I/O in R and Variable Binding

"<-" and "="

When binding a variable in R we using "<-".

```
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 78.53 26.78    l
## 2 32.99 38.97    s
## 3 42.58 63.86    g
## 4 32.50 79.32    b
## 5 87.69  1.75    a
## 6 33.06 32.18    y
## 7 24.52 17.51    t
## 8 94.25 76.93    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 78.53 26.78    l
## 2 32.99 38.97    s
## 3 42.58 63.86    g
## 4 32.50 79.32    b
## 5 87.69  1.75    a
## 6 33.06 32.18    y
## 7 24.52 17.51    t
## 8 94.25 76.93    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 78.53 26.78    l
## 2        2 32.99 38.97    s
## 3        3 42.58 63.86    g
## 4        4 32.50 79.32    b
## 5        5 87.69  1.75    a
## 6        6 33.06 32.18    y
## 7        7 24.52 17.51    t
## 8        8 94.25 76.93    b
## 9        9 84.85 95.84    b
## 10      10 30.89 32.42    q
## 11      11 55.85  6.53    x
## 12      12 40.41 94.85    w
```

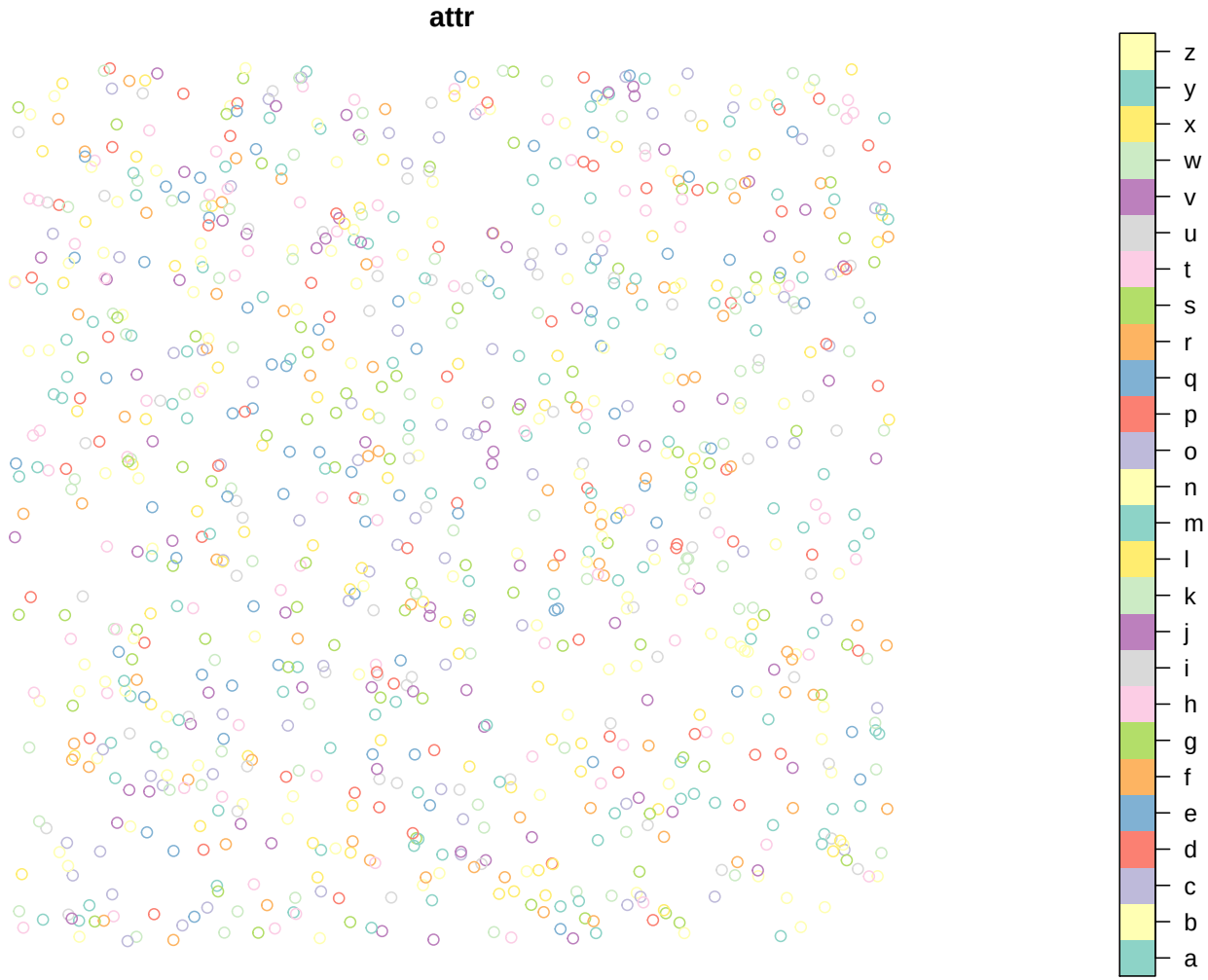
Spatial Vector Data

```
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: 32.5 ymin: 1.75 xmax: 87.69 ymax: 79.32
## epsg (SRID):    NA
## proj4string:    NA
##   attr            geometry
## 1    l POINT (78.53 26.78)
## 2    s POINT (32.99 38.97)
## 3    g POINT (42.58 63.86)
## 4    b  POINT (32.5 79.32)
## 5    a  POINT (87.69 1.75)
## 6    y POINT (33.06 32.18)
```

Spatial Vector Data

```
plot(myDataSf)
```

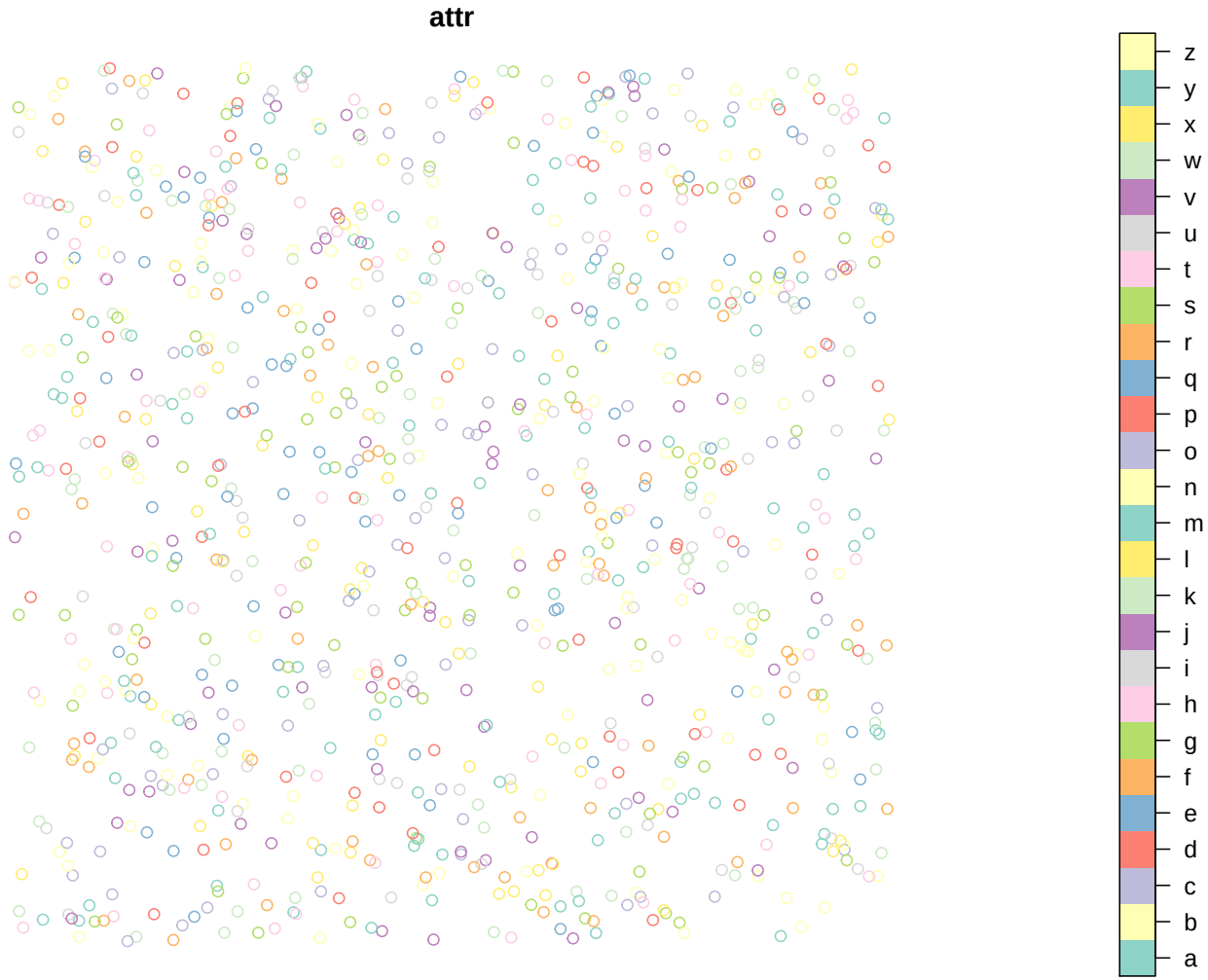


Spatial Vector Data

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

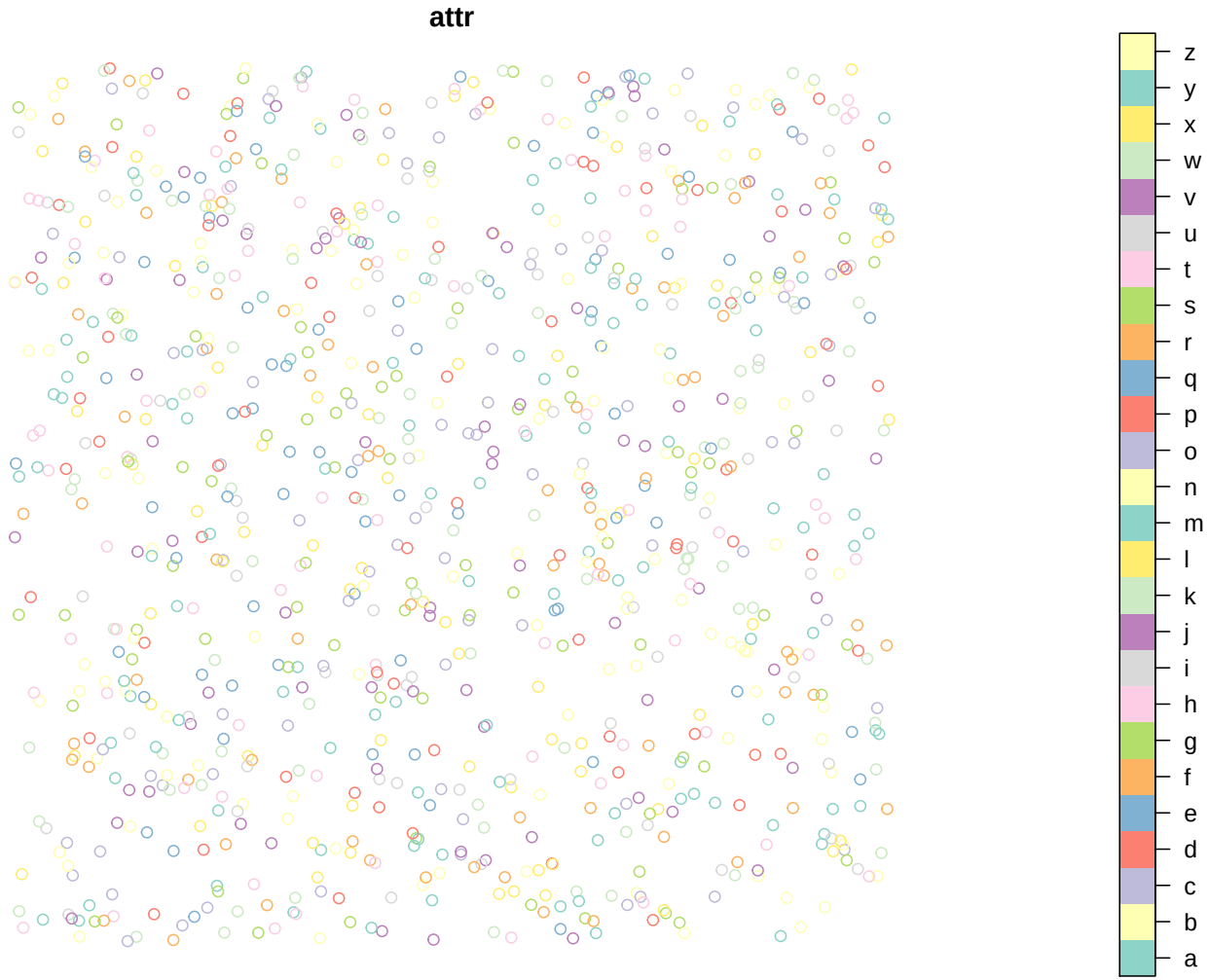
Spatial Vector Data

```
plot(myDataSf)
```



Spatial Vector Data

```
plot(myDataSf)
```

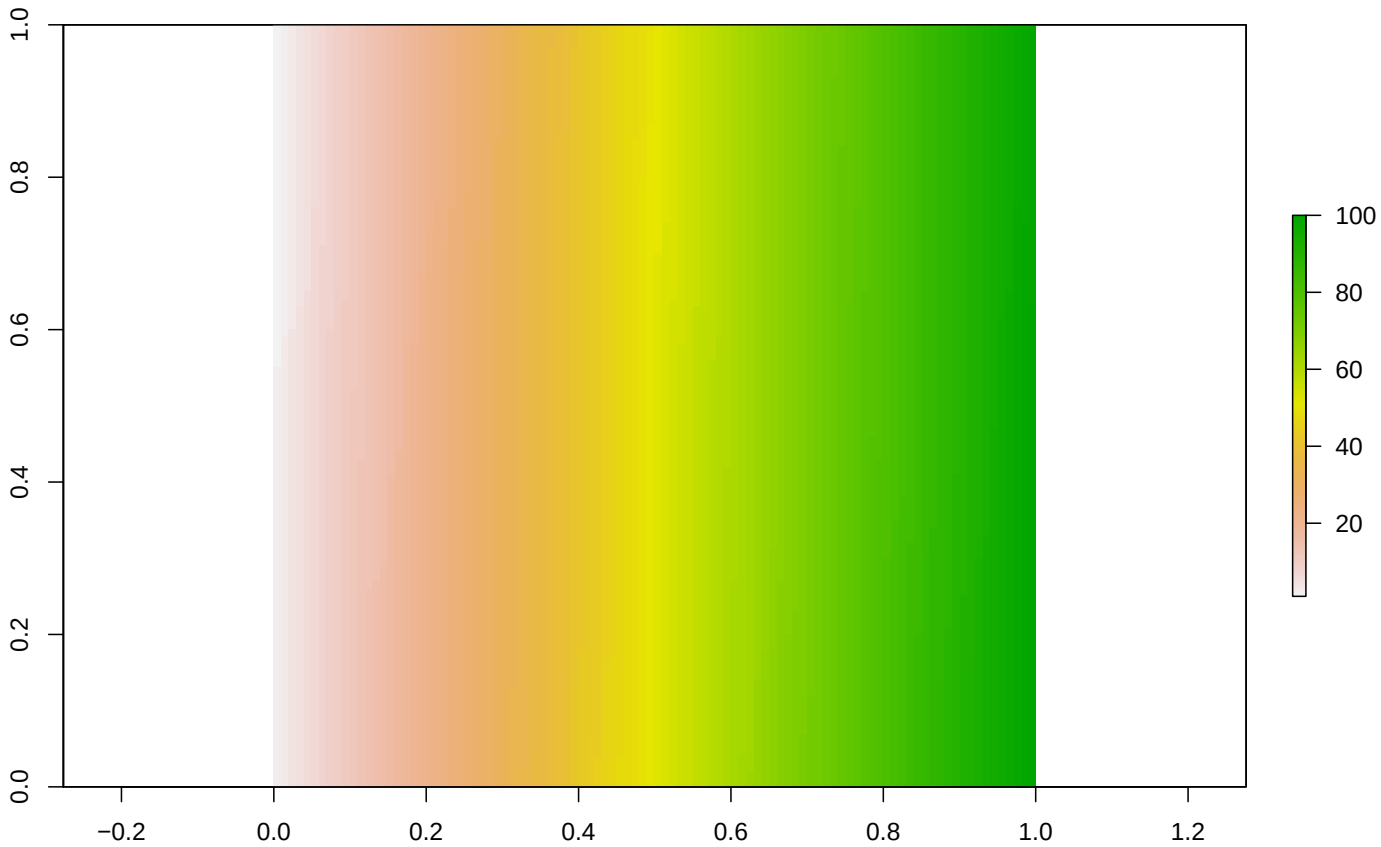


Spatial Raster Data

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

Spatial Raster Data

```
plot(myRaster)
```

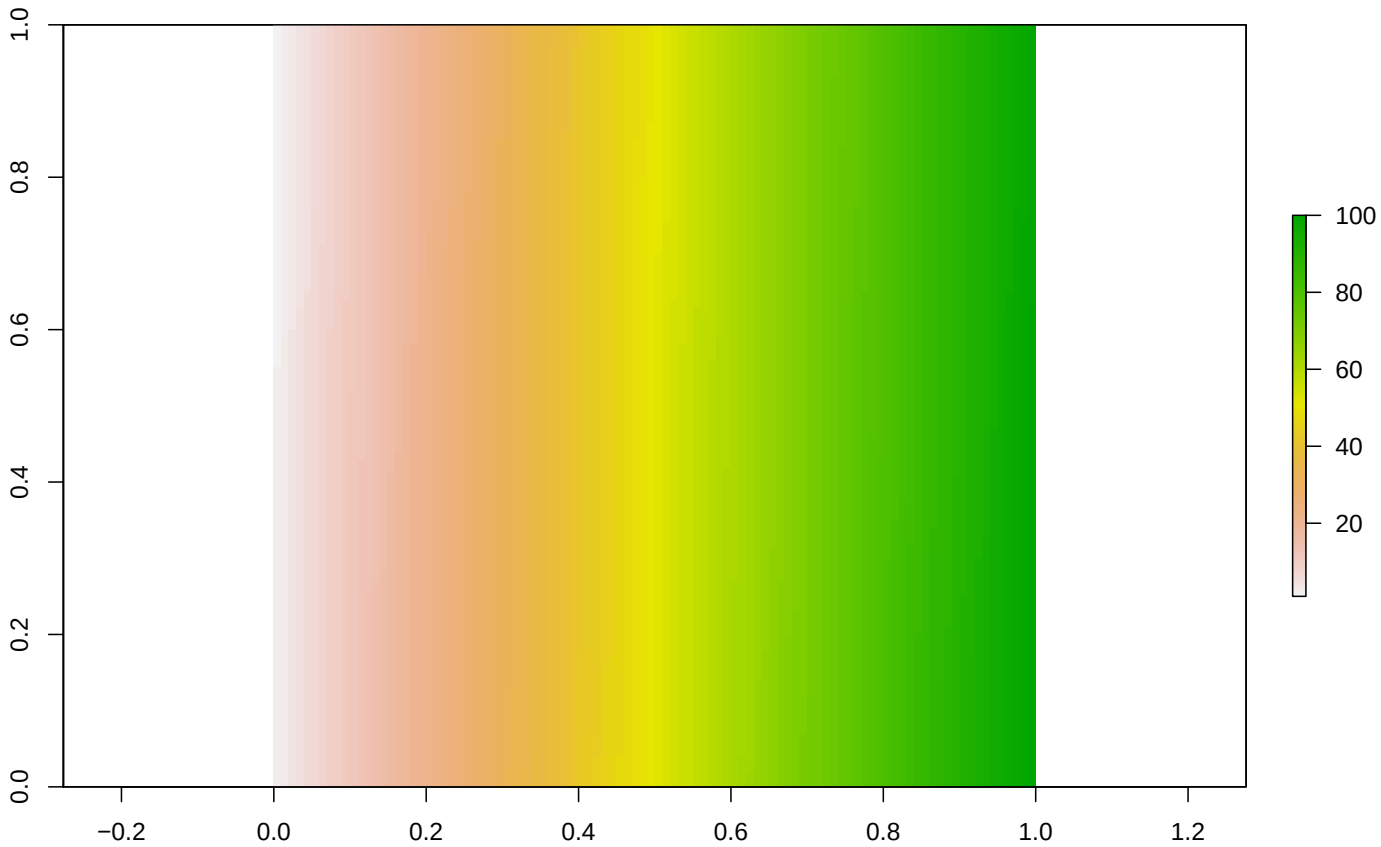


Spatial Raster Data

```
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.

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

Saving R Objects

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

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