

I/O in R and Variable Binding

“<-” and “=”

When binding a variable in R use “<-”.

```
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, "./slides_rmd/data/myData.csv", row.names = F)
myData <- read.csv("./slides_rmd/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, quietly = T)
write.xlsx(myData, "./slides_rmd/data/myData.xls", row.names = F)
read.xlsx("./slides_rmd/data/myData.xls", 1)
```

##		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
## 9		63.28	84.09	n
## 10		7.12	65.79	m
## 11		21.39	94.51	b
## 12		18.48	51.65	p
## 13		69.02	9.41	u

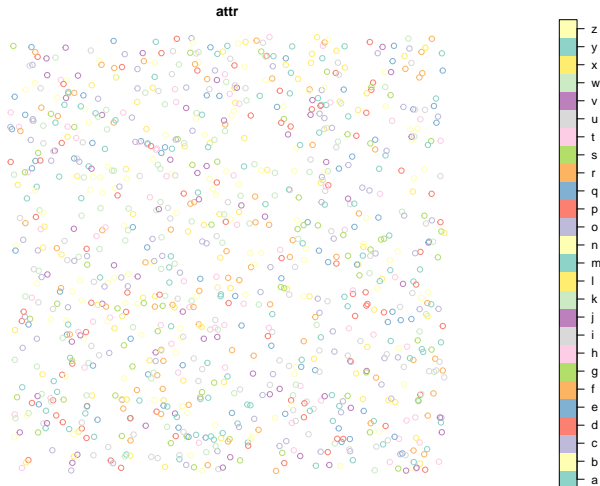
Spatial Vector Data

```
library(sf, quietly = T)
myData    <- read.csv("./slides_rmd/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

```
plot(myDataSf)
```

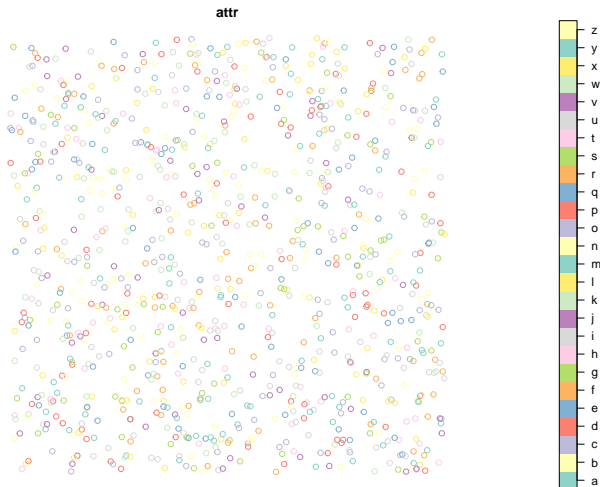


Spatial Vector Data

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

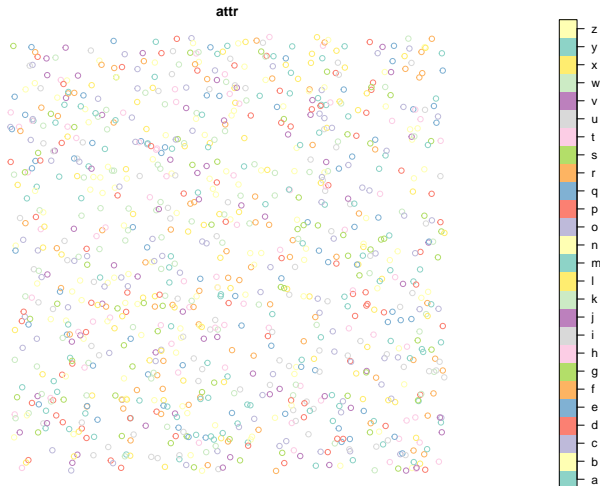
Spatial Vector Data

```
plot(myDataSf)
```



Spatial Vector Data

```
plot(myDataSf)
```

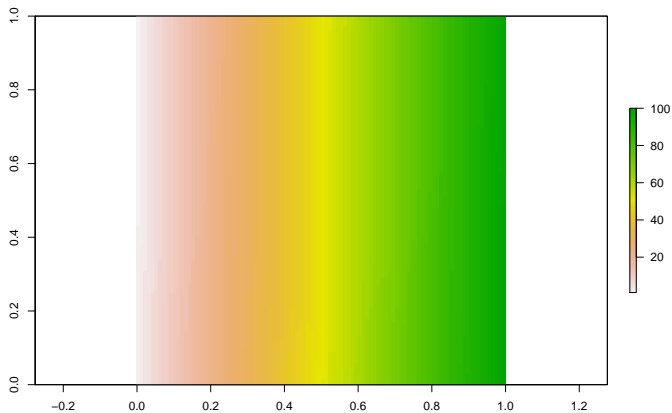


Spatial Raster Data

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

Spatial Raster Data

```
plot(myRaster)
```

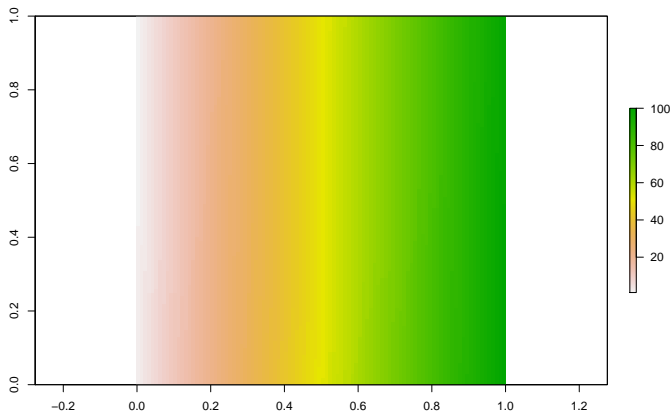


Spatial Raster Data

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

Saving R Objects

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

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