delta downscaling

THIS IS WORK IN PROGRESS, DO NOT USE FOR ANY ANALYSIS

Note that we will use the draft functions for downscaling, which can be accessed by usign the prefix `pastclim:::`. The `:::` access internal functions. You will also get a message reminding you that these functions are not final and thus should not be used for real analysis.

Delta downscaling a dataset in `pastclim`

The `terra` package can downscale reconstructions using a bilinear method. Other `R` packages also offer various approaches to downscale rasters.

For palaeoclimate reconstructions, the delta method has been shown to be very effective (Beyer et al, REF), but sea level changes require some care on how to apply such a method. `pastclim` includes functions to use the delta method for downscaling. We will focus on South East Asia, as this area was greatly affected by sea level changes over the last glaciation. Specifically, we will look at the area around Borneo (for real applications, we would reccomend using a bigger extent in areas of large changes in land extent, as interpolating over a small extent can lead to greater artefacts; for this example, we keep the extent small to reduce computational time).

An example for one variable

When downscaling, it is generally a better idea to downscale the monthly variables, and then recompute the BIOCLIM variables. We will usethe monthly variables from the Beyer2020 dataset, and focus on the time steps that are also available in the Example dataset (i.e. a subset of the full Beyer2020 dataset, to reduce computational time):

```
library(terra)
#> terra 1.6.47
library(pastclim)
sea_ext<- terra::ext(110, 120, -5, 5)
tavg_vars <- c(paste0("temperature_0",1:9),paste0("temperature_",10:12))
time_steps <- get_time_steps(dataset = "Example")
```

The series for this region could be generated with the following command:

```
tavg_series <- region_series(bio_variables =tavg_vars,
                             time_bp =  time_steps,
                             dataset = "Beyer2020",
                              ext = sea_ext)
```

In this vignette, we will fetch pre-generated series stored in the package

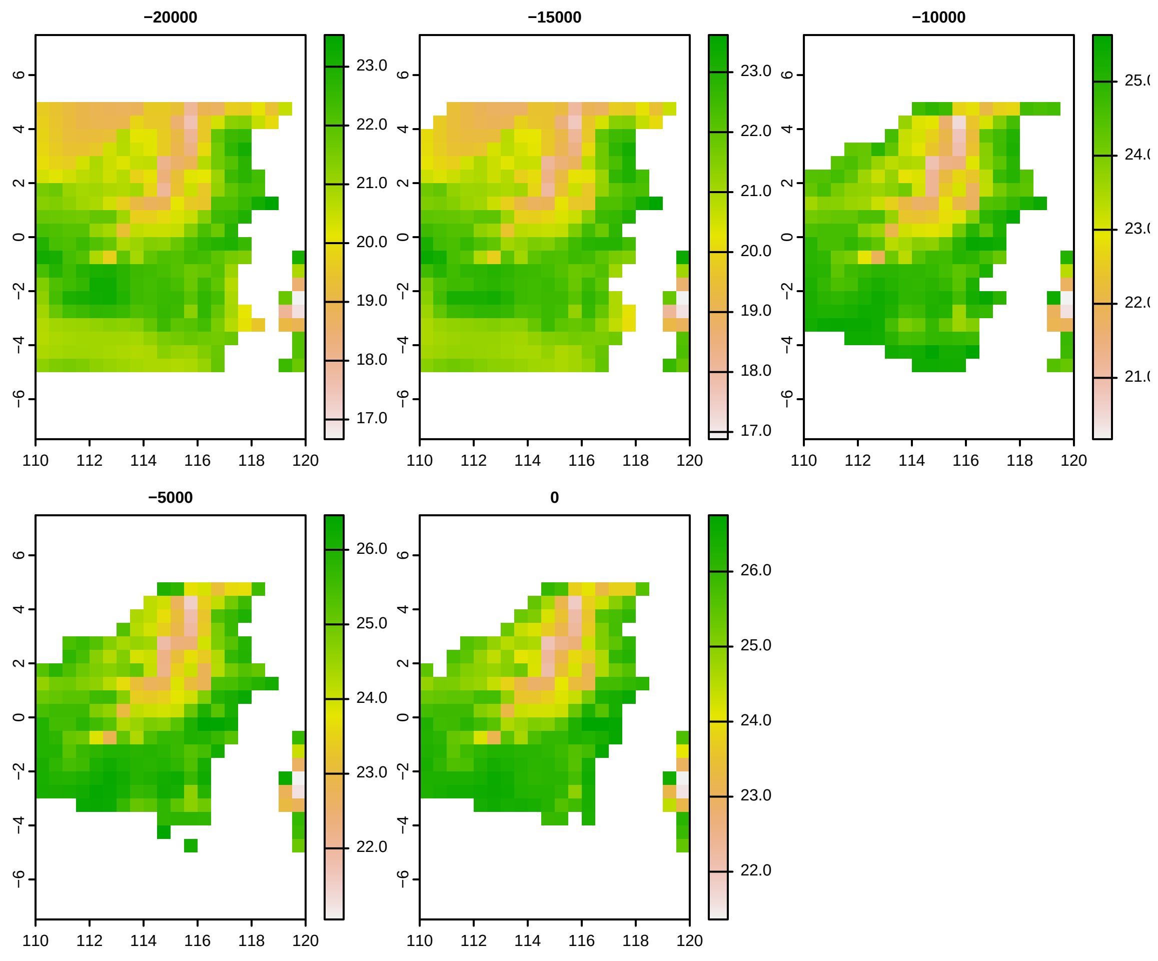
```
#fetch the data
```

Downscaling is performed one variable at a time. We will start with temperature in January. So, we first need to extract the `SpatRaster` of model low resolution data from the `SpatRasterDataset`:

```
tavg_model_lres_rast <- tavg_series$temperature_01
tavg_model_lres_rast
#> class       : SpatRaster
#> dimensions  : 20, 20, 5  (nrow, ncol, nlyr)
#> resolution  : 0.5, 0.5  (x, y)
#> extent      : 110, 120, -5, 5  (xmin, xmax, ymin, ymax)
#> coord. ref. : lon/lat WGS 84
#> source(s)   : memory
#> varname     : temperature_01
#> names       : temper~-20000, temper~-15000, temper~-10000, temper~_-5000, temper~e_01_0
#> min values  :      16.66488,      16.87925,      20.17156,      21.04706,      21.36333
#> max values  :      23.53901,      23.61953,      25.62949,      26.46226,      26.75000
#> time (years): -18050 to 1950
```

And we can now plot it:

```
plot(tavg_model_lres_rast, main = time_bp(tavg_model_lres_rast))
```



We can see how that the reconstructions are rather coarse (the Beyer2020 dataset uses 0.5x0.5 degree cells). We now need a set of high resolutions observations for the variable of interest that we will use to generate the delta raster used to downscale reconstructions. We will use data from WorldClim2 at 10 minute resolution (but other datasets such as CHELSA would be equally suitable). For WorldClim, there are some convenient functions to download and then load variables:

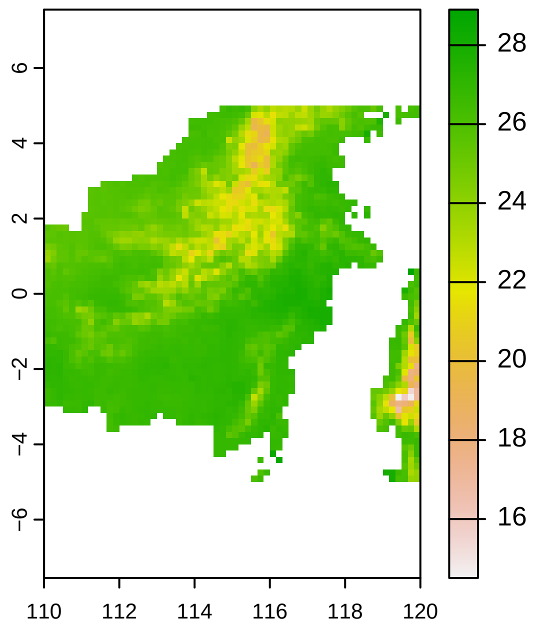
```
pastclim:::download_worldclim("tavg",10)
#> /home/andrea/.local/share/R/pastclim/wc2.1_10m_tavg.nc already exists
#> [1] FALSE
```

Once the variable is downloaded, we can load it at any time with:

```
tavg_obs_hres_series <- pastclim:::load_worldclim("tavg",10)
```

We want to crop these reconstructions to the extent of interest

```
tavg_obs_hres_series <- terra::crop(tavg_obs_hres_series, sea_ext)
# extract the january raster
tavg_obs_hres_rast <- tavg_obs_hres_series[[1]]
plot(tavg_obs_hres_rast)
```



We need to make sure that the extent of the modern observations is the same as the extent of the model reconstructions:

```
ext(tavg_obs_hres_rast)==ext(tavg_model_lres_rast)
#> [1] TRUE
```

If that was not the case, we would use `terra::crop` to match the extents.

We also need a high resolution global relief map (i.e. integrating both topographic and bathymetric values) to reconstruct past coastlines following sea level change. The relief raster will need to have the same extent and resolution as the high resolution observations. We can download one (based on ETOPO2022) with:

```
relief_rast <- pastclim:::download_relief(tavg_obs_hres_rast)
#> Registered S3 methods overwritten by 'adehabitatMA':
#>   method                       from
#>   print.SpatialPixelsDataFrame sp
#>   print.SpatialPixels          sp
#> Querying NOAA database ...
#> This may take seconds to minutes, depending on grid size
#> Building bathy matrix ...
```

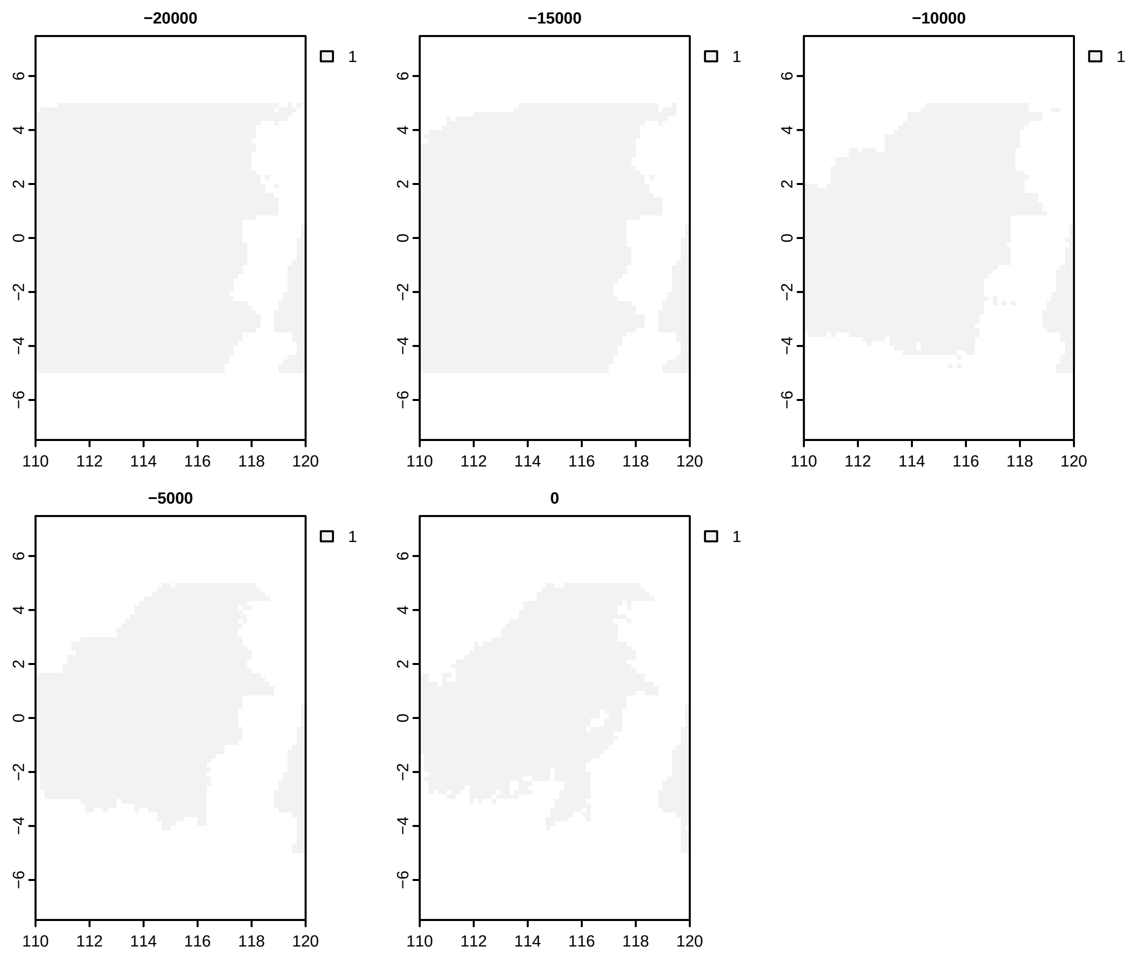
We can quickly confirm that the resulting relief raster has the same extent as the model reconstructions, as well as the same resolution of the high resolution climate observations:

```
ext(relief_rast) == ext(tavg_model_lres_rast)
#> [1] TRUE

ext(relief_rast) == ext(tavg_obs_hres_rast)
#> [1] TRUE
ncol(relief_rast) == ncol(tavg_obs_hres_rast)
#> [1] TRUE
nrow(relief_rast) == nrow(tavg_obs_hres_rast)
#> [1] TRUE
```

We can now generate a high resolution land mask for the periods of interest. By default, we use the sea level reconstructions from Spratt et al 2016, but a different reference can be used by setting sea levels for each time step (see the man page for `make_land_mask` for details):

```
high_res_mask <- pastclim:::make_land_mask(relief_rast = relief_rast,
                                 time_bp = time_bp(tavg_model_lres_rast))
#> This function is still under development; do not use it for real analysis
plot(high_res_mask, main=time_bp(high_res_mask))
#> Warning in time_bp(high_res_mask): the time units of SpatRaster are not 'years'
#> it might be a problem with the time units not being properly set in the original
#> nc file
```

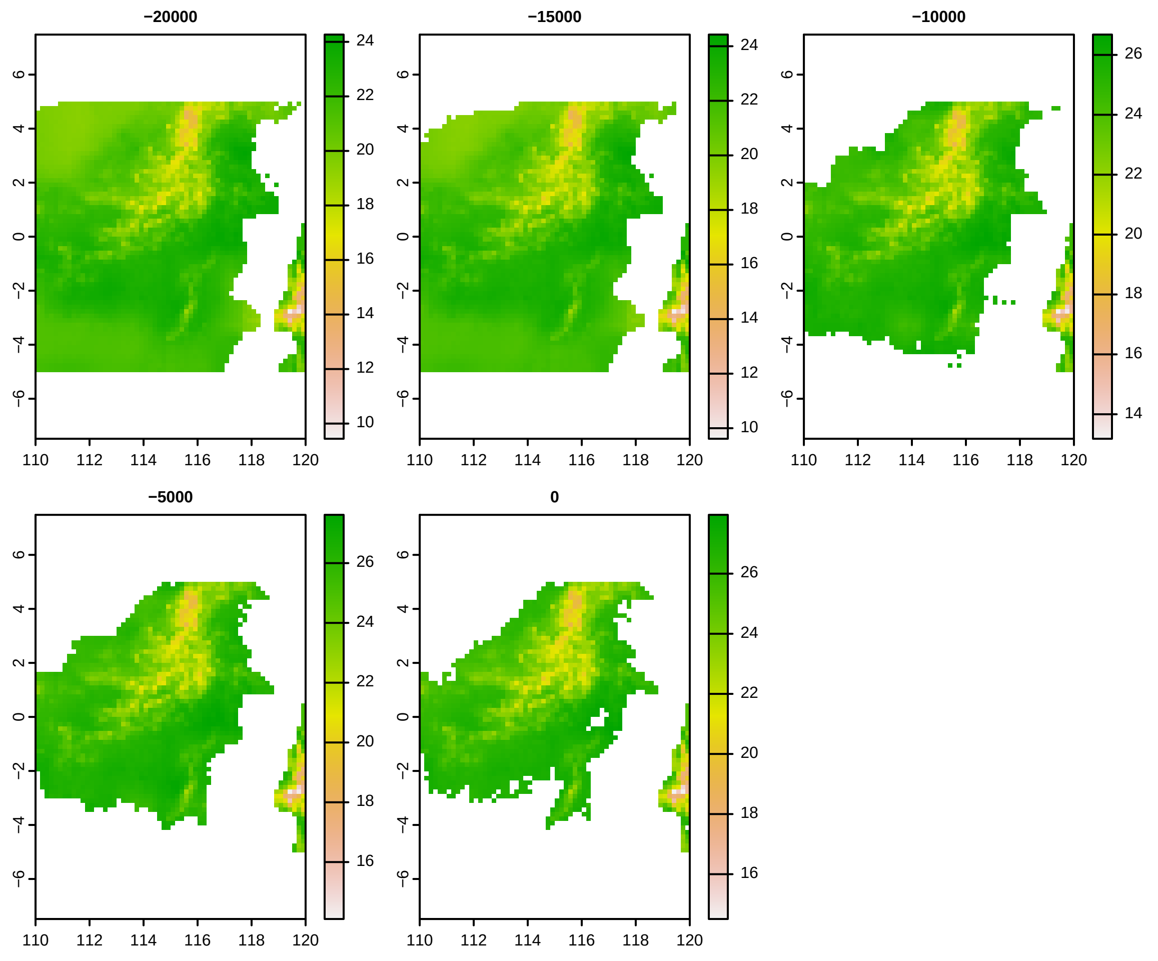


We can now compute a delta raster and use it to downscale the model reconstructions:

```
delta_rast<-pastclim:::delta_compute(x=tavg_model_lres_rast, ref_time = 0,
                                      obs = tavg_obs_hres_rast)
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
model_downscaled <- pastclim:::delta_downscale (x = tavg_model_lres_rast,
                                                delta_rast = delta_rast,
                                                x_landmask_high = high_res_mask)
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
model_downscaled
#> class       : SpatRaster
#> dimensions  : 60, 60, 5  (nrow, ncol, nlyr)
#> resolution  : 0.1666667, 0.1666667  (x, y)
#> extent      : 110, 120, -5, 5  (xmin, xmax, ymin, ymax)
#> coord. ref. : lon/lat WGS 84
#> source(s)   : memory
#> names       : temper~-20000, temper~-15000, temper~-10000, temper~_-5000, temper~e_01_0
#> min values   :       9.442662,        9.61378,       13.18173,       14.09556,       14.50825
#> max values   :      24.266855,       24.42670,       26.67793,       27.61437,       27.95691
#> time (years): -18050 to 1950
```

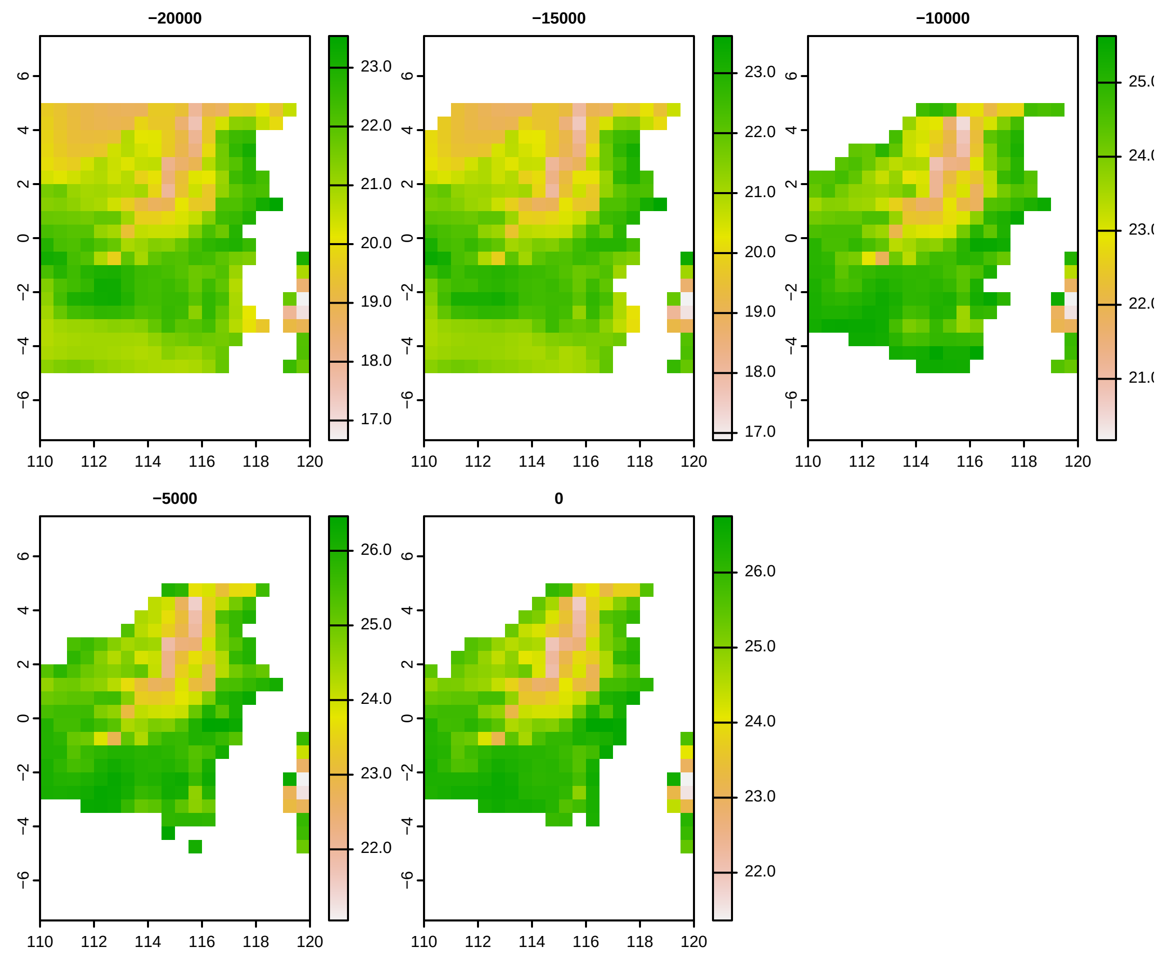
Let's inspect the resulting data:

```
plot(model_downscaled, main = time_bp(model_downscaled))
```



And, as a reminder, the original reconstructions (note that the colour scales are not the same! `terra` chooses a scale for each time step based on the time specific range):

```
plot(tavg_model_lres_rast, main = time_bp(tavg_model_lres_rast))
```



Computing the bioclim variables

To compute the bioclim variables, we need to repeat the procedure above for temperature and precipitation for all months. Let us start with temperature. We loop over each month, create a `SpatRaster` of downscaled temperature, add it to a list, and finally conver the list into a `SpatRasterDataset`

```
tavg_downscaled_list<-list()
for (i in 1:12){
  delta_rast<-pastclim::delta_compute(x=tavg_series[[i]], ref_time = 0,
                                       obs = tavg_obs_hres_series[[i]])
  tavg_downscaled_list[[i]] <- pastclim::delta_downscale (x = tavg_series[[i]],
                                                  delta_rast = delta_rast,
                                                  x_landmask_high = high_res_mask)
}
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
tavg_downscaled <- terra::sds(tavg_downscaled_list)
```

Quickly inspect the resulting dataset:

```
tavg_downscaled
#> class       : SpatRasterDataset
#> subdatasets : 12
#> dimensions  : 60, 60 (nrow, ncol)
#> nlyr        : 5, 5, 5, 5, 5, 5, 5, 5, 5
#> resolution  : 0.1666667, 0.1666667  (x, y)
#> extent      : 110, 120, -5, 5  (xmin, xmax, ymin, ymax)
#> coord. ref. : lon/lat WGS 84
#> source(s)   : memory
```

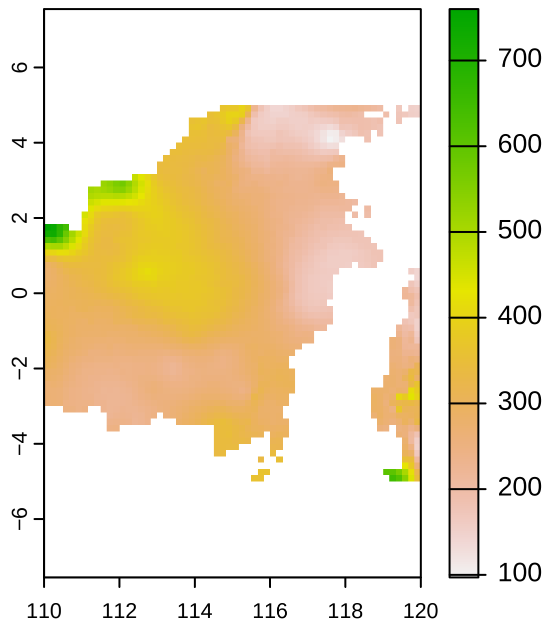
As expected, we have 12 months (subdatasets), each with 5 time steps.

We now need to create a series for precipitation:

```r
prec_vars <- c(paste0("precipitation_0",1:9),paste0("precipitation_",10:12))
prec_series <- region_series(bio_variables = prec_vars,
                             time_bp =  time_steps,
                             dataset = "Beyer2020",
                             ext = sea_ext)
```

Get some high resolution observations:

```r
pastclim:::download_worldclim("prec",10)
#> /home/andrea/.local/share/R/pastclim/wc2.1_10m_prec.nc already exists
#> [1] FALSE
prec_obs_hres_series <- pastclim:::load_worldclim("prec",10)
prec_obs_hres_series <- terra::crop(prec_obs_hres_series, sea_ext)
# extract the january raster
prec_obs_hres_rast <- prec_obs_hres_series[[1]]
plot(prec_obs_hres_rast)
```



And finally downscale precipitation:

```r
prec_downscaled_list<-list()
for (i in 1:12){
  delta_rast<-pastclim:::delta_compute(x=prec_series[[i]], ref_time = 0,
                                       obs = prec_obs_hres_series[[i]])
  prec_downscaled_list[[i]] <- pastclim:::delta_downscale (x = prec_series[[i]],
                                                            delta_rast = delta_rast,
                                                            x_landmask_high = high_res_mask)
}
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> This function is still under development; do not use it for real analysis
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
#> [inverse distance weighted interpolation]
prec_downscaled <- terra::sds(prec_downscaled_list)
```

We are now ready to compute the bioclim variables:

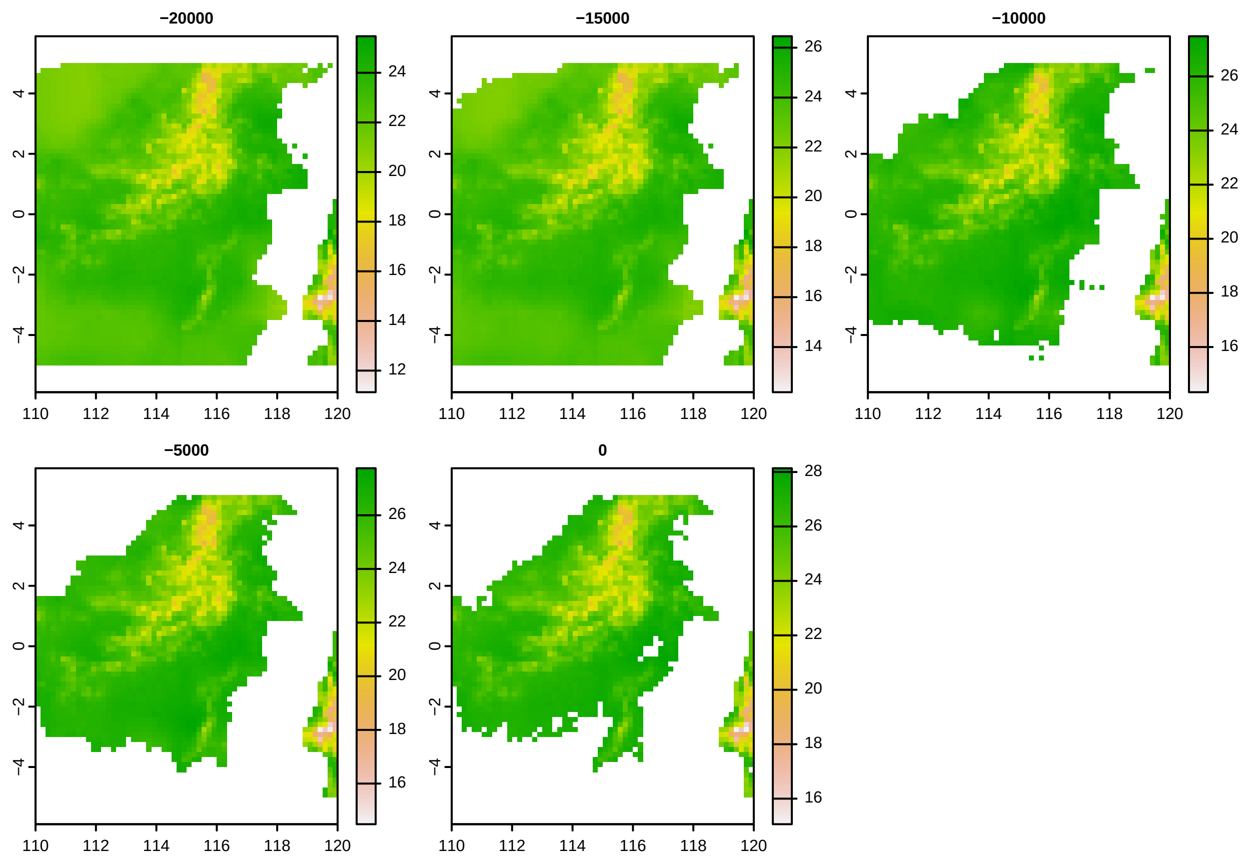
```
bioclim_downscaled<-bioclim_vars(tavg =tavg_downscaled, prec = prec_downscaled)
```

Let's inspect the object:

```
bioclim_downscaled
#> class       : SpatRasterDataset
#> subdatasets : 17
#> dimensions  : 60, 60 (nrow, ncol)
#> nlyr        : 5, 5, 5, 5, 5, 5, 5, 5
#> resolution  : 0.1666667, 0.1666667  (x, y)
#> extent      : 110, 120, -5, 5  (xmin, xmax, ymin, ymax)
#> coord. ref. : lon/lat WGS 84
#> source(s)   : memory
```

And plot the first variable (bio01):

```
plot(bioclim_downscaled[[1]],main=time(bioclim_downscaled[[1]]))
```


−20000
−15000
−10000
−5000
0