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.

```r
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.

```r
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 71.58 94.40    c
## 2 73.53 75.44    o
## 3 26.00 57.22    d
## 4 99.54 46.35    w
## 5 89.61 52.52    t
## 6 87.55 82.54    z
## 7 18.71  7.24    w
## 8 68.46 30.28    n
```

Basic I/O

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

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

```
##       x     y attr
## 1 71.58 94.40    c
## 2 73.53 75.44    o
## 3 26.00 57.22    d
## 4 99.54 46.35    w
## 5 89.61 52.52    t
## 6 87.55 82.54    z
## 7 18.71  7.24    w
## 8 68.46 30.28    n
```

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 71.58 94.40    c
## 2        2 73.53 75.44    o
## 3        3 26.00 57.22    d
## 4        4 99.54 46.35    w
## 5        5 89.61 52.52    t
## 6        6 87.55 82.54    z
## 7        7 18.71  7.24    w
## 8        8 68.46 30.28    n
## 9        9 99.60 42.60    u
## 10      10 58.31 95.35    s
## 11      11 12.64  3.44    s
## 12      12 96.57 31.99    r
```

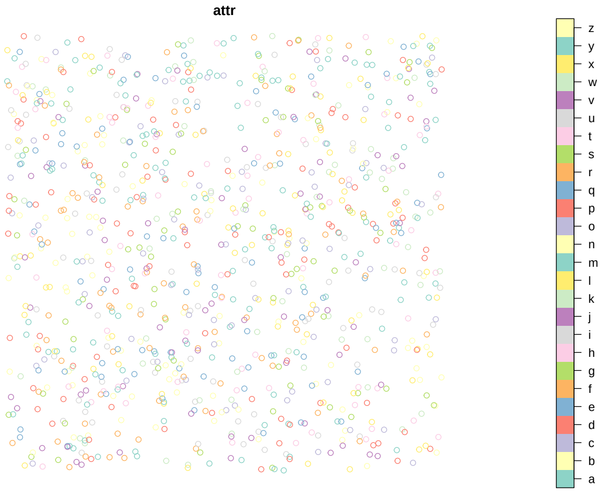
Spatial Vector Data

```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: 26 ymin: 46.35 xmax: 99.54 ymax: 94.4
## epsg (SRID):    NA
## proj4string:    NA
##   attr             geometry
## 1    c  POINT (71.58 94.4)
## 2    o POINT (73.53 75.44)
## 3    d    POINT (26 57.22)
## 4    w POINT (99.54 46.35)
## 5    t POINT (89.61 52.52)
## 6    z POINT (87.55 82.54)
```

Spatial Vector Data

```r
plot(myDataSf)
```

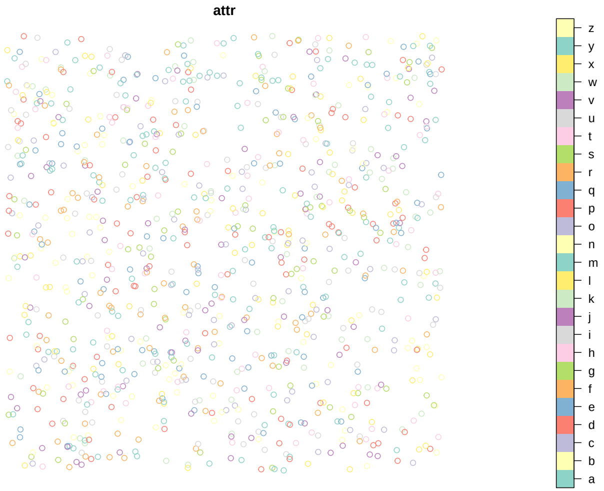


Spatial Vector Data

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

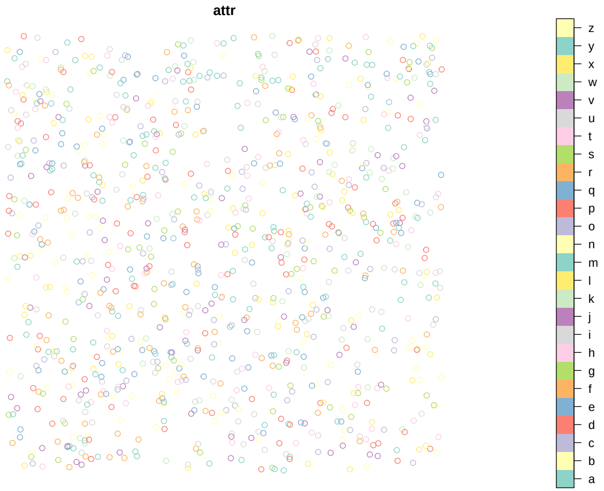
Spatial Vector Data

```r
plot(myDataSf)
```



Spatial Vector Data

```r
plot(myDataSf)
```

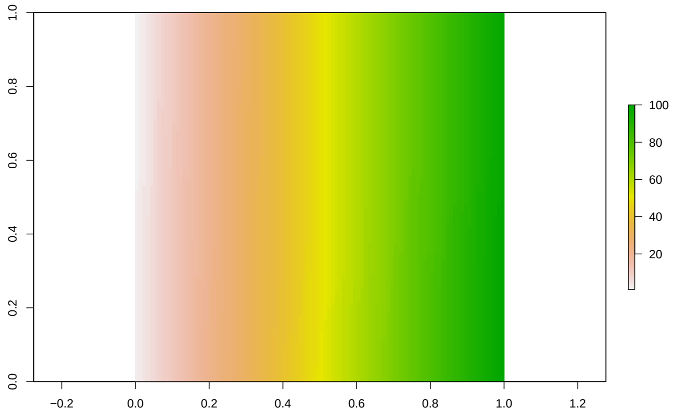


Spatial Raster Data

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

Spatial Raster Data

```
plot(myRaster)
```

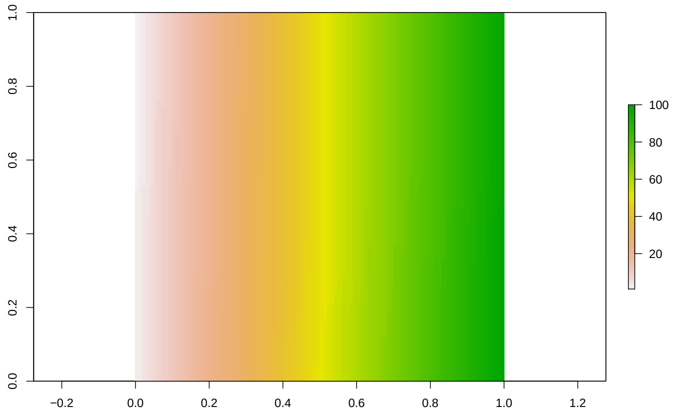


Spatial Raster Data

```r
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"
```