Welcome to

instats

The Session Will Begin Shortly

START

Spatial Data Analysis and Visualization in R

Session 15: Working with Spatiotemporal Data Cubes in R using the Stars Package

instats

Contents

- Use **stars** for space–time raster data
- Visualize with **tmap 4.1**
- Learn cube slicing, aggregation, and animation

Required Packages

```r
library(stars)
library(tmap)
library(sf)
library(dplyr)
```

Load a tif file ((Landsat data) into stars

```r
tif_file <- system.file("tif/L7_ETMs.tif", package = "stars")
x <- read_stars(tif_file)
x
```

```
stars object with 3 dimensions and 1 attribute
attribute(s):
             Min. 1st Qu. Median     Mean 3rd Qu. Max.
L7_ETMs.tif     1      54     69 68.91242      86  255
dimension(s):
     from  to  offset delta                     refsys point x/y
x       1 349  288776  28.5 SIRGAS 2000 / UTM zone 25S FALSE [x]
y       1 352 9120761 -28.5 SIRGAS 2000 / UTM zone 25S FALSE [y]
band    1   6      NA    NA                         NA    NA
```

Dimensions

```
st_dimensions(x)
```

```
     from  to  offset delta                    refsys point x/y
x       1 349  288776  28.5 SIRGAS 2000 / UTM zone 25S FALSE [x]
y       1 352 9120761 -28.5 SIRGAS 2000 / UTM zone 25S FALSE [y]
band    1   6      NA    NA                        NA    NA
```

- Think of an object with k dimensions as a k-dimensional cube
- In this example, 2 of them contain geographical axes, x and y.
- The third dimension band contains **spectral bands**

Land dataset

The `land` dataset (included in **tmap**) has four attributes

```
land
```

```
stars object with 2 dimensions and 4 attributes
attribute(s):
               cover                                   cover_cls
 Water bodies     :393060   Water                          :393060
 Snow / Ice       : 61986   Snow/ice                       : 61986
 Herbaceous       : 21377   Forest                         : 48851
 Tree Open        : 16171   Other natural vegetation       : 32611
 Sparse vegetation: 12247   Bare area/Sparse vegetation: 26904
 Cropland         : 11658   Cropland                       : 17843
 (Other)          : 66701   (Other)                        :  1945
     trees           elevation
 Min.   :  0.00   Min.   :-412
 1st Qu.:  0.00   1st Qu.: 218
 Median :  0.00   Median : 608
 Mean   : 15.59   Mean   :1140
 3rd Qu.: 19.00   3rd Qu.:1941
 Max.   :100.00   Max.   :6410
 NA's   :393060   NA's   :389580
```

Use `names(land)` to obtain the names

Attributes

- Each attribute is a different instance using common coordinates (dimensions)
- When the dimensions are x and y, each attribute can be seen as a different layer
- Data in each attributes can have different formats: e.g. factors, numbers, etc.

Indexing stars objects

Number of indices: 1 + number of dimensions.

```
x[,,,]
```

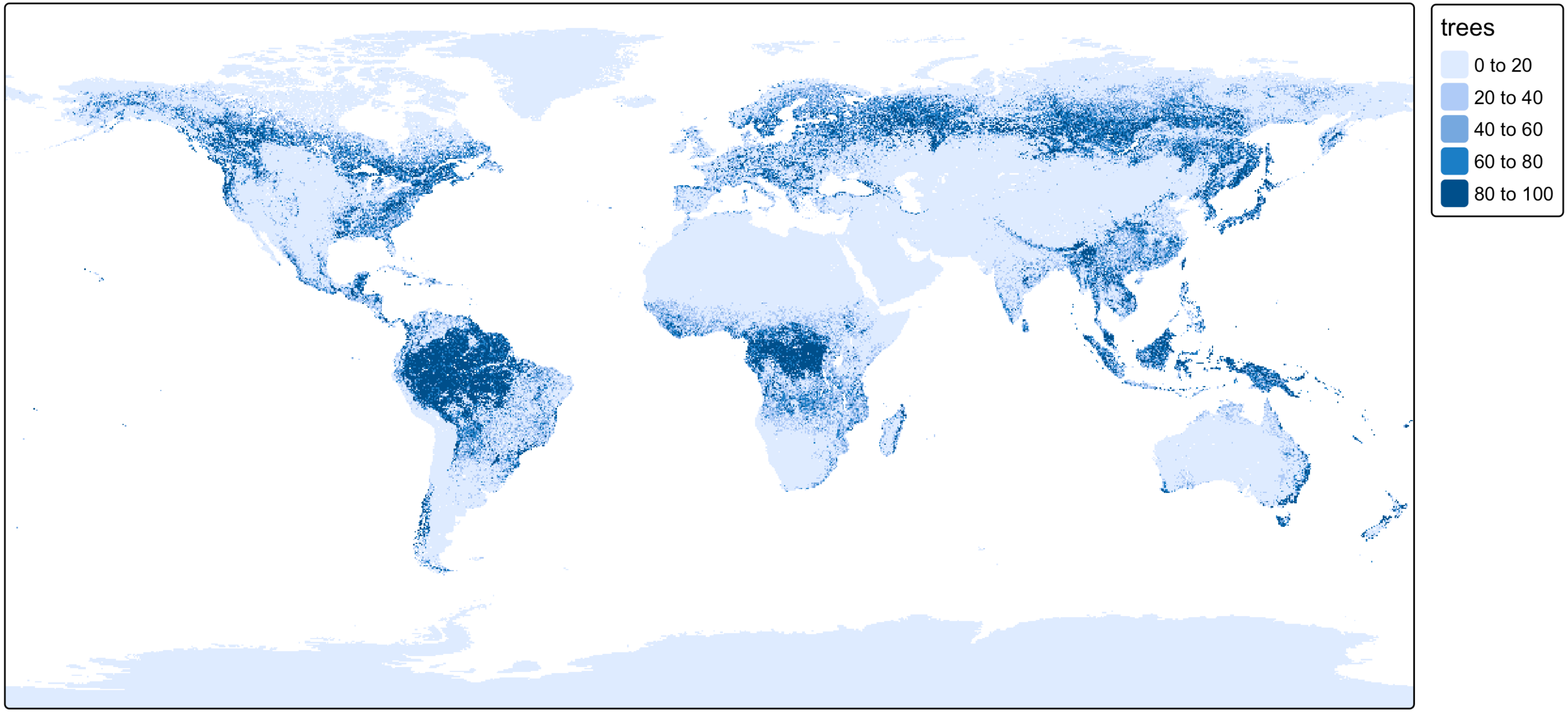
E.g. `x[, 1:10, 1:10, 1:3]` gives the first 10 values of both x and y, and the first 3 values of `"band"`.

```
land[,,]
```

E.g. `land[2,,]` gives the second attribute, and all `x` and `y` values.

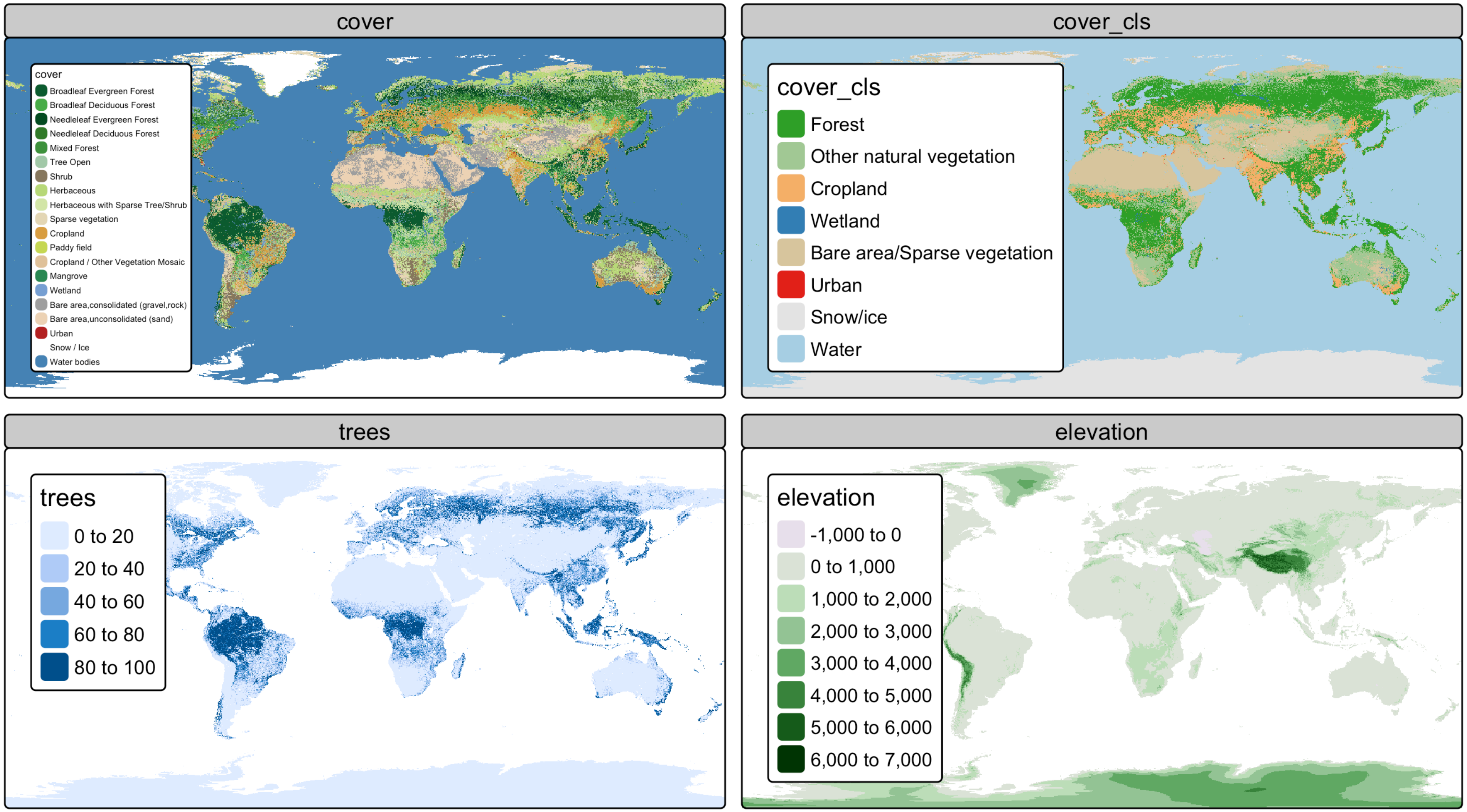
Plotting one attribute

```
tm_shape(land) +
    tm_raster("trees")
```



Plotting all attributes

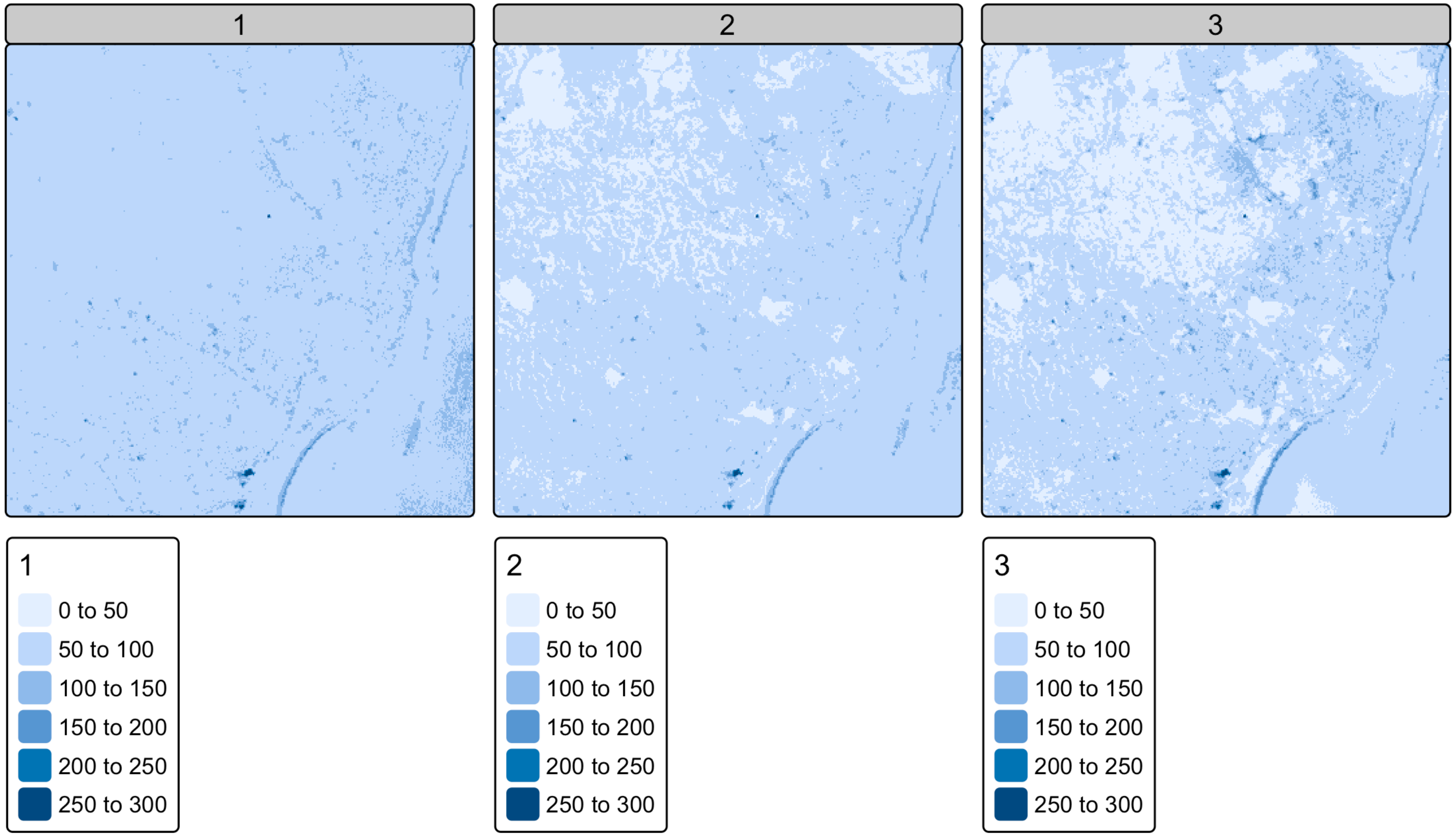
```
tm_shape(land) +
    tm_raster()
```



Plotting 3rd dimension

Plot the first three bands from the third dimension, called `"band"`.

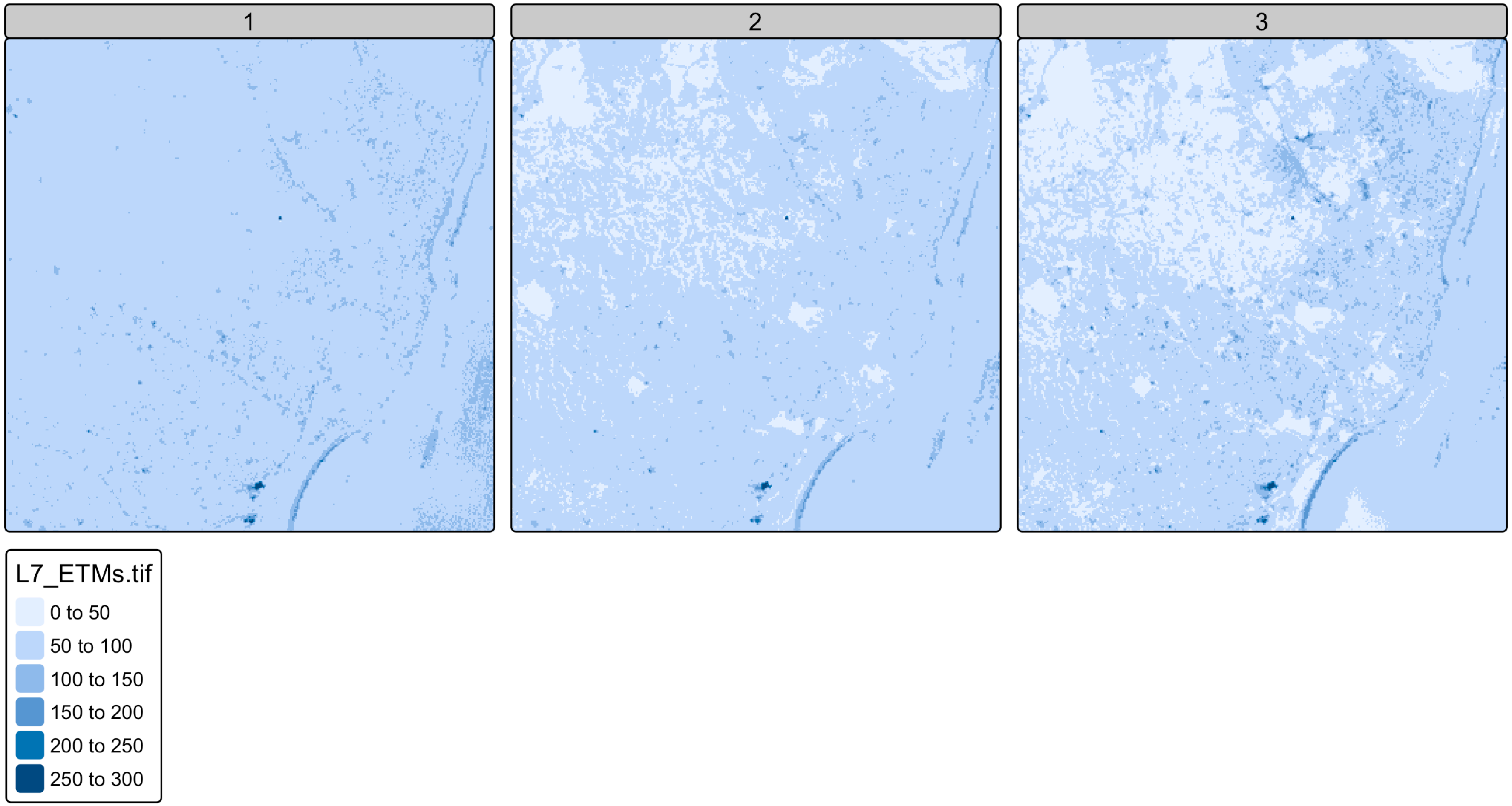
```
tm_shape(x) +
    tm_raster(col = tm_vars(dimvalues = 1:3))
```



tmap automatically selects the non-x-y band to cut slices.

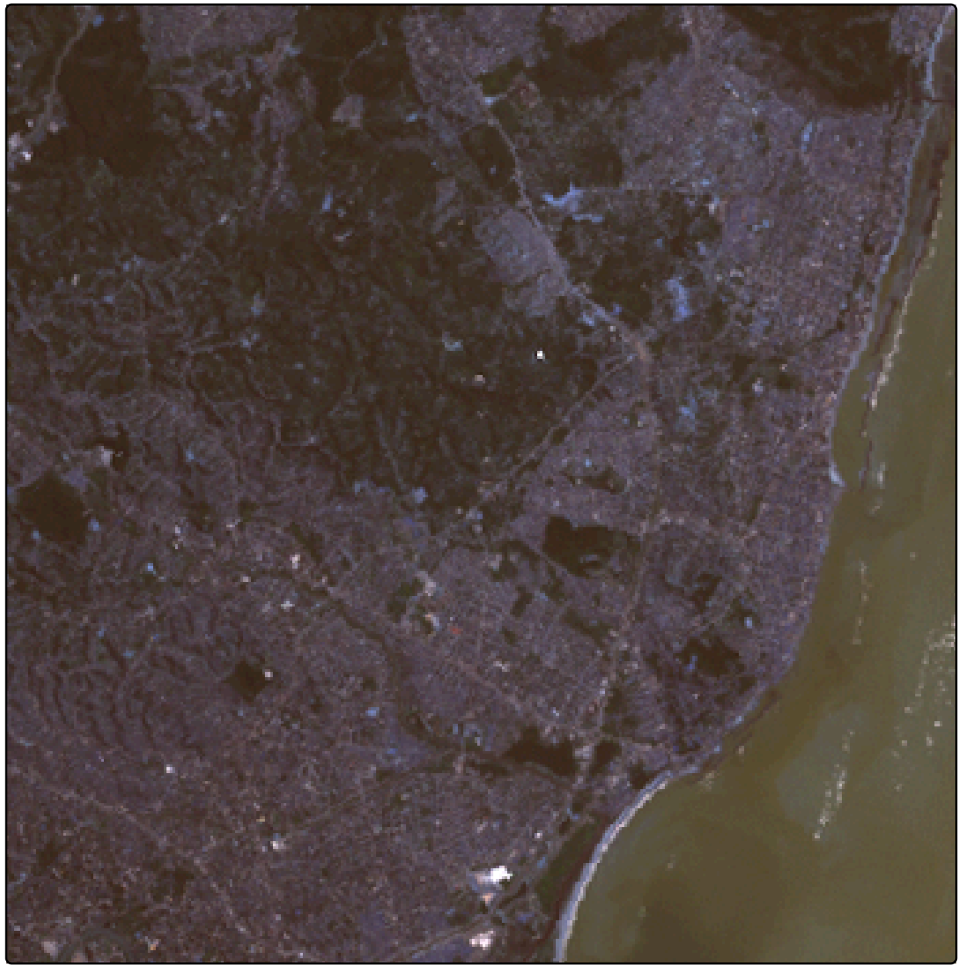
Alternative approach

```r
tm_shape(x[,,,1:3]) +
  tm_raster()
```



Plotting an rgb image

```
tm_shape(x) +
    tm_rgb(col = tm_vars(dimvalues = 1:3, multivariate = TRUE))
```



Multivariate

- Is specified with `tm_vars()`
- When `multivariate = FALSE`, each (dimension)value is shown as a facet
- When `multivariate = TRUE`, the dimension values are mapped to one map variable (in this case color)

Converting from a dimension to multiple attributes

```
(x_splitted = split(x, "band"))
```

```
stars object with 2 dimensions and 6 attributes
attribute(s):
    Min. 1st Qu. Median     Mean 3rd Qu. Max.
X1    47      67     78 79.14772      89  255
X2    32      55     66 67.57465      79  255
X3    21      49     63 64.35886      77  255
X4     9      52     63 59.23541      75  255
X5     1      63     89 83.18266     112  255
X6     1      32     60 59.97521      88  255
dimension(s):
  from  to  offset delta                     refsys point x/y
x    1 349  288776  28.5 SIRGAS 2000 / UTM zone 25S FALSE [x]
y    1 352 9120761 -28.5 SIRGAS 2000 / UTM zone 25S FALSE [y]
```

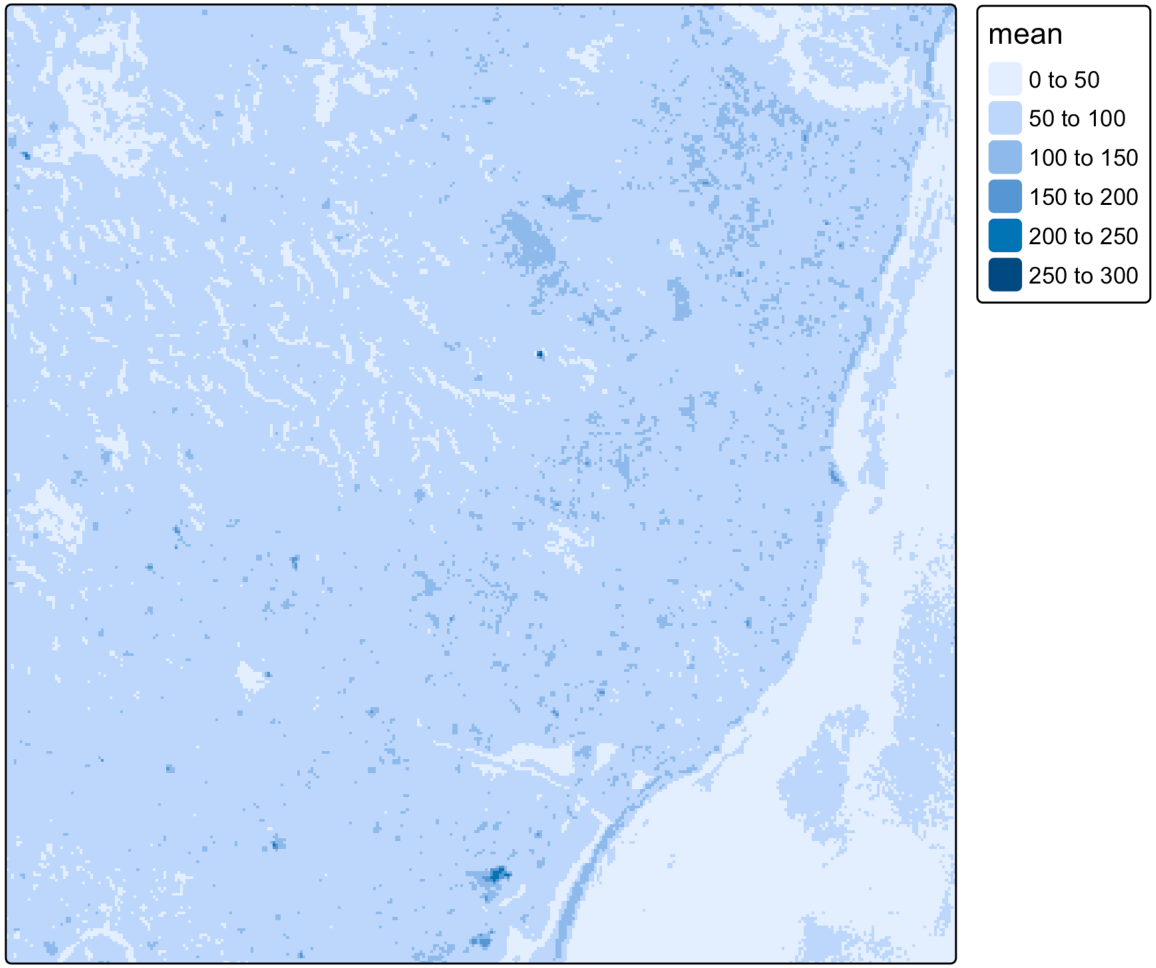
Converting from multiple attributes to a new dimension

```r
x_merged = merge(x_splitted, name = "band")
names(x_merged) = "L7_ETMs"

x_merged
```

```
stars object with 3 dimensions and 1 attribute
attribute(s):
         Min. 1st Qu. Median     Mean 3rd Qu. Max.
L7_ETMs     1      54     69 68.91242      86  255
dimension(s):
     from  to  offset delta                     refsys point    values x/y
x       1 349  288776  28.5 SIRGAS 2000 / UTM zone 25S FALSE      NULL [x]
y       1 352 9120761 -28.5 SIRGAS 2000 / UTM zone 25S FALSE      NULL [y]
band    1   6      NA    NA                         NA    NA X1,...,X6
```

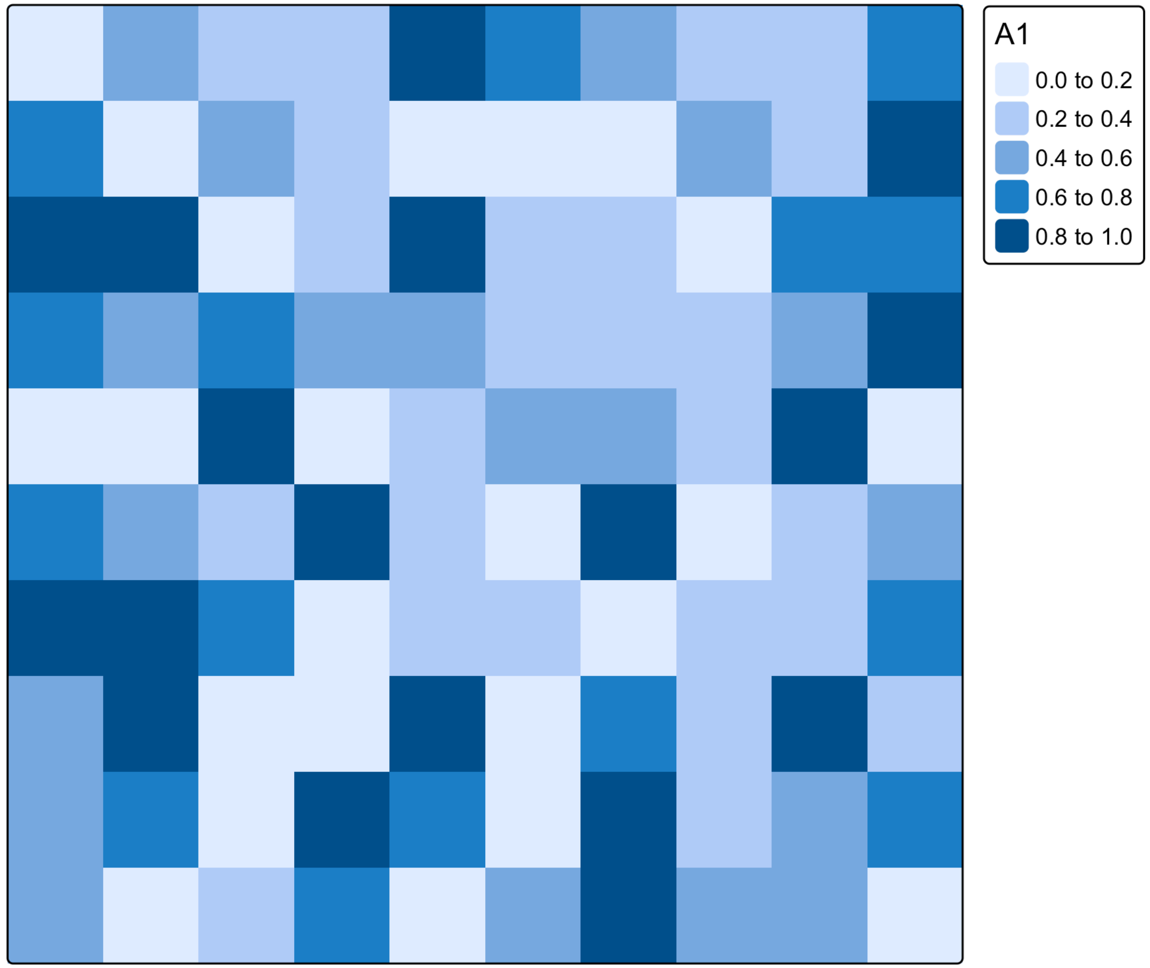
Aggregate

```
x_mean <- st_apply(x, c("x", "y"), mean)
tm_shape(x_mean) +
  tm_raster()
```



Create stars from array

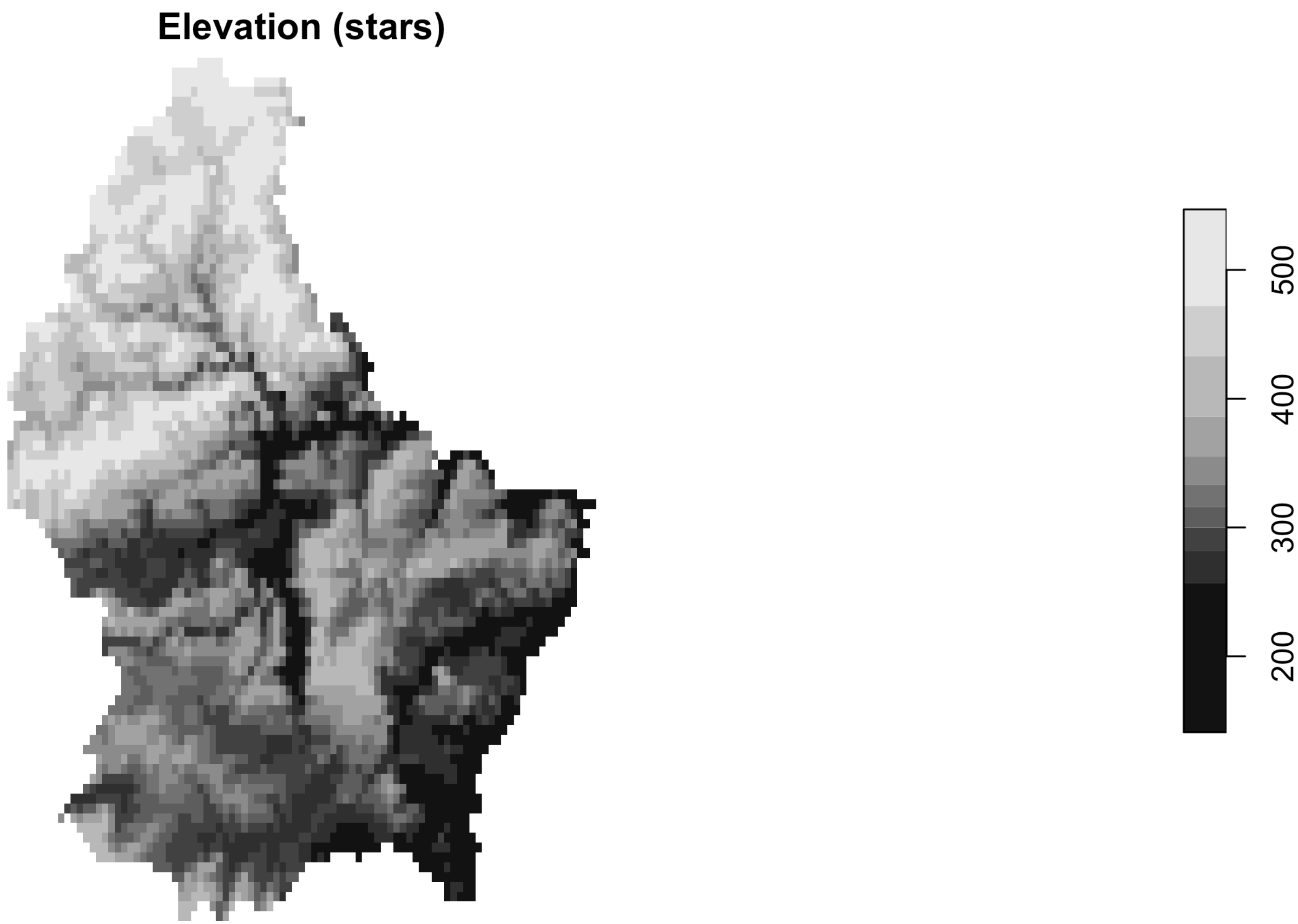
```r
arr = array(runif(100), dim = c(10, 10))
s = st_as_stars(arr)
st_crs(s) = 4326

qtm(s)
```


A1
0.0 to 0.2
0.2 to 0.4
0.4 to 0.6
0.6 to 0.8
0.8 to 1.0

Reading and writing with stars

Read GeoTIFF with stars

```r
s = read_stars(system.file("ex/elev.tif", package = "terra"))
plot(s, main = "Elevation (stars)")
```



Metadata and Dimensions

```r
st_dimensions(s)
```

```
  from to offset     delta refsys point x/y
x    1 95  5.742  0.008333 WGS 84 FALSE [x]
y    1 90  50.19 -0.008333 WGS 84 FALSE [y]
```

```r
st_crs(s)
```

```
Coordinate Reference System:
  User input: WGS 84
  wkt:
GEOGCRS["WGS 84",
    ENSEMBLE["World Geodetic System 1984 ensemble",
        MEMBER["World Geodetic System 1984 (Transit)"],
        MEMBER["World Geodetic System 1984 (G730)"],
        MEMBER["World Geodetic System 1984 (G873)"],
        MEMBER["World Geodetic System 1984 (G1150)"],
        MEMBER["World Geodetic System 1984 (G1674)"],
        MEMBER["World Geodetic System 1984 (G1762)"],
        MEMBER["World Geodetic System 1984 (G2139)"],
        MEMBER["World Geodetic System 1984 (G2296)"],
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]],
        ENSEMBLEACCURACY[2.0]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
```

Write Raster with stars

```r
write_stars(s, "elevation_output_stars.tif")
```

stars Proxy Objects

- Useful for large rasters or NetCDF files
- Load metadata only, not full data

```
# Read in proxy mode (only header is read)
s_proxy = read_stars(system.file("ex/elev.tif", package = "terra"), proxy = TRUE)
s_proxy
```

```
stars_proxy object with 1 attribute in 1 file(s):
$elev.tif
[1] "[...]/elev.tif"

dimension(s):
  from to offset     delta refsys point x/y
x    1 95  5.742  0.008333 WGS 84 FALSE [x]
y    1 90  50.19 -0.008333 WGS 84 FALSE [y]
```

- Convert to regular stars object if needed:

```
s_loaded = st_as_stars(s_proxy)
```

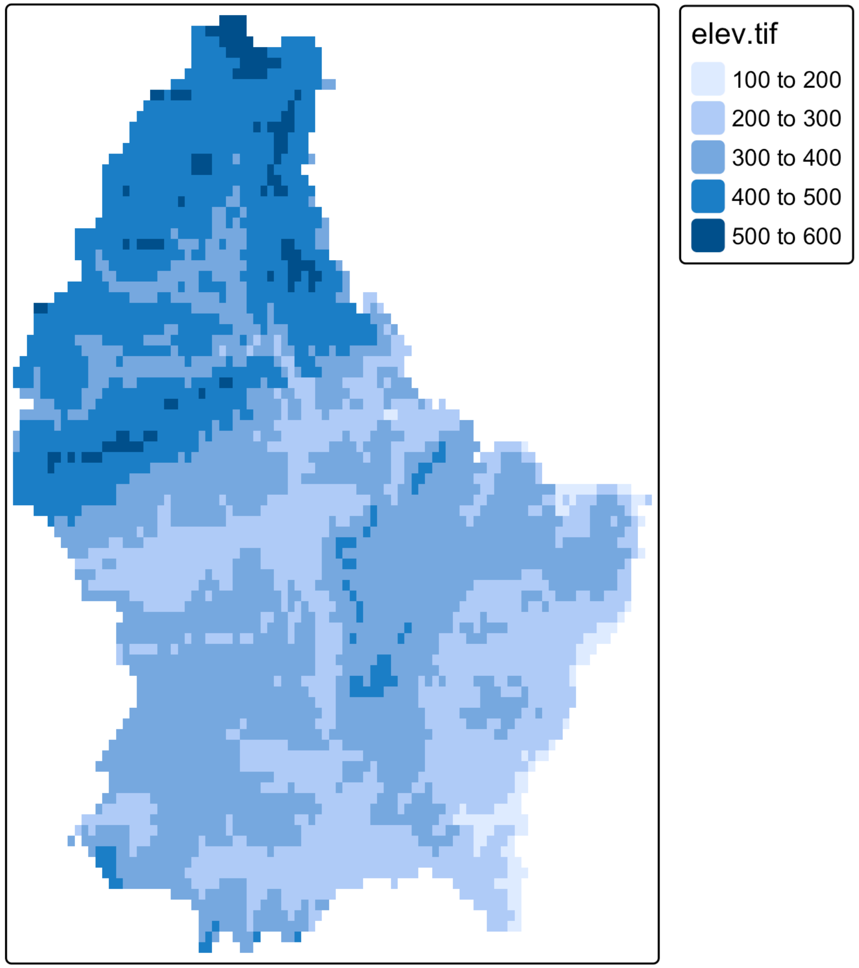
Read NetCDF with stars (example)

```r
s_nc = read_stars("my_file.nc", proxy = TRUE)
```

Downsampling, Warping, and Transforming Raster Data with terra

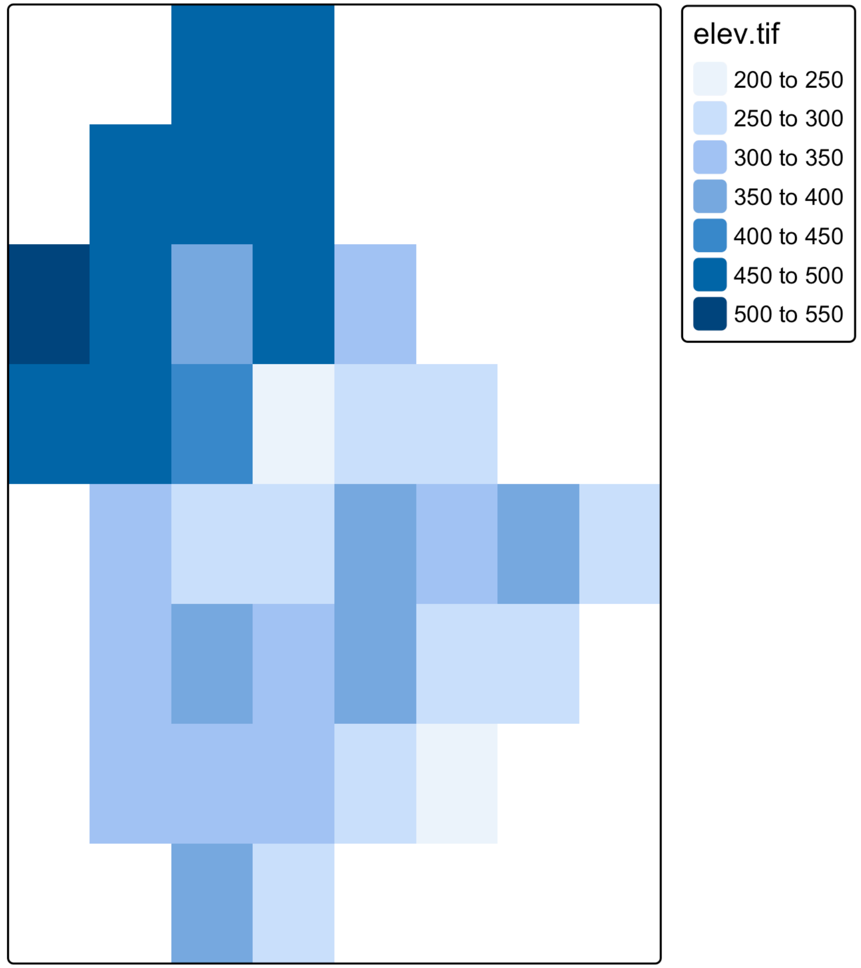
Read Raster with stars

```r
s <- read_stars(system.file("ex/elev.tif", package = "terra"))
qtm(s)
```



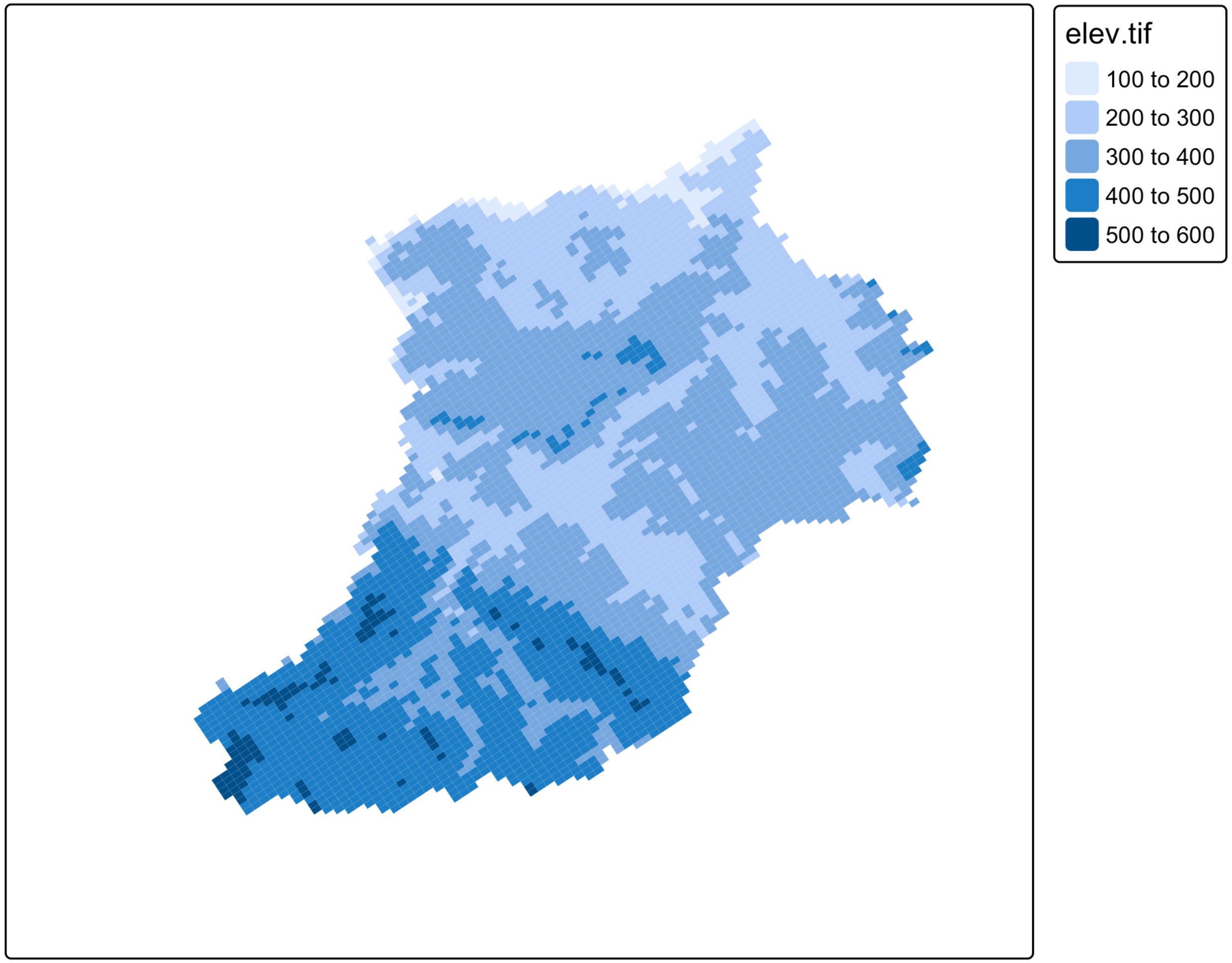
Downsample with st_warp()

```
# Create low-res grid
lowres <- st_as_stars(st_bbox(s), nx = 8, ny = 8)
st_crs(lowres) <- st_crs(s)

# Warp to low-res
s_lowres <- st_warp(s, lowres)
qtm(s_lowres)
```


elev.tif
200 to 250
250 to 300
300 to 350
350 to 400
400 to 450
450 to 500
500 to 550

Transform CRS with stars

```r
s_reproj <- st_transform(s, crs = 32612)
qtm(s_reproj)
```



Comparison between stars and terra

Metadata access

Operation	terra	stars
Get CRS	`crs(x)`	`st_crs(x)`
Set CRS	`crs(x) <- ...`	`st_set_crs(x, ...)`
Get bounding box	`ext(x)`	`st_bbox(x)`
Get resolution	`res(x)`	`dim(x)`
Get number of rows	`nrow(x)`	`nrow(x)`
Get number of columns	`ncol(x)`	`ncol(x)`

Sampling, warping, and transformation

Operation	**terra**	**stars**
Downsampling	`aggregate()`	`st_warp()` with coarse grid
Upsampling	`disagg()`	Use dense grid in `st_warp()`
Warp to new grid	`resample()`	`st_warp()`
CRS transform	`project()`	`st_transform()`

Raster - vector conversion

Operation	terra	stars
Vector → raster	`rasterize()`	`st_rasterize()`
Raster → points	`as.points()`	`st_as_sf(..., as_points=TRUE)`
Raster → polygons	`as.polygons()`	`st_as_sf(..., as_points=FALSE)`

Summary

- stars is great for high-dimensional raster data
- stars complements terra — not a replacement
- Key strengths:
    - → Multi-band/multi-time
    - → Flexible array manipulation
    - → Clean sf integration

STOP