Reading in the Active Duty Marital Data

Sean Conway

8/19/2021

Marital DOD Data

The data come from data.gov. The file contains count data on the marital statuses of military service members, as well as their pay grade/family status. The data file is called `ActiveDuty_MaritalStatus.xls`.

BUT…

These data are not optimized for `R`.

Active Duty Family

Marital Status Report

Data Reflect Selection(s):
Select Service : Total DoD
Select Year : Apr-10

	Single Without Children			Single With Children			Joint Service Marriage			Civilian Marriage			Total		
Pay Grade	Male	Female	Total	Male	Female	Total	Male	Female	Total	Male	Female	Total	Male	Female	Total
E-1	31,229	5,717	36,946	563	122	685	139	141	280	5,060	719	5,779	36,991	6,699	43,690
E-2	53,094	8,388	61,482	1,457	275	1,732	438	579	1,017	12,483	1,682	14,165	67,472	10,924	78,396
E-3	131,091	21,019	152,110	4,264	1,920	6,184	3,579	4,902	8,481	54,795	6,641	61,436	193,729	34,482	228,211
E-4	112,710	16,381	129,091	9,491	4,662	14,153	8,661	9,778	18,439	105,556	9,961	115,517	236,418	40,782	277,200
E-5	57,989	11,021	69,010	10,937	6,576	17,513	12,459	11,117	23,576	130,944	8,592	139,536	212,329	37,306	249,635
E-6	19,125	4,654	23,779	10,369	4,962	15,331	8,474	6,961	15,435	110,322	5,827	116,149	148,290	22,404	170,694
E-7	5,446	1,913	7,359	6,530	2,585	9,115	5,065	3,291	8,356	70,001	3,206	73,207	87,042	10,995	98,037
E-8	1,009	438	1,447	1,786	513	2,299	1,423	651	2,074	21,079	820	21,899	25,297	2,422	27,719
E-9	381	202	583	579	144	723	458	150	608	8,215	291	8,506	9,633	787	10,420
TOTAL ENLISTED	412,074	69,733	481,807	45,976	21,759	67,735	40,696	37,570	78,266	518,455	37,739	556,194	1,017,201	166,801	1,184,002
O-1	13,495	3,081	16,576	402	229	631	426	669	1,095	6,959	828	7,787	21,282	4,807	26,089
O-2	11,029	2,715	13,744	426	299	725	910	1,194	2,104	10,070	1,096	11,166	22,435	5,304	27,739
O-3	14,551	5,056	19,607	1,442	940	2,382	3,017	3,174	6,191	38,963	3,886	42,849	57,973	13,056	71,029
O-4	3,480	1,720	5,200	1,190	534	1,724	1,958	1,639	3,597	31,864	2,416	34,280	38,492	6,309	44,801
O-5	1,244	810	2,054	729	267	996	1,072	806	1,878	22,296	1,578	23,874	25,341	3,461	28,802
O-6	353	349	702	261	94	355	364	182	546	10,004	715	10,719	10,982	1,340	12,322
O-7	5	7	12	7	1	8	9	6	15	410	18	428	431	32	463
O-8	4	7	11	0	0	0	7	2	9	272	16	288	283	25	308
O-9	1	1	2	1	0	1	1	1	2	144	1	145	147	3	150
O-10	1	0	1	0	0	0	1	1	2	35	0	35	37	1	38
TOTAL OFFICER	44,163	13,746	57,909	4,458	2,364	6,822	7,765	7,674	15,439	121,017	10,554	131,571	177,403	34,338	211,741
W-1	354	68	422	160	81	241	113	107	220	2,371	97	2,468	2,998	353	3,351
W-2	658	151	809	358	143	501	295	204	499	5,164	134	5,298	6,475	632	7,107
W-3	221	77	298	283	88	371	178	110	288	3,790	94	3,884	4,472	369	4,841
W-4	116	47	163	169	35	204	117	45	162	2,567	71	2,638	2,969	198	3,167
W-5	25	12	37	24	2	26	11	5	16	650	13	663	710	32	742
TOTAL WARRANT	1,374	355	1,729	994	349	1,343	714	471	1,185	14,542	409	14,951	17,624	1,584	19,208
GRAND TOTAL	457,611	83,834	541,445	51,428	24,472	75,900	49,175	45,715	94,890	654,014	48,702	702,716	1,212,228	202,723	1,414,951

Top of Page

TotalDoD AirForce MarineCorps Navy Army

So we will need to do some work to read in the data.

Reading in the data - First Excel Sheet

While eventually we will want to read in all the sheets at once, we will start out by reading in the first Excel Sheet, *TotalDoD* (see above).

We will use the `read_excel()` function from the `readxl` package.

First, we're going to manually specify the names of our columns. This involves doing a bit hard coding (reading in messy data is the only time hard coding is recommended!).

```r
col_names_dod <- c("pay_grade","single_withoutchildren_male",
                                  "single_withoutchildren_female",
                                  "single_withoutchildren_total",
                                  "single_withchildren_male",
                                  "single_withchildren_female",
                                  "single_withchildren_total",
                                  "married_jointservice_male",
                                  "married_jointservice_female",
                                  "married_jointservice_total",
                                  "married_civilian_female",
                                  "married_civilian_male",
                                  "married_civilian_total",
                                  "married_male_total",
                                  "married_female_total",
                                  "married_total_total")
```

Note that we named these columns so they can be adequately separated later on.

Next we have to use `read_excel()`, but we have to specify a number of argumnets. We first specify the `path` (note - `file_path` is a variable that I created ahead of time to be specific to my computer. Yours will be different). Next, we specify `sheet`, the number of the sheet we wish to read in (we can also specify the sheet name). Next, we carefully chooose the `range` of cells in the file we read in, based on our visual inspection of the file. This is another case where we must hard-code it. Here, we want the range to go from `B10` to `Q37`. We also need to manually specify `col_names` from our `col_names_dod` vector that we created above. Once we read it in, we will remove any rows containing the word "total" from the column `pay_grade`

```r
file_path
```

```
## [1] "/Users/seanconway/Github/Datasets/R/data/ActiveDuty_MaritalStatus.xls"
```

```r
marital_dod_1 <- read_excel(path=file_path,
                        sheet = 1, range = "B10:Q37",col_names = col_names_dod) %>%
  filter(str_detect(pay_grade, regex("total",ignore_case = TRUE),negate = T))
marital_dod_1
```

```
## # A tibble: 24 x 16
##    pay_grade single_withoutc~ single_withoutc~ single_withoutc~ single_withchil~
##    <chr>                <dbl>            <dbl>            <dbl>            <dbl>
##  1 E-1                  31229             5717            36946              563
##  2 E-2                  53094             8388            61482             1457
##  3 E-3                 131091            21019           152110             4264
##  4 E-4                 112710            16381           129091             9491
##  5 E-5                  57989            11021            69010            10937
##  6 E-6                  19125             4654            23779            10369
##  7 E-7                   5446             1913             7359             6530
##  8 E-8                   1009              438             1447             1786
##  9 E-9                    381              202              583              579
## 10 O-1                  13495             3081            16576              402
## # ... with 14 more rows, and 11 more variables:
## #   single_withchildren_female <dbl>, single_withchildren_total <dbl>,
## #   married_jointservice_male <dbl>, married_jointservice_female <dbl>,
## #   married_jointservice_total <dbl>, married_civilian_female <dbl>,
```

```
## #   married_civilian_male <dbl>, married_civilian_total <dbl>,
## #   married_male_total <dbl>, married_female_total <dbl>,
## #   married_total_total <dbl>
```

We've read in the data!

This `tibble` looks okay, but there's still much work to be done. First, we need to remove any of the columns that contain the word `"total"`. We don't need these aggregated totals, as they will only muddle the data (plus we can calculate them ourselves if needed).

```
marital_dod_2 <- marital_dod_1 %>%
  select(c(pay_grade,!contains("total")))
marital_dod_2
```

```
## # A tibble: 24 x 9
##    pay_grade single_withoutc~ single_withoutc~ single_withchil~ single_withchil~
##    <chr>                <dbl>            <dbl>            <dbl>            <dbl>
##  1 E-1                  31229             5717              563              122
##  2 E-2                  53094             8388             1457              275
##  3 E-3                 131091            21019             4264             1920
##  4 E-4                 112710            16381             9491             4662
##  5 E-5                  57989            11021            10937             6576
##  6 E-6                  19125             4654            10369             4962
##  7 E-7                   5446             1913             6530             2585
##  8 E-8                   1009              438             1786              513
##  9 E-9                    381              202              579              144
## 10 O-1                  13495             3081              402              229
## # ... with 14 more rows, and 4 more variables: married_jointservice_male <dbl>,
## #   married_jointservice_female <dbl>, married_civilian_female <dbl>,
## #   married_civilian_male <dbl>
```

Next, we'll use `pivot_longer()` to combine the column names (except for `pay_grade`) into a single column, `status`. We do this because the variable `status` is currently spread across columns (it is **wide**). We want our data to be tidy - where each row is a single observation.

We specify `cols` as every column **except** `pay_grade` with `!contains(pay_grade)`. We also specify that the column names will be moved to `status` and the column values will be moved to `count`.

```
marital_dod_3 <- marital_dod_2 %>%
  pivot_longer(cols = !contains("pay_grade"),
               names_to = "status", values_to = "count")
marital_dod_3
```

```
## # A tibble: 192 x 3
##    pay_grade status                       count
##    <chr>     <chr>                        <dbl>
##  1 E-1       single_withoutchildren_male  31229
##  2 E-1       single_withoutchildren_female  5717
##  3 E-1       single_withchildren_male       563
##  4 E-1       single_withchildren_female     122
##  5 E-1       married_jointservice_male      139
##  6 E-1       married_jointservice_female    141
##  7 E-1       married_civilian_female       5060
```

```
##   8 E-1       married_civilian_male           719
##   9 E-2       single_withoutchildren_male   53094
## 10 E-2       single_withoutchildren_female  8388
## # ... with 182 more rows
```

Our data is now tidy! Now we have just a bit more to do. We should use `separate()` to separate `pay_grade` into two columns (`enlisted` and `pay_grade`), as well as separate `status` into three columns (`relationship`, `family_status`, and `gender`).

```
marital_dod_tidy <- marital_dod_3 %>%
  separate(col=pay_grade, into=c("enlisted","pay_grade"),
           sep="-") %>%
  separate(col=status, into = c("relationship", "family_status","gender"),
           sep = "_")
marital_dod_tidy
```

```
## # A tibble: 192 x 6
##    enlisted pay_grade relationship family_status   gender count
##    <chr>    <chr>     <chr>        <chr>           <chr>  <dbl>
##  1 E        1         single       withoutchildren male   31229
##  2 E        1         single       withoutchildren female  5717
##  3 E        1         single       withchildren    male     563
##  4 E        1         single       withchildren    female   122
##  5 E        1         married      jointservice    male     139
##  6 E        1         married      jointservice    female   141
##  7 E        1         married      civilian        female  5060
##  8 E        1         married      civilian        male     719
##  9 E        2         single       withoutchildren male   53094
## 10 E        2         single       withoutchildren female  8388
## # ... with 182 more rows
```

Whala! We successfully read in and cleaned a very messy Excel spreadsheet. Next, we will use the `purrr` package to read multiple sheets at once.

Reading in the Data - All Sheets

As we saw, there are multiple sheets in `ActiveDuty_MaritalStatus.xls`. We will need to do a bit of work to read them all in, while maintaining best practices (i.e., **NOT** copying and pasting code).

First, we define the column names we want. Note that this is the same as above, but we need to include am empty string, since we will be forced to read in the "first" column (which is empty).

```
col_names_all <- c("", "pay_grade","single_withoutchildren_male",
              "single_withoutchildren_female",
              "single_withoutchildren_total",
              "single_withchildren_male",
              "single_withchildren_female",
              "single_withchildren_total",
              "married_jointservice_male",
              "married_jointservice_female",
              "married_jointservice_total",
              "married_civilian_female",
```

```
               "married_civilian_male",
               "married_civilian_total",
               "married_male_total",
               "married_female_total",
               "married_total_total")
```

Next, we use the `excel_sheets()` function to extract a character vector of all sheet names from our file

```
sheets <- excel_sheets(file_path)
sheets
```

```
## [1] "TotalDoD"    "AirForce"    "MarineCorps" "Navy"        "Army"
```

Next, is the bulk of our workload. We write a custom function that will read in each sheet, as well as clean and tidy it. The function is called `read_mar_sheets()`, and it takes a single argument: `sheet_name`.

The function first reads in the data, using `read_excel()`. Note that we specify the `path`from the `file_path` variable I created above. The `sheet` is specified by the user as `sheet_name`. We also ask `read_excel()` to trim whitespace (`trim_ws=TRUE`). The `col_names` are defined with `col_names_all` from above, and `skip=9` tells `read_excel()` to skip the first 9 rows (this number comes from a visual inspection of the Excel file).

After the sheet is read in, we use the pipe (`%>%`) to continue perfoming operations on it. We create a column `branch`, equal to the value of `sheet_name`. Then, we use `select(2:last_col())` to remove the first column from the data (recall that the first column is blank). We then use `select(c(pay_grade, !contains("total")))` to remove any columns with the word "total" in their name.

`filter(str_detect(pay_grade, regex("total",ignore_case = TRUE),negate = T))` removes the word "total" from the column `pay_grade`.

Just like above, we use `pivot_longer()` to make the data tidy, and then use `separate()` to separate out `pay_grade` and `status` into different columns.

This whole operation assigns a `tibble` to `data` (in the function scope), and a `return` statement returns this tibble to the user.

```
read_mar_sheets <- function(sheet_name){
  data <- read_excel(path = `file_path`,
             sheet = sheet_name,
             trim_ws = TRUE,
             col_names = col_names_all,
             skip=9) %>%
    mutate("branch"=sheet_name) %>%
    select(2:last_col()) %>% # immediately remove out blank column
    select(c(pay_grade,!contains("total"))) %>% # remove columns with the word total
    filter(str_detect(pay_grade, regex("total",ignore_case = TRUE),negate = T)) %>%
    pivot_longer(cols = !contains(c("pay_grade","branch")), # these columns can remain as is
               names_to = "status", values_to = "count") %>%
    separate(col=pay_grade, into=c("enlisted","pay_grade"),
             sep="-") %>%
    separate(col=status, into = c("relationship", "family_status","gender"),
           sep = "_")
  return(data)
}
```

Our function is obviously customized for this specific operation. It also contains some hard coding, which is necessary given the data file. However, note that the sheets are actually fairly consistent (e.g., the same amount of white space, very similar row values, etc.). In the future, you may encounter even trickier Excel files, which will require creative programming to read into `R`.

We can efficiently use this function with `purrr::map()` to read in all our sheets simulataneously. `purrr` is package that's part of the tidyverse, and it contains functions (like `map()` that allow us to vectorize operations.

Here is an example, using a simple custom function called `add_1`, which adds 1 to its input.

```
x <- c(1,2,3,4,5)
add_1 <- function(x) x+1
purrr::map(x, add_1)
```

```
## [[1]]
## [1] 2
##
## [[2]]
## [1] 3
##
## [[3]]
## [1] 4
##
## [[4]]
## [1] 5
##
## [[5]]
## [1] 6
```

Going back to vector `sheets`, we can use `purrr:map()` to vectorize our read-in operation. This wil create a list of tibbles, which can then be easily bound together into one large tibble using `bind_rows()`.

```
data_list <- purrr::map(sheets, read_mar_sheets)
data_list
```

```
## [[1]]
## # A tibble: 192 x 7
##    enlisted pay_grade branch   relationship family_status   gender count
##    <chr>    <chr>     <chr>    <chr>        <chr>           <chr>  <dbl>
##  1 E        1         TotalDoD single       withoutchildren male   31229
##  2 E        1         TotalDoD single       withoutchildren female  5717
##  3 E        1         TotalDoD single       withchildren    male     563
##  4 E        1         TotalDoD single       withchildren    female   122
##  5 E        1         TotalDoD married      jointservice    male     139
##  6 E        1         TotalDoD married      jointservice    female   141
##  7 E        1         TotalDoD married      civilian        female  5060
##  8 E        1         TotalDoD married      civilian        male     719
##  9 E        2         TotalDoD single       withoutchildren male   53094
## 10 E        2         TotalDoD single       withoutchildren female  8388
## # ... with 182 more rows
##
## [[2]]
## # A tibble: 152 x 7
##    enlisted pay_grade branch   relationship family_status   gender count
##    <chr>    <chr>     <chr>    <chr>        <chr>           <chr>  <dbl>
```

```
##  1 E        1         AirForce single       withoutchildren male    7721
##  2 E        1         AirForce single       withoutchildren female  1550
##  3 E        1         AirForce single       withchildren    male      27
##  4 E        1         AirForce single       withchildren    female     5
##  5 E        1         AirForce married      jointservice    male      49
##  6 E        1         AirForce married      jointservice    female    27
##  7 E        1         AirForce married      civilian        female  1064
##  8 E        1         AirForce married      civilian        male     178
##  9 E        2         AirForce single       withoutchildren male    4380
## 10 E        2         AirForce single       withoutchildren female  1010
## # ... with 142 more rows
##
## [[3]]
## # A tibble: 192 x 7
##    enlisted pay_grade branch      relationship family_status   gender count
##    <chr>    <chr>     <chr>       <chr>        <chr>           <chr>  <dbl>
##  1 E        1         MarineCorps single       withoutchildren male    6232
##  2 E        1         MarineCorps single       withoutchildren female   583
##  3 E        1         MarineCorps single       withchildren    male      54
##  4 E        1         MarineCorps single       withchildren    female     3
##  5 E        1         MarineCorps married      jointservice    male      20
##  6 E        1         MarineCorps married      jointservice    female    19
##  7 E        1         MarineCorps married      civilian        female   611
##  8 E        1         MarineCorps married      civilian        male      21
##  9 E        2         MarineCorps single       withoutchildren male   15916
## 10 E        2         MarineCorps single       withoutchildren female  1336
## # ... with 182 more rows
##
## [[4]]
## # A tibble: 184 x 7
##    enlisted pay_grade branch relationship family_status   gender count
##    <chr>    <chr>     <chr>  <chr>        <chr>           <chr>  <dbl>
##  1 E        1         Navy   single       withoutchildren male    7820
##  2 E        1         Navy   single       withoutchildren female  2275
##  3 E        1         Navy   single       withchildren    male     117
##  4 E        1         Navy   single       withchildren    female    34
##  5 E        1         Navy   married      jointservice    male      30
##  6 E        1         Navy   married      jointservice    female    57
##  7 E        1         Navy   married      civilian        female   806
##  8 E        1         Navy   married      civilian        male     162
##  9 E        2         Navy   single       withoutchildren male   11198
## 10 E        2         Navy   single       withoutchildren female  2718
## # ... with 174 more rows
##
## [[5]]
## # A tibble: 192 x 7
##    enlisted pay_grade branch relationship family_status   gender count
##    <chr>    <chr>     <chr>  <chr>        <chr>           <chr>  <dbl>
##  1 E        1         Army   single       withoutchildren male    9456
##  2 E        1         Army   single       withoutchildren female  1309
##  3 E        1         Army   single       withchildren    male     365
##  4 E        1         Army   single       withchildren    female    80
##  5 E        1         Army   married      jointservice    male      40
##  6 E        1         Army   married      jointservice    female    38
```

```
##  7 E        1         Army   married      civilian        female  2579
##  8 E        1         Army   married      civilian        male     358
##  9 E        2         Army   single       withoutchildren male   21600
## 10 E        2         Army   single       withoutchildren female  3324
## # ... with 182 more rows
```

```
marital_tidy_all <- data_list %>%
  bind_rows()
marital_tidy_all
```

```
## # A tibble: 912 x 7
##    enlisted pay_grade branch   relationship family_status   gender count
##    <chr>    <chr>     <chr>    <chr>        <chr>           <chr>  <dbl>
##  1 E        1         TotalDoD single       withoutchildren male   31229
##  2 E        1         TotalDoD single       withoutchildren female  5717
##  3 E        1         TotalDoD single       withchildren    male     563
##  4 E        1         TotalDoD single       withchildren    female   122
##  5 E        1         TotalDoD married      jointservice    male     139
##  6 E        1         TotalDoD married      jointservice    female   141
##  7 E        1         TotalDoD married      civilian        female  5060
##  8 E        1         TotalDoD married      civilian        male     719
##  9 E        2         TotalDoD single       withoutchildren male   53094
## 10 E        2         TotalDoD single       withoutchildren female  8388
## # ... with 902 more rows
```

Conclusion

You now know more about reading in messy data files to `R`. This is by no means the only solution to this problem. You may have even found a way that's more efficient!