In [ ]:
```
# Load libraries
using Plots, Rasters, ArchGDAL
using WhereTheWaterFlows, ImageComponentAnalysis
const WWF = WhereTheWaterFlows # this is how module aliasing is done in Julia
```

```
WhereTheWaterFlows
```

In [ ]:
```
function PreProcessTopo( DEM )
    h     = Float32.(DEM.data[:,:,1])::Matrix{Float32}
    h   .= h[:,end:-1:1]
    h[h.==DEM.missingval] .= NaN
    return h
end

function PreProcessCoords( DEM_array )
    x     = Array(DEM_array)
    x   .-= x[1]
    x   .*= 111e3
end
```

```
PreProcessCoords (generic function with 1 method)
```

Step 0: Read the data

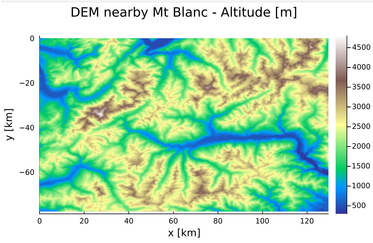
In [ ]:
```
DEM             = Raster("../data/MontBlanc.tif")
```

```
4202×2517 Raster{Int16,2} with dimensions:
  X Projected{Float64} LinRange{Float64}(6.61958, 7.78653, 4202) ForwardOrdered Regular Intervals{Start} crs: WellKnownText,
  Y Projected{Float64} LinRange{Float64}(46.1382, 45.4393, 2517) ReverseOrdered Regular Intervals{Start} crs: WellKnownText
and reference dimensions:
  Band Categorical{Int64} 1:1 ForwardOrdered
extent: Extent(X = (6.619583333358065, 7.786805555580442), Y = (45.439305555553624, 46.138472222220386))missingval: -32768crs: GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]],AXIS["Latitude",NORTH],AXIS["Longitude",EAST],AUTHORITY["EPSG","4326"]]
parent:
             46.1382    46.1379    46.1376  …    45.4399    45.4396    45.4393
 6.61958  1034       1050       1071          1008       1016       1027
 6.61986  1022       1034       1056          1008       1017       1027
 6.62014  1011       1022       1037          1010       1017       1026
 ⋮                                  ⋱                    ⋮
 7.78597  1418       1424       1433           639        639        637
 7.78625  1383       1372       1379           643        641        637
 7.78653  1342       1353       1354           646        638        630
```

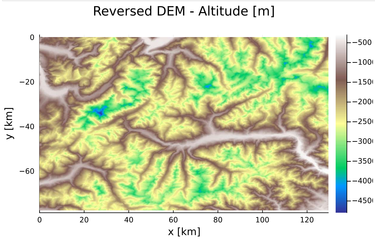
In [ ]:
```
h     = PreProcessTopo( DEM )                   # read h and
x     = PreProcessCoords( DEM.dims[1] )         # read x
y     = PreProcessCoords( DEM.dims[2] )[end:-1:1] # read y and reverse it
h_rev = .-copy(h);                              # flip the topography
```

Step 1: Plot the data

In [ ]:
```
p = heatmap(x./1e3, y./1e3, h', color=:terrain, aspect_ratio=1,
xlabel="x [km]", ylabel="y [km]", title="DEM nearby Mt Blanc – Altitude [m]")
```

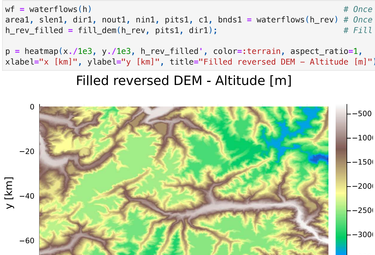


In [ ]:
```
p = heatmap(x./1e3, y./1e3, h_rev', color=:terrain, aspect_ratio=1,
xlabel="x [km]", ylabel="y [km]", title="Reversed DEM – Altitude [m]")
```



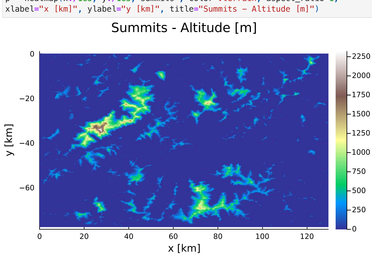
Step 2: Let's fill the sinks using WhereTheWaterFlows

In [ ]:
```
wf = waterflows(h)                                          # Once on h
area1, slen1, dir1, nout1, nin1, pits1, c1, bnds1 = waterflows(h_rev) # Once on h_rev
h_rev_filled = fill_dem(h_rev, pits1, dir1);                # Fill the reversed topo

p = heatmap(x./1e3, y./1e3, h_rev_filled', color=:terrain, aspect_ratio=1,
xlabel="x [km]", ylabel="y [km]", title="Filled reversed DEM – Altitude [m]")
```



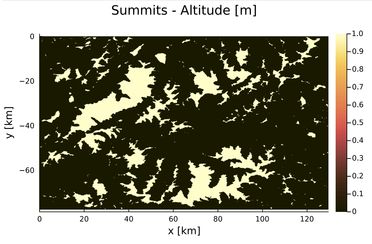
Step 3: Difference between the filled reversed DEM and the reversed DEM

In [ ]:
```
summits = h_rev_filled .- h_rev;

p = heatmap(x./1e3, y./1e3, summits', color=:terrain, aspect_ratio=1,
xlabel="x [km]", ylabel="y [km]", title="Summits – Altitude [m]")
```



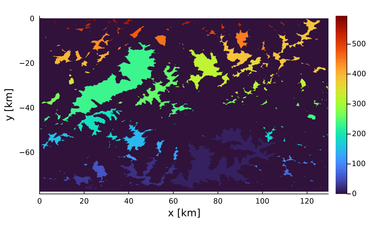
Step 4: Mask the summits

In [ ]:
```
ϵ = 50.0
mask_summits = summits .> ϵ # filter noise au passage

p = heatmap(x./1e3, y./1e3, mask_summits', color=:lajolla, aspect_ratio=1,
xlabel="x [km]", ylabel="y [km]", title="Summits – Altitude [m]")
```



Step 5: Label components using ImageComponentAnalysis

In [ ]:
```
components = label_components(mask_summits)

p = heatmap(x./1e3, y./1e3, components', color=:turbo, aspect_ratio=1,
xlabel="x [km]", ylabel="y [km]", title="Summits – Components (n = $(maximum(components)) with ϵ = $(ϵ) m)")
```



In [ ]:
```
```