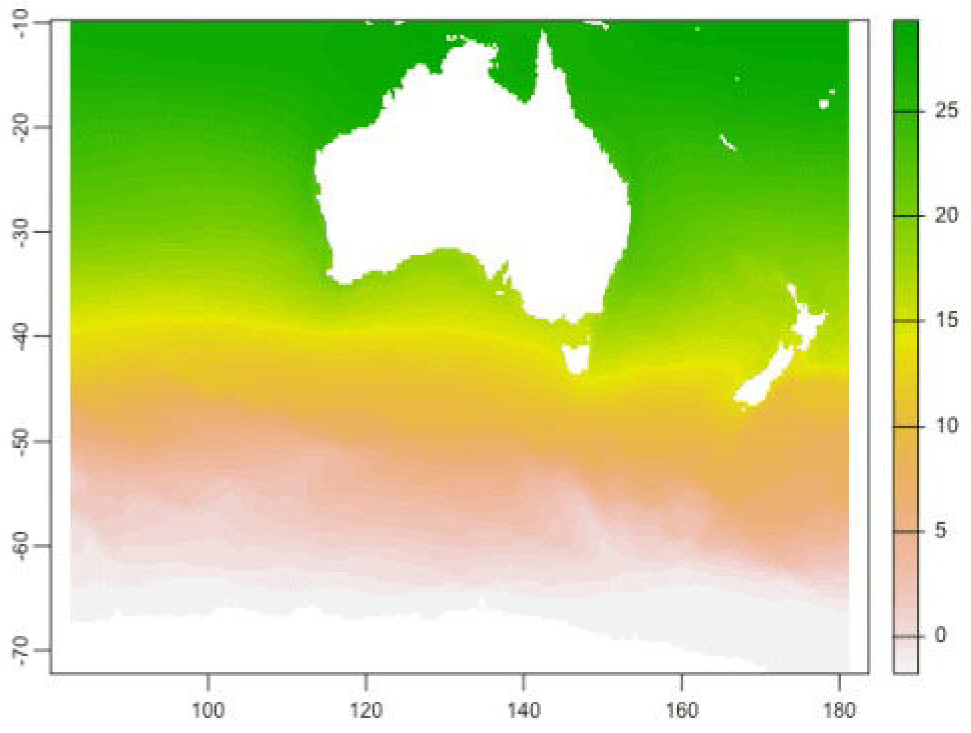
1 How long will it take to learn terra's syntax?

First, let's take a look at some basic syntax and compare it with raster

You can read in data much the same way, with the command `rast()`:

```
library(terra)
r <- rast("data-for-course/spatial-data/MeanAVHRRSST.grd")
plot(r)
```



```
ext(r)
```

```
## SpatExtent : 82.5, 181.25, -72.25, -9.75 (xmin, xmax, ymin, ymax)
```

Now let's crop and reproject it:

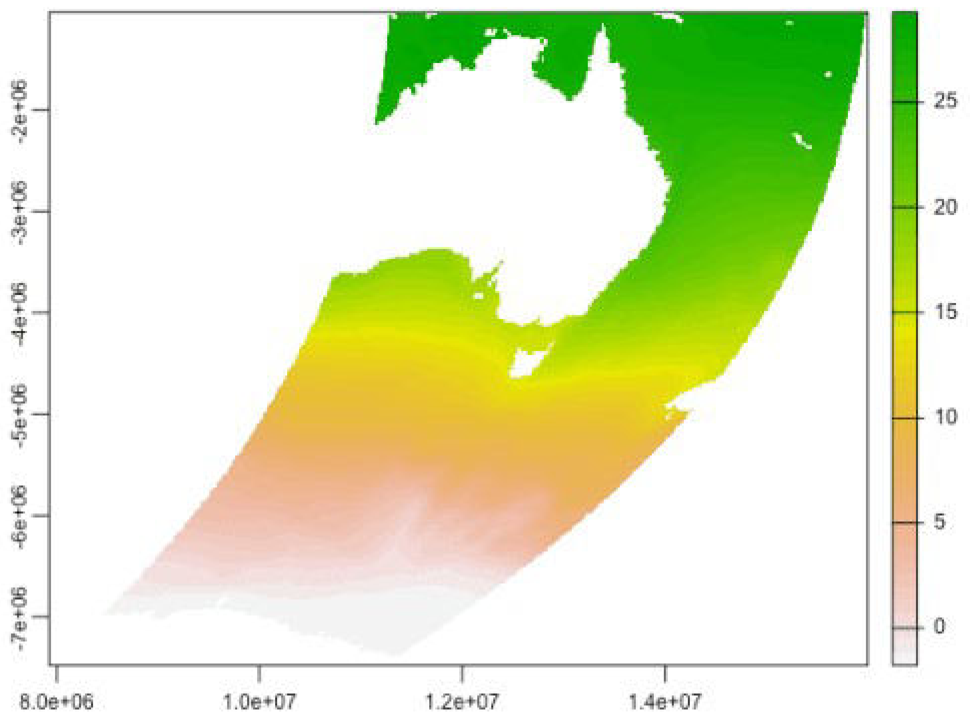
```
#create an extent object
ext2 <- ext(r)

#constrain it in x direction
ext2[1] <- 120
ext2[2] <- 170

r2 <- crop(r, ext2)

r3 <- project(r2, "+proj=robin +lon_0=0 +x_0=0 +y_0=0 +ellps=WGS84
+datum=WGS84 +units=m +no_defs +towgs84=0,0,0")

plot(r3)
```



So much of the syntax is familiar (or identical), if slightly different. It took me about 10 minutes to translate what I know from raster to terra syntax.

Note there are some important caveats with terra when it comes to cluster computations and saving data see `?terra` for more information.

2 How much help is available online?

It's early days yet. But the terra package documentation is outstanding, as good as it was for raster. This was one reason raster became so popular.

`?terra` provides a very helpful description, a menu of functions and at the very end a translation of function names from raster to terra (many are the same)

So users will be once again grateful to Robert Hijmans and the authorship team for the effort tney put into package documentation

There are a few courses/ blogs online if you google it and some limited posts on stackexchange sites.

No vignette with the package as yet.

So the verdict is that the documentation of the package and functions is excellent. Currently, there is limited existing documentation of troubleshooting errors and bugs online. So you might have to ask yourself. But online content will grow as the package becomes more popular.

3 Is terra faster than raster?

I take the author's word that its faster, but let's see how much faster:

```
library(microbenchmark)
```

```
r_raster <- raster::raster("data-for-course/spatial-data/
MeanAVHRRSST.grd")
robin_proj <- "+proj=robin +lon_0=0 +x_0=0 +y_0=0 +ellps=WGS84
+datum=WGS84 +units=m +no_defs +towgs84=0,0,0"

tout <- microbenchmark(
project(r, robin_proj),
raster::projectRaster(r_raster, crs = robin_proj),
times = 10
)
tout

## Unit: milliseconds
##                                                expr     median
##                              project(r, robin_proj)      76.3
##  raster::projectRaster(r_raster, crs = robin_proj)     529.6
```

So something like 7 times faster for the computationally demanding task of reprojecting a raster.

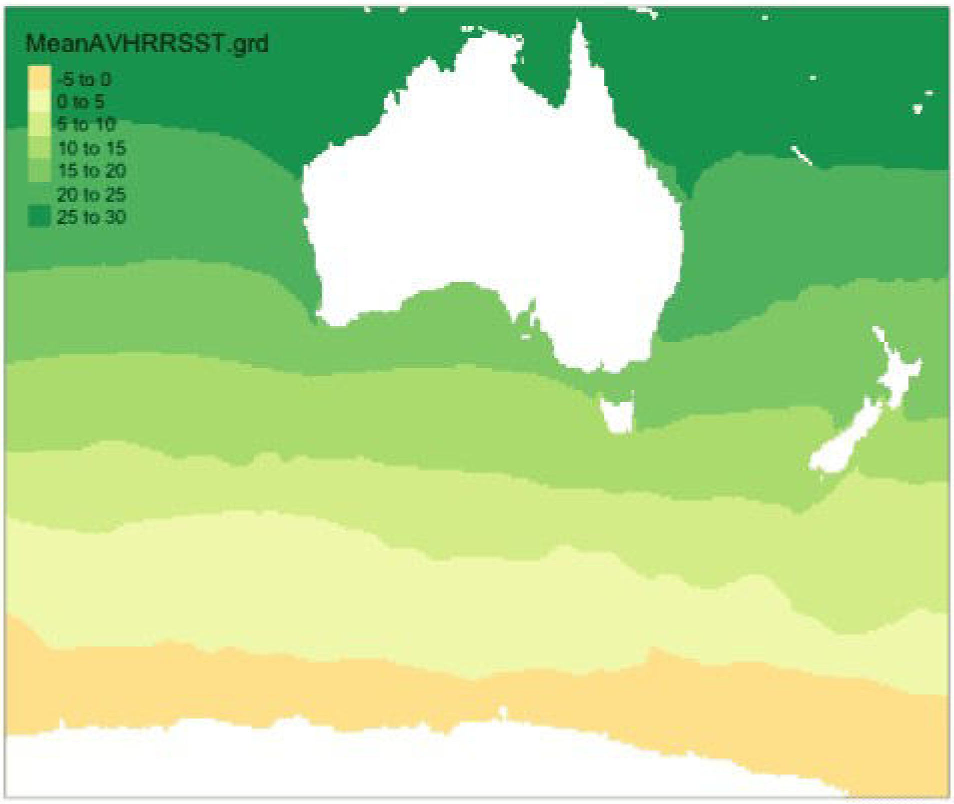
4 Will terra be compatible with other packages I use?

The answer here obviously depends on what packages you want to use. A key one for me is tmap for mapping. This doesn't work with `terra` unfortunately. So the next question, how onerous is it to convert a `terra` raster to a `raster` raster?

```
library(tmap)

r_raster <- raster::raster(r)

tm_shape(r_raster) +
  tm_raster()
```



The multi-tool function `raster()` does the job, so I'll take that for now.

Summary

`terra` looks set to replace `raster`. It is faster and just as easy to use as `raster`. Making the switch to `terra` isn't as hard as it may seem, its use will seem very familiar to `raster` users.