Ayudantía 02

Martín Sielfeld

02-04-2020

## Introducción a data cleaning

Para los ejercicios a continuación es necesario que tengan instalados los siguientes paquetes, que cargaremos a continuación a esta sesión:

library(data.table) ## Debiese tenerlo instalado ya  
library(readxl) ## Lectura de datos en formato xlsx

Estaremos ocupando las bases de datos de los casos de COVID-19 a nivel mundial, por lo que creamos el objeto de clase data.table:

Recordar que la opción stringsAsFactors = F le dice a data.table que al crear el objeto, no convierta las variables en formato string (characters) en factores (variables con etiqueta). Esto permite que sea el usuario quien determine si es necesario (y la forma) que la variable sea convertida a factor o no.

casos <- data.table(read\_excel("COVID\_19\_up\_to\_29\_02\_2020\_cases.xlsx"),stringsAsFactors = F)  
muertes <- data.table(read\_excel("COVID\_19\_up\_to\_29\_02\_2020\_deaths.xlsx"),stringsAsFactors = F)  
mas\_dias <- data.table(read\_excel("COVID\_19\_from\_01\_03\_2020\_to\_29\_03\_2020\_all.xlsx"),stringsAsFactors = F)

### Clase de las bases y de sus columnas

Es bueno tener en cuenta, en primera instancia, con la clase de datos que vamos a estar trabajando. Muchas veces ocurre que datos numéricos se encuentran codificados como clase “character” o “factor”, lo que dificulta el trabajo con funciones descriptivas para hacer análisis básico de los datos. Se puede observar rápidamente la clase de cada una de las variables/columnas de las bases de datos respectivas mediante las funciones “str” y “sapply”:

str(casos)

## Classes 'data.table' and 'data.frame': 4087 obs. of 9 variables:  
## $ dateRep : POSIXct, format: "2019-12-31" "2019-12-31" ...  
## $ day : num 31 31 31 31 31 31 31 31 31 31 ...  
## $ month : num 12 12 12 12 12 12 12 12 12 12 ...  
## $ year : num 2019 2019 2019 2019 2019 ...  
## $ cases : num 0 0 0 0 0 0 0 0 0 0 ...  
## $ countriesAndTerritories: chr "Afghanistan" "Algeria" "Armenia" "Australia" ...  
## $ geoId : chr "AF" "DZ" "AM" "AU" ...  
## $ countryterritoryCode : chr "AFG" "DZA" "ARM" "AUS" ...  
## $ popData2018 : num 37172386 42228429 2951776 24992369 8847037 ...  
## - attr(\*, ".internal.selfref")=<externalptr>

sapply(casos, FUN = class)

## $dateRep  
## [1] "POSIXct" "POSIXt"   
##   
## $day  
## [1] "numeric"  
##   
## $month  
## [1] "numeric"  
##   
## $year  
## [1] "numeric"  
##   
## $cases  
## [1] "numeric"  
##   
## $countriesAndTerritories  
## [1] "character"  
##   
## $geoId  
## [1] "character"  
##   
## $countryterritoryCode  
## [1] "character"  
##   
## $popData2018  
## [1] "numeric"

sapply(muertes, FUN = class)

## day month year   
## "numeric" "numeric" "numeric"   
## deaths countriesAndTerritories   
## "numeric" "character"

sapply(mas\_dias, FUN = class)

## $dateRep  
## [1] "POSIXct" "POSIXt"   
##   
## $day  
## [1] "numeric"  
##   
## $month  
## [1] "numeric"  
##   
## $year  
## [1] "numeric"  
##   
## $cases  
## [1] "numeric"  
##   
## $deaths  
## [1] "numeric"  
##   
## $countriesAndTerritories  
## [1] "character"  
##   
## $geoId  
## [1] "character"  
##   
## $countryterritoryCode  
## [1] "character"  
##   
## $popData2018  
## [1] "numeric"

Observamos que str entrega más datos sobre los data.table, pero es menos cómodo de ver que sapply. Observamos que la columna dateRep de los objetos casos y mas\_dias, la cual representa la fecha completa (año,mes,día), es de clase POSIXct.

casos$dateRep <- as.Date(casos$dateRep,"GMT")  
mas\_dias$dateRep <- as.Date(mas\_dias$dateRep,"GMT")

Al igual que la función as.Date, existen múltiples funciones para transformas columnas o bases de datos de una clase a otra. Las más comunes son as.numeric (que transforma a valores numericos, que no necesariamente poseen inicialmente clase numeric), as.character (transforma el/los elemento/s de una columna a character) y as.factor, que los transforma a números subyacentes con etiquetas de tipo character. Por ejemplo:

h <- "01"  
class(h)

## [1] "character"

h <- as.numeric(h)  
class(h)

## [1] "numeric"

h ## Notar que desaparece el "0"

## [1] 1

Veamos ahora las categorías de las variables de mas\_dias (que después descubriremos que son todas las variables diponibles para las tres bases de datos):

sapply(mas\_dias, FUN = unique)

## $dateRep  
## [1] "2020-03-01" "2020-03-02" "2020-03-03" "2020-03-04" "2020-03-05"  
## [6] "2020-03-06" "2020-03-07" "2020-03-08" "2020-03-09" "2020-03-10"  
## [11] "2020-03-11" "2020-03-12" "2020-03-13" "2020-03-14" "2020-03-15"  
## [16] "2020-03-16" "2020-03-17" "2020-03-18" "2020-03-19" "2020-03-20"  
## [21] "2020-03-21" "2020-03-22" "2020-03-23" "2020-03-24" "2020-03-25"  
## [26] "2020-03-26" "2020-03-27" "2020-03-28" "2020-03-29"  
##   
## $day  
## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23  
## [24] 24 25 26 27 28 29  
##   
## $month  
## [1] 3  
##   
## $year  
## [1] 2020  
##   
## $cases  
## [1] 0 1 3 4 574 43 54 205 6 240 9  
## [12] 2 5 595 32 30 18 385 561 15 7 686  
## [23] 17 13 20 127 48 28 523 146 10 600 31  
## [34] 14 8 119 34 39 835 667 516 37 11 22  
## [45] 117 73 66 586 587 49 23 438 12 27 170  
## [56] 138 591 769 44 518 61 26 74 33 59 101  
## [67] 190 284 1234 778 46 483 113 76 122 105 25  
## [78] 60 103 163 21 1076 16 1247 47 367 56 24  
## [89] 55 95 45 410 743 1492 77 248 159 42 68  
## [100] 67 121 29 -9 75 286 237 1797 131 615 200  
## [111] 51 151 372 157 881 977 85 242 435 78 116  
## [122] 52 271 64 252 19 497 958 2313 212 238 114  
## [133] 501 136 152 83 287 115 35 160 96 802 1075  
## [144] 2651 111 132 110 864 158 134 351 41 143 38  
## [155] 128 785 693 57 1289 2547 62 58 107 1227 155  
## [166] 267 511 130 36 838 733 1365 90 40 1522 149  
## [177] 433 777 197 79 84 924 1043 1209 72 6230 176  
## [188] 2000 108 841 251 823 156 199 120 81 1210 1174  
## [199] 1053 4000 125 278 92 86 1438 89 63 887 316  
## [210] 145 53 1097 1144 1178 69 167 3526 292 139 93  
## [221] 1987 450 407 1766 314 243 137 88 91 50 1404  
## [232] 1042 1192 4207 70 346 194 2538 360 680 2988 144  
## [243] 550 309 193 104 99 172 1861 5940 80 1046 191  
## [254] 244 5322 135 409 129 87 3431 878 168 647 4835  
## [265] 165 453 462 283 210 227 1617 4049 82 1237 126  
## [276] 5986 534 235 147 2833 952 154 311 706 5374 224  
## [287] 375 558 260 71 106 1847 3276 141 966 102 171  
## [298] 6557 186 153 637 184 150 118 98 4946 123 233  
## [309] 277 1035 7123 611 607 418 257 1559 3311 94 1028  
## [320] 188 5560 65 573 206 320 3646 894 289 665 8459  
## [331] 855 342 345 216 192 3838 4438 1411 219 371 4789  
## [342] 545 239 460 4517 1044 293 967 11236 796 526 310  
## [353] 313 2446 2342 1762 204 488 5249 811 195 302 100  
## [364] 6584 256 774 343 1427 8789 376 606 668 232 1426  
## [375] 220 133 2931 4954 2206 439 5210 234 852 350 142  
## [386] 633 7937 925 1452 13963 1141 1298 482 164 408 3922  
## [397] 5780 2389 255 297 6153 1019 140 549 182 112 218  
## [408] 8578 296 1000 1196 2129 16797 1049 502 671 304 217  
## [419] 169 3809 6294 2926 369 5959 1172 425 724 263 196  
## [430] 7871 1390 2069 2885 18695 431 594 1850 487 697 299  
## [441] 384 208 4611 3965 109 3076 294 584 5974 226 1159  
## [452] 264 211 272 249 902 228 202 8189 401 1048 1704  
## [463] 2546 19979  
##   
## $deaths  
## [1] 0 1 35 9 8 2 42 11 6 5 32 12 17 4 37 28 3  
## [18] 31 15 27 30 41 7 49 21 36 23 133 43 98 22 54 167 63  
## [35] 196 13 75 189 10 18 85 252 97 173 14 113 370 152 129 347 20  
## [52] 16 135 19 182 69 147 473 107 128 149 429 169 78 625 235 33 110  
## [69] 112 123 795 324 56 80 649 394 48 131 186 127 601 34 462 119 240  
## [86] 122 743 514 87 211 231 143 685 738 249 365 55 157 660 655 24 58  
## [103] 115 246 299 72 144 971 769 26 181 411 64 319 139 887 93 832 38  
## [120] 260 484  
##   
## $countriesAndTerritories  
## [1] "Afghanistan"   
## [2] "Algeria"   
## [3] "Armenia"   
## [4] "Australia"   
## [5] "Austria"   
## [6] "Azerbaijan"   
## [7] "Bahrain"   
## [8] "Belarus"   
## [9] "Belgium"   
## [10] "Brazil"   
## [11] "Cambodia"   
## [12] "Canada"   
## [13] "Cases\_on\_an\_international\_conveyance\_Japan"  
## [14] "China"   
## [15] "Croatia"   
## [16] "Czech\_Republic"   
## [17] "Denmark"   
## [18] "Dominican\_Republic"   
## [19] "Ecuador"   
## [20] "Egypt"   
## [21] "Estonia"   
## [22] "Finland"   
## [23] "France"   
## [24] "Georgia"   
## [25] "Germany"   
## [26] "Greece"   
## [27] "Iceland"   
## [28] "India"   
## [29] "Indonesia"   
## [30] "Iran"   
## [31] "Iraq"   
## [32] "Ireland"   
## [33] "Israel"   
## [34] "Italy"   
## [35] "Japan"   
## [36] "Kuwait"   
## [37] "Lebanon"   
## [38] "Lithuania"   
## [39] "Luxembourg"   
## [40] "Malaysia"   
## [41] "Mexico"   
## [42] "Monaco"   
## [43] "Nepal"   
## [44] "Netherlands"   
## [45] "New\_Zealand"   
## [46] "Nigeria"   
## [47] "North\_Macedonia"   
## [48] "Norway"   
## [49] "Oman"   
## [50] "Pakistan"   
## [51] "Philippines"   
## [52] "Qatar"   
## [53] "Romania"   
## [54] "Russia"   
## [55] "San\_Marino"   
## [56] "Singapore"   
## [57] "South\_Korea"   
## [58] "Spain"   
## [59] "Sri\_Lanka"   
## [60] "Sweden"   
## [61] "Switzerland"   
## [62] "Taiwan"   
## [63] "Thailand"   
## [64] "United\_Arab\_Emirates"   
## [65] "United\_Kingdom"   
## [66] "United\_States\_of\_America"   
## [67] "Vietnam"   
## [68] "Andorra"   
## [69] "Jordan"   
## [70] "Latvia"   
## [71] "Morocco"   
## [72] "Portugal"   
## [73] "Saudi\_Arabia"   
## [74] "Senegal"   
## [75] "Sint\_Maarten"   
## [76] "Tunisia"   
## [77] "Argentina"   
## [78] "Chile"   
## [79] "Poland"   
## [80] "Ukraine"   
## [81] "Hungary"   
## [82] "Liechtenstein"   
## [83] "Slovenia"   
## [84] "Bhutan"   
## [85] "Bosnia\_and\_Herzegovina"   
## [86] "Palestine"   
## [87] "South\_Africa"   
## [88] "Cameroon"   
## [89] "Colombia"   
## [90] "Costa\_Rica"   
## [91] "Holy\_See"   
## [92] "Peru"   
## [93] "Serbia"   
## [94] "Slovakia"   
## [95] "Togo"   
## [96] "Bulgaria"   
## [97] "Maldives"   
## [98] "Malta"   
## [99] "Moldova"   
## [100] "Paraguay"   
## [101] "Albania"   
## [102] "Bangladesh"   
## [103] "Brunei\_Darussalam"   
## [104] "Cyprus"   
## [105] "Mongolia"   
## [106] "Panama"   
## [107] "Burkina\_Faso"   
## [108] "Democratic\_Republic\_of\_the\_Congo"   
## [109] "Bolivia"   
## [110] "Cote\_dIvoire"   
## [111] "Cuba"   
## [112] "Honduras"   
## [113] "Jamaica"   
## [114] "Turkey"   
## [115] "Aruba"   
## [116] "Curaçao"   
## [117] "Gabon"   
## [118] "Ghana"   
## [119] "Guyana"   
## [120] "Saint\_Vincent\_and\_the\_Grenadines"   
## [121] "Trinidad\_and\_Tobago"   
## [122] "Ethiopia"   
## [123] "Guinea"   
## [124] "Kenya"   
## [125] "Sudan"   
## [126] "Antigua\_and\_Barbuda"   
## [127] "Equatorial\_Guinea"   
## [128] "Eswatini"   
## [129] "Guatemala"   
## [130] "Kazakhstan"   
## [131] "Mauritania"   
## [132] "Namibia"   
## [133] "Rwanda"   
## [134] "Saint\_Lucia"   
## [135] "Seychelles"   
## [136] "Suriname"   
## [137] "Uruguay"   
## [138] "Venezuela"   
## [139] "Bahamas"   
## [140] "Central\_African\_Republic"   
## [141] "Congo"   
## [142] "Kosovo"   
## [143] "Uzbekistan"   
## [144] "Benin"   
## [145] "Liberia"   
## [146] "Myanmar"   
## [147] "Somalia"   
## [148] "United\_Republic\_of\_Tanzania"   
## [149] "Barbados"   
## [150] "Gambia"   
## [151] "Montenegro"   
## [152] "Djibouti"   
## [153] "El\_Salvador"   
## [154] "French\_Polynesia"   
## [155] "Guam"   
## [156] "Kyrgyzstan"   
## [157] "Nicaragua"   
## [158] "Zambia"   
## [159] "Bermuda"   
## [160] "Cayman\_Islands"   
## [161] "Chad"   
## [162] "Faroe\_Islands"   
## [163] "Fiji"   
## [164] "Gibraltar"   
## [165] "Greenland"   
## [166] "Guernsey"   
## [167] "Haiti"   
## [168] "Jersey"   
## [169] "Mauritius"   
## [170] "Cape\_Verde"   
## [171] "Isle\_of\_Man"   
## [172] "Madagascar"   
## [173] "Montserrat"   
## [174] "New\_Caledonia"   
## [175] "Niger"   
## [176] "Papua\_New\_Guinea"   
## [177] "Zimbabwe"   
## [178] "Angola"   
## [179] "Eritrea"   
## [180] "Timor\_Leste"   
## [181] "Uganda"   
## [182] "Dominica"   
## [183] "Grenada"   
## [184] "Mozambique"   
## [185] "Syria"   
## [186] "Belize"   
## [187] "United\_States\_Virgin\_Islands"   
## [188] "Laos"   
## [189] "Libya"   
## [190] "Turks\_and\_Caicos\_islands"   
## [191] "Mali"   
## [192] "Saint\_Kitts\_and\_Nevis"   
## [193] "Anguilla"   
## [194] "British\_Virgin\_Islands"   
## [195] "Guinea\_Bissau"   
## [196] "Puerto\_Rico"   
##   
## $geoId  
## [1] "AF" "DZ" "AM" "AU" "AT" "AZ"   
## [7] "BH" "BY" "BE" "BR" "KH" "CA"   
## [13] "JPG11668" "CN" "HR" "CZ" "DK" "DO"   
## [19] "EC" "EG" "EE" "FI" "FR" "GE"   
## [25] "DE" "EL" "IS" "IN" "ID" "IR"   
## [31] "IQ" "IE" "IL" "IT" "JP" "KW"   
## [37] "LB" "LT" "LU" "MY" "MX" "MC"   
## [43] "NP" "NL" "NZ" "NG" "MK" "NO"   
## [49] "OM" "PK" "PH" "QA" "RO" "RU"   
## [55] "SM" "SG" "KR" "ES" "LK" "SE"   
## [61] "CH" "TW" "TH" "AE" "UK" "US"   
## [67] "VN" "AD" "JO" "LV" "MA" "PT"   
## [73] "SA" "SN" "SX" "TN" "AR" "CL"   
## [79] "PL" "UA" "HU" "LI" "SI" "BT"   
## [85] "BA" "PS" "ZA" "CM" "CO" "CR"   
## [91] "VA" "PE" "RS" "SK" "TG" "BG"   
## [97] "MV" "MT" "MD" "PY" "AL" "BD"   
## [103] "BN" "CY" "MN" "PA" "BF" "CD"   
## [109] "BO" "CI" "CU" "HN" "JM" "TR"   
## [115] "AW" "CW" "GA" "GH" "GY" "VC"   
## [121] "TT" "ET" "GN" "KE" "SD" "AG"   
## [127] "GQ" "SZ" "GT" "KZ" "MR" "NA"   
## [133] "RW" "LC" "SC" "SR" "UY" "VE"   
## [139] "BS" "CF" "CG" "XK" "UZ" "BJ"   
## [145] "LR" "MM" "SO" "TZ" "BB" "GM"   
## [151] "ME" "DJ" "SV" "PF" "GU" "KG"   
## [157] "NI" "ZM" "BM" "KY" "TD" "FO"   
## [163] "FJ" "GI" "GL" "GG" "HT" "JE"   
## [169] "MU" "CV" "IM" "MG" "MS" "NC"   
## [175] "NE" "PG" "ZW" "AO" "ER" "TL"   
## [181] "UG" "DM" "GD" "MZ" "SY" "BZ"   
## [187] "VI" "LA" "LY" "TC" "ML" "KN"   
## [193] "AI" "VG" "GW" "PR"   
##   
## $countryterritoryCode  
## [1] "AFG" "DZA" "ARM" "AUS" "AUT" "AZE" "BHR" "BLR" "BEL" "BRA" "KHM"  
## [12] "CAN" "N/A" "CHN" "HRV" "CZE" "DNK" "DOM" "ECU" "EGY" "EST" "FIN"  
## [23] "FRA" "GEO" "DEU" "GRC" "ISL" "IND" "IDN" "IRN" "IRQ" "IRL" "ISR"  
## [34] "ITA" "JPN" "KWT" "LBN" "LTU" "LUX" "MYS" "MEX" "MCO" "NPL" "NLD"  
## [45] "NZL" "NGA" "MKD" "NOR" "OMN" "PAK" "PHL" "QAT" "ROU" "RUS" "SMR"  
## [56] "SGP" "KOR" "ESP" "LKA" "SWE" "CHE" "TWN" "THA" "ARE" "GBR" "USA"  
## [67] "VNM" "AND" "JOR" "LVA" "MAR" "PRT" "SAU" "SEN" "SXM" "TUN" "ARG"  
## [78] "CHL" "POL" "UKR" "HUN" "LIE" "SVN" "BTN" "BIH" "PSE" "ZAF" "CMR"  
## [89] "COL" "CRI" "VAT" "PER" "SRB" "SVK" "TGO" "BGR" "MDV" "MLT" "MDA"  
## [100] "PRY" "ALB" "BGD" "BRN" "CYP" "MNG" "PAN" "BFA" "COD" "BOL" "CIV"  
## [111] "CUB" "HND" "JAM" "TUR" "ABW" "CUW" "GAB" "GHA" "GUY" "VCT" "TTO"  
## [122] "ETH" "GIN" "KEN" "SDN" "ATG" "GNQ" "SWZ" "GTM" "KAZ" "MRT" "NAM"  
## [133] "RWA" "LCA" "SYC" "SUR" "URY" "VEN" "BHS" "CAF" "COG" "XKX" "UZB"  
## [144] "BEN" "LBR" "MMR" "SOM" "TZA" "BRB" "GMB" "MNE" "DJI" "SLV" "PYF"  
## [155] "GUM" "KGZ" "NIC" "ZMB" "BMU" "CYM" "TCD" "FRO" "FJI" "GIB" "GRL"  
## [166] "GGY" "HTI" "JEY" "MUS" "CPV" "IMN" "MDG" "MSR" "NCL" "NER" "PNG"  
## [177] "ZWE" "AGO" "ERI" "TLS" "UGA" "DMA" "GRD" "MOZ" "SYR" "BLZ" "VIR"  
## [188] "LAO" "LBY" "TCA" "MLI" "KNA" NA "VGB" "GNB" "PRI"  
##   
## $popData2018  
## [1] 37172386 42228429 2951776 24992369 8847037 9942334  
## [7] 1569439 9485386 11422068 209469333 16249798 37058856  
## [13] 3000 1392730000 4089400 10625695 5797446 10627165  
## [19] 17084357 98423595 1320884 5518050 66987244 3731000  
## [25] 82927922 10727668 353574 1352617328 267663435 81800269  
## [31] 38433600 4853506 8883800 60431283 126529100 4137309  
## [37] 6848925 2789533 607728 31528585 126190788 38682  
## [43] 28087871 17231017 4885500 195874740 2082958 5314336  
## [49] 4829483 212215030 106651922 2781677 19473936 144478050  
## [55] 33785 5638676 51635256 46723749 21670000 10183175  
## [61] 8516543 23780452 69428524 9630959 66488991 327167434  
## [67] 95540395 77006 9956011 1926542 36029138 10281762  
## [73] 33699947 15854360 41486 11565204 44494502 18729160  
## [79] 37978548 44622516 9768785 37910 2067372 754394  
## [85] 3323929 4569087 57779622 25216237 49648685 4999441  
## [91] 1000 31989256 6982084 5447011 7889094 7024216  
## [97] 515696 483530 3545883 6956071 2866376 161356039  
## [103] 428962 1189265 3170208 4176873 19751535 84068091  
## [109] 11353142 25069229 11338138 9587522 2934855 82319724  
## [115] 105845 159849 2119275 29767108 779004 110210  
## [121] 1389858 109224559 12414318 51393010 41801533 96286  
## [127] 1308974 1367000 17247807 18276499 4403319 2448255  
## [133] 12301939 181889 96762 575991 3449299 28870195  
## [139] 385640 4666377 5244363 1845300 32955400 11485048  
## [145] 4818977 53708395 15008154 56318348 286641 2280102  
## [151] 622345 958920 6420744 277679 165768 6315800  
## [157] 6465513 17351822 63968 64174 15477751 48497  
## [163] 883483 33718 56025 63026 11123176 106800  
## [169] 1265303 543767 84077 26262368 5900 284060  
## [175] 22442948 8606316 14439018 30809762 NA 1267972  
## [181] 42723139 71625 111454 29495962 16906283 383071  
## [187] 106977 7061507 6678567 31458 19077690 52441  
## [193] 29802 1874309 3195153

Ejercicio: ¿Cuál es la unidad observacional (la más granular) de cada base de datos?, ¿Cuáles son las variables comunes entre estas bases?

sapply(list(casos,muertes,mas\_dias),FUN = names)

## [[1]]  
## [1] "dateRep" "day"   
## [3] "month" "year"   
## [5] "cases" "countriesAndTerritories"  
## [7] "geoId" "countryterritoryCode"   
## [9] "popData2018"   
##   
## [[2]]  
## [1] "day" "month"   
## [3] "year" "deaths"   
## [5] "countriesAndTerritories"  
##   
## [[3]]  
## [1] "dateRep" "day"   
## [3] "month" "year"   
## [5] "cases" "deaths"   
## [7] "countriesAndTerritories" "geoId"   
## [9] "countryterritoryCode" "popData2018"

### Funciones cbind, rbind y merge

Antes de filtrar los datos y corregir errores, sería bueno unificar las bases de datos de casos, muertes y mas\_dias, de tal manera de después modificar una única tabla de datos. Viendo nuevamente el chunck “str vs. sapply”, se puede apreciar que las bases de datos casos y muertes poseen datos hasta el 29/02/2020, pero uno tiene los casos de infectados (y otros datos más), mientras que el otro posee los datos de muertos (y otros más). Por lo tanto, podemos unir la base de datos mediante dos métodos posibles:

dim(casos)[1]==dim(muertes)[1] # ver si tienen el mismo número de filas.

## [1] TRUE

cbind(casos,muertes[,.(deaths)]) # Para que no se repitan las columnas, nos quedamos solo con la columna "deaths" de "muertes".

## dateRep day month year cases countriesAndTerritories geoId  
## 1: 2019-12-31 31 12 2019 0 Afghanistan AF  
## 2: 2019-12-31 31 12 2019 0 Algeria DZ  
## 3: 2019-12-31 31 12 2019 0 Armenia AM  
## 4: 2019-12-31 31 12 2019 0 Australia AU  
## 5: 2019-12-31 31 12 2019 0 Austria AT  
## ---   
## 4083: 2020-02-29 29 2 2020 2 Thailand TH  
## 4084: 2020-02-29 29 2 2020 0 United\_Arab\_Emirates AE  
## 4085: 2020-02-29 29 2 2020 2 United\_Kingdom UK  
## 4086: 2020-02-29 29 2 2020 6 United\_States\_of\_America US  
## 4087: 2020-02-29 29 2 2020 0 Vietnam VN  
## countryterritoryCode popData2018 deaths  
## 1: AFG 37172386 0  
## 2: DZA 42228429 0  
## 3: ARM 2951776 0  
## 4: AUS 24992369 0  
## 5: AUT 8847037 0  
## ---   
## 4083: THA 69428524 0  
## 4084: ARE 9630959 0  
## 4085: GBR 66488991 0  
## 4086: USA 327167434 0  
## 4087: VNM 95540395 0

La función “cbind” une dos bases de datos por columnas. Notar que si una columna es más corta que la otra, los datos faltantes de la columna más corta se llenan con datos repetidos. ¿Pero que ocurre si los datos de muertes estan desordenados? Es decir, puede exisitr el problema que los datos de casos de la fila 20 no sean los correspondientes con los de muerte de la misma fila (no es el mismo país, no es la misma fecha, etc…). En ese caso lo mejor que se puede hacer es un merge entre ambas bases:

casosymuertes <- merge(casos,muertes, by = c("day","month","year","countriesAndTerritories"), all = T,sort=F)

Aquí lo que indico a R es que me adjunte los datos de las columnas de la segunda base de datos (muertes) a la primera (casos), de acuerdo a que ambas filas (de ambas bases de datos) cumplan con los mismos elementos estipualdos en el aergumento by. Es decir, se unen los casos que tengan los mismos valores en las columnas “day”, “month”, “year” y “countriesAndTerritories”. Ojo que, en caso de tener más de una fila que cumpla con dichas categorías, entonces habran errores en el merge (R no tiene instrucciones de a que fila asignar dicho dato repetido). En ese caso sería bueno hacer merge por más condiciones o revisar si la base de datos tiene algún error.

Por último, es necesario unir la continuación de la base de datos para el mes de marzo con la que acabamos de crear. Si observamos la base “mas\_dias”, notaremos que tiene las mismas variables que “casosymuertes”:

View(mas\_dias)  
sapply(mas\_dias, FUN = class)

## dateRep day month   
## "Date" "numeric" "numeric"   
## year cases deaths   
## "numeric" "numeric" "numeric"   
## countriesAndTerritories geoId countryterritoryCode   
## "character" "character" "character"   
## popData2018   
## "numeric"

sapply(casosymuertes, FUN = class)

## day month year   
## "numeric" "numeric" "numeric"   
## countriesAndTerritories dateRep cases   
## "character" "Date" "numeric"   
## geoId countryterritoryCode popData2018   
## "character" "character" "numeric"   
## deaths   
## "numeric"

Por lo tanto podemos unir las dos bases de datos:

al\_29\_03\_2020 <- rbind(casosymuertes,mas\_dias)

### Filtrar NAs y corregir

Anteriormente observamos que en la variable “countryterritoryCode” existen valores NA. ¿Pero cómo corroborar para el resto de las variables si también existen NAs? ¿Y para las otras bases?

# Que filas presentan NAs  
al\_29\_03\_2020[rowSums(is.na(al\_29\_03\_2020))>0,]

## day month year countriesAndTerritories dateRep cases geoId  
## 1: 22 3 2020 Eritrea 2020-03-22 1 ER  
## 2: 23 3 2020 Eritrea 2020-03-23 0 ER  
## 3: 24 3 2020 Eritrea 2020-03-24 0 ER  
## 4: 25 3 2020 Eritrea 2020-03-25 0 ER  
## 5: 26 3 2020 Eritrea 2020-03-26 3 ER  
## 6: 27 3 2020 Anguilla 2020-03-27 2 AI  
## 7: 27 3 2020 Eritrea 2020-03-27 2 ER  
## 8: 28 3 2020 Anguilla 2020-03-28 0 AI  
## 9: 28 3 2020 Eritrea 2020-03-28 0 ER  
## 10: 29 3 2020 Anguilla 2020-03-29 0 AI  
## 11: 29 3 2020 Eritrea 2020-03-29 0 ER  
## countryterritoryCode popData2018 deaths  
## 1: ERI NA 0  
## 2: ERI NA 0  
## 3: ERI NA 0  
## 4: ERI NA 0  
## 5: ERI NA 0  
## 6: <NA> NA 0  
## 7: ERI NA 0  
## 8: <NA> NA 0  
## 9: ERI NA 0  
## 10: <NA> NA 0  
## 11: ERI NA 0

al\_29\_03\_2020[!complete.cases(al\_29\_03\_2020),] ## Otro ejemplo

## day month year countriesAndTerritories dateRep cases geoId  
## 1: 22 3 2020 Eritrea 2020-03-22 1 ER  
## 2: 23 3 2020 Eritrea 2020-03-23 0 ER  
## 3: 24 3 2020 Eritrea 2020-03-24 0 ER  
## 4: 25 3 2020 Eritrea 2020-03-25 0 ER  
## 5: 26 3 2020 Eritrea 2020-03-26 3 ER  
## 6: 27 3 2020 Anguilla 2020-03-27 2 AI  
## 7: 27 3 2020 Eritrea 2020-03-27 2 ER  
## 8: 28 3 2020 Anguilla 2020-03-28 0 AI  
## 9: 28 3 2020 Eritrea 2020-03-28 0 ER  
## 10: 29 3 2020 Anguilla 2020-03-29 0 AI  
## 11: 29 3 2020 Eritrea 2020-03-29 0 ER  
## countryterritoryCode popData2018 deaths  
## 1: ERI NA 0  
## 2: ERI NA 0  
## 3: ERI NA 0  
## 4: ERI NA 0  
## 5: ERI NA 0  
## 6: <NA> NA 0  
## 7: ERI NA 0  
## 8: <NA> NA 0  
## 9: ERI NA 0  
## 10: <NA> NA 0  
## 11: ERI NA 0

# Cuantas filas con NAs hay  
al\_29\_03\_2020[rowSums(is.na(al\_29\_03\_2020))>0,.N]

## [1] 11

al\_29\_03\_2020[!complete.cases(al\_29\_03\_2020),.N] ## Otro ejemplo

## [1] 11

¿Que pasa si queremos saber el porcetnaje de NAs por columna, de tal manera de ver si tenemos datos suficientemente representativos?

apply(al\_29\_03\_2020, 2, function(col){  
 paste0(round(sum(is.na(col))/length(col),5)\*100,"%")  
 })

## day month year   
## "0%" "0%" "0%"   
## countriesAndTerritories dateRep cases   
## "0%" "0%" "0%"   
## geoId countryterritoryCode popData2018   
## "0%" "0.04%" "0.146%"   
## deaths   
## "0%"

Como dichos datos no son representatiovs, los eliminaremos (igual no trabajaremos con los datos de los países que estamos eliminando):

al\_29\_03\_2020\_completo <- al\_29\_03\_2020[complete.cases(al\_29\_03\_2020),]

Supongamos que, dada la contingencia, solo nos interesa ver nuetro país, Perú, Alemania e Italia. Como no sabemos si Perú está con tilde o no (para el filtrado), buscamos una secuencia que inicie con “Per”:

unique(grep("Per",al\_29\_03\_2020\_completo[,countriesAndTerritories], value = T))

## [1] "Peru"

Ahora que sabemos, filtramos:

chileyvecinos <- al\_29\_03\_2020\_completo[countriesAndTerritories %in% c("Chile","Germany","Peru","Italy"),]

A su vez, suponga que los únicos datos relevantes para la investigación: día, mes, año, país, número de casos, muertos y población del país a 2018. Podemos filtrar las columnas deseadas:

chileyvecinos <- chileyvecinos[,.(dateRep,day,month,year,countriesAndTerritories,cases,deaths,popData2018)]

Supongamos que se nos entrega información sobre los últimos dos días de marzo para dichos países, tal que:

dias30\_y\_31 <- data.table(dateRep = c(rep(as.Date("2020-3-31"),4),rep(as.Date("2020-03-30"),4)),  
 day = c(rep(31,4),rep(30,4)),  
 month = rep(3,8),  
 year = rep(2020,8),  
 countriesAndTerritories = rep(c("Chile","Germany","Perú","Italy"),2),  
 cases = c(340, 4500, 280, 9800, 300, 4300, 250, 9500),  
 deaths = c(12,100,8,25,80,5,8,20),  
 popData2018 = rep(c(18729160,82927922,31989256,60431283),4))

Por lo tanto la agregaremos a nuestra base de datos vecinal:

chileyvecinos\_completa <- rbind(chileyvecinos,dias30\_y\_31)

Notaremos algunas particularidades de dicha base nueva:

* Los datos no se encuentran ordenados por fecha
* Perú ahora precenta un tilde en la u.

Para corregir estor errores haremos lo siguiente:

chileyvecinos\_completa[countriesAndTerritories == "Perú", "countriesAndTerritories"] <- "Peru"  
  
chileyvecinos\_completa[order(dateRep)]

## dateRep day month year countriesAndTerritories cases deaths  
## 1: 2019-12-31 31 12 2019 Germany 0 0  
## 2: 2019-12-31 31 12 2019 Italy 0 0  
## 3: 2020-01-01 1 1 2020 Germany 0 0  
## 4: 2020-01-01 1 1 2020 Italy 0 0  
## 5: 2020-01-02 2 1 2020 Germany 0 0  
## ---   
## 239: 2020-03-31 31 3 2020 Italy 9800 25  
## 240: 2020-03-31 31 3 2020 Chile 340 12  
## 241: 2020-03-31 31 3 2020 Germany 4500 100  
## 242: 2020-03-31 31 3 2020 Peru 280 8  
## 243: 2020-03-31 31 3 2020 Italy 9800 25  
## popData2018  
## 1: 82927922  
## 2: 60431283  
## 3: 82927922  
## 4: 60431283  
## 5: 82927922  
## ---   
## 239: 60431283  
## 240: 18729160  
## 241: 82927922  
## 242: 31989256  
## 243: 60431283

chileyvecinos\_completa <- chileyvecinos\_completa[order(year, month, day)] # Otra alternativa

## Colapsando y creando variables

Supongamos que deseamos sacar la suma de todos los muertos y casos por mes y país, así como el probedio de la población. Eso se puede hacer en base a un colapso de la base “chileyvecinos\_completa”:

pormes <- chileyvecinos\_completa[, .(cases = sum(cases,na.rm = T),deaths = sum(deaths,na.rm = T), popData2018 = round(mean(popData2018, na.rm = T),0)), by = c("month","countriesAndTerritories")]

Por último, supongamos que deseamos calcular la tasa de infección de la población y la tasa de mortalidad de la población:

pormes[, `Tasa mortalidad` := paste0(round(deaths/cases,4)\*100,"%")]  
pormes[, `Tasa infección` := paste0(round(cases/popData2018,4)\*100,"%")]