R lectures

17/04/2019

Repaso de la clase anterior

Vectorización

Se trata de operaciones que aplican a un vector, pero elemento por elemento.

Los pasos se simplifican al no pensar en los elementos del vector, si no en el vector en sí. Los bucles en una función vectorizada están hechos en C y no en R, y por lo tanto son mucho más rápidos.

```
# sin vectorización
for (i in 1:length(x)) z[i] <- x[i] + y[i]

# con vectorización
z <- x + y
```

- Ref: Vectorise from Advanced R

Funcionales - Familia *apply()

- **lapply()**: LLama a una función especificada en cada componente de una lista y devuelve otra lista.
- **sapply()**: En algunos casos, la lista que devuelve lapply() puede ser simplificada a un vector o a una matriz.
- **apply()**: es para matrices. Tienen la particularidad que podemos elegir aplicar una función a filas o a columnas.
- **tapply()**: para aplicar funciones a vectores con factores (o enteros que R interpreta como factores).

Combinan algo similar a vectorización con funciones. Suelen ser la respuesta a "cómo hago para procesar (por ejemplo transformar o extraer información de) cada elemento de este objeto?"

Ref: ver también el paquete plyr, esta web y este paper

Matemática

Funciones matemáticas básicas

```
- exp(): función exponencial, base e
- log(): logaritmo natural
- log10(): logaritmo base 10
- sqrt(): raiz cuadrada
- abs(): valor absoluto
- sin(), cos(), etc.: funciones trigonométricas
- min(),  max(): valor mínimo y máximo de un vector
- which.min() and which.max(): índice del valor mínimo y máximo
- pmin() and pmax(): mínimos y máximos para varios vectores, por elemento
- sum() and prod(): suma y producto de elementos de vectores
- cumsum() and cumprod(): suma acumulada y producto acumulado de elementos de vectores
- round(), floor(), and ceiling(): redondeo al entero más próximo, al menor o al mayor,
- factorial(): función factorial
- %%: operador módulo y %/%: operador división por enteros
```

```
x <- c(12,5,13)
cumsum(x)
# [1] 12 17 30
cumprod(x)
# [1] 12 60 780
```

Cálculo

R tiene capacidades tanto para hacer cálculos numéricos como analíticos:

```
# derivada analítica
D(expression(exp(x^2)),"x")
exp(x^2) * (2 * x)

# integral numérica
integrate(function(x) x^2,0,1)
0.3333333 with absolute error < 3.7e-15
```

Existen paquetes para ecuaciones diferenciales (`odesolve`), y para extender la capacidad de procesado simbólico usando el sistema Yacas (`ryacas`). Ver CRAN.

Álgebra Lineal

Multiplicación de matrices

Multiplicamos directamente con el operador %*%

```
a
     [,1] [,2]
[1,]    1    2
[2,]    3    4
b
     [,1] [,2]
[1,]    1   -1
[2,]    0    1
a %*% b
     [,1] [,2]
[1,]    1    1
[2,]    3    1
```

`solve()` para sistemas de ecuaciones lineales e invertir una matriz

```
a <- matrix(c(1,1,-1,1), nrow=2, ncol=2)
b <- c(2,4)

solve(a,b) ## resuelve el sistema de ecuaciones
[1] 3 1

solve(a) ## sin el segundo argumento, invierte la matriz
     [,1] [,2]
[1,]  0.5  0.5
[2,] -0.5  0.5
```

Autovectores y autovalores

```
a
     [,1] [,2]
[1,]    1   -1
[2,]    1    1

e <- eigen(a)
$values
[1] 1+1i 1-1i

$vectors
                      [,1]                  [,2]
[1,] 0.7071068+0.0000000i 0.7071068+0.0000000i
[2,] 0.0000000-0.7071068i 0.0000000+0.7071068i

eigenvector1 <- e$vectors[,1]
eigenvalue1 <-  e$values[1]

a %*% eigenvector1 / eigenvalue1
                      [,1]
[1,] 0.7071068+0.0000000i
[2,] 0.0000000-0.7071068i
```

Otras operaciones posibles son:

```
- t(): transpuesta de una matriz
- qr(): descomposición QR
- chol(): descompocisión Cholesky
- det(): Determinante
- eigen(): Autovalores/autovectores
- diag(): extrae la diagonal de una matriz cuadrada
- svd(A): descomposición en valores singulares
- ...
```

Estadística

Estadística descriptiva

Conceptos para entender la estructura de un grupo (digamos, `N > 30`) de datos.

Una manera de hacer estadística descriptiva en columnas de listas (o data.frames) es con `sapply`:

```
sapply(mydata, mean, na.rm=TRUE)
```

donde podemos usar `mean`, `sd`, `var`, `min`, `max`, `median`, `range`, o `quantile`, entre otras).

O `summary()`, que es equivalente y funciona con data.frames.

Tenemos también `group_by() + summarise()` con las mismas funciones básicas.

Distribuciones

En general R usa la siguiente convención:

- `dDIST(x, ...)` es la función distribución de probabilidad (PDF). Devuelve la prob. de observar un valor x
- `pDIST(x, ...)` es la función cumulativa de probabilidad (CDF). Devuelve la prob. de obervar un valor menor a x (mayor si usamos `lower.tail=F`)
- `rDIST(n, ...)` es un generador de números aleatorios que devuelve n valores sacados de una distr. DIST.
- `qDIST(p, ...)` es la función cuartil que devuelve el x que corresponde al percentil `p` de DIST. Si `lower.tail=F`, devuelve `1` menos `p`.

Distribution	Density/pmf	cdf	Quantiles	Random Numbers
Normal	`dnorm()`	`pnorm()`	`qnorm()`	`rnorm()`
Chi square	`dchisq()`	`pchisq()`	`qchisq()`	`rchisq()`
Binomial	`dbinom()`	`pbinom()`	`qbinom()`	`rbinom()`

Distribución Normal

Construyo una figura de la función normal usando un vector entre -5 y 5 con 100 puntos.

```
library(ggplot2)
set.seed(8888) ## elijo la semilla para poder "controlar" la aleatoridad
x <- seq(from=-5, to=5, length.out=100) # el intervalo [-5 5]
f <- dnorm(x) # normal con media 0 y sd 1
ggplot(data.frame(col1=x, col2=f), aes(x=col1, y=col2)) + geom_line()
```

Otras distribuciones

Construyo un vector de 10^5 puntos que contenga valores estocásticos extraidos de una dist. Binomial de `n=5` (número de intentos) y `p=0.5` (probabilidad de éxito).

```
x <- rbinom(100000,5,0.5)
mean(x)
# [1] 2.5004

mean(x[x<=4])
# [1] 2.418766
```

Modelado estadístico

Modelado se refiere a proponer determinadas relaciones entre variables, típicamente cuál es la relación entre una variable dependiente o *variable respuesta* y otras variables independientes o *variables explicativas*.

En R la función `lm()` se usa para regresión lineal (*linear models*) y `glm()` para *generalized linear models*.

Regresión lineal - `lm()`

Construimos un "modelo" (una relación) entre variables dependientes e independientes optimizando parámetros para poder predecir.

1 - Propongo una determinada relación de variables.

2 - Calculo coeficientes del modelo.

3 - Compruebo que tan bien se ajusta el modelo a nuevas observaciones.

```
y[i] ~ f(x[i,]) = b[1] x[i,1] + ... b[n] x[i,n]
## b[i] son los coeficientes o betas
```

Ejemplo con datos de `2011 US Census PUMS`

Pueden bajar los datos de acá.

```
## hacemos la regresión:
load("psub.RData")
dtrain <- subset(psub, ORIGRANDGROUP >= 500)
dtest  <- subset(psub, ORIGRANDGROUP < 500)
model  <- lm(log(PINCP,base=10) ~ AGEP + SEX + COW + SCHL, data=dtrain)
dtest$predLogPINCP <- predict(model,newdata=dtest)

## resultados:
summary(model)

## graficamos:
library(ggplot2)
ggplot(data=dtest,aes(x=predLogPINCP,y=log(PINCP,base=10))) + geom_point(alpha=0.2,colo
geom_smooth(aes(x=predLogPINCP, y=log(PINCP,base=10)),color="black") +
geom_line(aes(x=log(PINCP,base=10), y=log(PINCP,base=10)),color="blue",linetype=2) +
scale_x_continuous(limits=c(4,5)) +
scale_y_continuous(limits=c(3.5,5.5))

## residuos:
ggplot(data=dtest,aes(x=predLogPINCP, y=predLogPINCP-log(PINCP,base=10))) +
geom_point(alpha=0.2,color="black") +
```

Regresión lineal generalizada - `glm()`

Los modelos lineales asumen que el valor predicho es continuo y que los errores van a ser "normales". Los modelos lineales generalizados relajan estas suposiciones.

```
## expresión general
glm(formula, family=familytype(link=linkfunction), data=)
```

Ejemplito: Regresión logística, para variables categóricas.

```
# F es un factor binario
# x1, x2 y x3 son predictores continuos
fit <- glm(F~x1+x2+x3,data=mydata,family=binomial())
summary(fit) # resultados
exp(coef(fit)) # coeficientes
predict(fit, type="response") # predicciones
residuals(fit, type="deviance") # residuos
```

Estadística avanzada - material infinito

- Paquete stats
- CRAN view de distribuciones
- Modern Applied Statistics with S. Fourth Edition - (MASS book)
- The elements of statistical learning - (ElemStatLearn book)

Otros paquetes de interés

- Numerical Mathematics
- Ecuaciones diferenciales
  - tienen también el libro "2012 - Book - Solving Differential Equations in R.pdf" en Google Classroom
- Series temporales
- Optimización y programación matemática
- Aritmética de precisión múltiple con gmp
- Paquete gsl, una interface a la Biblioteca Científica GNU
- Mil cosas más :)

Práctica 9

Descargar práctica 9.