

Curso de Introducción a la sintaxis de R mediante
RStudio
Sesión 1: Introducción a RStudio

Servei d'Estadística Aplicada
Universitat Autònoma de Barcelona

17, 19, 24 y 26 de enero, 2023

Contenido de la Sesión

- 1 Presentación del software R
- 2 Instalación de R e Inicio de la sesión
- 3 Sintaxis elemental en R
- 4 RStudio
- 5 Ayuda de R
- 6 Presentación de la programación en R
- 7 Paquetes y librerías
- 8 Scripts
- 9 Interfaces de usuario

R

- Entorno interactivo para el análisis estadístico y gráfico.
- Agilidad para implementar y compartir nuevas técnicas.
- Permite generar gráficos de alta calidad.
- Múltiples recursos de soporte y ayuda.
- Elevada presencia en determinados ámbitos de investigación.
- Libre distribución (sin garantías).
- Multiplataforma (Windows, Linux, Mac).

- R es un lenguaje de programación de alto nivel con funciones orientadas a objetos.
- Es un lenguaje interpretado (como Java) y no compilado (como C, Fortran, Pascal, etc.), los comandos escritos con el teclado son ejecutados directamente sin necesidad de construir ejecutables.
- La sintaxis es simple e intuitiva.
- La estructura y facilidad de uso de R permite implementar funciones y rutinas propias a medida que aparecen nuevas necesidades.
- Sinergia con otros lenguajes: C++, Python, Latex.
- Existen programas complementarios tipo GUI (graphical user interface: R Commander, DeduceR, Rattle, RExcel) o IDE (integrated development environment: RStudio, TinnR, emacs, ...).

Instalación de R

- Para instalar R es necesario descargar los ficheros necesarios de la página web del proyecto R: <http://cran.r-project.org>.
- Actualmente (enero 2023), la versión más reciente de R para el entorno Windows es la R-4.2.2.
- Una vez bajado este fichero, hay que ejecutarlo y seguir las instrucciones del programa de instalación.
- Posteriormente se pueden instalar librerías y paquetes complementarios (actualmente, más de 10000 paquetes disponibles).
- Una vez instalado podremos abrirlo desde el listado de programas en el menú inicio. Al abrirse aparece un mensaje de bienvenida indicando la versión instalada.
- No es necesario visitar constantemente la página del proyecto para comprobar si existen actualizaciones disponibles.
- Si fuera necesario instalar una nueva versión, no es necesario desinstalar las versiones antiguas, se pueden tener instaladas varias versiones del programa.

- La ventana principal corresponde a la consola de R (ventana de comandos) en la cual se pueden escribir comandos y en la que aparecerán los resultados de los análisis o tareas.
- Los comandos se escriben en el margen izquierdo tras el símbolo ">" o PROMPT.
- Para ejecutar un comando escrito utilizamos la tecla de "intro".
- Algunos comandos iniciales de interés son:
 - `demo()` permite realizar demostraciones del uso de R.
 - `license()`: R es un software libre y viene sin GARANTÍA ALGUNA, aunque es posible redistribuirlo bajo ciertas circunstancias.
 - `citation()`: referencia para la citación de R o paquetes en publicaciones científicas.

- Es recomendable indicar el directorio de trabajo asociado a cada proyecto.
- Se puede cambiar de directorio desde el menú **Archivo** → **Cambiar dir...** o con el comando `setwd("C:/Directorio")`.
- A partir de ahora, si se omite la ruta, R entenderá que el fichero referido se encuentra en el directorio de trabajo.
- El comando `getwd()` permite recuperar qué directorio es el directorio de trabajo actual.

- En la consola, las flechas \uparrow y \downarrow permiten recuperar comandos ejecutados anteriormente.
- Para separar instrucciones en una misma línea se emplea ";".
- Si un comando contiene errores, aparecerán los mensajes pertinentes.
- Si un comando está incompleto, el software quedará a la espera. La tecla ESC restablece el PROMPT.
- La combinación Ctrl+C interrumpe la edición o ejecución en curso.
- La combinación Ctrl+Z elimina la ejecución en curso.
- `q()` es el comando para salir de R (alternativamente, cerrar la ventana).

- Las órdenes elementales de sintaxis consisten en expresiones o en asignaciones.
- Si una orden consiste en una expresión o cálculo, esta se evalúa, se imprime y su valor se pierde. Así, el uso más básico de R sería como calculadora científica. Por ejemplo:

```
> 2+4  
[1] 6
```

- Ejemplo de operadores implementados en R: +, -, * , /, ^ o **, %% (módulo), %/% (división entera).
- Ejemplos de funciones implementadas en R:
`log(x)`, `exp(x)`, `log10(x)`, `sqrt(x)`, `factorial(x)`, `floor(x)`,
`round(x,digits=0)`, `abs(x)`, ...

- Al contrario de las evaluaciones, una asignación, evalúa una expresión, no la imprime y guarda su valor en un objeto.
- Las asignaciones se realizan mediante los signos "<-", "->" o "=":

```
> a <- 6*4+3
> sqrt(a) -> b
> B=log(b)
```

- Para visualizar el contenido de un objeto el comando corresponde al nombre del objeto (o bien poniendo la asignación entre paréntesis):

```
> a
[1] 27
> (sqrt(a) -> b)
[1] 5.196152
```

- La colección de objetos almacenados en cada momento se denomina espacio de trabajo (workspace).
- Desde el menú **Archivo** → **Guardar área de trabajo** o con el comando `save.image()` se puede guardar una imagen del espacio de trabajo realizado hasta ese momento.
- El área de trabajo (Workspace) corresponde a todos los objetos creados durante la sesión. La extensión para estos archivos es `.RData`.
- Podemos retomar el área de trabajo en posteriores sesiones abriendo desde el menú **Archivo** → **Cargar área de trabajo** la imagen anteriormente guardada.
- También es posible guardar todos los comandos ejecutados de la sesión desde el menú **Archivo** → **Guardar Histórico**.

- RStudio es un entorno de desarrollo integrado (IDE) para R. Incluye una consola, editor de sintaxis que apoya la ejecución de código, así como herramientas para el trazado, la depuración y la gestión del espacio de trabajo.
- Para instalar RStudio, descargar el programa desde la página web: <https://www.rstudio.com>.
- Actualmente (enero 2022), la versión más reciente de RStudio para el entorno Windows es 2022.12.0.
- Una vez bajado este fichero, hay que ejecutarlo y seguir las instrucciones del programa de instalación.
- Es necesario tener instalado R para poder utilizar RStudio.

- R dispone de una ayuda muy completa sobre todas las funciones, procedimientos y elementos que configuran el lenguaje.
- Además de las opciones de menú propias de R, desde la ventana de comandos se puede acceder a información específica sobre las funciones de R con el comando `help()` o mediante `?`.

```
> help(log)
> ?log
```

- Para acceder al código de una función de R basta con introducir el nombre de la función sin los argumentos.

```
> log
```

- Cuando no sepamos el nombre exacto del comando o de la función que necesitamos, pero sí el tema sobre el que queremos ayuda (en inglés), podemos utilizar la función `help.search()`.

```
> help.search("data input")
```

y posteriormente:

```
> ?read.table
```

- Es posible acceder a manuales desde la página web de CRAN:
<http://cran.r-project.org/>.
- Algunos de los manuales disponibles son:
 - An Introduction to R: introduce el lenguaje, y presenta R como herramienta para el análisis estadístico y gráfico.
 - R Reference: contiene todos los ficheros de ayuda de la versión estándar de R junto con los de las librerías recomendadas.
 - R Data Import/Export: describe las utilidades para importar y exportar datos en R incluyendo librerías disponibles en CRAN.
 - R Language Definition: documenta el lenguaje R. De gran utilidad cuando se van a programar funciones.
 - Writing R Extensions: cubre cómo crear nuestras propias librerías, escribir los archivos de ayuda y conectar con lenguajes externos (C, C++, Fortran, etc.).
 - R Internals: es una guía para las estructuras internas de R.
 - R Installation and Administration.

- Desde el propio programa, a través del menú Ayuda se puede acceder a las FAQ.
- Bibliografía de R: En la página web del proyecto existe "Contributed Documentation" que además es de libre distribución.
 - John M. Chambers (2008), *Software for Data Analysis: Programming with R*. Springer, New York.
 - Peter Dalgaard (2008), *Introductory Statistics with R*, 2nd edition. Springer.
 - Michael J. Crawley (2007), *The R book*. John Wiley and Sons, Ltd.
- Congresos: useR y R-Hispano.
- The R Journal.
- R-bloggers.

- R es mucho más que una calculadora científica en la que aplicar unos comandos o instrucciones.
- R es un lenguaje Orientado a Objetos (Object-oriented programming, OOP). Bajo este término se esconde la simplicidad y flexibilidad de R.
- R almacena los resultados en objetos para ser observados o analizados posteriormente, produciendo unas salidas mínimas.
- El usuario puede extraer sólo aquella parte de los resultados que le interesa.
- Otros programas como SAS o SPSS proporcionan sin solicitarlo una salida copiosa para cualquier análisis.

- Cuando en el mundo real nos referimos a un objeto significa que hablamos de algo más o menos abstracto que puede ser cualquier cosa.
- En OOP la generalización (o definición) de un objeto se llama Clase.
- Todo objeto tiene atributos o propiedades. Se puede acceder a las propiedades de un objeto mediante distintas funciones.
- Se pueden asociar acciones o métodos a los objetos. Cualquier proceso que implica una acción o pauta de comportamiento por parte de un objeto se define en su clase para que luego pueda manifestarse en cualquiera de sus objetos.

Tipos de objetos

- Vectores: son el tipo básico de objeto en R.
- Matrices: vectores indexados por dos o más índices. Generalizaciones multidimensionales de los vectores.
- Factores: sirven para representar datos categóricos.
- Listas: son una forma generalizada de vector cuyos elementos no tienen por qué ser del mismo tipo y a menudo son a su vez vectores o listas.
- Dataframes: las bases de datos (o data frames) son estructuras similares a una matriz, en que cada columna puede ser de un tipo distinto a las otras.
- Funciones: son también objetos de R que pueden almacenarse en el espacio de trabajo.

Programación orientada a Objetos

- Consecuente con sus orígenes en UNIX, R distingue entre mayúsculas y minúsculas, de tal modo que "A" y "a" son símbolos distintos y se referirán a objetos distintos.
- Los nombres de los objetos pueden contener sólo letras mayúsculas o minúsculas (sin acentos), junto con números y puntos (sin blancos, `_`, `%`, `$`, etc.).
- Los corchetes o dobles corchetes se utilizan para seleccionar partes de un objeto así como el dólar `$` o la arroba `@`.
- Durante una sesión de trabajo con R los objetos que se crean se van almacenando por su nombre.
- La función `objects()` se puede utilizar para obtener los nombres de los objetos almacenados en R. Es equivalente a la función `ls()`.
- Es posible eliminar objetos con el comando `rm()`.

Vectores

- La función principal para definir un vector es a través de sus componentes, con la función concatenar: `c()`.

```
> v <- c(1,2,3,b)
```

```
> v
```

```
[1] 1.000000 2.000000 3.000000 5.196152
```

- La función `length()` muestra el número de componentes de un vector y `NA` identifica los elementos faltantes (missings).
- Para referirnos a la componente *n*ésima del vector "v" escribiremos `v[n]`.

```
> v[4]
```

```
[1] 5.196152
```

Vectores

- Se pueden modificar los elementos de vectores, ampliarlos o borrarlos:

```
> v[2] <- 22
```

```
> v[5] <- a
```

```
> v
```

```
[1] 1.000000 22.000000 3.000000 5.196152 27.000000
```

- Lo que convierte a R en una potente herramienta de trabajo es que está diseñado de forma que la mayoría de operaciones y de funciones están definidas con carácter vectorial, es decir para operar componente a componente.
- Es conveniente explotar esta posibilidad ya que agiliza mucho el tiempo de computación.
- Las funciones estadísticas como la suma, el producto o la media aritmética devuelven un solo valor para cada vector.

Matrices

- La función `matrix(vector,...)` organiza los componentes de un vector en forma de matriz.
- La opción `nrow/ncol` permite especificar el número de filas/columnas. El número de filas/columnas se determina mediante el redondeo por exceso de la longitud del vector entre el número de filas/columnas:

```
> A <- matrix(1:9,ncol=3)
```

- La función `dim(matriz)` muestra el número de filas y columnas de la matriz:

```
> dim(A)
[1] 3 3
```

Matrices

- Los operadores "+", "-", "*" y "/" se pueden aplicar a dos o más matrices de la misma dimensión. La ejecución se realiza componente a componente (el resultado es otra matriz de la misma dimensión):

```
> A*A
      [,1] [,2] [,3]
[1,]    1   16   49
[2,]    4   25   64
[3,]    9   36   81
```

- El producto matricial se denota por %*%:

```
> A%*%A
      [,1] [,2] [,3]
[1,]   30   66  102
[2,]   36   81  126
[3,]   42   96  150
```

Matrices

- Con `A[i,j]`, `A[i,]` y `A[,j]` nos referimos a un elemento, a una fila o a una columna de la matriz `A`, respectivamente. Si se utiliza un vector como subíndice obtenemos la submatriz correspondiente:

```
> A[2,3];A[1,]  
[1] 8  
[1] 1 4 7
```

- Es posible poner nombres tanto a las filas como a las columnas de una matriz y utilizarlos para extraer elementos de la matriz:

```
> rownames(A) <- c("Fila 1","Fila 2","Fila 3")  
> colnames(A) <- c("Col 1","Col 2","Col 3")  
> A["Fila 1","Col 3"]  
[1] 7
```


- Múltiples usuarios desarrollan técnicas y las comparten generando paquetes de funciones adicionales.
- Para acceder a tales paquetes, es necesario descargarlos de la web del proyecto, aunque esto se realiza directamente desde RStudio. Para instalarlos es necesario seguir los pasos:
 - Desde el menú **Tools** → **Install Packages...**
 - Seleccionar el CRAN mirror (recomendable 0-Cloud).
 - Seleccionar el paquete deseado. Si el paquete necesita otros paquetes, se instalarán automáticamente.
 - Alternativamente, utilizar la función:
`install.packages("nombrepaquete")`.
 - La instalación no implica que los paquetes ya puedan ser utilizados. Es necesario cargarlos desde la ventana **Packages** o con la función:
`library("nombrepaquete")`.

- Para operaciones que requieran varias instrucciones consecutivas, resulta especialmente útil trabajar con un fichero de comandos editable (script).
- R proporciona por defecto la posibilidad de trabajar con scripts como ventanas del propio programa: **File** → **New File** → **R Script**.
- En las ventanas script se pueden entrar varios comandos, separados por líneas o por ";". Pueden ser ejecutados conjuntamente o por separado desde la pestaña **Run line or selection** o bien con Ctrl + R (en Windows).
- El signo # como prefijo indica la introducción de un comentario.
- Los scripts se pueden guardar y utilizar posteriormente. La extensión de este tipo de ficheros es .R.
- Mediante la función `source()` se puede cargar un script de R entero (también desde el menú **Code** → **Source File...**):

```
> source("F:/Curso R/script.R")
```

- Existen diferentes editores que pueden facilitar el trabajo con R como RStudio o TinnR. Estos editores permiten abrir y editar diferentes scripts y ejecutarlos en R.
 - La ventaja de estos editores sobre la ventana de los scripts en R es que ofrecen una serie de opciones no existentes en R.
- También existen varias Interfaces Gráficas de Usuario (GUI en inglés) que permiten trabajar mediante menús sin necesidad de conocer el lenguaje de programación de R como R Commander o Deducer.
 - Estos programas permiten trabajar en un entorno más amigable para el usuario, no obstante, son mucho menos flexibles ya que limitan el uso de opciones de muchas funciones.
- Finalmente, Shiny es una interfaz de presentación de resultados que permite construir aplicaciones web interactivas.

WinEdt

- Editor para scripts bajo licencia, disponible en el sistema operativo Windows www.winedt.com.
- Clásicamente empleado para LaTeX.

```
WinEdt 10.0 | NSIS - [C:\Program Files\WinEdt Team\WinEdt 10\WinShelf\NSIS Installer\WinEdt.nsi]
File Edit Search Insert Document Project View Tools Macros Accessories Options Window Help
WinEdt.nsi
Tree - WinEdt.nsi
  WinEdt.nsi
  Sections
  Full
  Uninstall
  Functions
    .onInit
    AdditionalTasks
    AdditionalTasksLeave
    AssociateAdmin
    AssociateUser
    InstallType
    InstallTypeLeave
    un.AdditionalTasks
    un.AdditionalTasksLeave
    un.AssociateAdmin
    un.AssociateUser
    un.onInit
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
```

Otras interfaces de usuario IDE

Tinn-R

- Editor de libre distribución, similar a WinEdt. Se puede descargar desde <https://sourceforge.net/projects/tinn-r>.

The screenshot displays the Tinn-R IDE interface. The main window shows a LaTeX document with the following content:

```
10 \usepackage{geometry}
11 \usepackage{sectsty}
12 \usepackage{enumitem}
13 \allsectionsfont{\sffam
14 \usepackage{font=small,
15 singlelinecheck=false,
16 justification=justify,
17 labelssep=endash,
18 format=hang]{caption}
19 \geometry{verbose,
20 a4paper,
21 tmargin=1.0cm,
22 bmargin=1.0cm,
23 imargin=1.5cm,
24 rmargin=1cm,
25 headsep=5em,
26 footskip=0cm}
27
28 <echo=F>>
29 #-----
30 # Dadas para correção
31 # Matrículas
32 mat.1 <- 201411366
33 mat.2 <- 201410874
34 mat.3 <- 200720065
35
36 # Sem distinção de sexo
37 # Usou o default da funç
38 default.t <- T
39 default.r <- T
40
41 # Tempo: opções do alun
42
```

The console window on the right shows the following R output:

```
> (x=rmnorm(10))
[1] -1.2715500 -0.3515033 -1.0280353 -0.7256812
[5] -0.2424566 -0.2532621 -0.3455120 -0.6729134
[9] -0.6476809 0.803671 0.1026880 -2.4024819
[13] 0.7149172 0.5882213 -0.9838768 0.4888251
[17] 0.3172518 0.9566414

> mean(x)
[1] -0.2367788

> var(x)
[1] 0.7610039

> sd
function (x, na.rm = FALSE)
sqrt(var(if (is.vector(x) || is.factor(x)) x else
na.rm = na.rm))
<bytecode: 0x000000017d86b70>
<environment: namespace:stats>

> 1:10/2
[1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

> rmnorm(n=1e3, l
```

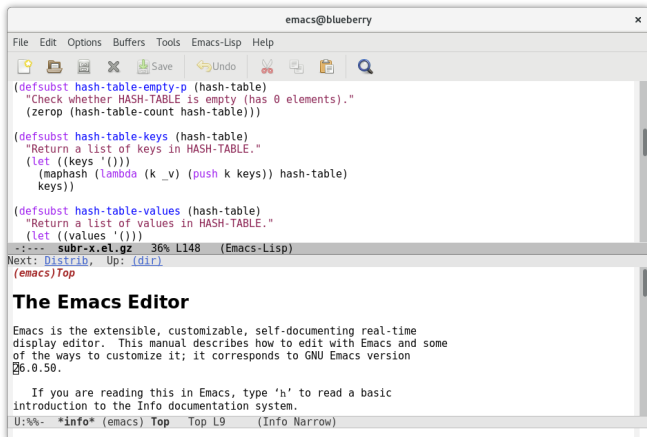
The file explorer on the left shows a list of files and folders, including a table of cities:

Name	City
Brazil (BA)	Ilheus
Brazil (PR)	Curitiba
Brazil (RJ)	Rio de Janeiro
Brazil (SP 1)	Sao Paulo
Brazil (SP 2)	Procioba

The status bar at the bottom indicates the current file is UTF-8, WIN 10/9/05, 1, Editing Normal, Size 24.39 KB, R hotkeys active, R -> TCP/IP [stats], <norm>

Emacs

- Editor de libre distribución muy potente. Puede descargarse desde www.gnu.org/software/emacs.



The screenshot shows the Emacs editor window titled "emacs@blueberry". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Emacs-Lisp", and "Help". The toolbar contains icons for search, file operations, save, undo, redo, and search. The main text area contains the following Emacs Lisp code:

```
(defsubst hash-table-empty-p (hash-table)
  "Check whether HASH-TABLE is empty (has 0 elements)."
  (zerop (hash-table-count hash-table)))

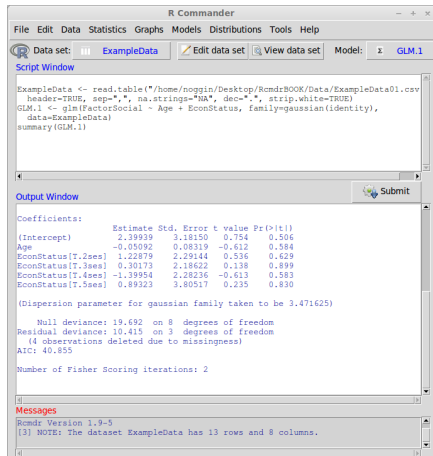
(defsubst hash-table-keys (hash-table)
  "Return a list of keys in HASH-TABLE."
  (let ((keys '()))
    (maphash (lambda (k _v) (push k keys)) hash-table)
    keys))

(defsubst hash-table-values (hash-table)
  "Return a list of values in HASH-TABLE."
  (let ((values '()))
```

Below the code, the status bar shows "---- subr-x.el.gz 36% L148 (Emacs-Lisp)". The bottom panel displays the help page for "Distrib", with "Up: (dir)" and "(emacs)Top". The main heading is "The Emacs Editor", followed by a paragraph: "Emacs is the extensible, customizable, self-documenting real-time display editor. This manual describes how to edit with Emacs and some of the ways to customize it; it corresponds to GNU Emacs version 26.0.50." Below this is another paragraph: "If you are reading this in Emacs, type 'h' to read a basic introduction to the Info documentation system." The bottom status bar shows "U:%%- *info* (emacs) Top Top L9 (Info Narrow)".

R Commander

- GUI muy potente y abierto a personalizaciones.



The screenshot shows the R Commander interface. The top menu bar includes File, Edit, Data, Statistics, Graphs, Models, Distributions, Tools, and Help. Below the menu, there are buttons for 'Data set: ExampleData', 'Edit data set', 'View data set', and 'Model: GLM.1'. The 'Script Window' contains the following R code:

```
ExampleData <- read.table("/home/noggin/Desktop/RcmdrBOOK/Data/ExampleData01.csv",
  header=TRUE, sep=";", na.strings="NA", dec=".", strip.white=TRUE)
GLM.1 <- glm(FactorSocial ~ Age + EconStatus, family=gaussian(identity),
  data=ExampleData)
summary(GLM.1)
```

The 'Output Window' displays the results of the GLM fit:

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.39939    3.18150   0.754  0.506
Age          -0.05092    0.08319  -0.612  0.584
EconStatus[T.2ses] 1.22879    2.29144   0.536  0.629
EconStatus[T.3ses] 0.30173    2.18622   0.138  0.899
EconStatus[T.4ses] -1.39954    2.28236  -0.613  0.583
EconStatus[T.5ses] 0.89323    3.80517   0.235  0.830

(Dispersion parameter for gaussian family taken to be 3.471625)

Null deviance: 19.692  on 8  degrees of freedom
Residual deviance: 10.415  on 3  degrees of freedom
(4 observations deleted due to missingness)
AIC: 40.855

Number of Fisher Scoring iterations: 2
```

The 'Messages' window at the bottom shows the following message:

```
Rcmdr: Version 1.9-5
[3] NOTE: The dataset ExampleData has 13 rows and 8 columns.
```

R Commander

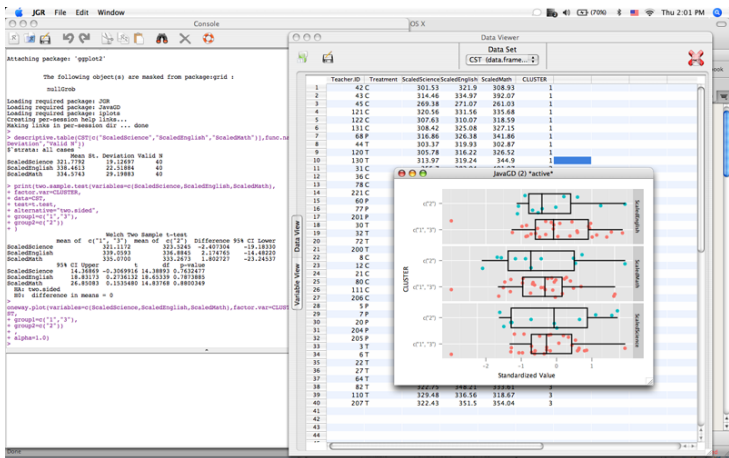
- Permite acceder a muchas capacidades del entorno estadístico R sin que el usuario tenga que conocer el lenguaje de programación propio de este entorno.
- Para abrir R Commander debemos ejecutar la siguiente instrucción desde el programa R:

```
> install.packages("Rcmdr")  
> library("Rcmdr")
```


Interfaces gráficas GUI

DeduceR

- GUI muy visual y ágil.



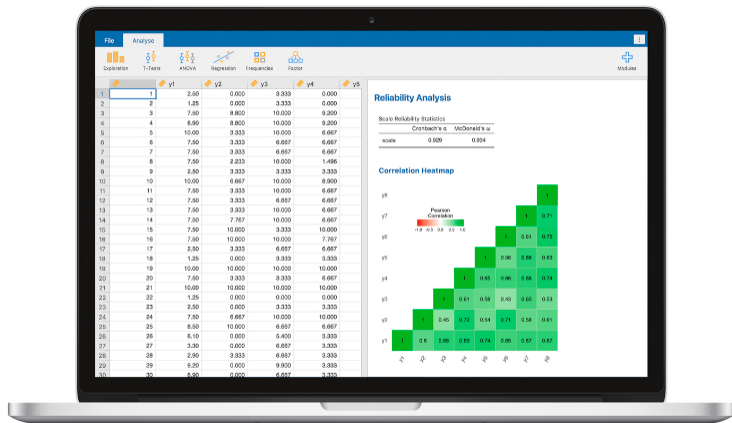
DeduceR

- Está diseñado para ser usado con la consola de R basada en Java: JGR.
- Para abrir Deducer debemos ejecutar la siguiente instrucción desde el programa R:

```
> install.packages(c("JGR", "Deducer"))  
> library(JGR)  
> JGR()
```

jamovi

- No necesita instalar R, una alternativa a SPSS. Puede descargarse desde www.jamovi.org.



Interfaces de presentación de resultados

Shiny

- Permite la construcción de aplicaciones web interactivas: <https://shiny.rstudio.com>.

The screenshot displays the RStudio interface with a Shiny application named 'Movie explorer' running in the browser. The application features a scatter plot of 'Number of Oscars' (y-axis, 100-300) versus 'Years since' (x-axis, 10-100). The plot is color-coded by 'Box Office' status, with a legend for 'Yes' (orange) and 'No' (grey). On the left, there are four interactive sliders: 'Minimum number of reviews on Rotten Tomatoes' (range 50-300), 'Year released' (range 1950-2010), 'Minimum number of Oscar wins (all categories)' (range 0-10), and 'Dollars at Box Office (millions)' (range 0-1000). Below the sliders, there are two checkboxes: 'Genre (in movie can have multiple genres)' (set to 'All') and 'Director names contains (e.g., Miyazaki)' (set to '(Top Level)'). The plot shows 1014 movies selected. The source code in the editor includes R code for plotting terrain colors and Shiny-specific code for the application.

```
16
17- ## Using Terrain Colors
18
19- {r}
20 image(x, y, volcano, col=terrain.colors(100),axes=FALSE)
21 contour(x, y, volcano, levels=seq(50, 200, by=5), add=TRUE, col="brown")
22 axis(1, at=x.at)
23 axis(2, at=y.at)
24 box()
25 title(main="Maunga Whau Volcano", sub = "col=terrain.colors(100)", font.main=4)
26
```

Curso de Introducción a la sintaxis de R mediante
RStudio
Sesión 2: Manejo de datos

Servei d'Estadística Aplicada
Universitat Autònoma de Barcelona

17, 19, 24 y 26 de enero, 2023

Contenido de la Sesión

- 1 Características de los objetos en R
- 2 Vectores. Factores. Generación de secuencias regulares
- 3 Variables indexadas (arrays)
- 4 Matrices. Operaciones con matrices
- 5 Listas

Características de los objetos en R

- R es un lenguaje orientado a objetos: las variables, datos, funciones, resultados, etc. se guardan en la memoria activa del ordenador en forma de objetos con un nombre específico.
- Se pueden modificar o manipular estos objetos con operadores (aritméticos, lógicos y comparativos) y funciones (que a su vez son objetos).
- Las estructuras de datos más usuales son: variables, vectores, matrices, factores, variables indexadas, cadenas de caracteres, listas y dataframes.
- Existen otros como series temporales y es posible crear nuevas estructuras de datos.
- Recordar el funcionamiento de `objects()`, `ls()` y `rm()`.

Características de los objetos en R

- Los objetos están compuestos de elementos. Los elementos más simples pueden ser de los siguientes tipos:
 - Numeric** Número real con doble precisión. Los podemos escribir como enteros (3, -2), con fracción decimal (3.27) o con notación científica (3.12e-47).
 - Complex** Números complejos de la forma $a+bi$.
 - Character** Cadenas alfanuméricas de texto.
 - Logical** Variables lógicas. Pueden ser TRUE o FALSE.

Ejemplo

```
> nombre<-"Luis"  
> varon<-TRUE  
> edad<-23  
> estatura<-1.77
```


Datos especiales: NA's

- En algunos casos las componentes de un objeto pueden no ser completamente conocidas.
- Cuando un elemento $\sqrt{-17+0i}$ es "not available" se le asigna el valor especial NA.
- En general una operación con elementos NA resulta NA, a no ser que mediante una opción de la función, se puedan omitir o tratar los datos faltantes de forma especial.
- La opción por defecto en cualquier función es `na.rm=FALSE` (que indica que NO elimina los NA), que da como resultado NA cuando existe al menos un dato faltante.
- Con la opción `na.rm=TRUE`, la operación se efectúa con los datos válidos.

Ejemplo

```
> x<-NA
> x+1
[1] NA
> y<-c(x,3,5,x)
> mean(y)
[1] NA
> mean(y,na.rm=TRUE)
[1] 4
```

Datos especiales: Inf y NaN's

- En la mayoría de los casos, no hay que preocuparse de si los elementos de un objeto numérico son enteros, reales o incluso complejos. Los cálculos se realizarán internamente como números de doble precisión, reales o complejos según el caso.
- Para trabajar con números complejos, hay que indicar explícitamente la parte compleja.
- En determinadas ocasiones los cálculos realizados pueden llevar a respuestas con valor infinito positivo (representado por R como Inf) o infinito negativo (-Inf).
- Es posible realizar y evaluar cálculos que involucren Inf. Sin embargo, a veces, determinados cálculos llevan a expresiones que no son números (representados por R como NaN's, del inglés "not a number").

Ejemplo

```
> sqrt(-17)
[1] NaN
> sqrt(-17+0i)
[1] 0+4.123106i
> x<-5/0
> exp(-x)
[1] 0
> exp(x)-exp(x)
[1] NaN
```

Clasificación de los objetos en R

- Los objetos se pueden clasificar en dos grandes grupos:
 - Atómicos** Todos los elementos que los componen son del mismo tipo (o modo), como por ejemplo los vectores, las matrices o las series temporales.
 - Recursivos** Pueden combinar una colección de otros objetos de diferente tipo (o modo), como son los dataframes y las listas.
- Existen otras estructuras recursivas, por ejemplo:
 - Modo function** Está formado por las funciones que constituyen R, unidas a las funciones escritas por cada usuario.
 - Modo expression** Corresponde a una parte avanzada de R.
- Además, es posible crear nuevas estructuras.
- La función `str(objeto)` devuelve el tipo (o modo) y el valor de un objeto.

Atributos de los objetos

- Los atributos de un objeto suministran información específica sobre el propio objeto.
- El modo o tipo de un objeto es un caso especial de un atributo de un objeto. Con el modo de un objeto se designa el tipo básico de sus elementos básicos.
- Los atributos de un objeto suministran información específica sobre el propio objeto. Todos los objetos tienen dos atributos intrínsecos: el modo y su longitud.
- Las funciones `mode(objeto)` y `length(objeto)` se pueden utilizar para obtener el modo y longitud de cualquier objeto.

Ejemplo

```
> x<-c(1,3)
> mode(x);length(x)
[1] "numeric"
[1] 2
```

Atributos de los objetos

- Mediante `attributes(objeto)` podemos obtener una lista de los atributos no intrínsecos y con `attr(objeto, atributo)` podemos usar el atributo seleccionado (p.e. para asignarle un valor).
- Los atributos son distintos según el tipo de objeto. Una pequeña lista de atributos es la siguiente:

Atributos	Tipo
mode, length	Todos
storage.mode	Todos
names	Vectores y listas
dim, dimnames	Matrices y arrays
tsp	Series temporales
levels	Factores
class	Cualquier clase

Modificación de la longitud de un objeto

- Un objeto, aunque esté vacío, tiene modo.

Ejemplo

```
> v<-numeric()
> #almacena en v una estructura vacía de vector numérico.
> character()
character(0)
```

- Una vez creado un objeto con un tamaño cualquiera, pueden añadirse nuevos elementos sin más que asignarlos a un índice que esté fuera del rango previo.

Ejemplo

```
> v[3]<-17
> #transforma v en un vector de longitud 3,
> #(cuyas dos primeras componentes serán NA).
```

- Esta regla se aplica a cualquier estructura, siempre que los nuevos elementos sean compatibles con el modo inicial de la estructura.

- Si una expresión se utiliza como una orden por sí misma, su valor se imprime y se pierde. Así pues, la orden $1/x$ simplemente imprime el inverso de lo que sea el objeto x sin modificar su valor.
- Podemos crear vectores con valores iniciales, FALSE, 0,0+0i, " ", etc. mediante la función que indica el tipo de dato (que corresponde a su modo) y entre paréntesis el número de elementos a crear.

Ejemplo

```
> x1<-logical(4)
> #Inicializa un vector de longitud 4
> x2<-numeric(4); x3<-complex(4)
> #Inicializa (a 0) vectores numéricos de longitud 4
> x4<-character(4)
> #Inicializa un vector de caracteres de longitud 4
```

- Los elementos de un objeto de tipo lógico tienen dos posibilidades, FALSE o TRUE. Se pueden abreviar en F y T respectivamente.
- Los objetos lógicos son generalmente fruto de una comparación.
- Las comparaciones que dan un resultado lógico son: "<", "<=", ">", ">=", "==" y "!=".
- Los objetos lógicos se pueden utilizar con la aritmética ordinaria, en cuyo caso son transformados en vectores numéricos, FALSE se convierte en 0 y TRUE en 1.

- La aritmética entre objetos lógicos se puede llevar a cabo con los siguientes operadores lógicos:
 - !x Negación de x. Los T los convierte en F y viceversa.
 - x&y Intersección, operador lógico y: T y T da T, otra comparación da F.
 - x|y Unión, operador lógico o: F y F da F, otra comparación da T.
 - xor(x,y) Exclusivo OR, $\text{xor}(T,F) == T$, otra comparación da F.
 - all Para una secuencia de argumentos lógicos, all devuelve el valor lógico que indica si todos los elementos son TRUE.
 - any Para una secuencia de argumentos lógicos, any devuelve el valor lógico que indica si algún elemento es TRUE.

Ejemplo

```
> a <- c(5,3,6,8); b <- c(4,9,3,8)
> (x <- a<=5); (y <- b<=5)
[1] TRUE TRUE FALSE FALSE
[1] TRUE FALSE TRUE FALSE
> x&y; x|y
[1] TRUE FALSE FALSE FALSE
[1] TRUE TRUE TRUE FALSE
> xor(x,y); any(x)
[1] FALSE TRUE TRUE FALSE
[1] TRUE
> all(x)
[1] FALSE
```

- La mayoría de las funciones producen un error cuando el tipo de datos que esperan no coincide con los que ponemos en los argumentos.
- Existen dos posibilidades:
 - Comprobar el tipo de datos utilizando funciones `is.tipo()`, que responde con un valor lógico.
 - Forzar al tipo de datos deseados **coercionando**, para lo cual se pueden utilizar funciones del tipo `as.tipo()`, que fuerzan el tipo de datos.
- Para conocer si un elemento es NA, la comparación lógica `==`, no puede funcionar ya que NA es la nada. Para ello existe la función `is.na()`, que responde con un valor lógico: TRUE para los elementos del vector que son NA.
- De manera similar, se puede utilizar `is.finite()`, `is.nan()`, etc.

- Lista de los tipos más importantes que se pueden comprobar o forzar:

Tipo	Comprobación	Coerción
array	is.array()	as.array()
character	is.character()	as.character()
complex	is.complex()	as.complex()
double	is.double()	as.double()
factor	is.factor()	as.factor()
integer	is.integer()	as.integer()
list	is.list()	as.list()
logical	is.logical()	as.logical()
matrix	is.matrix()	as.matrix()
NA	is.na()	-
NaN	is.nan()	-
NULL	is.null()	as.null()
numeric	is.numeric()	as.numeric()
ts	is.ts()	as.ts()
vector	is.vector()	as.vector()
date	is.date()	as.date()
times	is.times()	as.times()

Ejemplo

```
> x<-c(1:10)
> is.numeric(x)
[1] TRUE
> is.vector(x)
[1] TRUE
> is.complex(x)
[1] FALSE
> is.character(x)
[1] FALSE
> x<-as.character(x)
> #fuerza el vector x a tomar los valores
> #("1" "2" "3" "4" "5" "6" "7" "8" "9" "10")
```

Características de los vectores

- Extienden los tipos básicos.
- Concatenación de elementos.
- Todos los elementos del vector han de ser del mismo tipo.
- Se crean con la función `c()`.
- Se puede usar `vector[5]` para acceder al quinto elemento.

Ejemplo

```
> nombre<-c("Luis","María")
> edad<-c(23,24)
> varon<-c(TRUE,FALSE);adult<-edad>18
> estatura<-c(1.77,1.64)
> estatura[2]
[1] 1.64
```


Variables categóricas o factores

- Un factor es un vector utilizado para especificar una clasificación discreta de los elementos de otro vector de igual longitud.
- En R existen dos tipos de factores:
 - **Nominales** No ordenados. No existe jerarquía entre ellos (p.e., colores).
 - **Ordinales** Ordenados. Existe jerarquía entre ellos (p.e., grupos de edad).
- Se pueden crear a partir de un vector numérico con las funciones `as.factor()`, `as.ordered()` o con la función `gl()`.
- También a partir de un vector de caracteres utilizando `factor()`.
- Las etiquetas se asignan con `levels()`.

Ejemplo

```
> f<-as.factor(c(1,2,3,1,2,1,1,3,2)) #Factor Nominal
> levels(f)<-c("Bajo","Medio","Alto")
> ford<-as.ordered(f) #Factor Ordinal
```

Vectores de caracteres

- R tiene algunos vectores de caracteres predefinidos: LETTERS, letters, month.name y month.abb.
- La función `paste()` une todos los vectores de caracteres que se le suministran y construye una sola cadena de caracteres. Muy útil para gráficos.
- También admite argumentos numéricos, que convierte inmediatamente en cadenas de caracteres.
- En su forma predeterminada, en la cadena final, cada argumento original se separa del siguiente por un espacio en blanco, aunque ello puede cambiarse utilizando el argumento `sep="cadena"`, que sustituye el espacio en blanco por cadena, la cual podría ser incluso vacía.

Ejemplo

```
> labs<-paste(c("X","Y"),1:9,sep="")
> labs
[1] "X1" "Y2" "X3" "Y4" "X5" "Y6" "X7" "Y8" "X9"
```

Generación de secuencias regulares

- R tiene grandes facilidades para generar vectores de secuencias de números.
- El operador más usual es ":" que genera una secuencia desde:hasta en incrementos (o decremento si hasta es menor que desde) de uno.
- Este operador tiene la prioridad más alta en una expresión.
- La función `seq()` permite generar sucesiones más complejas. Dispone de varios argumentos (`from`, `to`, `by`, `length`). Existen funciones similares que realizan los cálculos más rápidamente (muy útiles a la hora de programar).
- Una función relacionada es `rep()`, que permite duplicar un objeto de formas diversas. Su forma más sencilla es `rep(x,times=5)` que produce cinco copias de `x`, una tras otra.
- La función `sequence(nvec)` genera secuencias de vectores concatenados. `nvec` es un vector de enteros que especifica los límites superiores de las secuencias que se generan. Todas comienzan en uno y se van concatenando en el vector resultante.

Comprobar las siguientes expresiones:

```
> pi:1
[1] 3.141593 2.141593 1.141593
> seq(0,1,length=10)
[1] 0.0000000 0.1111111 0.2222222 0.3333333 0.4444444
[6] 0.5555556 0.6666667 0.7777778 0.8888889 1.0000000
> seq(3)
[1] 1 2 3
> seq(1,5,by=0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> -1:1/0
[1] -Inf NaN Inf
```

Comprobar las siguientes expresiones:

```
> rep(1:4,2)
[1] 1 2 3 4 1 2 3 4
> rep(1:4,c(2,3,1,2))
[1] 1 1 2 2 2 3 4 4

> rep(1:4,c(2,2))
> #error, tamaños no coinciden

> sequence(c(2,3))
[1] 1 2 1 2 3
```

Extrayendo muestras de un vector

- R permite seleccionar muestras de un vector utilizando la función `sample(x,size,replace=FALSE,prob)`.
- Obtiene una muestra de tamaño `size` de `x`, con o sin reemplazamiento, pudiendo tener los elementos de `x` probabilidades distintas a la uniforme (por defecto). Si `x` tiene longitud 1, se considera el vector `1:x`.
- Para que las muestras extraídas sean siempre iguales, es necesario definir una semilla, mediante la instrucción `set.seed`

Ejemplo

```
> x<-c(1:10)
> sample(x,3); sample(x)
[1] 2 9 10
[1] 9 3 6 5 8 7 1 10 4 2
> y<-sample(5:15,5); y
[1] 13 10 5 7 12
```

- R permite seleccionar subconjuntos de elementos de un vector añadiendo al nombre del vector un vector índice entre corchetes [].
- El vector índice puede ser de tres tipos distintos:
 - 1 Un vector lógico. El vector índice debe ser de la misma longitud que el vector del cual se quieren seleccionar elementos. Los valores correspondientes a TRUE en el vector índice son seleccionados, los correspondientes a FALSE omitidos.
 - 2 Un vector de números enteros. En caso de ser positivos, los correspondientes elementos indicados en el índice son concatenados, mientras que si son negativos los valores especificados son omitidos. No hace falta que sea de la misma longitud que el vector de origen.
 - 3 Un vector de nombres. Esta posibilidad se aplica cuando el objeto tiene el atributo names que identifica sus componentes. La función `names` permite añadir etiquetas (o nombres) a un vector numérico.
- La función `which()` genera un vector numérico con las posiciones seleccionadas.

- Vectores índice lógicos:

```
> x<-c(0,NA,1,2)
> y<-x[!is.na(x)]
> (x+1)[!is.na(x) & x>0] -> z
```

- Vectores de números enteros:

```
> x<-letters[c(13,1,18,9,15)]
> paste(x,collapse="")
[1] "mario"
> x[-(4:5)]
[1] "m" "a" "r"
```

- Vector de nombres:

```
> frutos<-c(5,10,1,20)
> names(frutos)<-c("pera","banana","melon","naranja")
> postre<-frutos[c("pera","melon")]
```


Ejemplo

```
> x<-10:20;sum(x)
[1] 165
> length(x)
[1] 11
> sum(x<15)
[1] 5
> x[x<15]
[1] 10 11 12 13 14
> sum(x[x<15])
[1] 60
> length(x[x<15])
[1] 5
> sum(x*(x<15))
[1] 60
> sum(which(x<15))
[1] 15
```

Variables indexadas (arrays)

- Una variable indexada (array) es un objeto con elementos todos del mismo tipo con un atributo adicional (dim) el cual a su vez es un vector numérico de dimensiones (números enteros positivos) formado por varios índices. Vectores y matrices son casos particulares.
- Los elementos del vector de dimensiones indican los límites superiores de los índices. Los límites inferiores siempre valen 1.
- Existen dos maneras de crear una variable indexada:

- 1 Un vector puede transformarse en una variable indexada cuando se asigna un vector de dimensiones al atributo dim.

```
> z<-numeric(1500);dim(z)<-c(3,5,100)
```

- 2 Utilizando la función `array(vector de datos,vector de dimensiones)`:

```
> h<-numeric(24);Z<-array(h,dim=c(3,4,2))
```

Variables indexadas (arrays)

- Al igual que en Fortran, la creación de la variable indexada sigue la regla de que el primer índice es el que se mueve más rápido y el último es el más lento.

Ejemplo

Si se define una variable indexada, a , con vector de dimensiones $c(3,4,2)$, la variable indexada tendrá $3 \times 4 \times 2 = 24$ elementos que se formarán a partir de los elementos originales en el orden $a[1,1,1]$, $a[2,1,1]$, ..., $a[2,4,2]$, $a[3,4,2]$.

- Para referenciar a un elemento de un array se debe indicar el nombre de la variable y , entre corchetes, los índices que lo refieren separados por comas.

Ejemplo

$a[1,2,1]$, ..., $a[2,4,2]$, etc.

Variables indexadas (arrays)

- Se puede referenciar a una parte de un array mediante una sucesión de vectores índices, teniendo en cuenta que si un vector índice es vacío equivale a utilizar todo el rango de valores para dicho índice.

Ejemplo

$a[2,]$ es una variable indexada 4×2 , con vector de dimensión $c(4,2)$ cuyos elementos son ($a[2,1,1]$, $a[2,2,1]$, $a[2,3,1]$, $a[2,4,1]$, $a[2,1,2]$, $a[2,2,2]$, $a[2,3,2]$, $a[2,4,2]$) en ese orden. $a[,,]$ equivale a la variable completa; coincide con a .

- Si se especifica una variable indexada con un solo índice o vector índice, sólo se utilizan los elementos correspondientes del vector de datos (el vector de dimensión se ignora).

Ejemplo

$a[2]$ sería el elemento 2 del vector a , es decir el $a[2,1,1]$ del array.

- Las matrices son un caso particular de array con dos dimensiones: un objeto con un atributo adicional el cual a su vez es un vector numérico de longitud 2, que define el número de filas y columnas de la matriz.
- Todos los elementos deben ser del mismo tipo.
- Una matriz se define con el comando `matrix()` especificando el número de filas y columnas o asignando la `dim` a un vector.
- Recordar que la matriz se crea por columnas, aunque con la opción `byrow=TRUE` lo hace por filas.
- Se pueden asignar nombres a las filas y columnas con el atributo `dimnames`.
- Las funciones `is.matrix()` y `as.matrix()` comprueban o fuerzan el carácter de matriz de un objeto.

- La situación más sencilla es la de seleccionar unas determinadas filas con un vector y unas columnas con otro vector, siguiendo las reglas comentadas en los arrays.
- R tiende a reducir complejidad: al seleccionar una columna, el objeto resultante es un vector y no una matriz. Para mantener las dimensiones, añadir `drop=F`.
- Se pueden seleccionar elementos de una matriz poniendo entre corchetes otra matriz, de las mismas dimensiones que la original, con valores lógicos que indicarán con `TRUE` los elementos a considerar.
- Cuando se realizan operaciones que mezclan variables indexadas (matrices, p.e.) y vectores conviene tener en cuenta que si se asigna un vector más corto a una matriz, se extiende repitiendo sus elementos hasta alcanzar el tamaño deseado.

Ejemplo

```
> x<-c(1:6);dim(x)<-c(2,3)
> dimnames(x)<-list(c("Fila1","Fila2"),c("Col1","Col2","Col3"))
> ejema<-matrix(1:12,ncol=3,byrow=T,dimnames=list(letters[1:4],LETTERS[1:3]))
> ejema[1,1]
[1] 1
> ejema[,c(2,3)]
  B C
a  2 3
b  5 6
c  8 9
d 11 12
```

Ejemplo

```
> ejema[,c(-1,-3),drop=F]
  B
a 2
b 5
c 8
d 11
> sel<-matrix(rep(c(T,F),6),4,3)
> ejema[sel]
[1] 1 7 2 8 3 9
```


- Funciones útiles para trabajar con matrices:

Función	Utilidad
<code>ncol(x)</code> <code>nrow(x)</code> <code>t(x)</code> <code>cbind(...)</code> <code>rbind(...)</code> <code>diag(x)</code> <code>col(x)</code> <code>row(x)</code>	Número de columnas de x. Número de filas de x. Transpuesta de x. Combina secuencias de vectores/matrices por col's. Combina secuencias de vectores/matrices por filas. Extrae diagonal de matriz o crea matriz diagonal. Crea una matriz con elemento ij igual al valor j. Crea una matriz con elemento ij igual al valor i.
<code>apply(x,margin,FUN)</code>	Aplica la función FUN a la dimensión especificada. en margin 1 indica filas, 2 indica columnas.
<code>outer(x,y,fun="*")</code> <code>x %o %y</code>	Para dos vectores x e y, crea una matriz. $A[i,j]=FUN(x[i],y[j])$. Por defecto crea el producto externo.

- Algunas operaciones con matrices

Función	Utilidad
<code>x%*%y</code> <code>crossprod(x,y=x)</code> <code>cov(x,y=x,use="all.obs")</code> <code>cor(x,y=x,use="all.obs")</code>	Multiplicación de matrices. Idem que <code>t(x)%*%y</code> , pero más rápida. Matriz de varianzas-covarianzas. Matriz de correlaciones.
<code>scale(x,center=,scale=)</code>	Resta a las columnas la media si <code>center=TRUE</code> . Si <code>center</code> es un vector, resta dichos valores. Idem para <code>scale</code> , después de centrar, pero divide por la desviación típica si <code>scale=TRUE</code> o los valores asignados si es un vector.
<code>chol(x)</code>	Descomposición de Choleski.
<code>solve(a,b,tol=1e-7)</code>	Resolución de la ecuación <code>a%*%x=b</code> . <code>tol</code> =tolerancia para detectar dependencias lineales en las columnas de <code>a</code> .
<code>eigen(x)</code> <code>sdv(x)</code>	Cálculo de valores y vectores propios. Descomposición en valores singulares.

Ejemplo

```
> x<-matrix(1:6,2,3) ; x[,2]
[1] 3 4
> x[1,1:2]
[1] 1 3
> y<-matrix(1:6,3,2) ; y[3,] ; y[3]
[1] 3 6
[1] 3
> ncol(x); nrow(y)
[1] 3
[1] 3
> t(x)
      [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

Ejemplo

```
> cbind(1,x)
      [,1] [,2] [,3] [,4]
[1,]    1    1    3    5
[2,]    1    2    4    6
> cbind(1:3,1:6)
      [,1] [,2]
[1,]    1    1
[2,]    2    2
[3,]    3    3
[4,]    1    4
[5,]    2    5
[6,]    3    6
> diag(x) # no es necesario que la matriz sea cuadrada
[1] 1 4
> apply(x,1,sum)
[1] 9 12
```

- Una lista es un objeto consistente en una colección ordenada de objetos que se suelen llamar componentes.
- No es necesario que los componentes sean del mismo tipo, ni de la misma longitud: una lista puede estar compuesta de, por ejemplo, un vector numérico de tamaño 2, un valor lógico, un vector de tamaño 3, una matriz y una función.
- Se construyen con la función `list()` o concatenando otras listas.
- Son una parte importante de la programación de funciones en R.
- Los componentes siempre están numerados y pueden ser referidos por dicho número, o por su nombre (si lo tiene, por defecto no lo tiene).
- Aspectos a tener en cuenta al seleccionar partes de una lista:
 - La selección de elementos se hace con doble corchete o con el nombre del elemento precedido del símbolo del dólar.
 - Si se utiliza el corchete simple se está considerando una sublista (de menos componentes) de la lista.

Ejemplo

```
> ejemplolista<-list(nombre="Pedro",casado=T,esposa="María",
  no.hijos=3,edad.hijos=c(4,7,9))
> ejemplolista
$nombre
[1] "Pedro"

$casado
[1] TRUE

$esposa
[1] "María"

$no.hijos
[1] 3

$edad.hijos
[1] 4 7 9
```

Ejemplo

```
> ejemplolista[5]
$edad.hijos
[1] 4 7 9
> is.vector(ejemplolista[5]); is.list(ejemplolista[5])
[1] TRUE
[1] TRUE
> is.vector(ejemplolista[[5]]); is.list(ejemplolista[[5]])
[1] TRUE
[1] FALSE
> ejemplolista[[5]][2]
[1] 7
> ejemplolista[[5]]
[1] 4 7 9
```

Ejemplo

```
> ejemplolista$casado
[1] TRUE
> ejemplolista$nombre
[1] "Pedro"
> is.recursive(ejemplolista) # vemos el tipo de objeto
[1] TRUE
> is.atomic(ejemplolista)
[1] FALSE
> listamasgrande<-c(ejemplolista,list(edad=40))
```


Curso de Introducción a la sintaxis de R mediante
RStudio
Sesión 3: Base de datos en R

Servei d'Estadística Aplicada
Universitat Autònoma de Barcelona

17, 19, 24 y 26 de enero, 2023

Contenido de la Sesión

- 1 Bases de datos o data frames
- 2 Explorar ficheros de datos
- 3 Lectura de ficheros de datos

- Las bases de datos en estadística son, habitualmente de la forma:

Covariables				
individuo	anyos	implante	edad	sexo
1	1.3	2	22	H
2	0.4	2	21	M
3	1.1	2	34	H
4	2.3	1	42	H
5	3.1	3	17	M
6	1.3	1	43	H

- R organiza este tipo de información en objetos del tipo data frame, un caso particular de lista.
- Los data frames son apropiados para describir "matrices de datos" donde cada fila representa a un individuo y cada columna una variable (característica) que puede ser numérica o categórica.
- Al tratarse de una "matriz de datos" muchas de las funciones vistas para las matrices numéricas son de aplicación a los objetos de tipo data frame.

Bases de datos o data frames

- Los data frames se crean con la función `data.frame()`.

Ejemplo

```
> datos<-data.frame(individuo=c(1,2,3,4,5,6),
  anyos=c(1.3,0.4,1.1,2.3,3.1,1.3),tipo=c(2,3,3,1,3,1),
  edad=c(22,21,34,42,17,43),
  sexo=c("H", "M", "H", "H", "M", "H"))
```

- Los elementos de esta función pueden ser vectores (numéricos, caracteres o lógicos), factores, matrices numéricas u otros data frames.
- Puesto que un vector representa una variable de la base de datos, las columnas de una matriz representarán varias variables. La longitud de los vectores debe ser la misma y coincidir con el número de filas de las matrices.
- Los datos que no son numéricos, la función `data.frame()` los considera factores, con tantos niveles como valores distintos encuentre.

Las funciones `attach()` y `detach()`

- Para trabajar con las variables de una base de datos, se puede utilizar la notación estándar de las listas, `$nombre` o `[[]]`, pero resulta más natural emplear simplemente el nombre de la columna.
- El problema es que R guarda el nombre del data frame pero no de sus variables. Para poderlas utilizar por su nombre como vectores, hay que utilizar la función `attach(nombre de data frame)`. La operación inversa se realiza con la función `detach(df)`.

Ejemplo

```
> datos$anyos
> anyos
> attach(datos)
> anyos
> detach(datos)
> anyos
```

Explorar ficheros de datos

- Al igual que las matrices, los data frames tienen un atributo de dimensión. Además, también pueden tener atributos adicionales como nombres de filas, nombres de columnas y comentarios.
- Las funciones `dim(df)` y `attributes(df)` permiten acceder a los atributos del data frame.

Ejemplo

```
> dim(datos)
[1] 6 5
> attributes(datos)
$names
[1] "individuo" "anyos"      "tipo"      "edad"
[5] "sexo"

$class
[1] "data.frame"

$row.names
[1] 1 2 3 4 5 6
```

Explorar ficheros de datos

- La función `rownames(df)` permiten asignar un nombre a las filas del data frame.
- También podemos cambiar los nombres de las columnas existentes utilizando las funciones `colnames(df)` o `names(df)`.
- Por último, al igual que los vectores, las listas y las matrices, podemos añadir un comentario a un data frame sin que ello afecte a su funcionamiento con la función `comment(df)`.

Ejemplo

```
> rownames(datos) <- c("fila 1","fila 2","fila 3","fila 4","fila 5","fila 6")
> names(datos) <- c("Sujeto", "Años", "Tipo", "Edad", "Género")
> comment(datos) <- c("Esta base de datos contiene...")
> attributes(datos)
$names
[1] "Sujeto" "Años"   "Tipo"   "Edad"   "Género"

$class
[1] "data.frame"

$row.names
[1] "fila 1" "fila 2" "fila 3" "fila 4" "fila 5" "fila 6"

$comment
```

Explorar ficheros de datos

- Para acceder a las componentes de un data frame podemos hacer servir los operadores de listas [, [[o \$.
- Acceder con [[o \$ es similar. Sin embargo, difiere para [en que, indexar con [nos devolverá un data frame pero los otros dos lo reducirán a un vector.

Ejemplo

```
> datos["Sujeto"]
      Sujeto
fila 1      1
fila 2      2
fila 3      3
fila 4      4
fila 5      5
fila 6      6
> datos[["Sujeto"]]
[1] 1 2 3 4 5 6
> datos$'Años'
[1] 1.3 0.4 1.1 2.3 3.1 1.3
```


Explorar ficheros de datos

- También se puede acceder al data frame como a una matriz proporcionando un índice para la fila y la columna.

Ejemplo

```
> datos[c(2,3),]
      Sujeto Años Tipo Edad Género
fila 2      2  0.4   3   21      M
fila 3      3  1.1   3   34      H
```

- Finalmente, un data frame puede examinarse utilizando funciones como `str(df)` y `head(df)`.

Ejemplo

```
> str(datos)
'data.frame':      6 obs. of  5 variables:
 $ Sujeto: num  1 2 3 4 5 6
 $ Años  : num  1.3 0.4 1.1 2.3 3.1 1.3
 $ Tipo  : num  2 3 3 1 3 1
 $ Edad  : num  22 21 34 42 17 43
 $ Género: chr  "H" "M" "H" "H" ...
 - attr(*, "comment")= chr "Esta base de datos contiene...."
> #head(datos)
```

Cargar bases de datos de R

- R proporciona dos formatos de archivo propios para almacenar datos, `.RDS` (o `.Rds`) y `.RData` (o `.Rda`). Los archivos RDS pueden almacenar un único objeto R, y los archivos RData pueden almacenar múltiples objetos R.
- Para abrir un archivo RDS se utiliza la función `readRDS()` y para abrir un archivo RData la función `load()`.
- No es necesario asignar la salida a un objeto. Los objetos R de un archivo RData se cargarán en la sesión R con sus nombres originales.
- Un truco útil es poner paréntesis alrededor del comando de carga (`load()`) para que R imprima los nombres de cada objeto que se cargará en la sesión.
- Tanto `readRDS()` como `load()` toman como primer argumento la ruta de archivo (al igual que otras funciones de lectura y escritura de R). Si el archivo está en el directorio de trabajo, la ruta del archivo será el nombre mismo del archivo.

Ejemplo

```
> (load("ADL2.RData"))  
[1] ".Random.seed" "ADL"
```

- Para leer datos de un fichero en formato ASCII existen dos funciones, `scan()` y `read.table()`.
- Como los requisitos de lectura de R son bastante estrictos, muchas veces conviene modificar el archivo de datos externo previamente.
- Para asignar una base de datos a un data frame se utiliza la función `read.table()`. La forma más sencilla es con un fichero en el que:
 - La primera línea del archivo contiene los nombres de las variables.
 - En cada una de las siguientes líneas, el primer elemento es la etiqueta de la fila, y a continuación aparecen los valores de cada variable (sino, R asigna automáticamente etiquetas a las filas).

Ejemplo

```
> read.table(file="BD1.txt",header=T)
```

La función `scan()`

- La función `scan()` es más genérica que la anterior, y vale para asignar cualquier tipo de objetos (vectores, matrices, listas, etc.).
- Si el segundo argumento es un sólo elemento, todos los elementos del archivo deben ser del tipo indicado y se leen en un sólo vector.

Ejemplo

En su uso más básico la función `scan()` permite introducir los valores interactivamente.

```
> x<-scan()
```

En el prompt nos va pidiendo valores que se van incorporando al vector pulsando intro y acaba con intro dos veces. Probar a introducir 1, 5, 8 y 3.

La función scan()

Ejemplo

Supongamos que el fichero "entrada.txt" contiene los datos de tres vectores, de igual longitud, el primero tipo carácter y los otros dos tipo numérico, escritos de tal modo que en cada línea aparecen los valores correspondientes de cada uno de ellos:

```
> entrada<-scan("Entrada.txt",list("",0,0))
```

Podemos referirnos a los vectores con:

```
> etiqueta<-entrada[[1]]  
> x<-entrada[[2]]  
> y<-entrada[[3]]
```

También podríamos haber utilizado:

```
> entrada<-scan("Entrada.txt",list(etiqueta="",x=0,y=0))
```

lo que nos permitiría utilizar la notación \$ para referirnos a los vectores:

```
> etiqueta<-entrada$etiqueta  
> x<-entrada$x  
> y<-entrada$y
```

Importar ficheros de Excel

- La lectura de datos desde un fichero de Excel se puede realizar con la función `read.xlsx()` de la librería `xlsx`:

```
> install.packages("xlsx")
> library(xlsx)
> read.xlsx(file="BD2.xls",sheetIndex=1,header=T)
```

- También se puede leer un fichero de Excel directamente a partir de la función `readWorksheet()`. Para utilizar esta función es necesario instalar previamente el paquete `XLConnect`:

```
> library(XLConnect)
> readWorksheet(loadWorkbook("BD2.xls"),sheet=1)
```

- Otra alternativa es puede guardar los datos (desde Excel) en formato `.csv` y posteriormente utilizar la función `csv.get()` de la librería `Hmisc`:

```
> library(Hmisc)
> csv.get(file="BD3.csv",header=T)
```

- El paquete `Hmisc` también permite importar datos de SPSS, SAS y STATA (funciones `spss.get()`, `sas.get()` y `stata.get()` respectivamente).
- Otro paquete útil para importar bases de datos es `foreign`. Este paquete permite importar y exportar datos en formato `.arff`, `.dbf` y `.dta`, así como importar ficheros de SPSS y otras extensiones.
 - > `library(foreign)`
 - > `read.dbf("BD4.dbf")`

Exportar bases de datos

- Para exportar ficheros en formato .txt se utiliza la función `write.table()`.
> `write.table(datos, file="c:/datos.txt", sep="\t")`
- Para exportar a formato Excel se utiliza la función `write.xlsx()` de la librería `xlsx`.
> `library(xlsx)`
> `write.xlsx(datos, file="c:/datos.xlsx")`
- Para exportar a otros formatos como SPSS, SAS o Stata se utilizan las funciones `write.foreign()` y `write.dta()` de la librería `foreign`.
> `library(foreign)`
> `#Exportat a SPSS`
> `write.foreign(datos, "c:/datos.txt", "c:/datos.sps", package="SPSS")`
> `#Exportat a SAS`
> `write.foreign(datos, "c:/datos.txt", "c:/datos.sas", package="SAS")`

En ambos casos escribe un archivo de datos de texto y un programa SPSS/SAS para leerlo.

```
> #Exportar a Stata  
> write.dta(datos, "c:/datos.dta")
```


Guardar datos (Workspace)

- La función `save(objetos, list = character(0), file = "nomfich.RData", ascii = FALSE)` nos permite guardar los objetos que queramos en un archivo `.RData`.

Ejemplo

```
> #Guardar datos sesión  
> save(datos,file="datos.RData")
```

- Podemos grabar toda la imagen R con la función `save.image()`, es un caso particular de la función `save(list=ls(all.names = TRUE),file=".Rdata")`.
- Al cargar los datos desde el menú **Archivo** → **Cargar área de trabajo** de RStudio o hacer doble click sobre el fichero `.RData` en el fondo estamos utilizando la función `load()`.
- Finalmente, RStudio incorpora diferentes menús para abrir/importar datos y para guardar/exportar datos sin necesidad de recurrir a las funciones de R.

Curso de Introducción a la sintaxis de R mediante
RStudio
Sesión 4: Gestión de bases de datos

Servei d'Estadística Aplicada
Universitat Autònoma de Barcelona

17, 19, 24 y 26 de enero, 2023

Contenido de la Sesión

- 1 Filtrar bases de datos
- 2 Ordenar bases de datos
- 3 Reestructurar bases de datos
 - Funciones de la familia `apply`
- 4 Fusionar bases de datos
- 5 Transformar y Recodificar variables

Crear un subconjunto de datos

- En cualquier momento es posible llevar a cabo un filtrado de datos utilizando expresiones lógicas y vectores índices.
- En el caso de los data frames es posible seleccionar filas y columnas deseadas y guardarlas en una nueva base de datos. También se puede crear un subconjunto de datos con la función `subset()`.

Ejemplo

```
> #Cargar datos sesión  
> load("datos.RData")  
> datos.filtrados<-datos[datos$sexo=="H",]  
> mas.peq<-subset(datos,anyos<1,select=c(edad,sexo))
```

Eliminar filas o columnas

- R no tiene una función que elimine las Filas o columnas de un data frame sin embargo, podemos crear un subconjunto que solo contenga las filas o columnas deseadas como se ha visto antes.
- No obstante, hay formas más sencillas de eliminar filas y columnas. Por ejemplo:

Ejemplo

```
> #Eliminar la 4a fila
> datos2 <- datos[-4,]
> #Eliminar la 4a, 5a y 1a filas
> datos3 <- datos[-c(4,5,1),]
> #Eliminar la 4a fila y la segunda columna
> datos4 <- datos[-4, -2]
> #Eliminar columnas por nombre
> datos5 <- datos[!names(datos) %in% c("anyos","tipo")]
```

- Para ordenar un data frame en R, se utiliza la función `order(df)`. Por defecto, la ordenación es ASCENDENTE. Si se antepongo a la variable de ordenación un signo menos el orden será DESCENDENTE.

Ejemplo

```
> #Orden descentente por individuo
> datos_ord <- datos[order(-datos$individuo),]
> #Alternativa
> attach(datos)
> datos_ord <- datos[order(-individuo),]
> detach(datos)
```

En R existen diversas funciones que nos ayudan a reestructurar y visualizar la base de datos de diferentes maneras.

- La función `t(df)` permite transponer un data frame entero.
- En el paquete `reshape` se encuentran múltiples funciones para llevar a cabo la reestructuración. Las más habituales son:
 - `melt` : se "funden" los datos para que cada fila sea una combinación única de las variables indicadas como identificadoras.
 - `cast` : a partir de un objeto de tipo `melt` reestructura o agrega los datos.
 - `recast` Pasos `melt` y `cast` en uno solo.

La documentación del paquete `reshape` incorpora un video demostrativo.

Ejemplo

```
> library(reshape)
> datos2<-data.frame(id=c(1,1,1,2,2,2),
  visita=c(1,2,3,1,2,3),
  h1=c(143, 128, 122, 145, 139, 137),
  h2=c(82, 81, 79, 100, 102, 99))
> datos2.melt <- melt(datos2, id=c("id","visita"), na.rm=TRUE)
> #Medias por individuos
> id.medias <- cast(datos2.melt, id ~ variable, mean)
> #o bien
> recast(datos2, id ~ variable, id.var=1:2, fun.aggregate="mean")
> #Medias por visitas
> visita.medias <- cast(datos2.melt, visita ~ variable, mean)
> #o bien
> recast(datos2, visita ~ variable, id.var=1:2, fun.aggregate="mean")
```


Funciones de la familia `apply`

Alternativamente, se pueden utilizar las funciones de la familia `apply` para agregar/resumir/filtrar la información de un data frame.

Las funciones de la familia `apply` son las siguientes:

- `lapply`: Devuelve una lista de resultados al aplicar una función a un vector/es.
- `sapply`: Devuelve un vector de resultados al aplicar una función a un vector/es.
- `apply`: Devuelve el vector o la lista de valores obtenidos aplicando una función a un vector o matriz por grupos.
- `tapply`: Devuelve una lista con los resultados obtenidos al aplicar una función a un vector por grupos.
- `ddply`: Devuelve un data.frame de resultados al aplicar una función a un data.frame por grupos.

Funciones de la familia `apply`

- Para aplicar la función `apply()` se utilizan tres parámetros:
 - El objeto array o matriz
 - La dimensión sobre la que actuaremos
 - La función que se aplicará
- En el caso de las matrices, si el segundo argumento es 1, significa que la función se aplicará por filas, mientras que si es un 2, se aplicará a las columnas.
- En muchos casos se puede utilizar una instrucción `apply()` en lugar de un bucle.
- El resultado es un vector o una matriz con nombres (si los había), y es más eficiente que un bucle.

Ejemplo

```
> medias.por.fila <- apply(datos.todo, 1, mean)
> medias.por.columnas <- apply(datos.todo[,3:6], 2, mean)
```

Funciones de la familia `apply`

- Las funciones `lapply()` y `sapply()` son como la función `apply()` pero no hace falta especificar que el segundo argumento es 2, es decir, estas dos funciones realizan los cálculos por columnas.

Ejemplo

```
> lapply(datos.todo[,3:6], mean)
> lapply(datos.todo[,3:6], mean, na.rm=TRUE)
> sapply(datos.todo[,3:6], mean)
```

Funciones de la familia apply

- La función `tapply(x,y,function)` realiza los calculos especificados en `function` a la variable `x` según la variable `y`.

Ejemplo

```
> x.media <- tapply(datos.todo$h1,datos.todo$visita,mean)
> x.var <- tapply(datos.todo$h1,datos.todo$visita,var)
> x.mediana <- tapply(datos.todo$h1,datos.todo$visita,
                      median)
```

Funciones de la familia apply

- La función `ddply()` permite resumir los datos para cada uno de los subconjuntos definidos.
- Esta función se encuentra dentro de la librería `plyr`.

Ejemplo

```
> library(plyr)
> ddply(datos, .(sexo, ec), summarize,
  mean = round(mean(edad),2), sd = round(sd(edad),2))
```

Funciones estadísticas básicas

Función	Utilidad
<code>sum(...,na.rm=FALSE)</code>	Suma
<code>max(...,na.rm=FALSE)</code>	Máximo
<code>min(...,na.rm=FALSE)</code>	Mínimo
<code>which.min(...)</code>	Posición del mínimo
<code>which.max(...)</code>	Posición del máximo
<code>mean(...,na.rm=FALSE)</code>	Media
<code>weighted.mean(...,w,na.rm=FALSE)</code>	Media ponderada
<code>median(...,na.rm=FALSE)</code>	Mediana
<code>quantile(...,prob=C(0,0.25,0.5,0.75,1),na.rm=FALSE)</code>	Cuantiles
<code>IQR(...,na.rm=FALSE)</code>	Rango inter-cuatílico
<code>range(...,na.rm=FALSE,finite=FALSE)</code>	Rango
<code>var(...,na.rm=FALSE)</code>	Varianza
<code>sd(...,na.rm=FALSE)</code>	Desviación típica
<code>summary(...,na.rm=FALSE)</code>	Min, 1c, mediana, media, 3c, max
<code>table(...,na.rm=FALSE)</code>	Recuento casos

A la hora de fusionar bases de datos podemos estar interesados en añadir nuevas variables (columnas) o añadir nuevos individuos (filas).

- Añadir columnas: para combinar dos base de datos se utiliza la función `merge()`. Se fusionan en base a una o más variables clave.

Ejemplo

```
> datos3<-data.frame(id=c(1,1,1,2,2,2),  
  visita=c(1,2,3,1,2,3),  
  col1=c(273, 248, 259, 210, 201, 197),  
  col2=c(222, 217, 208, 197, 200, 189))  
> datos23 = merge (datos2, datos3, by=c("id","visita"))
```


- Añadir filas: para fusionar dos bases de datos verticalmente se utiliza la función `rbind()`. Para ello es necesario que ambas bases de datos tengan exactamente las mismas variables pero no es necesario que estén en el mismo orden.

Ejemplo

```
> masdatos<-data.frame(id=c(3,3,3),
  visita=c(1,2,3), col1=c(301, 289, 291),
  col2=c(250, 241, 256), h1=c(123, 128, 125),
  h2=c(74, 75, 69))
> datos.todo = rbind(datos23, masdatos)
```

- En cualquier momento se puede modificar o manipular cualquier objeto con operadores (aritméticos, lógicos y comparativos) y funciones. Así, por ejemplo, si una variable no es normal, puede transformarse para conseguir dicha normalidad.
- ¡Cuidado! Para que la transformación/recodificación forme parte del data frame, es necesario indicarlo específicamente (no con attach).
- Para modificar el contenido interno de los data frames se puede utilizar la función `transform()`. Esta función permite:
 - Transformar variables existentes.
 - Crear nuevas variables mediante la transformación de variables existentes.

Ejemplo

```
> attach(datos)
> edad.final<-edad+anyos
> # no afecta al data.frame
> datos.1<-transform(datos,edad.final=edad+anyos)
> # si afecta
> datos.2<-transform(datos.1,edad=edad+1)
> # altera la variable
```

Transformaciones y Recodificaciones

Mediante código puede realizarse la recodificación de cualquier variable.

Ejemplo

```
> datos$ec[datos$edad<=25]<-"joven"  
> datos$ec[datos$edad>25]<-"adulto"  
> datos$sexo.cod[datos$sexo=="H"]<-0  
> datos$sexo.cod[datos$sexo=="M"]<-1
```

Por defecto, la variable resultante es carácter o numérica, pero no factor. Para transformarlas a variables tipo Factor, se realiza:

Ejemplo

```
> datos$ec<-as.factor(datos$ec)  
> datos$sexo.cod<-as.factor(datos$sexo.cod)
```



- Mediante la función `recode()` de la librería `car` es posible llevar a cabo recodificaciones de otras variables.
- Se debe especificar cuál es la variable a recodificar, cuáles son los criterios de recodificación y si la nueva variable resultante de la recodificación será un factor o no.
- ¡Cuidado! Las recodificaciones tampoco afectarán al propio data frame si no se indica específicamente.

Licencia de este material



 Creative Commons

<http://creativecommons.org/licenses/by-sa/3.0/es/>

Usted es libre de:

-  Compartir: copiar y redistribuir el material en cualquier medio o formato
-  Adaptar: remezclar, transformar y crear a partir del material

Bajo las condiciones siguientes:

-  Reconocimiento. Debe reconocer los créditos de la obra de la manera especificada por el autor o el licenciador (pero no de una manera que sugiera que tiene su apoyo o apoyan el uso que hace de su obra).
-  Compartir bajo la misma licencia. Si transforma o modifica esta obra para crear una obra derivada, sólo puede distribuir la obra resultante bajo la misma licencia, una similar o una compatible.