Tasks for chapter 7: Going beyond linearity: regional climatology

The task is to compute a climatology of daily temperatures (min, max, average) for every location in a region from irregularly distributed measurement locations. You'll need the package "stars" to handle spatial (geographical objects) plus the "colorspace" package for plotting. And of course "mgcv" to estimate the GAMs.

1. **Data preparation**
   Load `tirolTdaily.rda`, which contains daily min/max/average Temperatures for 30+ stations in Tirol. Data are strung together one station after the other. Check the names of the variables contained in `tirolTdaily`.

Extract the names of all stations using `stations <- unique(...)`.

Add day of year `yday` to `tirolTdaily`.

Use `readRDS` to read in the digital elevation data from `dem.rds` (assign to `dem`) and coordinates of boundaries of Tirol from `tirol.gadm.rds` (assign to `tirol`).

Visually ascertain that loading these data worked by plotting them. The TODO Georg: update geo-framework "raster" package's `plot()` handles all the spatial details. A simple `plot(dem, col = gray.colors(51))` will do the job. You can also use any other color palette of your choice. Similarly plot the state boundaries.

2. **Fit GAM with altitude and yday**
   Fit a GAM `tModS` to the daily mean temperatures with a smooth function for `alt` (altitude) with `k=5`. You may want to read the help page with `choose.k` to learn more details about choosing a value for `k`. Let the second additive term be a cyclic spline for `yday` (with `k=10`).

3. **Fit GAM with seasonally varying altitude and lat-lon, respectively**
   Fit another GAM `tModA` to daily average temperatures with two further additive terms: one to account for the fact that the altitude effect changes seasonally (think of inversions present in the valley in cold season but not in warm season); and another one to account for possible regional (lat/lon) changes with season (yday). You specify these interaction effects with
   `ti(yday, alt, bs = c("cc","cr"), k = c(5,5))`.

Note that since you need different base functions (cyclic for yday, but only a regular spline for altitude) you need to concatenate them with `c(,)`.

Seasonally different effects for different parts of Tirol can be specified with
`ti(yday, lat, lon, bs = c("cc", "tp"), d = c(1, 2), k = c(8, 20))`. Here you are additionally declaring the 2-dimensionality of the lat/lon data with the option `d`. You can experiment with setting different values for k (again after reading `?choose.k`).

Similarly fit a GAM `tModMax` for maximum temperatures and another one `tModMin` for minimum temperatures.

Plot the effects (=functional shape) of each of these additive terms (while the other terms are held constant, similar to the interpretation of the effect of a term in a simple linear model). The `plot.gam(tModA)` produces 4 subplots, since you specified 4 additive terms for this model. You will therefore need to subdivide your figure window before, using `par(mfrow = c(2,2))` for a 2x2 arrangement of the subfigures. The subfigures are labeled with the functional form of the additive term, e.g. "s(alt,3.92") for the smooth of the altitude. The number states the effective degrees of freedom of this smooth. The contour lines in the `ti(yday,alt)` subfigure show the seasonally varying altitude effect. The labeling of the lines is in units of the modeled variable, which in our case is temperature in degrees Celsius. Dashed contour lines indicate +- 1 standard deviations. The three-dimensional seasonally varying lat/lon effect `ti(yday, lat, lon)` will be shown for a selection of days of year. Try interpreting the effects of all 4 additive terms on temperature. Does the shape/patterns of the effects vary between average and extreme temperatures?

4. **Gridded temperature climatology**
   As a final product we want to use the fitted GAMs to obtain a gridded climatology of the respective temperatures for all grid cells contained within the state boundaries of Tirol. As grid cells we use the ones given by the digital elevation model (which we had assigned to `dem`).

```
dem  <- crop(dem, extent(tirol))
```

Then identify pixels outside the state of Tirol and create an object of class SpatialPoints:

```
sp   <- SpatialPoints(coordinates(dem), proj4string = crs(tirol))
```

Find the indices of grid cells in Tirol:
`take <- which(!is.na(over(sp, tirol)))`.

Explanation: at the spatial locations of object x, `over(x,y)` retrieves the indexes or attributes from spatial object y.

Assign NA to values outside Tirol: `values(dem)[!take] <- NA`

Create a `newdata` data.frame for the prediction for the new data locations, which are all DEM pixels within the border of Tirol:
`newdata <- as.data.frame(coordinates(dem))`

Give the lat/lon coordinates their proper names "lat" and "lon" with `names()`. Add altitude to `newdata` with `values(dem)` and then subset newdata to within boundaries of Tirol:
`newdata <- newdata[take, ]`

With this, you have climatologies with a daily resolution(!!!).

5. **Compute climatology for a particular day of the year**
   Predict/compute the climatology for a particular date, choose such a date, e.g. `date <- as.POSIXlt("2019-12-01")` and then add a `yday` variable to `newdata` from `date$yday`.

Predict/compute the climatologies for your chosen date for each of the 4 GAMs you have estimated using `predict(model, newdata = newdata)` where "model" is one of your 4 GAMs.

To be able to plot the climatologies we need get them into RasterStack object to store the predictions. A RasterStack is a collection of RasterLayer objects with the same spatial extent and resolution.

```
res <- dem
```

Get the values of the Raster objects, after first initializing them:
```
values(res) <- NA #initialize
res <- stack(res, res, res, res)
```
which creates 4 layers, one for each of the 4 models.
Let's name them:
```
names(res) <- c("tmean", "tsimple", "tmax", "tmin")
```

Then store the estimates (predictions):
`values(res$tmean)[take] <- tmpA` where `tmpA` is the variable you had assigned to the predictions for the 4-term average temperature model. Similarly for the other 3 models.

Plotting the climatologies for you chosen date is as simple as:
`plot(res$tmean)`.

To make it look nicer, you may want to use a diverging color scheme from the colorspace palette, e.g.
`col = diverge_hcl(51, h = c(260, 305),power = 1)`

You can also give the climatology plots a descriptive title by adding which climatology and which date it is, e.g.
`main = sprintf("Daily mean temperature %s", strftime(date, "%b %d"))` where `sprintf` is a low-level printing function.