In-class Assignment 8

Hai Shu (NetID: hs120)

Question 1 (1 pt): Tidy the simple tibble below. Do you need to make it wider or longer? What are the variables?

```
preg <- tribble(
  ~cancer, ~male, ~female,
  "yes",     NA,    10,
  "no",      20,    12
)
```

Answer: We need to make it longer; the two columns `male` and `female` should be made into a `gender` column.

```
preg %>%
  pivot_longer(cols = c('male', 'female'), names_to = 'gender', values_to = 'count')
```

```
## # A tibble: 4 x 3
##   cancer gender count
##   <chr>  <chr>  <dbl>
## 1 yes    male      NA
## 2 yes    female    10
## 3 no     male      20
## 4 no     female    12
```

Question 2 (1 pt): Both `unite()` and `separate()` have a `remove` argument. What does it do? Why would you set it to `FALSE`? Try the following code with this `remove` argument. In the following questions, `table3` to `table5` are available in the R package `tidyverse`.

```
table3 %>% separate(rate, into = c("cases", "population"))

table5 %>% unite(new, century, year, sep = "")
```

Answer: The `remove` argument drops the original input column(s), and is `TRUE` by default. You would set it to `FALSE` if you wanted to maintain the original input column(s).

```
table3 %>% separate(rate, into = c('cases', 'population'), remove = F)
```

```
## # A tibble: 6 x 5
##   country      year rate              cases  population
##   <chr>       <dbl> <chr>             <chr>  <chr>
## 1 Afghanistan  1999 745/19987071      745    19987071
## 2 Afghanistan  2000 2666/20595360     2666   20595360
## 3 Brazil       1999 37737/172006362   37737  172006362
## 4 Brazil       2000 80488/174504898   80488  174504898
## 5 China        1999 212258/1272915272 212258 1272915272
## 6 China        2000 213766/1280428583 213766 1280428583
```

```
table5 %>% unite(new, century, year, sep = "", remove = F)
```

```
## # A tibble: 6 x 5
##   country     new   century year  rate
##   <chr>       <chr> <chr>   <chr> <chr>
## 1 Afghanistan 1999  19      99    745/19987071
## 2 Afghanistan 2000  20      00    2666/20595360
## 3 Brazil      1999  19      99    37737/172006362
## 4 Brazil      2000  20      00    80488/174504898
## 5 China       1999  19      99    212258/1272915272
## 6 China       2000  20      00    213766/1280428583
```

Question 3 (1 pt): Sometimes when a data source has primarily been used for data entry, missing values indicate that the previous value should be carried forward (see the 1st chunk below). You can fill in these missing values with `fill()`. It takes a set of columns where you want missing values to be replaced by the most recent non-missing value (sometimes called last observation carried forward); see the 2nd chunk below. The function `fill()` has a `.direction` argument. What does it do? Try the 2nd chunk with different options of `.direction`.

```
treatment <- tribble(
  ~ person,           ~ treatment, ~response,
  "Derrick Whitmore", 1,           7,
  NA,                 2,           10,
  NA,                 3,           9,
  "Katherine Burke",  1,           4
)
```

```
treatment %>% fill(person)
```

```
## # A tibble: 4 x 3
##   person           treatment response
##   <chr>                <dbl>    <dbl>
## 1 Derrick Whitmore         1        7
## 2 Derrick Whitmore         2       10
## 3 Derrick Whitmore         3        9
## 4 Katherine Burke          1        4
```

Answer: The `.direction` argument indicates how the missing values will be filled. `down` means the missing values will be replaced by the last non-missing value above them, while `up` means they will be replaced by the first non-missing value below them.

```r
treatment %>% fill(person, .direction = 'down')
```

```
## # A tibble: 4 x 3
##   person           treatment response
##   <chr>                <dbl>    <dbl>
## 1 Derrick Whitmore         1        7
## 2 Derrick Whitmore         2       10
## 3 Derrick Whitmore         3        9
## 4 Katherine Burke          1        4
```

```r
treatment %>% fill(person, .direction = 'up')
```

```
## # A tibble: 4 x 3
##   person           treatment response
##   <chr>                <dbl>    <dbl>
## 1 Derrick Whitmore         1        7
## 2 Katherine Burke          2       10
## 3 Katherine Burke          3        9
## 4 Katherine Burke          1        4
```

```r
treatment %>% fill(person, .direction = 'downup')
```

```
## # A tibble: 4 x 3
##   person           treatment response
##   <chr>                <dbl>    <dbl>
## 1 Derrick Whitmore         1        7
## 2 Derrick Whitmore         2       10
## 3 Derrick Whitmore         3        9
## 4 Katherine Burke          1        4
```

```r
treatment %>% fill(person, .direction = 'updown')
```

```
## # A tibble: 4 x 3
##   person           treatment response
##   <chr>                <dbl>    <dbl>
## 1 Derrick Whitmore         1        7
## 2 Katherine Burke          2       10
## 3 Katherine Burke          3        9
## 4 Katherine Burke          1        4
```

Question 4 (1 pt): Create the data frame `table1_na` using the following code chunk. Use `replace_with_na_all()` in R package `naniar` to replace all non-standard missing values with `NA` in the data frame.

```r
table1_na <- table1
table1_na$cases <- as.character(table1_na$cases)
table1_na$population <- as.character(table1_na$population)
table1_na$cases[c(2,4,6)] <- c("N/A", "na", ".")
table1_na$population[c(1,6)] <- c(NA, "--")
table1_na
```

```
## # A tibble: 6 x 4
##   country      year cases  population
##   <chr>       <dbl> <chr>  <chr>
## 1 Afghanistan  1999 745    <NA>
## 2 Afghanistan  2000 N/A    20595360
## 3 Brazil       1999 37737  172006362
## 4 Brazil       2000 na     174504898
## 5 China        1999 212258 1272915272
## 6 China        2000 .      --
```

```
library(naniar)
```

Answer:

```
## Warning: package 'naniar' was built under R version 4.3.3
```

```
table1_na %>% replace_with_na_all(~.x %in% c("N/A", "na", ".", '--'))
```

```
## # A tibble: 6 x 4
##   country      year cases  population
##   <chr>       <dbl> <chr>  <chr>
## 1 Afghanistan  1999 745    <NA>
## 2 Afghanistan  2000 <NA>   20595360
## 3 Brazil       1999 37737  172006362
## 4 Brazil       2000 <NA>   174504898
## 5 China        1999 212258 1272915272
## 6 China        2000 <NA>   <NA>
```

Question 5 (1 pt): After the processing in Question 4, you may notice that the columns `cases` and `population` of `table1_na` are in the `character` type. Change the two columns into the `integer` type.

```
table1_na %>% mutate(across(cases:population, as.numeric))
```

Answer:

```
## Warning: There were 2 warnings in `mutate()`.
## The first warning was:
## i In argument: `across(cases:population, as.numeric)`.
## Caused by warning:
## ! NAs introduced by coercion
## i Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <dbl>  <dbl>      <dbl>
## 1 Afghanistan  1999    745         NA
## 2 Afghanistan  2000     NA   20595360
## 3 Brazil       1999  37737  172006362
## 4 Brazil       2000     NA  174504898
## 5 China        1999 212258 1272915272
## 6 China        2000     NA         NA
```