I/O in R and Variable Binding

“<-” and “=”

When binding a variable in R we using “<-”.

```r
x <- 2
y <- 2
x + y
```

```
## [1] 4
```

“<-” and “=”

“<-” works both ways, but the reverse usage is uncommon and confusing.

```
2 -> x
2 -> y
x + y
```

```
## [1] 4
```

“<-” and “=”

“=” is typically used when assigning arguments in a function.

```
fun <- function(x, y){
    x + y
}
fun(x = 2, y = 2)
```

```
## [1] 4
```

Basic I/O

I/O in R varies depending of the data.

R can connect to nearly any data source you can think of.

```
myData <- data.frame(x = round(runif(1000, 1, 100), 2),
                     y = round(runif(1000, 1, 100), 2),
                     attr = sample(letters, 1000, replace = T))
head(myData, 8)
```

```
##       x     y attr
## 1 27.29 53.55    w
## 2 37.84 68.80    z
## 3 57.71 38.95    w
## 4 90.91 95.54    l
## 5 20.97 12.72    e
## 6 89.94  4.87    c
## 7 94.52 50.95    p
## 8 66.42 58.27    b
```

Basic I/O

read.csv and write.csv are common and are fine for most tasks.

```
write.csv(myData, "./data/myData.csv", row.names = F)
myData <- read.csv("./data/myData.csv")
head(myData, 8)
```

```
##       x     y attr
## 1 27.29 53.55    w
## 2 37.84 68.80    z
## 3 57.71 38.95    w
## 4 90.91 95.54    l
## 5 20.97 12.72    e
## 6 89.94  4.87    c
## 7 94.52 50.95    p
## 8 66.42 58.27    b
```

Excel with `xlsx()`

`xlsx()` is great for basic reading and writing to Excel.

```
library(xlsx)

write.xlsx(myData, "./data/myData.xls")
read.xlsx("./data/myData.xls", 1)
```

```
##       NA.     x     y attr
## 1       1 27.29 53.55    w
## 2       2 37.84 68.80    z
## 3       3 57.71 38.95    w
## 4       4 90.91 95.54    l
## 5       5 20.97 12.72    e
## 6       6 89.94  4.87    c
## 7       7 94.52 50.95    p
## 8       8 66.42 58.27    b
## 9       9 63.28 84.09    n
## 10     10  7.12 65.79    m
## 11     11 21.39 94.51    b
## 12     12 18.48 51.65    p
```

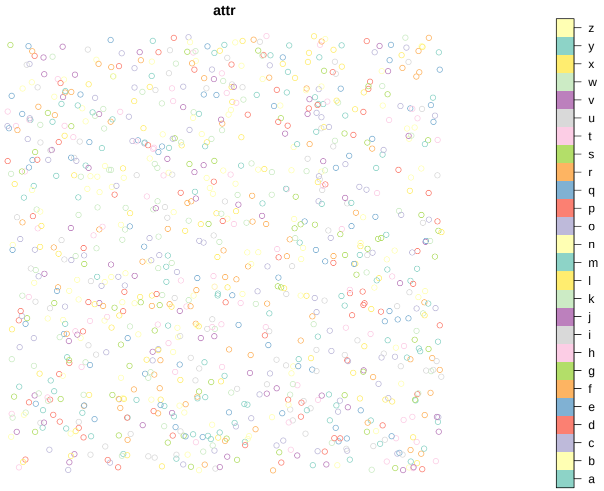
Spatial Vector Data

```r
library(sf, quietly = T)
myData   <- read.csv("./data/myData.csv")
myDataSf <- st_as_sf(myData, coords = c("x", "y"))
head(myDataSf)
```

```
## Simple feature collection with 6 features and 1 field
## geometry type:  POINT
## dimension:      XY
## bbox:           xmin: 20.97 ymin: 4.87 xmax: 90.91 ymax: 95.54
## epsg (SRID):    NA
## proj4string:    NA
##   attr            geometry
## 1    w POINT (27.29 53.55)
## 2    z  POINT (37.84 68.8)
## 3    w POINT (57.71 38.95)
## 4    l POINT (90.91 95.54)
## 5    e POINT (20.97 12.72)
## 6    c  POINT (89.94 4.87)
```

Spatial Vector Data

```r
plot(myDataSf)
```

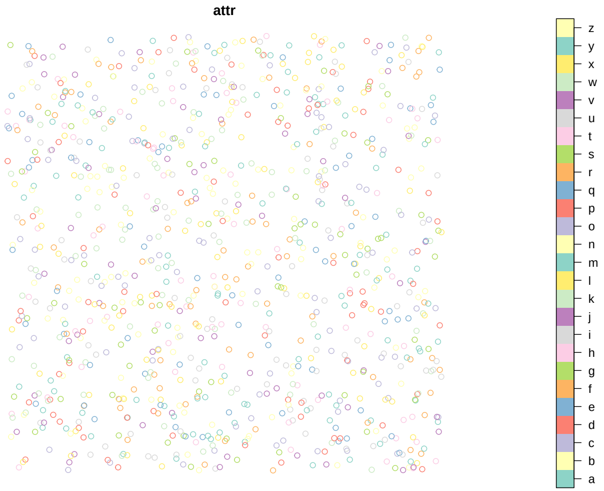


Spatial Vector Data

```r
write_sf(myDataSf, "./data/myDataSf.shp", driver = "ESRI Shapefile")
myDataSf <- read_sf("./data/myDataSf.shp")
```

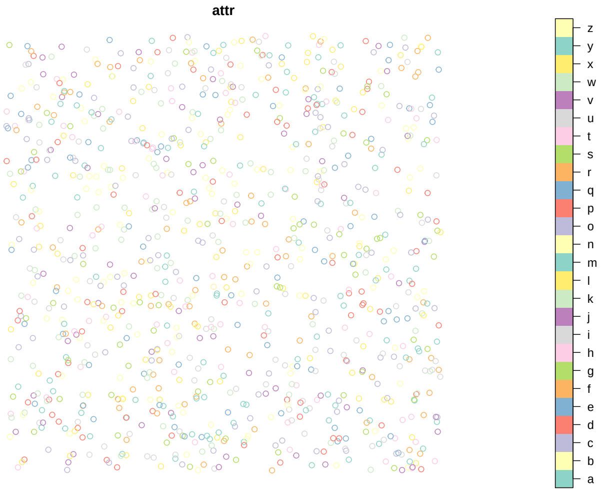
Spatial Vector Data

```r
plot(myDataSf)
```



Spatial Vector Data

```r
plot(myDataSf)
```

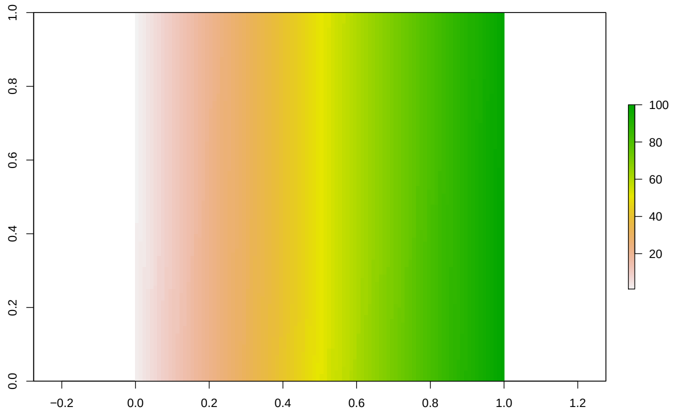


Spatial Raster Data

```r
library(raster, quietly = T)
myMatrix <- matrix(sort(round(runif(10000, 1, 100))), nrow = 100)
myRaster <- raster(myMatrix)
```

Spatial Raster Data

```r
plot(myRaster)
```

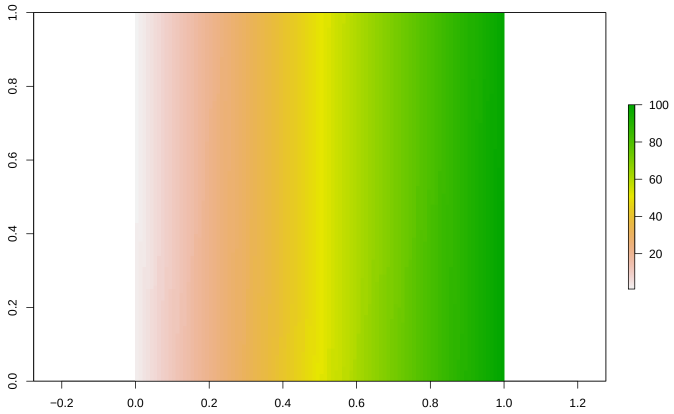


Spatial Raster Data

```
writeRaster(myRaster, "./data/myRaster.tif", overwrite = T)
myRaster <- raster("./data/myRaster.tif")
```

Spatial Raster Data

```
plot(myRaster)
```



Saving R Objects

saveRDS() and readRDS() are the preferred methods for saving R objects to disk when interoperability is not important.

```r
myRaster <- raster("./data/myRaster.tif")
saveRDS(myRaster, "./data/myRaster.rds")
```

Saving R Objects

```r
myRaster <- readRDS("./data/myRaster.rds")
class(myRaster)
```

```
## [1] "RasterLayer"
## attr(,"package")
## [1] "raster"
```