Categorical variables

(Chriss Jordan Oboa)

Contents

Setup and packages

For this activity, we will just need to make use of the `tidyverse` package. Load it below.

```
library(tidyverse)
```

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

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

Loading in data

In the code chunk below, read in the dataset that is stored in the file `GSS_clean.csv`, which is located inside of the `data` folder here in RStudio Cloud. Give the dataset the name `GSS` and use the read_csv command.

```
GSS<-read.csv("data/GSS_clean.csv")
```

Categorical variables

This dataset is from the General Social Survey (GSS). From the GSS website,

"The General Social Survey (GSS) is a nationally representative survey of adults in the United States conducted since 1972. The GSS collects data on contemporary American society in order to monitor and explain trends in opinions, attitudes and behaviors. The GSS has adapted questions from earlier surveys, thereby allowing researchers to conduct comparisons for up to 80 years."

You can find more information about the GSS here: https://gss.norc.org/About-The-GSS.

Use `glimpse`, `head`, or `View` to take a look at the dataset.

How many cases are in this dataset? How many variables?

There are 2348 cases and 45 variables.

###Name some variables in this dataset that are categorical? Govt_or_Private_employee Marital_status College_major

Visualization for categorical variables

There are two main ways to visualize one categorical variable: **bar charts**, and pie charts. We're going to focus on bar charts here, because they are easier to create in R. Pie charts are generally not recommended – see the discussion on pie charts in our textbook.

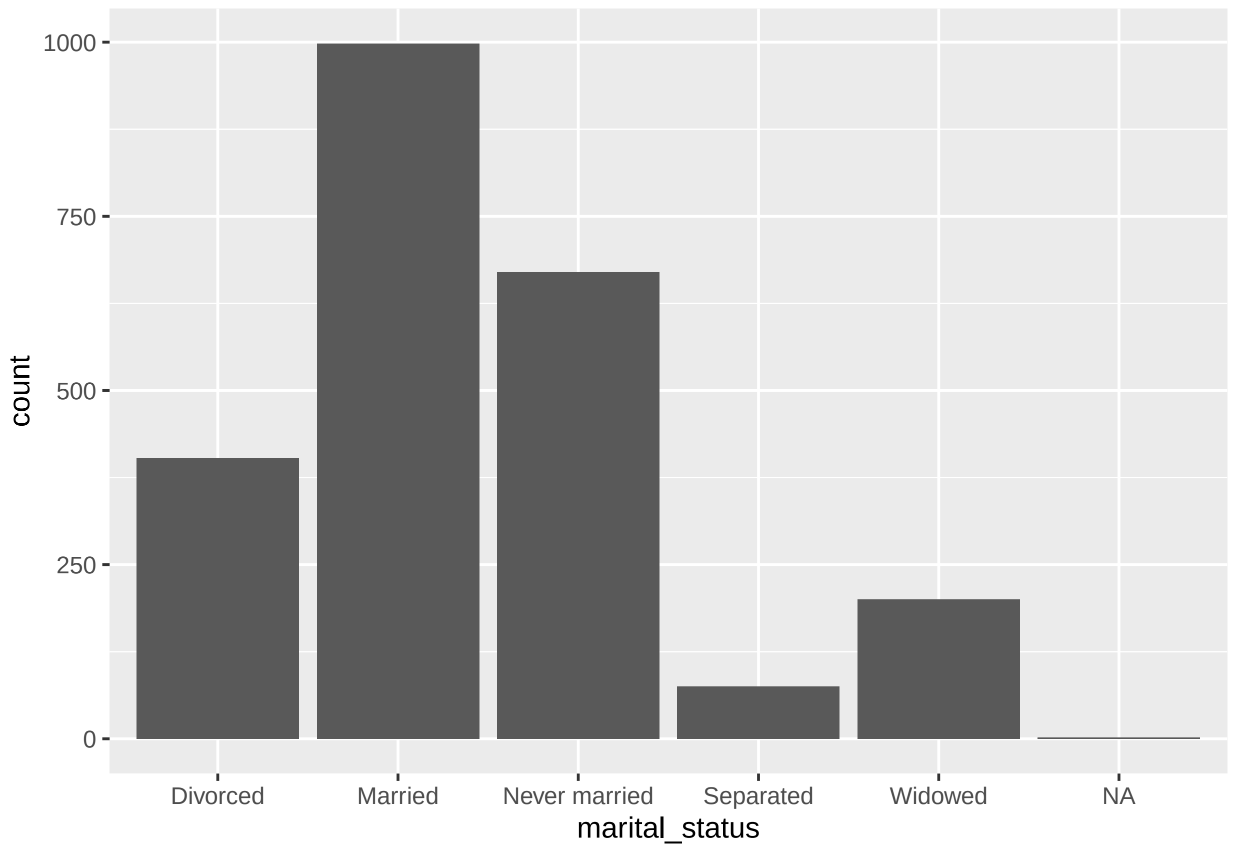
The variable we will focus on first is `marital_status` which gives the marital status of each respondent. To see the different categories or "levels" associated with this variable, we can do the following:

```
GSS %>%
  distinct(marital_status)
```

```
##   marital_status
## 1  Never married
## 2      Separated
## 3        Married
## 4       Divorced
## 5        Widowed
## 6           <NA>
```

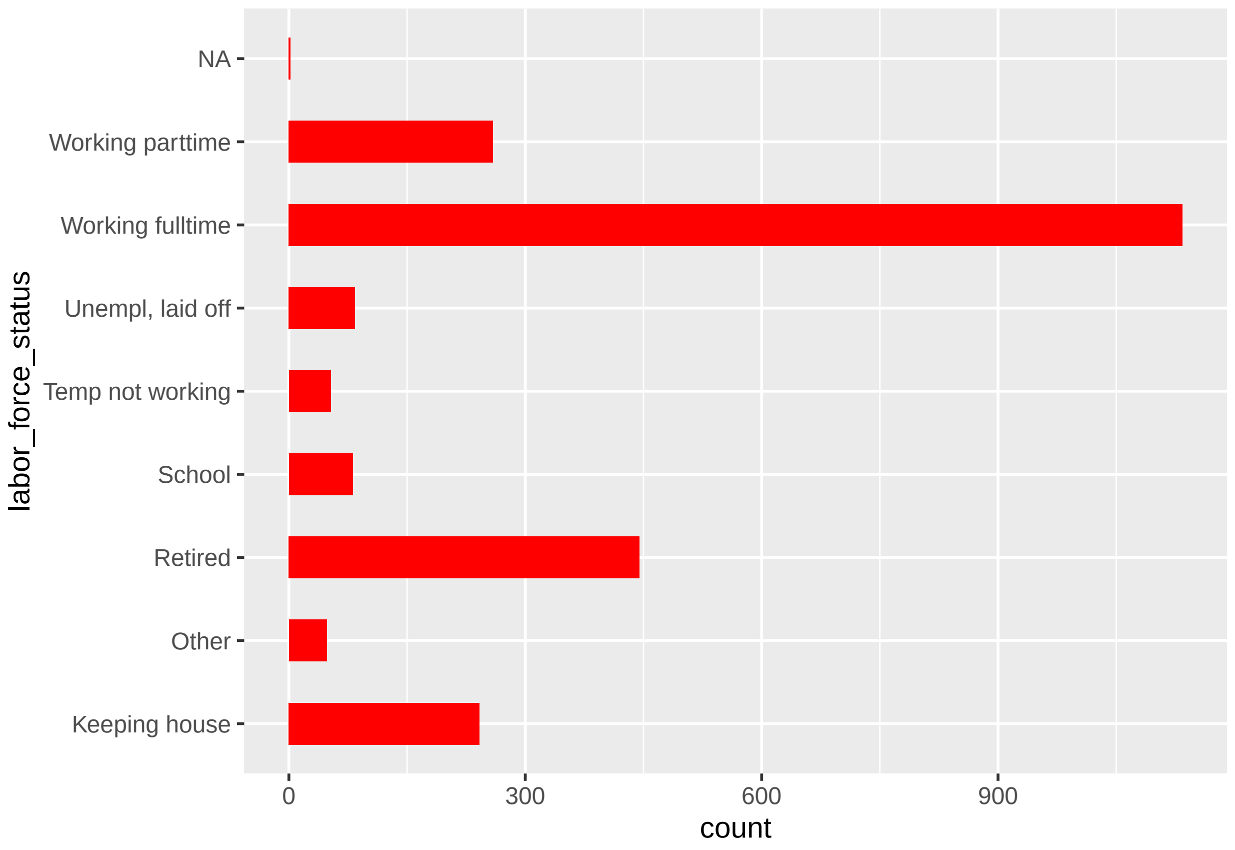
Next, we will again use `ggplot` to help us graph the distribution of a categorical variable in a bar plot.

```
ggplot(data = GSS, mapping = aes(x = marital_status)) +
  geom_bar()
```



Below, try making your own bar chart of the `labor_force_status`.

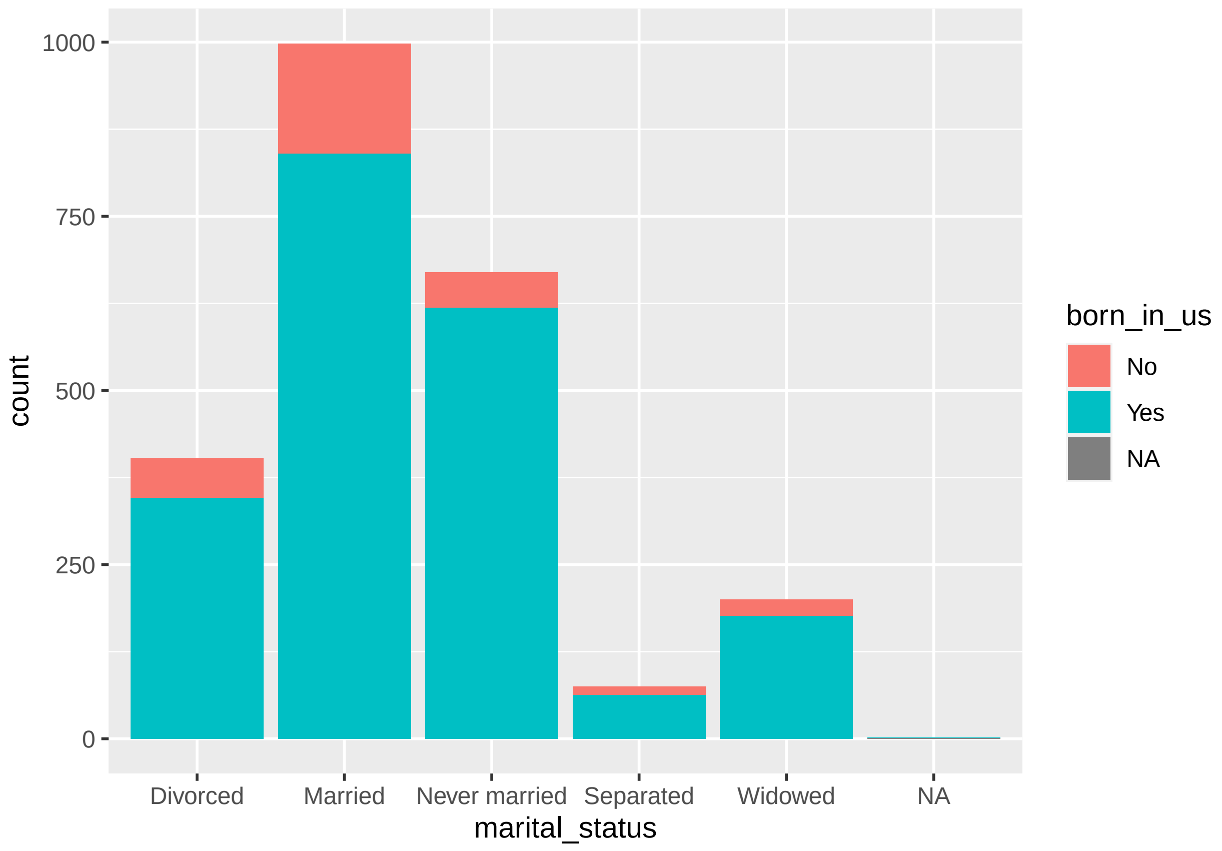
```r
ggplot(data = GSS, mapping = aes(y = labor_force_status)) +
  geom_bar(width=0.5, fill="red")
```



Two categorical variables

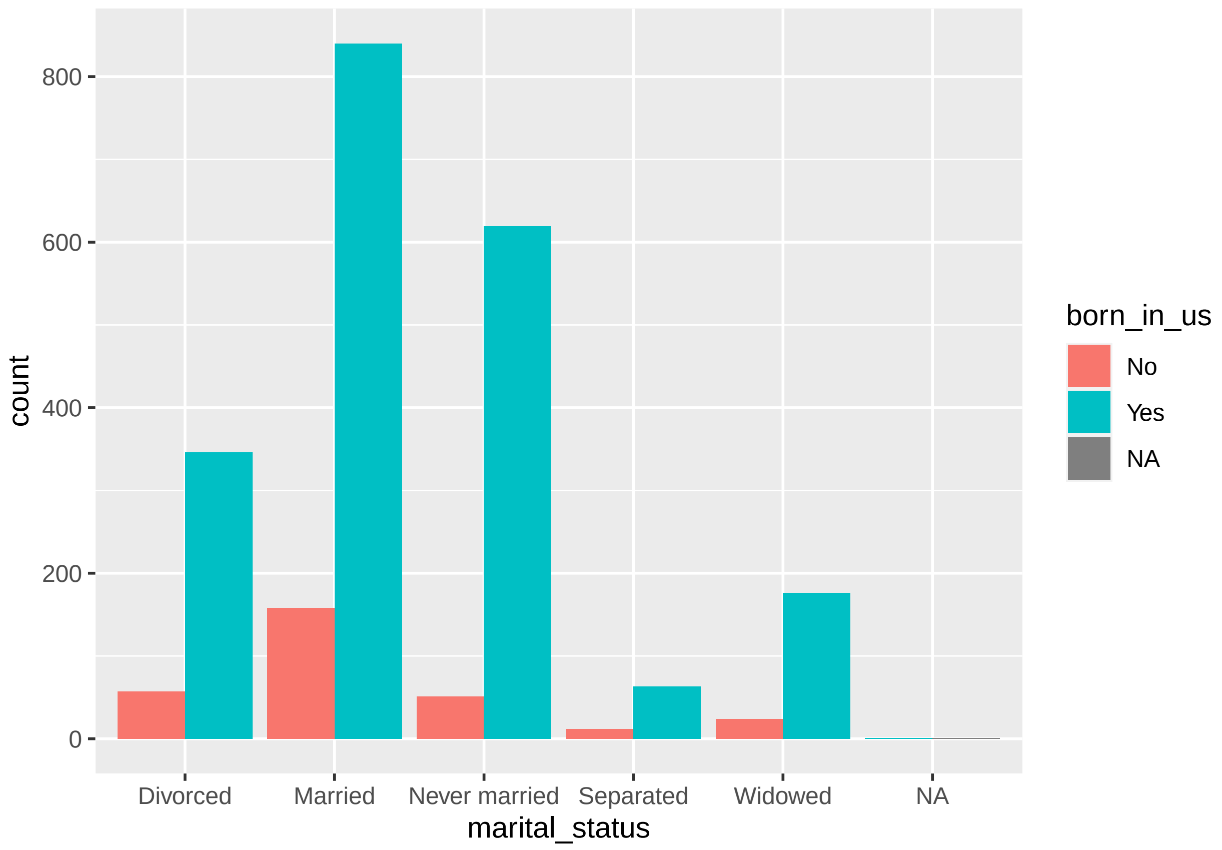
If we want, we can make a barchart showing **two** categorical variables. To do this, we add another argument to our `aes()` function. We will tell R to fill in the `marital_status` bars with color, according to the `born_in_us` variable. This makes a segmented or stacked barchart.

```
ggplot(data = GSS, mapping = aes(x = marital_status, fill = born_in_us)) +
  geom_bar()
```



We can also make this barchart a side-by-side or “dodged” barchart by providing the `position` argument to the geom_bar function.

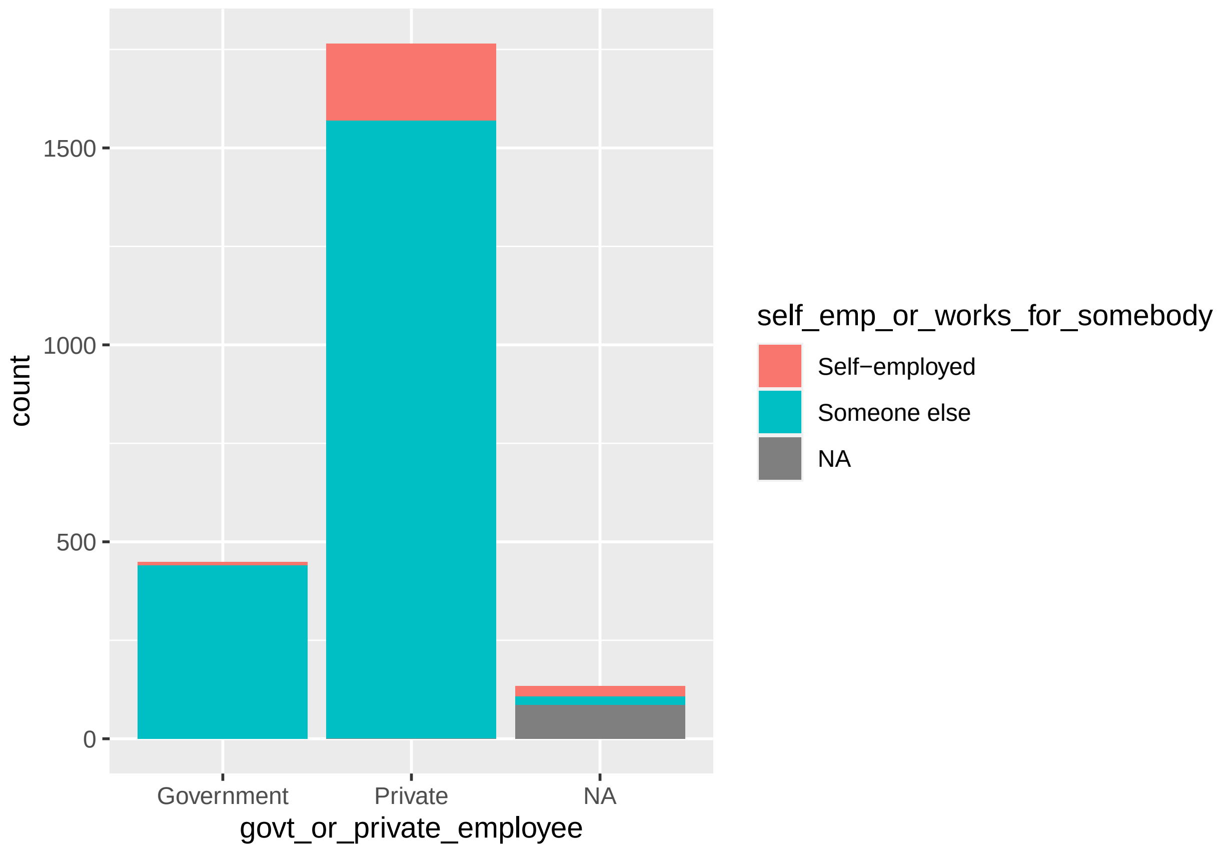
```
ggplot(data = GSS, mapping = aes(x = marital_status, fill = born_in_us)) +
  geom_bar(position = "dodge")
```



Try making a barchart that shows both the `govt_or_private_employee` and `self_emp_or_works_for_somebody` variables.

```
ggplot(data = GSS, mapping = aes(x = govt_or_private_employee, fill = self_emp_or_works_for_somebody))+
  geom_bar(potision = "dodge")
```

```
## Warning: Ignoring unknown parameters: potision
```



Summary statistics for categorical variables

The other part of Exploratory Data Analysis (EDA) is making summary statistics. For a categorical variable, this means making frequency tables and relative frequency tables. Let's explore how to do that.

We will use the pipe, `%>%`, to string together three main R functions:

- `summarize()`, which summarizes data (going from many rows down to just one or several)
- `mutate()`, which makes new variables in our dataset (sticking on one more column)
- `group_by()`, which works especially well with `summarize()` to group data for operations

Let's start with a short data pipeline.

```
GSS %>%
  summarize(n = n())
```

```
##      n
## 1 2348
```

What does this code tell us? how many variables are in the dataset.

We can add another piece to the pipeline to make it more interesting.

```
GSS %>%
  group_by(marital_status) %>%
  summarize(n = n())
```

```
## # A tibble: 6 x 2
##   marital_status     n
##   <chr>          <int>
## 1 Divorced         403
## 2 Married          998
## 3 Never married    670
## 4 Separated         75
## 5 Widowed          200
## 6 <NA>               2
```

```
#frequency table
```

What does this code give us? It gives us all type of marital status and the number of people that fall into each category.

Finally, we might want to add one more piece to the pipeline to create a relative frequency table,

```
GSS %>%
  group_by(marital_status)%>%
  summarize(n = n()) %>%
  mutate(proportion = n/sum(n))
```

```
## # A tibble: 6 x 3
##   marital_status     n proportion
##   <chr>          <int>      <dbl>
## 1 Divorced         403   0.172
## 2 Married          998   0.425
## 3 Never married    670   0.285
## 4 Separated         75   0.0319
## 5 Widowed          200   0.0852
## 6 <NA>               2   0.000852
```

Two categorical variables

We can extend this to two categorical variables by adding a second variable into our `group_by()` code. Let's make a relative frequency table showing the relationship between `marital_status` and `general_happiness`.

```
GSS %>%
  group_by(marital_status, general_happiness)%>%
  summarize(n = n()) %>%
  mutate(proportion = n/sum(n))
```

```
## `summarise()` has grouped output by 'marital_status'. You can override using the `.groups` argument.
```

```
## # A tibble: 20 x 4
## # Groups:   marital_status [6]
##    marital_status general_happiness     n proportion
##    <chr>          <chr>             <int>      <dbl>
##  1 Divorced       Not too happy        84    0.208
##  2 Divorced       Pretty happy        242    0.600
##  3 Divorced       Very happy           77    0.191
##  4 Married        Not too happy        61    0.0611
##  5 Married        Pretty happy        504    0.505
##  6 Married        Very happy          432    0.433
##  7 Married        <NA>                  1    0.00100
##  8 Never married  Not too happy       135    0.201
##  9 Never married  Pretty happy        409    0.610
## 10 Never married  Very happy          124    0.185
## 11 Never married  <NA>                  2    0.00299
## 12 Separated      Not too happy        19    0.253
## 13 Separated      Pretty happy         40    0.533
## 14 Separated      Very happy           15    0.2
## 15 Separated      <NA>                 1    0.0133
## 16 Widowed        Not too happy       37    0.185
## 17 Widowed        Pretty happy       111    0.555
## 18 Widowed        Very happy          52    0.26
## 19 <NA>           Pretty happy         1    0.5
## 20 <NA>           Very happy           1    0.5
```

What happens if you switch the order of the variables in the `group_by()`? of all the people that indicate they are divoced, only 20% are happy.

```r
GSS %>%
  group_by(general_happiness, marital_status)%>%
  summarize(n = n()) %>%
  mutate(proportion = n/sum(n))
```

```
## `summarise()` has grouped output by 'general_happiness'. You can override using the `.groups` argumen
```

```
## # A tibble: 20 x 4
## # Groups:   general_happiness [4]
##    general_happiness marital_status     n proportion
##    <chr>             <chr>          <int>      <dbl>
##  1 Not too happy     Divorced          84   0.25
##  2 Not too happy     Married           61   0.182
##  3 Not too happy     Never married    135   0.402
##  4 Not too happy     Separated         19   0.0565
##  5 Not too happy     Widowed           37   0.110
##  6 Pretty happy      Divorced         242   0.185
##  7 Pretty happy      Married          504   0.386
##  8 Pretty happy      Never married    409   0.313
##  9 Pretty happy      Separated         40   0.0306
## 10 Pretty happy      Widowed          111   0.0849
## 11 Pretty happy      <NA>               1   0.000765
## 12 Very happy        Divorced          77   0.110
## 13 Very happy        Married          432   0.616
## 14 Very happy        Never married    124   0.177
## 15 Very happy        Separated         15   0.0214
## 16 Very happy        Widowed           52   0.0742
## 17 Very happy        <NA>               1   0.00143
## 18 <NA>              Married            1   0.25
## 19 <NA>              Never married      2   0.5
## 20 <NA>              Separated          1   0.25
```

of all the people that indicate they are divoced, only 20% are happy.