569	272.6	4.03	25.299	1.549	92.959	0.479
28492	########	19:00:00	9	0	0	0	2.98	277	5.369	25	1.529	92.959	0.481
28493	########	19:10:00	5	0	0	0	3.13	277.2	5.179	24.899	1.519	92.97	0.484
28494	########	19:20:00						0	3.66	24.2	1.5	92.989	0.495
28495	########	19:40:00	0	0	0	0	1.91	290.299	3.359	23.5	1.5	93.01	0.518
28496	########	19:50:00	0	0	0	0	1.72	285.6	2.41	23.1	1.5	93.01	0.527

Fig. 14. Row 28494's values did not line up with the correct variables, so we shifted it to the right.



36423	########	0:50:00	0	0	0	0	0.769	38.299	1.179	11.5	1.21	93	0.893
36424	########	1:00:00	0	0	0	0	0.639	45.7	1.11	11	1.2	93.01	0.918
36425	Instrumer	MCC Hub											
36426	Serial Nur	F00001B4											
36427	Date	Time	Solar_0 (V	Precipitat	Strikes_0 (	Strike Dist	Wind Spe	Wind Dire	Gust Winc	Air Tempe	Vapor Pre	Atmosphe	UnCorrect
36428	########	6:33:29											
36429	########	6:40:00	52	0	0	0	0.61	93	0.93	8.6	1.07	93.129	0.955
36430	########	6:50:00	76	0	0	0	0.6	71.5	0.939	9.199	1.09	93.139	0.939
36431	########	7:00:00	103	0	0	0	0.639	70.3	0.939	9.6	1.139	93.129	0.955

38694	########	0:50:00	0	0	0	0	0.479	230.899	1.149	9.5	0.899	92.819	0.754
38695	########	1:00:00	0	0	0	0	0.66	250.3	1.559	9.199	0.899	92.819	0.774
38696	Instrumer	MCC Hub											
38697	Serial Nur	F00001B4											
38698	Date	Time	Solar_0 (V	Precipitat	Strikes_0 (	Strike Dist	Wind Spe	Wind Dire	Gust Winc	Air Tempe	Vapor Pre	Atmosphe	UnCorrect
38699	########	6:43:37											
38700	########	6:50:00	35	0	0	0	0.79	101.699	2.75	4.4	0.769	92.91	0.923
38701	########	7:00:00	72	0	0	0	0.6	24.5	1.25	5	0.79	92.93	0.906
38702	########	7:10:00	102	0	0	0	0.61	217.1	1.23	5.3	0.819	92.94	0.925

Fig. 15. Rows 36428 and 38699 had no useful values, so we deleted them.

A.2. *Cleaning code pipelines.* The cleaning code for the weather data is provided below:

```
weather = read_csv("weather_data/2021␣April␣20_edited.csv",
                   col_types = cols(Date = col_date(format = "%d/%m/%Y")),
                   skip = 2,
                   guess_max = 3) %>%
  janitor::clean_names() %>%
  slice(-1:-22431) %>%
  janitor::remove_empty(which = "rows", quiet = FALSE) %>%
  select(-c(battery_voltage_v, battery_temperature_deg_c,
            external_supply_v, diagnostic_message)) %>%
  mutate(timestamp = as.POSIXct(paste(date, time), tz = "UTC")) %>%
  select(timestamp, everything(), -date, -time) %>%
  mutate(timestamp = as.POSIXct(round.POSIXt(timestamp, "mins"))) %>%
  mutate(across(c(wind_direction_0_deg, wind_speed_0_m_s), ~ na_if(.x, -9990))) %>%
  mutate(un_corrected_rh_0_rh = case_when(un_corrected_rh_0_rh > 1 ~ 1,
                                          TRUE ~ un_corrected_rh_0_rh),
         corrected_rh_0_percent_rh = case_when(corrected_rh_0_percent_rh > 100 ~ 100,
                                               TRUE ~ corrected_rh_0_percent_rh),
         evapotranspiration_mm = case_when(evapotranspiration_mm < 0 ~ 0,
                                           TRUE ~ evapotranspiration_mm),
         vpd_k_pa = case_when(vpd_k_pa < 0 ~ 0, TRUE ~ vpd_k_pa),
         air_temperature_0_deg_c = case_when(air_temperature_0_deg_c < 0 ~ 0,
                                             TRUE ~ air_temperature_0_deg_c)) %>%
  tsibble() %>%
  fill_gaps() %>%
  mutate(across(where(is.numeric), ~ linterp(.x, 1000)))
```

The cleaning code for the rumen temperature data is provided below:

```
cohort_1 = readxl::read_excel("animal_details/cohort␣1.␣animal␣details.xlsx")

cohort_1_ids = cohort_1 %>%
  drop_na(rumen_bolus_id) %>%
  pull(animal_id)

file_names = list.files("internal_temps/export/")

rumen_temp = file_names %>%
  map_df(.f = read_fn) %>%
  distinct() %>%
  filter(animal_id %in% cohort_1_ids) %>%
  group_by(animal_id) %>%
  distinct(timestamp, .keep_all = TRUE) %>%
  ungroup() %>%
  tsibble(key = animal_id) %>%
  fill_gaps(.full = TRUE) %>%
  left_join(cohort_1, by = "animal_id") %>%
  select(timestamp, animal_id, pen_id, treatment,
         temperature_c, temp_without_drinkcycles) %>%
  filter(animal_id != 244) %>%
  filter(timestamp >= as.Date("2021-01-02"),
         timestamp <= as.Date("2021-04-07")) %>%
  group_by_key() %>%
  mutate(temperature_c = linterp(temperature_c, max_length = 12),
         temp_without_drinkcycles = linterp(temp_without_drinkcycles,
                                            max_length = 12)) %>%
  replace_with_synth_data(25 , "2021-01-02␣09:00:00","2021-01-08␣17:00:00","fwd") %>%
  replace_with_synth_data(163, "2021-01-28␣09:30:00","2021-01-29␣01:20:00","fwd") %>%
  replace_with_synth_data(163, "2021-02-24␣12:40:00","2021-02-26␣01:30:00","fwd") %>%
  replace_with_synth_data(299, "2021-01-09␣10:20:00","2021-01-09␣19:20:00","fwd") %>%
  replace_with_synth_data(358, "2021-01-02␣00:00:00","2021-01-08␣14:50:00","fwd") %>%
  replace_with_synth_data(105, "2021-01-02␣00:00:00","2021-01-02␣05:40:00","fwd") %>%
  replace_with_synth_data(134, "2021-03-15␣11:00:00","2021-03-15␣18:20:00","fwd") %>%
  replace_with_synth_data(168, "2021-03-20␣08:30:00","2021-03-23␣17:40:00","fwd") %>%
  replace_with_synth_data(187, "2021-03-15␣11:10:00","2021-03-19␣06:50:00","fwd") %>%
  replace_with_synth_data(257, "2021-03-15␣10:50:00","2021-03-24␣00:30:00","fwd") %>%
  replace_with_synth_data(262, "2021-03-15␣11:00:00","2021-03-16␣04:00:00","fwd") %>%
  replace_with_synth_data(262, "2021-03-22␣22:10:00","2021-03-23␣09:30:00","fwd") %>%
  replace_with_synth_data(267, "2021-04-06␣07:00:00","2021-04-07␣00:00:00","bck") %>%
  replace_with_synth_data(330, "2021-03-15␣11:00:00","2021-03-15␣15:20:00","fwd") %>%
  replace_with_synth_data(368, "2021-04-06␣07:00:00","2021-04-07␣00:00:00","bck") %>%
  mutate(temp_without_drinkcycles = linterp(temp_without_drinkcycles, max_length = 12))
```

A.3. *Custom-written functions.* The function for performing linear interpolation is provided below:

```
linterp = function(series, max_length) {

  a = series
  start_pos = 0
  start_val = 0
  end_pos = 0
  end_val = 0
  ravine = FALSE

  for (i in 1:length(a)) {

    if (is.na(a[i]) && ravine == FALSE) {
      start_pos = i
      ravine = TRUE
      if (start_pos == 1) {
        start_val = -999
      } else {
        start_val = a[i-1]
      }
    }

    if (!is.na(a[i]) && ravine == TRUE) {
      end_pos = i-1
      end_val = a[i]
      ravine = FALSE

      if (start_pos == 1) {
        start_val = end_val
      }

      len = end_pos-start_pos+1
      imputed_vals = approx(c(start_val, end_val), n = len+2)$y[c(-1, -(len+2))]

      if (len <= max_length) {
        for (j in start_pos:end_pos) {
          a[j] = imputed_vals[j-start_pos+1]
        }
        if (len >= 2) {
          message(paste0("Imputed␣over␣", len, "␣timepoints"))
        }
      } else {
        message(paste0("Skipped␣", len, "␣timepoints"))
      }

    } else if (i == length(a) && ravine == TRUE) {
      end_pos = i
      end_val = start_val
      ravine = FALSE

      len = end_pos-start_pos+1
      imputed_vals = approx(c(start_val, end_val), n = len+2)$y[c(-1, -(len+2))]

      if (len <= max_length) {
        for (j in start_pos:end_pos) {
          a[j] = imputed_vals[j-start_pos+1]
        }
        if (len >= 2) {
          message(paste0("Imputed␣over␣", len, "␣timepoints"))
        }
      } else {
        message(paste0("Skipped␣", len, "␣timepoints"))
      }
    }
  }

  return(a)
}
```

The function for reading in a rumen temperature data file given its file name is provided below:

```
read_fn = function(file_name){

  df = readxl::read_excel(paste0("internal_temps/export/", file_name), skip = 10) %>%
    janitor::clean_names() %>%
    select(timestamp, temperature_c, temp_without_drinkcycles)

  animal_id = readxl::read_excel(paste0("internal_temps/export/", file_name),
                                 range = "B3") %>%
    colnames() %>% as.numeric()

  df = df %>% mutate(animal_id = animal_id)

  return(df)
}
```

The functions for interpolating long sequences of missing values with synthetic data are provided below:

```
synth_data = function(dat, id, start_time, end_time,
                      pull_from = c("bck","fwd"), extra_days = 0) {

  dat_copy = dat

  start_time = ymd_hms(start_time)
  end_time = ymd_hms(end_time)
  difference = difftime(end_time, start_time, units = "days")
  n_days = ceiling(difference) + extra_days

  if (pull_from == "bck") {
    synth_start = start_time - days(n_days)
    synth_end = synth_start + difference
  } else if (pull_from == "fwd") {
    synth_end = end_time + days(n_days)
    synth_start = synth_end - difference
  }

  dat_copy = dat %>%
    filter(animal_id == id,
           timestamp >= synth_start,
           timestamp <= synth_end) %>%
    select(temperature_c, temp_without_drinkcycles)

  return(dat_copy)
}


replace_with_synth_data = function(dat, id, start_time, end_time,
                                   pull_from = c("bck","fwd"), extra_days = 0) {

  dat_copy = dat

  dat_copy$temperature_c[
    dat_copy$animal_id == id &
      dat_copy$timestamp >= ymd_hms(start_time) &
      dat_copy$timestamp <= ymd_hms(end_time)
  ] = synth_data(dat, id, start_time, end_time, pull_from) %>%
      pull(temperature_c)

  dat_copy$temp_without_drinkcycles[
    dat_copy$animal_id == id &
      dat_copy$timestamp >= ymd_hms(start_time) &
      dat_copy$timestamp <= ymd_hms(end_time)
  ] = synth_data(dat, id, start_time, end_time, pull_from) %>%
      pull(temp_without_drinkcycles)

  return(dat_copy)
}
```

A.4. *Semi-automatic web scraping script.* The PyAutoGUI script for scraping the rumen temperature data from smaXtec's web interface is provided below:

```
alternator = True
aug.PAUSE = 0.5; aug.click(x=51, y=14); aug.moveTo(x=1685, y=602); aug.scroll(-58)

for i in range(100):

    aug.click()
    aug.moveTo(x=445, y=263)
    aug.click()
    aug.hotkey('ctrl', 'a')
    aug.hotkey('backspace')
    aug.typewrite('12/01/2020')
    aug.moveTo(x=445, y=305)
    aug.click()
    aug.hotkey('ctrl', 'a')
    aug.hotkey('backspace')
    aug.typewrite('04/14/2021')
    aug.moveTo(x=1753, y=495)
    aug.PAUSE = 3
    aug.click()

    while True:
        image = aug.screenshot()
        if image.getpixel((572, 106)) != (166, 206, 57):
            break

    aug.PAUSE = 0.5; aug.moveTo(x=1685, y=650)

    if alternator:
        aug.scroll(-58)
        alternator = False
    else:
        aug.scroll(-57)
        alternator = True
```

8. Appendix B: Methodology

B.1. *Day/night aggregating code.* The code for aggregating the weather data and rumen temperature data into day/night components is provided below:

```
# Read rumen temp
rumen_temp = read_csv("rumen_temp.csv") %>%
  group_by(animal_id) %>%
  mutate(diff_temperature_c = difference(temperature_c),
         diff_temperature_c = case_when(is.na(diff_temperature_c) ~ 0,
                                        TRUE ~ diff_temperature_c),
         rolling_centre = slide_dbl(diff_temperature_c, median, .before = 246),
         rolling_spread = slide_dbl(diff_temperature_c, mad, .before = 246),
         drink_event    = case_when(diff_temperature_c <
                                      rolling_centre - 5 rolling_spread ~
                                      temperature_c),
         lead           = lead(drink_event),
         drink_event    = case_when(is.na(lead) ~ drink_event)) %>%
  ungroup() %>%
  select(timestamp, animal_id, treatment,
         temperature_c, temp_without_drinkcycles, drink_event)

# Read weather
weather = read_csv("weather.csv") %>%
  mutate(calculated_bgt = 1.33 air_temperature_0_deg_c -
           2.65 sqrt(air_temperature_0_deg_c) +
           3.21 log10(solar_0_w_m_2 + 1) + 3.5,
         hli = case_when(calculated_bgt <  25 ~ 10.66 +
                           0.28 corrected_rh_0_percent_rh +
                           1.3 calculated_bgt - wind_speed_0_m_s,
                         calculated_bgt >= 25 ~ 8.62  +
                           0.38 corrected_rh_0_percent_rh + 1.55 calculated_bgt -
                           0.5 wind_speed_0_m_s + exp(2.4-wind_speed_0_m_s)),
         ahl_77_86 = calculate_ahl(hli, 77, 86, 6),
         ahl_77_85 = calculate_ahl(hli, 77, 85, 6),
         ahl_77_84 = calculate_ahl(hli, 77, 84, 6),
         ahl_77_83 = calculate_ahl(hli, 77, 83, 6),
         ahl_77_82 = calculate_ahl(hli, 77, 82, 6),
         ahl_77_81 = calculate_ahl(hli, 77, 81, 6))

# Summarise rumen temp
rumen_temp_summarised = rumen_temp %>%
  drop_na(temp_without_drinkcycles) %>%
  mutate(date = date(timestamp)) %>%
  group_by(date, animal_id) %>%
  mutate(drink_events = sum(!is.na(drink_event))) %>%
  mutate(light = case_when(hour(timestamp) >= 0  & hour(timestamp) < 6  ~
                             "night",
                           hour(timestamp) >= 10 & hour(timestamp) < 16 ~
                             "day")) %>%
  drop_na(light) %>%
  group_by(date, animal_id, light) %>%
  summarise(treatment = treatment[1],
            drink_events = drink_events[1],
            avg_temp = mean(temp_without_drinkcycles),
            min_temp = min(temp_without_drinkcycles),
            max_temp = max(temp_without_drinkcycles)) %>%
  ungroup()

# Summarise weather
weather_summarised = weather %>%
  mutate(date = date(timestamp)) %>%
  mutate(light = case_when(hour(timestamp) >= 0  & hour(timestamp) < 6  ~
                             "night",
                           hour(timestamp) >= 10 & hour(timestamp) < 16 ~
                             "day")) %>%
  drop_na(light) %>%
  group_by(date, light) %>%
  summarise(max_ahl = max(ahl_77_86),
            max_ahl_77_85 = max(ahl_77_85),
            max_ahl_77_84 = max(ahl_77_84),
            max_ahl_77_83 = max(ahl_77_83),
            max_ahl_77_82 = max(ahl_77_82),
            max_ahl_77_81 = max(ahl_77_81),
            avg_vpd = mean(vpd_k_pa),
            qr3_vpd = quantile(vpd_k_pa)["75%"]) %>%
  ungroup() %>%
  mutate(vpd_threshold = case_when(qr3_vpd > 2 ~ "greater than",
                                   qr3_vpd <= 2 ~ "less than"))

# Join
day_night_agg = left_join(rumen_temp_summarised, weather_summarised,
                          by = c("date", "light"))
```

9. Appendix C: Results

C.1. *Detecting drinking events using the differenced temperature series.* Figure 16 illustrates an example of using a differenced temperature series to detect drinking events. The drops are much more noticeable.

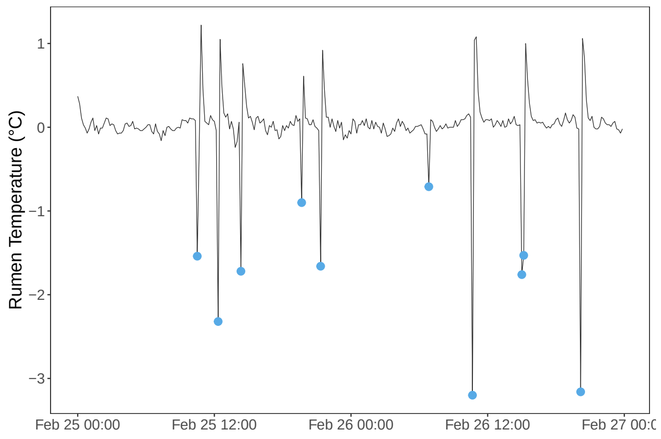


Fig. 16. Detecting drinking events on the differenced temperature series using robust measures of centre and spread. The blue dots represent time points flagged as drinking events.

C.2. *Detecting heat events via clustering.* Figure 17 illustrates the use of hierarchical clustering to detect days with heat events.

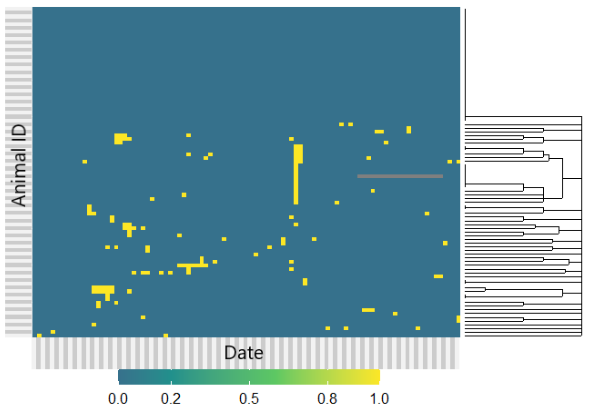


Fig. 17. Hierarchical clustering to try to identify days where animals experienced more-than-average heat stress. Each yellow cell represents an animal/day combination where that animal experienced heat stress on that day.