Homework 9

Zahlen Zbinden

2024-03-04

```
library(ggplot2)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.3     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v lubridate 1.9.2     v tibble    3.2.1
v purrr     1.0.2     v tidyr     1.3.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```
library(nycflights13)
```

```
Warning: package 'nycflights13' was built under R version 4.3.2
```

1. In R, the function t.test() conducts one and two sample t-tests. For instance the following code runs Welch's two sample t-test using the sleep data in R.

```
my_test_output <- t.test(extra ~ group, data = sleep)
my_test_output
```

```

    Welch Two Sample t-test

data:  extra by group
```

```
t = -1.8608, df = 17.776, p-value = 0.07939
alternative hypothesis: true difference in means between group 1 and group 2 is not equal to
95 percent confidence interval:
 -3.3654832  0.2054832
sample estimates:
mean in group 1 mean in group 2
           0.75            2.33
```

Verify that my_test_output is built on top of a list. Then, return the names of the elements of that list.

```
# verifty that my_test_output is a list
is.list(my_test_output)
```

```
[1] TRUE
```

```
# return the names of the elements in the list
names(my_test_output)
```

```
 [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
 [6] "null.value"  "stderr"      "alternative" "method"      "data.name"
```

Turn your code from the previous tasks into a function called conf_int() that, extracts the confidence intervale values from any t.test() output

```
conf_int <- function(test_object) {
  return(test_object$conf.int)
}
```

```
conf_int(my_test_output)
```

```
[1] -3.3654832  0.2054832
attr(,"conf.level")
[1] 0.95
```

2. The following code is an example of taking two vectors of the same length and joining them together element wise to createa a single character vector

```r
farm <- c(1, 1, 2, 2, 3, 4)
field <- c("a", "b", "a", "b", "a", "a")
paste(farm, field, sep = "_")
```

```
[1] "1_a" "1_b" "2_a" "2_b" "3_a" "4_a"
```

For instance, you might want to use this to generatea singel identifiying variable from a couple of variables.

Turn this code into a function called join_with_underscore(), that takes two vecotrs x and y as input, and joins them into a single character string.

```r
join_with_underscore <- function(list1, list2) {
  return(paste(list1, list2, sep = "_"))
}
```

```r
# check that it works by testing with farm and field
join_with_underscore(farm, field)
```

```
[1] "1_a" "1_b" "2_a" "2_b" "3_a" "4_a"
```

3. Reduce the repetition in this code by using across():

```r
starwars |>
  mutate(
    n_films = length(films),
    n_vehicles = length(vehicles),
    n_starships = length(starships)
  )
```

```
# A tibble: 87 x 17
   name     height  mass hair_color skin_color eye_color birth_year sex   gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
 1 Luke Sk~    172    77 blond      fair       blue            19   male  mascu~
 2 C-3PO       167    75 <NA>       gold       yellow         112   none  mascu~
 3 R2-D2        96    32 <NA>       white, bl~ red             33   none  mascu~
 4 Darth V~    202   136 none       white      yellow          41.9 male  mascu~
 5 Leia Or~    150    49 brown      light      brown           19   fema~ femin~
 6 Owen La~    178   120 brown, gr~ light      blue            52   male  mascu~
 7 Beru Wh~    165    75 brown      light      blue            47   fema~ femin~
 8 R5-D4        97    32 <NA>       white, red red              NA   none  mascu~
 9 Biggs D~    183    84 black      light      brown            24   male  mascu~
10 Obi-Wan~    182    77 auburn, w~ fair       blue-gray        57   male  mascu~
# i 77 more rows
# i 8 more variables: homeworld <chr>, species <chr>, films <list>,
#   vehicles <list>, starships <list>, n_films <int>, n_vehicles <int>,
#   n_starships <int>
```

starwars |> mutate(across(films:starships, length()))

```
starwars |>
  mutate(
    across(films:starships, length)
  )
```

```
# A tibble: 87 x 14
   name     height  mass hair_color skin_color eye_color birth_year sex   gender
   <chr>     <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
 1 Luke Sk~    172    77 blond      fair       blue            19   male  mascu~
 2 C-3PO       167    75 <NA>       gold       yellow         112   none  mascu~
 3 R2-D2        96    32 <NA>       white, bl~ red             33   none  mascu~
 4 Darth V~    202   136 none       white      yellow          41.9 male  mascu~
 5 Leia Or~    150    49 brown      light      brown           19   fema~ femin~
 6 Owen La~    178   120 brown, gr~ light      blue            52   male  mascu~
 7 Beru Wh~    165    75 brown      light      blue            47   fema~ femin~
 8 R5-D4        97    32 <NA>       white, red red              NA   none  mascu~
 9 Biggs D~    183    84 black      light      brown           24   male  mascu~
10 Obi-Wan~    182    77 auburn, w~ fair       blue-gray       57   male  mascu~
# i 77 more rows
# i 5 more variables: homeworld <chr>, species <chr>, films <int>,
#   vehicles <int>, starships <int>
```

4. Reduce the repition in this code, by writing two functions and using across()

```
set.seed(1846689310)
flights_small <- flights |>
                 slice(sample(n(), size = 10))
```

```
flights_small |>
  mutate(
    sched_arr_time_hour = stringr::str_sub(sched_arr_time, start = -4, end = -3) |> parse_
    sched_arr_time_min = stringr::str_sub(sched_arr_time, start = -2, end = -1) |> parse_n
    arr_time_house = stringr::str_sub(arr_time, start = -4, end = -3) |> parse_number(),
    arr_time_min = stringr::str_sub(arr_time, start = -2, end = -1) |> parse_number(),
    .keep = "used"
  )
```

```
# A tibble: 10 x 6
   arr_time sched_arr_time sched_arr_time_hour sched_arr_time_min arr_time_house
      <int>          <int>               <dbl>              <dbl>          <dbl>
 1     1902           1920                  19                 20             19
 2     1725           1759                  17                 59             17
 3     2259           2220                  22                 20             22
 4     1330           1409                  14                  9             13
 5       11             22                  NA                 22             NA
 6     2135           2210                  22                 10             21
 7     1405           1418                  14                 18             14
 8      919            908                   9                  8              9
 9     2102           2035                  20                 35             21
10     2125           2130                  21                 30             21
# i 1 more variable: arr_time_min <dbl>
```

```
time_hour <- function(x) {
  stringr::str_sub(x, start = -4, end = -3) |> parse_number()
}

time_min <- function(x) {
  stringr::str_sub(x, start = -2, end = -1) |> parse_number()
}
```

```
flights_small |>
  mutate(
    across(
      c(sched_arr_time, arr_time),
      list(hour = ~ time_hour(.), min = ~ time_min(.)),
      .names = "{.col}_{.fn}"
    )
  )
```

```
# A tibble: 10 x 23
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
```

```
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
 1  2013     1     4     1623           1625        -2     1902           1920
 2  2013     5     6     1533           1535        -2     1725           1759
 3  2013     1    17     2014           1935        39     2259           2220
 4  2013     8    26     1155           1200        -5     1330           1409
 5  2013     7    11     2120           2100        20       11             22
 6  2013    11    18     1901           1910        -9     2135           2210
 7  2013    11     4     1221           1226        -5     1405           1418
 8  2013    11    27      704            705        -1      919            908
 9  2013     7    21     1716           1716         0     2102           2035
10  2013     3    13     1930           1929         1     2125           2130
# i 15 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>, sched_arr_time_hour <dbl>,
#   sched_arr_time_min <dbl>, arr_time_hour <dbl>, arr_time_min <dbl>
```