Week2_Homework

2162398

2022-10-12

Part 2

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6      v purrr   0.3.5
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

2.1Load in well-being-wide.csv using read_csv as wb1. Your code should assume that the file is in the current working directory

```
#1. Load in well-being-wide.csv #wd:"/Users/leo/Documents/GitHub/PS923-Warwick-BES"
wb1=read_csv("well-being-wide.csv")
```

```
## Rows: 8 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): sub, dep
## dbl (5): t1, t2, t3, t4, t5
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

1.2 Create a new tidy version of wb1 called wb2 (i.e., from wide into long format). Specifically, collate columns t1 to t5, call the name column time and the value column well_being. I recommend usingthe tidy data (i.e., wb2) for the following tasks.

```
#wb1-wb2 Convert w1 to tidy version
wb2=wb1%>%
  pivot_longer(!sub & !dep,
  names_to = "time",values_to = "well_being")%>%
  arrange(time,sub,dep,well_being)
  str(wb2)
```

```
## tibble [40 x 4] (S3: tbl_df/tbl/data.frame)
##  $ sub       : chr [1:40] "s1" "s2" "s3" "s4" ...
##  $ dep       : chr [1:40] "Psych" "Psych" "Psych" "Psych" ...
##  $ time      : chr [1:40] "t1" "t1" "t1" "t1" ...
##  $ well_being: num [1:40] 6 4 8 4 8 7 6 7 6 5 ...
```

```
  #wb2_test Compare the created wb2 data frames to meet the homework requirements
wb2_test=read_csv("well-being-long.csv")
```

```
## Rows: 40 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): sub, dep, time
## dbl (1): well_being
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
str(wb2_test)
```

```
## spec_tbl_df [40 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ sub       : chr [1:40] "s1" "s2" "s3" "s4" ...
##  $ dep       : chr [1:40] "Psych" "Psych" "Psych" "Psych" ...
##  $ time      : chr [1:40] "t1" "t1" "t1" "t1" ...
##  $ well_being: num [1:40] 6 4 8 4 8 7 6 7 6 5 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   sub = col_character(),
##   ..   dep = col_character(),
##   ..   time = col_character(),
##   ..   well_being = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

1.3 What is the mean well-being at each time point?

```
mean_by_timepoint=wb2%>%
  group_by(time)%>%
  summarise(mean=mean(well_being))
 mean_by_timepoint
```

```
## # A tibble: 5 x 2
##   time   mean
##   <chr> <dbl>
## 1 t1     6.25
## 2 t2     5.75
## 3 t3     7.38
## 4 t4     8
## 5 t5     7.62
```

1.4 What is the mean well-being at each time point, separated by dep?

```
mean_by_dep=wb2%>%
  group_by(dep,time)%>%
  summarise(mean=mean(well_being))
```

```
## ‘summarise()‘ has grouped output by ’dep’. You can override using the ‘.groups‘
## argument.
```

```
mean_by_dep
```

```
## # A tibble: 10 x 3
## # Groups:   dep [2]
##    dep   time   mean
##    <chr> <chr> <dbl>
##  1 Econ  t1     7
##  2 Econ  t2     6.5
##  3 Econ  t3     8.25
##  4 Econ  t4     9
##  5 Econ  t5     8.25
##  6 Psych t1     5.5
##  7 Psych t2     5
##  8 Psych t3     6.5
##  9 Psych t4     7
## 10 Psych t5     7
```

1.5 Create a new tidy data.frame (or tibble), called wb_pre, that only contains the data from time points t1 and t2.

```
wb_pre=wb2%>%
  mutate(time=factor(time,c("t1","t2")),
         sub=factor(sub),
         dep=factor(dep))%>%
         filter(!is.na(time))
wb_pre
```

```
## # A tibble: 16 x 4
##    sub   dep   time  well_being
##    <fct> <fct> <fct>      <dbl>
##  1 s1    Psych t1             6
##  2 s2    Psych t1             4
##  3 s3    Psych t1             8
##  4 s4    Psych t1             4
##  5 s5    Econ  t1             8
##  6 s6    Econ  t1             7
##  7 s7    Econ  t1             6
##  8 s8    Econ  t1             7
##  9 s1    Psych t2             6
## 10 s2    Psych t2             5
## 11 s3    Psych t2             8
## 12 s4    Psych t2             1
## 13 s5    Econ  t2             7
## 14 s6    Econ  t2             7
## 15 s7    Econ  t2             5
## 16 s8    Econ  t2             7
```

1.6 Add a new column well_being_z to wb2 that contains the by-department standarised (also called z-standardised) well-being values [standardised = (value - mean(value)) / sd(value))].

```
wb2=wb2%>%
  group_by(dep)%>%
  mutate(well_being_z=(well_being-mean(well_being))/sd(well_being))
wb2
```

```
## # A tibble: 40 x 5
## # Groups:   dep [2]
##    sub   dep   time  well_being well_being_z
##    <chr> <chr> <chr>      <dbl>        <dbl>
##  1 s1    Psych t1             6      -0.0875
##  2 s2    Psych t1             4      -0.963
##  3 s3    Psych t1             8       0.788
##  4 s4    Psych t1             4      -0.963
##  5 s5    Econ  t1             8       0.161
##  6 s6    Econ  t1             7      -0.645
##  7 s7    Econ  t1             6      -1.45
##  8 s8    Econ  t1             7      -0.645
##  9 s1    Psych t2             6      -0.0875
## 10 s2    Psych t2             5      -0.525
## # ... with 30 more rows
```

1.7 Add a new column pre_post to wb2 that has the value “pre” for time t1 and t2 and post otherwise (i.e., for time t3 to t5). Thus, this column contains the information of whether the data is from before or after the intervention

```
wb2=wb2%>%
  mutate(pre_post=
  case_when(time=="t1"~"pre",
   time=="t2"~"pre",
  time!="t1"~"post"))
wb2
```

```
## # A tibble: 40 x 6
## # Groups:   dep [2]
##    sub   dep   time  well_being well_being_z pre_post
##    <chr> <chr> <chr>      <dbl>        <dbl> <chr>
##  1 s1    Psych t1             6      -0.0875 pre
##  2 s2    Psych t1             4      -0.963  pre
##  3 s3    Psych t1             8       0.788  pre
##  4 s4    Psych t1             4      -0.963  pre
##  5 s5    Econ  t1             8       0.161  pre
##  6 s6    Econ  t1             7      -0.645  pre
##  7 s7    Econ  t1             6      -1.45   pre
##  8 s8    Econ  t1             7      -0.645  pre
##  9 s1    Psych t2             6      -0.0875 pre
## 10 s2    Psych t2             5      -0.525  pre
## # ... with 30 more rows
```

1.8 Create a new tibble with the name av_wb, that has the average well-being as well as the average standardised well-being before the intervention (i.e., average of t1 and t2) and after the intervention (i.e., average of t3 to t5) per participant. This tibble should also contain the dep variable.

```
ab_wb=wb2%>%
  group_by(pre_post,sub,dep)%>%
  summarise(mean=mean(well_being),
            mean_z=mean(well_being_z))
```

```
## ‘summarise()‘ has grouped output by ’pre_post’, ’sub’. You can override using
## the ‘.groups‘ argument.
```

```
ab_wb
```

```
## # A tibble: 16 x 5
## # Groups:   pre_post, sub [16]
##    pre_post sub   dep    mean  mean_z
##    <chr>    <chr> <chr> <dbl>   <dbl>
##  1 post     s1    Psych  8     0.788
##  2 post     s2    Psych  6.67  0.204
##  3 post     s3    Psych  9     1.23
##  4 post     s4    Psych  3.67 -1.11
##  5 post     s5    Econ   8.33  0.430
##  6 post     s6    Econ   9.33  1.24
##  7 post     s7    Econ   7.33 -0.376
##  8 post     s8    Econ   9     0.968
##  9 pre      s1    Psych  6    -0.0875
## 10 pre      s2    Psych  4.5  -0.744
## 11 pre      s3    Psych  8     0.788
## 12 pre      s4    Psych  2.5  -1.62
## 13 pre      s5    Econ   7.5  -0.242
## 14 pre      s6    Econ   7    -0.645
## 15 pre      s7    Econ   5.5  -1.86
## 16 pre      s8    Econ   7    -0.645
```

1.9 Calculate the difference between average pre and average post well-being value for each participant you created in task 8. Which participant has the largest and which has the smallest difference?

```
#s6 has the lagrgest difference and s5 has the smallest difference
diff_wb=ab_wb%>%
  select(-mean_z)%>%
  spread(pre_post,mean)%>%
  mutate(diff=post-pre)
  #Find the participant with the maximum value of the difference
  max_diff_wb=diff_wb%>%
    group_by(sub)%>%
   summarise(max=max(diff),)%>%
   arrange(diff_wb,desc(max))%>%
   select(sub,max)
  #Find the participant with the minimum value of the difference
  min_diff_wb=diff_wb%>%
    group_by(sub)%>%
   summarise(min=min(diff))%>%
   arrange(diff_wb,desc(min))%>%
   select(sub,min)
```

```
max_diff_wb
```

```
## # A tibble: 8 x 2
##   sub     max
##   <chr> <dbl>
## 1 s1    2
## 2 s2    2.17
## 3 s3    1
## 4 s4    1.17
## 5 s5    0.833
## 6 s6    2.33
## 7 s7    1.83
## 8 s8    2
```

```
min_diff_wb
```

```
## # A tibble: 8 x 2
##   sub     min
##   <chr> <dbl>
## 1 s1    2
## 2 s2    2.17
## 3 s3    1
## 4 s4    1.17
## 5 s5    0.833
## 6 s6    2.33
## 7 s7    1.83
## 8 s8    2
```

```
diff_wb
```

```
## # A tibble: 8 x 5
## # Groups:   sub [8]
##   sub   dep    post   pre  diff
##   <chr> <chr> <dbl> <dbl> <dbl>
## 1 s1    Psych  8      6   2
## 2 s2    Psych  6.67   4.5 2.17
## 3 s3    Psych  9      8   1
## 4 s4    Psych  3.67   2.5 1.17
## 5 s5    Econ   8.33   7.5 0.833
## 6 s6    Econ   9.33   7   2.33
## 7 s7    Econ   7.33   5.5 1.83
## 8 s8    Econ   9      7   2
```

1.10 Calculate the difference between average pre and average post standardised well-being value for each participant created in task 8. Which participant has the largest and which has the smallest difference? Does the same participant have the largest or smallest unstandardised and standardised well-being scores? If not, what does it mean and how can it happen?

```
#s6 has the lagrgest unstandardised and standardised well-being scores
#
```

```
z_diff_wb=ab_wb%>%
  select(-mean)%>%
  spread(pre_post,mean_z)%>%
  mutate(diff=post-pre)

z_max_diff_wb=z_diff_wb%>%
  summarise(max=max(diff))%>%
  arrange(z_diff_wb,max)%>%
  select(sub,max)


z_min_diff_wb=z_diff_wb%>%
  summarise(min=min(diff))%>%
  arrange(z_diff_wb,desc(min))%>%
  select(sub,min)

z_diff_wb
```

```
## # A tibble: 8 x 5
## # Groups:   sub [8]
##   sub   dep     post     pre  diff
##   <chr> <chr>  <dbl>   <dbl> <dbl>
## 1 s1    Psych  0.788 -0.0875 0.875
## 2 s2    Psych  0.204 -0.744  0.948
## 3 s3    Psych  1.23   0.788  0.438
## 4 s4    Psych -1.11  -1.62   0.511
## 5 s5    Econ   0.430 -0.242  0.672
## 6 s6    Econ   1.24  -0.645  1.88
## 7 s7    Econ  -0.376 -1.86   1.48
## 8 s8    Econ   0.968 -0.645  1.61
```

```
z_max_diff_wb
```

```
## # A tibble: 8 x 2
##   sub     max
##   <chr> <dbl>
## 1 s1    0.875
## 2 s2    0.948
## 3 s3    0.438
## 4 s4    0.511
## 5 s5    0.672
## 6 s6    1.88
## 7 s7    1.48
## 8 s8    1.61
```

```
z_min_diff_wb
```

```
## # A tibble: 8 x 2
##   sub     min
##   <chr> <dbl>
## 1 s1    0.875
## 2 s2    0.948
## 3 s3    0.438
## 4 s4    0.511
## 5 s5    0.672
## 6 s6    1.88
## 7 s7    1.48
## 8 s8    1.61
```

11. Calculate the average difference well-being score and also the average standardised difference score. Do you think the difference is larger than zero, when taking the variability in the data into account?

```
av_wb=diff_wb%>%
  summarise(av_diff=mean(diff))
av_wb_z=z_diff_wb%>%
  summarise(av_diff_z=mean(diff))
av_wb
```

```
## # A tibble: 8 x 2
##   sub   av_diff
##   <chr>   <dbl>
## 1 s1      2
## 2 s2      2.17
## 3 s3      1
## 4 s4      1.17
## 5 s5      0.833
## 6 s6      2.33
## 7 s7      1.83
## 8 s8      2
```

```
av_wb_z
```

```
## # A tibble: 8 x 2
##   sub   av_diff_z
##   <chr>     <dbl>
## 1 s1        0.875
## 2 s2        0.948
## 3 s3        0.438
## 4 s4        0.511
## 5 s5        0.672
## 6 s6        1.88
## 7 s7        1.48
## 8 s8        1.61
```

Part 2

2.1 Write a function h1 which uses a for-loop and the explicit formulation

```
h1=function(x,n){
  result=1
 for (i in 1:n) {
   result=result+x^i

 }
  return(result)
}
```

2.2 Write a function h2 which uses a while-loop and the explicit formulation

```
h2=function(x,n){
  result=1
  i=1
  while (i<=n) {
    result=result+x**i
    i=i+1

  }
  return(result)
}
```

2.3Write a function h3 which uses no loop, but vectorised operations for the explicit formulation

```
h3=function(x,n){
  vector=c(0:n)
  result=sum(x^vector)
  return(result)
}
```

2.4

```
h4=function(x,n){
  #when x=1
  if(x==1){
    n_tmp=c(0:n)
    return(sum(x^n_tmp))

  }else{
    return(1-x^(n+1)/(1-x))
  }
}
```

2.5 Check your functions h1, h2, h3, and h4 against the values given in the following table.

```
#h1-testing
h1(0.3,55)
```

```
## [1] 1.428571
```

```
h1(6.6,8)
```

```
## [1] 4243336
```

```
h1(1,12)
```

```
## [1] 13
```

```
#h2-testing
h2(0.3,55)
```

```
## [1] 1.428571
```

```
h2(6.6,8)
```

```
## [1] 4243336
```

```
h2(1,12)
```

```
## [1] 13
```

```
#h3-testing
h3(0.3,55)
```

```
## [1] 1.428571
```

```
h3(6.6,8)
```

```
## [1] 4243336
```

```
h3(1,12)
```

```
## [1] 13
```

```
#h4-testing
h4(0.3,55)
```

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

```
h4(6.6,8)
```

```
## [1] 4243337
```

```
h4(1,12)
```

```
## [1] 13
```