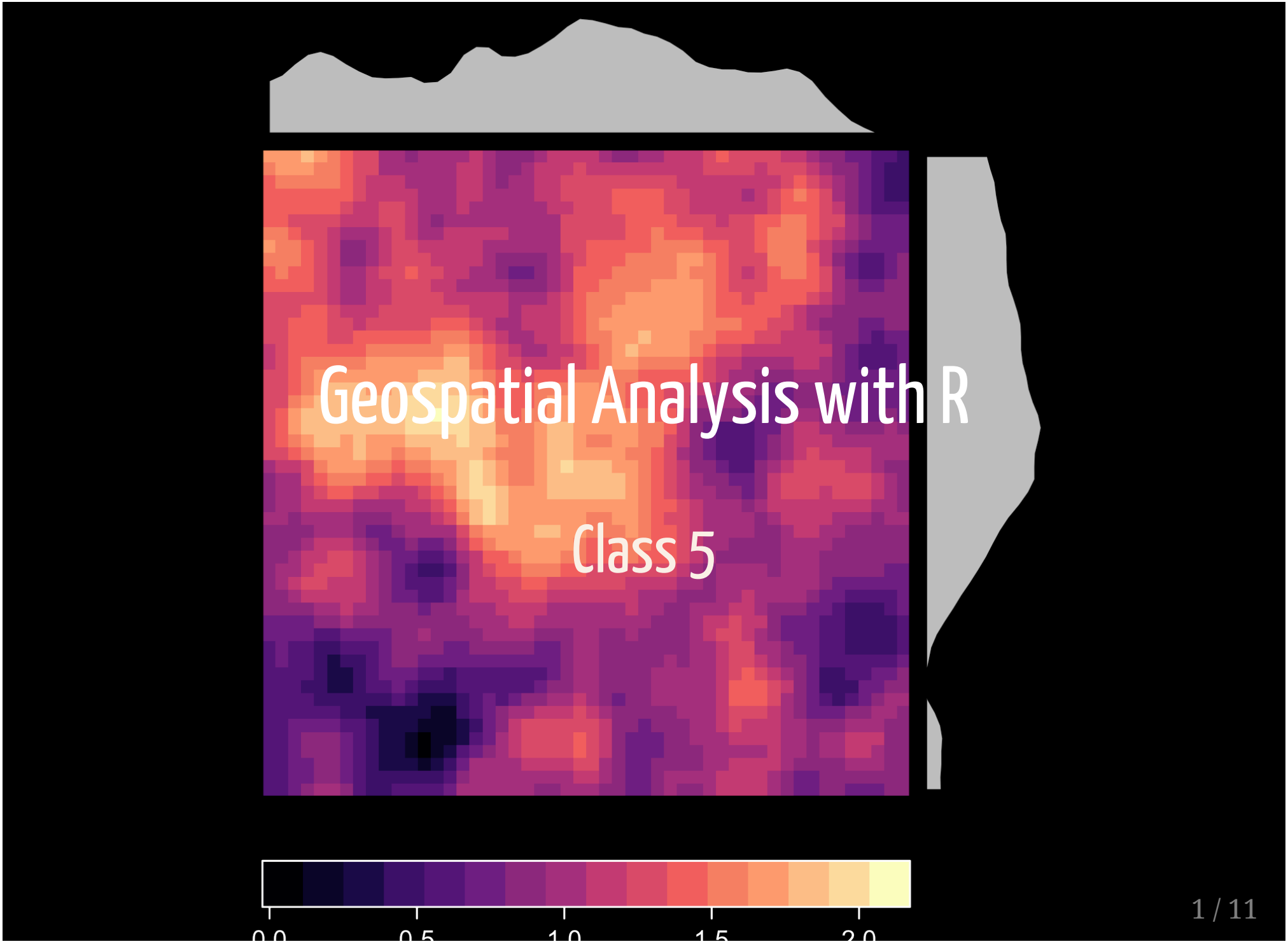
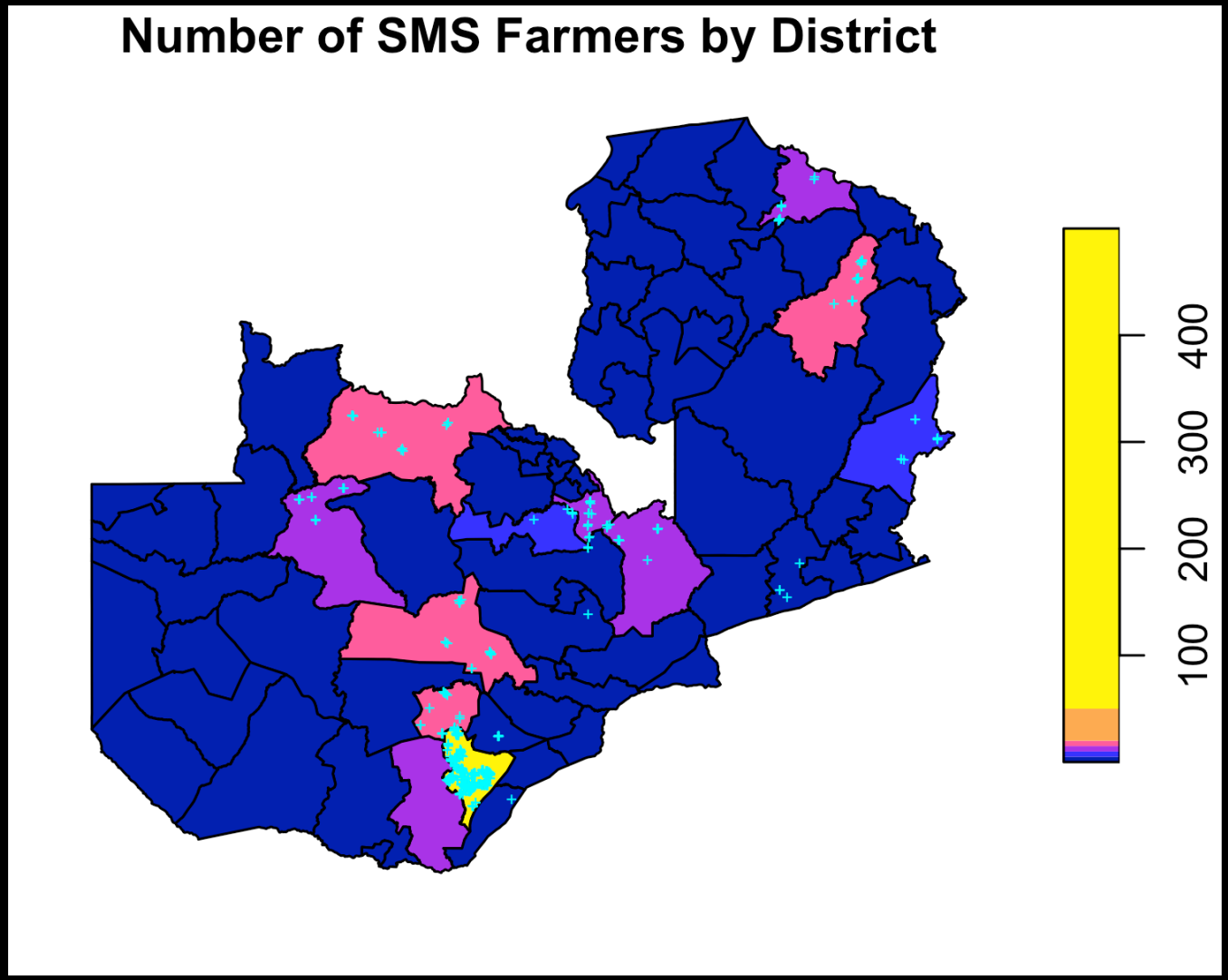

Geospatial Analysis with R
Class 5
0.0
0.5
1.0
1.5
2.0


Number of SMS Farmers by District
100
200
300
400

```r
library(sf)
library(dplyr)
f <- system.file("extdata", "farmer_spatial.csv", package = "geospaar")
farmers <- readr::read_csv(f)
farmers <- farmers %>% select(uuid, lat, lon) %>% distinct() %>%
  st_as_sf(., coords = c("lon", "lat"))
f <- system.file("extdata", "districts.shp", package = "geospaar")
dists <- read_sf(f)
st_crs(farmers) <- st_crs(dists)

# join farmers with districts (h/t https://mattherman.info/blog/point-in-poly/)
farmers_in_dists <- st_join(farmers, dists, join = st_within) %>%
  tidyr::drop_na()
farmer_count <- count(as_tibble(farmers_in_dists), distName)
dists_w_farmers <- left_join(dists, farmer_count) %>%
  mutate(n = ifelse(is.na(n), 0, n))

png("inst/slides/figures/sms-farmers.png", height = 4, width = 5, res = 300,
    units = "in")
plot(dists_w_farmers["n"], breaks = c(0, 5, 10, 15, 20, 50, 500),
     reset = FALSE, main = "Number of SMS Farmers by District")
plot(farmers %>% filter(uuid %in% unique(farmers_in_dists$uuid)) %>%
       st_geometry(), add = TRUE, pch = "+", col = "cyan", cex = 0.5)
dev.off()
```

Today

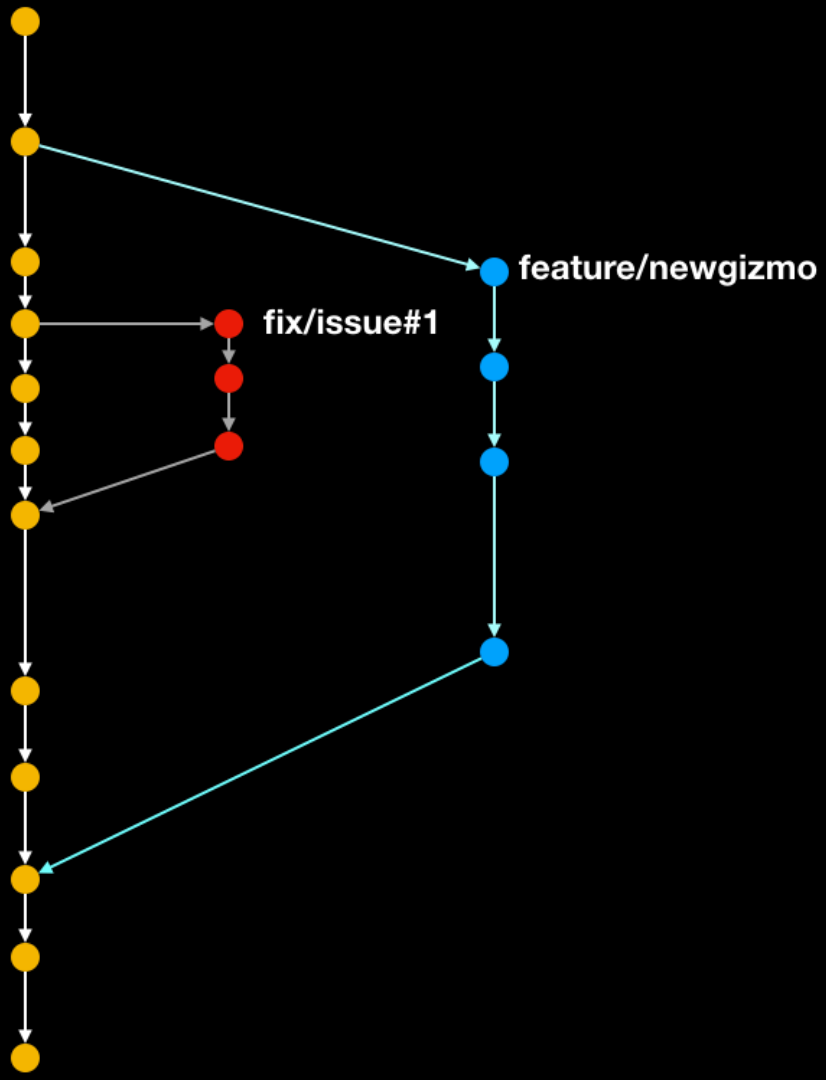
- `git` merging exercise
- The `R` ecosystem

What we should know by now

- Key concepts/tools of reproducibility and why we use them
- We should know:
  - How to set up R package project with `git` VCS
  - How to keep project synced between local and remote repos
  - How to document functions
  - Where your library lives
  - How package source differs from installed package
  - What the key ingredients in a package are
  - **data folder and lazy loads**
  - **inst folder and how to get at it**

Git/GitHub practical

Merging



The R Ecosystem

- 1.1 A taxonomy of R
  - 1.1.1 Species (data types)
  - 1.1.2 Genus (data structures and functions)
    - 1.1.2.1 Data structures
      - 1.1.2.1.1 One dimensional
      - 1.1.2.1.2 Two or more dimensions
    - 1.1.2.2 Functions
      - 1.1.2.2.1 Primitives
      - 1.1.2.2.2 Operators
      - 1.1.2.2.3 Control structures
      - 1.1.2.2.4 Base, package, and user-defined functions
      - 1.1.2.2.5 Generic functions
  - 1.1.3 Family (classes)
    - 1.1.3.1 OOP
- 1.2 Environments
  - 1.2.1 The global environment
  - 1.2.2 The package environment and namespaces
  - 1.2.3 The function environment
    - 1.2.3.1 Question to answer

OO

Base system?

```
x <- 1:10
!is.object(x)  # if TRUE, base object.
```

```
## [1] TRUE
```

S3 system?

```
x <- lm(x ~ rev(x))
!is.object(x)  # if TRUE, base object.
```

```
## [1] FALSE
```

```
!isS4(x)  # it's S3
```

```
## [1] TRUE
```

OO

S4 system?

```
x <- raster::raster(nrow = 10, ncol = 10)
!is.object(x)  # if TRUE, base object.
```

```
## [1] FALSE
```

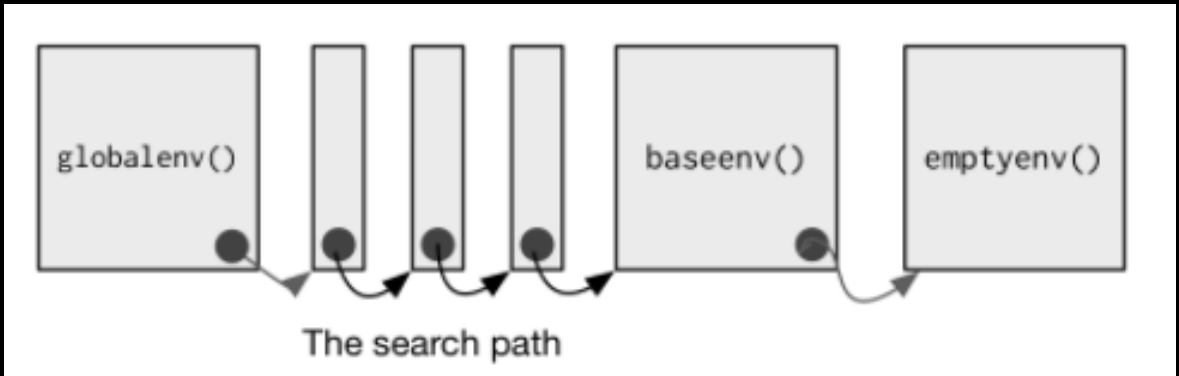
```
!isS4(x)  # it's S3
```

```
## [1] FALSE
```

```
!is(x, "refClass") # it's S4; otherwise it's RC.
```

```
## [1] TRUE
```

Environments



http://adv-r.had.co.nz/Environments.html