### R code from vignette source 'spatialdata.Rnw'

### Encoding: ISO8859-1

###################################################

### code chunk number 1: foo

###################################################

options(width = 60)

###################################################

### code chunk number 2: aa1

###################################################

v0 <- 7

v0

v1 <- c(1.25, 2.9, 3.0)

v1

# going from 5 to 7

v2 <- 5:7

v2

###################################################

### code chunk number 3: aa3

###################################################

x <- c('a', 'bc', 'def', 'gh', 'i')

x

class(x)

length(x)

###################################################

### code chunk number 4: a3

###################################################

f1 <- as.factor(x)

f1

f2 <- as.factor(5:7)

f2[1]

as.integer(f2)

###################################################

### code chunk number 5: a4

###################################################

fc2 <- as.character(f2)

as.integer(fc2)

###################################################

### code chunk number 6: a5

###################################################

# first element

v2[1]

# elements 2 to 3

v2[2:3]

# all elements but the first two

v2[-c(1:2)]

###################################################

### code chunk number 7: a6a

###################################################

# are the elements of v1 2?

v1 == 2

# are the elements of v1 larger than 2?

f <- v1 > 2

f

# element wise multiplication

v3 <- v1 \* v2

v3

# add all elements

sum(v3)

###################################################

### code chunk number 8: a6b

###################################################

a <- 1:6

b <- 1:2

a \* b

###################################################

### code chunk number 9: a7a

###################################################

# initialization of output variables

s <- 0

r <- vector(length=length(v1))

# i goes from 1 to 3 (the length of v1)

for (i in 1:length(v1)) {

s <- s + v1[i]

r[i] <- v1[i] \* v2[i]

}

s

r

# another example, with an if/else branch:

f <- vector(length=length(v1))

# i goes from 1 to 3 (the length of v1)

for (i in 1:length(v1)) {

if (v1[i] > 2) {

f[i] <- TRUE

} else {

f[i] <- FALSE

}

}

f

###################################################

### code chunk number 10: b1

###################################################

matrix(ncol=3, nrow=3)

###################################################

### code chunk number 11: b2

###################################################

matrix(1:6, ncol=3, nrow=2)

matrix(1:6, ncol=3, nrow=2, byrow=TRUE)

# the above can also be achieved using the transpose function

# (note the reversal of ncol and nrow valus)

t(matrix(1:6, ncol=2, nrow=3))

###################################################

### code chunk number 12: b3

###################################################

v1 <- c(1,2,3)

v2 <- 5:7

# column bind

m1 <- cbind(v1, v2)

m1

# row bind

m2 <- rbind(v1, v2, v1\*v2)

m2

m3 <- cbind(m1, m2)

# get the column names

colnames(m3)

# set the column names

colnames(m3) <- c('ID', 'V2', 'X', 'Y', 'Z')

m3

# dimensions of m3 (nrow, ncol))

dim(m3)

###################################################

### code chunk number 13: b4

###################################################

# one value

m3[2,2]

# equivalent to

m3[5]

# 2 columns and rows

m3[1:2,1:2]

# entire row

m3[2, ]

# entire column

m3[ ,2]

# you can also use column- or rownames for subsetting

m3[c('v1', 'v2') , c('ID', 'X')]

###################################################

### code chunk number 14: b5

###################################################

# sum values in each row

apply(m3, 1, sum)

# get mean for each column

apply(m3, 2, mean)

###################################################

### code chunk number 15: d1

###################################################

d <- data.frame(ID=as.integer(1:4), name=c('Ana', 'Rob', 'Liu', 'Veronica'),

sex=as.factor(c('F','M','M', 'F')), score=c(10.2, 9, 13.5, 18),

stringsAsFactors=FALSE)

d

class(d)

class(d$name)

sapply(d, class)

###################################################

### code chunk number 16: d2

###################################################

d$name

d[, 'name']

d[,2]

###################################################

### code chunk number 17: d3

###################################################

# tabulate single variable

table(d$sex)

# contingency table

table(d[ c('name', 'sex')])

# mean score by sex

tapply(d$score, d$sex, mean)

aggregate(d[, 'score', drop=F], d[, 'sex', drop=FALSE], mean)

###################################################

### code chunk number 18: d5a

###################################################

e <- list(d , m3, 'abc')

e

###################################################

### code chunk number 19: d5a

###################################################

e[2][1]

e[[2]][1]

###################################################

### code chunk number 20: d6

###################################################

lapply(e, NROW)

sapply(e, length)

###################################################

### code chunk number 21: d7

###################################################

sumsquare <- function(a, b) {

d <- a + b

dd <- d \* d

return(dd)

}

###################################################

### code chunk number 22: d8

###################################################

sumsquare(1,2)

x <- 1:3

y <- 3:5

sumsquare(x,y)

###################################################

### code chunk number 23: d9

###################################################

nun <- function(x)length(unique(x))

data <- c('a', 'b', 'a', 'c', 'b')

nun(data)

###################################################

### code chunk number 24: c1

###################################################

name <- toupper(letters[1:10])

longitude <- c(-116.7, -120.4, -116.7, -113.5, -115.5,

-120.8, -119.5, -113.7, -113.7, -110.7)

latitude <- c(45.3, 42.6, 38.9, 42.1, 35.7, 38.9,

36.2, 39, 41.6, 36.9)

precip <- (runif(length(latitude))\*10)^3

stations <- cbind(longitude, latitude)

# plot locations, with size (cex) proportional to precip

plot(stations, cex=1+precip/500, pch=20, col='red', main='Precipitation')

text(stations, name, pos=4)

# add a legend

breaks <- c(100, 500, 1000, 2000)

legend("topright", legend=breaks, pch=20, pt.cex=1+breaks/500, col='red', bg='gray')

###################################################

### code chunk number 25: c2

###################################################

lon <- c(-116.8, -114.2, -112.9, -111.9, -114.2, -115.4, -117.7)

lat <- c(41.3, 42.9, 42.4, 39.8, 37.6, 38.3, 37.6)

x <- cbind(lon, lat)

plot(stations)

polygon(x, col='blue', border='light blue')

lines(x, lwd=3, col='red')

points(x, cex=2, pch=20)

points(stations)

###################################################

### code chunk number 26: c3

###################################################

wst <- data.frame(longitude, latitude, name, precip)

wst

###################################################

### code chunk number 27: sp1

###################################################

longitude <- c(-116.7, -120.4, -116.7, -113.5, -115.5,

-120.8, -119.5, -113.7, -113.7, -110.7)

latitude <- c(45.3, 42.6, 38.9, 42.1, 35.7, 38.9,

36.2, 39, 41.6, 36.9)

lonlat <- cbind(longitude, latitude)

library(sp)

st1 <- SpatialPoints(lonlat)

df <- data.frame(ID=1:nrow(lonlat), precip=(latitude-30)^3)

st2 <- SpatialPointsDataFrame(st1, data=df)

class(st1)

class(st2)

st2

###################################################

### code chunk number 28: sp2

###################################################

lon <- c(-116.8, -114.2, -112.9, -111.9, -114.2, -115.4, -117.7)

lat <- c(41.3, 42.9, 42.4, 39.8, 37.6, 38.3, 37.6)

lonlat <- cbind(lon, lat)

# close the ring of the polygon

lonlat <- rbind(lonlat, lonlat[1,])

pols <- SpatialPolygons( list( Polygons(list(Polygon(lonlat)), 1)))

str(pols)

class(pols)

###################################################

### code chunk number 29: sp3

###################################################

plot(st2, axes=TRUE)

plot(pols, border='blue', col='yellow', lwd=3, add=TRUE)

points(st2, col='red', pch=20, cex=3)

###################################################

### code chunk number 30: sp4

###################################################

proj4string(pols)

###################################################

### code chunk number 31: sp5

###################################################

proj4string(pols) <- CRS("+proj=longlat +datum=WGS84")

###################################################

### code chunk number 32: sp6

###################################################

library(rgdal)

polsrob <- spTransform(pols, CRS("+proj=robin +datum=WGS84"))

proj4string(polsrob)

bbox(pols)

bbox(polsrob)

plot(polsrob, axes=TRUE)

###################################################

### code chunk number 33: r1

###################################################

library(raster)

# create empty RasterLayer

r <- raster(ncol=10, nrow=10, xmx=-80, xmn=-150, ymn=20, ymx=60)

r

# assign values

r[] <- 1:ncell(r)

r

# plot

plot(r)

# add polygon and points

plot(pols, border='blue', col='yellow', lwd=3, add=TRUE)

points(st2, col='red', pch=20, cex=3)

###################################################

### code chunk number 34: r2

###################################################

r2 <- r \* r

r3 <- sqrt(r)

s <- stack(r, r2, r3)

s

plot(s)