Working with Geospatial Data in R

# Introducing sp objects

# Data frames aren't a great way to store spatial data

```
> head(ward_sales)
  ward       lon      lat group order num_sales avg_price
1    1 -123.3128 44.56531   0.1     1       159  311626.9
2    1 -123.3122 44.56531   0.1     2       159  311626.9
3    1 -123.3121 44.56531   0.1     3       159  311626.9 ...
4    1 -123.3119 44.56531   0.1     4       159  311626.9
5    1 -123.3119 44.56485   0.1     5       159  311626.9
6    1 -123.3119 44.56430   0.1     6       159  311626.9

> nrow(ward_sales)
[1] 4189
```

- No easy way to keep coordinate reference system information

# Data frames aren't a great way to store spatial data

```
> head(ward_sales)
  ward       lon      lat group order num_sales avg_price
1    1 -123.3128 44.56531   0.1     1       159  311626.9
2    1 -123.3122 44.56531   0.1     2       159  311626.9
3    1 -123.3121 44.56531   0.1     3       159  311626.9 ...
4    1 -123.3119 44.56531   0.1     4       159  311626.9
5    1 -123.3119 44.56485   0.1     5       159  311626.9
6    1 -123.3119 44.56430   0.1     6       159  311626.9

> nrow(ward_sales)
[1] 4189
```

- Inefficient for complicated spatial objects

# Data frames aren't a great way to store spatial data

```
> head(ward_sales)
  ward       lon      lat group order num_sales avg_price
1    1 -123.3128 44.56531   0.1     1       159  311626.9
2    1 -123.3122 44.56531   0.1     2       159  311626.9
3    1 -123.3121 44.56531   0.1     3       159  311626.9 ...
4    1 -123.3119 44.56531   0.1     4       159  311626.9
5    1 -123.3119 44.56485   0.1     5       159  311626.9
6    1 -123.3119 44.56430   0.1     6       159  311626.9

> nrow(ward_sales)
[1] 4189
```

- Hierarchical structure gets forced into a flat structure

# The sp package:

- provides classes for storing different types of spatial data
- provides methods for spatial objects, for manipulation
- is useful for point, line and polygon data
- is a standard, so new spatial packages expect data in an sp object

Working with Geospatial Data in R

# Let’s practice!

Working with Geospatial Data in R

# sp and S4

# Two types of sp object

```
> summary(countries_sp)
Object of class SpatialPolygons
Coordinates:
   min       max
x -180 180.00000
y  -90  83.64513
Is projected: FALSE
proj4string :
[+proj=longlat +datum=WGS84
  +no_defs +ellps=WGS84
  +towgs84=0,0,0]
```

```
> summary(countries_spdf)
Object of class SpatialPolygonsDataFrame
Coordinates:
   min       max
x -180 180.00000
y  -90  83.64513
Is projected: FALSE
proj4string :
[+proj=longlat +datum=WGS84
  +no_defs +ellps=WGS84
  +towgs84=0,0,0]
Data attributes:
     name                iso_a3
 Length:177          Length:177          ...
 Class :character    Class :character
 Mode  :character    Mode  :character
```

# Two types of sp object

```
> summary(countries_sp)
Object of class SpatialPolygons
Coordinates:
   min       max
x -180 180.00000
y  -90  83.64513
Is projected: FALSE
proj4string :
[+proj=longlat +datum=WGS84
  +no_defs +ellps=WGS84
  +towgs84=0,0,0]
```

```
> summary(countries_spdf)
Object of class SpatialPolygonsDataFrame
Coordinates:
   min       max
x -180 180.00000
y  -90  83.64513
Is projected: FALSE
proj4string :
[+proj=longlat +datum=WGS84
  +no_defs +ellps=WGS84
  +towgs84=0,0,0]
Data attributes:
     name                iso_a3
 Length:177          Length:177          ...
 Class :character    Class :character
 Mode  :character    Mode  :character
```

# SpatialPolygons object

```
> str(countries_sp, max.level = 2)
Formal class 'SpatialPolygons' [package "sp"] with 4 slots
  ..@ polygons   :List of 177
  .. .. [list output truncated]
  ..@ plotOrder  : int [1:177] 7 136 28 169 31 23 9 66 84 5 ...
  ..@ bbox       : num [1:2, 1:2] -180 -90 180 83.6
  .. ..- attr(*, "dimnames")=List of 2
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
```

# SpatialPolygonsDataframe object

```
> str(countries_spdf, max.level = 2)
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots
  ..@ data       :'data.frame': 177 obs. of  6 variables:
  ..@ polygons   :List of 177
  .. .. [list output truncated]
  ..@ plotOrder  : int [1:177] 7 136 28 169 31 23 9 66 84 5 ...
  ..@ bbox       : num [1:2, 1:2] -180 -90 180 83.6
  .. ..- attr(*, "dimnames")=List of 2
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
```

?

# S4

- One of R's object oriented (OO) systems
- Key OO concepts
  - **class**: defines a type of object, their attributes and their relationship to other classes.
  - **methods**: functions, behavior depends on class of input
- S4 objects can have a recursive structure, elements are called **slots**
- http://adv-r.had.co.nz/OO-essentials.html#s4

# Accessing slots

```
> # 1. Use a dedicated method
> proj4string(countries_sp)
[1] "+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"

> # 2. Use the @ followed by unquoted slot name
> countries_sp@proj4string
CRS arguments:
 +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0

> # 3. Use slot() with quoted slot name
> slot(countries_sp, "proj4string")
CRS arguments:
 +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
```

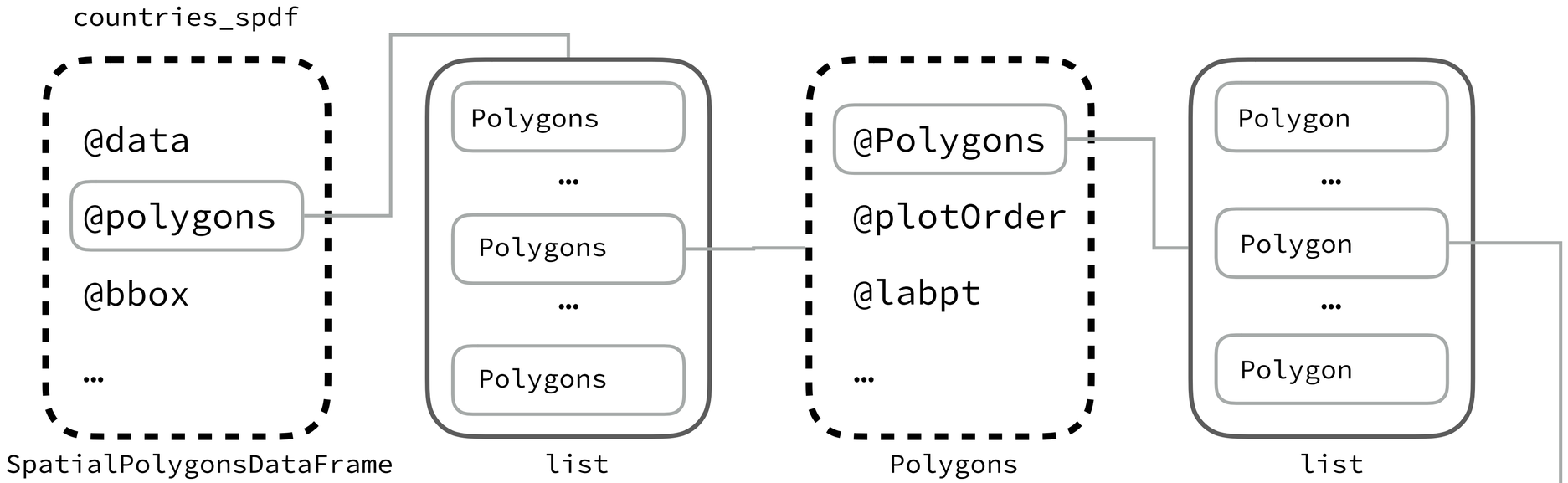
Working with Geospatial Data in R

# Let’s practice!

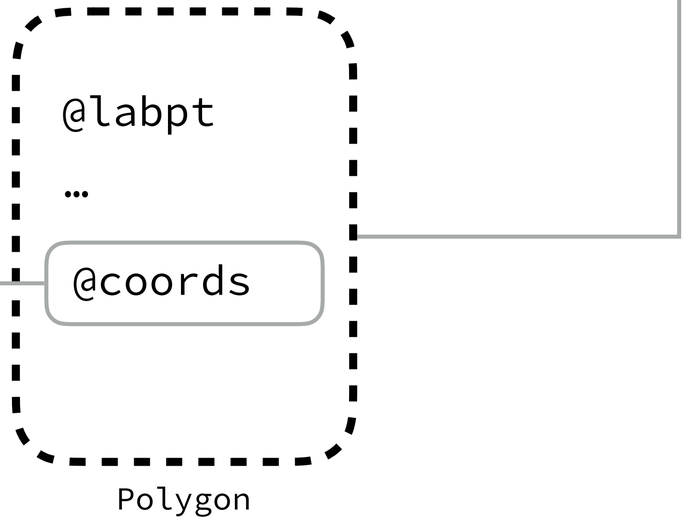
Working with Geospatial Data in R

# More sp classes and methods

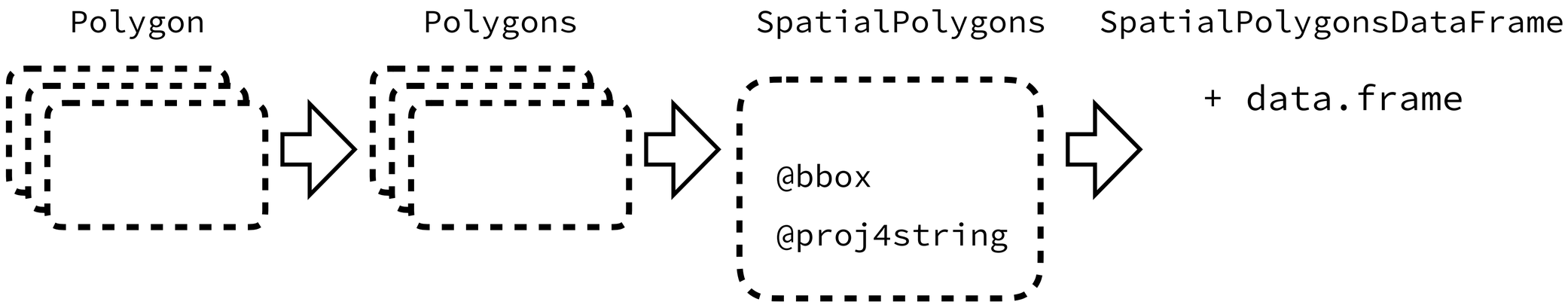
# Hierarchy of SpatialPolygonsDataFrame



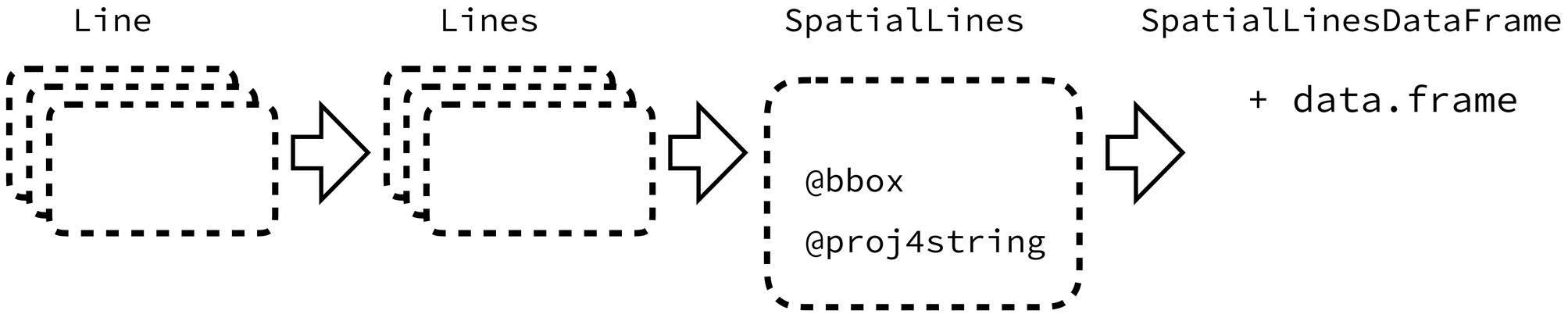

```
             [,1]     [,2]
[1,]  -94.81758 49.38905
[2,]  -94.64000 48.84000
[3,]  -94.32914 48.67074
[4,]  -93.63087 48.60926
[5,]  -92.61000 48.45000
```



# Other sp classes

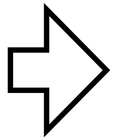


# Other sp classes



# Other sp classes

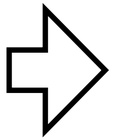
```
          [,1]     [,2]
[1,]  -94.81758 49.38905
[2,]  -94.64000 48.84000
[3,]  -94.32914 48.67074
[4,]  -93.63087 48.60926
[5,]  -92.61000 48.45000
```



SpatialPoints

@bbox

@proj4string



SpatialPointsDataFrame

\+ data.frame

# Subsetting sp objects

```
> # Subset by index
> str(countries_spdf[1, ], max.level = 2)
Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5
slots
  ..@ data       :'data.frame': 1 obs. of  6 variables:
  ..@ polygons   :List of 1
  ..@ plotOrder  : int 1
  ..@ bbox       : num [1:2, 1:2] 60.5 29.3 75.2 38.5
  .. ..- attr(*, "dimnames")=List of 2
  ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot
```

Working with Geospatial Data in R

# Let’s practice!

Working with Geospatial Data in R

# Introduction to `tmap`

# `tmap` displays spatial data

- Similar philosophy to `ggplot2`:
  - a plot is built up in layers
  - `ggplot2` expects data in data frames,
    `tmap` expects data in spatial objects
  - layers consist of a type of graphical representation and mapping from visual properties to variables

# Building plot in layers

```
> library(tmap)
> data(Europe)

> tm_shape(Europe) +
    tm_borders()
```

A SpatialPolygonsDataFrame

Specify spatial data

Add a layer to the plot

# Building plot in layers

```
> library(tmap)
> data(Europe)

> tm_shape(Europe) +
    tm_borders() +
    tm_fill(col = "part") +
    tm_compass() +
    tmap_style("cobalt")
```

# Building plot in layers

```
> library(tmap)
> data(Europe)

> tm_shape(Europe) +
    tm_borders() +
    tm_fill(col = "part") +
    tm_compass() +
    tmap_style("cobalt")
```

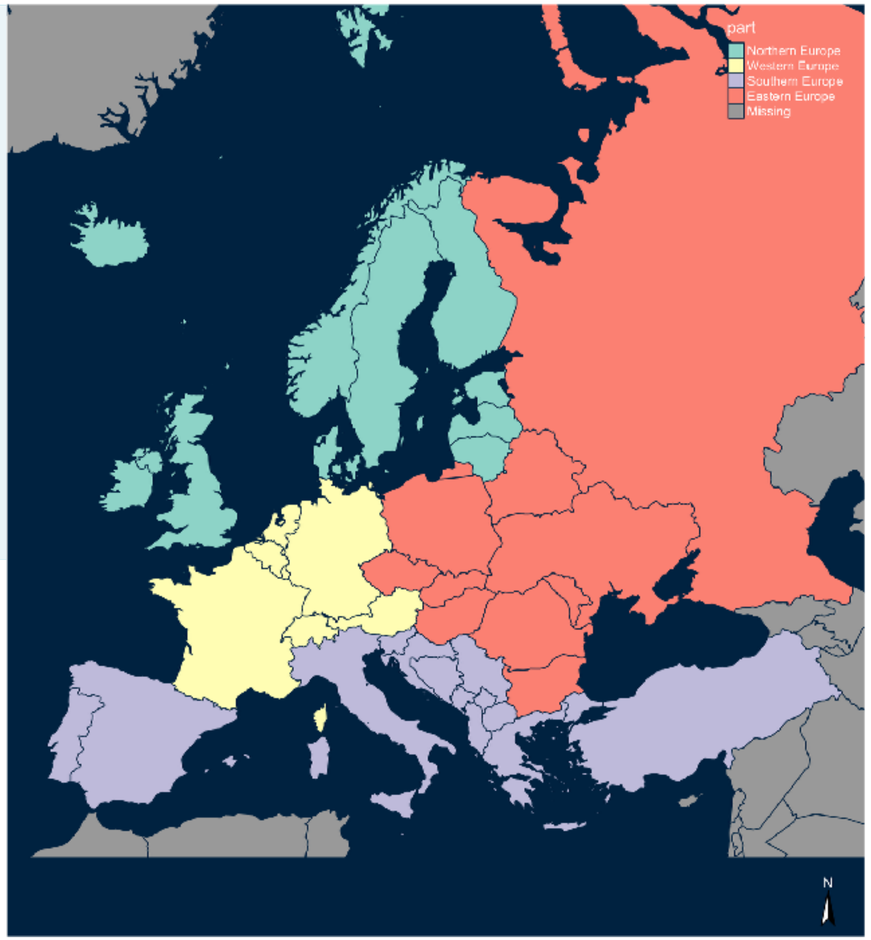
### Adding another data layer

```
tm_fill()
tm_borders()
tm_polygons()
tm_bubbles()
tm_dots()
tm_lines()
tm_raster()
tm_text()
```

# Building plot in layers

```
> library(tmap)
> data(Europe)

> tm_shape(Europe) +
    tm_borders() +
    tm_fill(col = "part") +
    tm_compass() +
    tmap_style("cobalt")
```



# Key differences to ggplot2

- No `scale_` equivalents, tweaks to scales happen in relevant layer call
- `tm_shape()` defines default data for any subsequent layers, you can have many in a single plot
- No need for `x` and `y` aesthetics, these are inherent in spatial objects
- No special evaluation, when mapping variables they must be quoted

Working with Geospatial Data in R

# Let’s practice!