Australian Government
Bureau of Meteorology

A bias correction method designed for weather and climate extremes

Andrew Dowdy

October 2023

Bureau Research Report - 087

A bias correction method designed for weather and climate extremes

Andrew Dowdy[1,2,3]

[1]Bureau of Meteorology, Docklands, VIC, Australia

[2]ARC Centre of Excellence for Climate Extremes, The University of Melbourne, Melbourne, VIC, Australia

[3]School of Geography, Earth and Atmospheric Sciences, The University of Melbourne, Melbourne, VIC, Australia

Bureau Research Report No. 087

October 2023

National Library of Australia Cataloguing-in-Publication entry

Author: Andrew Dowdy

Title: A bias correction method designed for weather and climate extremes

ISBN: 978-1-925738-75-9

ISSN: 2206-3366

Series: Bureau Research Report – BRR087

Enquiries should be addressed to:

Author: Andrew Dowdy

University of Melbourne
Grattan St, Parkville,
Victoria 3052, Australia

andrew.dowdy@unimelb.edu.au

Copyright and Disclaimer

Contents

Abstract

This report provides documentation for a bias correction method designed for use with weather and climate data. Given the importance of extremes for hazards, this method was designed with attention to detail around the extremes of the sample data. The method is known as Quantile Matching for Extremes (QME), with example code and results presented here for bias correcting model output using observations-based data as the reference. The method is univariate and does not include spatial or temporal adjustment factors when calculating the bias correction factors, making it relatively quick and simple to apply with relatively low computational requirements. It doesn't require assumptions about the shape of distribution tails for extreme values and it retains the rank order of data at a location in the bias corrected output data. The method's code has a modular structure for its components and is designed to be adaptable depending on the application, including for other data types, regions, or for data with different time steps (such as sub-daily, monthly, etc.), noting that other methods can subsequently be applied to the bias corrected QME output data (such as if wanting to consider rarer extremes using parametric approaches based on GEV distribution, etc.).

The method is based on quantile-quantile matching of histograms for the Input Data (e.g., model data to be bias corrected) and Reference Data (e.g., observations-based data). Weather and climate processes were considered throughout the design of the method, leading to the development of several features of this method. For example, 500 histogram bins are used together with scaling of the data to help resolve details in the sample extremes. However, rather than applying quantile-quantile matching over the full histogram range, a user-defined limit can be selected for this, such as not doing this matching for values equal to or exceeding the third highest value. For those values, the bias correction amount is used as calculated for the neighbouring histogram bin (i.e., the histogram bin that is one place less extreme than the third highest value in the sample). This way of handling the sample upper extremes, and similarly for the lower extremes, can help avoid potentially excessive influences from very rare events that might sometimes occur in a sample period. Another one of the optional features of this method can be used to account for large shifts in a temperature distribution over time, by removing the long-term trend from the temperature data prior to applying the bias correction, then adding the trend back in after the bias correction has been applied. This can be useful for application to model projections of future climate changes, with weather features (e.g., fronts, cyclones and highs) superimposing short-term temperature variations on long-term climate trends.

1. Introduction

While weather and climate models have generally shown improvement over time in their simulation of physical processes, their output data can still contain some systematic biases. These biases can relate to a wide range of factors, such as model resolution typically being coarser than the scale of some physical processes (including

for convection, turbulence, orographic features, etc.). To help with the use of data in practical applications, methods for bias correction (or sometimes also called bias adjustment) can be applied for post processing the model outputs. There are numerous different types of methods available for bias correction, including that can be applied across various spatio-temporal scales, as described in studies such as Teutschbein and Seibert (2012), Maraun et al. (2017) and references therein.

This report describes the QME method for bias correction that has been developed specifically for application to weather and climate model data, including with attention to detail for considering how extremes are handled. This method was originally developed in 2019 as research code (written in the programming language IDL), with subsequent updates including for application to more models and variables. This report provides documentation for the QME method including detailing the key features of the method and individual components of the code structure. Examples of results are also provided for an application using climate model data for Australia.

The IDL-based QME code is included in the Appendix here. This has enhancements over the previous development versions of the code, with the code now written in a style intended for broader use and applications (rather than for research purposes as was previously the case). The code also now allows monthly application, including with options such as a 3-month moving average included, whereas earlier versions were designed only for application to four individual seasons (December–February, March–May, June–August and September–November). There are several examples of previous research and development applications using the QME method and bias corrected data in past studies (Dowdy et al. 2019, 2021; Srikanthan et al. 2022; Wilson et al. 2022; Vogel et al. 2023; Wasco et al. 2023). As noted in research studies such as those, bias corrected model data based on applying the QME method have been used for various climate change adaptation and decision-making purposes, including in the Energy Sector for Climate Information (ESCI) project, the National Hydrological Projections (NHP) project, the National Environmental Science Program (NESP), the Australian Climate Service (ACS) and several other similar projects.

2. Method overview and code structure

2.1. General overview of the code structure

Some general details on the code structure are described in this section. The code is written in IDL and called "qme.pro" as documented here in the Appendix. The key procedures in the code are listed as follows in the dot points below here. In addition to those procedures, several other procedures are used that are more self-explanatory (such as for making a 3-month moving average, reading and writing netcdf files and for postscript graphics settings).

- "read_input_data " – This procedure reads in the Input Data intended to be bias corrected (e.g., model data).
- "read_reference_data " – This procedure reads in the Reference Data used to aim for when bias correcting the Input Data (e.g., observations-based data).
- "scale_data" – This procedure scales the data to fill more of the histogram range (from 0 to 500) to help provide enhanced detail on extremes.
- "unscale_data" – This procedure reverts from the scaled values back to the unscaled values.
- "limit_data" – This procedure ensures that the scaled values don't exceed the histogram range of values (from 0 to 500).
- "train_qme" – This procedure calculates the bias correction amount, based on the histograms for the Input Data and Reference Data.

Following the procedures in the code, the main body of the code then has several components, with options to switch these on or off, depending on which aspect is intended to be run. These components of the code are listed in the dot points below, with further details provided in subsequent sections of this document. These dot points are listed in the flow order that they are run in the code, starting first with reading the Input Data and the Reference Data, then accounting for the temperature trend if this option was selected, then training the bias correction and then applying the bias correction.

- "run_input_read" – Read the Input Data to be bias corrected (e.g., model data) and populate a histogram.
- "run_reference_read" – Read the Reference Data (e.g., observations-based data) and populate a histogram.
- "run_account trend" – Optional step to calculate long-term mean temperatures, such as for temperature trends, if wanting better matching of histograms that might have large changes in their average position in the future due to climate change.
- "run_training" – Use the Input Data and Reference Data histograms to calculate a calibration factor (i.e., bias correction amount) for each of the histogram bins.
- "run_apply_cal" – Apply the calibration factors to all Input Data, including if need be for data not used in the training period (e.g., model projections of future climate).

The components in the main body of the QME code are described in more detail in the following subsections. The components for reading the Input Data and Reference Data into histograms are described in Section 2.2, the component for training the bias correction factors based on the histograms of the Input Data and Reference Data is described in Section 2.3, with the application of the bias correction to the Input Data described in Section 2.4.

2.2. Reading the data for training method

The "run_input_read" and "run_reference_read" components of the code read the Input Data (e.g., time series of model simulation data for a given location) and Reference Data (e.g., time series of observations-based data for a given location), respectively. This is done to populate a histogram for the Input Data and a histogram for the Reference Data. Those are the histograms that are used in subsequent sections of the code for calculating the calibration amount for each quantile of the histograms (e.g., currently using 501 integer bins in the histograms, from 0 to 500).

The code allows application for a single grid cell as well as one- or two-dimensional arrays of grid cells. There is an option to use a mask file for selecting the subset of array locations to apply the code to, such as if only wanting to apply it for some land locations, while noting that ocean locations or other atmospheric levels above the surface can also be used if needed for a particular application.

The code as shown in the Appendix is set up as an example for application to weather and climate data gridded in latitude and longitude through Australia (including for some current work in collaboration with an initiative called the National Partnership for Climate Projections: NPCP). The examples presented in this report are not intended for any other purpose than to document and demonstrate the application of the QME method, rather than for any decision-making purposes relating to climate or climate change guidance. For Reference Data in this documentation report example, the AGCD version 1 (also known as AWAP) dataset is used (Jones et al. 2009), comprising a gridded analysis of daily observations data. It is used here on a 0.2-degree grid in latitude and longitude, while noting previous research applications of QME during development stages have used other grids including the native 0.05-degree AGCD grid. For the current example, the Input Data are from BARPA downscaling of ACCESS-ESM1-5 data (from the CMIP6 set of global climate models) using the SSP3-7.0 emissions pathway, as well as from BARPA downscaling of ERA5 reanalysis (Hersbach et al. 2020; Su et al. 2022).

At each grid cell location, a histogram (i.e., occurrence frequency distribution) is produced for the training period, including for both the Input Data and for the Reference Data. For this example application using ERA5-based BARPA downscaling for the Input Data, a training period is used for historical years from 1980 to 1999, as that allows the following 20 years from 2000 to 2019 to be used for demonstrating the application of the method to data that are independent of the training period. This example is detailed further in Section 3, including some figures showing results based on the output from applying the QME method to the data as described here. For practical applications (rather than for demonstration purposes as shown in this report), it is recommended to use to longest period possible of high-quality data for training. For example, previous applications of the QME method, such as those in the reference list provided in Section 1 here, have used training periods of about 40–50 years. Training periods shorter than this are generally not recommended for variables such as rainfall that may have large interannual variability including for the occurrence of rainfall extremes.

After the observations and model data are read in, they are limited in range using the procedure in the code called "limit_data", then scaled using the procedure called "scale_data". For each grid cell location, the data are used to populate a histogram for the observations and a histogram for the model data. This is done individually for each month of the year.

The histograms have 501 bins for this Australian application example. The limits and scaling procedures mentioned above are designed so that the processed data will then fit within the range 0 to 500. Examples of limits used in this application example are from -30 to 60 C for tasmax, from -45 to 50 C for tasmin, as well as from 0 to 1250 mm/day for pr, while noting that users can add their own scaling and limits including for new variables they may like to run this code on. Other variables have been used previously for QME applications, including wind speed, humidity, solar radiation and fire weather measures (such as the FFDI). The scaling can be very simple such as for temperature variables here, or more complex such as for rainfall, as follows:

- (tasmax + 35) * 5
- (tasmin + 55) * 5
- alog(pr + 1) * 70, where alog is the natural logarithm

The rainfall (pr) scaling is designed to give detail at very low values as well as at very high values, with a value of zero scaled to a value of zero. For example, a small value, of 0.1 mm/day has a scaled value of alog(0.1 + 1) * 70 = 6.7 which is rounded to an integer value of 7, such that it would be stored in the histogram bin number 7 (with arrays starting from bin number 0 in IDL). For the example of a very large rainfall amount of 1250 mm/day, the scaled value is alog(1250 + 1) * 70 = 499.2 which would be stored in the second highest bin in the histogram (i.e., 499), noting that rainfall of 1250 mm/day is set as the upper limit in the code (as mentioned above).

If a value exceeds one of the limits it is set equal to that limit, while noting that the limits can be changed if need be. For example, this upper limit for pr could be increased by changing the scaling equation above, or increasing the limit and histogram size (e.g., from 500 to 1000 bins in the histogram, while noting that would also increase computational requirements).

As mentioned in the previous section, if the variable being used is temperature there is an optional step (not essential to select) that can be used to help account for the long-term temperature trend. If this option is selected, the trend is calculated over all data, including the training period and data outside of the training period (e.g., for future projections data that the method will be applied to). This long-term climate trend is calculated from annual temperature data, using a 31-yr running-mean (i.e., box-car moving average) to produce a smoothed time series of the annual temperature data. The next step for that smoothed time series of annual temperature data (i.e., based on applying the 31-yr moving average) is to convert it to an anomaly with respect to the mean value for the first 31 years of the training period. This is done by first calculating the mean value for the first 31 years of the training period, then subtracting that value from each year in the smoothed time series of annual temperature data. The result is a

time series of annual values that represent the long-term temperature trend, expressed as an anomaly with respect to the average temperature in the first 31 years of the training period. This anomaly time series can be used by the QME method to help account for the long-term trend in temperature. If this option is selected to be used, the anomaly for a given year is removed from the temperature data for that year prior to applying the calibration, then added back in after applying the calibration. Further details on this are provided in Section 2.4, including an example to demonstrate its application in Section 3.1.

The choice of a 31-yr period for the moving average is somewhat arbitrary, but selected to be reasonably long to reduce short-term variability. It is noted that this removal of the long-term climatological temperature trend is only intended for a general shift of the histogram to help account for very large climate changes. Relatively short-term variations are not intended to be accounted for by this option, including noting that near-edge effects from smoothing with a running-mean are not considered a significant limitation (e.g., only a very minor influence on the amount by which a histogram is shifted overall). One option considered was a smoothing period equal to the training period, with the reference year in the middle of that period, but that might preclude application to shorter training periods as the variability could be large for the mean values in some cases.

If the temperature data are not continuous in time, such as just available for two periods (sometimes referred to as 'time slices'), other approaches can be used rather than using a long-term moving average for the temperature trend. One example of this (as presented in Section 3 of this report) is for a historical time slice from 1980 to 2019 and a future time slice from 2080 to 2099, in which case the mean temperature difference between those two periods can be used. For the example of that case, the mean temperature difference between those two periods can be removed from the future period of temperature data prior to applying the bias correction, then added in afterwards.

2.3. Training the bias correction factors

The component of the code called "run_training” includes running a quantile-quantile matching routine over the histograms, then applying additional attention to detail for the upper and lower tails of the histograms. This section of code can be run for individual months of the year, as well as for individual grid cell locations, such as shown in the Appendix for the example application to Australian climate data. There is an optional feature that users can select so that each month’s histogram also uses data from adjacent months to help increase the sample size. This can be applied once to provide a 3-month moving average. There is also a user option to apply this twice, which results in a 1-2-3-2-1 weighted moving average over 5 months centred on a given month. As noted in Dowdy (2020), different weather systems can preferentially occur around a particular time of year and models can vary in their ability to represent some types of weather phenomena, such that the application of this method for individual periods of

the year may help account for different biases specific to different weather phenomena to some degree.

Quantile-quantile matching is conducted to estimate the bias to be corrected at each quantile for data in the reference period, referred to in the code as 'training' the method. In addition to bias correction of data used in the training the method, these same quantile-based bias corrections can then also be applied to data not used for training the method, such as for data in a future period based on model projections. The quantile-quantile matching used in the QME code can be selected to be done based on different methods. One method is based on stepping through each of the 501 bins of the histograms to do the quantile matching. This can be done twice, based on first progressing upwards and then progressing downwards through the histogram bins, with the results then averaged from those two directions. It can also be done just in one direction, for a user option called "_quick" as shown in the code. These two versions usually produce near-identical results, such that the quicker version is recommended as the default for applications in general.

Another option can be selected for how to do the quantile-quantile matching that is slower (about twice as long to run this component of the code) but can sometimes help provide more detail, while noting it also produces near-identical results to the quicker options in most cases. This more detailed version progresses through the rank order of the sample values in the Input Data histogram, based on finding their corresponding value with the same rank in the Reference Data histogram. This uses a 2-D type of approach for binning the data (with the dimensions corresponding to the values of the Input Data and Reference Data, respectively), to help account for cases where the histogram bins may not always align well between the rank order of the Input Data and the Reference Data. This 2-D approach allows for a weighted average approach to determine the calibration factor, rather than relying more on either the Input Data bins or the Reference Data bins to determine that (as would be the case if only a 1-D approach was used). That method provides a 2-D array of 501x501 values (called "biass_corr_cell2d"), representing a first pass at the calibrated values for the Input Data histogram. That is then converted to a 1-D array (called "bias_corr_cell") based on a weighted average (by occurrence frequency of samples) of the relevant values for the 2-D array into the 1-D array.

Regardless of the method selected to do the quantile matching, the next step is to calculate the bias for the tails of the histogram. This method for handling the sample extremes is intended to help avoid unwanted effects such as those associated with the potential for overfitting or an excessive influence of very rare events that might sometimes occur in a sample period. Users can select the threshold where this special focus occurs, such as for values equal to or more extreme than the third highest value in the sample data (noting that the default recommendation is to use the third highest value for this threshold, as detailed further in Section 4). For those values, the bias correction amount is used as calculated for the neighbouring histogram bin (i.e., the histogram bin that is one place less extreme than the third highest value in the sample). That same bias correction amount for those sample extremes is also used for bias

correction of values outside the range used for training (i.e., for values that might occur in the model data in the future projected climate that might exceed those in the model data during training period). Consistent with how this is done for high values, this approach is also used for low values (e.g., for values equal to or lower than the third lowest value in the sample data). Another way to bias correct these extremes is based on the average bias of the sample extreme values (e.g., using a user-defined sample size) but that approach is more computationally expensive than the method used in this version of QME.

For both the high and low values, an additive bias adjustment is applied, with a user option also available to use a multiplicative adjustment. Multiplicative adjustment is sometimes recommended for rainfall, but in practice for this method based on a limited sample size an additive adjustment may be more practical in some cases (e.g., an additive approach will result in a more constrained magnitude of bias correction for extreme values, which users might find preferable in some cases such as with potential uncertainties in the sign of the bias adjustment).

An optional sample size limit can be selected to prevent the code being run on data with too few values, as well as checks for diversity of values in the histograms. A default value of 50 is used, but this can be set by the user depending on the application. The method also checks for various data quality issues, with notes provided in output log text if issues are found, while noting it is recommended that data checks are also done by the user prior to applying this method code. Some user knowledge of details in the Input Data and Reference Data is also recommended, such as for helping understand physical processes causing extreme values. For example, if an intense cyclone occurred nearby a location in the Input Data but not in the Reference Data then a difference in some extremes may be expected.

An additional user option is to set a maximum limit of increase for the bias correction, as well as a minimum value of the data for when this limit starts to be applied to it, with this all using the unscaled values of a variable rather than the scaled values. The default settings for rainfall in the code is for a maximum increase of 50% applied to values great than or equal to 10, such that model rainfall data of 20 mm/day could potentially be bias corrected up to a maximum value of 30 mm/day (i.e., a 50% limit for the increase). This option is intended to be useful to help prevent large increases in some cases that might be a result of a limited sample size rather than representative of what would be the case if a longer time period of data was available. Users can select these values depending on their preference for a given application, or this option can be not used at all if that is preferred (e.g., as could be the default suggested in general for other variables apart from rainfall).

Various post-processing steps are applied for the bias correction factors including checking limits for all variables (i.e., within the range 0 to 500 for the scaled values consistent with the histogram range). For the case of rainfall, the default option is that a value of 0 is unchanged by applying this bias correction, but if this option is not wanted it can be removed by commenting out the corresponding code in the procedure called train_qme. The code also includes applying a running mean (boxcar) smoothing over the

array of bias correction values (i.e., over the range from 0 to 500, corresponding to the histogram bins). This uses a 21-point moving average as the default, as well as noting that users can choose their own number of points (rather than the default of 21) as might be preferred for a particular application. In cases with relatively limited sample sizes, this smoothing may help provide more realistic variation in bias correction values over the array range from 0 to 500 (e.g., smoothing out spurious spikes and dips that might not be present if a larger sample size was available). However, this option is not essential to use if users prefer not to (e.g., for large sample sizes the shape may already be smooth and so this additional smoothing might not be essential).

2.4. Applying the bias correction factors

In the code component called "run_apply_cal", the model data are first read in over the full period of years as set in the code by the user. If the variable is tasmax or tasmin (and if this option is not switched off) the 31-yr running-mean average temperature anomaly is removed, following steps as described above in Section 2.2. This is done for all years more than 15 years after the start year of the training period (i.e., the first year for which a full 31-year period was used for the average anomaly). It is only intended to account for larger changes (e.g., in the case of large changes in future temperature as noted above), noting a range of other ways this could be done if wanting to account for smaller-scale variations. Additionally, mean values in different time periods can also be used to calculate the anomaly values if the data are not continuous, such as in the case of 'time slices' as described in Section 2.2 (with an example of this case for time slices presented in Section 3.1).

For the next step in this component of the code, the data are then scaled and limited, consistent with how this was done in earlier components of the code (i.e., for "run_input_read" and "run_reference_read"). The bias correction is then applied, based on using the corresponding value in the array that was produced from the "run_training" section of code (as described in the previous Section 2.3 above). The bias corrected data are then unscaled using the "unscale_data" procedure. If the 31-yr running-mean average temperature anomaly was removed previously, it is then added back in at this point in the method.

The code produces annual files for the bias corrected output data, such as shown in this example for the code provided here in the Appendix. These output data are netcdf files, with some general metadata included. More details can be added in the metadata as need be, either in the QME code where the current metadata details are produced or added in postprocessing these files further in subsequent steps that users might like to apply.

3. Results and examples of outputs

3.1. Application example for temperature

Some results are presented here based on application of the QME approach to climate model data. This includes examples for application to downscaled data from ERA5 reanalysis as well as from the CMIP6 model ACCESS-ESM1-5 model for the SSP3-7.0 emissions pathway, using the BARPA regional climate modelling approach (Su et al. 2022).

In addition to the code sections described above, there is also code including at the end of the program to make some figures for checking the outputs. This includes analysis for 12 locations as follows: 'Perth', 'Northwest', 'Darwin', 'Desert', 'Adelaide', 'Melbourne', 'Townsville', 'Hobart', 'Richmond', 'Bankstown', 'Lockyer Valley' and 'Archerfield'. Latitude and longitude details for those locations are as provided in the code. These locations were selected due to representing relatively diverse climate regions, as well as having high quality station data for some cases but not for others (such as the Desert location), so as to provide a diversity of cases for checking the output in this way.

Figure 1 presents an example of QME application using BARPA downscaling of ACCESS-ESM1-5 data from 1980 to 2019 as the training period, with the resultant bias correction also applied for a future time period of model data from 2080 to 2099 (for SSP3-7.0). This is based on daily maximum temperature (tasmax) data for January for the grid cell of Bankstown (in Sydney, NSW), using the AGCD observations-based data as the Reference Data. The figures include time series as well as histograms of the Reference Data (orange), uncorrected Input Data (green) and bias corrected Input Data (blue) data. The histograms are shown for the training period (i.e., 1980 to 2019) as well as the other application time period too (i.e., 2080 to 2099).

The default settings were used for the options that can be selected in the code (noting details also listed in Section 4 for default settings), including limiting the quantile-quantile matching up to the third most extreme values for the tails as well as using a 21-point boxcar moving average applied as smoothing over the range for the bias correction values. The option to account for the long-term temperature trend was also selected to be used for this example. However, as the future time slice is only 20 years (and is not connected with data continuously for other years back to the training period), a 31-year moving average is not used to account for the long-term temperature change. In this case, the long-term temperature change is calculated as the difference between the mean value of the future period and the mean value of the training period. The temperature change is removed from the data (in this case just from data for the future time period) prior to applying the bias correction, then added back into it afterwards.

This example highlights the benefits in using a quantile-based approach, including as the median values of the uncalibrated data (green) are closer to the observations-based data (orange) than is the case for the higher values where the bias

is considerably larger. This example is also relevant to the method option to account for large shifts in the histogram, such as due to climate change later this century, as it could help avoid applying excessively large bias correction to values closer to the median in cases like this example where future climate change has caused the histogram to shift up on average while still retaining its general shape.

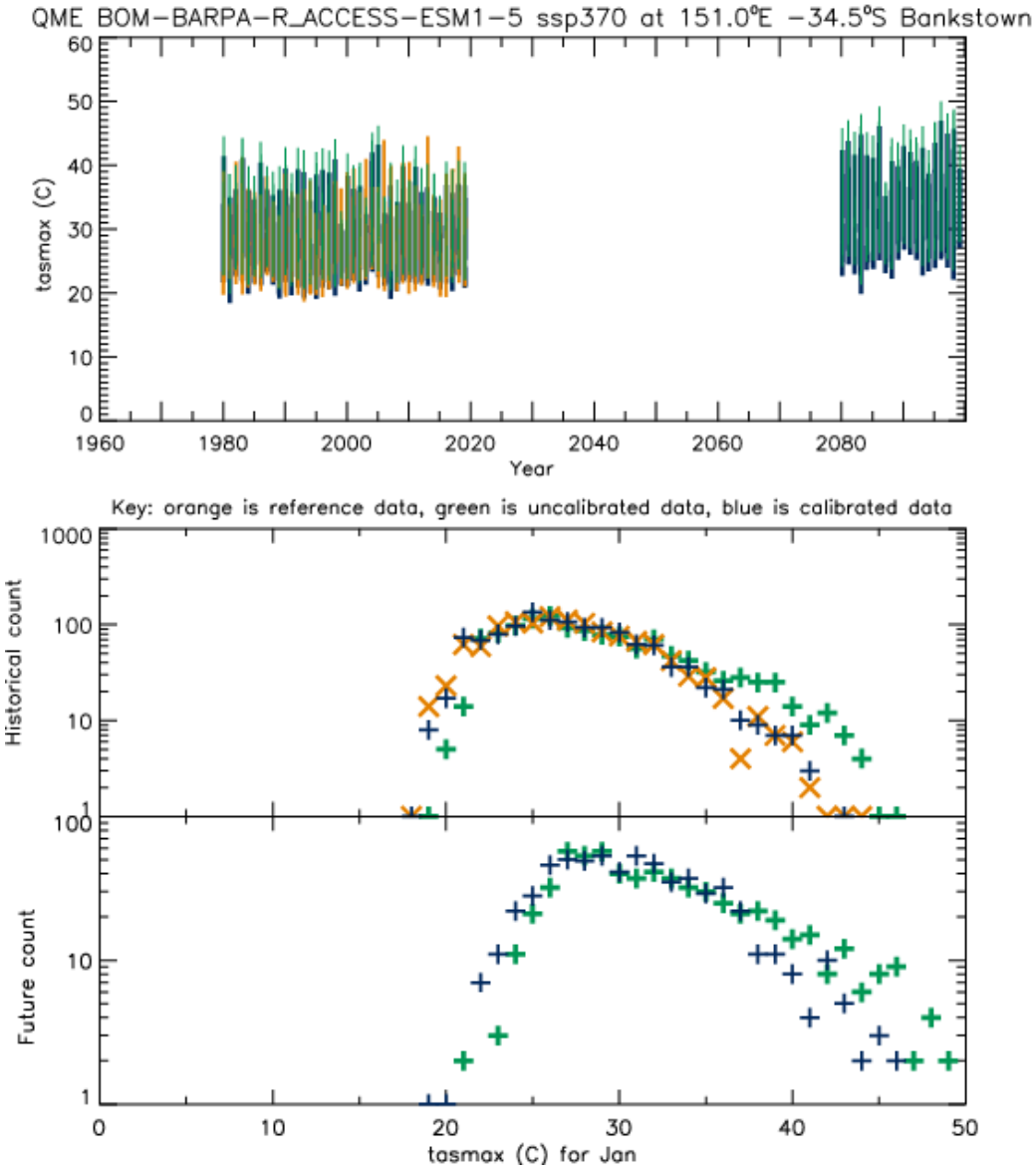

Figure 1: Example of the output figures for checking purposes, shown for tasmax (°C) at Bankstown (151.0 E and -34.5 S grid cell) for January. Data are from BARPA downscaling of the ACCESS-ESM1-5 model for 1980 to 2019 as the training period for calculating the bias correction values, including future emissions pathway of SSP3-7.0 for application of the bias correction values to the period 2080 to 2099. The figures include time series (upper panel) as well as histograms of the Reference Data (orange), Input Data (green) and bias corrected Input Data (blue). The histograms are shown for the training period (middle panel) as well as for the other application time period (lower panel).

3.2. Application example for precipitation

Figure 2 is similar to Figure 1, but for daily total precipitation (pr) at Hobart, using BARPA downscaling data from the ERA5 reanalysis (Hersbach et al. 2020). The period 1980 to 1999 was used in this case for training the bias correction, with application of the bias correction over the period from 1980 to 2019. Given the relatively short training period for this demonstration case (i.e., a 20-yr period), the option for including adjacent months was selected to help increase the sample size, such that the results shown here for July are also based on June and August as adjacent months for increased sample size during the training period (with further details on default settings as tabulated in Section 4).

This shows an example of the QME method applied to a different shape of distribution to that of temperature, with a skewed upper tail as is commonly the case for precipitation. As can be seen from these example results, the main effect of applying the bias correction in this case is to reduce the magnitude of extreme values in the model data (e.g., higher than about 30 mm/day) to be more similar to those of the observations-based data.

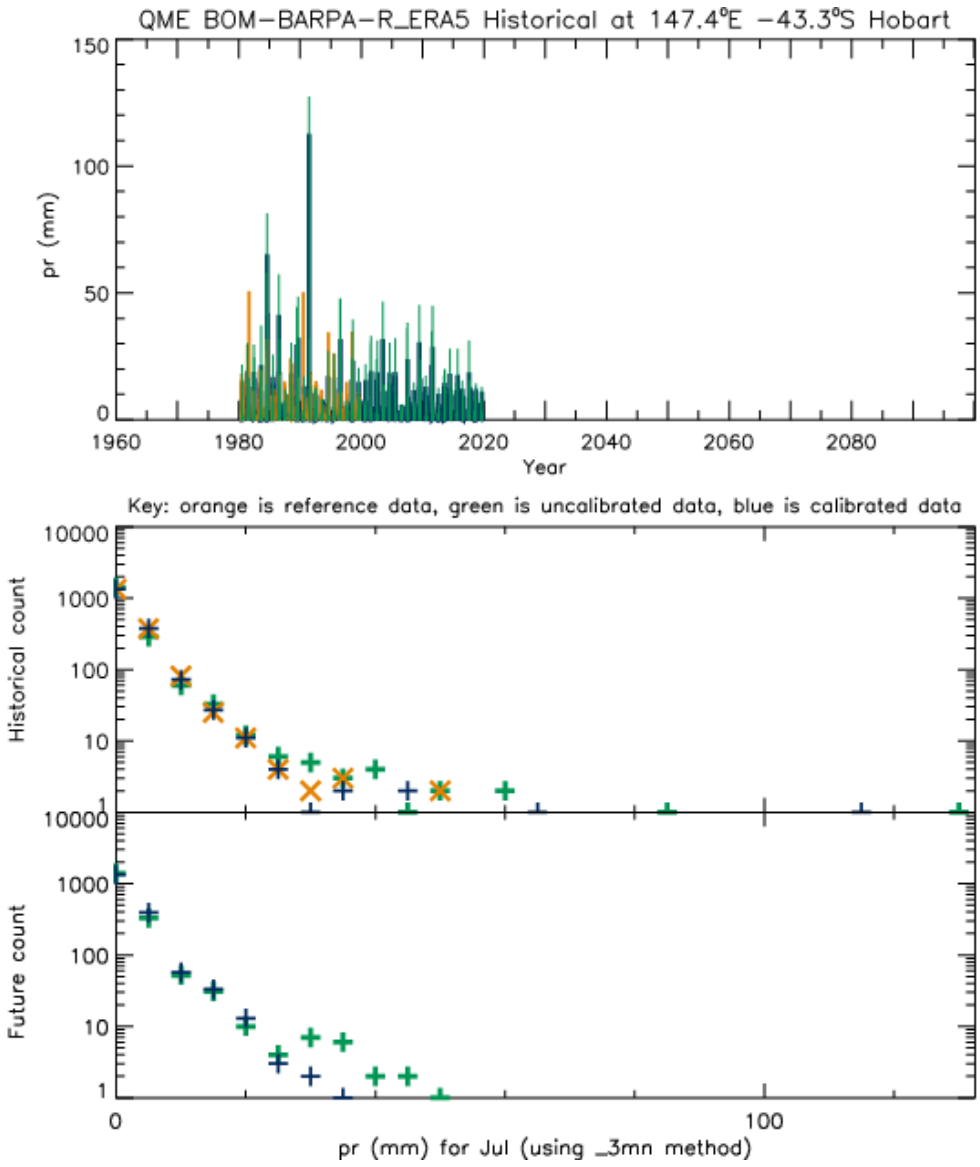

Figure 2: As for Figure 1, but for precipitation (mm/day) at Hobart (147.4 E and -43.3 S grid cell), based on BARPA downscaling from ERA5. The method is trained on the period 1980 to 1999, then applied through to 2019. This is shown for July, with adjacent months also used to help increase sample size.

In addition to figures similar to the two examples shown above, some other diagnostic figures are also made at various places in this code, just for general checking purposes. These include looking at the mean climatology when reading the obs data and similarly when reading the model data. The temperature trend over time is also plotted for the 12 locations listed above, within the code component called "run_account trend". Various text output is also produced during the different components of the code, for general checking purposes. This provides a form of log output when running the code.

4. Summary

This report has detailed the features of the QME method and its code components. Key features of the QME method are summarised in the following dot points:

- The method is univariate, preserves rank order and doesn't include spatial or temporal adjustment factors, with an intention that the bias corrected data are suitable for the application of subsequent processing and analysis methods (e.g., for parametric or multivariate approaches, including for considering rarer extremes such as based on methods using generalised extreme value distributions).
- It is designed to be relatively quick and simple to apply with low computational requirements, while including attention to detail around extremes in the data, given these characteristics can be beneficial when bias correcting very large data sets (e.g., ensembles of global or regional model data) for analysis of climate and weather hazards.
- Scaling and data limits are used to help provide more detail around extremes values in the sample data.
- The upper and lower tails of the histograms are handled with special attention to detail, with the quantile matching limited to a user-defined sample size (e.g., with a default value of 3 currently used in the code). This value of 3 is somewhat arbitrary but found to be suitable in previous applications to weather data with about 20–30 years or longer for training the method, while noting that for some variables such as rainfall longer time periods might be better suited in general if using a value of 3 for this option. Alternatively, a higher value such as in the range 5–10 could be used for data with relatively high variability such as rainfall based on less than about a 20–30 yr training period.
- For a given grid cell, the bias correction factors are calculated individually over the 501 values of the histogram, with a smoothing then applied to those values. The default smoothing is +/- 21 bins of the histogram, noting that this can be modified by users if need be.
- A maximum magnitude increase for the bias correction can be selected as an option, as well as a minimum value that can be selected for when this limit starts to be applied. For example, a default option suggested for rainfall in the code is for a maximum increase of 50% applied to values great than or equal to 10, in which case model simulated rainfall of 20 mm/day could be bias corrected up to a maximum of 30 mm/day, while noting that larger or smaller magnitude limits can also be selected for this option. This is useful to help prevent large changes in some cases that might be a result of a limited sample size rather than representative of what would be the case if a longer time period of data was available. It has been found to be useful for rainfall in some cases, but this

optional feature is likely not needed to be selected for other variables such as temperature.

- Different methods can be selected for how the quantile matching is done, including a quick version, as well as a somewhat slower more detailed version that produces only slightly different results to the quicker options in most cases (noting that the quickest version is set as the default).
- Using a large sample size for training is facilitated as much as possible, including an option to include data from adjacent months in the training period of data.
- An optional sample size limit can be selected to prevent the code being run on data with too few values, as well as checks for diversity of values in the histograms. A default value of 50 is used as default, but it can be set by users depending on the application.
- Various post-processing steps are applied for the bias correction factors including checking limits for all variables (i.e., within the range 0 to 500 for the scaled values consistent with the histogram range).
- An optional mask file can be used to select which grid cells it is applied to.
- Large shifts in the histograms between different portions of the data can be accounted for, such as caused by long-term trends in temperature. For example, the long-term trend in temperature can be removed from the data prior to applying the bias correction, to help account for large shifts in the distribution mean position in a changing climate, then added back in after the bias correction has been applied. This can be useful in cases such as weather features like fronts and highs superimposing variations on top of long-term temperature trends, including noting potential for large shifts in the temperature distribution in the future climate.

The code in the Appendix is shown for default settings. These default settings are also listed as follows here in Table 1. These settings should be suitable for most applications in general for weather and climate data, while noting that for specific applications a user may like to use settings other than these suggested default options.

The QME method is characterised by various features as summarised in this report, as well as intended to be adaptable depending on a particular application. Adapted versions of the QME method could be used for various applications such as different data types, regions and time steps (e.g., subhourly or monthly), with potential to add features if need be for a given purpose (e.g., parametric fits could be tried for potential future potential additions based on this code). The code provided in the Appendix uses the programming language IDL, while noting it can be translated readily into other languages. For example, a Python translation is currently underway for testing for use in potential future applications and next steps, building on from this IDL version of the QME code documented here.

Table 1: Some suggested default settings are tabulated here for various options that can be selected for the QME method, as is also shown in the code provided in the Appendix. These settings can be changed by users depending on what might be needed for a particular application or specific data set characteristics.

Variable name and default value	**Purpose**
xtr = 3	This number of values in the extremes of each tail (upper and lower tails individually) of the histograms are handled differently, rather than simply running the quantile matching over the full range of the histograms.
cal_smth = 21	This is the size of the boxcar moving average applied over the range of values in bias correction array (i.e., over the 501 bins in the bias correction array), noting this is applied after calculating the bias correction values.
mthd = '_quick'	This selects the method used for the quantile matching. Options include setting this to '_quick' as default, for a single iteration in one direction over the histogram range, or setting it to '' for talking the average of results from a double iteration (upwards and downwards directions) over the histogram range. A third option is setting it to '_detail' for a slower method with slightly more detail (but normally no noticeable difference in results to those other options.
mn_smth = '' as the default in general, or mn_smth = '_3mn' recommended for data with large variability such as rainfall and/or for relatively small sample sizes of training data (e.g., less than ~20 years)	This option can be set to '', or to '_3mn' if wanting to include adjacent months during the training period to increase sample size. A third option is also available to set to '_5mn' to use adjacent month from -2 to +2 months (using a 1, 2, 3, 2, 1 weighting centred on the give month).
account_trend = 1	Set to 1 if wanting to account for large shifts in a histogram when applying the calibration, or 0 if not wanting to use this, noting it is currently set up only as an option for temperature data to help account for long-term climate change trends.
figs = 1	Set to 1 to produce some diagnostic figures just for testing purposes, or set to 0 if not wanting to do this.
ssze_lim = 50	This is sample size limit, with the method not run if the sample size is less than this.

mltp = ''	Default is to set this to '' which results in an additive bias correction for the extreme values (including for values outside of training range). Alternatively, can set this to 'mltp' for a multiplicative method of bias correction for the extreme values. Although multiplicative can be considered for some distribution types (e.g., rainfall), uncertainties such as due to sample size limitations can mean that additive might still be preferable for those distribution types in some cases when applying this QME method (noting that this may be different to the case when applying other bias correction methods such as those that have parametric approaches). As such, the default here is set to '' but can be changed if users want to.
lmt = 1.5 (found to be useful for rainfall in some cases, with no notable effect in general when used or not for temperature)	Can set a maximum magnitude of increase equal to lmt, or if not wanting to use this feature then set the value of lmt = -1 which means that the changes due to bias correction are not limited in their range of increase. An example of using this limit as a default setting for rainfall could be lmt = 1.5 and lmt_thrsh = 10. (see next row of this table), which would limit the bias correction to a maximum potential change of 100% for rainfall higher than 10 mm.
lmt_thrsh = 10. (as noted above, this optional feature can be useful in some cases for rainfall, with no notable effect in general when used or not for temperature)	This is the minimum value for the data that will have a limit applied to it for the magnitude of bias correction (see above line).

References

Dowdy A, Ye H, Pepler A, Thatcher M, Osbrough S, Evans J, Di Virgilio G and McCarthy N (2019). Future changes in extreme weather and pyroconvection risk factors for Australian wildfires. Scientific Reports, 9(1), p.10073.

Dowdy A, Brown A, Pepler A, Thatcher M, Rafter T, Evans J, Ye H, Su C-H, Bell S and Stassen C (2021). Extreme temperature, wind and bushfire weather projections using a standardised method. Bureau of Meteorology, Melbourne, Australia, Bureau Research Report BRR55, ISBN: 978-1-925738-32-2.

Hersbach H, Bell B, Berrisford P, Hirahara S, Horányi A, Muñoz-Sabater J, Nicolas J, Peubey C, Radu R, Schepers D, Simmons A (2020). The ERA5 global reanalysis. Quarterly Journal of the Royal Meteorological Society, 146(730), 1999-2049.

Jones D, Wang W, Fawcett R (2009). High-quality spatial climate data-sets for Australia. Australian Meteorological and Oceanographic Journal, 58(4), p.233.

Maraun D, Shepherd TG, Widmann M, Zappa G, Walton D, Gutiérrez JM, Hagemann S, Richter I, Soares PM, Hall A and Mearns LO (2017). Towards process-informed bias correction of climate change simulations. Nature Climate Change, 7(11), 764-773.

Su C-H, Stassen C, Howard E, Ye H, Bell S, Pepler A, Dowdy A, Tucker S, Franklin C (2022). BARPA: New development of ACCESS-based regional climate modelling for Australian Climate Service. Bureau Research Report No. 069, Bureau of Meteorology, available from http://www.bom.gov.au/research/publications/researchreports/BRR-069.pdf.

Srikanthan S, Bende-Michl U, Wilson L, Sharples W, Vogel E, Peter J, Hope P, Loh S, Khan Z, Duong V, Roussis J, Dowdy A, Oke A, Matic V, Turner M, Kociuba G, Thomas S, Azarniv A, Donnelly C and Carrara E (2022). National Hydrological Projections - Design and Methodology. Bureau of Meteorology, Melbourne, Australia, Bureau Research Report BRR61, ISBN: 978-1-925738-37-7.

Teutschbein C and Seibert J (2012). Bias correction of regional climate model simulations for hydrological climate-change impact studies: Review and evaluation of different methods. Journal of Hydrology, 456, 12-29.

Vogel E, Johnson F, Marshall L, Bende-Michl U, Wilson L, Peter J, Wasko C, Srikanthan S, Sharples W, Dowdy A and Hope P (2023). An evaluation framework for downscaling and bias correction in climate change impact studies. Journal of Hydrology, p.129693.

Wasko C, Guo D, Ho M, Nathan R and Vogel E (2023). Diverging projections for flood and rainfall frequency curves. Journal of Hydrology, 620, p.129403.

Wilson L, Bende-Michl U, Sharples W, Vogel E, Peter J, Srikanthan S, Khan Z, Matic V, Oke A, Turner M and Duong V (2022). A national hydrological projections service for Australia. Climate Services, 28, p.100331.

Acknowledgements

The scientific testing and code design work was undertaken originally through the Australian Climate Change Science Program (ACCSP) for developing the QME bias correction method and concepts, with subsequent support through the Energy Sector for Climate Information (ESCI) project for broader applicability to a wide variety of data types and models. Recent collaborations using data with QME bias correction applied include with the National Hydrological Projections project (http://www.bom.gov.au/research/publications/researchreports/BRR-061.pdf), the National Environmental Science Program (https://www.dcceew.gov.au/science-research/nesp), the Australian Climate Service (https://www.acs.gov.au/) and the National Partnership for Climate Projections (https://www.dcceew.gov.au/climate-change/policy/climate-science/climate-science/climate-change-future), including with support through the University of Melbourne and the Bureau of Meteorology. The modelling groups that produced the ERA5, BARPA and ACCESS-ESM1-5 data are gratefully acknowledged, together with the processing of NPCP data by Damien Irving (CSIRO). Thank you to Sugata Narsey and Harvey Ye for providing useful feedback on this documentation. Thank you to Andew Gammon for current work underway in adapting the QME method from IDL into Python.

Appendix – QME code in IDL

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;    _   _   _
;   / \ / \ / \
;  ( Q | M | E )  Quantile Matching for Extremes
;   \_/ \_/ \_/
;
; Univariate bias correction method using quantile matching, with
; particular attention to sample extremes.
;
; Key features:
; - Optimisation of steps throughout the method for enhanced
; confidence when calibrating very high and very low values.
; - Data scaling can be used for enhanced detail on sample extremes
; in the histograms used for quantile-quantile matching.
; - The bias correction amount for very high values, such as equal
```

```
; to or exceeding the third highest value in a sample, can be made
; equal to the bias correction amount as calculated for the
; neighbouring histogram bin (i.e., the histogram bin that is one
; place less extreme than the third highest value in the sample).
; This is also done similarly for very low values (such as equal to
; or lower than the third lowest value in the sample). This option
; can be used to avoid unwanted influences from very small sample
; sizes and outliers in the histogram tails, or switched off if not
; wanted.
; - Users can select additive or multiplicative bias adjustment
; for extremes, as well as an option to limit the percentage
; magnitude change (such as might be useful in cases with
; relatively small sample sizes and/or large variability).
; - A relatively quick method is available for the quantile
; matching, as well as a slower approach with slightly more detail,
; but normally no noticeable difference in outputs for these
; different methods.
; - If using time series data, the QME method can be applied for
; individual months, with an option also available to include
; adjacent months for increased sample size when training the
; method.
; - User options also include accounting for large shifts in the
; input data histogram over different portions of the sample. For
; example, this 'trend-accounting' option can be useful for
; nonstationary time series, such as temperature data with
; short-term weather variations superimposed on a long-term trend.
; It is noted that using this option does not necessarily mean the
; trend is preserved by applying this bias correction.
; - No assumptions are required on distribution shape for extremes,
; noting that larger sample sizes are recommended for use where
; feasible including to help estimates for rarer events.
; - The bias-corrected output data preserve rank order after
; applying this method. The output data are intended to be
; suitable for use in subsequent applications, such as for input
; to parametric methods to estimate very rare extremes, or for
; input to other bias-correction methods based on multivariate or
; spatio-temporal adjustments, etc.).
;
; To apply this method, default settings are provided as a simple
; approach for general use, with other user options for settings
; also being available if need be (e.g., as described above). For
; user inputs, see places in code indicated with "***** User
; inputs here *****", such as for directory and file names for the
; Input Data and Reference Data, as well as creating directories
; and setting metadata options for the output files.
; As an example, the Input Data could be from model output at a
; single location with the Reference Data based on observations at
; that location. The bias correction is trained using the time
; period available for both the Input Data and Reference Data,
; with subsequent application over the full time period of the
; Input Data. The code is set up here to run individually for
; each grid cell of a 2-dimensional array, such as could be used
; to bias correct each individual grid cell locations through a
; region.
```

```
; A mask file can be used as an option, if wanting to apply the
; method only on a subset of data (with other values set to
; -999.99 in output files).
;
; This method is intended to be adaptable (such as for different
; data types, etc.) and updatable (e.g., potential for new code
; versions in the future). Early versions were developed around
; 2019 as research code in IDL, with refinements then including
; for broader applications to other data sources. Examples of
; past applications include
; -
https://www.sciencedirect.com/science/article/pii/S0022169423003451
; - https://www.nature.com/articles/s41598-019-46362-x
; - https://doi.org/10.1016/j.jhydrol.2023.129693
; - https://doi.org/10.1016/j.cliser.2022.100331
; - https://doi.org/10.1071/ES20001
; -
http://www.bom.gov.au/research/publications/researchreports/BRR-
055.pdf
; -
http://www.bom.gov.au/research/publications/researchreports/BRR-
061.pdf
; This version (September 2023) is set up for an example
; application to daily values of weather data gridded over
; Australia. Some lines of code here are prefaced by
; "***** Nonessential code - just for checking this Australian
; application *****" as they are not required for applying this
; method in general, but are just for testing some diagnostic
; figures and log file outputs for this example application.
;
; Contact: andrew.dowdy@unimelb.edu.au
; Further details in documentation and comments through the code
; below.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;
;
pro train_qme, dist_mdl, dist_obs, res, westerly_lon,
southerly_lat, var_name, file_str, xtr, mn_smth, ssze_lim, smth,
mltp, mthd, lmt, lmt_thrsh
;
; This procedure calculates the bias correction amount for each
; quantile.
; This is the main component of the QME method, noting that it can
; be run from an IDL command line if need be after compiling the
; code (e.g., ".comp qme.pro").
;
; Inputs:
; - "dist_mdl" is the array for the histogram distributions of the
; input data (e.g., from a model). It should have 3 dimensions for
; longitudes, latitudes and histogram bin numbers.
```

```
; - "dist_obs" is the array for the histogram distributions of the
; reference data (e.g., from observations). It should have 3
; dimensions for longitudes, latitudes and histogram bin numbers.
; - "res" is the grid spacing in degrees of the latitude and
; longitudes (which should be equal, or if not then users could
; modify the code accordingly).
; - "westerly_lon" is the most westerly longitude for these data
; (in degrees east).
; - "southerly_lat" is the most southerly latitude for these data
; (in degrees north).
; - "var_name" is a string for the type of variable being used
; (e.g., "pr" for precipitation data).
; - "file_string" is a string used for the bulk of the file path
; name (e.g., directory and file name) in addition to other details
; also included in the names when writing netcdf output files.
; - "xtr" is the number of samples used for defining each tail of
; the histogram.
; - "mn_smth" is for options to include adjacent months added to a
; given month, applied once if this equals "_3mn" or applied twice
; for "_5mn".
; - "ssze_lim" is the user-selected sample size limit for applying
; this method.
; - "smth" is the size of the boxcar moving average applied over
; the histogram bins for their bias correction values.
; - "mltp" is a string that can be set to "mltp" for multiplicative
; bias correction for tails and values outside of training range,
; or if set to "" (or anything else) then additive bias correction
; is used.
; - "mthd" is set to '' as default for efficient method, or
; '_detail' for slower method with slightly more detail (but
; normally no noticeable difference).
; - "lmt" is the maximum limit of the magnitude change (%) from
; applying the bias correction, which can be set equal to -1 for
; no limit.
; - "lmt_thrsh" is the threshold magnitude that if exceeded will
; have the maximum limit applied for the magnitude of change
; (i.e., a maximum magnitude change of lmt, in %).
;
; Outputs are saved as netcdf files including for the bias
; correction values and quality checking information.
;
; This procedure calls some other procedures: scale_data and
; unscale_data, writencdf3d and writencdf2d, as well as
; three_mnth_sum.
;
;
;
; Do some general preparations.
;
  ; Get some information on the data array dimensions.
  nlons = n_elements(dist_mdl[*, 0, 0]) ; Longitude.
  nlats = n_elements(dist_mdl[0, *, 0]) ; Latitude.
  reso = n_elements(dist_mdl[0, 0, *]) - 1 ; Resolution (i.e.,
number of bins in the histogram).
```

```
  biascorr = fltarr(nlons, nlats, reso + 1, 12) ; The bias
correction array for different locations (longitudes and
latitudes), for 12 months of the year.
  qchck = fltarr(nlons, nlats, 12) ; Quality check information for
different locations (longitudes and latitudes), for 12 months of
the year.
;
  ; Check both data arrays are consistent.
  if nlons ne n_elements(dist_obs[*, 0, 0]) then stop
  if nlats ne n_elements(dist_obs[0, *, 0]) then stop
  if reso ne n_elements(dist_obs[0, 0, *]) - 1 then stop
;
  ; Make some arrays used in subsequent sections of this procedure.
  scaled_reso = findgen(reso + 1) ; Make an array of integers
increasing from 0 to reso.
  unscaled_reso = findgen(reso + 1) ; For use in next line, to be
converted from sclaed_reso to unscaled values.
  unscale_data, unscaled_reso, var_name ; This array contains
unscaled values, converted from scaled_reso, for use below in
calibrating histogram tails.
;
  ; Include adjacent months if a user option was selected for this.
  if mn_smth eq '_3mn' then begin
    three_mnth_sum, dist_mdl ; Apply a 3-month moving sum to the
histogram data.
    three_mnth_sum, dist_obs ; Apply a 3-month moving sum to the
histogram data.
    xtr *= 3. ; To account for the summation of the sample.
  endif
  if mn_smth eq '_5mn' then begin
    three_mnth_sum, dist_mdl ; Apply a 3-month moving sum to the
histogram data.
    three_mnth_sum, dist_obs ; Apply a 3-month moving sum to the
histogram data.
    three_mnth_sum, dist_mdl ; Apply another 3-month moving sum to
the histogram data, resulting in a [1, 2, 3, 2, 1] type of moving
sum centred on a given month.
    three_mnth_sum, dist_obs ; Apply another 3-month moving sum to
the histogram data, resulting in a [1, 2, 3, 2, 1] type of moving
sum centred on a given month.
    xtr *= 9. ; To account for the summation of the sample.
   endif
;
;
;
; Calculate the bias correction values, determined for each
individual month, as follows.
  for mn = 0, 11 do begin ; Loop through the 12 months.
;
    dist_mdl_mn = dist_mdl[*, *, *, mn]
    dist_obs_mn = dist_obs[*, *, *, mn]
;
```

```
    print, mn, total(dist_obs_mn), max(dist_obs_mn),
min(dist_obs_mn), total(dist_mdl_mn), max(dist_mdl_mn),
min(dist_mdl_mn) ; General check (nonessential).
;
    for lon = 0, nlons - 1 do begin
      for lat = 0, nlats - 1 do begin
        dist_obs_cell = float(dist_obs_mn[lon, lat, *]) ; Select
Reference Data histogram for this grid cell.
        dist_mdl_cell = float(dist_mdl_mn[lon, lat, *]) ; Select
Input Data histogram for this grid cell.
;
        ; Do some data quality checks here, noting that users can
modify this or add other checks if need be.
        data_ok = 0 ; Used to check if data are suitable or not.
        dvrs = where(dist_mdl_cell gt 0, dvrs_count) ; Check for
diversity of data values.
        if dvrs_count le 1 then begin ; Users could also set a
different value here for a limit (e.g., le 5, etc.).
          data_ok -= 1 ; Not enough unique input data values.
        endif
        dvrs = where(dist_obs_cell gt 0, dvrs_count) ; Check for
diversity of data values.
        if dvrs_count le 1 then begin ; Users could also set a
different value here for a limit (e.g., le 5, etc.).
          data_ok -= 2 ; Not enough unique reference data values.
        endif
        if min( [total(dist_obs_cell), total(dist_mdl_cell)] ) lt
ssze_lim then begin
          data_ok -= 4 ; Sample size too small.
        endif
        if data_ok eq 0 then begin
;
          ; Do some checks and preparations prior to quantile
matching.
          ; Check there are the same number of counts (i.e., sample
size) in the model and obs distributions (histograms).
          tots_obs = total(dist_obs_cell)
          tots_mdl = total(dist_mdl_cell)
          if tots_obs ne tots_mdl then begin ; Scale up histogram
with smaller counts so that it has the same sample size as the
other histogram.
            if tots_mdl gt tots_obs then dist_obs_cell *=
(tots_mdl/tots_obs) $
             else dist_mdl_cell *= (tots_obs/tots_mdl)
            tots_obs = total(dist_obs_cell)
            tots_mdl = total(dist_mdl_cell)
            if round(tots_obs) ne round(tots_mdl) then begin
              print,'Error - inconsistent sample sizes: ',
round(tots_obs), round(tots_mdl)
              stop
            endif
          endif
          ; Find the array positions that have data counts gt 0.
          vals_obs = where(dist_obs_cell[*] gt 0., vals_obs_count)
```

```
          vals_mdl = where(dist_mdl_cell[*] gt 0., vals_mdl_count)
          if ((vals_obs_count le 0) or (vals_mdl_count le 0)) then
begin
            print,'Error - lack of positive values in histogram: ',
vals_obs_count, vals_mdl_count
            stop
          endif
;
          ; Do quantile matching over the histogram range.
          if mthd ne '_detail' then begin ; This is a relatively
quick method, with a more detailed option as follows below.
            biascorr_cell = fltarr(reso + 1) ; Array for the bias
correction values, for one grid cell.
            obs_pos = vals_obs[0] ; Position in array for
observations data.
            for mdl_pos = vals_mdl[0], vals_mdl[vals_mdl_count - 1]
do begin ; Position in array for model data.
              while ((total(dist_obs_cell[0: obs_pos]) lt
total(dist_mdl_cell[0: mdl_pos])) and (obs_pos lt reso)) do obs_pos
+= 1
              biascorr_cell[mdl_pos] = obs_pos
            endfor
            if mthd ne '_quick' then begin ; Can skip this for some
extra speed, with relatively little reduction in quality.
              obs_pos = vals_obs[vals_obs_count - 1] ; Position in
array for observations data.
              for mdl_pos = vals_mdl[vals_mdl_count - 1],
vals_mdl[0], -1 do begin ; Position in array for model data.
                while ((total(dist_obs_cell[obs_pos: reso]) lt
total(dist_mdl_cell[mdl_pos: reso])) and (obs_pos lt reso)) do
obs_pos -= 1
                biascorr_cell[mdl_pos] += obs_pos
              endfor
              ; Use the average value from the increasing and
decreasing method versions above.
              biascorr_cell /= 2.
            endif
;
          endif else begin ; More detailed and somewhat slower
version.
;
            w_obs = vals_obs[0] ; Index start position in the obs
histogram.
            w_mdl = vals_mdl[0] ;Index start position in the model
histogram.
            biascorr_cell2d = fltarr(reso + 1,reso + 1) ; 2-D array
for matching counts for model and obs data.
            biascorr_cell = fltarr(reso + 1) ; 1-D array for the
calibration values.
            if tots_mdl lt 1000. then begin ; For smallish sample
sizes can loop through all data.
              for i = 0., tots_mdl do begin ; Populate the 2-D
array of counts for matching model and obs data.
```

```
                while total(dist_mdl_cell[0: w_mdl]) lt i do w_mdl
+= 1
                while total(dist_obs_cell[0: w_obs]) lt i do w_obs
+= 1
                biascorr_cell2d[w_mdl, w_obs] += 1.
              endfor
            endif else begin ; For larger sample sizes, can use a
larger step size through the central portion of the sample.
              for i = 0., 489. do begin ; Populate the 2-D array of
counts for matching model and obs data.
                while total(dist_mdl_cell[0: w_mdl]) lt i do w_mdl
+= 1
                while total(dist_obs_cell[0: w_obs]) lt i do w_obs
+= 1
                biascorr_cell2d[w_mdl, w_obs] += 1.
              endfor
              stp_sz = 5. ; Can use a faster step size through
middle of the distribution (with potential to increase this for
very large samples).
              for i = 490., tots_mdl - 490., stp_sz do begin ;
Populate the 2-D array of counts for matching model and obs data.
                while total(dist_mdl_cell[0: w_mdl]) lt i do w_mdl
+= 1
                while total(dist_obs_cell[0: w_obs]) lt i do w_obs
+= 1
                biascorr_cell2d[w_mdl, w_obs] += 1.
              endfor
              for i = tots_mdl - 489., tots_mdl do begin ; Populate
the 2-D array of counts for matching model and obs data.
                while total(dist_mdl_cell[0: w_mdl]) lt i do w_mdl
+= 1
                while total(dist_obs_cell[0: w_obs]) lt i do w_obs
+= 1
                biascorr_cell2d[w_mdl, w_obs] += 1.
              endfor
            endelse
            ; Use the 2-D array from the above steps to derive a 1-
D bias correction array (based on weighted average of counts along
each row of the 2-D array).
            ; This step also includes linear interpolation over
gaps.
            gaps_arr = fltarr(reso + 1)
            for w_mdl = 0, reso do begin
              if (total(biascorr_cell2d[w_mdl, *]) gt 0.) then
begin
                biascorr_cell[w_mdl] = total(biascorr_cell2d[w_mdl,
*]*findgen(reso + 1)) / total(biascorr_cell2d[w_mdl, *]) ; Weighted
average.
              endif else gaps_arr[w_mdl] = 1
            endfor
            ; Linearly interpolate over gaps.
            for w_mdl = vals_mdl[0] + 1, vals_mdl[vals_mdl_count -
1] - 1 do begin
              if gaps_arr[w_mdl] eq 1 then begin
```

```
                w_st = w_mdl - 1
                while ((gaps_arr[w_mdl] eq 1) and (w_mdl lt reso))
do w_mdl += 1
                i = (biascorr_cell[w_mdl] - biascorr_cell[w_st]) /
float(w_mdl-w_st) ; Increment for linear interp.
                for ww = (w_st + 1), (w_mdl - 1) do
biascorr_cell[ww] = biascorr_cell[ww - 1] + i
              endif
            endfor
;
          endelse ; Slower but more detailed version.
;
          ; Now focus on tails.
          bias_tails = fltarr(reso + 1)
          ; Lower tail.
          xtr_low = vals_mdl[0] ; Find bias at the xtr lowest mdl
value.
          while(total(dist_mdl_cell[0: xtr_low]) lt xtr) do xtr_low
+= 1 ; Position in histogram of xtr lowest value.
          xtr_low += 1
          val = biascorr_cell[xtr_low]
          unscale_data, val, var_name
          if mltp eq 'mltp' then begin
            bias =  val/unscaled_reso[xtr_low]
            bias_tails[0: xtr_low] = unscaled_reso[0: xtr_low] *
bias
          endif else begin
            bias = val - unscaled_reso[xtr_low] ; Bias, as an
unscaled anomaly.
            bias_tails[0: xtr_low] = unscaled_reso[0: xtr_low] +
bias
          endelse
          ; Upper tail.
          xtr_high = vals_mdl[vals_mdl_count - 1] ; Find bias at
the xtr highest mdl value.
          while(total(dist_mdl_cell[xtr_high: *]) lt xtr) do
xtr_high -= 1 ; Position in histogram of xtr highest value.
          xtr_high -= 1
          val = biascorr_cell[xtr_high]
          unscale_data, val, var_name
          if mltp eq 'mltp' then begin
            bias =  val/unscaled_reso[xtr_high]
            bias_tails[xtr_high: *] = unscaled_reso[xtr_high: *] *
bias
          endif else begin
            bias = val - unscaled_reso[xtr_high] ; Bias, as an
unscaled anomaly.
            bias_tails[xtr_high: *] = unscaled_reso[xtr_high: *] +
bias
          endelse
          if var_name eq 'pr' then begin ; Ensure data for pr are
>= 0. Users can do similar here for other variables if need be.
            q = where(bias_tails le 0., qcount)
            if qcount gt 0 then bias_tails[q] = 0.
```

```
            bias_tails[0] = 0. ; For rainfall, a value of zero is
unchanged by the bias correction.
          endif
          scale_data, bias_tails, var_name
          biascorr_cell[0: xtr_low] = bias_tails[0: xtr_low]
          biascorr_cell[xtr_high: *] = bias_tails[xtr_high: *]
;
          ; Do some post-processing
          unscale_data, biascorr_cell, var_name
          if lmt ne -1 then begin ; Apply a limit of lmt (in %) for
the magnitude change.
            for x = 0, reso do begin
              if biascorr_cell[x] gt lmt_thrsh then
biascorr_cell[x] = min([biascorr_cell[x], unscaled_reso[x]*lmt])
; Only doing limit for positive values at the moment, but the
following could be used if wanting to limit negative numbers too.
;             if biascorr_cell[x] lt (-1.*lmt_thrsh) then
biascorr_cell[x] = max([biascorr_cell[x], unscaled_reso[x]*lmt])
            endfor
          endif
          if smth gt 1 then biascorr_cell = smooth(biascorr_cell -
unscaled_reso, smth, /edge_truncate) + unscaled_reso ; Apply a
smoothing to the bias correction values (using the anomaly data).
          if var_name eq 'pr' then begin ; Ensure data for pr are
>= 0. Users can do similar here for other variables if need be.
            q = where(biascorr_cell le 0., qcount)
            if qcount gt 0 then biascorr_cell[q] = 0. ; Default
option is for zero rainfall values to be unchanged by this bias
correction, but can comment out this line if not wanting this
option.
          endif
          scale_data, biascorr_cell, var_name ; Scale the array.
          biascorr[lon, lat, *, mn] = biascorr_cell - scaled_reso ;
Convert to anomaly for storing in the gridded array.
        endif else begin ; Data not suitable.
          qchck[lon, lat, mn] = data_ok ; Store the error code.
        endelse
      endfor ; lat.
    endfor ; lon.
  endfor ; mn.
;
  if var_name eq 'pr' then biascorr[*, *, 0, *] = 0. ; Ensure zero
rainfall is retained as zero.
;
  writenc4d, file_str + '_biascorr' + '.nc',
'QME_calibration_factors', 'for_' + var_name, $
   'biascorr', 'unitless', biascorr, res, westerly_lon,
southerly_lat
  writenc3d, file_str + '_qchck' + '.nc', 'QME_quality_check_data',
'for_' + var_name, $
   'qchck', 'unitless', qchck, res, westerly_lon, southerly_lat
end
;
;
```

```
;
;
;
pro scale_data, dat, var_name
;
; Scale the data array 'dat' for better representation of extremes
in the histograms for quantile matching.
; This scaling can be varied depending on the input variable
'var_name'.
; The scaled data need to range from 0 to reso for the histograms
used in quantile matching.
; The limit_data procedure below also ensures data stay within this
range from 0 to reso.
;
;***** User inputs here *****
; If wanting additional variables, users can add to the code below.
if var_name eq 'tasmax' then dat = (dat + 35.)*5.
if var_name eq 'tasmin' then dat = (dat + 55.)*5.
if var_name eq 'pr' then dat = alog(dat + 1.)*70.
if var_name eq 'wswd' then dat *= 10.
if var_name eq 'rsds' then dat *= 10.
if var_name eq 'rh' then dat *= 4.
;
end
;
;
;
;
;
pro unscale_data, dat, var_name
;
; Revert from scale_dat procedure, as above.
;
;***** User inputs here *****
; Users can add additional variables to the code below, with these
variables shown here as examples of how to set limits for
variables.
if var_name eq 'tasmax' then dat = dat/5. - 35.
if var_name eq 'tasmin' then dat = dat/5. - 55.
if var_name eq 'pr' then dat = exp(dat/70.) - 1.
if var_name eq 'wswd' then dat /= 10.
if var_name eq 'rsds' then dat /= 10.
if var_name eq 'rh' then dat /= 4.
;
end
;
;
;
;
;
pro limit_data, dat, var_name
;
; Apply limits to data in the array 'dat', with the limits
dependent on the variable name 'var_name'.
```

```
; This is used for the Input Data (e.g., model data) and Reference
Data (e.g., observations-based data) when preparing histograms for
use in quantile mapping.
;
;***** User inputs here *****
; These limits are somewhat arbitrary and users could add
additional variables or modify the code below as need be for a
particular application.
; These limits need to fit within the range of 0 to reso quantiles,
after applying scaling in the above procedure scale_data.
if var_name eq 'tasmax' then begin ; For daily maximum temperature.
  lim_upper = 60. ; Upper limit of 60 degrees C.
  lim_lower = -30. ; Lower limit of -30 degrees C.
endif
if var_name eq 'tasmin' then begin ; For daily minimum temperature.
  lim_upper = 40. ; Upper limit of 40 degrees C.
  lim_lower = -50. ; Lower limit of -50 degrees C.
endif
if var_name eq 'pr' then begin ; For daily precipitation.
  lim_upper = 1250. ; Upper limit of 1250 mm.
  lim_lower = 0. ; Lower limit of 0 mm.
endif
if var_name eq 'wswd' then begin ; For daily average wind speed.
  lim_upper = 45. ; Upper limit of 45 m/s.
  lim_lower = 0. ; Lower limit 0 m/s.
endif
if var_name eq 'rsds' then begin ; For daily average solar
radiation (downwelling at surface).
  lim_upper = 45. ; Upper limit 45 MJ/m/m.
  lim_lower = 0. ; Lower limit 0. MJ/m/m.
endif
if var_name eq 'rh' then begin ; For relative humidity.
  lim_upper = 110. ; Upper limit 110%.
  lim_lower = 0. ; Lower limit 0%.
endif
;
if ((size(dat, /n_dimensions) gt 3)) then stop ; Allow for 1-3
dimensions.
if size(dat, /n_dimensions) eq 1 then begin ; For 1-D arrays.
  x = where(dat ge lim_upper, x_count)
  if x_count gt 0 then dat[x] = lim_upper
  x = where(dat le lim_lower, x_count)
  if x_count gt 0 then dat[x] = lim_lower
endif
if size(dat, /n_dimensions) eq 2 then begin ; For 2-D arrays.
  for x = 0, n_elements(dat[*, 0]) - 1 do begin
    y = where(dat[x, *] ge lim_upper, y_count)
    if y_count gt 0 then dat[x, y] = lim_upper
    y = where(dat[x, *] le lim_lower, y_count)
    if y_count gt 0 then dat[x, y] = lim_lower
  endfor
endif
if size(dat, /n_dimensions) eq 3 then begin ; For 3-D arrays.
  for x = 0, n_elements(dat[*, 0, 0]) - 1 do begin
```

```
    for y = 0, n_elements(dat[0, *, 0]) - 1 do begin
      z = where(dat[x, y, *] ge lim_upper, z_count)
      if z_count gt 0 then dat[x, y, z] = lim_upper
      z = where(dat[x, y, *] le lim_lower, z_count)
      if z_count gt 0 then dat[x, y, z] = lim_lower
    endfor
  endfor
endif
;
end
;
;
;
;
;
pro NCDF_get_1field, file_name, var_name, var_data
;
; Read data for a variable 'var_name' from a netcdf file
'file_name', with data then stored in the array 'var_data'.
;
file_id = ncdf_open(file_name)
ncdf_varget, file_id, var_name, var_data
ncdf_close, file_id
;
end
;
;
;
;
;
pro read_input_data, mdl, mdl_name, empat, var_name, year_string,
dat_yr
;
; Procedure to read Input Data (such as for a given model, variable
and year).
;
;***** User inputs here *****
; Users can change the content in this section, such as depending
on the data source (e.g., model name) and variable (temperature,
rainfall, etc.), as well as include pre-processing and checking
steps if need be.
;
; Inputs:
; - mdl is a number used to select the data to be calibrated (e.g.,
select which model data to use from a list of several models in the
main body program below).
; - mdl_name is string text for a name corresponding to mdl.
; - empat is the emissions pathway name (such as RCP4.5, SSP3-7.0,
historical, etc.).
; - var_name is the name of the variable (such as tasmax, pr,
etc.).
; - year_string is string text for the year as used in file paths.
; - dat_yr is the array to store the data being read here, for
subsequent use in the main body of the code.
```

```
;
; Examples for reading Input Data are shown as follows below, based
on an application for Australian weather conditions.
;
year = float(year_string)
ndays_yr = julday(12, 31, year) - julday(1, 1, year) + 1
;
if mdl eq 0 then begin ; Data are from BARPA downscaling of ERA5.
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name + '/ECMWF-
ERA5/BOM-BARPA-R/raw/task-reference/' + var_name + '_NPCP-
20i_ECMWF-ERA5_evaluation_r1i1p1f1_BOM-BARPA-R_v1_day_' +
year_string + '01-' + year_string + '12.nc', var_name, dat_yr
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name + '/ECMWF-
ERA5/BOM-BARPA-R/raw/task-reference/' + var_name + '_NPCP-
20i_ECMWF-ERA5_evaluation_r1i1p1f1_BOM-BARPA-R_v1_day_' +
year_string + '01-' + year_string + '12.nc', 'lon', lon
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name + '/ECMWF-
ERA5/BOM-BARPA-R/raw/task-reference/' + var_name + '_NPCP-
20i_ECMWF-ERA5_evaluation_r1i1p1f1_BOM-BARPA-R_v1_day_' +
year_string + '01-' + year_string + '12.nc', 'lat', lat
endif
;
if mdl eq 1 then begin ; UQ-DES-CCAM ERA5 for NPCP testing.
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name + '/ECMWF-
ERA5/UQ-DES-CCAM-2105/raw/task-reference/' + var_name + '_NPCP-
20i_ECMWF-ERA5_evaluation_r1i1p1f1_UQ-DES-CCAM-2105_v1_day_' +
year_string + '0101-' + year_string + '1231.nc', var_name, dat_yr
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name + '/ECMWF-
ERA5/UQ-DES-CCAM-2105/raw/task-reference/' + var_name + '_NPCP-
20i_ECMWF-ERA5_evaluation_r1i1p1f1_UQ-DES-CCAM-2105_v1_day_' +
year_string + '0101-' + year_string + '1231.nc', 'lat', lat
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name + '/ECMWF-
ERA5/UQ-DES-CCAM-2105/raw/task-reference/' + var_name + '_NPCP-
20i_ECMWF-ERA5_evaluation_r1i1p1f1_UQ-DES-CCAM-2105_v1_day_' +
year_string + '0101-' + year_string + '1231.nc', 'lat', lat
endif
;
if mdl eq 2 then begin ; Data are from CSIRO-CCAM-2203 downscaling
of ERA5.
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name + '/ECMWF-
ERA5/CSIRO-CCAM-2203/raw/task-reference/' + var_name + '_NPCP-
20i_ECMWF-ERA5_evaluation_r1i1p1f1_CSIRO-CCAM-2203_v1_day_' +
year_string + '0101-' + year_string + '1231.nc', var_name, dat_yr
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name + '/ECMWF-
ERA5/CSIRO-CCAM-2203/raw/task-reference/' + var_name + '_NPCP-
20i_ECMWF-ERA5_evaluation_r1i1p1f1_CSIRO-CCAM-2203_v1_day_' +
year_string + '0101-' + year_string + '1231.nc', 'lon', lon
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name + '/ECMWF-
ERA5/CSIRO-CCAM-2203/raw/task-reference/' + var_name + '_NPCP-
20i_ECMWF-ERA5_evaluation_r1i1p1f1_CSIRO-CCAM-2203_v1_day_' +
year_string + '0101-' + year_string + '1231.nc', 'lat', lat
endif
;
```

```
if mdl eq 3 then begin ; Data are from BARPA downscaling of ACCESS-
ESM1-5.
  if year le 2014 then empat_yr = 'historical' else empat_yr =
empat
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name + '/CSIRO-
ACCESS-ESM1-5/BOM-BARPA-R/raw/task-reference/' + var_name + '_NPCP-
20i_CSIRO-ACCESS-ESM1-5_'+empat_yr+'_r6i1p1f1_BOM-BARPA-R_v1_day_'
+ year_string + '0101-' + year_string + '1231.nc', var_name, dat_yr
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name + '/CSIRO-
ACCESS-ESM1-5/BOM-BARPA-R/raw/task-reference/' + var_name + '_NPCP-
20i_CSIRO-ACCESS-ESM1-5_'+empat_yr+'_r6i1p1f1_BOM-BARPA-R_v1_day_'
+ year_string + '0101-' + year_string + '1231.nc', 'lon', lon
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name + '/CSIRO-
ACCESS-ESM1-5/BOM-BARPA-R/raw/task-reference/' + var_name + '_NPCP-
20i_CSIRO-ACCESS-ESM1-5_'+empat_yr+'_r6i1p1f1_BOM-BARPA-R_v1_day_'
+ year_string + '0101-' + year_string + '1231.nc', 'lat', lat
endif
;
dat_yr = float(dat_yr) ; Ensure data are float type.
;
end
;
;
;
;
;
pro read_reference_data, mdl, mdl_name, empat, var_name,
year_string, dat_yr
;
; Procedure to read Reference Data (such as based on observations
for a given variable and year).
;
;***** User inputs here *****
; Users can change the content in this section, such as depending
on the data source (e.g., model name) and variable (temperature,
rainfall, etc.), as well as include pre-processing and checking
steps if need be.
;
; Inputs:
; - mdl is a number used to select the data to be calibrated (e.g.,
select which model data to use from a list of several models in the
main body program below).
; - mdl_name is string text for a name corresponding to mdl.
; - empat is the emissions pathway name (such as RCP4.5, SSP3-7.0,
historical, etc.).
; - var_name is the name of the variable (such as tasmax, pr,
etc.).
; - year_string is string text for the year as used in file paths.
; - dat_yr is the array to store the data being read here, for
subsequent use in the main body of the code.
;
; Examples for reading Input Data are shown as follows below, based
on an application for Australian weather conditions.
;
```

```
if ((mdl ge 0) and (mdl le 3)) then begin ; Using 0.2-degree
aggregation of AGCD gridded analysis of observations.
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name +
'/observations/AGCD/raw/task-reference/' + var_name + '_NPCP-
20i_AGCD_v1-0-1_day_' + year_string + '0101-' + year_string +
'1231.nc', var_name, dat_yr
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name +
'/observations/AGCD/raw/task-reference/' + var_name + '_NPCP-
20i_AGCD_v1-0-1_day_' + year_string + '0101-' + year_string +
'1231.nc', 'lon', lon
  ncdf_get_1field, '/g/data/ia39/npcp/data/' + var_name +
'/observations/AGCD/raw/task-reference/' + var_name + '_NPCP-
20i_AGCD_v1-0-1_day_' + year_string + '0101-' + year_string +
'1231.nc', 'lat', lat
endif
;
dat_yr = float(dat_yr) ; Ensure data are float type.
;
end
;
;
;
;
;
pro three_mnth_sum, dat
;
; Procedure to make 3-month moving sum.
; Input: dat, with 3 or 4 dimensions (last of which is of size 12
for months, or 13 to include all year in some cases).
;
if size(dat, /n_dimensions) eq 3 then begin ; Array has 3
dimensions.
  if not( (n_elements(dat[0, 0, *]) eq 12) or (n_elements(dat[0, 0,
*] eq 13)) ) then stop ; Check this is 12, or 13 for all year also
included.
  dat2 = dat ; Temporary file.
  dat[*, *, 0] = (dat2[*, *, 11] + dat2[*, *, 0] + dat2[*, *, 1]) ;
Sum with adjacent months for January = 0 (i.e., add Dec = 11 and
Feb = 1).
  dat[*, *, 11] = (dat2[*, *, 10] + dat2[*, *, 11] + dat2[*, *, 0])
; Sum with adjacent months for December = 11 (i.e., add Nov = 10
and Jan =0).
  for mn = 1, 10 do dat[*, *, mn] = (dat2[*, *, mn - 1] + dat2[*,
*, mn] + dat2[*, *, mn + 1]) ; Sum with adjacent months for each of
the months Feb = 1 to Nov = 10.
endif else begin
  if size(dat, /n_dimensions) eq 4 then begin ; Array has 4
dimensions.
    if not( (n_elements(dat[0, 0, 0, *]) eq 12) or
(n_elements(dat[0, 0, 0, *] eq 13)) ) then stop ; Check this is 12,
or 13 for all year also included.
    dat2 = dat ; Temporary file.
```

```
    dat[*, *, *, 0] = (dat2[*, *, *, 11] + dat2[*, *, *, 0] +
dat2[*, *, *, 1]) ; Sum with adjacent months for January = 0 (i.e.,
add Dec = 11 and Feb = 1).
    dat[*, *, *, 11] = (dat2[*, *, *, 10] + dat2[*, *, *, 11] +
dat2[*, *, *, 0]) ; Sum with adjacent months for December = 11
(i.e., add Nov = 10 and Jan =0).
    for mn = 1, 10 do dat[*, *, *, mn] = (dat2[*, *, *, mn - 1] +
dat2[*, *, *, mn] + dat2[*, *, *, mn + 1]) ; Sum with adjacent
months for each of the months Feb =1 to Nov = 10.
  endif else begin
    print, 'Array dimensions not suitable: ', size(dat,
/n_dimensions)
    stop
  endelse
endelse
;
end
;
;
;
;
;
pro writenc2d, flnm, meta_txt1, meta_txt2, var_name, units, dat,
reso, westerly_lon, southerly_lat
;
; Write a 2-dimensional netcdf file, for the array 'dat'.
; The dimensions are longitude and latitude.
;
  nlons = n_elements(dat[*, 0])
  nlats = n_elements(dat[0, *])
;
  id = NCDF_CREATE(flnm, /CLOBBER)
  NCDF_control, id, /FILL
  yid = NCDF_DIMDEF(id, 'lat', nlats)
  xid = NCDF_DIMDEF(id, 'lon', nlons)
  NCDF_ATTPUT, id, /GLOBAL, meta_txt1, meta_txt2
  latid = NCDF_VARDEF(id, 'lat', [yid], /FLOAT)
  NCDF_ATTPUT, id, latid, 'units', 'degrees_north'
  lonid = NCDF_VARDEF(id, 'lon', [xid], /FLOAT)
  NCDF_ATTPUT, id, lonid, 'units', 'degrees_east'
  vid = NCDF_VARDEF(id, var_name, [xid, yid], /FLOAT)
  NCDF_ATTPUT, id, vid, 'units', units
;
  NCDF_CONTROL, id, /ENDEF ; Put file in data mode.
  NCDF_VARPUT, id, lonid, findgen(nlons)*reso + westerly_lon
  NCDF_VARPUT, id, latid, findgen(nlats)*reso + southerly_lat
  NCDF_VARPUT, id, vid, dat
  NCDF_close, id
  FILE_CHMOD, flnm, /U_WRITE, /G_WRITE, O_WRITE=0, /U_read,
/G_read, /O_read, /U_execute, G_execute=0, O_execute=0
end
;
;
;
```

```
;
;
pro writenc3d, flnm, meta_txt1, meta_txt2, var_name, units, dat,
reso, westerly_lon, southerly_lat
;
; Write a 3-dimensional netcdf file, for the array 'dat'.
; The first two dimensions are longitude and latitude, as well as
the third being unspecified for general purpose at various places
in this code.
;
  nlons = n_elements(dat[*, 0, 0])
  nlats = n_elements(dat[0, *, 0])
  nvals = n_elements(dat[0, 0, *])
;
  id = NCDF_CREATE(flnm, /CLOBBER)
  NCDF_control, id, /FILL
  tid = NCDF_DIMDEF(id, 'values', nvals)
  yid = NCDF_DIMDEF(id, 'lat', nlats)
  xid = NCDF_DIMDEF(id, 'lon', nlons)
  NCDF_ATTPUT, id, /GLOBAL, meta_txt1, meta_txt2
  timid = NCDF_VARDEF(id, 'values', [tid], /FLOAT)
  NCDF_ATTPUT, id, timid, 'not_applicable', 'values'
  latid = NCDF_VARDEF(id, 'lat', [yid], /FLOAT)
  NCDF_ATTPUT, id, latid, 'units', 'degrees_north'
  lonid = NCDF_VARDEF(id, 'lon', [xid], /FLOAT)
  NCDF_ATTPUT, id, lonid, 'units', 'degrees_east'
  vid = NCDF_VARDEF(id, var_name, [xid, yid, tid], /FLOAT)
  NCDF_ATTPUT, id, vid, 'units', units
;
  NCDF_CONTROL, id, /ENDEF ; Put file in data mode.
  NCDF_VARPUT, id, lonid, findgen(nlons)*reso + westerly_lon
  NCDF_VARPUT, id, latid, findgen(nlats)*reso + southerly_lat
  NCDF_VARPUT, id, timid, indgen(nvals)
  NCDF_VARPUT, id, vid, dat
  NCDF_close, id
  FILE_CHMOD, flnm, /U_WRITE, /G_WRITE, O_WRITE=0, /U_read,
/G_read, /O_read, /U_execute, G_execute=0, O_execute=0
end
;
;
;
;
;
pro writenc4d, flnm, meta_txt1, meta_txt2, var_name, units, dat,
reso, westerly_lon, southerly_lat
;
; Write a 4-dimensional netcdf file, for the array 'dat'.
; The first two dimensions are longitude and latitude, followed by
months for the third dimension, with the fourth being unspecified
for general purpose (e.g., histogram counts).
;
  nlons = n_elements(dat[*, 0, 0, 0])
  nlats = n_elements(dat[0, *, 0, 0])
  nmns = n_elements(dat[0, 0, *, 0])
```

```
  nvals = n_elements(dat[0, 0, 0, *])
;
  id = NCDF_CREATE(flnm, /CLOBBER)
  NCDF_control, id, /FILL
  tid = NCDF_DIMDEF(id, 'values', nvals)
  zid = NCDF_DIMDEF(id, 'month', nmns)
  yid = NCDF_DIMDEF(id, 'lat', nlats)
  xid = NCDF_DIMDEF(id, 'lon', nlons)
  NCDF_ATTPUT, id, /GLOBAL, meta_txt1, meta_txt2
  timid = NCDF_VARDEF(id, 'values', [tid], /FLOAT)
  NCDF_ATTPUT, id, timid, 'not_applicable', 'values'
  mnid = NCDF_VARDEF(id, 'month', [zid], /FLOAT)
  NCDF_ATTPUT, id, mnid, 'units', 'month_of_year'
  latid = NCDF_VARDEF(id, 'lat', [yid], /FLOAT)
  NCDF_ATTPUT, id, latid, 'units', 'degrees_north'
  lonid = NCDF_VARDEF(id, 'lon', [xid], /FLOAT)
  NCDF_ATTPUT, id, lonid, 'units', 'degrees_east'
  vid = NCDF_VARDEF(id, var_name, [xid, yid, zid, tid], /FLOAT)
  NCDF_ATTPUT, id, vid, 'units', units
;
  NCDF_CONTROL, id, /ENDEF ; Put file in data mode.
  NCDF_VARPUT, id, lonid, findgen(nlons)*reso + westerly_lon
  NCDF_VARPUT, id, latid, findgen(nlats)*reso + southerly_lat
  NCDF_VARPUT, id, mnid, findgen(12) + 1
  NCDF_VARPUT, id, timid, indgen(nvals)
  NCDF_VARPUT, id, vid, dat
  NCDF_close, id
  FILE_CHMOD, flnm, /U_WRITE, /G_WRITE, O_WRITE=0, /U_read,
/G_read, /O_read, /U_execute, G_execute=0, O_execute=0
end
;
;
;
;
;
pro ps_on, filename, EPS = eps, COLOR = color
;
; Nonessential code: This is used for making a figure if wanting to
have a quick check of some results.
;
; This procedure sets the plotting output to be postscript, with
keywords for color and eps file type options, as well as size
details below.
;
set_plot, 'ps', /interpolate
if keyword_set(color) then color = 1 else color = 0
if not keyword_set(eps) then eps = 0
;
xsize = 17.8
ysize = 20.7
xoff = 1.9
yoff = 1.7
;
device,color = color, filename = filename, landscape = landscape, $
```

```
       encapsulated = eps, xsize = xsize, ysize = ysize, $
       xoff = xoff, yoff = yoff, bits = 8
end
;
;
;
;
;
pro ps_off
;
; Nonessential code: Close off from the above procedure for
plotting postscript files.
;
device, /close
end
;
;
;
;
;
;
;
;
;
;;;;;;;;;;;;;;;;;;;;  main body of code
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;
!except = 2 ; For checking if any maths issues when testing code on
new data, but can comment this out for final application to run
slightly faster if need be.
;
reso = 500. ; Histogram resolution (i.e., number of bins, noting it
is actually 1 more due to array position zero also being used).
Users can change this if also changing the scale and limit
procedures above so that the data fit as intended within the range
limits of the histogram.
;
;
;
;
;
;***** User inputs here *****
; The following sections of code are the main areas where user
inputs can be made. This includes options for which component of
the code to run, settings selections, details for the data (spatial
info, etc.) and other specifications for output files.
;
;
;
; Users can select which of the QME code components to run (= 1) or
not (= 0). These components are shown as the larger code sections
below in this file.
```

```
run_input_read     = 1 ; Read the Input Data to create histograms
for the training portion of data.
run_reference_read = 1 ; Read the Reference Data to create
histograms for the training portion of data.
run_account_trend  = 1 ; Boxcar moving average used to help account
for large shifts in a histogram when applying the calibration
(e.g., useful in cases such as weather features like fronts and
highs superimposing variations on long-term temperature trends).
run_training       = 1 ; Calculate the calibration factors between
the histograms of the Input Data and Reference Data based on the
training portion of data.
run_apply_cal      = 1 ; Apply the calibration factors to all of
the Input Data, including for the training portion and other
portions of the data.
;
;
;
; Users can add details on models, emission pathways and time
periods in the following section of code.
mdl_arr = strarr(100) ; Array of unique names for each model to be
calibrated (can be added to as need be).
mdl_arr[0] = 'BOM-BARPA-R_ERA5'
mdl_arr[1] = 'UQ-CCAM_ERA5'
mdl_arr[2] = 'CSIRO-CCAM_ERA5'
mdl_arr[3] = 'BOM-BARPA-R_ACCESS-ESM1-5'
;
for mdl = 0, 0 do begin ; Select which model to use from the above
user-defined list.
  mdl_st = mdl_arr[mdl]
  empat = 'ssp370' ; If using climate projections can select which
emission pathway (such as 'rcp85', 'historical' for reanalysis,
'ssp3-7.0', etc.).
;
  ; Select which variables to use (users can modify or add other
variables etc.)
  var_name_mdl = ['tasmax', 'tasmin', 'pr', 'wswd', 'rsds', 'rh'] ;
Name of variables in model data files, including for output files.
  var_units = ['C', 'C', 'mm', 'm/s', 'MJ/m/m', '%'] ; Units for
the variables in output files.
  netcdf_details = ['Daily_maximum_temperature_2m',
'Daily_mainimum_temperature_2m', 'Daily_rainfall', 'Wind_speed',
'Daily_solar_radiation', 'Relative_humidity']
  for var = 0, 0 do begin
;
    ; Select which time periods to use for a particular
application.
    if mdl le 2 then begin
      st_year_train = 1980
      en_year_train = 1999
      st_year_apply = 1980 ; Also can include the training period
when applying.
      en_year_apply = 2019
    endif
    if mdl eq 3 then begin
```

```
      st_year_train = 1980
      en_year_train = 1999 ; or 2019, etc.
      st_year_apply = 1980 ; Also can include the training period
when applying.
      en_year_apply = 2019 ; or 2099, etc. ; Options for time
slices are available, such as to jump from 2019 to a time slice
from 2080 to 2099 as shown for this example model in the below
sections called 'run_account_trend' and 'run_apply_cal'.
    endif
;
    nyrs_trend = en_year_apply - st_year_train + 1. ; Number of
years used for option to account for long-term trend. This is to
check it is long enough period to apply a 31-yr moving average. It
is calculated from the start of the training period through to the
end of the applying period which for this Australian climate
example can include future projections.
;
;
;
;    Users can add spatial details for the data in the following
section of code (with examples shown here for Australian
applications).
    if mdl le 3 then begin ; For Australian climate example tests
using 0.2-degree AGCG aggregation.
      nlons = 211 ; Number of longitudes for NPCP application,
starting from 112.0 East.
      westerly_lon = 112. ; The most westerly longitude (in degrees
east) of the data, just for use in metadata.
      nlats = 171 ; Number of latitudes for NPCP application,
starting from -44.5 South.
      southerly_lat = -44. ; The most southerly latitude (in
degrees north) of the data, just for use in metadata.
      res = 0.2 ; Resolution of grid in degrees (spacing of grid
cells) as used for NPCP application.
    endif
;
;   Can select a tighter area for land, rather than the full
spatial grid, to save compute time for some parts of this code.
    st_lon = 0
    en_lon = nlons - 1
    st_lat = 0
    en_lat = nlats -1
;
;   List of locations used for checking results and figures (for
this Australian application example).
;   The following few lines are just for this example application
at 0.2-degree grid spacing.
    loc_name = ['Perth', 'Northwest', 'Darwin', 'Desert',
'Adelaide', 'Melbourne', 'Townsville', 'Hobart', 'Richmond',
'Bankstown', 'Lockyer Valley', 'Archerfield']
    loc_lon_arr = round( ([115.86, 122.24, 130.85, 138.00, 138.53,
144.98, 146.76, 147.32, 150.75, 151.03, 152.17, 152.99] -
westerly_lon)/res ) ; Order from low to high is needed here if
using quick testing options in later code sections.
```

```
    loc_lat_arr = round( ([-31.95, -17.96, -13.00, -30.00, -34.94,
-37.83, -19.26, -42.88, -33.60, -33.92, -27.63, -27.57] -
southerly_lat)/res )
    nlocs = n_elements(loc_lon_arr)
;
;   Land-sea mask (lsm) file used for some parts of the code (e.g.,
AWAP/AGCD data currently used are land only).
    if mdl ge 99 then begin ; Option to add mask file here, with
grid cells having a value of 1.0 if they are to be used and 0.0 if
not (with no other values used).
      stop ; This option is not currently used for this example
application, but users could read in a mask file here, for the
variable called lsm (see 2 lines below for example dimensions).
    endif else begin ; For msk = 0, don't mask out any locations
(i.e., set all locations = 1.0 in a mask file).
      lsm = fltarr(nlons, nlats)
      lsm[*, *] = 1. ; Use all locations.
    endelse
;
;
;
    ; Users can set some details for the output files here.
    tmpdir = '/g/data/eg3/ajd548/qme_dev/IDL_out/'; Output
directory for temporary files and during development if need be.
    if mdl le 10 then outdir = tmpdir
    date_made = 'Sep2023' ; For output metadata general text.
;
; Users can select some settings here, or simply use defaults as
suggested in comments here.
    xtr = 3 ; Sample size of extreme values used for average bias
to calibrate histogram tails and values outside of training range.
Default value of 3 used here, but can also use values that are
higher (e.g., values of the order 5 or 10 might be useful for
moderate sample sizes) or lower (with a minimum of 2). Users can
also select a value of 1 which means that this feature of the code
is not used.
    cal_smth = 21 ; Boxcar smoothing size applied after calculating
the calibration array. Default value of 21 is suitable for many
applications, or can use larger values (e.g., for data with large
variability and/or smallish sample size) or smaller values
(including with a value of 1 if not wanting this smoothing). Odd
(rather than even) numbers are recommended for this boxcar moving
average approach.
    mthd = '_quick' ; Set to '_quick' as default for an efficient
method option, or '' for a slightly slower but still relatively
efficient method. Can also set this to '_detail' for a slower
method with slightly more detail, but normally no noticeable
difference in outputs as compared to those other more efficient
options.
    if var eq 2 then mn_smth = '_3mn' else mn_smth = '' ; If using
time series data can set to '' to train method for each month
(suitable if many years of data are available) or '_3mn' to also
include adjacent months (suitable for moderate sample sizes). Can
also set this to '_5mn' resulting in a [1, 2, 3, 2, 1] moving sum
```

```
centred on a given month). The default example code here is using
'_3m' for precipitation (var = 2) with '' for other variables not
using this option.
    account_trend = 1 ; Set to 1 if wanting to account for large
shifts in a histogram when applying the calibration, or 0 if not
wanting to use this, noting it is currently set up only as a
default option for temperature data periods longer than 31 years
(with a 31-yr boxcar moving average is used to calculate the
trend). This can be useful in cases such as weather features like
fronts and highs superimposing variations on top of long-term
temperature trends. An option to apply this for different time
slices of data is also included in the code sections below of
'run_account_trend' and 'run_apply_trend'.
    ssze_lim = 50 ; Sample size limit for training the method,
raising a qualty check flagged in output log file for smaller
sample sizes than this limit.
    mltp = '' ; Set to 'mltp' for a multiplicative method of bias
correction in the histogram tails (and values outside of training
range), or set to '' for an additive method that is the default in
general for this method. Although multiplicative can be considered
for some distribution types (e.g., rainfall), uncertainties such as
due to sample size limitations can mean that additive might still
be preferable for those distribution types in some cases when
applying this QME method (noting that this may be different to the
case when applying other bias correction methods such as those that
have parametric approaches).
    if var eq 2 then lmt = 1.5 else lmt = -1. ; Can set a maximum
magnitude of increase equal to lmt here, or if not wanting to use
this feature then set the value of lmt = -1 which means that the
changes due to bias correction are not limited in their magnitude.
A default example of using this limit for rainfall could be lmt =
1.5, as well as lmt_thrsh = 10. set in the following line of code
here, which would limit the bias correction to a maximum potential
increase of 50% for rainfall values higher than 10 mm (e.g., a
value of 20 mm/day could be bias corrected up to an upper limit of
30 mm/day).
    lmt_thrsh = 10. ; Can set a threshold above which the limit,
lmt (from the above line), is applied for the maximum magnitude of
change. Values below this limit will not have the limit applied.
;
    figs = 1 ; Option to make some test figures to check the
outputs (nonessential: used for this Australia application
example).
;
; Details to list at the start of the output log information, with
more print output also in code sections below.
     print, 'QME running: ', var_name_mdl[var] + ' ', mdl_st + ' ',
empat, st_year_train, en_year_train, st_year_apply, en_year_apply,
cal_smth, mn_smth, mthd, account_trend, xtr, ssze_lim, mltp, lmt,
lmt_thrsh
     print, ' '
;
;
;
```

```
;
;
;
;
; The following sections are for different components that can be
switched on (= 1) or off (= 0) in the user options above.
; For simplicity, code indentation is restarted here.
;
;
;
;
;
; This section of code is to make histograms for the Input Data
(such as from multiple years of model data for the Australian
weather example used here).
if run_input_read eq 1 then begin
;
  dist_mn = intarr(nlons, nlats, reso + 1, 12) ; Histogram
(occurrence frequency distribution), for 12 months in this
application.
;
  for year = st_year_train, en_year_train do begin ; Loop through
the the years of data.
    print, 'Making histograms for Input Data: ', year
    year_string = strcompress(year, /remove_all)
    julday1_year = julday(1, 1, year) ; The Julian day for the
first day of this year.
    clim_check = fltarr(nlons, nlats) ; For a quick check of model
climatology (figure made below).
;
    read_input_data, mdl, mdl_st, empat, var_name_mdl[var],
year_string, dat_yr
    ndays_yr = n_elements(dat_yr[0, 0, *])
    if ((ndays_yr lt 0) or (ndays_yr gt 366)) then stop
    print, ' - unscaled max & min for first day of year, as well as
ndays_yr: ', max(dat_yr*lsm), min(dat_yr*lsm), ndays_yr
    limit_data, dat_yr, var_name_mdl[var] ; Apply some limits.
    scale_data, dat_yr, var_name_mdl[var] ; Scale the data for more
details on extremes.
    dat_yr = round(dat_yr) ; Convert to integer.
    if ((min(dat_yr) lt 0) or (max(dat_yr) gt reso)) then stop ; As
a check, but shouldn't occur.
;
    for doy = 0, ndays_yr - 1 do begin ; Loop through days to
populate the histogram.
      caldat, julday1_year + doy, mn, dy, yr
      month_string = strcompress(mn, /remove_all)
      if mn lt 10 then month_string = '0' + month_string
      mn -= 1 ; Array position for this month.
      for lon = st_lon, en_lon do begin
        for lat = st_lat, en_lat do begin
          if lsm[lon, lat] eq 1. then begin ; Only use selected
locations if using a mask file (e.g., land locations only).
            val = dat_yr[lon, lat, doy]
```

```
            dist_mn[lon, lat, val, mn] += 1
          endif
        endfor
      endfor
      clim_check += dat_yr[*, *, doy] ; Just used for a general
check of spatial features.
    endfor ; doy.
    if ((max(dist_mn) gt 32000) or (min(dist_mn) lt 0)) then stop ;
Check integer limits (> 32767 converts to negative values).
    print, ' - scaled max & min for first day of year, as well as
ndays_yr: ', max(dat_yr[*, *, 0]*lsm), min(dat_yr[*, *, 0]*lsm),
ndays_yr
  endfor ; yr.
;
  writenc4d, tmpdir + 'dist_mdl_' + var_name_mdl[var] + '_' +
mdl_st + '_' + empat + '.nc', $
   'Input_histogram', 'for_' + mdl_st + '_' + empat, 'dist_mdl',
'unitless', dist_mn, res, westerly_lon, southerly_lat
;
  ; Reduce memory for large arrays.
  dat_yr = 0
  dist_mn = 0
  clim_check = 0
;
  print, ' '
endif ; run_model_read.
;
;
;
;
;
;
;
; This section of code is to make histogram for the Reference Data
(such as observations-based data for the Australian example used
here).
if run_reference_read eq 1 then begin
;
  dist_mn = intarr(nlons, nlats, reso + 1, 12) ; Histogram
(occurrence frequency distribution), for 12 months in this
application.
;
; ***** Nonessential code - just for checking this Australian
application *****
  ndays_h = julday(12, 31, en_year_train) - julday(1, 1, 1960) + 1
; Just used in figures at end, for general checking.
  chk_obs_ts = fltarr(ndays_h, nlocs) ; Time series
  chk_obs_ts[*, *] = -999.99 ; Used to indicate no data
  chk_obs_dist = fltarr(1201, nlocs, 13) ; Histogram, with 13 for
12 months + 1 for all year.
;
  for year = st_year_train, en_year_train do begin ; Loop through
the the years of data
    print, 'Making histograms for Reference Data: ', year
```

```
    year_string = strcompress(year, /remove_all)
    julday1_year = julday(1, 1, year) ; The Julian day for the
first day of this year.
    clim_check = fltarr(nlons, nlats) ; For a quick check of model
climatology (figure made below).
;
    read_reference_data, mdl, mdl_st, empat, var_name_mdl[var],
year_string, dat_yr
    ndays_yr = n_elements(dat_yr[0, 0, *])
    if ((ndays_yr lt 0) or (ndays_yr gt 366)) then stop
    print, ' - unscaled max & min for first day of year, as well as
ndays_yr: ', max(dat_yr*lsm), min(dat_yr*lsm), ndays_yr
;
    limit_data, dat_yr, var_name_mdl[var] ; Apply some limits.
;
    ; ***** Nonessential code - just for checking this Australian
application *****
    for doy = 0, ndays_yr - 1 do begin ; Loop through days to
populate the histogram.
      caldat, julday1_year + doy, mn, dy, yr
      month_string = strcompress(mn, /remove_all)
      if mn lt 10 then month_string = '0' + month_string
      mn -= 1 ; Array position for this month.
      jdoy_index = julday(1, 1, yr) - julday(1, 1, 1960) + doy
      for loc = 0, nlocs - 1 do begin ; Save the unscaled data in
time series and distribution.
        val = dat_yr[loc_lon_arr[loc], loc_lat_arr[loc], doy]
        if var_name_mdl[var] eq 'tasmin' then val += 10. ; +10. for
tasmin, to focus on positive values in these arrays (can adjust for
colder cases if need be in future applications).
        val = 0. > val < 1200. ; Could use other limits if need be,
or other checking of ranges.
        chk_obs_ts[jdoy_index, loc] = val
        chk_obs_dist[val, loc, mn] += 1.
      endfor ; loc.
    endfor
;
    scale_data, dat_yr, var_name_mdl[var] ; Scale the data for more
details on extremes.
    dat_yr = round(dat_yr) ; Convert to integer.
    if ((min(dat_yr) lt 0) or (max(dat_yr) gt reso)) then stop ; As
a check, but shouldn't occur.
;
    for doy = 0, ndays_yr - 1 do begin ; Loop through days to
populate the histogram.
      caldat, julday1_year + doy, mn, dy, yr
      month_string = strcompress(mn, /remove_all)
      if mn lt 10 then month_string = '0' + month_string
      mn -= 1 ; Array position for this month.
      for lon = st_lon, en_lon do begin
        for lat = st_lat, en_lat do begin
          if lsm[lon, lat] eq 1. then begin ; This if for the
option to use a mask file to select locations (e.g., only using
land locations).
```

```
            val = dat_yr[lon, lat, doy]
            dist_mn[lon, lat, val, mn] += 1
          endif
        endfor
      endfor
      clim_check += dat_yr[*, *, doy] ; Just used for a general
check of spatial features.
    endfor ; doy.
    if ((max(dist_mn) gt 32000) or (min(dist_mn) lt 0)) then stop ;
Check integer limits (> 32767 converts to negative values).
    print, ' - scaled max & min for first day of year, as well as
ndays_yr: ', max(dat_yr[*, *, 0]*lsm), min(dat_yr[*, *, 0]*lsm),
ndays_yr
  endfor ; yr.
;
  writenc4d, tmpdir + 'dist_obs_' + var_name_mdl[var] + '_' +
mdl_st + '_' + empat + '.nc', $
   'Reference_histogram', 'for_' + mdl_st + '_' + empat,
'dist_obs', 'unitless', dist_mn, res, westerly_lon, southerly_lat
;
  ; ***** Nonessential code - just for checking this Australian
application *****
  for loc = 0, nlocs - 1 do begin
    for mn = 0, 11 do chk_obs_dist[*, loc, 12] += chk_obs_dist[*,
loc, mn] ; Make annual data from sum of all months.
  endfor
  save, filename = tmpdir + 'chk_obs_' + var_name_mdl[var] + '_' +
mdl_st + '_' + empat + '.sav', chk_obs_ts, chk_obs_dist
;
  ; Reduce memory for large arrays.
  dat_yr = 0
  dist_mn = 0
  clim_check = 0
  chk_obs_ts = 0
  chk_obs_dist = 0
;
  print, ' '
endif ; run_obs_read.
;
;
;
;
;
;
;
; This section of code is to calculate the long-term trend using a
running mean (i.e., boxcar moving average).
; This uses a 31-yr running mean (i.e., +/- 15 years) of annual
temperature (e.g., tasmax or tasmin) for these Australian climate
application examples.
if ( (run_account_trend eq 1) and ((var_name_mdl[var] eq 'tasmax')
or (var_name_mdl[var] eq 'tasmin')) ) then begin
;
```

```
  print, 'Calculating long-term trend (31-yr boxcar moving
average)'
  if nyrs_trend le 31 then begin
    print,'Not enough years for this trend otion to be used.'
  endif else begin ; There is enough data to calculate the trend
using this approach.
    mean_model = fltarr(nlons, nlats, nyrs_trend)
    for year = st_year_train, en_year_apply do begin
;
;     Time slice optional code (section 1 of 4): If using a time
slice option can adapt the following line, or comment out if not:
;      if ((mdl eq 3) and (year eq 2020)) then year = 2080 ; Can
use something like this for time slice (non-continuous) data, as
well as more code about 16 and 26 lines below here. Alternatively,
can switch off this option (i.e., run_account_trend = 0) if data
are not continuous.
;
      end_day = 365 - 1 ; No need to include leap year extra day or
monthly variants as only used for general long-term trend.
      year_indx = year - st_year_train ; Index for array.
      year_string = strcompress(year, /remove_all)
      julday1_year = julday(1, 1, year) ; The Julian day for the
first day of this year.
      read_input_data, mdl, mdl_st, empat, var_name_mdl[var],
year_string, dat_yr
      for doy = 0, end_day do mean_model[*, *, year_indx] +=
dat_yr[*, *, doy]
    endfor ; yr.
    mean_model /= 365. ; Convert to annual average values.
    temp_smth = 31. ; For smoothing below. Only intended for
estimating the general shift of the histogram to help account for
very large climate changes.
    ref_yr = 15. ; Anomaly array with respect to the start year +
15 years, noting this is only needed for an approximate training
period value.
    for lon = 0, nlons - 1 do begin
      for lat = 0, nlats - 1 do begin
        if lsm[lon, lat] eq 1. then begin
;
;         Time slice optional code (section 2 of 4): If using a
time slice option can adapt the following 4 lines, which just
calculates the mean difference between time slices, or comment out
these lines if not:
;          if (mdl eq 3) then begin ; If data are not continuous
(e.g., time slide data), can add/adapt code similar to the
following example lines.
;            mean_model[lon, lat, 100:119] = total(mean_model[lon,
lat, 100:119])/20. - total(mean_model[lon, lat, 0:39])/40. ;
Convert second time slice to anomaly using the mean of the first
time slice.
;            mean_model[lon, lat, 0:39] = 0.
;          endif else begin
;
```

```
            mean_model[lon, lat, *] = smooth(mean_model[lon, lat,
*], temp_smth, /edge_truncate) ; Convert to 31-yr moving average
values.
            mean_model[lon, lat, *] -= mean_model[lon, lat, ref_yr]
; Convert to anomaly with respect to ref_yr.
;
;         Time slice optional code (section 3 of 4): The following
line is for the case if using a time slice option can adapt the
following line, or comment out if not.
;          endelse
;
        endif
      endfor
    endfor
;
    writenc3d, tmpdir + 'mean_model_' + var_name_mdl[var] + '_' +
mdl_st + '_' + empat + '.nc', $
     'Annual_temperature_trend_from_' + strcompress(st_year_train,
/remove_all), 'for_' + mdl_st + '_' + empat, 'mean_model', $
     'C', mean_model, res, westerly_lon, southerly_lat
;
    ; ***** Nonessential code - just for checking this Australian
application *****
    ps_on, tmpdir + 'mean_mdl_' + var_name_mdl[var] + '_' + mdl_st
+ '_' + empat + '.eps', /eps, /color
    loadct,5
    !p.multi = [0, 1, 1]
    plot, mean_model[loc_lon_arr[0], loc_lat_arr[0], *], thick = 3,
yrange = [-2, 6], $
     position = [.2, .2, .9, .9], xtitle = 'Year since ' +
strcompress(st_year_train, /remove_all), $
     ytitle = 'Temperature anomaly (C)', title = 'Mean temperatures
at test locations'
    for loc = 1, nlocs - 1 do oplot, mean_model[loc_lon_arr[loc],
loc_lat_arr[loc], *], color = loc*10, thick = 2
    ps_off
    print, 'Check min and max for each year: '
    for i = 0, (en_year_apply - st_year_train) do print, i,
min(mean_model[*, *, i]*lsm), max(mean_model[*, *, i]*lsm)
    mean_model = 0 ; Reduce memory requirements for large arrays.
  endelse
;
  dat_yr = 0 ; Reduce memory requirements for large arrays.
  mean_model = 0
;
  print, ' '
endif ; run_temp_mean.
;
;
;
;
;
;
;
```

```
; This section of code is to calculate the calibration factors
(i.e., training).
if run_training eq 1 then begin
  print, 'Calculating calibration factors'
;
  ncdf_get_1field, tmpdir + 'dist_mdl_' + var_name_mdl[var] + '_' +
mdl_st + '_' + empat + '.nc', 'dist_mdl', dist_mdl
  ncdf_get_1field, tmpdir + 'dist_obs_' + var_name_mdl[var] + '_' +
mdl_st + '_' + empat + '.nc', 'dist_obs', dist_obs
;
  train_qme, dist_mdl, dist_obs, res, westerly_lon, southerly_lat,
var_name_mdl[var], tmpdir + var_name_mdl[var] + '_' + mdl_st + '_'
+ empat, xtr, mn_smth, ssze_lim, cal_smth, mltp, mthd, lmt,
lmt_thrsh
;
  dist_mdl = 0 ; Reduce memory needs.
  dist_obs = 0 ; Reduce memory needs.
;
  print, ' '
endif ; run_training.
;
;
;
;
;
;
;
;
; Apply the calibration to the model data.
if run_apply_cal eq 1 then begin
;
  print, 'Applying calibration'
  ncdf_get_1field, tmpdir + var_name_mdl[var] + '_' + mdl_st + '_'
+ empat + '_biascorr.nc', 'biascorr', biascorr
;  ncdf_get_1field, tmpdir + var_name_mdl[var] + '_' + mdl_st + '_'
+ empat + '_qchck.nc', 'qchck', qchck ; Users can check these
values if uncertain of data quality.
;
  if ( (account_trend eq 1) and (nyrs_trend gt 31) and
((var_name_mdl[var] eq 'tasmax') or (var_name_mdl[var] eq
'tasmin')) ) then $
   ncdf_get_1field, tmpdir + 'mean_model_' + var_name_mdl[var] +
'_' + mdl_st + '_' + empat + '.nc', 'mean_model', mean_model
;
  ; ***** Nonessential code - just for checking this Australian
application *****
  ndays_all = julday(12, 31, 2099) - julday(1, 1, 1960) + 1
  chk_mdl_ts = fltarr(ndays_all, nlocs) ; Time series for
uncalibrated model data.
  chk_mdl_ts[*, *] = -999.99 ; Infill value: not needed, but helps
make the time series look nicer and easier to interpret.
  chk_mdl_bc_ts = chk_mdl_ts ; Time series for calibrated model
data.
```

```
  chk_mdl_dist_h = fltarr(1201, nlocs, 13) ; Distributions for
uncalibrated training (historical) data.
  chk_mdl_dist_f = fltarr(1201, nlocs, 13) ; Distributions for
uncalibrated future data.
  chk_mdl_bc_dist_h = fltarr(1201, nlocs, 13) ; Distributions for
calibrated training (historical) data.
  chk_mdl_bc_dist_f = fltarr(1201, nlocs, 13) ; Distributions for
calibrated future data.
;
  for year = st_year_apply, en_year_apply do begin ; Loop through
years to apply calibration.
;
;  Time slice optional code (section 4 of 4): If using data for
different time slices users can adapt the following line, or
comment out if not using this option (including if data are for
continuous years rather than multiple time slice periods):
;   if ((mdl eq 3) and (year eq 2020)) then year = 2080 ; Can use
something like this for time slice (non-continuous) data.
;
    year_string = strcompress(year, /remove_all)
    julday1_year = julday(1, 1, year) ; The Julian day for the
first day of this year.
;
    read_input_data, mdl, mdl_st, empat, var_name_mdl[var],
year_string, dat_model_yr
    ndays_yr = n_elements(dat_model_yr[0, 0, *])
    ndays_yr_real = julday(1, 1, year + 1) - julday(1, 1, year)
    if ndays_yr ne ndays_yr_real then print, 'Days of year
difference: ' + mdl_st + ' ' + rcp, ndays_yr, ndays_yr_real
    dat_model_yr_bc = fltarr(nlons, nlats, ndays_yr_real) ; To
store the calibrated data for the year.
;
    for doy = 0, ndays_yr - 1 do begin ; Loop through each day of
the year.
      dat_model = dat_model_yr[*, *, doy] ; Select this day from
the year of data
;
      ; Do some time details.
      if ((ndays_yr_real eq 366) and (ndays_yr eq 365) and (doy ge
59)) then x_dy = 1 else x_dy = 0 ; Can be used to account for
missing leap year cases.
      caldat, julday1_year + doy + x_dy, mn, dy, yr
      day_string = strcompress(dy, /remove_all)
      if dy lt 10 then day_string = '0' + day_string
      month_string = strcompress(mn, /remove_all)
      if mn lt 10 then month_string = '0' + month_string
      mn -= 1 ; Array position for this month.
;
    ; ***** Nonessential code - just for checking this Australian
application *****
      jdoy_index = julday(1, 1, yr) - julday(1, 1, 1960) + doy
      for loc = 0, nlocs - 1 do begin
        if var_name_mdl[var] ne 'tasmin' then val =
dat_model[loc_lon_arr[loc], loc_lat_arr[loc]] else $
```

```
         val = dat_model[loc_lon_arr[loc], loc_lat_arr[loc]] + 10.
; +10. for tasmin range into negative values (can adjust for colder
cases).
        val = 0. > val < 1200. ; Could use a higher limit if need
be or other checking steps added.
        chk_mdl_ts[jdoy_index, loc] = val
        if ((year ge st_year_train) and (year le en_year_train))
then chk_mdl_dist_h[val, loc, mn] += 1.
        if year gt en_year_train then chk_mdl_dist_f[val, loc, mn]
+= 1.
      endfor
;
      if ( (account_trend eq 1) and (nyrs_trend gt 31) and
((var_name_mdl[var] eq 'tasmax') or (var_name_mdl[var] eq
'tasmin')) and (year gt (st_year_train + 15.)) ) $
       then dat_model -= mean_model[*, *, year - st_year_train] ;
Remove running mean as distribution shift is large.
      limit_data, dat_model, var_name_mdl[var] ; Apply some limits.
      scale_data, dat_model, var_name_mdl[var] ; Convert to scaled
versions.
      if ((min(dat_model*lsm) lt 0) or (max(dat_model*lsm) gt
reso)) then stop ; This shouldn't occur, but included as a double
check step.
;
      ; Apply the calibration.
      for lon = st_lon, en_lon do begin
        for lat = st_lat, en_lat do begin
          if lsm[lon, lat] eq 1. then begin
            ; Users might also perhaps use qchck information here,
such as if wanting to only apply bias correction in some cases
(e.g., potentially useful for applications with data quality issues
for some locations).
            dat_model[lon, lat] += biascorr[lon, lat,
round(dat_model[lon, lat]), mn]
        endif ; lsm.
        endfor ; lat.
      endfor ; lon.
;
      unscale_data, dat_model, var_name_mdl[var] ; Convert back
from previous scaling.
;
      ; If accounting for trend (e.g., for temperature variables)
then reapply the future trend.
      if ( (account_trend eq 1) and (nyrs_trend gt 31) and
((var_name_mdl[var] eq 'tasmax') or (var_name_mdl[var] eq
'tasmin')) and (year gt (st_year_train + 15.)) ) $
       then dat_model += mean_model[*, *, year - st_year_train]
;
      ; Check values aren't < 0 for variables this is relevant for.
      if ((var_name_mdl[var] eq 'pr') or (var_name_mdl[var] eq
'wswd') or (var_name_mdl[var] eq 'rsds')) then begin
        for lat = 0, nlats - 1 do begin
          q = where(dat_model[*, lat]*lsm[*, lat] lt 0., qcount)
          if qcount gt 0 then dat_model[q, lat] = 0.
```

```
        endfor
      endif
;
      dat_model_yr_bc[*, *, doy + x_dy] = dat_model ; Add this day
to annual array.
;
      ; ***** Nonessential code - just for checking this Australian
application *****
      jdoy_index = julday(1, 1, yr) - julday(1, 1, 1960) + doy
      for loc = 0, nlocs - 1 do begin
        if var_name_mdl[var] ne 'tasmin' then val =
dat_model[loc_lon_arr[loc], loc_lat_arr[loc]] else $
       val = dat_model[loc_lon_arr[loc], loc_lat_arr[loc]] + 10. ;
+10. for tasmin range into negative values (can adjust for colder
cases).
        val = 0. > val < 1200.
        chk_mdl_bc_ts[jdoy_index, loc] = val
        if ((year ge st_year_train) and (year le en_year_train))
then chk_mdl_bc_dist_h[val, loc, mn] += 1.
        if year gt en_year_train then chk_mdl_bc_dist_f[val, loc,
mn] += 1.
      endfor
;
    endfor ; doy.
;
    for lon = 0, nlons - 1 do begin ; Set regions not used to -
999.99.
      for lat = 0, nlats - 1 do begin
        if lsm[lon, lat] ne 1. then dat_model_yr_bc[lon, lat, *] =
-999.99
        ; Users might also perhaps use qchck information here, such
as if wanting to only apply bias correction in some cases (e.g.,
potentially useful for applications with data quality issues for
some locations).
      endfor
    endfor
;
    ; Write a netcdf file for the calibrated data.
    flnm = outdir + var_name_mdl[var] + '_' + mdl_st + '_' + empat
+ '_' + year_string + '.nc'
    id =  NCDF_CREATE(flnm, /CLOBBER)
    NCDF_control, id, /FILL
    tid = NCDF_DIMDEF(id, 'time', ndays_yr_real)
    yid = NCDF_DIMDEF(id, 'lat', nlats)
    xid = NCDF_DIMDEF(id, 'lon', nlons)
    NCDF_ATTPUT, id, /GLOBAL, netcdf_details[var],
'QME_calibrated_data_based_on_' + mdl_st + '_' + empat +
'_created_' + date_made + '_with_-999.99_for_no_application.'
    timid = NCDF_VARDEF(id, 'time', [tid], /FLOAT)
    NCDF_ATTPUT, id, timid, 'units', 'days since 1900-01-01 00:00'
    latid = NCDF_VARDEF(id, 'lat', [yid], /FLOAT)
    NCDF_ATTPUT, id, latid, 'units', 'degrees_north'
    lonid = NCDF_VARDEF(id, 'lon', [xid], /FLOAT)
    NCDF_ATTPUT, id, lonid, 'units', 'degrees_east'
```

```
    vid = NCDF_VARDEF(id, var_name_mdl[var], [xid, yid, tid],
/FLOAT)
    NCDF_ATTPUT, id, vid, 'units', var_units[var]
    NCDF_CONTROL, id, /ENDEF ; Put file in data mode.
    NCDF_VARPUT, id, lonid, findgen(nlons)*res + westerly_lon
    NCDF_VARPUT, id, latid, findgen(nlats)*res + southerly_lat
    NCDF_VARPUT, id, timid, julday1_year - julday(1, 1, 1900) +
indgen(ndays_yr_real)
    NCDF_VARPUT, id, vid, dat_model_yr_bc
    NCDF_close, id
    FILE_CHMOD, flnm, /U_WRITE, /G_WRITE, O_WRITE = 0, /U_read,
/G_read, /O_read, /U_execute, G_execute = 0, O_execute = 0
;
  endfor ; yr.
;
  ; ***** Nonessential code - just for checking this Australian
application *****
  ; Make all-year combination of the monthly values.
  for loc = 0, nlocs - 1 do begin
    for mn = 0, 11 do begin
      chk_mdl_dist_h[*, loc, 12] += chk_mdl_dist_h[*, loc, mn]
      chk_mdl_dist_f[*, loc, 12] += chk_mdl_dist_f[*, loc, mn]
      chk_mdl_bc_dist_h[*, loc, 12] += chk_mdl_bc_dist_h[*, loc,
mn]
      chk_mdl_bc_dist_f[*, loc, 12] += chk_mdl_bc_dist_f[*, loc,
mn]
    endfor
  endfor
  save, filename = tmpdir + 'chk_mdl_' + var_name_mdl[var] + '_' +
mdl_st + '_' + empat + '.sav', $
   chk_mdl_ts, chk_mdl_dist_h, chk_mdl_dist_f, chk_mdl_bc_ts,
chk_mdl_bc_dist_h, chk_mdl_bc_dist_f
;
  biascorr = 0 ; Reduce memory needs as these large array are no
longer needed.
  dat_model = 0
  dat_model_yr = 0
  dat_model_yr_bc = 0
  chk_mdl_ts = 0
  chk_mdl_dist_h = 0
  chk_mdl_dist_f = 0
  chk_mdl_bc_ts = 0
  chk_mdl_bc_dist_h = 0
  chk_mdl_bc_dist_f = 0
;
  print, ' '
endif ; run_apply_cal.
;
;
;
;
;
;
;
```

```
; ***** Nonessential code - just for checking this Australian
application *****
;
; This section of code is to make some quick diagnostic figures
(time series and shape of histograms) for point locations.
if figs eq 1 then begin
;
  print, 'Making some diagnostic test figures'
;
  mn_st = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
'Sep', 'Oct', 'Nov', 'Dec', 'AllYear']
;
  ; Make some useful time details.
  ndays_all = julday(12, 31 ,2099) - julday(1, 1, 1960) + 1.
  ndays_h = julday(12, 31, en_year_train) - julday(1, 1, 1960) + 1.
  jday_start_train = julday(1, 1, st_year_apply) - julday(1, 1,
1960)
  jday_start_apply = julday(1, 1, st_year_apply) - julday(1, 1,
1960)
  jday_end_train = julday(12, 31, en_year_train) - julday(1, 1,
1960)
  jday_end_apply = julday(12, 31, en_year_apply) - julday(1, 1,
1960)
;
  for mn = 12, 12 do begin ; 0: 11 for 12 months and 12 for all-
year.
;
    restore, tmpdir + 'chk_obs_' + var_name_mdl[var] + '_' + mdl_st
+ '_' + empat + '.sav'
    restore, tmpdir + 'chk_mdl_' + var_name_mdl[var] + '_' + mdl_st
+ '_' + empat + '.sav'
;
    ; If the method used was '_3mn', apply 3-month smoothing to the
output data for figures.
    if mn_smth eq '_3mn' then begin ; A 3-month moving average was
used for training, so include adjacent months in histograms. Note:
the calibrated histogram will not exactly match the obs histogram
including due to the calibration for adjacent months being
influenced by their adjacent months (e.g., the obs histogram for
March has Feb and Apr added to it, but the calibrated data for Feb
are also influenced by Jan and similarly for Apr influenced by
May). In contrast, the figures for mn_smth = '' should be more
similar between obs and calibrated histograms.
      three_mnth_sum, chk_obs_dist
      three_mnth_sum, chk_mdl_dist_h
      three_mnth_sum, chk_mdl_dist_f
      three_mnth_sum, chk_mdl_bc_dist_h
      three_mnth_sum, chk_mdl_bc_dist_f
    endif
    if mn_smth eq '_5mn' then begin ; A 3-month moving average was
used twice for training (resulting in a [1, 2, 3, 2, 1] moving
average centred on a given month), so include adjacent months in
histograms. Note: the calibrated histogram will not exactly match
the obs histogram including due to the calibration for adjacent
```

```
months being influenced by their adjacent months (e.g., the obs
histogram for March has Feb and Apr added to it, but the calibrated
data for Feb are also influenced by Jan and similarly for Apr
influenced by May). In contrast, the figures for mn_smth = ''
should be more similar between obs and calibrated histograms.
      three_mnth_sum, chk_obs_dist
      three_mnth_sum, chk_mdl_dist_h
      three_mnth_sum, chk_mdl_dist_f
      three_mnth_sum, chk_mdl_bc_dist_h
      three_mnth_sum, chk_mdl_bc_dist_f
      three_mnth_sum, chk_obs_dist
      three_mnth_sum, chk_mdl_dist_h
      three_mnth_sum, chk_mdl_dist_f
      three_mnth_sum, chk_mdl_bc_dist_h
      three_mnth_sum, chk_mdl_bc_dist_f
    endif
;
    if var_name_mdl[var] eq 'tasmin' then begin ; Subtract 10 C, as
this was added previously to use positive values in arrays (this
can be changed for colder applications, including changing the code
elsewhere in sections above where 10 C is added).
      chk_obs_ts -= 10.
      chk_mdl_ts -= 10.
      chk_mdl_bc_ts -= 10.
    endif
;
;
;
    ; Select data for month (or months) to show in time series
figure.
    if (mn ne 12) then begin ; Show all data in time series when
ssn = 12 (representing all months of the year).
      for dy_count = jday_start_apply, jday_end_apply do begin
        caldat, julday(1, 1, 1960) + dy_count, month, day, year
      month -= 1 ; To be consistent with IDL array positions that
start from 0.
        incld = 0 ; Used to check for exceptions.
;
      if mn_smth eq '' then begin ; If trained without 3-month
average, require exact match for month.
          if mn eq month then incld = 1
        endif
;
        if mn_smth eq '_3mn' then begin
        if (abs(mn - month)) le 1 then incld = 1 else begin ;
Account for adjacent month application.
            if ((mn eq 0) and (month eq 11)) then incld = 1 ;
January exception to include December.
          if ((mn eq 11) and (month eq 0)) then incld = 1 ;
December exception to include January.
          endelse
        endif
;
```

```
      if incld eq 0 then begin ; Not including this day in time
series figure.
          chk_mdl_bc_ts[dy_count, *] = -999.
          chk_mdl_ts[dy_count, *] = -999.
          if (dy_count le (ndays_h - 1.)) then chk_obs_ts[dy_count,
*] = -999.
        endif
      endfor
    endif
;
    for loc = 0, nlocs - 1 do begin
;
      ps_on, 'IDL_out/QME_' + var_name_mdl[var] + '_' +
mdl_arr[mdl] + '_' + empat + '_' + $
       strmid(strcompress((loc_lon_arr[loc]*res + westerly_lon),
/remove_all), 0, 5) + 'E_' + $
       strmid(strcompress((loc_lat_arr[loc]*res + southerly_lat),
/remove_all), 0, 5) + 'S_' + $
       mn_st[mn] + mn_smth + mthd + '.eps', /eps, /color
      !p.multi = [0, 1, 5]
      loadct, 4
;
;
;
;     First do the time series plots.
      if var_name_mdl[var] ne 'tasmin' then minv = .01 else minv =
-9.99 ; Minimum value to show in the figures (can be modified if
need be). tasmin has been shifted previously by adding 10 C, such
that this line here allows values to about -10 C.
      yrnge = [ minv, max([ chk_mdl_bc_ts[*, loc], chk_mdl_ts[*,
loc], chk_obs_ts[*, loc] ])*1.1 ]
      plot, chk_mdl_bc_ts[*, loc], yrange = yrnge, $
       thick = 5, position = [.14, .63, .98, .95], $
       title = 'QME ' + mdl_arr[mdl] + ' ' + empat + ' at ' +
strmid(strcompress((loc_lon_arr[loc]*res + 112.), /remove_all), 0,
5) + $
       '!eO!nE ' + strmid(strcompress((loc_lat_arr[loc]*res -
44.5), /remove_all), 0, 5) + '!eO!nS ' + loc_name[loc], $
       charsize = 1.9, xticks = 14, xtickv = findgen(15)*10*365,
xminor = 2, xticklen = 0.05, $
       xtickname = ['1960', ' ', '1980', ' ','2000', ' ', '2020', '
', '2040', ' ', '2060', ' ', '2080', ' ', ' '], $
       xtitle = 'Year!c!cKey: orange is reference data, green is
uncalibrated data, blue is calibrated data', $
       ytitle = var_name_mdl[var] + ' (' + var_units[var] + ') ',
charthick = 2, xthick = 2, ythick = 2, min_v = minv, /nodata
;
      oplot, chk_mdl_bc_ts[*, loc], color = 45, thick = 7, min_v =
minv
      oplot, chk_obs_ts[*, loc], color = 200, thick = 5, min_v =
minv
      oplot, chk_mdl_ts[*, loc], thick = 2.5, color = 85, min_v =
minv
;
```

```
;
;
;      Now do the histogram plots.
;
      xbs = 1. ; x-axis bin size for data aggregation. If adjusted
here for individual variables also adjust code below (see example
for pr).
;
      if var_name_mdl[var] eq 'tasmax' then begin
        xrng = [0, 55]
        xtcks = 5
        xtckv = [0, 10, 20, 30, 40, 50]
        xtcknm = ['0', '10', '20', '30', '40', '50']
        xtcknm_blank = [' ', ' ', ' ', ' ', ' ', ' ']
        sm = 1
        xmnr = 2
      endif
      if var_name_mdl[var] eq 'tasmin' then begin
        xrng = [0, 50]
        xtcks = 5
        xtckv = [0, 10, 20, 30, 40, 50]
        xtcknm = ['-10', '0', '10', '20', '30', '40'] ; Starting
from -10 as tasmax had 10 C added to it previously.
        xtcknm_blank = [' ', ' ', ' ', ' ', ' ', ' ']
        sm = 1
        xmnr = 2
      endif
      if var_name_mdl[var] eq 'pr' then begin
        xmax = 0 ; Find the maximum for the range in the figures
(there are better ways to code this than the following lines: to do
for Version 2).
        x = where(chk_obs_dist[*, loc, mn] gt 0.)
        if max(x) gt xmax then xmax = max(x)
        x=where(chk_mdl_dist_h[*, loc, mn] gt 0.)
        if max(x) gt xmax then xmax = max(x)
        x=where(chk_mdl_bc_dist_h[*, loc, mn] gt 0.)
        if max(x) gt xmax then xmax = max(x)
        x=where(chk_mdl_dist_f[*, loc, mn] gt 0.)
        if max(x) gt xmax then xmax = max(x)
        x=where(chk_mdl_bc_dist_f[*, loc, mn] gt 0.)
        if max(x) gt xmax then xmax = max(x)
        xmax *= 1.05
        xrng = [0, xmax]
        if xmax gt 100 then begin
          xtcks = round(xmax/100.)
          xtckv = findgen(xtcks + 1.) * 100.
          xtcknm = strcompress(round(findgen(xtcks + 1.))*100.,
/remove_all)
          xtcknm_blank = [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '
', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
          xmnr = 5
        endif else begin
          xtcks = 5
          xtckv = [0, 20, 40, 60, 80, 100]
```

```
          xtcknm = ['0', '20', '40', '60', '80', '100  ']
          xtcknm_blank = [' ', ' ', ' ', ' ', ' ', ' ']
          xmnr = 2
        endelse
        sm = 1
      if xmax gt 100. then xbs = 5. else xbs = 2. ; x-axis bin size
for data aggregation.
      if xmax gt 200. then xbs = 10. ; x-axis bin size for data
aggregation.
;
      endif
      if var_name_mdl[var] eq 'wswd' then begin
        xrng = [0, 25]
        xtcks = 5
        xtckv = [0, 5, 10, 15, 20, 25]
        xtcknm = ['0', '5', '10', '15', '20', '25  ']
        xtcknm_blank = [' ', ' ', ' ', ' ', ' ', ' ']
        sm = 1
        xmnr = 5
      endif
      if var_name_mdl[var] eq 'rsds' then begin
        xrng = [0, 40]
        xtcks = 4
        xtckv = [0, 10, 20, 30, 40]
        xtcknm = ['0', '10', '20', '30', '40  ']
        xtcknm_blank = [' ', ' ', ' ', ' ', ' ', ' ']
        sm = 1
        xmnr = 2
      endif
;
      plotarr_mdl = fltarr(1001) ; Uncalibrated model.
      plotarr_mdl_cal = fltarr(1001) ; Calibrated model.
      plotarr_obs = fltarr(1001) ; Observations.
      plotarr_mdl[0] = chk_mdl_dist_h[0, loc, mn]
      for i = 1, 1000./xbs - 1 do plotarr_mdl[i] =
total(chk_mdl_dist_h[1. + (i - 1.)*xbs: i*xbs, loc, mn])
      plotarr_mdl_cal[0] = chk_mdl_bc_dist_h[0, loc, mn]
      for i = 1, 1000./xbs - 1 do plotarr_mdl_cal[i] =
total(chk_mdl_bc_dist_h[1. + (i - 1.)*xbs: i*xbs, loc, mn])
      plotarr_obs[0] = chk_obs_dist[0, loc, mn]
      for i = 1, 1000./xbs - 1 do plotarr_obs[i] =
total(chk_obs_dist[1. + (i - 1.)*xbs: i*xbs, loc, mn])
      q = plotarr_mdl + plotarr_mdl_cal + plotarr_obs ; To find the
upper range of data for x-axis.
      w = where(q ge 1)
;
      plot, plotarr_mdl, position = [.14, .3, .98, .54], $
       yrange = [1, max([plotarr_mdl, plotarr_mdl_cal,
plotarr_obs])*1.05], /ylog, min_v = 1, $
       xtickv = xtckv/xbs, xticks = xtcks, xtickname =
xtcknm_blank, xrange = [0, max(w) + 0.5], xminor = xmnr, xticklen =
0.05, $
       thick = 4, psym = 2, symsize = 1.1, charsize = 2., $
```

```
       ytitle = '  Historical count', charthick = 2, xthick = 2,
ythick = 2, /nodata
      oplot, plotarr_mdl, color = 85, thick = 8, psym = 1, symsize
= 1.5
      oplot, plotarr_obs, color = 200, thick = 6, psym = 7, symsize
= 1.5
      oplot, plotarr_mdl_cal, thick = 5, psym = 1, symsize = 1.5,
color = 45
;
      plotarr_mdl = fltarr(1001) ; Uncalibrated model.
      plotarr_mdl_cal = fltarr(1001) ; Calibrated model.
      plotarr_mdl[0] = chk_mdl_dist_f[0, loc, mn]
      for i = 1, 1000./xbs - 1 do plotarr_mdl[i] =
total(chk_mdl_dist_f[1. + (i - 1.)*xbs: i*xbs, loc, mn])
      plotarr_mdl_cal[0] = chk_mdl_bc_dist_f[0, loc, mn]
      for i = 1, 1000./xbs - 1 do plotarr_mdl_cal[i] =
total(chk_mdl_bc_dist_f[1. + (i - 1.)*xbs: i*xbs, loc, mn])
      q = plotarr_mdl + plotarr_mdl_cal ; To find the upper range
of data for x-axis.
      if mn_smth ne '' then meth_str = ' (using ' + mn_smth + '
method)' else meth_str = '' ; A string to note on the figure which
method was used.
      plot, plotarr_mdl, position = [.14, .06, .98, .3], $
       yrange = [1, max([ plotarr_mdl, plotarr_mdl_cal] )*1.05],
/ylog, min_v = 1, $
       xtickv = xtckv/xbs, xticks = xtcks, xtickname = xtcknm,
xrange = [0, max(w) + 0.5], xminor= xmnr, xticklen = 0.05, $
       thick = 4, psym = 2, symsize = 1.1, charsize = 2., $
       ytitle = 'Future count   ', charthick = 2.5, xthick = 2,
ythick = 2, /nodata, $
       xtitle = var_name_mdl[var] + ' (' + var_units[var] + ') for
' + mn_st[mn] + meth_str
      oplot, plotarr_mdl, color = 85, thick = 8, psym = 1, symsize
= 1.5
      oplot, plotarr_mdl_cal, thick = 5, psym = 1, symsize = 1.5,
color = 45
      ps_off
;
;
;
    endfor ; loc.
;
  endfor ; mn.
;
  chk_obs_ts = 0
  chk_obs_dist = 0
  chk_mdl_ts = 0
  chk_mdl_dist_h = 0
  chk_mdl_dist_f = 0
  chk_mdl_bc_ts = 0
  chk_mdl_bc_dist_h = 0
  chk_mdl_bc_dist_f = 0
;
endif ; run_figs.
```

```
;
;
;
;
;
  endfor ; var [for conciseness, code indentation not used above in
this loop].
;
endfor ; mdl.
;
end
;
```