Chapter 4: 4.1~4.6

Chapter 4 The tidyverse

Up to now we have been manipulating vectors by reordering and subsetting them through indexing. However, once we start more advanced analyses, the preferred unit for data storage is not the vector but the data frame. In this chapter we learn to work directly with data frames, which greatly facilitate the organization of information. We will be using data frames for the majority of this book. We will focus on a specific data format referred to as tidy and on specific collection of packages that are particularly helpful for working with tidy data referred to as the tidyverse.

We can load all the tidyverse packages at once by installing and loading the tidyverse package:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

We will learn how to implement the tidyverse approach throughout the book, but before delving into the details, in this chapter we introduce some of the most widely used tidyverse functionality, starting with the dplyr package for manipulating data frames and the purrr package for working with functions. Note that the tidyverse also includes a graphing package, ggplot2, which we introduce later in Chapter 7 in the Data Visualization part of the book; the readr package discussed in Chapter 5; and many others. In this chapter, we first introduce the concept of tidy data and then demonstrate how we use the tidyverse to work with data frames in this format.

4.1 Tidy data

We say that a data table is in tidy format if each row represents one observation and columns represent the different variables available for each of these observations. The murders dataset is an example of a tidy data frame.

```
#>        state abb region population total
#> 1    Alabama  AL  South     4779736   135
#> 2     Alaska  AK   West      710231    19
#> 3    Arizona  AZ   West     6392017   232
#> 4   Arkansas  AR  South     2915918    93
#> 5 California  CA   West    37253956  1257
#> 6   Colorado  CO   West     5029196    65
```

Each row represent a state with each of the five columns providing a different variable related to these states: name, abbreviation, region, population, and total murders.

To see how the same information can be provided in different formats, consider the following example:

```
#>        country year fertility
#> 1      Germany 1960     2.41
#> 2 South Korea 1960     6.16
#> 3      Germany 1961     2.44
#> 4 South Korea 1961     5.99
#> 5      Germany 1962     2.47
#> 6 South Korea 1962     5.79
```

This tidy dataset provides fertility rates for two countries across the years. This is a tidy dataset because each row presents one observation with the three variables being country, year, and fertility rate.
However, this dataset originally came in another format and was reshaped for the dslabs package. Originally, the data was in the following format:

```
#>        country 1960 1961 1962
#> 1      Germany 2.41 2.44 2.47
#> 2 South Korea 6.16 5.99 5.79
```

The same information is provided, but there are two important differences in the format: 1) each row includes several observations and 2) one of the variables, year, is stored in the header. For the tidyverse packages to be optimally used, data need to be reshaped into tidy format, which you will learn to do in the Data Wrangling part of the book. Until then, we will use example datasets that are already in tidy format.

Although not immediately obvious, as you go through the book you will start to appreciate the advantages of working in a framework in which functions use tidy formats for both inputs and outputs. You will see how this permits the data analyst to focus on more important aspects of the analysis rather than the format of the data.

4.2 Exercises

1. Examine the built-in dataset co2. Which of the following is true:

```
#The answer is d. co2 is not tidy: to be tidy we would have to wrangle it to have three columns (year,
co2
```

```
##      Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
## 1959 315.42 316.31 316.50 317.56 318.13 318.00 316.39 314.65 313.68 313.18
## 1960 316.27 316.81 317.42 318.87 319.87 319.43 318.01 315.74 314.00 313.68
## 1961 316.73 317.54 318.38 319.31 320.42 319.61 318.42 316.63 314.83 315.16
## 1962 317.78 318.40 319.53 320.42 320.85 320.45 319.45 317.25 316.11 315.27
## 1963 318.58 318.92 319.70 321.22 322.08 321.31 319.58 317.61 316.05 315.83
## 1964 319.41 320.07 320.74 321.40 322.06 321.73 320.27 318.54 316.54 316.71
## 1965 319.27 320.28 320.73 321.97 322.00 321.71 321.05 318.71 317.66 317.14
## 1966 320.46 321.43 322.23 323.54 323.91 323.59 322.24 320.20 318.48 317.94
## 1967 322.17 322.34 322.88 324.25 324.83 323.93 322.38 320.76 319.10 319.24
## 1968 322.40 322.99 323.73 324.86 325.40 325.20 323.98 321.95 320.18 320.09
## 1969 323.83 324.26 325.47 326.50 327.21 326.54 325.72 323.50 322.22 321.62
## 1970 324.89 325.82 326.77 327.97 327.91 327.50 326.18 324.53 322.93 322.90
## 1971 326.01 326.51 327.01 327.62 328.76 328.40 327.20 325.27 323.20 323.40
```

```
## 1972 326.60 327.47 327.58 329.56 329.90 328.92 327.88 326.16 324.68 325.04
## 1973 328.37 329.40 330.14 331.33 332.31 331.90 330.70 329.15 327.35 327.02
## 1974 329.18 330.55 331.32 332.48 332.92 332.08 331.01 329.23 327.27 327.21
## 1975 330.23 331.25 331.87 333.14 333.80 333.43 331.73 329.90 328.40 328.17
## 1976 331.58 332.39 333.33 334.41 334.71 334.17 332.89 330.77 329.14 328.78
## 1977 332.75 333.24 334.53 335.90 336.57 336.10 334.76 332.59 331.42 330.98
## 1978 334.80 335.22 336.47 337.59 337.84 337.72 336.37 334.51 332.60 332.38
## 1979 336.05 336.59 337.79 338.71 339.30 339.12 337.56 335.92 333.75 333.70
## 1980 337.84 338.19 339.91 340.60 341.29 341.00 339.39 337.43 335.72 335.84
## 1981 339.06 340.30 341.21 342.33 342.74 342.08 340.32 338.26 336.52 336.68
## 1982 340.57 341.44 342.53 343.39 343.96 343.18 341.88 339.65 337.81 337.69
## 1983 341.20 342.35 342.93 344.77 345.58 345.14 343.81 342.21 339.69 339.82
## 1984 343.52 344.33 345.11 346.88 347.25 346.62 345.22 343.11 340.90 341.18
## 1985 344.79 345.82 347.25 348.17 348.74 348.07 346.38 344.51 342.92 342.62
## 1986 346.11 346.78 347.68 349.37 350.03 349.37 347.76 345.73 344.68 343.99
## 1987 347.84 348.29 349.23 350.80 351.66 351.07 349.33 347.92 346.27 346.18
## 1988 350.25 351.54 352.05 353.41 354.04 353.62 352.22 350.27 348.55 348.72
## 1989 352.60 352.92 353.53 355.26 355.52 354.97 353.75 351.52 349.64 349.83
## 1990 353.50 354.55 355.23 356.04 357.00 356.07 354.67 352.76 350.82 351.04
## 1991 354.59 355.63 357.03 358.48 359.22 358.12 356.06 353.92 352.05 352.11
## 1992 355.88 356.63 357.72 359.07 359.58 359.17 356.94 354.92 352.94 353.23
## 1993 356.63 357.10 358.32 359.41 360.23 359.55 357.53 355.48 353.67 353.95
## 1994 358.34 358.89 359.95 361.25 361.67 360.94 359.55 357.49 355.84 356.00
## 1995 359.98 361.03 361.66 363.48 363.82 363.30 361.94 359.50 358.11 357.80
## 1996 362.09 363.29 364.06 364.76 365.45 365.01 363.70 361.54 359.51 359.65
## 1997 363.23 364.06 364.61 366.40 366.84 365.68 364.52 362.57 360.24 360.83
##        Nov    Dec
## 1959 314.66 315.43
## 1960 314.84 316.03
## 1961 315.94 316.85
## 1962 316.53 317.53
## 1963 316.91 318.20
## 1964 317.53 318.55
## 1965 318.70 319.25
## 1966 319.63 320.87
## 1967 320.56 321.80
## 1968 321.16 322.74
## 1969 322.69 323.95
## 1970 323.85 324.96
## 1971 324.63 325.85
## 1972 326.34 327.39
## 1973 327.99 328.48
## 1974 328.29 329.41
## 1975 329.32 330.59
## 1976 330.14 331.52
## 1977 332.24 333.68
## 1978 333.75 334.78
## 1979 335.12 336.56
## 1980 336.93 338.04
## 1981 338.19 339.44
## 1982 339.09 340.32
## 1983 340.98 342.82
## 1984 342.80 344.04
## 1985 344.06 345.38
```

```
## 1986 345.48 346.72
## 1987 347.64 348.78
## 1988 349.91 351.18
## 1989 351.14 352.37
## 1990 352.69 354.07
## 1991 353.64 354.89
## 1992 354.09 355.33
## 1993 355.30 356.78
## 1994 357.59 359.05
## 1995 359.61 360.74
## 1996 360.80 362.38
## 1997 362.49 364.34
```

2. Examine the built-in dataset ChickWeight. Which of the following is true:

```
ChickWeight
```

```
##      weight Time Chick Diet
## 1        42    0     1    1
## 2        51    2     1    1
## 3        59    4     1    1
## 4        64    6     1    1
## 5        76    8     1    1
## 6        93   10     1    1
## 7       106   12     1    1
## 8       125   14     1    1
## 9       149   16     1    1
## 10      171   18     1    1
## 11      199   20     1    1
## 12      205   21     1    1
## 13       40    0     2    1
## 14       49    2     2    1
## 15       58    4     2    1
## 16       72    6     2    1
## 17       84    8     2    1
## 18      103   10     2    1
## 19      122   12     2    1
## 20      138   14     2    1
## 21      162   16     2    1
## 22      187   18     2    1
## 23      209   20     2    1
## 24      215   21     2    1
## 25       43    0     3    1
## 26       39    2     3    1
## 27       55    4     3    1
## 28       67    6     3    1
## 29       84    8     3    1
## 30       99   10     3    1
## 31      115   12     3    1
## 32      138   14     3    1
## 33      163   16     3    1
## 34      187   18     3    1
## 35      198   20     3    1
```

```
## 36        202        21         3         1
## 37         42         0         4         1
## 38         49         2         4         1
## 39         56         4         4         1
## 40         67         6         4         1
## 41         74         8         4         1
## 42         87        10         4         1
## 43        102        12         4         1
## 44        108        14         4         1
## 45        136        16         4         1
## 46        154        18         4         1
## 47        160        20         4         1
## 48        157        21         4         1
## 49         41         0         5         1
## 50         42         2         5         1
## 51         48         4         5         1
## 52         60         6         5         1
## 53         79         8         5         1
## 54        106        10         5         1
## 55        141        12         5         1
## 56        164        14         5         1
## 57        197        16         5         1
## 58        199        18         5         1
## 59        220        20         5         1
## 60        223        21         5         1
## 61         41         0         6         1
## 62         49         2         6         1
## 63         59         4         6         1
## 64         74         6         6         1
## 65         97         8         6         1
## 66        124        10         6         1
## 67        141        12         6         1
## 68        148        14         6         1
## 69        155        16         6         1
## 70        160        18         6         1
## 71        160        20         6         1
## 72        157        21         6         1
## 73         41         0         7         1
## 74         49         2         7         1
## 75         57         4         7         1
## 76         71         6         7         1
## 77         89         8         7         1
## 78        112        10         7         1
## 79        146        12         7         1
## 80        174        14         7         1
## 81        218        16         7         1
## 82        250        18         7         1
## 83        288        20         7         1
## 84        305        21         7         1
## 85         42         0         8         1
## 86         50         2         8         1
## 87         61         4         8         1
## 88         71         6         8         1
## 89         84         8         8         1
```

5

```
## 90    93  10   8  1
## 91   110  12   8  1
## 92   116  14   8  1
## 93   126  16   8  1
## 94   134  18   8  1
## 95   125  20   8  1
## 96    42   0   9  1
## 97    51   2   9  1
## 98    59   4   9  1
## 99    68   6   9  1
## 100   85   8   9  1
## 101   96  10   9  1
## 102   90  12   9  1
## 103   92  14   9  1
## 104   93  16   9  1
## 105  100  18   9  1
## 106  100  20   9  1
## 107   98  21   9  1
## 108   41   0  10  1
## 109   44   2  10  1
## 110   52   4  10  1
## 111   63   6  10  1
## 112   74   8  10  1
## 113   81  10  10  1
## 114   89  12  10  1
## 115   96  14  10  1
## 116  101  16  10  1
## 117  112  18  10  1
## 118  120  20  10  1
## 119  124  21  10  1
## 120   43   0  11  1
## 121   51   2  11  1
## 122   63   4  11  1
## 123   84   6  11  1
## 124  112   8  11  1
## 125  139  10  11  1
## 126  168  12  11  1
## 127  177  14  11  1
## 128  182  16  11  1
## 129  184  18  11  1
## 130  181  20  11  1
## 131  175  21  11  1
## 132   41   0  12  1
## 133   49   2  12  1
## 134   56   4  12  1
## 135   62   6  12  1
## 136   72   8  12  1
## 137   88  10  12  1
## 138  119  12  12  1
## 139  135  14  12  1
## 140  162  16  12  1
## 141  185  18  12  1
## 142  195  20  12  1
## 143  205  21  12  1
```

```
## 144    41    0   13   1
## 145    48    2   13   1
## 146    53    4   13   1
## 147    60    6   13   1
## 148    65    8   13   1
## 149    67   10   13   1
## 150    71   12   13   1
## 151    70   14   13   1
## 152    71   16   13   1
## 153    81   18   13   1
## 154    91   20   13   1
## 155    96   21   13   1
## 156    41    0   14   1
## 157    49    2   14   1
## 158    62    4   14   1
## 159    79    6   14   1
## 160   101    8   14   1
## 161   128   10   14   1
## 162   164   12   14   1
## 163   192   14   14   1
## 164   227   16   14   1
## 165   248   18   14   1
## 166   259   20   14   1
## 167   266   21   14   1
## 168    41    0   15   1
## 169    49    2   15   1
## 170    56    4   15   1
## 171    64    6   15   1
## 172    68    8   15   1
## 173    68   10   15   1
## 174    67   12   15   1
## 175    68   14   15   1
## 176    41    0   16   1
## 177    45    2   16   1
## 178    49    4   16   1
## 179    51    6   16   1
## 180    57    8   16   1
## 181    51   10   16   1
## 182    54   12   16   1
## 183    42    0   17   1
## 184    51    2   17   1
## 185    61    4   17   1
## 186    72    6   17   1
## 187    83    8   17   1
## 188    89   10   17   1
## 189    98   12   17   1
## 190   103   14   17   1
## 191   113   16   17   1
## 192   123   18   17   1
## 193   133   20   17   1
## 194   142   21   17   1
## 195    39    0   18   1
## 196    35    2   18   1
## 197    43    0   19   1
```

```
## 198     48      2     19      1
## 199     55      4     19      1
## 200     62      6     19      1
## 201     65      8     19      1
## 202     71     10     19      1
## 203     82     12     19      1
## 204     88     14     19      1
## 205    106     16     19      1
## 206    120     18     19      1
## 207    144     20     19      1
## 208    157     21     19      1
## 209     41      0     20      1
## 210     47      2     20      1
## 211     54      4     20      1
## 212     58      6     20      1
## 213     65      8     20      1
## 214     73     10     20      1
## 215     77     12     20      1
## 216     89     14     20      1
## 217     98     16     20      1
## 218    107     18     20      1
## 219    115     20     20      1
## 220    117     21     20      1
## 221     40      0     21      2
## 222     50      2     21      2
## 223     62      4     21      2
## 224     86      6     21      2
## 225    125      8     21      2
## 226    163     10     21      2
## 227    217     12     21      2
## 228    240     14     21      2
## 229    275     16     21      2
## 230    307     18     21      2
## 231    318     20     21      2
## 232    331     21     21      2
## 233     41      0     22      2
## 234     55      2     22      2
## 235     64      4     22      2
## 236     77      6     22      2
## 237     90      8     22      2
## 238     95     10     22      2
## 239    108     12     22      2
## 240    111     14     22      2
## 241    131     16     22      2
## 242    148     18     22      2
## 243    164     20     22      2
## 244    167     21     22      2
## 245     43      0     23      2
## 246     52      2     23      2
## 247     61      4     23      2
## 248     73      6     23      2
## 249     90      8     23      2
## 250    103     10     23      2
## 251    127     12     23      2
```

```
## 252     135     14     23      2
## 253     145     16     23      2
## 254     163     18     23      2
## 255     170     20     23      2
## 256     175     21     23      2
## 257      42      0     24      2
## 258      52      2     24      2
## 259      58      4     24      2
## 260      74      6     24      2
## 261      66      8     24      2
## 262      68     10     24      2
## 263      70     12     24      2
## 264      71     14     24      2
## 265      72     16     24      2
## 266      72     18     24      2
## 267      76     20     24      2
## 268      74     21     24      2
## 269      40      0     25      2
## 270      49      2     25      2
## 271      62      4     25      2
## 272      78      6     25      2
## 273     102      8     25      2
## 274     124     10     25      2
## 275     146     12     25      2
## 276     164     14     25      2
## 277     197     16     25      2
## 278     231     18     25      2
## 279     259     20     25      2
## 280     265     21     25      2
## 281      42      0     26      2
## 282      48      2     26      2
## 283      57      4     26      2
## 284      74      6     26      2
## 285      93      8     26      2
## 286     114     10     26      2
## 287     136     12     26      2
## 288     147     14     26      2
## 289     169     16     26      2
## 290     205     18     26      2
## 291     236     20     26      2
## 292     251     21     26      2
## 293      39      0     27      2
## 294      46      2     27      2
## 295      58      4     27      2
## 296      73      6     27      2
## 297      87      8     27      2
## 298     100     10     27      2
## 299     115     12     27      2
## 300     123     14     27      2
## 301     144     16     27      2
## 302     163     18     27      2
## 303     185     20     27      2
## 304     192     21     27      2
## 305      39      0     28      2
```

```
## 306      46     2    28    2
## 307      58     4    28    2
## 308      73     6    28    2
## 309      92     8    28    2
## 310     114    10    28    2
## 311     145    12    28    2
## 312     156    14    28    2
## 313     184    16    28    2
## 314     207    18    28    2
## 315     212    20    28    2
## 316     233    21    28    2
## 317      39     0    29    2
## 318      48     2    29    2
## 319      59     4    29    2
## 320      74     6    29    2
## 321      87     8    29    2
## 322     106    10    29    2
## 323     134    12    29    2
## 324     150    14    29    2
## 325     187    16    29    2
## 326     230    18    29    2
## 327     279    20    29    2
## 328     309    21    29    2
## 329      42     0    30    2
## 330      48     2    30    2
## 331      59     4    30    2
## 332      72     6    30    2
## 333      85     8    30    2
## 334      98    10    30    2
## 335     115    12    30    2
## 336     122    14    30    2
## 337     143    16    30    2
## 338     151    18    30    2
## 339     157    20    30    2
## 340     150    21    30    2
## 341      42     0    31    3
## 342      53     2    31    3
## 343      62     4    31    3
## 344      73     6    31    3
## 345      85     8    31    3
## 346     102    10    31    3
## 347     123    12    31    3
## 348     138    14    31    3
## 349     170    16    31    3
## 350     204    18    31    3
## 351     235    20    31    3
## 352     256    21    31    3
## 353      41     0    32    3
## 354      49     2    32    3
## 355      65     4    32    3
## 356      82     6    32    3
## 357     107     8    32    3
## 358     129    10    32    3
## 359     159    12    32    3
```

```
## 360    179   14   32   3
## 361    221   16   32   3
## 362    263   18   32   3
## 363    291   20   32   3
## 364    305   21   32   3
## 365     39    0   33   3
## 366     50    2   33   3
## 367     63    4   33   3
## 368     77    6   33   3
## 369     96    8   33   3
## 370    111   10   33   3
## 371    137   12   33   3
## 372    144   14   33   3
## 373    151   16   33   3
## 374    146   18   33   3
## 375    156   20   33   3
## 376    147   21   33   3
## 377     41    0   34   3
## 378     49    2   34   3
## 379     63    4   34   3
## 380     85    6   34   3
## 381    107    8   34   3
## 382    134   10   34   3
## 383    164   12   34   3
## 384    186   14   34   3
## 385    235   16   34   3
## 386    294   18   34   3
## 387    327   20   34   3
## 388    341   21   34   3
## 389     41    0   35   3
## 390     53    2   35   3
## 391     64    4   35   3
## 392     87    6   35   3
## 393    123    8   35   3
## 394    158   10   35   3
## 395    201   12   35   3
## 396    238   14   35   3
## 397    287   16   35   3
## 398    332   18   35   3
## 399    361   20   35   3
## 400    373   21   35   3
## 401     39    0   36   3
## 402     48    2   36   3
## 403     61    4   36   3
## 404     76    6   36   3
## 405     98    8   36   3
## 406    116   10   36   3
## 407    145   12   36   3
## 408    166   14   36   3
## 409    198   16   36   3
## 410    227   18   36   3
## 411    225   20   36   3
## 412    220   21   36   3
## 413     41    0   37   3
```

```
## 414    48    2    37    3
## 415    56    4    37    3
## 416    68    6    37    3
## 417    80    8    37    3
## 418    83   10    37    3
## 419   103   12    37    3
## 420   112   14    37    3
## 421   135   16    37    3
## 422   157   18    37    3
## 423   169   20    37    3
## 424   178   21    37    3
## 425    41    0    38    3
## 426    49    2    38    3
## 427    61    4    38    3
## 428    74    6    38    3
## 429    98    8    38    3
## 430   109   10    38    3
## 431   128   12    38    3
## 432   154   14    38    3
## 433   192   16    38    3
## 434   232   18    38    3
## 435   280   20    38    3
## 436   290   21    38    3
## 437    42    0    39    3
## 438    50    2    39    3
## 439    61    4    39    3
## 440    78    6    39    3
## 441    89    8    39    3
## 442   109   10    39    3
## 443   130   12    39    3
## 444   146   14    39    3
## 445   170   16    39    3
## 446   214   18    39    3
## 447   250   20    39    3
## 448   272   21    39    3
## 449    41    0    40    3
## 450    55    2    40    3
## 451    66    4    40    3
## 452    79    6    40    3
## 453   101    8    40    3
## 454   120   10    40    3
## 455   154   12    40    3
## 456   182   14    40    3
## 457   215   16    40    3
## 458   262   18    40    3
## 459   295   20    40    3
## 460   321   21    40    3
## 461    42    0    41    4
## 462    51    2    41    4
## 463    66    4    41    4
## 464    85    6    41    4
## 465   103    8    41    4
## 466   124   10    41    4
## 467   155   12    41    4
```

```
## 468   153   14   41   4
## 469   175   16   41   4
## 470   184   18   41   4
## 471   199   20   41   4
## 472   204   21   41   4
## 473    42    0   42   4
## 474    49    2   42   4
## 475    63    4   42   4
## 476    84    6   42   4
## 477   103    8   42   4
## 478   126   10   42   4
## 479   160   12   42   4
## 480   174   14   42   4
## 481   204   16   42   4
## 482   234   18   42   4
## 483   269   20   42   4
## 484   281   21   42   4
## 485    42    0   43   4
## 486    55    2   43   4
## 487    69    4   43   4
## 488    96    6   43   4
## 489   131    8   43   4
## 490   157   10   43   4
## 491   184   12   43   4
## 492   188   14   43   4
## 493   197   16   43   4
## 494   198   18   43   4
## 495   199   20   43   4
## 496   200   21   43   4
## 497    42    0   44   4
## 498    51    2   44   4
## 499    65    4   44   4
## 500    86    6   44   4
## 501   103    8   44   4
## 502   118   10   44   4
## 503   127   12   44   4
## 504   138   14   44   4
## 505   145   16   44   4
## 506   146   18   44   4
## 507    41    0   45   4
## 508    50    2   45   4
## 509    61    4   45   4
## 510    78    6   45   4
## 511    98    8   45   4
## 512   117   10   45   4
## 513   135   12   45   4
## 514   141   14   45   4
## 515   147   16   45   4
## 516   174   18   45   4
## 517   197   20   45   4
## 518   196   21   45   4
## 519    40    0   46   4
## 520    52    2   46   4
## 521    62    4   46   4
```

```
## 522     82     6    46    4
## 523    101     8    46    4
## 524    120    10    46    4
## 525    144    12    46    4
## 526    156    14    46    4
## 527    173    16    46    4
## 528    210    18    46    4
## 529    231    20    46    4
## 530    238    21    46    4
## 531     41     0    47    4
## 532     53     2    47    4
## 533     66     4    47    4
## 534     79     6    47    4
## 535    100     8    47    4
## 536    123    10    47    4
## 537    148    12    47    4
## 538    157    14    47    4
## 539    168    16    47    4
## 540    185    18    47    4
## 541    210    20    47    4
## 542    205    21    47    4
## 543     39     0    48    4
## 544     50     2    48    4
## 545     62     4    48    4
## 546     80     6    48    4
## 547    104     8    48    4
## 548    125    10    48    4
## 549    154    12    48    4
## 550    170    14    48    4
## 551    222    16    48    4
## 552    261    18    48    4
## 553    303    20    48    4
## 554    322    21    48    4
## 555     40     0    49    4
## 556     53     2    49    4
## 557     64     4    49    4
## 558     85     6    49    4
## 559    108     8    49    4
## 560    128    10    49    4
## 561    152    12    49    4
## 562    166    14    49    4
## 563    184    16    49    4
## 564    203    18    49    4
## 565    233    20    49    4
## 566    237    21    49    4
## 567     41     0    50    4
## 568     54     2    50    4
## 569     67     4    50    4
## 570     84     6    50    4
## 571    105     8    50    4
## 572    122    10    50    4
## 573    155    12    50    4
## 574    175    14    50    4
## 575    205    16    50    4
```

```
## 576     234   18    50    4
## 577     264   20    50    4
## 578     264   21    50    4
```

```
# The answer is d. ChickWeight is tidy: it is stored in a data frame.
```

3. Examine the built-in dataset BOD. Which of the following is true:

```
BOD
```

```
##    Time demand
## 1     1    8.3
## 2     2   10.3
## 3     3   19.0
## 4     4   16.0
## 5     5   15.6
## 6     7   19.8
```

```
# The answer is c. BOD is tidy: each row is an observation with two values (time and demand)
```

4. Which of the following built-in datasets is tidy (you can pick more than one):

```
# DNase
# Formaldehyde
# Orange
```

4.3 Manipulating data frames

The dplyr package from the tidyverse introduces functions that perform some of the most common operations when working with data frames and uses names for these functions that are relatively easy to remember. For instance, to change the data table by adding a new column, we use mutate. To filter the data table to a subset of rows, we use filter. Finally, to subset the data by selecting specific columns, we use select.

4.3.1 Adding a column with mutate

We want all the necessary information for our analysis to be included in the data table. So the first task is to add the murder rates to our murders data frame. The function mutate takes the data frame as a first argument and the name and values of the variable as a second argument using the convention name = values. So, to add murder rates, we use:

```
library(dslabs)
data("murders")
murders <- mutate(murders, rate=total/population*100000)
```

This is one of dplyr's main features. Functions in this package, such as mutate, know to look for variables in the data frame provided in the first argument. In the call to mutate above, total will have the values in murders$total. This approach makes the code much more readable.

We can see that the new column is added:

```
head(murders)
```

```
##        state abb region population total    rate
## 1    Alabama  AL  South   4779736   135 2.824424
## 2     Alaska  AK   West    710231    19 2.675186
## 3    Arizona  AZ   West   6392017   232 3.629527
## 4   Arkansas  AR  South   2915918    93 3.189390
## 5 California  CA   West  37253956  1257 3.374138
## 6   Colorado  CO   West   5029196    65 1.292453
```

Although we have overwritten the original murders object, this does not change the object that loaded with data(murders). If we load the murders data again, the original will overwrite our mutated version.

4.3.2 Subsetting with filter

Now suppose that we want to filter the data table to only show the entries for which the murder rate is lower than 0.71. To do this we use the filter function, which takes the data table as the first argument and then the conditional statement as the second. Like mutate, we can use the unquoted variable names from murders inside the function and it will know we mean the columns and not objects in the workspace.

```
filter(murders, rate<=0.71)
```

```
##           state abb        region population total      rate
## 1        Hawaii  HI          West    1360301     7 0.5145920
## 2          Iowa  IA North Central    3046355    21 0.6893484
## 3 New Hampshire  NH     Northeast    1316470     5 0.3798036
## 4  North Dakota  ND North Central     672591     4 0.5947151
## 5       Vermont  VT     Northeast     625741     2 0.3196211
```

4.3.3 Selecting columns with select

Although our data table only has six columns, some data tables include hundreds. If we want to view just a few, we can use the dplyr select function. In the code below we select three columns, assign this to a new object and then filter the new object:

```
new_table <- select(murders, state, region, rate)
filter(new_table, rate<=0.71)
```

```
##           state        region      rate
## 1        Hawaii          West 0.5145920
## 2          Iowa North Central 0.6893484
## 3 New Hampshire     Northeast 0.3798036
## 4  North Dakota North Central 0.5947151
## 5       Vermont     Northeast 0.3196211
```

In the call to select, the first argument murders is an object, but state, region, and rate are variable names.

4.4 Exercises

1. Load the dplyr package and the murders dataset.

```
library(dplyr)
library(dslabs)
data(murders)
```

You can add columns using the dplyr function mutate. This function is aware of the column names and inside the function you can call them unquoted:

```
murders <- mutate(murders, population_in_millions = population / 10^6)
```

We can write population rather than murders$population. The function mutate knows we are grabbing columns from murders.

Use the function mutate to add a murders column named rate with the per 100,000 murder rate as in the example code above. Make sure you redefine murders as done in the example code above (murders <- [your code]) so we can keep using this variable.

```
murders<-mutate(murders,rate=total/ population*100000)
```

2. If rank(x) gives you the ranks of x from lowest to highest, rank(-x) gives you the ranks from highest to lowest. Use the function mutate to add a column rank containing the rank, from highest to lowest murder rate. Make sure you redefine murders so we can keep using this variable.

```
murders<-mutate(murders, rank=rank(-rate))
```

3. With dplyr, we can use select to show only certain columns. For example, with this code we would only show the states and population sizes:

```
select(murders, state, population) %>% head()
```

```
##         state population
## 1     Alabama    4779736
## 2      Alaska     710231
## 3     Arizona    6392017
## 4    Arkansas    2915918
## 5  California   37253956
## 6    Colorado    5029196
```

Use select to show the state names and abbreviations in murders. Do not redefine murders, just show the results.

```
select(murders, state, abb)
```

```
##               state abb
## 1           Alabama  AL
## 2            Alaska  AK
## 3           Arizona  AZ
```

```
## 4               Arkansas  AR
## 5             California  CA
## 6               Colorado  CO
## 7            Connecticut  CT
## 8               Delaware  DE
## 9   District of Columbia  DC
## 10               Florida  FL
## 11               Georgia  GA
## 12                Hawaii  HI
## 13                 Idaho  ID
## 14              Illinois  IL
## 15               Indiana  IN
## 16                  Iowa  IA
## 17                Kansas  KS
## 18              Kentucky  KY
## 19             Louisiana  LA
## 20                 Maine  ME
## 21              Maryland  MD
## 22         Massachusetts  MA
## 23              Michigan  MI
## 24             Minnesota  MN
## 25           Mississippi  MS
## 26              Missouri  MO
## 27               Montana  MT
## 28              Nebraska  NE
## 29                Nevada  NV
## 30         New Hampshire  NH
## 31            New Jersey  NJ
## 32            New Mexico  NM
## 33              New York  NY
## 34        North Carolina  NC
## 35          North Dakota  ND
## 36                  Ohio  OH
## 37              Oklahoma  OK
## 38                Oregon  OR
## 39          Pennsylvania  PA
## 40          Rhode Island  RI
## 41        South Carolina  SC
## 42          South Dakota  SD
## 43             Tennessee  TN
## 44                 Texas  TX
## 45                  Utah  UT
## 46               Vermont  VT
## 47              Virginia  VA
## 48            Washington  WA
## 49         West Virginia  WV
## 50             Wisconsin  WI
## 51               Wyoming  WY
```

4. The dplyr function filter is used to choose specific rows of the data frame to keep. Unlike select which is for columns, filter is for rows. For example, you can show just the New York row like this:

```
filter(murders, state == "New York")
```

```
##      state abb    region population total population_in_millions   rate rank
## 1 New York  NY Northeast   19378102   517                 19.3781 2.66796   29
```

You can use other logical vectors to filter rows.

Use filter to show the top 5 states with the highest murder rates. After we add murder rate and rank, do not change the murders dataset, just show the result. Remember that you can filter based on the rank column.

```
filter(murders, rank<=5)
```

```
##                  state abb        region population total
## 1 District of Columbia  DC         South     601723    99
## 2            Louisiana  LA         South    4533372   351
## 3             Maryland  MD         South    5773552   293
## 4             Missouri  MO North Central    5988927   321
## 5       South Carolina  SC         South    4625364   207
##   population_in_millions      rate rank
## 1              0.601723 16.452753    1
## 2              4.533372  7.742581    2
## 3              5.773552  5.074866    4
## 4              5.988927  5.359892    3
## 5              4.625364  4.475323    5
```

5. We can remove rows using the != operator. For example, to remove Florida, we would do this:

```
no_florida <- filter(murders, state != "Florida")
```

Create a new data frame called no_south that removes states from the South region. How many states are in this category? You can use the function nrow for this.

```
no_south <- filter(murders,region!="South")
nrow(no_south)
```

```
## [1] 34
```

6. We can also use %in% to filter with dplyr. You can therefore see the data from New York and Texas like this:

```
filter(murders, state %in% c("New York", "Texas"))
```

```
##      state abb    region population total population_in_millions    rate rank
## 1 New York  NY Northeast   19378102   517               19.37810 2.66796   29
## 2    Texas  TX     South   25145561   805               25.14556 3.20136   16
```

Create a new data frame called murders_nw with only the states from the Northeast and the West. How many states are in this category?

```
murders_nw <- filter(murders, region %in% c("Northeast","West"))
nrow(murders_nw)
```

[1] 22

7. Suppose you want to live in the Northeast or West and want the murder rate to be less than 1. We want to see the data for the states satisfying these options. Note that you can use logical operators with filter. Here is an example in which we filter to keep only small states in the Northeast region.

```
filter(murders, population < 5000000 & region == "Northeast")
```

```
##           state abb   region population total population_in_millions      rate
## 1   Connecticut  CT Northeast    3574097    97              3.574097 2.7139722
## 2         Maine  ME Northeast    1328361    11              1.328361 0.8280881
## 3 New Hampshire  NH Northeast    1316470     5              1.316470 0.3798036
## 4  Rhode Island  RI Northeast    1052567    16              1.052567 1.5200933
## 5       Vermont  VT Northeast     625741     2              0.625741 0.3196211
##   rank
## 1   25
## 2   44
## 3   50
## 4   35
## 5   51
```

Make sure murders has been defined with rate and rank and still has all states. Create a table called my_states that contains rows for states satisfying both the conditions: it is in the Northeast or West and the murder rate is less than 1. Use select to show only the state name, the rate, and the rank.

```
my_states <- filter(murders, region %in% c("Northeast","West")& rate<1)
select(my_states,state,rate,rank)
```

```
##           state      rate rank
## 1        Hawaii 0.5145920   49
## 2         Idaho 0.7655102   46
## 3         Maine 0.8280881   44
## 4 New Hampshire 0.3798036   50
## 5        Oregon 0.9396843   42
## 6          Utah 0.7959810   45
## 7       Vermont 0.3196211   51
## 8       Wyoming 0.8871131   43
```

4.5 The pipe: %>%

With dplyr we can perform a series of operations, for example select and then filter, by sending the results of one function to another using what is called the pipe operator: %>%. Some details are included below.

We wrote code above to show three variables (state, region, rate) for states that have murder rates below 0.71. To do this, we defined the intermediate object new_table. In dplyr we can write code that looks more like a description of what we want to do without intermediate objects:

original data -> select -> filter

For such an operation, we can use the pipe %>%. The code looks like this:

```
murders %>% select(state, region, rate) %>% filter(rate <= 0.71)
```

```
##             state       region      rate
## 1         Hawaii         West 0.5145920
## 2           Iowa North Central 0.6893484
## 3 New Hampshire     Northeast 0.3798036
## 4  North Dakota North Central 0.5947151
## 5        Vermont     Northeast 0.3196211
```

This line of code is equivalent to the two lines of code above. What is going on here?

In general, the pipe sends the result of the left side of the pipe to be the first argument of the function on the right side of the pipe. Here is a very simple example:

```
16 %>% sqrt()
```

```
## [1] 4
```

We can continue to pipe values along:

```
16 %>% sqrt() %>% log2()
```

```
## [1] 2
```

The above statement is equivalent to log2(sqrt(16)).

Remember that the pipe sends values to the first argument, so we can define other arguments as if the first argument is already defined:

```
16 %>% sqrt() %>% log(base = 2)
```

```
## [1] 2
```

Therefore, when using the pipe with data frames and dplyr, we no longer need to specify the required first argument since the dplyr functions we have described all take the data as the first argument. In the code we wrote:

```
murders %>% select(state,region,rate) %>% filter(rate <= 0.71)
```

```
##             state       region      rate
## 1         Hawaii         West 0.5145920
## 2           Iowa North Central 0.6893484
## 3 New Hampshire     Northeast 0.3798036
## 4  North Dakota North Central 0.5947151
## 5        Vermont     Northeast 0.3196211
```

murders is the first argument of the select function, and the new data frame (formerly new_table) is the first argument of the filter function.

Note that the pipe works well with functions where the first argument is the input data. Functions in tidyverse packages like dplyr have this format and can be used easily with the pipe.

4.6 Exercises

1. The pipe %>% can be used to perform operations sequentially without having to define intermediate objects. Start by redefining murder to include rate and rank.

```
data(murders)
murders <- mutate(murders, rate =  total / population * 100000,
                  rank = rank(-rate))
```

In the solution to the previous exercise, we did the following:

```
my_states <- filter(murders, region %in% c("Northeast", "West") &
                     rate < 1)

select(my_states, state, rate, rank)
```

```
##            state       rate rank
## 1         Hawaii 0.5145920   49
## 2          Idaho 0.7655102   46
## 3          Maine 0.8280881   44
## 4 New Hampshire 0.3798036   50
## 5         Oregon 0.9396843   42
## 6           Utah 0.7959810   45
## 7        Vermont 0.3196211   51
## 8        Wyoming 0.8871131   43
```

The pipe %>% permits us to perform both operations sequentially without having to define an intermediate variable my_states. We therefore could have mutated and selected in the same line like this:

```
mutate(murders, rate =  total / population * 100000,
       rank = rank(-rate)) %>%
  select(state, rate, rank)
```

```
##                    state       rate rank
## 1                Alabama  2.8244238   23
## 2                 Alaska  2.6751860   27
## 3                Arizona  3.6295273   10
## 4               Arkansas  3.1893901   17
## 5             California  3.3741383   14
## 6               Colorado  1.2924531   38
## 7            Connecticut  2.7139722   25
## 8               Delaware  4.2319369    6
## 9   District of Columbia 16.4527532    1
## 10               Florida  3.3980688   13
## 11               Georgia  3.7903226    9
## 12                Hawaii  0.5145920   49
## 13                 Idaho  0.7655102   46
## 14              Illinois  2.8369608   22
## 15               Indiana  2.1900730   31
## 16                  Iowa  0.6893484   47
## 17                Kansas  2.2081106   30
## 18              Kentucky  2.6732010   28
```

```
## 19           Louisiana  7.7425810    2
## 20               Maine  0.8280881   44
## 21            Maryland  5.0748655    4
## 22       Massachusetts  1.8021791   32
## 23            Michigan  4.1786225    7
## 24           Minnesota  0.9992600   40
## 25         Mississippi  4.0440846    8
## 26            Missouri  5.3598917    3
## 27             Montana  1.2128379   39
## 28            Nebraska  1.7521372   33
## 29              Nevada  3.1104763   19
## 30       New Hampshire  0.3798036   50
## 31          New Jersey  2.7980319   24
## 32          New Mexico  3.2537239   15
## 33            New York  2.6679599   29
## 34      North Carolina  2.9993237   20
## 35        North Dakota  0.5947151   48
## 36                Ohio  2.6871225   26
## 37            Oklahoma  2.9589340   21
## 38              Oregon  0.9396843   42
## 39        Pennsylvania  3.5977513   11
## 40        Rhode Island  1.5200933   35
## 41      South Carolina  4.4753235    5
## 42        South Dakota  0.9825837   41
## 43           Tennessee  3.4509357   12
## 44               Texas  3.2013603   16
## 45                Utah  0.7959810   45
## 46             Vermont  0.3196211   51
## 47            Virginia  3.1246001   18
## 48          Washington  1.3829942   37
## 49       West Virginia  1.4571013   36
## 50           Wisconsin  1.7056487   34
## 51             Wyoming  0.8871131   43
```

Notice that select no longer has a data frame as the first argument. The first argument is assumed to be the result of the operation conducted right before the %>%.

Repeat the previous exercise, but now instead of creating a new object, show the result and only include the state, rate, and rank columns. Use a pipe %>% to do this in just one line.

```
murders %>% filter(region %in% c("Northeast","West") & rate<1) %>% select(state, rate, rank)
```

```
##             state      rate rank
## 1          Hawaii 0.5145920   49
## 2           Idaho 0.7655102   46
## 3           Maine 0.8280881   44
## 4   New Hampshire 0.3798036   50
## 5          Oregon 0.9396843   42
## 6            Utah 0.7959810   45
## 7         Vermont 0.3196211   51
## 8         Wyoming 0.8871131   43
```

2. Reset murders to the original table by using data(murders). Use a pipe to create a new data frame called my_states that considers only states in the Northeast or West which have a murder rate lower

than 1, and contains only the state, rate and rank columns. The pipe should also have four components separated by three %>%. The code should look something like this:

```
#  my_states <- murders %>%
#  mutate SOMETHING %>%
#  filter SOMETHING %>%
#  select SOMETHING
```

The answer is below.

```
data(murders)
my_states <- murders %>%
    mutate(rate = total/population * 1e+05, rank = rank(-rate)) %>%
    filter(region %in% c("Northeast", "West") & rate < 1) %>%
    select(state, rate, rank)
my_states
```

```
##            state       rate rank
## 1         Hawaii 0.5145920   49
## 2          Idaho 0.7655102   46
## 3          Maine 0.8280881   44
## 4  New Hampshire 0.3798036   50
## 5         Oregon 0.9396843   42
## 6           Utah 0.7959810   45
## 7        Vermont 0.3196211   51
## 8        Wyoming 0.8871131   43
```