Advanced climwin

Liam D. Bailey and Martijn van de Pol

2020-05-26

In our introductory vignette we described the basic features available in the `climwin` package. Below, we will look in more detail at more advanced features available to users. We will cover:

- Fitting more complex baseline models (e.g. mixed effects models)
- Testing for climate thresholds.
- Dealing with data grouped over multiple years.
- Carrying out spatial replication.
- Fitting more complex curves to climate window data using `weightwin`.

More complex baseline models

In our previous vignette we fitted a simple linear model to be used as our baseline and studied the linear effect of temperature; however, in many cases users may want to design more complex baseline models. We will discuss a number of examples here.

Adding interactions with climate

Until now, we have assumed that the effects of climate do not interact with any other variables. This may not always be the case. As of v1.1.0, users can build a custom model structure with the ability to include interactions between climate and other variables. In our example using the Mass dataset, we might expect the effect of temperature on chick mass to vary with the average age of the mothers.

Firstly, it is necessary to create a variable called 'climate' in the biological dataset. The variable 'climate' doesn't need to contain any information, it is simply a placeholder to specify where climate data will be included in the model.

NOTE: It is necessary to call the new object 'climate' so that `climwin` can distinguish this variable from others. Other variable names will not work.

```
Mass$climate <- 1
```

Now that we have our 'climate' variable, we can add it into our baseline model. We can then specify how we want our climatic data to interact with other variables in the model.

`baseline = lm(Mass ~ climate*Age, data = Mass)`

In this example, climatic data from each tested climate window will be included in the model interacting with parent age.

```
Interaction <- slidingwin(xvar = list(Temp = MassClimate$Temp),
                          cdate = MassClimate$Date,
                          bdate = Mass$Date,
                          baseline = lm(Mass ~ climate*Age, data = Mass),
                          cinterval = "day",
                          range = c(150, 0),
                          type = "absolute", refday = c(20, 05),
                          stat = "mean",
                          func = "lin")
```

If we look at the best model from this analysis, we can see the interaction between temperature and age included in the model coefficients.

```
summary(Interaction[[1]]$BestModel)
```

```

Call:
lm(formula = yvar ~ climate + Age + climate:Age, data = modeldat)

Residuals:
    Min      1Q  Median      3Q     Max 
-5.6266 -1.5716  0.2878  1.6086  4.7510 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 170.2628     7.1678  23.754  < 2e-16 ***
climate      -5.5466     0.9200  -6.029 3.32e-07 ***
Age          -2.6046     2.6603  -0.979    0.333    
climate:Age   0.4024     0.3395   1.185    0.242    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.449 on 43 degrees of freedom
Multiple R-squared:  0.7778,	Adjusted R-squared:  0.7623 
F-statistic: 50.17 on 3 and 43 DF,  p-value: 4.267e-14
```

Mixed effects models

It will often be important to fit models with random effects terms (e.g. Individual ID). `climwin` includes functionality for mixed effects models using both `lme4` and `nlme`. Random effects terms can easily be added to the baseline model. We can use the 'Offspring' dataset in R as an example.

`baseline = lmer(Offspring ~ 1 + (1|BirdID), data = Offspring, REML = F)`

NOTE: Mixed effects models should be fitted using a maximum likelihood approach (i.e. REML = FALSE). If this is not specified in the baseline model, climwin* will make this adjustment internally.*

In a more specific scenario, we may want to include our climatic variable as a random effect in our model. As above, this can be done by creating a 'climate' variable in our dataset and including it in the baseline model.

`baseline = lmer(Mass ~ 1 + (1|climate), data = Mass, REML = F)`

Climate thresholds

Many studies, may be interested in testing climate windows using climatic thresholds. When testing climatic thresholds, we assume that a biological response is driven by days that surpass a particular climatic value. For example, seed germination may be influenced by the number of days over 30 degrees. Alternatively, temperature may only influence organism survival when it falls below freezing. A common example of such a climate threshold is growing degree days often used in the study of plant systems.

These can be achieved in `slidingwin` using the three parameters 'upper', 'lower' and, 'binary'. When a value is provided for the parameter 'upper', `slidingwin` will create a new climate dataset where all values equal to or below this threshold are set at 0. Similarly, when a value is set for 'lower', all values equal to or above this threshold will be set at 0. When values are provided for both ‘upper’ and ‘lower’, all values that fall *outside* these two threshold will be set at 0.

```
upper = 30
```

Date	Original temperature	Threshold temperature
01/06/2015	25.9	0
02/06/2015	24.0	0
03/06/2015	32.5	32.5
04/06/2014	28.1	0
05/06/2014	30.5	30.5
06/06/2014	30.0	0
07/06/2014	31.2	31.2
08/06/2014	27.0	0
…	…	…

```
upper = 30 lower = 25
```

Date	Original temperature	Threshold temperature
01/06/2015	25.9	25.9
02/06/2015	24.0	0
03/06/2015	32.5	0
04/06/2014	28.1	28.1
05/06/2014	30.5	0
06/06/2014	30.0	0
07/06/2014	31.2	0
08/06/2014	27.0	27.0
…	…	…

In some circumstances we may assume that all values past the climatic threshold will have an equally large impact on the biological response. In this case, we would set ‘binary’ to TRUE so that all non-zero values are set at 1. By default however, ‘binary’ will be set at FALSE, so that all values past the climatic threshold keep their original value.

```
upper = 30 binary = TRUE
```

Date	Original temperature	Threshold temperature
01/06/2015	25.9	0
02/06/2015	24.0	0
03/06/2015	32.5	1
04/06/2014	28.1	0
05/06/2014	30.5	1
06/06/2014	30.0	0
07/06/2014	31.2	1

Date	Original temperature	Threshold temperature
08/06/2014	27.0	0
…	…	…

A worked example

Below we will provide a worked example using the `Mass` and `MassClimate` dataset. In this example, lets imagine we are interested in testing the impact of the number of days above freezing on our mass response variable. To do this we would set both our 'upper' and 'binary' parameters.

`upper = 0` `binary = TRUE`

As we are interested in measuring the *number* of days above freezing, we set our stat parameter to 'sum'. Otherwise, we used model parameter values identical to our earlier vignette.

```
library(climwin)
```

```
MassWin <- slidingwin(xvar = list(Temp = MassClimate$Temp),
                      cdate = MassClimate$Date,
                      bdate = Mass$Date,
                      baseline = lm(Mass ~ 1, data = Mass),
                      cinterval = "day",
                      range = c(150, 0),
                      upper = 0, binary = TRUE,
                      type = "absolute", refday = c(20, 05),
                      stat = "sum",
                      func = "lin")
```

When we examine the best model data, we can see that our climate data is now count data.

```
head(MassWin[[1]]$BestModelData)
```

Yvar	climate
140	0
138	0
136	1
135	2
134	0
134	0

Biological measurements grouped across two years

Many long-term datasets that will be suitable for `climwin` are likely to be measured during Northern hemisphere spring/summer (e.g. breeding data). These biological records are measured during the middle of the year, meaning that biological records can be easily grouped by year. Yet in other circumstances biological measurements will fall across two years, particularly in Southern hemisphere species where spring/summer falls across the new year period [e.g., 1].

This can cause issues when fitting ‘absolute’ climate windows. As described in the introductory vignette, ‘absolute’ windows will use a set reference day for all biological records. Where biological measurements cross two years however, measurements from the same breeding season can be split up. In the table below, where a reference day of November 1st is used, all those measurements taken at the start of the breeding season are given a date of November 1st 2014 while all values following the new year are set at November 1st 2015. This is obviously unrealistic, as biological measurements in January 2015 cannot be impacted by climatic conditions that will occur 11 months in the future!

Date	Reference Date
05/11/2014	01/11/2014
10/11/2014	01/11/2014
01/12/2014	01/11/2014
12/12/2014	01/11/2014
01/01/2015	01/11/2015
07/01/2015	01/11/2015

As a solution, `climwin` includes a ‘cohort’ parameter that allows users to specify which biological measurements should be grouped together (e.g. when they are from same breeding season). Each biological record should be given a cohort level (see below), which is taken into account when setting the reference day for climate window analyses.

Date	Cohort	Reference Date
05/11/2014	2014	01/11/2014
10/11/2014	2014	01/11/2014
01/12/2014	2014	01/11/2014
12/12/2014	2014	01/11/2014
01/01/2015	2014	01/11/2014
07/01/2015	2014	01/11/2014

Some example code showing how the ‘cohort’ argument can be used is provided below. In this example we use data from a southern hemisphere breeding bird.

```r
SizeWin <- slidingwin(xvar = list(Temp = SizeClimate$Temperature),
                      cdate = SizeClimate$Date,
                      bdate = Size$Date,
                      baseline = lm(Size ~ 1, data = Size),
                      cohort = Size$Cohort,
                      cinterval = "day",
                      range = c(150, 0),
                      type = "absolute", refday = c(01, 10),
                      stat = "mean",
                      func = "lin")
```

Spatial replication

To detect climate signals using `climwin` can often require large amounts of data, particularly if the relationship between climate and biological response is weak [2]. To obtain the required data through temporal replication requires the collection of data over multiple years, often decades; however, spatial replication may also allow users to expand their sample size over a shorter period by collecting data from multiple sites/populations.

Using spatial replication assumes that the relationship between the biological response and climatic predictor is consistent across the different measured populations. Where this assumption is valid, spatial replication can help expand the amount of data available for `climwin` analyses.

A worked example

Spatially replicated data can be analysed using the `slidingwin` function with the addition of the 'spatial' parameter. As with regular `slidingwin` analysis, analysis with spatial replication requires a separate biological and climate dataset. However, these datasets should now contain an additional variable which specifies the site at which biological and climate data were collected. Below, we have called this parameter 'SiteID'.

Date	Mass (g)	SiteID
04/06/2015	120	A
05/06/2015	123	A
07/06/2015	110	B
07/06/2015	140	A
06/06/2015	138	B
…	…	…

Date	Temperature	SiteID
01/06/2015	15	A
02/06/2015	16	A
03/06/2015	12	A
04/06/2015	18	A
05/06/2015	20	A
06/06/2015	23	A
07/06/2015	21	A
01/06/2015	10	B
02/06/2015	12	B
03/06/2015	9	B
04/06/2015	5	B
05/06/2015	13	B
06/06/2015	10	B
07/06/2015	11	B
…	…	…

NOTE: The climate dataset for spatially replicated `climwin` analysis will often include duplicate dates. In a regular `climwin` analysis this will lead to errors.

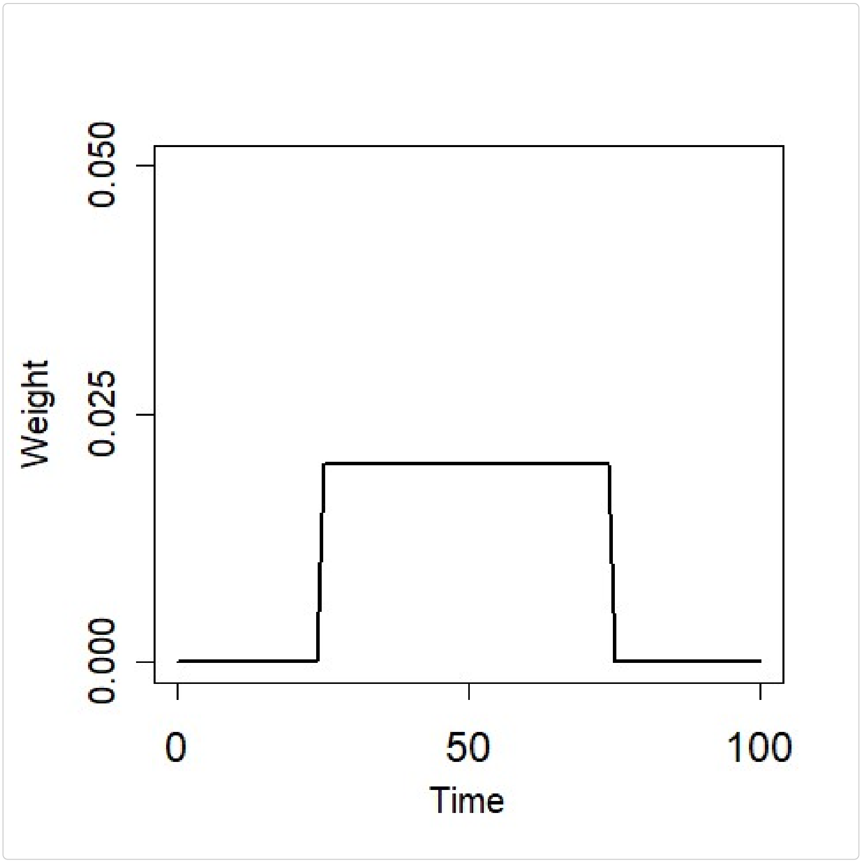
With these new datasets, we can carry out a `slidingwin` analysis with the addition of a 'spatial' parameter.

```
MassWin <- slidingwin(xvar = list(Temp = Climate$Temp),
                        cdate = Climate$Date,
                        bdate = Biol$Date,
                        baseline = lm(Mass ~ 1, data = Biol),
                        cinterval = "day",
                        range = c(150, 0),
                        type = "absolute", refday = c(20, 05),
                        stat = "mean",
                        func = "lin", spatial = list(Biol$SiteID, Climate$SiteID))
```

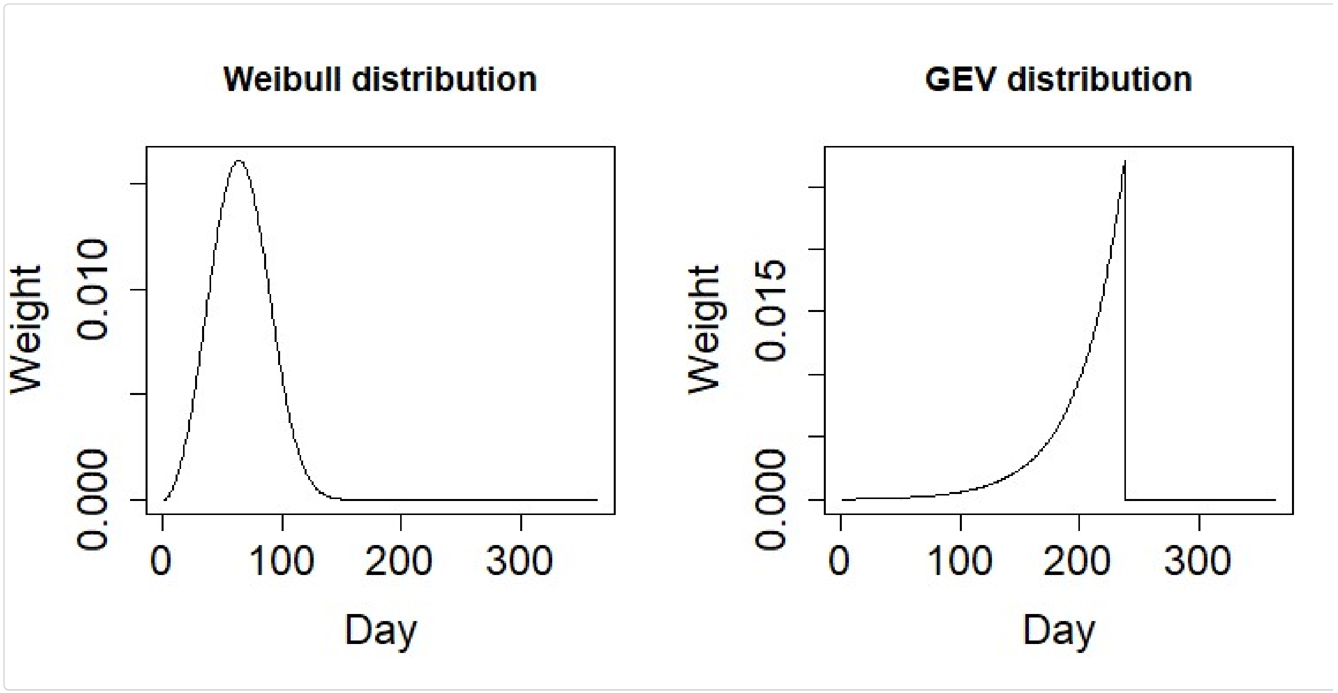
The ‘spatial’ parameter is a list item that includes the SiteID variable for the biological and climate datasets respectively. When `slidingwin` fits individual climate windows, each biological record will be matched with the climate data from the corresponding site.

`weightwin` function

When we run regular `slidingwin` analyses we assume that all days within the climate window are evenly weighted (see figure below). While this is often a convenient assumption, it can be considered biologically unrealistic.



This assumption can be a problem when we are considering more complex hypotheses. For example, we may be interested in looking for climate windows where the importance of climate decays slowly over time. The function `weightwin` allows users to fit either Weibull (below left) and generalised extreme value (GEV; below right) weight distributions to climate data as an alternative to a uniform weight distribution, thus removing the assumption of uniformity.



Instead of varying the start and end date of climate windows like `slidingwin`, `weightwin` instead uses an optimisation function to vary the shape, scale and location of either of these weight functions. Each weight function is then applied to the climate data, which is then used to produce a climate model and $\Delta AICc$ value. Therefore, although the method of optimising climate data is different, the ultimate output (i.e. $\Delta AICc$ of a climate window compared to a null model) is the same.

A worked example

`weightwin` can often be useful to use with climate data where we have already identified a climate window using `slidingwin`. Here, we will use `weightwin` to further investigate the `Mass` and `MassClimate` data included with the `climwin` package.

The basic parameters in `weightwin` are the same as `slidingwin` (though note the absence of the 'stat' parameter). In addition however, we must designate which weight distribution we want to use. In this case we consider a Weibull function.

```
weightfunc = "W"
```

Next, we must set the location, scale and shape values for the starting distribution that will be used to begin the optimisation procedure. The default values (3, 0.2, 0) are often appropriate for fitting Weibull distributions. However, you can explore different parameter values using the `explore` function.

The optimisation function used in `weightwin` has the potential to get 'stuck' during the optimisation routine. With the argument 'n' we can run our `weightwin` analysis multiple times to ensure we are finding the true optimal weight distribution.

```
n = 5
```

```
set.seed(100)

weight <- weightwin(n = 5, xvar = list(Temp = MassClimate$Temp), cdate = MassClimate$Date,
                    bdate = Mass$Date,
                    baseline = lm(Mass ~ 1, data = Mass),
                    range = c(150, 0),
                    func = "lin", type = "absolute",
                    refday = c(20, 5),
                    weightfunc = "W", cinterval = "day",
                    par = c(3, 0.2, 0))
```

If we look at the iterations object in the `weightwin` output, we can see the outcome of all 5 iterations.

```
weight$iterations
```

start_shape	start_scale	start_location	deltaAICc
3.0	0.2	0.0	-68.21592
3.1	2.7	-4.5	-68.21592
0.7	2.7	-5.1	-68.21592
8.1	3.8	-4.5	-68.21592
1.8	6.3	-1.2	-35.34690

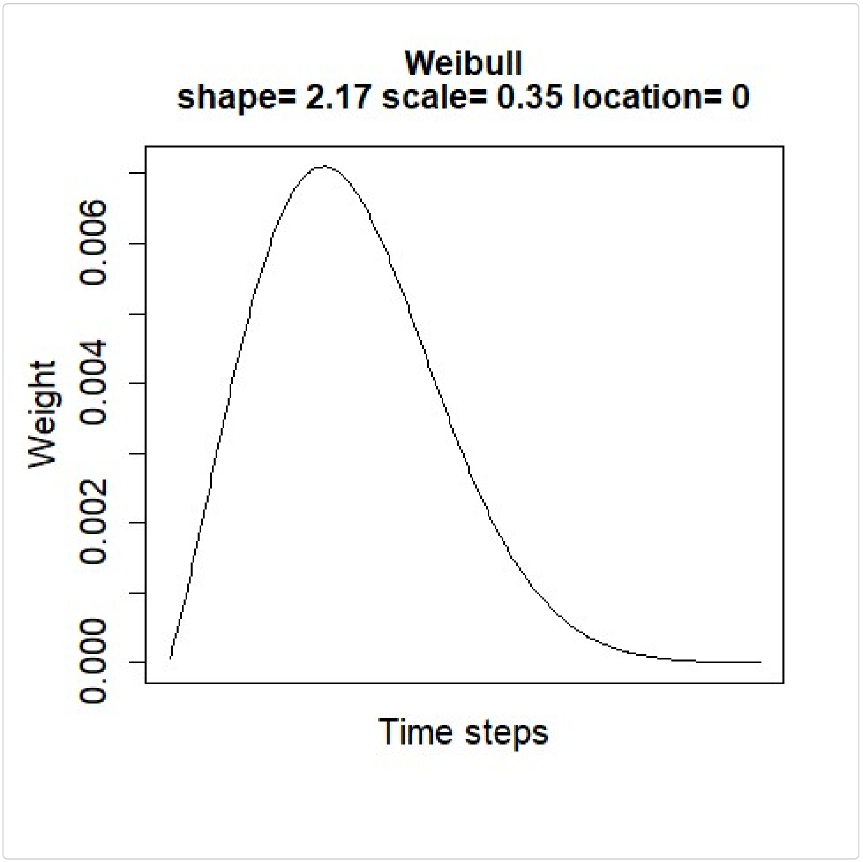
start_shape, start_scale and start_location show the starting parameters used for each iteration of `weightwin`. In the deltaAICc column we can see the outcomes of the different iterations. While 4 of the iterations reached the same peak, the 5th got stuck during the optimisation routine. This is a useful demonstration of why multiple iterations are important.

It seems that there is a clear peak that our optimisation routine has been able to identify, next we can visualise that peak using the `explore` function.

We can use the shape, scale and location parameters of the first iteration to fit our plot.

```
weight[[1]]$WeightedOutput
```

deltaAICc	start_shape	start_scale	start_location
-68.21592	2.17	0.35	0.00



In the above plot, we can see that the importance of temperature declines rapidly either side of the peak.

If we compare this value to the delta AICc obtained from the `slidingwin` function we can see that the `weightwin` function is better able to explain variation in our mass parameter as we have removed the assumption of a uniform weight distribution.

slidingwin	**weightwin**
-64.81	-68.22

Pros and cons

`weightwin` can provide greater detail on the relationship between climate and the biological response, such as the occurrence of exponential functions. Additionally, by using more diverse weight distributions, `weightwin` will often generate models with better $\Delta AICc$ values, which may be especially important when users are most interested in achieving high explanatory power. Furthermore, by using an optimisation routine `weightwin` often tests far fewer models than `slidingwin`, allowing for more rapid analysis.

Despite these benefits, `weightwin` will not always be the most appropriate function for all scenarios. Firstly, the nature of the fitted weight distributions means that `weightwin` can only detect single climate signals, which forces users to detect and compare potential climate signals with separate analyses. This is not an issue using `slidingwin`.

Secondly, `weightwin` can be a more technically complex process. While the above example works easily, in other scenarios the optimisation procedures may often get stuck on false optima or fail to converge. In these cases, users may be required to test different starting parameters and adjust optimisation characteristics such as step size. Often this procedure can be inhibitive for users with less technical knowledge.

Finally, `weightwin` can only be used for testing mean climate, with no capacity to consider other aggregate statistics. Therefore, whether one chooses to use `weightwin` or `slidingwin` will largely depend on the summary statistic used, the level of detail desired, and the user's technical knowledge.

To ask additional questions or report bugs please e-mail:

liam.bailey@anu.edu.au

or visit our google group