HW4_statistical_tests

AUTHOR
Darrell Sonntag

PUBLISHED
February 8, 2024

Objectives:

- Read in many files, and manipulate them
- plot many files of the indoor vs. outdoor correlation for each home
- Conduct many linear regression fits (and store the output)
- Conduct t-test of the summary statistics

Clone the following public repository to your computer

https://github.com/darrell-sonntag/EvapCoolerUtahCounty

Read in the files stored in side the two SidePak folders in the data folder - /Data/SidePak_txt_1 - /Data/SidePak_txt_2

Spend some time in class going over this...

```
list_1.txt = list.files("../../EvapCoolerUtahCounty/Data/SidePak_txt_1", pattern = "*.txt", full.names = TRUE)
list_2.txt = list.files("../../EvapCoolerUtahCounty/Data/SidePak_txt_2",pattern = "*.txt", full.names = TRUE)
```

Write a function to read in the txt files - Use read_csv - Skip the first 30 rows - Assign the column names 'Date', 'Time','Aerosol'

```
read_SidePak_1 <- function(flnm) {
  read_csv(flnm,col_names=c("Date","Time","Aerosol"),skip=30) %>%
    mutate(date.time = mdy_hms(paste(Date,Time))) %>%
            mutate(Aerosol = as.numeric(Aerosol)) %>%
            mutate(filename = flnm)
    }
```

For loop way

```
SidePak_1 <- data.frame() # create empty data.frame
for (i in 1:length(list_1.txt)) {
  data.i <- read_SidePak_1(list_1.txt[i])

  SidePak_1 <- bind_rows(SidePak_1,data.i)
}
```

Here's the tidyverse way of handling it

https://r4ds.hadley.nz/iteration#reading-multiple-files

```
SidePak_list <- map(list_1.txt,read_SidePak_1)
## You could use lapply instead of map
## map is in the tidyverse version of lapply (PURR), it maps a function (read_SidePak) to a vector or a list (list.txt), and stores the o

length(SidePak_list)
```

```
[1] 79
```

```
SidePak_1 <- list_rbind(SidePak_list)
## list_rbind binds all the elements of a list into a single dataframe
```

The second method is a little faster, and a little easier to follow the code... but both work!

Now...read in the other files, create a function called read_SidePak_2

Note this file is tab delimited, so use the read_delim function (not read_csv)

Hint: The format of this file is different. You don't need to skip the first 30 lines.

Make it so that read_SidePak_2 files have the same column names as the read_SidePak1 files

```
read_SidePak_2 <- function(flnm) {
  read_delim(flnm, delim = "\t",col_names = c("ID","Date","Time","Aerosol"),skip=1) %>%
            select(-ID) %>%
            mutate(date.time = mdy_hms(paste(Date,Time))) %>%
            mutate(Aerosol = as.numeric(Aerosol)) %>%
            mutate(filename = flnm)
    }
```

Run your function here, using map, and list_rbind

```
SidePak_2 <- list_2.txt %>%
             map(read_SidePak_2) %>%
             list_rbind()
```

Create new dataframe called SidePak

- Bind SidePak_1 and SidePak_2 together
- Convert Aerosol from units of mg.m3 to ug.m3 (multiply by 1000)
- Add season = "Summer" if In June, July, August, or September
- Hint: use month(Date) to find the month
- Use str_split_i(), to return a subset of a character string from filename https://stringr.tidyverse.org/reference/str_split.html
- Remove the path from filename, and the .txt extension
- Split the filename by pattern = "/"
- The first element is the House.Number
- The second is the Visit
- The third is the Location
- Filter out any rows that have missing Time
- Create a new variable called round.date.time using Round_date(date.time, unit="minute")

```
SidePak <- SidePak_1 %>%
           bind_rows(SidePak_2) %>%
           mutate(Aerosol = Aerosol*1000 ) %>% ## change units of Aerosol                from mg/m3 to ug/m3
            filter(!is.na(Time)) %>%
            mutate(Date=mdy(Date)) %>%
            #mutate(Time = hms(Time)) %>%
            mutate(season = ifelse(month(Date) >= 6 & month(Date) <= 9,
                         "Summer", "Winter"))%>%
            mutate(round.date.time = round_date(date.time,unit="minute")) %>%
            mutate(filename = str_split_i(filename,pattern = "/",i=6)) %>%
            mutate(filename = str_split_i(filename,pattern = ".txt",i=1)) %>%
            mutate(House.Number = str_split_i(filename,pattern = "_", i=1)) %>%
            mutate(Visit = str_split_i(filename,pattern = "_",i=2)) %>%
            mutate(Location =str_split_i(filename,pattern = "_", i=3))
```

Read in the first sheet of the "Research Data Master List.xlsx" file stored in the data folder Assign it to a data.frame called ac.data Just select the House.Number and the "Type of Air Conditioner" create a new shorthand variable called ac.type where Central = AC and Evaporative =EC

```
ac.data <-  read_excel(path = "../../EvapCoolerUtahCounty/Data/Research Data Master List.xlsx", sheet='Housing Survey Answers') %>%
            select(House.Number,'Type of Air Conditioner') %>%
            mutate(ac.type = case_when(
                    `Type of Air Conditioner`== 'Central' ~'AC',
                    `Type of Air Conditioner` == 'Evaporative' ~ 'EC'))
```

Create a new df called SidePak.ac - Join the ac.data to the SidePak data using 'House.Number' - Create a new variable called House.Number.Visit that combines the House.Number and the Visit

```
SidePak.ac <- SidePak %>%
              left_join(ac.data,by='House.Number') %>%
              mutate(House.Number.Visit = paste(House.Number, Visit, sep = " "))
```

Create a vector called hv with all the unique House.Number.Visits

```
hv <- unique(SidePak.ac$House.Number.Visit)
```

Create a series of time-series plots of the data from each visit x-axis is time, use geom_line, color for location Put the name of the House.Number.Visit on the title of each graph

```
pdf("../figs/sidepakplots.pdf",onefile = TRUE)
for(i in 1:length(hv)){
  data.i<- filter(SidePak.ac,House.Number.Visit==hv[i])
  print(ggplot(data = data.i) +
          geom_line(aes(x = date.time, y = Aerosol, color=Location))+
          theme(axis.text.x = element_text(size = 8)) +
          labs(title=hv[i]), y = expression(paste("SidePak PM"[2.5]," ug/m"^3)))
}
```

```
Warning: Removed 1331 rows containing missing values or values outside the scale range
(`geom_line()`).

Warning: Removed 347 rows containing missing values or values outside the scale range
(`geom_line()`).
```

```
dev.off()
```

```
png
  2
```

Note there are many days with missing data for either indoor or outdoor measurements

Create a data.frame that summarizes SidePak.ac by House.Number.Visit and Location

- the sum of all the Aerosol measurements
- the sum of non-missing observations
- and the interval of time with valid measurements

```
SidePak.qa.sum <- SidePak.ac %>%
  group_by(House.Number.Visit, Location) %>%
  summarize(
    sum.Aerosol = sum(Aerosol,na.rm = T),
    n.Aerosol = sum(!is.na(Aerosol)),
    min.time = min(round.date.time),
    max.time = max(round.date.time),
    interval = max.time - min.time) %>%
  mutate(interval.hour = as.numeric(as.duration(interval), "hours"))
```

```
`summarise()` has grouped output by 'House.Number.Visit'. You can override
using the `.groups` argument.
```

Create a dataframe called SidePak.qa.sum.wide from SidePak.qa.sum

- Pivot SidePak.qa.sum wider to identify how valid paired Indoor and Outdoor visits we have
- Filter out any visits that have Aerosol measurements that sum to zero from either the indoor or outdoor measurements
- Filter out data without at least 4 hours for both Indoor and Outdoor measurements
- In addition, filter out the following House.Number.Visits
  - H09 V2 (Indoor and outdoor data are separate dates)
  - H16 V1 (In & Out files are the same data)
  - H17 V3 (Indoor SidePak is much larger than indoor. Outdoor SidePak ends prematurely)

```
SidePak.qa.sum.wide <- SidePak.qa.sum %>%
  select(House.Number.Visit, Location, sum.Aerosol, n.Aerosol, interval.hour)%>%
  pivot_wider(names_from = Location, values_from = c(sum.Aerosol,n.Aerosol,interval.hour)) %>%
  filter(sum.Aerosol_In > 0 & sum.Aerosol_Out > 0) %>% ## remove houses with zero data
```

```
  filter(interval.hour_In > 4 & interval.hour_Out > 4)  %>% ## remove data without at least 4 hours for both measurements
   filter(!(House.Number.Visit %in% c('H09 V2','H16 V1','H17 V3')))
```

Create a vector with a list of the valid home visits called hv.qa

```
hv.qa <- SidePak.qa.sum.wide$House.Number.Visit
```

Create a new table called SidePak.correlation from SidePak.ac - Create a wide version, where there are two columns for Aerosol, one for location In and another for location Out - Use pivot_wider - Hint: Remember to un-select Time, date.time, and filename (which are unique to either the indoor or outdoor fields)

names(SidePak.ac)

```
SidePak.correlation <- SidePak.ac %>%
                       #select(-c(Time,filename,date.time)) %>% ## you can un-select the columns you don't want
                       select(House.Number, Visit, Location, House.Number.Visit, ac.type, Date, round.date.time, season, Aerosol) %>% # O
                       pivot_wider(names_from = Location, values_from = Aerosol)
```

Based on the QA steps above, and inspection of the sidepak plots, we identified windows of time with problematic data. We removed stretches where their appeared to be sources of indoor air pollution

We created a new dataframe called SidePak.correlation.qa

Update the commented line below

```
SidePak.correlation.qa <- SidePak.correlation %>%
  # Add a filter statement here to remove the homes that are not on the hv.qa list
  filter(House.Number.Visit %in% hv.qa) %>% # Make sure the House.Number.Visit is on the list of qa'd visits
  filter(!is.na(In) & !is.na(Out)) %>% # Both In and out Data are recorded for the same minute
  filter(!(House.Number == 'H03' & Visit == 'V1' & between(round.date.time,ymd_hms('2022-07-27 19:54:00'),ymd_hms('2022-07-28 11:20:00'))
  filter(!(House.Number == 'H03' & Visit == 'V2' & between(round.date.time,ymd_hms('2022-12-09 08:19:00'),ymd_hms('2022-12-09 11:01:00'))
  filter(!(House.Number == 'H08' & Visit == 'V2' & between(round.date.time,ymd_hms('2022-09-09 10:15:00'),ymd_hms('2022-09-09 14:03:00'))
  filter(!(House.Number == 'H10' & Visit == 'V2' & between(round.date.time,ymd_hms('2022-12-01 07:45:00'),ymd_hms('2022-12-01 09:55:00'))
  filter(!(House.Number == 'H12' & Visit == 'V1' & between(round.date.time,ymd_hms('2022-08-12 10:30:00'),ymd_hms('2022-08-12 14:34:00'))
  filter(!(House.Number == 'H17' & Visit == 'V2' & between(round.date.time,ymd_hms('2023-01-28 04:06:00'),ymd_hms('2023-01-28 05:43:00'))
  filter(!(House.Number == 'H29' & Visit == 'V2' & between(round.date.time,ymd_hms('2023-08-21 20:10:00'),ymd_hms('2023-08-21 23:16:00'))
  filter(!(House.Number == 'H33' & Visit == 'V1' & between(round.date.time,ymd_hms('2023-09-01 13:02:00'),ymd_hms('2023-09-01 16:36:00'))
```

Create series of plots with the correlation - Out concentration on the x-axis - In concentration on the y-axis - You can use a for loop, and pdf() and dev.off() to make a single file with all the plots (like we did in class).

Or you can make many individual plots using ggsave(). Here's an example here, https://r4ds.hadley.nz/iteration#saving-plots

```
library(ggpmisc)
library(ggplot2)

pdf("../figs/sidepakcorrelation.pdf",onefile = TRUE)
for(i in 1:length(hv.qa)){
  data.i<- filter(SidePak.correlation.qa,House.Number.Visit==hv.qa[i])
  print(ggplot(data = data.i, aes(x = Out, y = In)) +
          geom_point() +
          stat_poly_line() + ## geom_smooth() would also work. I used stat_poly_line to display the equation
          #stat_poly_eq(aes(label = paste(after_stat(eq.label),
          #                               after_stat(rr.label), sep = "*\", \"*"))) +
          theme(axis.text.x = element_text(size = 8)) +
          labs(title=paste(hv.qa[i],data.i$ac.type[1]),sep = " " ))
  }
dev.off()
```

```
png
  2
```

Now, create a dataframe that stores the linear model coefficients for each house.visit.

The data.frame should include the following variables:

- Visit = character()

- ac.type=character()
- Date = POSIXct() Use the starting time
- intercept =numeric()
- intercept.lower =numeric() lower 95% CI
- intercept.upper = numeric() upper 95% CI
- slope = numeric()
- slope.lower = numeric()
- slope.upper = numeric()
- p.value = numeric()
- r.squared = numeric()

There are multiple ways of doing this.

I provided two examples of doing this (in part to show the benefit of the Tidyverse methods)

```
1.  Loop + BaseR (What I did for my paper)

2.  Create a function and map it to the dataset using tidyverse functions (Recommended!)

3.  Bonus point +1 for someone who does this a more susinct way than I did with version 2.
```

Example 1. Here's the loop way, using mostly Base R, methods

Start with creating an empty data.frame

```
lm.coefficients <- data.frame(House.Number = character(),
                     Visit = character(),
                     ac.type=character(),
                     Date = POSIXct(),
                     intercept = numeric(),
                     intercept.lower =numeric(),
                     intercept.upper = numeric(),
                     slope = numeric(),
                     slope.lower = numeric(),
                     slope.upper = numeric(),
                     p.value = numeric(),
                     r.squared = numeric()
                      )
```

Conduct a loop, that loops through the vector of hv.qa

- Filters the data to just the data from that House.Number.Visit
- Fits a linear model
-
- Hint: Assign the linear fit to be linear object (call it lm_object)
  - Summary(lm_object) returns a list
    - You can use summary(lm_object)[['coefficients']] to return a matrix with the coefficients and p-values
    - You can access the values by giving the [rowname, column name]
    - For example,summary(lm_object)[['coefficients']]['(Intercept)','Estimate'] (gives you the value on the Intercept row, and the Estimate Column)
  - Just report the p-value with the slope term
  - confint(lm_object,.95) returns the 95% confidence intervals of the model parameters

```
for (i in 1: length(hv.qa)){
            data.i = filter(SidePak.correlation.qa,House.Number.Visit ==hv.qa[i])
            lm.i <- lm(In~Out,data.i,na.action = na.omit)
            lm.coefficients[i,'House.Number'] = data.i$House.Number[1]
            lm.coefficients[i,'Visit'] = data.i$Visit[1]
            lm.coefficients[i,'ac.type'] = data.i$ac.type[1]
            lm.coefficients[i,'Date'] = min(data.i$round.date.time)
            lm.coefficients[i,'intercept'] = summary(lm.i)[['coefficients']]['(Intercept)','Estimate']
            lm.coefficients[i,'intercept.lower'] = confint(lm.i,level = .95)['(Intercept)','2.5 %']
            lm.coefficients[i,'intercept.upper'] = confint(lm.i,level = .95)['(Intercept)','97.5 %']
            lm.coefficients[i,'slope'] = summary(lm.i)[['coefficients']]['Out','Estimate']
            lm.coefficients[i,'slope.lower'] = confint(lm.i,2,level=.95)[1]
            lm.coefficients[i,'slope.upper'] = confint(lm.i,2,level=.95)[2]
            lm.coefficients[i,'p.value'] = summary(lm.i)[['coefficients']]['Out','Pr(>|t|)']
```

```
        lm.coefficients[i,'r.squared']= summary(lm.i)$r.squared

}
```

2. Here's a function, map, tidyverse function, way (Recommended!)

Create a lm_coeff function that fits a linear model,

I used the get_regression_table from the 'moderndive' package to extract out the regression coefficients. The get_regression_table uses some very useful tidyverse functions, including tidy and clean_names. Take a look here:

https://moderndive.com/5-regression.html#underthehood

Instead of embedding the filtering within the function (which is also possible), mapped my functions to a list of dataframes that are split up by home.visits. I used the group_split() tidyverse function, to split up the dataframe frame into a list (similar to the baseR split() function). Then I used map() (similar to lapply() to each of the data.frames in the list. Here's is an example of using map here:

https://purrr.tidyverse.org/reference/map.html

Info on the group_split here. https://dplyr.tidyverse.org/reference/group_split.html

You could also use the base R version of split(), and then use lapply()

Map() returns a list of output. I then used list_rbind() to bind all the list elements of the back together into a dataframe/tibble).

```
lm_coeff <- function(data){
        lm.i <- lm(In~Out,data,na.action = na.omit)

        output <- get_regression_table(lm.i) %>%
                mutate(R2 = round(summary(lm.i)$r.squared,digits=3)) %>%
                ## add identifiers
                mutate(House.Number = data$House.Number[1]) %>%
                mutate(Visit = data$Visit[1]) %>%
                mutate(ac.type = data$ac.type[1]) %>%
                mutate(Date = data$round.date.time[1]) %>%
                mutate(season = data$season[1])

        return(output)
}


lm.coefficients.2 <- SidePak.correlation.qa %>%
                group_by(House.Number.Visit) %>%
                group_split() %>%
                map(function(df) lm_coeff(data = df)) %>%
                list_rbind() %>%
                mutate(term = ifelse(term=='Out','Slope',term)) %>%
                select(9:13,1:8) ## rearrange the columns
```

Once you understand the tidyverse functions, the code is much more succinct using the tidyverse method!

Now print the lm results into a table

You can use kable or Or Hayden introduced me to tinytable https://vincentarelbundock.github.io/tinytable/

I'm trying tinytable() (since I have never really figured out kable in the library knitr) And tinytable seems easier

```
library(tinytable)
```

```
Warning: package 'tinytable' was built under R version 4.4.2
```

```
tt(lm.coefficients.2)
```

House.Number	Visit	ac.type	Date	season	term	estimate	std_error	statistic	p_value	lov
H02	V2	AC	2022-11-21 19:06:00	Winter	intercept	1.808	0.030	60.954	0.000	1.7

House.Number	Visit	ac.type	Date	season	term	estimate	std_error	statistic	p_value	lov
H02	V2	AC	2022-11-21 19:06:00	Winter	Slope	-0.017	0.003	-5.703	0.000	-0.(
H02	V3	AC	2023-08-15 18:13:00	Summer	intercept	3.377	0.134	25.254	0.000	3.1
H02	V3	AC	2023-08-15 18:13:00	Summer	Slope	-0.017	0.022	-0.773	0.440	-0.(
H03	V1	AC	2022-07-27 15:36:00	Summer	intercept	1.893	0.089	21.248	0.000	1.7
H03	V1	AC	2022-07-27 15:36:00	Summer	Slope	0.031	0.003	9.104	0.000	0.0
H03	V2	AC	2022-12-08 18:37:00	Winter	intercept	0.383	0.065	5.846	0.000	0.2
H03	V2	AC	2022-12-08 18:37:00	Winter	Slope	0.094	0.020	4.716	0.000	0.0
H03	V3	AC	2023-08-03 13:21:00	Summer	intercept	5.577	0.095	58.922	0.000	5.3
H03	V3	AC	2023-08-03 13:21:00	Summer	Slope	0.139	0.008	16.571	0.000	0.1
H04	V1	AC	2022-07-28 16:41:00	Summer	intercept	2.716	0.091	29.780	0.000	2.5
H04	V1	AC	2022-07-28 16:41:00	Summer	Slope	0.083	0.007	11.647	0.000	0.0
H04	V2	AC	2023-01-25 16:41:00	Winter	intercept	1.288	0.075	17.117	0.000	1.1
H04	V2	AC	2023-01-25 16:41:00	Winter	Slope	-0.009	0.002	-3.665	0.000	-0.(
H05	V2	AC	2023-02-02 17:57:00	Winter	intercept	2.850	0.243	11.706	0.000	2.3
H05	V2	AC	2023-02-02 17:57:00	Winter	Slope	0.108	0.005	21.631	0.000	0.0
H05	V3	AC	2023-08-21 17:56:00	Summer	intercept	6.227	0.114	54.425	0.000	6.0

House.Number	Visit	ac.type	Date	season	term	estimate	std_error	statistic	p_value	low
H05	V3	AC	2023-08-21 17:56:00	Summer	Slope	0.000	0.000	-0.516	0.606	-0.(
H06	V1	AC	2022-08-01 17:34:00	Summer	intercept	3.598	0.078	46.109	0.000	3.4
H06	V1	AC	2022-08-01 17:34:00	Summer	Slope	-0.015	0.009	-1.628	0.104	-0.(
H06	V2	AC	2022-11-17 10:35:00	Winter	intercept	6.065	0.034	180.565	0.000	5.9
H06	V2	AC	2022-11-17 10:35:00	Winter	Slope	-0.028	0.005	-5.783	0.000	-0.(
H07	V1	AC	2022-08-03 11:53:00	Summer	intercept	10.563	0.239	44.218	0.000	10.
H07	V1	AC	2022-08-03 11:53:00	Summer	Slope	-0.105	0.010	-10.800	0.000	-0.
H07	V2	AC	2023-03-09 19:19:00	Winter	intercept	3.065	0.036	85.462	0.000	2.9
H07	V2	AC	2023-03-09 19:19:00	Winter	Slope	0.006	0.022	0.272	0.786	-0.(
H08	V1	AC	2022-08-09 17:21:00	Summer	intercept	14.802	0.710	20.835	0.000	13.
H08	V1	AC	2022-08-09 17:21:00	Summer	Slope	-0.091	0.056	-1.618	0.107	-0.:
H08	V2	AC	2022-09-08 17:25:00	Summer	intercept	7.027	0.878	8.004	0.000	5.3
H08	V2	AC	2022-09-08 17:25:00	Summer	Slope	0.295	0.025	11.729	0.000	0.2
H08	V3	AC	2023-04-13 17:24:00	Winter	intercept	2.470	0.141	17.536	0.000	2.1
H08	V3	AC	2023-04-13 17:24:00	Winter	Slope	0.116	0.025	4.717	0.000	0.0
H09	V1	AC	2022-08-09 16:44:00	Summer	intercept	4.706	0.049	95.543	0.000	4.6

House.Number	Visit	ac.type	Date	season	term	estimate	std_error	statistic	p_value	lov
H09	V1	AC	2022-08-09 16:44:00	Summer	Slope	0.027	0.009	3.150	0.002	0.0
H10	V2	AC	2022-11-30 17:58:00	Winter	intercept	1.342	0.042	31.836	0.000	1.2
H10	V2	AC	2022-11-30 17:58:00	Winter	Slope	0.023	0.033	0.689	0.491	-0.(
H10	V3	AC	2023-08-03 18:02:00	Summer	intercept	4.431	0.089	49.671	0.000	4.2
H10	V3	AC	2023-08-03 18:02:00	Summer	Slope	0.061	0.015	4.159	0.000	0.0
H11	V1	AC	2022-08-16 18:59:00	Summer	intercept	3.655	0.109	33.431	0.000	3.4
H11	V1	AC	2022-08-16 18:59:00	Summer	Slope	0.084	0.005	16.580	0.000	0.0
H11	V2	AC	2022-11-16 18:09:00	Winter	intercept	7.585	0.195	38.926	0.000	7.2
H11	V2	AC	2022-11-16 18:09:00	Winter	Slope	0.242	0.022	10.817	0.000	0.1
H12	V1	AC	2022-08-11 18:30:00	Summer	intercept	1.910	0.196	9.761	0.000	1.5
H12	V1	AC	2022-08-11 18:30:00	Summer	Slope	0.769	0.034	22.499	0.000	0.7
H12	V2	AC	2023-03-27 18:03:00	Winter	intercept	5.659	0.239	23.660	0.000	5.1
H12	V2	AC	2023-03-27 18:03:00	Winter	Slope	-0.056	0.013	-4.279	0.000	-0.(
H13	V1	AC	2022-08-10 17:51:00	Summer	intercept	4.393	0.105	41.708	0.000	4.1
H13	V1	AC	2022-08-10 17:51:00	Summer	Slope	0.064	0.013	5.083	0.000	0.0
H13	V2	AC	2023-04-04 17:32:00	Winter	intercept	3.595	0.071	50.754	0.000	3.4

House.Number	Visit	ac.type	Date	season	term	estimate	std_error	statistic	p_value	low
H13	V2	AC	2023-04-04 17:32:00	Winter	Slope	-0.055	0.007	-7.421	0.000	-0.
H14	V1	AC	2022-08-15 15:39:00	Summer	intercept	5.903	0.195	30.325	0.000	5.5
H14	V1	AC	2022-08-15 15:39:00	Summer	Slope	-0.077	0.022	-3.506	0.000	-0.
H14	V2	AC	2022-09-09 15:36:00	Summer	intercept	-6.591	0.520	-12.679	0.000	-7.
H14	V2	AC	2022-09-09 15:36:00	Summer	Slope	0.273	0.006	44.014	0.000	0.2
H15	V2	AC	2023-08-21 17:29:00	Summer	intercept	6.261	0.045	139.155	0.000	6.1
H15	V2	AC	2023-08-21 17:29:00	Summer	Slope	0.223	0.034	6.478	0.000	0.1
H16	V2	AC	2022-09-09 16:20:00	Summer	intercept	-16.408	0.765	-21.438	0.000	-17
H16	V2	AC	2022-09-09 16:20:00	Summer	Slope	0.539	0.009	61.130	0.000	0.5
H16	V3	AC	2023-03-30 17:47:00	Winter	intercept	1.872	0.103	18.146	0.000	1.6
H16	V3	AC	2023-03-30 17:47:00	Winter	Slope	0.346	0.059	5.858	0.000	0.2
H16	V4	AC	2023-08-14 17:22:00	Summer	intercept	3.852	0.148	26.018	0.000	3.5
H16	V4	AC	2023-08-14 17:22:00	Summer	Slope	0.020	0.010	1.933	0.053	0.0
H17	V2	EC	2023-01-27 16:21:00	Winter	intercept	3.875	0.270	14.356	0.000	3.3
H17	V2	EC	2023-01-27 16:21:00	Winter	Slope	0.026	0.014	1.837	0.067	-0.
H18	V1	EC	2022-09-01 19:37:00	Summer	intercept	4.619	0.084	54.863	0.000	4.4

House.Number	Visit	ac.type	Date	season	term	estimate	std_error	statistic	p_value	lov
H18	V1	EC	2022-09-01 19:37:00	Summer	Slope	0.268	0.005	48.797	0.000	0.2
H19	V1	EC	2022-09-02 17:48:00	Summer	intercept	5.233	0.416	12.587	0.000	4.4
H19	V1	EC	2022-09-02 17:48:00	Summer	Slope	0.172	0.035	4.911	0.000	0.1
H19	V2	EC	2023-02-10 17:32:00	Winter	intercept	2.211	0.041	53.386	0.000	2.1
H19	V2	EC	2023-02-10 17:32:00	Winter	Slope	0.045	0.005	9.338	0.000	0.0
H19	V3	EC	2023-08-31 18:51:00	Summer	intercept	0.906	0.091	9.935	0.000	0.7
H19	V3	EC	2023-08-31 18:51:00	Summer	Slope	0.096	0.016	5.913	0.000	0.0
H20	V1	EC	2022-09-08 19:04:00	Summer	intercept	6.082	0.472	12.899	0.000	5.1
H20	V1	EC	2022-09-08 19:04:00	Summer	Slope	0.817	0.012	65.683	0.000	0.7
H21	V1	EC	2022-09-08 19:44:00	Summer	intercept	-4.110	0.559	-7.357	0.000	-5.:
H21	V1	EC	2022-09-08 19:44:00	Summer	Slope	1.013	0.015	67.376	0.000	0.9
H21	V2	EC	2022-12-08 17:36:00	Winter	intercept	5.325	0.074	71.962	0.000	5.1
H21	V2	EC	2022-12-08 17:36:00	Winter	Slope	0.025	0.003	8.263	0.000	0.0
H22	V1	EC	2022-09-12 17:35:00	Summer	intercept	0.591	0.170	3.469	0.001	0.2
H22	V1	EC	2022-09-12 17:35:00	Summer	Slope	1.012	0.014	70.355	0.000	0.9
H22	V2	EC	2023-03-03 17:55:00	Winter	intercept	7.567	0.229	33.103	0.000	7.1

House.Number	Visit	ac.type	Date	season	term	estimate	std_error	statistic	p_value	lov
H22	V2	EC	2023-03-03 17:55:00	Winter	Slope	-0.081	0.024	-3.357	0.001	-0.
H23	V1	EC	2022-09-12 18:05:00	Summer	intercept	-0.692	0.086	-8.055	0.000	-0.8
H23	V1	EC	2022-09-12 18:05:00	Summer	Slope	1.030	0.009	114.720	0.000	1.0
H23	V2	EC	2023-04-06 17:50:00	Winter	intercept	1.721	0.057	30.049	0.000	1.6
H23	V2	EC	2023-04-06 17:50:00	Winter	Slope	-0.051	0.011	-4.404	0.000	-0.0
H24	V1	EC	2023-07-13 18:06:00	Summer	intercept	6.203	0.167	37.099	0.000	5.8
H24	V1	EC	2023-07-13 18:06:00	Summer	Slope	0.237	0.019	12.550	0.000	0.2
H25	V1	EC	2023-07-13 19:13:00	Summer	intercept	9.641	0.277	34.770	0.000	9.0
H25	V1	EC	2023-07-13 19:13:00	Summer	Slope	-0.184	0.026	-6.997	0.000	-0.2
H26	V1	EC	2023-07-20 13:40:00	Summer	intercept	2.927	0.194	15.128	0.000	2.5
H26	V1	EC	2023-07-20 13:40:00	Summer	Slope	0.489	0.019	25.206	0.000	0.4
H26	V2	EC	2023-08-28 18:01:00	Summer	intercept	5.435	0.133	40.940	0.000	5.1
H26	V2	EC	2023-08-28 18:01:00	Summer	Slope	0.359	0.020	17.685	0.000	0.3
H28	V1	EC	2023-07-27 14:47:00	Summer	intercept	6.129	0.123	49.641	0.000	5.8
H28	V1	EC	2023-07-27 14:47:00	Summer	Slope	0.067	0.011	6.244	0.000	0.0
H29	V2	EC	2023-08-21 18:21:00	Summer	intercept	3.402	0.086	39.406	0.000	3.2

House.Number	Visit	ac.type	Date	season	term	estimate	std_error	statistic	p_value	lov
H29	V2	EC	2023-08-21 18:21:00	Summer	Slope	0.196	0.020	9.917	0.000	0.1
H30	V1	EC	2023-08-10 17:31:00	Summer	intercept	3.264	0.103	31.791	0.000	3.0
H30	V1	EC	2023-08-10 17:31:00	Summer	Slope	0.200	0.012	16.380	0.000	0.1
H31	V1	AC	2023-08-24 17:54:00	Summer	intercept	2.319	0.409	5.673	0.000	1.5
H31	V1	AC	2023-08-24 17:54:00	Summer	Slope	0.381	0.030	12.797	0.000	0.3
H32	V1	EC	2023-08-28 18:39:00	Summer	intercept	6.283	0.158	39.806	0.000	5.9
H32	V1	EC	2023-08-28 18:39:00	Summer	Slope	0.062	0.020	3.110	0.002	0.0
H33	V1	AC	2023-08-31 18:00:00	Summer	intercept	2.563	0.075	34.301	0.000	2.4
H33	V1	AC	2023-08-31 18:00:00	Summer	Slope	-0.138	0.013	-10.888	0.000	-0.

Graph the intercept, slope, and R2 from each home visit using box plots, with separate box plots for the AC and EC homes.

First, I will create a data.frame called lm.coefficients.wide. I will first just select the id columns, the intercept and slope estimates, and the R2 Then I will pivot the intercept and slope wider to be their own columns

```
lm.coefficients.wide <- lm.coefficients.2 %>%
  select(House.Number, Visit, ac.type, Date, season, term, estimate, R2) %>%
  pivot_wider(names_from = term, values_from = estimate)
```

Now make a table called lm.coefficients.long

Then I will pivot the values of intercept, slope, and R2 in a column called 'value' And the name of the statistic (intercept, Slope, R2) in a column called 'statistic'

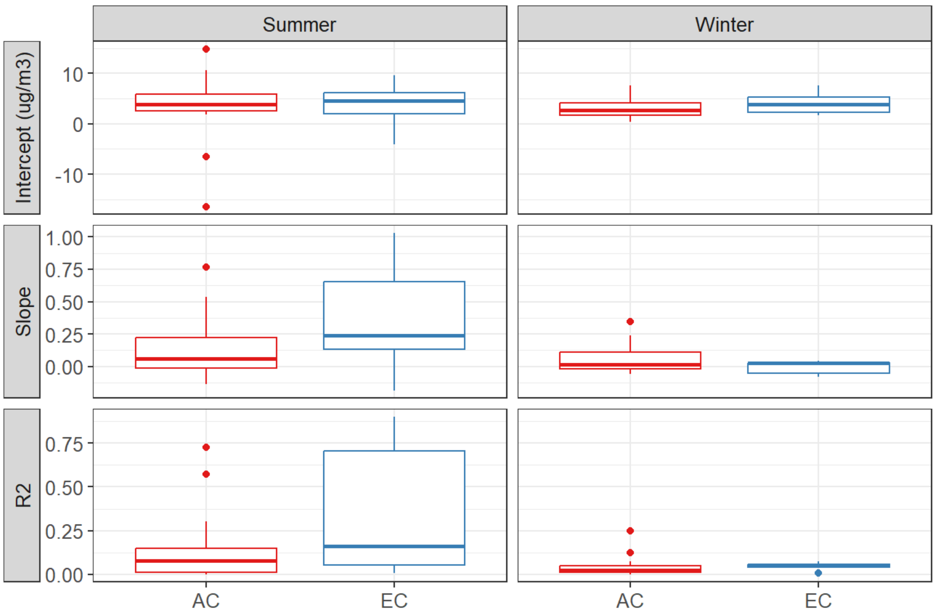
Keep the descriptors (House.Number,Visit, and ac.type)

```
lm.coefficients.long <- lm.coefficients.wide %>%
  pivot_longer(cols = c(intercept, Slope, R2), names_to ="statistic",
               values_to = "value") %>%
  mutate(statistic = factor(statistic,levels=c('intercept','Slope','R2'),ordered=T))
```

Re-create the following graph: - Graph the Intercept, Slope, and R2 from each home visit in a box plot organized by season and AC type - Use geom_boxplot - use facet_grid to create the following graph - Save your picture to your /figs/folder

```
ggplot(lm.coefficients.long,aes(x=ac.type,y=value,color=ac.type)) +
  geom_boxplot()+
```

```
  scale_color_brewer(palette='Set1')+
  facet_grid(statistic~season,scales='free_y',switch='y',
             labeller = as_labeller(c(Summer = 'Summer', Winter='Winter',intercept = "Intercept (ug/m3)",Slope="Slope",R2 = "R2"))) +
  labs(y = "", title='',x='')+
  theme_bw()+
  expand_limits(y = 0)+
  theme(axis.text.x = element_text(size=10), axis.text.y = element_text(size=10),
        axis.title = element_text(size = 12),plot.title = element_text(size = 18),
        strip.text = element_text(size=10),
        legend.text = element_text(size = 14),legend.position="none",
        strip.placement='outside')
```




```
ggsave("../figs/Boxplot.regression.comp.png", width=6, height=6, units="in")
```

Statistical Testing

Conduct a t-test to see if the mean statistic by season is statistically different by air conditioner type.

?t.test is the base r method ?t_test is a tidier version of t.test, but allows piping (library(infer))

First do an example, with just the slope coefficients in the summer

```
## first create a subset of just the slope values and in the summer

lm.coeff_slope_summer <- filter(lm.coefficients.long, statistic=='Slope',season=='Summer')

## Conduct a t-test to see that it works

t.test.ac <- t_test(lm.coeff_slope_summer, value~ac.type, order = c("AC", "EC"))
```

Then, create a function that conducts t_tests, and also stores the 'statistic' and 'season'

```
t_test_ac <- function(data){
              t.data  <- data %>%
              t_test(value~ac.type, order = c("AC", "EC")) %>%
              mutate(statistic = data$statistic[1]) %>%
              mutate(season = data$season[1]) %>%
              relocate(season,1)
```

```
            return(t.data)
}
```

Then, split up your lm.coefficients.long - use split() or group_split() - map your function to the split groups - list_rbind them together

```
t.test.ac <- lm.coefficients.long %>%
                group_by(statistic,season) %>%
                group_split() %>%
                map(function(df) t_test_ac(data = df)) %>%
                list_rbind()
```

Then display the results using a table

```
tt(t.test.ac)
```

season	statistic	t_df	p_value	alternative	estimate	lower_ci	upper_ci
Summer	intercept	32.669100	0.86959948	two.sided	-0.262104762	-3.48616277	2.96195324
Winter	intercept	7.001779	0.45817426	two.sided	-0.974633333	-3.91024568	1.96097902
Summer	Slope	20.558656	0.02752548	two.sided	-0.267695238	-0.50266480	-0.03272567
Winter	Slope	14.776767	0.12311353	two.sided	0.071366667	-0.02177796	0.16451129
Summer	R2	20.057776	0.03834966	two.sided	-0.220038095	-0.42704556	-0.01303063
Winter	R2	14.922276	0.84290630	two.sided	0.004766667	-0.04563774	0.05517107

Extra credit (+1)

Use Repeat the same box plot, except now, also plot the mean values for each statistic

Also use geom_signif from the library(ggsignif) to plot t-test results

Note:
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4111019/#:~:text=Through%20the%201960s%20it%20was,used%20to%20indicate%20P%20%3C%200.001.

```
## calculate means and 95% CI
lm.coefficient.means <- lm.coefficients.long %>%
  group_by(statistic,ac.type,season) %>%
  summarize(mean = mean(value,na.rm=T)) %>%
  mutate(statistic = factor(statistic,levels=c('intercept','Slope','R2'),ordered=T))
```

```
`summarise()` has grouped output by 'statistic', 'ac.type'. You can override
using the `.groups` argument.
```

```
library(ggsignif)
```

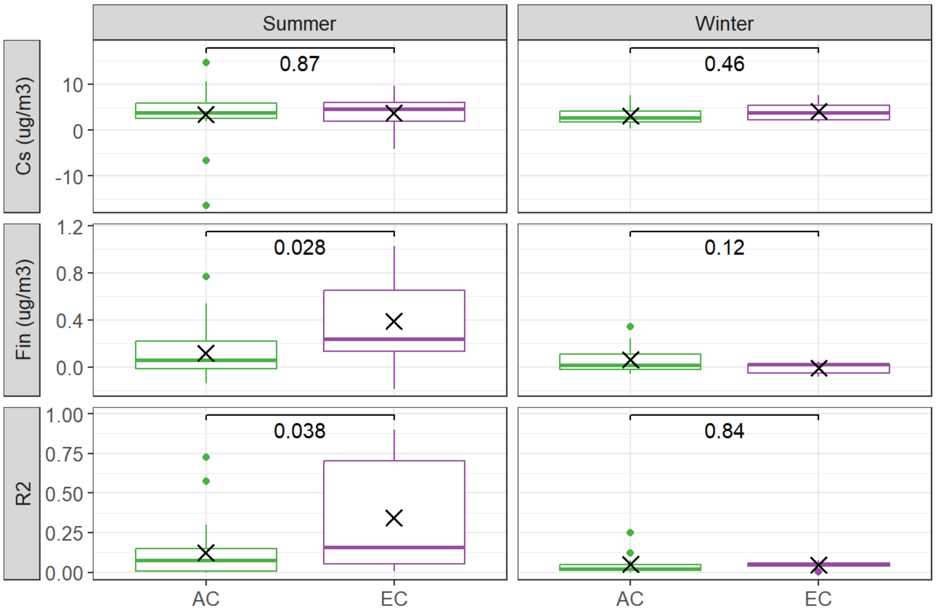
```
Warning: package 'ggsignif' was built under R version 4.4.2
```

```
library(RColorBrewer)

own.colors <- brewer.pal(n = 9, name = "Set1")[c(3:9)]

ggplot(lm.coefficients.long,aes(x=ac.type,y=value,color=ac.type)) +
  geom_boxplot()+
  geom_point(data=lm.coefficient.means,aes(x=ac.type,y=mean),color='black',shape=4,size=3,stroke=1)+
  geom_signif(comparisons=list(c('AC','EC')),
              map_signif_level=FALSE,test='t.test',
              color='black',vjust=1.5,margin_top=0.1)+
  scale_color_manual(values = own.colors,drop=F)+
  facet_grid(statistic~season,scales='free_y',switch='y',
             labeller = as_labeller(c(Summer = 'Summer', Winter='Winter',intercept = "Cs (ug/m3)",Slope="Fin (ug/m3)",R2 = "R2"))) +
```

```
  labs(y = "", title='',x='')+
  theme_bw()+
  expand_limits(y = 0)+
  theme(axis.text.x = element_text(size=10), axis.text.y = element_text(size=10),
        axis.title = element_text(size = 12),plot.title = element_text(size = 18),
        strip.text = element_text(size=10),
        legend.text = element_text(size = 14),legend.position="none",
        strip.placement='outside')
```




```
ggsave("../figs/Fancy.Boxplot.regression.comp.png", width=6, height=6, units="in")
```

Calculate the average SidePak concentrations and export it

Use SidePak.correlation.qa

names(SidePak.correlation.qa)

```
SidePak.visit.sum <- SidePak.correlation.qa %>%
                      group_by(House.Number,Visit,House.Number.Visit,ac.type,season) %>%
                      summarize(In_mean = mean(In,na.rm=T),Out_mean=mean(Out,na.rm=T),
                                Date = min(Date),Start_time = min(round.date.time))
```

```
`summarise()` has grouped output by 'House.Number', 'Visit',
'House.Number.Visit', 'ac.type'. You can override using the `.groups` argument.
```

Save the SidePak.visit.sum

```
write_csv(SidePak.visit.sum,"../../CE594R_data_science_R/data/SidePak.visit.summary.csv")
```

Move it back to a long format

```
SidePak.minute.qa <- SidePak.correlation.qa %>%
                    pivot_longer(cols = c('In','Out'), names_to='Location', values_to = 'Aerosol_ug_m3')
```

Export the long data

```
write_csv(SidePak.minute.qa,"../../CE594R_data_science_R/data/SidePak.minute.qa.csv")
```

hi people

i have the best dad in the whole entire universe

i love basketball