Àlgebra lineal i càlcul amb R

Daniel Liviano Solís
Maria Pujol Jover

PID_00208266

UOC
Universitat Oberta de Catalunya

Índex

Introducció

L'objectiu d'aquest mòdul és aprofundir més en aspectes matemàtics que en estadístics. De fet, es podria dir que l'objectiu d'aquest mòdul és doble: d'una banda, cobreix diferents parts de l'àlgebra lineal necessàries per a l'estudi de l'estadística, com són les operacions matricials i la descomposició de matrius; d'una altra, a més, vol cobrir altres aspectes que van més enllà de l'estadística, com poden ser el càlcul o els nombres complexos.

Per això mateix, encara que R sigui un programa inicialment concebut per a l'estudi de l'estadística, també el poden fer servir estudiants d'assignatures de física, enginyeria i matemàtiques en general, en les quals en veuen en profunditat elements d'àlgebra lineal avançada i anàlisi numèrica. Així doncs, es pot pensar que R és una alternativa viable a programes com MATLAB o OCTAVE, àmpliament utilitzats en diverses branques de l'enginyeria.

Tradicionalment, les qüestions de càlcul s'han estudiat en l'àmbit computacional amb els anomenats *sistemes d'àlgebra computacional* (CAS és l'acrònim en anglès), que tenen com a programes principals Mathematica, Maxima i SAGE. Aquests programes són molt potents portant a terme operacions de diferenciació, integració i resolució d'expressions algebraiques **d'una manera simbòlica**, és a dir, introduint una funció com $y = x^n$, el programa ens torna l'expressió simbòlica de la derivada, això és, $y' = nx^{n-1}$. A diferència d'aquests programes, el càlcul simbòlic no es pot considerar un avantatge d'R, encara que cada vegada més apareixen extensions en R que permeten fer càlculs d'aquesta manera[1]. Això no obstant, el punt fort d'R és l'**anàlisi numèrica**, que se centra en l'estudi dels algorismes que utilitzen una *aproximació numèrica* (en contraposició a la manipulació simbòlica) per als problemes d'anàlisi matemàtica. Els temes que s'inclouen en l'anàlisi numèrica cobreixen gairebé tots els àmbits de les matemàtiques i l'estadística. Heus aquí els més destacats :

1) Càlcul dels valors de la funció.

2) Interpolació, extrapolació i regressió.

3) Resolució d'equacions i sistemes d'equacions.

4) Estudi de problemes de descomposició matricial: vectors i valors propis, i valors singulars.

5) Optimització lineal i quadràtica.

[1] Per exemple, algunes llibreries que permeten fer càlcul simbòlic amb R són *rSymPy*, *ryacas* i *mosaic*.

6) Avaluació d'integrals mitjançant integració numèrica.

7) Estudi d'equacions diferencials.

Per tot això, des d'aquí animem tots els estudiants d'assignatures que incloguin alguns d'aquests temes a donar una oportunitat a R, ja que els permetrà consolidar els coneixements teòrics que hagin adquirit i també posar-los en pràctica amb dades reals o simulades.

Objectius

1. Aprendre a manejar vectors i matrius i a fer operacions entre aquests.

2. Dominar les tècniques principals de descomposició matricial.

3. Trobar l'arrel d'equacions algebraiques i resoldre sistemes d'equacions.

4. Estudiar i representar gràficament funcions de diferents classes.

5. Aproximar numèricament el càlcul de derivades i integrals.

6. Saber resoldre problemes de valor inicial amb equacions diferencials ordinàries i parcials.

7. Resoldre algorismes d'optimització lineal i quadràtiques.

8. Dominar la notació i les operacions bàsiques amb nombres i variables complexes.

1. Àlgebra vectorial i matricial

1.1. Operacions algebraiques

Reprenent el que hem vist en el primer mòdul, R permet emmagatzemar dades numèriques en vectors i matrius, la qual cosa permet efectuar operacions algebraiques amb vectors, matrius i escalars. Vegem les possibles operacions que R permet efectuar.

1.1.1. Suma i resta d'escalars, vectors i matrius

R permet efectuar operacions vectoritzades, la qual cosa permet sumar o restar un escalar i un vector. En aquest sentit, se suma o es resta el valor de l'escalar a tots els elements del vector, com mostra l'exemple següent:

$$c \pm \boldsymbol{v} = c \pm (v_1, \ldots, v_n) = (c \pm v_1, \ldots, c \pm v_n)$$

```
> v <- 1:4
> 5+v
[1] 6 7 8 9
```

Això sí, per a sumar o restar vectors, aquests han de tenir la mateixa longitud. Vegem l'exemple següent:

Seqüències numèriques

Fixeu-vos que el vector `a` inclou els valors $(1, 2, 3, 4, 5)$, mentre que el vector `b` els inclou de manera inversa, això és, $(5, 4, 3, 2, 1)$.

```
> a <- 1:5
> b <- 5:1
> a+b
[1] 6 6 6 6 6
> a-b
[1] -4 -2  0  2  4
```

Anàlogament al cas dels vectors, es pot efectuar la suma i la resta d'escalars i matrius:

$$c \pm A = c \pm \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \ldots & a_{nn} \end{pmatrix} = \begin{pmatrix} c \pm a_{11} & \ldots & c \pm a_{1n} \\ \vdots & \ddots & \vdots \\ c \pm a_{n1} & \ldots & c \pm a_{nn} \end{pmatrix}$$

```
> A <- matrix(1:4,2,2)
> 3 + A
     [,1] [,2]
[1,]    4    6
[2,]    5    7
```

La suma i la resta de matrius es fa element per element, de manera que aquestes han de tenir la mateixa mida. Altrament, obtindríem un missatge d'error.

Consell per a programar ordenadament

En una anàlisi en què incloguem vectors i matrius, és recomanable anomenar els vectors amb minúscules, de manera que s'evitin confusions.

```
> A <- matrix(1:4,2,2)
> B <- matrix(4,2,2)
> A + B
     [,1] [,2]
[1,]    5    7
[2,]    6    8
```

Per a sumar i restar vectors i matrius no és necessari que aquests concordin en longitud i mida. En R, aquesta propietat es denomina **reciclatge**, això és, els valors del vector se sumen a la matriu fins a completar-la, i en cas que el producte de files i columnes sigui superior a la longitud del vector, els valors d'aquest es repetiran fins a completar la matriu. Vegem-ne un exemple:

Primer columnes, després files

Recordeu que, a l'hora de fer una operació entre un vector i una matriu, R comença l'operació per la primera columna, sempre de dalt a baix i d'esquerra a dreta.

```
> (v <- 2*1:4)
[1] 2 4 6 8
> (A <- matrix(0,2,4))
     [,1] [,2] [,3] [,4]
[1,]    0    0    0    0
[2,]    0    0    0    0

> v + A
     [,1] [,2] [,3] [,4]
[1,]    2    6    2    6
[2,]    4    8    4    8
```

En cas que la longitud de la matriu (files per columnes) no sigui múltiple de la longitud del vector, l'operació es durà a terme igualment, però obtindrem un missatge d'error per a recordar-nos aquesta circumstància.

```
> (v <- 2*1:4)
[1] 2 4 6 8
> (A <- matrix(0,2,3))
     [,1] [,2] [,3]
[1,]    0    0    0
```

```
 [2,]    0    0    0
 > v + A
      [,1] [,2] [,3]
 [1,]    2    6    2
 [2,]    4    8    4
 Missatges d'avís perduts
 In v + A :
   longitud d'objecte més gran no és múltiple de la
      longitud d'un de més petit
```

1.1.2. Producte i divisió element per element

Com en el cas anterior, R també permet multiplicar o dividir un escalar i un vector. Per a això, ens servirem dels símbols " $*$ " per a la multiplicació i "/" per a la divisió.

$$c\,\boldsymbol{v} = c\,(v_1, \ldots, v_n) = (c\,v_1, \ldots, c\,v_n)$$

$$c/\boldsymbol{v} = c/(v_1, \ldots, v_n) = (c/v_1, \ldots, c/v_n)$$

```
> 2 * 1:4
[1] 2 4 6 8
> 2 / 1:4
[1] 2.0000000 1.0000000 0.6666667 0.5000000
```

La multiplicació element per element entre vectors requereix que aquests tinguin la mateixa longitud.

```
> 1:3 * 1:3
[1] 1 4 9
```

De la mateixa manera, la multiplicació element per element entre matrius només és possible si aquestes tenen la mateixa dimensió.

```
> (A <- matrix(1:4,2,2))
     [,1] [,2]
[1,]    1    3
[2,]    2    4

> A * t(A)
     [,1] [,2]
[1,]    1    6
[2,]    6   16
```

Compte amb les multiplicacions de matrius!

Com veurem més endavant, hi ha més d'un operador de multiplicació definit per a matrius, amb la qual cosa cal tenir **molt clar** quin tipus de multiplicació volem aplicar. Aquest exemple mostra una multiplicació **element per element**.

Un cas especial és el de la multiplicació element per element d'un vector i una matriu. Aquesta operació només es pot fer si la longitud de l'objecte més petit és múltiple de la del més gran, entenent la longitud de la matriu com el producte del nombre de files i columnes. Cal anar amb compte en efectuar aquesta mena d'operacions, ja que és fàcil equivocar-se i que el resultat no sigui el que volem obtenir.

```
> 1:3 * matrix(1,2,3)
     [,1] [,2] [,3]
[1,]    1    3    2
[2,]    2    1    3
```

Per acabar, és interessant destacar que altres operadors algebraics també es poden aplicar element per element. Vegem el cas de l'operador potència:

```
> (A <- matrix(1:4,2,2))
     [,1] [,2]
[1,]    1    3
[2,]    2    4

> A^2
     [,1] [,2]
[1,]    1    9
[2,]    4   16

> A^A
     [,1] [,2]
[1,]    1   27
[2,]    4  256
```

1.1.3. Producte escalar entre dos vectors

Si tenim dos vectors $\boldsymbol{a}$ i $\boldsymbol{b}$ de la mateixa longitud (n elements), el producte escalar entre tots dos es defineix de la manera següent:

Producte escalar

També conegut com a producte intern, producte interior o producte punt (en anglès, *dot product*), es tracta d'una operació binària definida sobre dos vectors d'un espai euclidià que té com a resultat un escalar.

$$\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

En R, aquesta operació es fa de la manera següent:

```
> a <- 1:3
> b <- c(2,5,8)
> sum(a*b)
[1] 36
```

Sempre que dos vectors són ortogonals ($\boldsymbol{a} \perp \boldsymbol{b}$), el seu producte escalar és igual a zero.

Ortogonalitat

Geomètricament, dos vectors ortogonals formen un angle recte. La paraula procedeix del grec *orthos* ('recte') i *gonía* ('angle').

```
> a <- c(1,1)
> b <- c(-1,1)
> sum(a*b)
[1] 0
```

1.1.4. Norma d'un vector

Espai euclidià

En geometria, el concepte d'espai euclidià inclou el pla euclidià per a un espai de 2 dimensions i l'espai euclidià per a un espai de 3 o més dimensions. Es denomina així en honor del matemàtic grec antic Euclides d'Alexandria.

També denominada la *magnitud* d'un vector, es tracta de la distància entre l'origen i el vector. Això és, en un espai euclidià $\mathbb{R}^n$, la norma del vector $\boldsymbol{v} = (v_1, \ldots, v_n)$ queda definida per la fórmula següent:

$$\|\boldsymbol{v}\| = \sqrt{v_1^2 + \cdots + v_n^2}$$

En la distribució bàsica d'R no hi ha cap funció predefinida que faci aquesta operació amb vectors. Això no obstant, és molt fàcil crear una funció que la faci. Aquest és un dels avantatges principals d'R, que podem crear les nostres pròpies funcions personalitzades i fer-les servir quan les necessitem.

Pitàgores tenia raó

Geomètricament, en el pla euclidià la norma es pot interpretar com la longitud de la hipotenusa, en aquest cas $\sqrt{1^2 + 1^2} = \sqrt{2} = 1,414214$.

```
> norm.vec <- function(x) sqrt(sum(x*x))
> a <- c(1,1)
> norm.vec(a)
[1] 1.414214
```

1.1.5. Producte matricial

No confongueu els tipus de multiplicació!

En R, el símbol $*$ es fa servir per a la multiplicació element per element, mentre que el símbol $\%*\%$ es fa servir per al producte matricial.

El producte matricial entre $A_{n\times m}$ i $B_{m\times p}$ s'obté aplicant la fórmula següent:

$$AB = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & \cdots & b_{1p} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{np} \end{pmatrix} =$$

$$\begin{pmatrix} a_{11}b_{11} + \cdots + a_{1n}b_{n1} & \cdots & a_{11}b_{1p} + \cdots + a_{1n}b_{np} \\ \vdots & \ddots & \vdots \\ a_{m1}b_{11} + \cdots + a_{mn}b_{n1} & \cdots & a_{m1}b_{1p} + \cdots + a_{mn}b_{np} \end{pmatrix}$$

Per a poder-lo aplicar, el nombre de columnes de A ha de ser igual al nombre de files de B, i la matriu resultant tindrà tantes files com A i tantes columnes com B. En R, aquesta operació es fa amb l'operador `%*%`:

```
> (A <- matrix(1:4,2,2))
     [,1] [,2]
[1,]    1    3
[2,]    2    4

> (B <- matrix(2,2,1))
     [,1]
[1,]    2
[2,]    2

> A %*% B
     [,1]
[1,]    8
[2,]   12
```

També és possible utilitzar el producte matricial amb vectors. Això és, reinterpretant els vectors $\boldsymbol{a}$ i $\boldsymbol{b}$ de longitud n com a vectors columna $\boldsymbol{a}_{1\times n}$ i $\boldsymbol{b}_{1\times n}$, el producte euclidià interior i exterior són els casos més senzills especials del producte de la matriu, mitjançant la transposició dels vectors columna en vectors fila.

El **producte interior** és el resultat de multiplicar un vector fila per un vector columna, la qual cosa equival al producte escalar:

El resultat del producte interior de dos vectors sempre és un escalar.

$$\boldsymbol{a}\cdot\boldsymbol{b}=\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}=\begin{pmatrix}a_1 & a_2 & \cdots & a_n\end{pmatrix}\begin{pmatrix}b_1\\ b_2\\ \vdots\\ b_n\end{pmatrix}=a_1b_1+a_2b_2+\cdots+a_nb_n=\sum_{i=1}^{n}a_ib_i$$

En R podem aplicar aquesta operació de dues maneres: amb l'operador de producte matricial `%*%` i amb la funció del producte creu `crossprod`, la qual cosa efectua l'operació A^TB tant per a vectors com per a matrius:

```
> a <- 1:3

> a %*% a
     [,1]
[1,]   14

> crossprod(a,a)
     [,1]
[1,]   14
```

El **producte exterior** és el resultat de multiplicar un vector columna per un vector fila:

El resultat del producte exterior de dos vectors sempre serà una matriu.

$$\boldsymbol{a} \otimes \boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^{\mathrm{T}} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \begin{pmatrix} b_1 & b_2 & \cdots & b_n \end{pmatrix} = \begin{pmatrix} a_1b_1 & a_1b_2 & \cdots & a_1b_n \\ a_2b_1 & a_2b_2 & \cdots & a_2b_n \\ \vdots & \vdots & \ddots & \vdots \\ a_nb_1 & a_nb_2 & \cdots & a_nb_n \end{pmatrix}.$$

Apliquem aquesta operació en R amb l'operador `%o%`, el qual calcula AB^T tant per a vectors com per a matrius:

```
> a <- 1:3

> a %o% a
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    2    4    6
[3,]    3    6    9
```

1.1.6. Producte de Kronecker

El producte de Kronecker, representat pel símbol ⊗, és una operació entre dues matrius que resulta en una matriu en blocs. És una generalització de vectors a matrius del producte exterior (representat pel mateix símbol). Aquesta operació consisteix en el següent: cada element de la primera matriu multiplica tots els elements de la segona matriu. Partint de les matrius $A_{m\times n}$ i $B_{p\times q}$, el producte de Kronecker $A \otimes B$ resultarà en una matriu en blocs de dimensió $mp \times nq$:

Leopold Kronecker

Leopold Kronecker (1823-1891) va ser un matemàtic alemany que va treballar en la teoria dels nombres i l'àlgebra.

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}$$

R disposa de la funció `kronecker` per a fer aquesta operació.

```
> (A <- matrix(1,1,3))
     [,1] [,2] [,3]
[1,]    1    1    1

> (B <- matrix(1:4,2,2))
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```
> kronecker(A,B)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    3    1    3    1    3
[2,]    2    4    2    4    2    4
```

1.2. Equacions algebraiques

Aquesta mena d'equacions involucra polinomis. Recordem que un polinomi de grau n pren la forma següent:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

Les equacions algebraiques, també denominades *polinòmiques*, són equacions en què s'estableix la igualtat de dos polinomis $P = Q$, essent l'objectiu trobar la solució, és a dir, el valor de x que satisfà la igualtat $P = Q$, o el que és el mateix $P - Q = 0$. Com veurem més endavant, R disposa de diverses alternatives per a solucionar equacions. En el cas dels polinomis, una de senzilla consisteix a crear vectors amb els coeficients del polinomi en ordre creixent $(a_0, a_1, \ldots, a_n)$, després restar-los i buscar l'arrel mitjançant la funció `polyroot`, que és la solució. Vegem-ne un exemple senzill:

$$2x^2 + 5x + 5 = x^2 + 3x + 4$$

$$x^2 + 2x + 1 = 0$$

Un nombre complex es representa en forma binomial com:

Notació complexa

R torna les arrels d'un polinomi en forma binomial $z = a + bi$, essent a la part real i b, la part imaginària. En aquest cas, en ser $b = 0$, les solucions són $(-1, -1)$.

```
> p1 <- c(5,5,2)
> p2 <- c(4,3,1)

> polyroot(p1-p2)
[1] -1-0i -1+0i
```

És important tenir en compte que el nombre d'arrels d'un polinomi és igual al seu grau, en aquest cas $n = 2$. A més, les arrels es poden expressar en nombres complexos[1], amb la qual cosa en aquest cas la solució s'interpreta com a $x = -1 \pm 0 \cdot i = -1$.

[1] Recordem que $i = \sqrt{-1}$. En la part final d'aquest mòdul veurem amb detall com R treballa amb nombres complexos.

1.3. Sistemes d'equacions lineals

En matemàtiques, la teoria de sistemes lineals és una part fonamental de l'àlgebra lineal. La fórmula general d'un sistema de m equacions lineals amb n incògnites pren la forma següent:

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \ldots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \ldots & + & a_{2n}x_n & = & b_2 \\ \ldots & & \ldots & & \ldots & & \ldots & & \ldots \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \ldots & + & a_{mn}x_n & = & b_m \end{array}$$

En aquest sistema, x_j són les incògnites, a_{ij} els coeficients i b_i els termes independents. Aquest tipus de sistemes pot ser:

- *Incompatible:* si no té solució.
- *Compatible:* si té solució. Serà *determinat* si té una única solució, i *indeterminat* si admet un conjunt infinit de solucions.

Utilitzant les propietats del producte de matrius, és possible expressar el sistema d'equacions lineals anterior en forma matricial: $A \cdot x = b$.

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Hi ha molts mètodes de resolució d'aquest tipus de sistemes, que depenen de les seves característiques. En R, la funció bàsica que resol sistemes d'equacions és `solve`. Vegem-ne l'aplicació resolent el sistema següent:

$$\begin{aligned} 3x + y + 2z &= 10 \\ 4x + 3y + 4z &= 21 \\ 2x + y + 2z &= 9 \end{aligned}$$

Primer expressem el sistema mitjançant matrius:

$$A = \begin{pmatrix} 3 & 1 & 2 \\ 4 & 3 & 4 \\ 2 & 1 & 2 \end{pmatrix}, \qquad b = \begin{pmatrix} 10 \\ 21 \\ 9 \end{pmatrix}$$

```
> e1 <- c(3,1,2)
> e2 <- c(4,3,4)
> e3 <- c(2,1,2)

> A <- rbind(e1,e2,e3)

> b <- c(10,21,9)
```

Tot seguit, utilitzem la funció `solve` per a calcular $x = A^{-1}b$, i obtenim la solució $x = 1$, $y = 3$ i $z = 2$.

```
> solve(A,b)
[1] 1 3 2
```

1.4. Descomposició matricial

1.4.1. Vectors i valors propis

En àlgebra lineal, la descomposició espectral (també anomenada *eigendescomposició* o *descomposició en vectors i valors propis*) és la factorització d'una matriu en una forma canònica, en la qual la matriu es representa en termes dels seus valors i vectors propis. Només les matrius diagonalitzables es poden factoritzar d'aquesta manera.

Analíticament, un vector no nul $\boldsymbol{v}$ de longitud N és un *vector propi* d'una matriu quadrada A de dimensió $N \times N$ si i solament si se satisfà l'equació lineal següent:

$$A\boldsymbol{v} = \lambda\boldsymbol{v}$$

Essent λ un escalar denominat el *valor propi* associat a $\boldsymbol{v}$, la interpretació és la següent: la matriu A s'interpreta com una transformació lineal aplicada al vector $\boldsymbol{v}$. Aleshores, aquest vector $\boldsymbol{v}$ serà aquell al qual la transformació A només allarga o encongeix (és a dir, canvia la longitud), però **no** en canvia la direcció. Aleshores, el valor propi λ serà la magnitud en la qual el vector propi s'allarga o s'encongeix.

Per a cada valor propi λ_i disposem de l'equació següent:

$$(A - \lambda_i\mathbf{I})\,\boldsymbol{v} = 0.$$

Essent I la matriu identitat $N \times N$.

En R, els valors i vectors propis es calculen mitjançant la funció `eigen`. Per exemple, suposem que volem calcular els valors i vectors propis de la matriu següent:

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

```
> A <- matrix(c(2,0,0,2),2,2)

> eigen(A)
$values
[1] 2 2

$vectors
     [,1] [,2]
[1,]    0   -1
[2,]    1    0
```

La funció `eigen`

Aquesta funció torna una llista amb dos elements: els valors (*values*) i els vectors (*vectors*), als quals s'accedeix mitjançant l'operador $.

En aquest cas, hem obtingut dos valors propis ($\lambda_1 = \lambda_2 = 2$) i dos vectors propis ($\boldsymbol{v}_1 = (0, 1)$ i $\boldsymbol{v}_2 = (-1, 0)$). És interessant apuntar que sempre obtindrem tants valors i vectors propis com la dimensió de la matriu de transformació A.

1.4.2. Descomposició en valors singulars

Aquesta descomposició, abreujada com a DVS (en anglès l'acrònim és SVD, *singular value decomposition*), està relacionada amb la descomposició espectral que hem vist anteriorment. Partint d'una matriu A, igual que en el cas anterior, la DSV efectua l'operació següent:

DVS

Aquesta descomposició té moltes aplicacions en estadística i altres disciplines, com el processament de senyals en enginyeria.

$$A = UDV^T$$

En què U i V són ortogonals, V^T és la matriu transposada de V i D és una matriu diagonal amb els valors singulars D_{ii}. De manera equivalent, també es pot expressar com a $D = U^T AV$. Vegem-ne un exemple amb la matriu anterior A. Como veiem, D conté els valors singulars de A.

```
> A <- matrix(c(2,0,0,2),2,2)

> svd(A)
$d
[1] 2 2

$u
```

```
     [,1] [,2]
[1,]    1    0
[2,]    0    1

$v
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

A més, en R hi ha altres descomposicions matricials disponibles, l'estudi de les quals sobrepassa els objectius d'aquests apunts. Les principals són la factorització QR (funció `qr`) i la factorització de Cholesky (funció `chol`).

2. Funcions

2.1. Introducció

En aquest capítol veurem les múltiples possibilitats per a treballar amb funcions matemàtiques del tipus $y = f(x)$ que ens ofereix R. La funció bàsica, explicada en el mòdul introductori, és `function`. Per exemple, la funció d'una variable $y = x^2 - 3$:

```
> y <- function(x) x^2 - 3
```

Amb la funció *y* introduïda, és immediat avaluar la funció per a valors específics de *x*, tant escalars com vectors:

```
> y(1)
[1] -2

> (x0 <- -3:3)
[1] -3 -2 -1  0  1  2  3

> y(x0)
[1]  6  1 -2 -3 -2  1  6
```

Avaluació de funcions

En aquest exemple, hem creat un vector de valors inicials x_0 amb els valors $(-3, -2, \ldots, 2, 3)$ i els hem avaluat amb la funció creada $y(x)$.

La funció `uniroot` es fa servir per a buscar les arrels d'una funció, és a dir, els valors de *x* per als quals es compleix $f(x) = 0$. Per a això, cal introduir un vector amb la part del domini en la qual volem buscar arrels. En aquest cas, buscarem en l'interval $x \in [-2, 0]$:

```
> uniroot(y,c(-2,0))
$root
[1] -1.73205

$f.root
[1] -3.278749e-06

$iter
[1] 5

$estim.prec
[1] 6.103516e-05
```

Com veiem, aquesta funció ens ofereix molta informació. La part rellevant és `$root`, la qual ens indica que $f(x_0) = 0$ si $x_0 = -1,73205$. La resta d'informació fa referència al nombre d'iteracions i a la precisió de l'estimació. En aquest punt és important destacar que l'interval introduït ha de contenir una única arrel, és a dir, la funció $f(x)$ ha de tallar la recta horitzontal una sola vegada. Si no és així, obtindrem un missatge d'error:

```
> uniroot(y,c(-2,2))
Error in uniroot(y, c(-2, 2)) :
  f() values at end points not of opposite sign
```

2.2. Classes de funcions

Hi ha diferents classificacions de funcions matemàtiques. Una classificació fonamental es basa en el nombre d'arguments o incògnites que té:

1) Funcions d'una variable: prenen la forma $y = f(x)$, com la funció que hem descrit abans.

2) Funcions de diverses variables: són una generalització del cas anterior a n arguments, això és, $y = f(x_1, \ldots, x_n)$.

Un exemple d'aquest segon cas seria la funció $y = x_1^2 + x_2^2$. La manera d'introduir-la en R és:

```
> y <- function(x1,x2) x1^2 + x2^2
```

Si ens limitem a les funcions d'una sola variable, les podem resumir en la taula 1:

Taula 1. Classes bàsiques de funcions

Classe	Forma funcional
Lineal	$mx + b$
Potència	x^n
Arrel	$\sqrt[n]{x}$
Racional	$P(x)/Q(x)$
Exponencial	a^x
Logarítmica	$log_a(x)$
Trigonomètrica	$sin(x)$
	$cos(x)$
	$tan(x)$

A més, les funcions poden ser **algebraiques** si es poden construir amb operacions algebraiques (suma, resta, multiplicació, divisió i arrel) o **transcendentals**, quan això no passa.

2.3. Representació gràfica

2.3.1. Gràfics d'una funció

Un dels avantatges principals de R sobre altres programes és la potència i la versatilitat dels gràfics. En aquest apartat només cobrim una minúscula part d'aquest potencial. La funció bàsica de representació gràfica en R és `plot`. Aquesta funció admet diversos tipus d'arguments, això és, tipus d'objectes que poden ser representats. En el cas de funcions d'una sola variable com `y`, es poden representar directament. El quadre següent recull les principals opcions que ens ofereix `plot`:

Taula 2. Opcions gràfiques elementals

Descripció	Instrucció	Opcions
Títol	`main`	`"texto"`
Nom de l'eix x	`xlab`	`"texto"`
Nom de l'eix y	`ylab`	`"texto"`
Domini	`xlim`	`c(a,b)`
Rang	`ylim`	`c(a,b)`
Tipus de gràfic	`type`	Línies: `"l"` Punts: `"p"` Tots dos: `"o"`
Amplada de línia	`lwd`	`1,2,...`
Color de línia	`col`	`"red"`, `"blue"`,...

Paràmetres gràfics

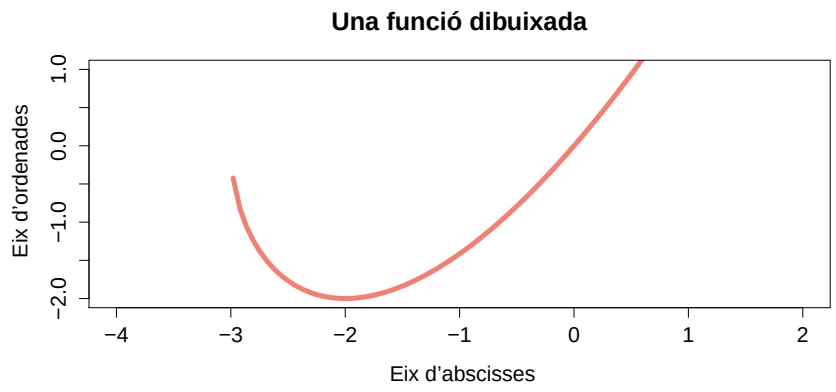
Si acudim a l'ajuda d'R, introduint en la consola la instrucció `help(par)`, obtindrem una llista extensa d'opcions gràfiques, amb les quals podrem personalitzar i modelar tots els aspectes del gràfic.

Considerem un exemple d'una funció amb una variable $y = x\sqrt{x+3}$:

```
> y <- function(x) x*sqrt(x+3)
```

Una representació de la funció $y = x\sqrt{x+3}$ podria ser la següent. El resultat es mostra en la figura 1:

Figura 1. Representació de la funció $y = x\sqrt{x+3}$



```
> plot(y,
+      main="Una funció dibuixada",
+      xlab=" Eix d'abscisses",
+      ylab=" Eix d'ordenades",
+      xlim=c(-4,2),
+      ylim=c(-2,1),
+      type="l",
+      lwd=5,
+      col="salmon")
Warning message:
In sqrt(x + 3) : NaNs produced
```

Consell de programació

A l'hora de programar gràfics, és aconsellable començar una nova línia després de cada coma, de manera que tinguem una instrucció per línia i visualment sigui tot més clar.

El missatge d'avís `In sqrt(x + 3) : NaNs produced` ens indica que hem intentat representar el gràfic en l'interval $x \in [-4, 3]$, però el domini de la funció és $[-3, \infty]$, amb la qual cosa els valors en l'interval $[-4, -3]$ no es poden mostrar, ja que no existeixen.

2.3.2. Gràfics de diverses funcions

Una altra manera de representar gràficament funcions és creant un vector de valors inicials. Això és, l'argument de la funció `plot` no serà la funció pròpiament, sinó una sèrie de valors resultants d'aquesta. Això és molt útil quan representem una funció amb més d'una variable, o més d'una funció en un gràfic, o quan la funció s'ha de representar per parts. En veurem un exemple amb les funcions trigonomètriques del sinus, el cosinus i la tangent. Aquestes són funcions implementades en R i, per tant, no cal crear-les. El primer pas serà crear un vector de valors inicials x_0. Si volem un gràfic d'alta qualitat, entre el màxim i el mínim hi haurà d'haver molts punts intermedis. En el nostre cas, avaluarem les tres funcions en l'interval $x \in [-5, 5]$, que conté 1.000 subintervals, de manera que el vector resultant tindrà una longitud de 1001:

```
> x0 <- seq(-5,5,0.01)

> (lon <- length(x0))
[1] 1001
```

Després creem tres vectors que siguin l'avaluació de les tres funcions en els punts x_0. Lògicament, la longitud dels vectors resultants serà la mateixa:

```
> y1 <- sin(x0)
> y2 <- cos(x0)
> y3 <- tan(x0)
```

El pas següent és la representació gràfica. Com que tenim més d'una funció és recomanable crear primer el marc del gràfic, això és, els valors màxim i mínim en els eixos de coordenades. Els rangs dels eixos seran $(1, 1001)$ per a les abscisses i $(-4, 4)$ per a les ordenades:

```
> (xrange <- c(1,lon))
[1]    1 1001

> (yrange <- c(-4,4))
[1] -4  4

> plot(xrange,yrange,type="n",xaxt="n",xlab="",ylab="")
```

Elaboració d'un gràfic

Aquest pas ens ha creat un quadre buit. Els dos primers arguments són els rangs. La instrucció `type="n"` especifica que no hi ha funció, `xaxt="n"` elimina l'eix d'abscisses, i tant `xlab=""` com `ylab=""` deixen els noms dels eixos en blanc.

Ara ja podem anar afegint valors al quadre mitjançant la funció `lines`, en què especifiquem el vector que volem representar, com també les característiques de la representació (tipus de línia, gruix, color, etc.). A més, amb la funció `abline` dibuixarem una línia horitzontal en el punt $y = 0$ mitjançant l'opció `h=0`:

```
> lines(y1,type="p",lwd=1,col="red")
> lines(y2,type="p",lwd=1,col="blue")
> lines(y3,type="p",lwd=1,col="orange")
> abline(h=0,lwd=2)
```

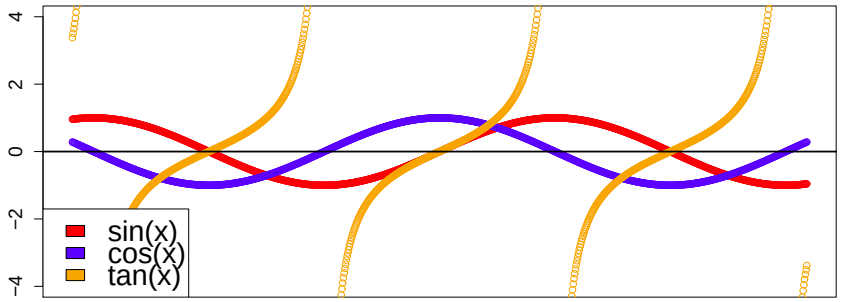
L'últim pas és afegir la llegenda, la qual cosa es fa amb la funció `legend`. Aquí especifiquem on volem la llegenda (a baix i a l'esquerra), el text associat a cada funció, el color de cada funció, i per acabar `cex`, que indica la proporció de la llegenda respecte al gràfic (un valor més gran indicarà una mida més gran). El resultat final es mostra en la figura 2:

```
> legend("bottomleft",
+        c("sin(x)","cos(x)","tan(x)"),
+        fill=c(" red","blue","orange"),
+        cex=1.5)
```

Paràmetres de la llegenda

Si introduïm la instrucció `help(legend)` obtindrem una descripció detallada de les opcions per a personalitzar el quadre de la llegenda.

Figura 2. Representació de les funcions $sin(x)$, $cos(x)$ i $tan(x)$



Un tipus de funció molt comú és el de la **funció per parts** o **funció definida a trossos**. Considerem la funció següent:

$$f(x) = \begin{cases} 4x^2 + \frac{x!}{x^x} & \text{si } x > 0 \\ 5 & \text{si } x = 0 \\ 6\log(|x|) & \text{si } x < 0 \end{cases}$$

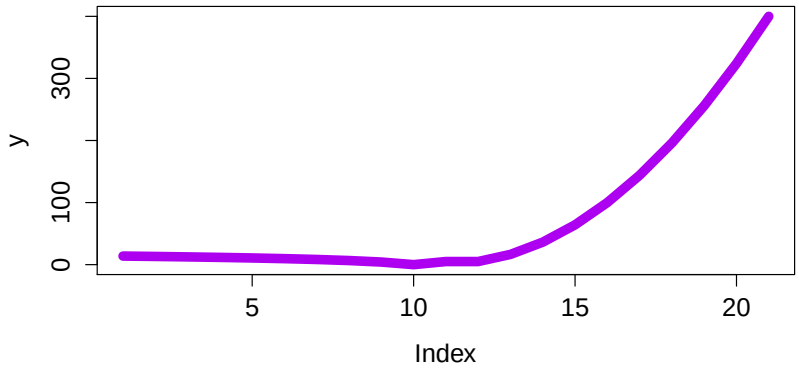
Suposem que volem avaluar-la i representar gràficament en l'interval $x \in (-10, 10)$. Per a això haurem d'utilitzar cicles i condicionals, tal com es mostra a continuació:

Cicles i condicionals

Aquesta estructura pot semblar enrevessada i complexa. Val la pena que la llegiu més d'una vegada per intentar entendre què és el que estem demanant a R.

```
> x0 <- (-10:10)
> n <- length(x0)
> y <- rep(0,n)
> for (i in 1:n){
+     x <- x0[i]
+     if (x>0){
+         y[i] <- 4*x*x + (factorial(x)/x^x)
+     }else if (x==0){
+         y[i] <- 5
+     }else{
+         y[i] <- 6*log(abs(x))}
+ }

> plot(y,type="l",lwd=8,col="purple")
```

Figura 3. Representació d'una funció definida a trossos



2.3.3. Gràfics en tres dimensions

Aquest tipus de gràfics es fan servir per a representar funcions en $\mathbb{R}^3$, és a dir, del tipus $y = f(x_1, x_2)$, de manera que s'ha de visualitzar sobre un sistema de tres eixos de coordenades (és a dir, un espai euclidià tridimensional).

Partim d'un exemple conegut: la funció paraboloide hiperbòlic (també coneguda com a punt de sella):

$$z = f(x, y) = x^2 - y^2$$

Suposem que volem avaluar aquesta funció per als valors de x i y inclosos en el domini $[-100, 100]$. El primer pas serà crear els vectors de valors inicials x_0 i y_0, i definir la funció z:

```
> x0 <- -100:100
> y0 <- -100:100
> z <- function(x,y) x*x-y*y
```

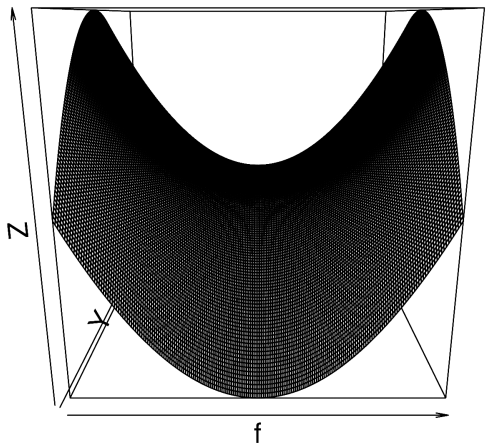
Arribats a aquest punt, hi ha una funció d'R anomenada `outer` que és de gran utilitat. Aquesta funció utilitza com a arguments els vectors $\boldsymbol{x} = (x_1, \ldots, x_n)$ i $\boldsymbol{y} = (y_1, \ldots, y_n)$ i una funció $f(x, y)$, i calcula la matriu següent $n \times n$:

$$\begin{pmatrix} f(x_1, y_1) & f(x_1, y_2) & \cdots & f(x_1, y_n) \\ f(x_2, y_1) & f(x_2, y_2) & \cdots & f(x_2, y_n) \\ \vdots & \vdots & \ddots & \vdots \\ f(x_n, y_1) & f(x_n, y_2) & \cdots & f(x_n, y_n) \end{pmatrix}$$

Un cop creada aquesta matriu, la podem representar gràficament mitjançant la funció `persp`, la qual representa els valors recollits en una matriu en un espai tridimensional. El gràfic resultant es mostra en la figura 4.

```
> f <- outer(x0,y0,z)
> persp(f,col="gold")
```

Figura 4. Representació de la funció hiperbòlica $z = x^2 - y^2$



2.4. Exemples de funcions en diversos camps

2.4.1. Física: teoria de la relativitat

La teoria de la relativitat estableix que la massa d'un cos augmenta amb la velocitat, i la fórmula de la *massa relativística* estableix la relació següent:

$$M = \frac{m}{\sqrt{1 - \frac{v^2}{c^2}}}.$$

En què la massa M depèn de la massa del cos en repòs m, la seva velocitat v i la velocitat de la llum al buit $c = 3,0 \times 10^5$ km/s. El primer pas és introduir la funció `M`:

```
> M <- function(m,v) m/sqrt(1-(v^2)/(300000^2))
```

1. Si una persona de 100 kg viatgés per l'espai a 2.000.000 km/h, quina massa tindria?

Primer hem de convertir la velocitat en km/h a km/s, i després avaluar la funció amb aquestes dades.

```
> v0 <- 2000000/3600

> M(100,v0)
[1] 100.0002
```

És a dir, la seva massa hauria augmentat en 2 dg, és a dir, en 0,2 g.

2. A quina velocitat ha de viatjar aquesta persona per a augmentar la seva massa en un 10%?

Per a resoldre aquesta qüestió hem d'utilitzar la funció `uniroot`, la qual busca les arrels d'una funció de manera que es compleixi $f(x, ...) = 0$. En el nostre cas, busquem el valor de la velocitat v_0 que compleixi la condició següent:

$$\frac{100}{\sqrt{1 - \frac{v_0^2}{c^2}}} - 110 = 0.$$

Si creem una segona funció (`M2`) amb la velocitat com a argument, podem trobar el valor que busquem:

```
> M2 <- function(v) 100/sqrt(1-(v^2)/(300000^2))-110

> uniroot(M2,c(0,200000))$root
[1] 124979.3
```

Això és, 124.979,3 km/s.

La massa en la teoria de la relativitat

La massa relativista és la massa que s'assigna a un cos en moviment. Abans de la teoria de la relativitat, la mecànica clàssica establia que el moment lineal d'un objecte estava determinat per la fórmula $p = mv$, i la seva energia per $E = E_0 + \frac{1}{2}mv^2$, essent E_0 l'energia en repòs (un concepte indeterminat) i $\frac{1}{2}mv^2$, l'energia cinètica. La teoria de la relativitat especial va completar aquestes definicions definint la massa en funció de la velocitat de l'objecte respecte a la velocitat de la llum c, això és, $M = \gamma m$, essent $\gamma = 1/\sqrt{1 - v^2/c^2}$.

3. Com es representa gràficament la relació entre la massa d'aquesta persona i la seva velocitat?

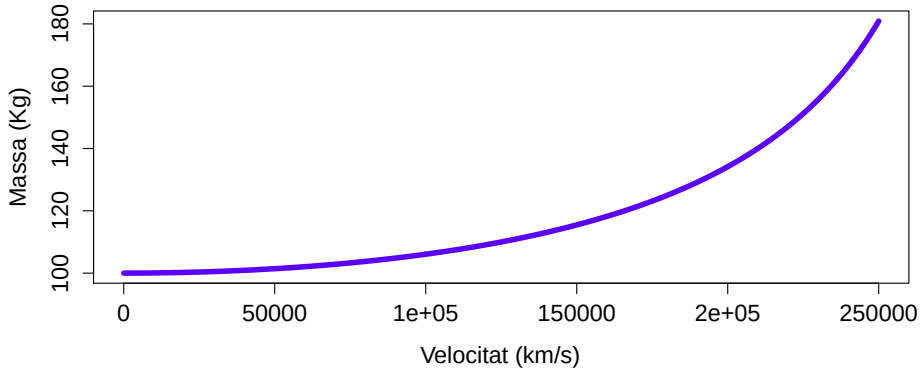
Primer creem el vector v_0 de valors inicials, avaluem la funció `M` amb els valors de v_0 i amb el valor fixat $m = 100$. Un cop tenim el vector `y`, elaborem el gràfic (figura 5). És interessant destacar que amb la funció `axis` hem personalitzat l'eix d'abscisses, fent que només es marquin els punts del vector `lab` amb els valors corresponents de v_0.

```
> v0 <- seq(0,250000,by=100)

> y <- M(v0,m=100)

> plot(y,type="l",lwd=5,col="blue",xaxt="n",
+ xlab="Velocitat (km/s)",ylab="Massa (Kg)")

> lab <- c(1+500*0:5)
> axis(1,at=lab,labels=v0[lab])
```

Personalitzant l'eix d'abscisses

Us recomanem que repliqueu aquest exemple en el vostre ordinador, parant atenció especialment als valors de `lab` i `v0[lab]`.

Figura 5. Representació de la funció de la massa relativística



2.4.2. Enginyeria: la funció catenària

La funció catenària és la corba que descriu una cadena suspesa pels extrems, sotmesa a un camp gravitatori uniforme. Matemàticament, la funció que cal estudiar és la següent[1]:

$$y = a\,\cosh\left(\frac{x}{a}\right) = \frac{a}{2}\left(e^{x/a} + e^{-x/a}\right).$$

La funció catenària

Els primers matemàtics que van examinar aquest problema van suposar, errròniament, que aquesta corba era una paràbola. Finalment, l'equació la van obtenir Gottfried Leibniz, Christiaan Huygens i Johann Bernoulli el 1691.

El paràmetre a és el quocient entre la tensió horitzontal i el pes per unitat de longitud. El primer pas és introduir la funció en llenguatge d'R:

[1] Recordem que la funció cosinus hiperbòlic es defineix com a $\cosh(x) = \frac{1}{2}\,(e^x + e^{-x})$.

```
> f.cat <- function(x,a){
+   f1 <- exp(x/a)
+   f2 <- exp(-x/a)
+   return(0.5*a*(f1+f2))
+ }
```

1. Crea una representació gràfica de tres corbes amb $a = 0,5$, $a = 1$ *i* $a = 1,5$.

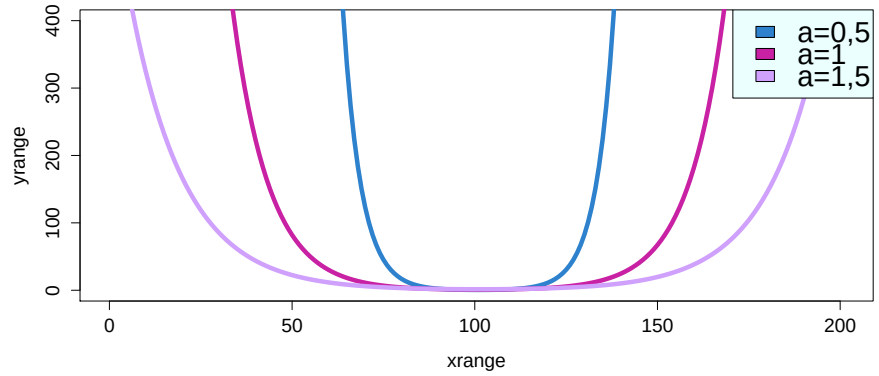
Com hem fet més amunt, avaluarem la funció per a aquests tres valors de a en el rang $x \in (-10, 10)$, amb la qual cosa obtenim `y1`, `y2` i `y3`. Finalment, creem el gràfic amb la llegenda corresponent (figura 6).

```
> x0 <- seq(-10,10,by=0.1)

> y1 <- f.cat(x0,a=0.5)
> y2 <- f.cat(x0,a=1)
> y3 <- f.cat(x0,a=1.5)

> xrange <- c(0,length(x0))
> yrange <- c(0,400)

> colors <- c("steelblue","violetred","plum")

> plot(xrange,yrange,type="n")
> lines(y1,type="l",lwd=5,col=colors[1])
> lines(y2,type="l",lwd=5,col=colors[2])
> lines(y3,type="l",lwd=5,col=colors[3])
> legend("topright",c(" a=0,5"," a=1"," a=1,5"),
         fill=colors,cex=1.5)
```

Posició de la llegenda

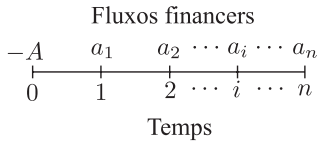
Fixeu-vos que en aquest cas hem situat la llegenda en la part superior dreta del gràfic (`topright`).

Figura 6. Representació de corbes catenàries



2.4.3. Finances: anàlisis d'inversions

Un projecte d'inversió consta d'una inversió inicial $-A$ i d'una sèrie de fluxos futurs esperats a_i durant els n períodes de la inversió. L'estructura temporal és la següent:



Els dos indicadors principals per a avaluar un projecte d'inversió són el valor actual net (VAN) i la taxa interna de retorn (TIR). El VAN consisteix a avaluar en el període $t = 0$ la inversió inicial i els fluxos a_i aplicant una taxa de descompte r:

Valor actual net (VAN)

El VAN compara el valor actual d'un euro amb el valor d'aquest mateix euro en el futur, tenint en compte la inflació i els fluxos de caixa que el projecte generi en el futur. Si el VAN d'un projecte és positiu, es pot acceptar. Això no obstant, si el VAN és negatiu, probablement el projecte s'hauria de rebutjar perquè els fluxos de caixa actualitzats amb la taxa de retorn no superarien la inversió inicial.

$$VAN = -A + \frac{a_1}{(1+r)} + \frac{a_2}{(1+r)(1+r)} + \ldots + \frac{a_n}{(1+r)(1+r)\cdots(1+r)}.$$

$$VAN = -A + \sum_{i=1}^{n} \frac{a_i}{(1+r)^i}.$$

La lògica d'aquest indicador és que els fluxos futurs valen menys que els fluxos presents, ja que el valor dels diners canvia amb el temps. Així, el valor en $t = 0$ de a_i es veurà reduït per un factor $(1+r)^i$. Si avaluem la inversió prenent un valor elevat de r, el VAN serà més petit. Per la seva banda, la TIR es defineix com el valor de r que fa que el VAN sigui zero, és a dir, que es compleixi $VAN = 0$. Una TIR positiva implica que $A < \sum_{i=1}^{n} a_i$, i per tant es recupera la inversió inicial.

Taxa interna de retorn (TIR)

La TIR és la taxa de descompte que fa que el valor present net dels fluxos d'efectiu d'un projecte sigui igual a zero. En termes generals, com més gran sigui la TIR d'un projecte, més recomanable serà portar-lo a terme. Per aquest motiu, la TIR es pot utilitzar per a classificar diversos projectes potencials considerats per una empresa.

1. Es considera un projecte que requereix una inversió inicial de 10.000 euros i una vida útil de 20 anys. Suposant un flux constant d'efectiu de 700 euros i una taxa de descompte del 3%, quin és el valor actual net d'aquest projecte?

El primer pas serà crear la funció `VAN` que tingui com a arguments el vector $V = (-A, a_1, \ldots, a_n)$ i la taxa de descompte r:

```
> VAN <- function(V,r) sum(V/(1+r)^(0:(length(V)-1)))
```

Una vegada introduïda aquesta funció, només cal introduir els valors inicials i avaluar la funció `VAN`:

```
> A <- 10000
> a <- 700
> n <- 20
> v0 <- c(-A,rep(a,n))

> VAN(v0,0.03)
[1] 414.2324
```

Aquest resultat indica que la inversió té un valor net positiu amb $r = 3\%$.

2. Quina és la TIR d'aquesta inversió?

Per a calcular la TIR cal buscar el valor de r que iguala la funció del `VAN` a zero, la qual cosa equival a buscar l'arrel de la funció. Per a tenir en compte valors tant positius com negatius de r, fixem un rang de cerca `c(-1,1)`. A més, especifiquem quin és el vector de valors inicials (`v0`).

```
> uniroot(VAN,c(-1,1),V=v0)$root
[1] 0.03443455
```

Segons aquest criteri, la rendibilitat d'aquesta inversió és del $3,44\%$.

2.4.4. Economia: funcions d'oferta i demanda

La llei de l'oferta i la demanda

En economia, aquesta teoria explica la interacció entre l'oferta d'un recurs (*supply* en anglès) i la demanda d'aquest recurs. La llei de l'oferta i la demanda defineix l'efecte que la disponibilitat i la demanda d'un producte tenen sobre la quantitat produïda i el seu preu de mercat. En general, si hi ha una oferta baixa i una demanda alta, el preu és més alt. Al contrari, com més gran és l'oferta i més baixa la demanda, més baix és el preu.

Suposem un mercat determinat en què el preu i la quantitat d'equilibri (p^* i q^* respectivament) es determinen a partir del punt d'intersecció de les funcions d'oferta (S) i demanda (D), que depenen de la quantitat q:

$$S(q) = 10 + \frac{3}{q}$$

$$D(q) = 100 \cdot q^2$$

1. Representeu gràficament les funcions d'oferta i de demanda.

El primer pas és introduir les funcions analíticament:

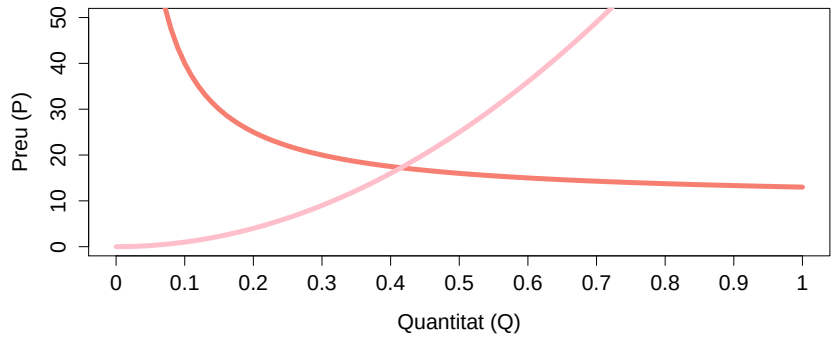
```
> S <- function(q) 10 + 3/q
> D <- function(q) 100*q^2
```

De manera semblant als casos que hem estudiat més amunt, la representació gràfica s'inicia amb la creació del vector de quantitats inicials q_0. La figura 7 en mostra el resultat.

```
> q0 <- seq(0,1,by=0.01)

> s0 <- S(q0)
> d0 <- D(q0)

> xrange <- c(1,length(q0))
> yrange <- c(0,50)

> plot(xrange,yrange,type="n",xaxt="n",
```

```
+      xlab="Quantitat (Q)",ylab="Preu (P)")
> lab <- 1+10*0:10
> axis(1,at=lab,labels=q0[lab])
> lines(s0,type="l",lwd=5,col="salmon")
> lines(d0,type="l",lwd=5,col="pink")
```

Figura 7. Funcions d'oferta i demanda



2. Quins són la quantitat i el preu d'equilibri?

R no ofereix (encara) la possibilitat de sumar i restar funcions simbòlicament, de manera que hem d'obtenir aquestes magnituds a partir dels vectors `s0` i `d0`. Si calculem la diferència `s0-d0`, obtindrem un vector amb la diferència de les funcions. El valor més proper a zero en aquest vector (en valor absolut) correspondrà al punt d'equilibri:

```
> dif <- abs(s0-d0)

> which.min(dif)
[1] 43
```

Aquest resultat indica que el valor núm. 43 del vector `q0` correspon a la quantitat d'equilibri. El preu d'equilibri el podem obtenir de les funcions d'oferta i demanda indistintament. Com veiem, la quantitat i el preu d'equilibri seran aproximadament $q^* \approx 0,42$ i $p^* \approx 17,15$.

```
> dif <- abs(s0-d0)

> v.min <- which.min(dif)

> (q.equil <- q0[v.min])
[1] 0.42

> (p.equil <- s0[v.min])
[1] 17.14286
```

3. Càlcul diferencial i integral

3.1. Aproximació a la derivada

L'objectiu d'aquesta secció és estudiar les possibilitats que R ofereix per al càlcul diferencial. Un aspecte destacable és que R no ofereix la potència d'altres programes (com *Mathematica*) pel que fa al càlcul simbòlic de derivades i integrals. Això no obstant, treballant amb un vector de dades inicials, R ens permet calcular rectes, corbes, secants, tangents i àrees aplicant tècniques d'anàlisi numèrica.

Càlcul

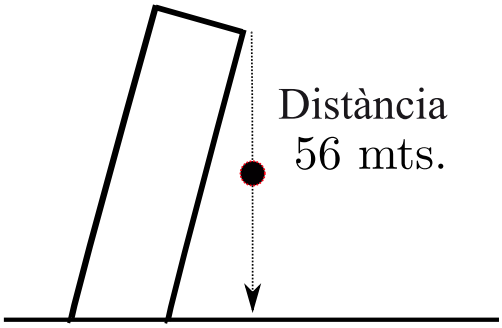
La paraula *càlcul* prové del llatí *calculus*, que era una pedra petita utilitzada per a comptar. El càlcul és l'estudi matemàtic del canvi, i té dues branques principals: el *càlcul diferencial*, que estudia les ràtios de canvi i els pendents de les corbes, i el *càlcul integral*, que s'encarrega de l'acumulació de quantitats i de les àrees que hi ha sota les corbes. Totes dues branques estan relacionades pel *teorema fonamental del càlcul*.

Per a aproximar el concepte de derivada, considerarem un exemple de física. El científic italià Galileo Galilei va estudiar la caiguda d'objectes per a descriure'n el comportament, i va arribar a la conclusió que la velocitat de caiguda no depenia de la massa de l'objecte. Va proposar l'equació següent:

$$s(t) = 4,9t^2$$

Aquesta és una *equació de moviment*, en la qual la distància recorreguda de l'objecte (s) és una funció del temps de caiguda (t). La forma quadràtica de l'equació suggereix que l'objecte cau més i més de pressa a mesura que augmenta el temps de caiguda[1]. Galileo Galilei va dur a terme una sèrie d'experiments deixant caure objectes des de dalt de la torre de Pisa (56 m d'altura), com mostra la il·lustració de la figura 8.

Figura 8. Experiments de Gal·lileu



Galileo Galilei (1564-1642)

Físic, matemàtic, astrònom i filòsof italià que ha tingut un paper fonamental en la història de la ciència. Es considera el pare de l'astronomia i de la física moderna, i en general el pare de la ciència moderna.

El primer pas per a analitzar aquest fenomen en R és introduir la funció de la caiguda, que anomenarem `s`:

```
> s <- function(t) 4.9*t^2
```

[1] Experiments posteriors han demostrat que arriba un moment en què l'objecte en caiguda lliure assoleix una velocitat terminal, a partir de la qual aquesta és constant i l'acceleració és nul·la.

Una primera dada fonamental que hem de conèixer és el temps total ($t = T$) que transcorre des que l'objecte cau de dalt de la torre fins que arriba a terra. Algebraicament, això és fàcil, n'hi ha prou de fer el càlcul següent:

$$56 = 4,9T^2 \Rightarrow T = \sqrt{\frac{56}{4,9}} \approx 3,38\ s$$

En R aquest càlcul és immediat: n'hi haurà prou de buscar el valor de t^* que compleixi la condició $s(t^*) - 56 = 0$, la qual cosa es fa buscant l'arrel d'aquesta expressió:

```
> (T <- uniroot(function(t) 4.9*t^2 - 56,c(0,5))$root)
[1] 3.380618
```

Un cop calculat el valor de T, podem visualitzar la funció en el domini que ens interessa, això és, $t \in (0, T)$. A més, també representarem una línia recta (vermella i discontínua) que unirà els punts $(0, 0)$ i $(T, 56)$. Es tracta d'una recta secant, ja que talla la paràbola per dos punts. El pendent de la recta secant és constant, i correspon a la *velocitat mitjana* de l'objecte des que és llançat fins que toca el terra. Això és, un objecte que recorre 56 m en $3,38$ s ha viatjat a una velocitat mitjana de $\bar{v} = 56/3,38 = 16,56$ m/s.

```
> plot(s,xlim=c(0,T),lwd=4,col='blue')
> lines(c(0,T),c(0,56),lty=2,lwd=4,col='red')
```

Figura 9. Funció del moviment i recta secant

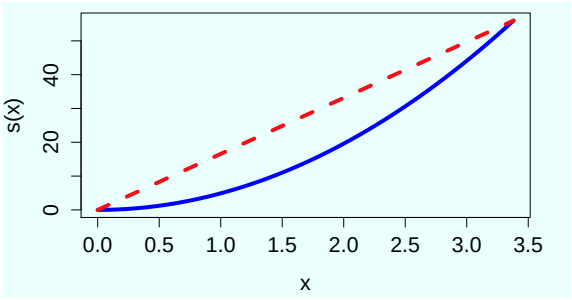


Figura 9

Fixeu-vos com la línia contínua mostra la distància que ha recorregut l'objecte en funció del temps. La línia secant (discontínua) mostra quin hauria estat el recorregut de l'objecte si la velocitat hagués estat constant durant tota la caiguda. De fet, el pendent de la línia discontínua coincideix amb la velocitat mitjana de l'objecte durant la caiguda.

Això no obstant, la velocitat no és constant al llarg de la funció. Per a obtenir la velocitat específica en un tram determinat de la caiguda, haurem de calcular la recta secant entre dos punts. Això és, la velocitat mitjana entre els moments t_0 i t_1 es calcula a partir de la recta que passa pels punts $(t_0, s(t_0))$ i $(t_1, s(t_1))$, tal com mostra la figura 10:

Figura 10. Velocitat mitjana entre dos punts

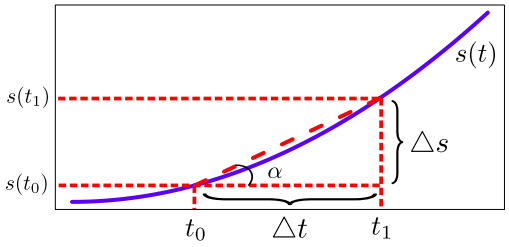


Figura 10

Aquest gràfic il·lustra el concepte de derivada, la qual seria el pendent de la línia vermella discontínua quan $\Delta t = t_1 - t_0$ tendeix a zero. En aquest cas, la derivada mostra quina és la velocitat instantània de l'objecte en un moment exacte del temps, no en un interval.

Matemàticament, la velocitat mitjana entre t_0 i t_1 correspon al pendent de la recta entre els dos punts, és a dir, la tangent de l'angle α:

$$\tan\alpha = \frac{s(t_1) - s(t_0)}{t_1 - t_0} = \frac{\Delta s}{\Delta t}$$

Arribats a aquest punt, com calcularíem la *velocitat instantània* en un punt? Si fem el càlcul anterior reduint cada vegada més Δt, en el límit la recta secant s'aproxima a una recta tangent en un punt determinat, amb la qual cosa obtindrem la velocitat per a un valor de t en comptes d'un interval. Matemàticament, la derivada de la funció $s(t)$ en el punt t_0 està determinada pel límit següent:

$$s'(t_0) = \lim_{\Delta t \to 0} \frac{s(t_0 + \Delta t) - s(t_0)}{\Delta t}.$$

En què $\triangle t$ també s'anomena *diferencial de t* i s'expressa com a dt. La derivada es pot generalitzar com una funció $s'(t)$ derivada de l'original $s(t)$, que indica, per a cada valor de t, el pendent de la funció original $s(t)$. En R es pot calcular la funció derivada de funcions senzilles de manera simbòlica, mitjançant les funcions `deriv` i `D`:

```
> D(expression(4.9*t^2),"t")
4.9 * (2 * t)
```

Això no obstant, si seguim una aproximació d'anàlisi numèrica, per a calcular el valor de la derivada per a un valor específic de t, el que podem fer és definir un valor arbitrari de dt prou petit i aplicar la fórmula que hem descrit abans. Per exemple, la velocitat de l'objecte al cap de dos segons de caiguda ($t = 2$) és:

```
> dt <- 0.0001
> t0 <- 2
> (s(t0+dt)-s(t0))/dt
[1] 19.60049
```

Això és, aproximadament $19,6$ m/s. Aquest càlcul es pot generalitzar per a calcular empíricament la primera derivada en el domini $t \in (0, T)$. El procediment és senzill: atribuïm a dt un valor arbitràriament petit i creem un vector de valors inicials $t_0 = (0, dt, 2dt, \ldots, T)$. Tot seguit avaluem la funció amb tots els valors del vector, això és, $s(t_0) = (s(0), s(dt), s(2dt), \ldots, s(T))$. Amb això ja tenim els valors de la funció $s(t)$, i per a calcular els valors de la funció derivada $s'(t)$ calcularem un vector amb els increments de $s(t)$:

$$[s(dt) - s(0),\ s(2dt) - s(dt), \ldots, s(T) - s(T - dt)]$$

Aquest vector el dividirem entre dt, amb la qual cosa obtindrem el vector amb els pendents en cada punt `ds.dt`. El resultat es pot veure en la figura 11.

```
> dt <- 0.01
> t.0 <- seq(0,3.38,by=dt)
> n <- length(t.0)

> f.s <-s(t.0)
> inc.s <- f.s[2:n] - f.s[1:(n-1)]
> ds.dt <- inc.s/dt

> xrange <- c(1,n)
> yrange <- c(0,max(f.s))

> plot(xrange,yrange,type='n',xaxt='n',
+      xlab='Temps (t)',ylab='Distància (s)')
> lab=1+50*0:6
> axis(1,at=lab,labels=t.0[lab])
>
> lines(f.s,type='l',lwd=4,col='blue')
> lines(ds.dt,type='l',lwd=4,col='red')

> legend('bottomright',c('s(t)','ds/dt'),
+        fill=c('blue','red'),cex=1.5)
```

Figura 11. Funció original i primera derivada

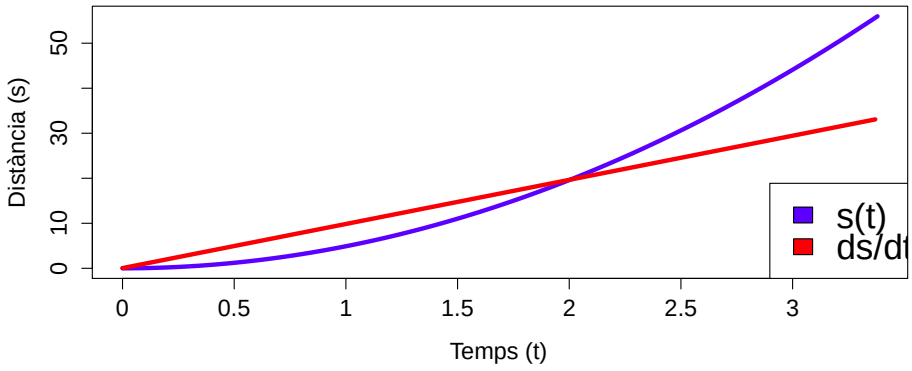


Figura 11

Aquí podem veure com l'objecte augmenta la velocitat de manera progressiva fins a arribar a terra a 33 m/s. aproximadament. En altres paraules, l'objecte té una acceleració positiva i constant, ja que la segona derivada de $s(t)$ és igual a zero (és a dir, el pendent de la derivada no varia).

3.2. Integració

La integració es pot entendre com el procés invers a la diferenciació. Continuant amb l'exemple anterior, hem vist que la funció de moviment $s(t)$ té una derivada $s'(t) = ds/dt$ que representa la velocitat instantània en cada moment t de l'objecte. Suposem que disposem de la funció de la velocitat $s'(t)$ i volem obtenir la funció de la distància total $s(t)$. La integració ens permet revertir el procés de diferenciació: $s(t) = \int s'(t)dt$. Vist d'una altra manera, mentre que la derivació es basa a calcular pendents ($\Delta s/\Delta t$), la integració consisteix a calcular àrees ($\Delta s\ \Delta t$).

Per a il·lustrar aquest concepte, continuarem amb l'exemple de la caiguda d'objectes al buit. La fórmula que hem utilitzat en l'exemple anterior correspon a una aproximació de Galileo Galilei, vàlida per a distàncies petites. Això no obstant, hi ha molts determinants que afecten la velocitat d'un objecte en caiguda lliure: l'acceleració deguda a la gravetat, l'àrea de l'objecte, la densitat de l'aire, la resistència aerodinàmica, etc. Una aproximació més exacta proposa que la velocitat d'un objecte en caiguda lliure augmenta fins a aproximar-se asimptòticament a una *velocitat terminal* v_∞, la qual depèn de molts factors. Així, la fórmula de la velocitat es defineix com[2]:

$$v(t) = v_\infty \tanh\left(\frac{gt}{v_\infty}\right)$$

En què $g = 9,81$ m/s^2 és l'acceleració gravitacional i v_∞ és la velocitat terminal, la qual pren el valor aproximat de 54 m/s en el cas d'un paracaigudista mitjà. El primer pas és introduir la funció de la velocitat en R i representar-la gràficament en la figura 12:

```
> f.v <- function(t) 54*tanh(9.81*t/54)

> plot(f.v,xlim=c(0,20),lwd=4,col='magenta',
+      xlab='Temps (t)',ylab='Velocitat (v)')
```

Figura 12. Velocitat d'un paracaigudista en caiguda lliure

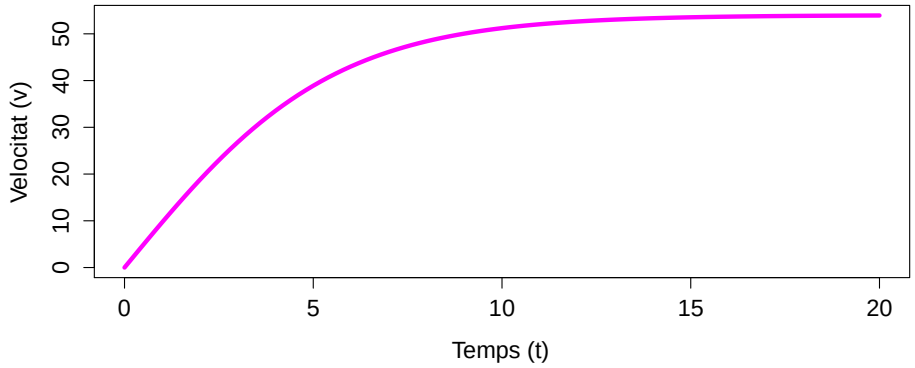


Figura 12

Aquest gràfic mostra la velocitat d'un paracaigudista en caiguda lliure assumint una densitat de l'aire constant. El famós salt estratosfèric de **Felix Baumgartner** el 14 d'octubre de 2012 va assolir una velocitat molt més gran en saltar d'una altura de 39 km, en què l'aire ofereix molta menys resistència.

Aquest gràfic es pot complementar representant l'acceleració, és a dir, el canvi de velocitat per unitat de temps. Matemàticament es defineix com la derivada de la funció de velocitat $v(t)$. Aprofitant la funció `D` que ens ofereix R, obtenim $v'(t)$ i hi donem el nom de `f.a`:

```
> D(expression(54*tanh(9.81*t/54)),"t")
> 54 * (9.81/54/cosh(9.81 * t/54)^2)
> f.a <- function(t) 54 * (9.81/54/cosh(9.81 * t/54)^2)
```

[2] Recordem que la funció tangent hiperbòlica es defineix com a $\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

Ara podem avaluar totes dues funcions en el domini $t_0 \in (0, 20)$, el resultat de les quals es mostra en la figura 13. L'acceleració al principi és aproximadament $v'(0) = 10$ i a mesura que el temps avança s'apropa asimptòticament a 0.

```
> t.0 <- 0:20

> plot(f.v(t.0),type='l',lwd=4,col='magenta',
+      xlab='Temps (t)',ylab='')

> lines(f.a(t.0),lwd=4,col='seagreen')

> legend('right',
+ c('Vel. (v(t))','Acc. (dv/dt)'),
+ fill=c('magenta','seagreen'),cex=1.5)
```

Figura 13. Velocitat i acceleració d'un paracaigudista en caiguda lliure

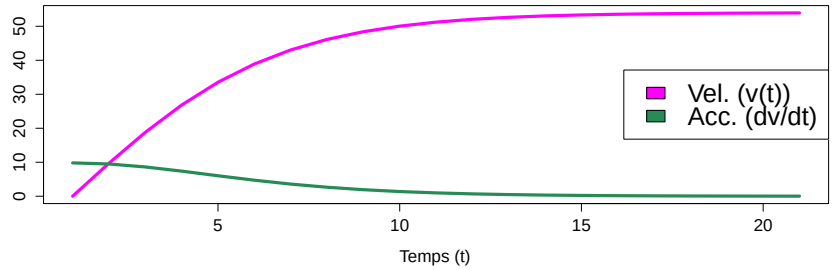


Figura 13

Recordem que, en física, l'acceleració indica el canvi de velocitat per unitat de temps.

Si estiguéssim interessats en la distància que recorre el paracaigudista hauríem de fer l'operació inversa, és a dir, aproximar la funció de moviment a partir de la funció de velocitat. Suposem que estem interessats en la distància que ha recorregut el paracaigudista durant els 10 primers segons de caiguda. Una opció és registrar la velocitat en intervals de dos segons i multiplicar-la per aquest interval $\Delta t = 2$. El resultat es mostra en la taula 3.

Taula 3. Aproximació de la distància a partir de la velocitat

j	t_j	$v(t_j)$	$\Delta t \cdot v(t_j)$
1	2	18,80	37,70
2	4	33,53	67,07
3	6	43,03	86,06
4	8	48,40	96,80
5	10	51,22	102,43
Distància total:			389,98

En R, reproduir els càlculs de la taula 3 és immediat:

```
> dt <- 2

> t.0 <- dt*1:5

> (v.0 <- f.v(t.0))
```

```
[1] 18.79992 33.53518 43.03142 48.40290 51.21927

> sum(2*v.0)
[1] 389.9774
```

La figura 14 ofereix una interpretació geomètrica dels càlculs de distància que s'han vist en la taula 3:

Figura 14. Càlcul aproximat de la distància

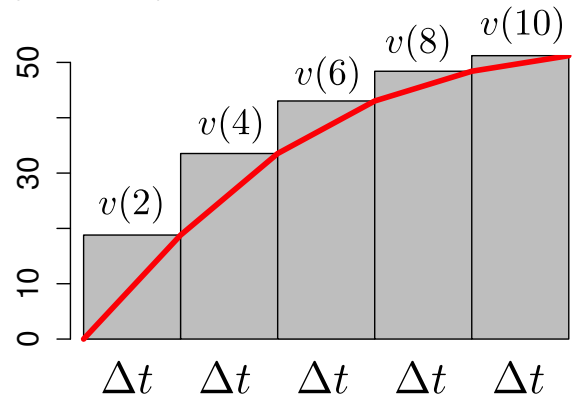


Figura 14

És important comprovar que la suma de les àrees de les cinc barres verticals es correspon amb els cinc valors que s'han obtingut en la taula 3, la suma dels quals és aproximadament de 390 m. Vegem també que aquesta quantitat aproximada és superior a la real, ja que la suma d'aquestes àrees és superior a l'àrea de sota de la corba.

El càlcul de l'àrea que hem fet es denomina *suma de Riemann*, i correspon a un mètode per a aproximar l'àrea sota una corba:

$$S = \sum_{j=1}^{n} v(t_j) \cdot \Delta t$$

En la figura 14 hem vist una aproximació amb $\Delta t = 2$ $n = 5$, de manera que no és una aproximació gaire exacta. Es pot comprovar que l'àrea gris és superior a l'àrea que queda per sota de la funció $v(t)$, amb la qual cosa la distància de 389, 98 m és superior a la real. Obtindrem un càlcul més exacte si fem que $\Delta t \to 0$ i $n \to \infty$. Això ens deixa pas a la definició de la *integral de Riemann*: per a calcular l'àrea sota una corba entre els punts a i b, calculem una partició de l'interval $[a, b]$ tal que $a = t_0 < t_1, \ldots, t_n < b$, essent Δt la distància entre els valors de l'interval.

$$S = \lim_{\Delta \to 0} \sum_{j=1}^{n} v(t_j) \cdot \Delta t = \int_a^b v(t)dt$$

La integral de Riemann és, doncs, el límit d'una suma. Per a calcular-la en R crearem una funció a la nostra mida:

```
> int.riemann <- function(fun,a,b,dt){
+     t.0 <- seq(a,b,by=dt)
+     f.0 <- fun(t.0)
+     suma <- sum(f.0*dt)
```

```
+     return(suma)
+ }
```

En el cas anterior havíem calculat l'àrea en l'interval $[0, 10]$ amb $\Delta t = dt = 2$:

```
> int.riemann(f.v,0,10,2)
[1] 389.9774
```

Aquest resultat serà més i més exacte a mesura que $dt \to 0$:

```
> int.riemann(f.v,0,10,1)
[1] 366.5903

> int.riemann(f.v,0,10,0.1)
[1] 344.2708

> int.riemann(f.v,0,10,0.01)
[1] 341.9732

> int.riemann(f.v,0,10,0.001)
[1] 341.7428
```

Hem triat aquesta manera d'aproximar la integral amb l'objectiu d'il·lustrar el concepte d'integral. En general, hi ha diversos mètodes per a aproximar l'àrea sota una corba, i R incorpora una funció anomenada `integrate`, la qual aproxima l'àrea sota una corba utilitzant una quadratura adaptativa amb una tolerància d'error d'aproximadament $1,22 \times 10^{-4}$.

```
> integrate(f.v,0,10)
341.7172 with absolute error < 3.8e-12
```

Calculant la integral d'aquesta manera, es comprova que la distància real s'aproxima a 341,71 m. Per tant, la funció d'*R* `integrate` és la que millor aproxima aquesta quantitat.

Les equacions diferencials

Aquestes equacions tenen un paper destacat en l'enginyeria, la física, l'economia i altres disciplines. Sorgeixen quan hi ha una relació determinista entre variables contínues (funcions) i les seves taxes de canvi en l'espai i/o temps (les seves derivades).

3.3. Equacions diferencials

Les **equacions diferencials** són equacions matemàtiques que contenen una funció desconeguda (d'una o més variables), a més d'una o més derivades d'aquesta funció. L'**ordre** d'una equació diferencial serà l'ordre de la derivada més alta en l'equació, i la **solució** serà una funció f si l'equació se satisfà quan $y = f(x, \ldots)$.

Les equacions diferencials més senzilles es poden resoldre utilitzant les regles de diferenciació i integració. Vegem com es resoldria l'equació diferencial $y' = cx$:

$$y' = cx \qquad \Rightarrow \qquad \frac{dy}{dx} = cx$$

$$dy = cx\,dx \qquad \Rightarrow \qquad \int dy = \int cx\,dx$$

$$y = \frac{cx^2}{2} + C$$

Això no obstant, resoldre equacions diferencials més complexes no és una tasca gens fàcil. En aquest sentit, R no és un programa capaç de resoldre equacions diferencials trobant la solució general de manera simbòlica. Tot i així, R és capaç de trobar una solució particular que satisfaci una condició del tipus $y(x_0) = y_0$. Aquesta es denomina una **condició inicial**, i el problema de trobar una solució per a l'equació diferencial que satisfaci aquesta condició inicial es denomina **problema de valor inicial**.

El paquet d'R que resol aquesta mena de problemes es denomina **deSolve**, el qual s'ha d'instal·lar prèviament[3]. És un paquet molt complet, i en la pàgina web del CRAN es pot trobar un programa d'aprenentatge molt complet sobre com aplicar-ne les diferents aplicacions. A més, si accedim a *cran.r-project.org / Packages / CRAN Task Views*, veurem una categoria anomenada *Differential Equations*, en la qual hi ha disponibles altres paquets que desenvolupen altres funcionalitats.

Considerem un exemple del camp de la psicologia: l'estudi de les **corbes d'aprenentatge**. Per a un estudiant que està aprenent R, la corba d'aprenentatge és una funció $Y(t)$, en què Y és el seu rendiment, el qual depèn del temps (t). Un model recurrent d'aprenentatge està determinat per l'equació diferencial següent:

$$\frac{dY}{dt} = k(M - Y)$$

En què dY/dt és la velocitat d'aprenentatge (la ràtio a la qual millora el rendiment), k és una constant i M és el nivell màxim al qual pot arribar un estudiant, de manera que sempre es compleix que $Y \leq M$. La *condició inicial* que fixem per a aquest problema és que $Y(t_0) = 0$, és a dir, que l'estudiant al principi no sap res d'R.

Per a aproximar-nos a aquest problema, hem d'introduir la condició inicial i els paràmetres, a més del període temporal en el qual volem estudiar l'equació. Assumirem que l'escala d'aprenentatge va de 0 a 10, de manera que $M = 10$, i que $k = 0,02$. A més, estudiarem la progressió de l'aprenentatge en un any, de manera que $t_0 = 0, \ldots, 365$ dies:

[3] Això és tan fàcil com introduir en la consola `install.packages("deSolve")`.

```
> library(deSolve)

> parametres<-c(k=0.02,M=10)

> cond.ini <- c(Y=0)

> temps <- 0:365
```

Noms de les variables

Fixeu-vos que, introduint els valors en vectors d'aquesta manera, assignem un nom a cada valor, la qual cosa és important per als càlculs posteriors.

El pas següent és introduir l'equació diferencial com una funció. És fonamental introduir-la d'una manera específica, tal com es mostra a continuació:

```
> f.apren <- function(t, cond.ini, parametres){
+     with(as.list(c(cond.ini, parametres)),{
+       dY <- k*(M-Y)
+       return(list(dY))
+     })}
```

Això és, els arguments de la funció són el temps, les condicions inicials i els paràmetres. A més, amb l'opció `with` incloem els valors dels vectors `cond.ini` i `paràmetres` creats anteriorment. Per acabar, és fonamental que l'*output* d'aquesta funció sigui una llista, tal com mostra la instrucció `return(list(dY))`. Un cop fet això, ja podem aproximar la solució a l'equació diferencial mitjançant la funció `ode`:

Compte! Si no hem instal·lat i/o carregat el paquet `deSolve` correctament, R no reconeixerà la funció `ode` i obtindrem un missatge d'error.

```
> sol <- ode(cond.ini,temps,f.apren,parametres)
```

El resultat serà una matriu amb dues columnes i tantes files com valors de *t*. Vegem les parts superior i inferior d'aquest resultat:

```
> head(sol)
     time         Y
[1,]    0 0.0000000
[2,]    1 0.1980142
[3,]    2 0.3921054
[4,]    3 0.5823544
[5,]    4 0.7688362
[6,]    5 0.9516254

> tail(sol)
       time        Y
[361,]  360 9.992534
[362,]  361 9.992682
[363,]  362 9.992827
```

```
[364,]  363 9.992969
[365,]  364 9.993108
[366,]  365 9.993245
```

La interpretació dels resultats d'una equació diferencial es fa gràficament. En el nostre cas, una representació en un pla bidimensional de les variables Y i t, que es mostra en la figura 15.

```
> plot(sol,xlab="temps (t)",ylab="Aprenentatge (Y)",
+ lwd=4,col="red",main="Corba d'aprenentatge")
```

Figura 15. Solució d'una equació diferencial

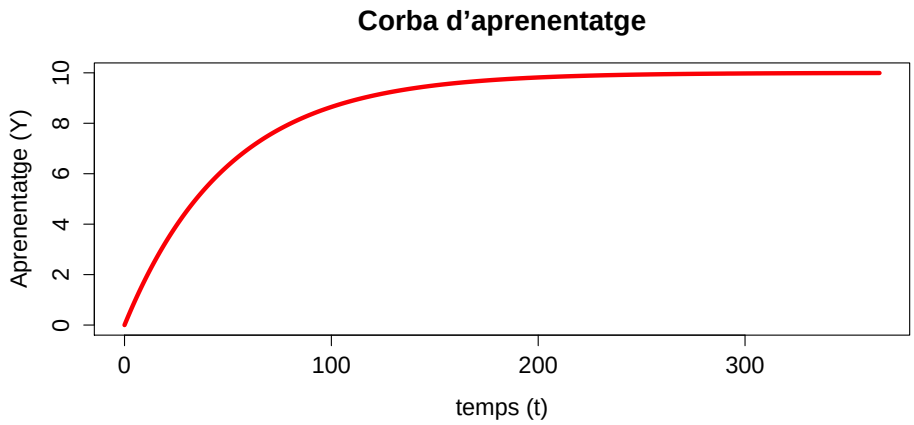


Figura 15

En aquesta corba es pot veure com en els primers dies el valor de la derivada de la corba(dY/dt) és molt gran, això és, s'aprèn molt ràpidament. A mesura que es consoliden els coneixements, aquesta derivada es redueix, ja que l'aprenentatge tendeix al seu límit superior $Y = 10$.

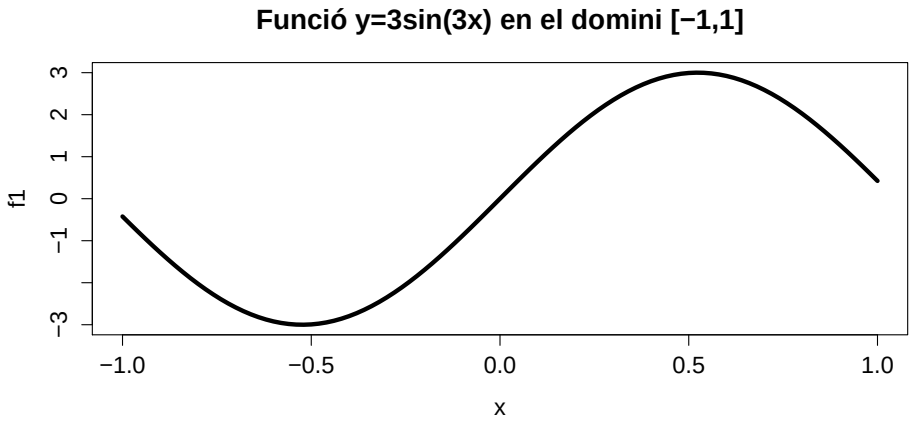
Cal destacar que aquest és un exemple molt senzill. R i el paquet `deSolve`, a més d'altres paquets destinats a equacions diferencials, ofereixen moltes possibilitats per a resoldre sistemes d'equacions diferencials i altres problemes més complexos.

4. Optimització

En general, la funció d'R `optimize` busca, per a una funció, el mínim i el màxim en l'interval del domini $[a, b]$ respecte al seu primer argument. Considerem la senzilla funció $y = 3sin(3x)$: la introduirem i la representarem gràficament en el domini $[-1, 1]$ en la figura 16:

```
> f1 <- function(x) 3*sin(3*x)

> plot(f1,from=-1,to=1,lwd=4)

> title("Funció y=3sin(3x) en el domini [-1,1]")
```

Figura 16. Representació de la funció $y = 3sin(3x)$



Per a trobar el valor mínim d'aquesta funció en l'interval $[-1, 1]$ n'hi haurà prou d'introduir la instrucció següent:

```
> optimize(f1,c(-1,1))
$minimum
[1] -0.5235812

$objective
[1] -3
```

Vegem com el resultat final és una llista amb dos elements: el valor de x en què hi ha el mínim $(-0,523)$ i el valor de la funció $f(x)$ en aquest punt, és a dir, -3. Si

volem buscar el màxim de la funció en comptes del mínim, n'hi haurà prou d'introduir l'opció `maximum=TRUE`, com es mostra en l'exemple següent:

```
> optimize(f1,c(-1,1),maximum=TRUE)
$maximum
[1] 0.5235812

$objective
[1] 3
```

En general, per a funcions més complexes i de més d'un argument, hi ha la funció `optim`. Considerem com a exemple la funció amb dos arguments $y = f(x_1, x_2) = 2x_1^2 + 2x_2^2 - 4$. Per a buscar el mínim d'aquesta funció l'haurem d'introduir en R d'una manera lleugerament diferent, així:

```
> f.y <- function(x){
+   x1 <- x[1]
+   x2 <- x[2]
+   2*x1^2 + 2*x2^2 - 4
+ }
```

Arguments de la funció

En aquesta funció, els dos arguments de la funció (x_1 i x_2) estan introduïts conjuntament en un sol vector x, que conté les dues variables.

Per a buscar el mínim d'aquesta funció, haurem d'introduir uns valors inicials de x_1^* i x_2^*, ja que la funció `optim` aplicarà un procés iteratiu a partir d'aquests valors per buscar la convergència en els valors de x_1 i x_2 que minimitzin la funció y. En aquest exemple, provarem aleatòriament amb els valors $x_1^* = -1$ i $x_2^* = 1,5$:

```
> optim(c(-1,1.5),f.y)
$par
[1]  9.038020e-05 -4.059674e-05
$value
[1] -4
$counts
function gradient
      69       NA
$convergence
[1] 0
$message
NULL
```

La funció `optim`

Tècnicament, aquesta funció es basa en els algorismes de gradient conjugat, Nelder-Mead i quasi-Newton. Per a processos d'optimització més complexos, es recomana veure l'ajuda d'aquesta funció escrivint `help(optim)` i considerar les diferents opcions que aquesta funció ofereix.

El resultat que ens interessa d'aquesta llista és `par`, el qual mostra un vector amb dos valors en notació científica, molt propers a zero. Així que podem assumir que els valors $x_1 = 0$ i $x_2 = 0$ constitueixen un mínim, essent el valor de la funció $y(0, 0) = -4$.

Les opcions d'optimització amb R no s'acaben aquí. Es poden resoldre problemes d'optimització amb restriccions, optimització quadràtica, etc. En la pàgina web del CRAN hi ha disponible una llista de categories de paquets per disciplines:

cran.r-project.org / Packages / CRAN Task Views

Aquí trobarem un element anomenat *Optimization*, i si hi accedim veurem els diferents paquets disponibles i les seves capacitats.

5. Anàlisi de variable complexa

En matemàtiques, molt sovint ens trobem amb la necessitat de resoldre equacions del tipus $x^2 = -1$. Sabem que no hi ha cap nombre real el quadrat del qual sigui un nombre negatiu, però això no significa que no hi hagi cap nombre que satisfaci aquesta equació. La resposta és en els **nombres complexos**. Aquests es basen en el nombre $i = \sqrt{-1}$, el qual satisfà que $i^2 = (\sqrt{-1})^2 = -1$. El nombre i rep el nom de **nombre imaginari**.

Hi ha dues maneres d'expressar nombres complexos. La primera és la **manera binòmica**, això és $a + bi$, essent a i b nombres reals. Aleshores es diu que a és la **part real** del nombre complex, i bi és la **part imaginària**. Com hem vist anteriorment, a l'hora de buscar les arrels d'una equació, R ens dóna nombres en notació complexa. Per exemple, l'equació $x^2 - 10x + 40 = 0$, les arrels de la qual són $x^* = 5 \pm \sqrt{-15} = 5 \pm \sqrt{15}i$:

```
> polyroot(c(40,-10,1))
[1] 5+3.872983i 5-3.872983i
```

Evidentment, també és possible introduir nombres complexos i fer-hi operacions. Vegem dues maneres d'introduir el nombre $z = 10 + 5i$:

```
> (z <- 10-5i)
[1] 10-5i

> (z <- complex(real=10,imaginary=-5))
[1] 10-5i
```

Vegem com es poden extreure i aïllar dels nombres complexos la part real (`Re`) i la imaginària (`Im`).

```
> Re(z)
[1] 10

> Im(z)
[1] -5
```

Quant a les operacions algebraiques permeses, la suma (resta) de nombres complexos s'efectua sumant (restant) les parts reals i les parts imaginàries per separat:

```
> z1<-5+1i
> z2<-10+4i

> z1+z2
[1] 15+5i

> z1-z2
[1] -5-3i
```

Quant a la multiplicació d'un nombre complex per un nombre real, multipliquem les parts real i imaginària del nombre complex pel nombre real. Per a multiplicar dos nombres complexos, hem d'aplicar la propietat distributiva dels polinomis, tenint en compte que $i^2 = -1$. I quant a la seva divisió, multiplicarem i dividirem pel conjugat del nombre complex del denominador:

```
> 5*z1
[1] 25+5i

> z1*z2
[1] 46+30i

> z1/z2
[1] 0.4655172-0.0862069i
```

Una transformació d'interès és el **conjugat** d'un nombre complex $z = a + bi$, el qual és un altre nombre complex anomenat $\bar{z}$ que té la part imaginària canviada de signe. Una de les seves propietats és que multiplicar un nombre complex pel conjugat corresponent resulta en un nombre real. En R, el conjugat s'extreu mitjançant l'operador `Conj`:

```
> (z3<-4+2i)
[1] 4+2i

> (z4<-Conj(z3))
[1] 4-2i

> z3*z4
[1] 20+0i
```

Una manera alternativa d'expressar els nombres complexos és amb **forma polar**, tal com es mostra en la figura 17:

Figura 17. Nombre complex en forma polar

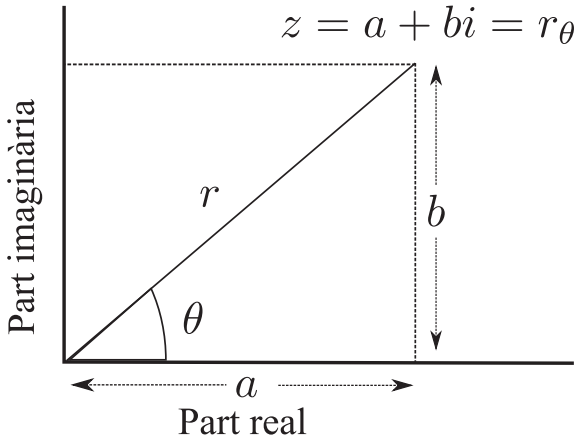


Figura 17

El nombre complex $z = a + bi$ també es pot expressar en forma polar. Per a això es consideren dos valors: r és el **mòdul**, que és la distància del punt a l'origen de les coordenades (a, b); i θ és l'**argument**, que és l'angle que formen aquestes coordenades amb l'eix d'abscisses. D'aquesta manera, en forma polar tenim que $z = r_\theta$.

A partir d'un nombre complex en forma binòmica $z = a + bi$, l'expressió en forma polar s'obté aplicant les fórmules següents:

$$r = \sqrt{a^2 + b^2}$$

$$\theta = \arctan\frac{a}{b} \qquad \text{si } a > 0$$

$$\theta = \arctan\frac{a}{b} + \pi \qquad \text{si } a < 0,\ b > 0$$

$$\theta = \arctan\frac{a}{b} - \pi \qquad \text{si } a < 0,\ b < 0$$

En tots els casos, obtindrem un angle θ dins de l'interval $[-\pi, \pi]$. En R, introduirem nombres complexos en forma polar de la manera següent:

```
> (z5<-complex(modulus=sqrt(2),argument=pi/4))
[1] 1+1i
```

En aquest cas, veiem que $a = b = 1$, ja que $\sqrt{1^2 + 1^2} = \sqrt{2}$ i $\tan(\pi/4) = 1$.

Finalment, veurem com es calcula l'arrel n-èsima de qualsevol nombre. Considerant l'equació d'Euler, $z^n = re^{\theta i}$ tindrà n solucions si també considerem nombres complexos:

$$z = \sqrt[n]{r}e^{(\frac{\theta}{n} + \frac{2\pi}{n}k)i}, \qquad k = 0, 1, \ldots, n-1.$$

Sabent que $e^{\pi i} = \cos(\pi) + i\sin(\pi)$, podem programar una funció que calculi les n arrels complexes de qualsevol nombre real. Anomenarem la funció `arrel.comp`, i tindrà

dos arguments: el nombre del qual volem l'arrel (r) i l'ordre de l'arrel (n). La solució serà un vector de nombres complexos amb les n arrels: $(z_1, \ldots, z_n)$:

```
> arrel.comp <- function(r,n){
+   sol <- array(data=NA, dim=n)
+   m1 <- (abs(r)^(1/n))
+   for (j in 0:n-1){
+     if (r<0){
+       m2 <- (pi/n)+2*j*(pi/n)
+     }else{
+       m2 <- 2*j*(pi/n)
+     }
+   sol[j+1]<-m1*complex(real=cos(m2),imaginary=sin(m2))
+   }
+   return(sol)
+ }
```

Vegem un parell d'exemples. Primer comprovarem que $\sqrt{9}$ té les arrels -3 i 3:

```
> arrel.comp(9,2)
[1]  3+0i -3+0i
```

Ara vegem les solucions de $\sqrt[3]{90}$:

```
> arrel.comp(90,3)
[1] 4.481405+0.00000i -2.240702+3.88101i
   -2.240702-3.88101i
```

Com hem vist en aquest cas, R és un programa molt flexible que ens permet implementar qualsevol tipus de funció o algorisme.

Bibliografia

Gibernans Bàguena, J.; Gil Estallo, À. J.; Rovira Escofet, C. (2009). *Estadística*. Barcelona: Material didàctic UOC.