ps2.Rmd

Sijun Zhang UMid: 89934761

2019/10/10

Problem Set 2

Problem 1

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
------------------------------------------- tidyverse 1.2.1 --
```

```
## √ ggplot2 3.2.1     √ purrr   0.3.2
## √ tibble  2.1.3     √ dplyr   0.8.3
## √ tidyr   1.0.0     √ stringr 1.4.0
## √ readr   1.3.1     √ forcats 0.4.0
```

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

```
# read in the  data: ------------------------------------------------------
## This data will be used in the question.
url_base <- 'https://www.eia.gov/consumption/residential/data/2015/csv/'

recs_file <- './recs2015_public_v4.csv'
if ( !file.exists(recs_file) ) {
  recs_url <- sprintf('%s/recs2015_public_v4.csv', url_base)
  recs <- readr::read_delim(recs_url, delim = ',')
  readr::write_delim(recs, path = recs_file, delim = ',')
} else {
  recs <- readr::read_delim(recs_file, delim = ',')
}
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   METROMICRO = col_character(),
##   UATYP10 = col_character(),
##   CLIMATE_REGION_PUB = col_character(),
##   IECC_CLIMATE_PUB = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

First we read data from the csv document.

National Average Home Temperature at Night in Winter

```
# Input: recc2015_public_v4.csv
# Output: the national average home temperature at night in winter,
# among homes that use space heating
winter_tmp_heated <- recs %>%
  transmute(DOEID, winter_tmp = TEMPNITE, space_heated = HEATHOME, weight = NWEIGHT) %>%
  filter((space_heated == 1) ) %>%
  replace_na(list(weight=0))

# Convert weights to long: ------------------------------------------------
weights_long <- recs %>%
  filter(HEATHOME == 1) %>%
  select(DOEID, BRRWT1:BRRWT96) %>%
  gather(key = 'repl', value = 'w', BRRWT1:BRRWT96)

# Join home type to weights: ----------------------------------------------
winter_tmp_heated_rep <- weights_long %>%
  left_join(winter_tmp_heated %>% mutate(DOEID = as.integer(DOEID)), by='DOEID')

# Check nothing is lost
if( nrow(weights_long) != nrow(winter_tmp_heated_rep) ) {
  stop("DOEID mismatch!")
}

winter_tmp_heated_avg_rep <- winter_tmp_heated_rep %>%
  group_by(repl) %>%
  summarize(winter_tmp_avg_r = sum(winter_tmp * w) / sum(w))

winter_tmp_heated_avg <- winter_tmp_heated %>%
  group_by(space_heated) %>%
  summarize(winter_tmp_avg = sum(winter_tmp * weight) / sum(weight))

# in this problem we all use \epsilon = 0.5
winter_tmp_heated_avg_num <- as.numeric(winter_tmp_heated_avg[,2])
winter_tmp_heated_stderr <- 2 * sqrt(mean( {sapply(winter_tmp_heated_avg_rep[,2], as.numeric) - winter_tm
p_heated_avg_num}^2 ))
winter_tmp_heated_lwr <- winter_tmp_heated_avg_num - qnorm(0.975) * winter_tmp_heated_stderr
winter_tmp_heated_upr <- winter_tmp_heated_avg_num + qnorm(0.975) * winter_tmp_heated_stderr

sprintf(paste0('The national average home temperature at night in winter is %f,',
        ' among homes using space heating, ', 'and the upperbound is %f, and the ',
        'lower bound is %f'),
        winter_tmp_heated_avg_num, winter_tmp_heated_upr, winter_tmp_heated_lwr )
```

```
## [1] "The national average home temperature at night in winter is 68.105179, among homes using space he
ating, and the upperbound is 68.283863, and the lower bound is 67.926495"
```

The Proportion of Homes using Each Level of “Main Space Heating Fuel”

Decoding the Census Division and Census (2010) Urban Type

```
decode_fuel = function(x) {
  if(!is.numeric(x)) stop('decode_fuel expects numeric input indexed from 1!')

  re <- as.character( factor(x, levels = c(1, 2, 3, 5, 7, 21),
                             labels = c('Natural_gas',
                                        'Propane',
                                        'Fuel_oil_kerosene',
                                        'Electricity',
                                        'Wood',
                                        'Some_other_fuel')
                             ))

  return(re)
}

decode_all_fuel = function(x) {
  return(sapply(x,decode_fuel))
}

decode_division = function(x) {
  if(!is.numeric(x)) stop('decode_divsion expects numeric input indexed from 1!')
  y <- x
  re <- switch (y,
                'New England',
                'Middle Atlantic',
                'East North Central',
                'West North Central',
                'South Atlantic',
                'East South Central',
                'West South Central',
                'Mountain North',
                'Mountain South',
                'Pacific')
  return(re)
}

decode_all_division = function(x) {
  return(sapply(x,decode_division))
}

decode_urban = function(x) {
  re <- as.character( factor(x, levels = c('U', 'C', 'R'),
                             labels = c('Urban Area', 'Urban Cluster', 'Rural')
                             ))
  return(re)
}

decode_all_urban = function(x) {
  return(sapply(x,decode_urban))
}
```

The above code tranform the numeric value of urban type and division information to their exact meanings.

Proportion Table

```
recs_fuel<- recs %>%
  transmute(DOEID, fuel_type = FUELHEAT, divis = DIVISION, urban_type = UATYP10,
            space_heated = HEATHOME, weight = NWEIGHT) %>%
  filter((fuel_type > 0) & (space_heated == 1) ) %>%
  mutate(fuel_type = decode_all_fuel(fuel_type),urban_type = decode_all_urban(urban_type),
         divis = decode_all_division(divis))

recs_fuel_rep <- weights_long %>%
  left_join(recs_fuel %>% mutate( DOEID=as.integer(DOEID) ) , by='DOEID' )

# Check nothing is lost
if( nrow(weights_long) != nrow(recs_fuel_rep) ) {
  stop("DOEID mismatch!")
}

recs_fuel_prop_rep <- recs_fuel_rep %>%
  group_by(divis,urban_type, fuel_type, repl) %>%
```

```
  summarize(sum_fuel = sum(w)) %>%
  tidyr::pivot_wider(names_from = fuel_type,
                     values_from = sum_fuel, values_fill = list(sum_fuel = 0)) %>%
  mutate(Total =Natural_gas + Propane + Fuel_oil_kerosene + Electricity + Wood + Some_other_fuel,
         Natural_gas_r = 100 * Natural_gas / Total,
         Propane_r = 100 * Propane / Total,
         Fuel_oil_kerosene_r = 100 * Fuel_oil_kerosene / Total,
         Electricity_r = 100 * Electricity / Total,
         Wood_r = 100 * Wood / Total,
         Some_other_fuel_r = 100 * Some_other_fuel / Total) %>%
  select(divis, urban_type, repl, Natural_gas_r, Propane_r, Fuel_oil_kerosene_r,
         Electricity_r, Wood_r,Some_other_fuel_r)

recs_fuel_prop <- recs_fuel %>%
  group_by(divis,urban_type, fuel_type) %>%
  summarize(sum_fuel = sum(weight)) %>%
  tidyr::pivot_wider(names_from = fuel_type,
                     values_from = sum_fuel, values_fill = list(sum_fuel = 0)) %>%
  mutate(Total =Natural_gas + Propane + Fuel_oil_kerosene + Electricity + Wood + Some_other_fuel,
         Natural_gas = 100 * Natural_gas / Total,
         Propane = 100 * Propane / Total,
         Fuel_oil_kerosene = 100 * Fuel_oil_kerosene / Total,
         Electricity = 100 * Electricity / Total,
         Wood = 100 * Wood / Total,
         Some_other_fuel = 100 * Some_other_fuel / Total) %>%
  select(-Total)

recs_fuel_prop_rep <- recs_fuel_prop_rep %>%
  left_join(recs_fuel_prop, by = c ('divis','urban_type'))

# Comptue standard errors: ------------------------------------------------------
recs_fuel_prop <- recs_fuel_prop_rep %>%
  group_by(divis, urban_type) %>%
  summarize(Natural_gas = mean(Natural_gas), Propane = mean(Propane), Fuel_oil_kerosene =
            mean(Fuel_oil_kerosene), Electricity = mean(Electricity), Wood = mean(Wood),
            Some_other_fuel = mean(Some_other_fuel),
            stderr_ng = 2*sqrt(mean({Natural_gas_r - Natural_gas}^2)),
            stderr_p = 2*sqrt(mean({Propane_r - Propane}^2)),
            stderr_k = 2*sqrt(mean({Fuel_oil_kerosene_r - Fuel_oil_kerosene}^2)),
            stderr_e = 2*sqrt(mean({Electricity_r - Electricity}^2)),
            stderr_w = 2*sqrt(mean({Wood_r - Wood}^2)),
            stderr_o = 2*sqrt(mean({Some_other_fuel_r - Some_other_fuel}^2))
            ) %>%
  group_by(divis, urban_type) %>%
  transmute(Natural_gas = paste0(as.character(round(Natural_gas,2)),' +- ' ,
                                 as.character(round(qnorm(0.975)*stderr_ng,2)), '%' ),
            Propane = paste0(as.character(round(Propane,2)),' +- ' ,
                             as.character(round(qnorm(0.975)*stderr_p,2)), '%' ),
            Fuel_oil_kerosene = paste0(as.character(round(Fuel_oil_kerosene,2)),' +- ' ,
                                       as.character(round(qnorm(0.975)*stderr_k,2)), '%' ),
            Electricity = paste0(as.character(round(Electricity,2) ),' +- ' ,
                                 as.character(round(qnorm(0.975)*stderr_e,2) ), '%' ),
            Wood = paste0(as.character(round(Wood,2) ),' +- ' ,
                          as.character(round(qnorm(0.975)*stderr_w) ), '%' ),
            Some_other_fuel = paste0(as.character(round(Some_other_fuel,2) ),' +- ' ,
                                     as.character(round(qnorm(0.975)*stderr_o,2) ), '%' )
            )


# Output a markdown table: ------------------------------------------------------
cap_title <- '**Table 1.** *Proprtion of homes using each level of "main space heating fuel".*'
cap_text0 <- 'Each row shows unique combination of census division and census (2010) urban type.'
cap <- paste(cap_title, cap_text0)

cols <- c('division', 'urban type', 'Natural_gas',
          'Propane', 'Fuel_oil_kerosene', 'Electricity', 'Wood', 'Some_other_fuel')

knitr::kable(recs_fuel_prop, digits=1, caption=cap, col.names = cols)
```

Table 1. *Proprtion of homes using each level of "main space heating fuel".* Each row shows unique combination of census division and census (2010) urban type.

division	urban type	Natural_gas	Propane	Fuel_oil_kerosene	Electricity	Wood	Some_other_fuel
East North Central	Rural	44.51 +- 13.03%	20.58 +- 13.81%	0 +- 0%	30.84 +- 11.2%	4.07 +- 2%	0 +- 0%
East North Central	Urban Area	80.56 +- 4.59%	0.48 +- 0.71%	0.43 +- 0.85%	17.79 +- 4.2%	0.73 +- 1%	0 +- 0%
East North Central	Urban Cluster	81.85 +- 9.27%	1.42 +- 2.61%	0 +- 0%	14.78 +- 7.46%	1.95 +- 2%	0 +- 0%
East South Central	Rural	16.58 +- 12.84%	19.24 +- 4.87%	1.12 +- 2.43%	56.1 +- 12.19%	6.95 +- 4%	0 +- 0%
East South Central	Urban Area	35.86 +- 14.14%	0 +- 0%	0 +- 0%	64.14 +- 14.14%	0 +- 0%	0 +- 0%
East South Central	Urban Cluster	24.72 +- 18.91%	6.31 +- 5.01%	0.7 +- 2.21%	68.26 +- 15.61%	0 +- 0%	0 +- 0%
Middle Atlantic	Rural	17.37 +- 15.41%	12.69 +- 7.89%	27.16 +- 9.2%	13.1 +- 11.17%	24.33 +- 6%	5.36 +- 3.78%
Middle Atlantic	Urban Area	65.76 +- 7.94%	1.2 +- 1.32%	15.93 +- 5.17%	16.02 +- 4.22%	0.55 +- 1%	0.54 +- 0.68%
Middle Atlantic	Urban Cluster	68.21 +- 23.58%	5.54 +- 6.19%	10.25 +- 9.22%	7.47 +- 9.26%	8.52 +- 14%	0 +- 0%
Mountain North	Rural	61.07 +- 13.29%	13.43 +- 7.72%	0 +- 0%	14.34 +- 15.08%	10.46 +- 5%	0.7 +- 1.71%
Mountain North	Urban Area	83.38 +- 9.89%	0.67 +- 1.19%	0 +- 0%	15.31 +- 9.23%	0.64 +- 1%	0 +- 0%
Mountain North	Urban Cluster	83.92 +- 13.57%	0 +- 0%	0 +- 0%	16.08 +- 13.57%	0 +- 0%	0 +- 0%
Mountain South	Rural	16.16 +- 29.07%	33.96 +- 4.14%	0 +- 0%	12.18 +- 13.03%	37.7 +- 24%	0 +- 0%
Mountain South	Urban Area	53.8 +- 19.21%	0.85 +- 2.53%	0 +- 0%	45.35 +- 19.3%	0 +- 0%	0 +- 0%
Mountain South	Urban Cluster	100 +- 0%	0 +- 0%	0 +- 0%	0 +- 0%	0 +- 0%	0 +- 0%
New England	Rural	0.43 +- 0.82%	20.77 +- 5.46%	57.21 +- 7.77%	0.93 +- 1.06%	20.67 +- 11%	0 +- 0%
New England	Urban Area	56.45 +- 14.73%	3.01 +- 3.72%	26.32 +- 10.64%	11.91 +- 5.85%	2.31 +- 3%	0 +- 0%
New England	Urban Cluster	36.99 +- 24.59%	3.08 +- 2.6%	46.66 +- 22.58%	8.95 +- 4.96%	4.32 +- 6%	0 +- 0%
Pacific	Rural	4.85 +- 3.21%	18.44 +- 15.6%	5.1 +- 3.45%	49.54 +- 12.52%	22.07 +- 12%	0 +- 0%
Pacific	Urban Area	65.85 +- 7.07%	0.11 +- 0.22%	0.24 +- 0.51%	32.33 +- 6.79%	1.48 +- 1%	0 +- 0%
Pacific	Urban Cluster	32.31 +- 14.49%	0 +- 0%	0 +- 0%	63.05 +- 18.95%	4.65 +- 6%	0 +- 0%
South Atlantic	Rural	8.7 +- 6.11%	8.93 +- 4.46%	5.01 +- 4%	70.99 +- 8.14%	6.36 +- 5%	0 +- 0%
South Atlantic	Urban Area	36.65 +- 7.65%	2.1 +- 1.77%	3.55 +- 1.94%	56.74 +- 6.75%	0.96 +- 1%	0 +- 0%

division	urban type	Natural_gas	Propane	Fuel_oil_kerosene	Electricity	Wood	Some_other_fuel
South Atlantic	Urban Cluster	34.95 +- 27.38%	3.02 +- 3.27%	0.91 +- 2.41%	61.12 +- 25%	0 +- 0%	0 +- 0%
West North Central	Rural	24.2 +- 12.44%	29.8 +- 10.25%	1.61 +- 1.72%	27.88 +- 9.4%	16.52 +- 5%	0 +- 0%
West North Central	Urban Area	77.29 +- 8.21%	1.51 +- 1.35%	0 +- 0%	21.2 +- 8.25%	0 +- 0%	0 +- 0%
West North Central	Urban Cluster	80.01 +- 12.95%	2.7 +- 3.11%	0.76 +- 2.18%	14.9 +- 10.07%	0 +- 0%	1.63 +- 3.24%
West South Central	Rural	26.72 +- 9.59%	15.1 +- 7.06%	0 +- 0%	51.72 +- 11.2%	6.46 +- 4%	0 +- 0%
West South Central	Urban Area	43.88 +- 7.75%	0 +- 0%	0 +- 0%	55.9 +- 7.87%	0.22 +- 0%	0 +- 0%
West South Central	Urban Cluster	43.76 +- 12.62%	0 +- 0%	0 +- 0%	56.24 +- 12.62%	0 +- 0%	0 +- 0%

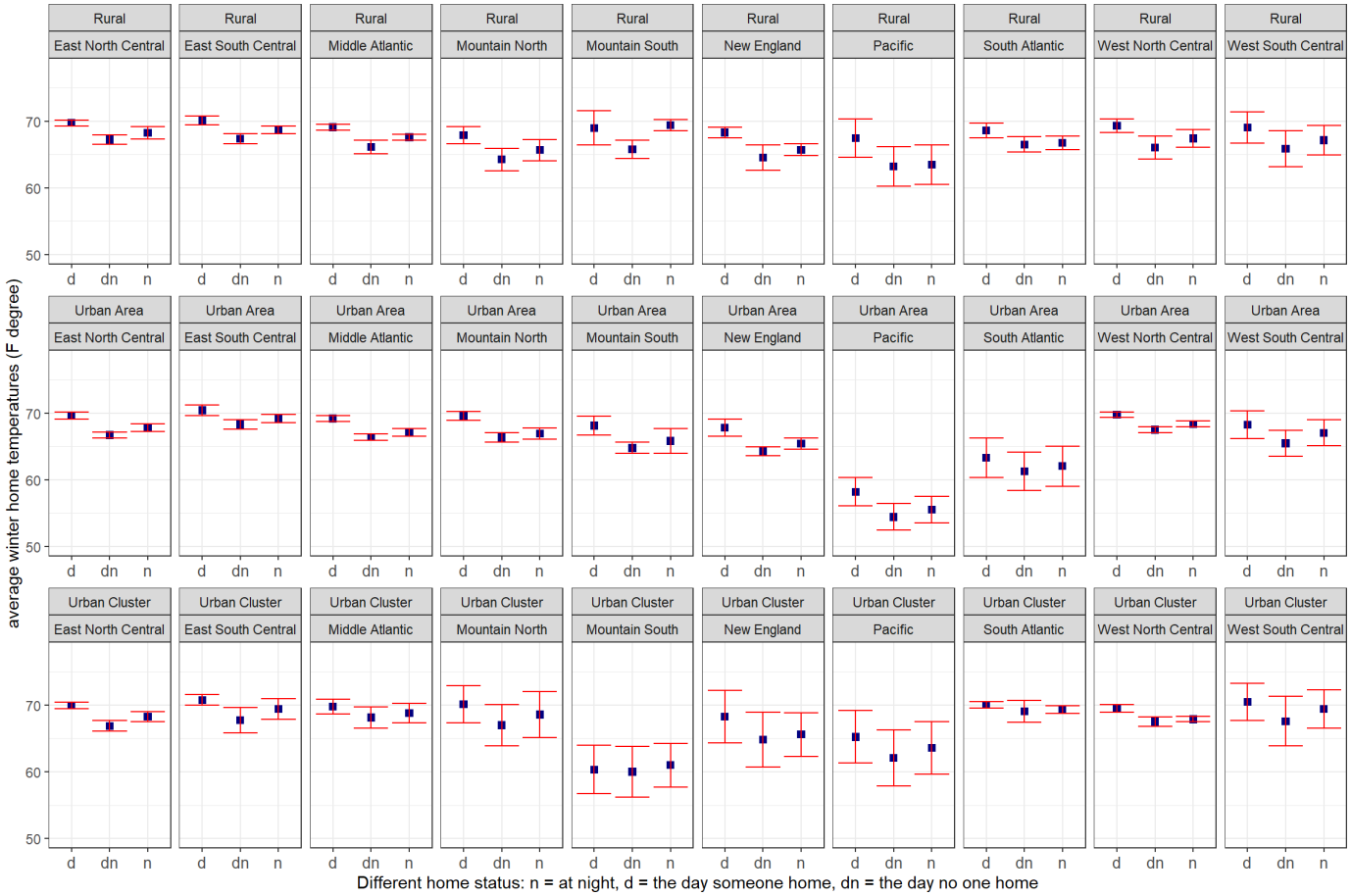
The 95% confidence intervals of each proportion of the fuel used in space heating are shown in the above table, the first value for each proportion represent the estimation of proportion using final weights as full sample.

Plot Comparing Home Temperature

```
# Transforming the inappliable temperature data from label -2 to NA.
fill_tmp_NA = function(x) {
  if (x == -2) {re <- NA}
  else {re <- x}
  return(re)
}

fill_all_tmp_NA = function(x) {
  sapply(x, fill_tmp_NA)
}


# in the Fay's balanced repeated replication (BRR) method of estimating standard error
# we set \epsilon = 0.5 and R = the number of replicate subsamples then the estimated
# standard error could be reduced to var(samples) /

recs_tmp <- recs %>%
  transmute(DOEID, divis = DIVISION, urban_type = UATYP10, d = TEMPHOME,
            dn = TEMPGONE, n = TEMPNITE, weight = NWEIGHT) %>%
  mutate(divis = decode_all_division(divis), urban_type = decode_all_urban(urban_type),
         d = fill_all_tmp_NA(d), dn = fill_all_tmp_NA(dn),
         n = fill_all_tmp_NA(n), weight = weight)


recs_tmp_avg <- recs_tmp %>%
  gather(key = 'tmp_type', value = 'tmp', d, dn, n) %>%
  group_by(divis, urban_type, tmp_type) %>%
  summarize(tmp_avg = sum(sapply(tmp, as.numeric) * weight, na.rm = T) / sum(weight))

weights_long_all <- recs %>%
  select(DOEID, BRRWT1:BRRWT96) %>%
  gather(key = 'repl', value = 'w', BRRWT1:BRRWT96)

recs_tmp_rep <- weights_long_all %>%
  left_join(recs_tmp %>% mutate( DOEID=as.integer(DOEID) ) , by='DOEID' )

# Check nothing is lost
if( nrow(recs_tmp_rep) != nrow(weights_long_all) ) {
    stop("DOEID mismatch!")
  }

recs_tmp_avg_rep <- recs_tmp_rep %>%
  gather(key = 'tmp_type', value = 'tmp', d, dn, n) %>%
  group_by(divis, urban_type, repl, tmp_type) %>%
  summarize(tmp_avg_r = sum(sapply(tmp, as.numeric) * w, na.rm = T) / sum(w) )

recs_tmp_avg_rep <- recs_tmp_avg_rep %>%
  left_join(recs_tmp_avg, by = c ('divis','urban_type', 'tmp_type'))

recs_tmp_avg <- recs_tmp_avg_rep %>%
  group_by(divis, urban_type, tmp_type) %>%
  summarize(tmp_avg = mean(tmp_avg),
            stderr = 2*sqrt(mean({tmp_avg_r - tmp_avg}^2))) %>%
  group_by(divis, urban_type, tmp_type) %>%
  mutate(lwr = tmp_avg - qnorm(.975)*stderr,
         upr = tmp_avg + qnorm(.975)*stderr
         )
```

Firstly, I prepare the average temperature for each case and pre-calculated the upper and lower bound of the average temperature.

```
p_sfd <- recs_tmp_avg %>%
  ungroup() %>%
  ggplot( aes( x = tmp_type, y = tmp_avg) ) +
  geom_point( col='navy', pch = 15, cex=2) +
  geom_errorbar( aes(ymin=lwr, ymax=upr), col='red' ) +
  facet_wrap(~urban_type+divis, scales='free_x', nrow=3, ncol=10) +
  theme_bw() +
  theme( axis.text.x =
           element_text(
             angle = 0,
             size = 10
           ), plot.margin = margin(0.2,0.2,0.2,0.2,"cm") ) +
  ylim( c(50, 78) ) +
  xlab('Different home status: n = at night, d = the day someone home, dn = the day no one home') +
  ylab('average winter home temperatures (F degree)')
p_sfd
```



The Median difference between the daytime (with someone home) and nighttime temperatures

```
decode_behavior = function(x) {
  if(!is.numeric(x)) stop('decode_behavior expects numeric input indexed from 1!')

  re <- as.character( factor(x, levels = c(1, 2, 3, 4, 5, 9),
                             labels = c('Set one temperature and leave it there most of the time',
                                        'Manually adjust the temperature at night or when no one is at ho
me',
                                        paste0('Program the thermostat to automatically adjust the',
                                        ' temperature during the day and night at certain times'),
                                        'Turn equipment on or off as needed',
                                        'Our household does not have control over the equipment',
                                        'Other')

  ))
  return(re)
}

decode_behavior_all = function(x) {
  return(sapply(x, decode_behavior))
```

```
}

# Key values for each observation: ------------------------------------------
recs_mtmp <- recs %>%
  filter(HEATHOME == 1) %>%
  transmute(DOEID, bhvr = decode_behavior_all(EQUIPMUSE), weight = NWEIGHT, day = TEMPHOME, night = TEMPN
ITE) %>%
  replace_na(list(weight=0))  %>%
  filter( (bhvr > 0) & (day > 0) & (night >0))

# Convert weights to long: ------------------------------------------------
weights_long <- recs %>%
  filter(HEATHOME == 1) %>%
  select(DOEID, BRRWT1:BRRWT96) %>%
  gather(key = 'repl', value = 'w', BRRWT1:BRRWT96)

recs_mtmp_rep <- weights_long %>%
  left_join(recs_mtmp %>% mutate( DOEID=as.integer(DOEID) ) , by='DOEID' )

# Check nothing is lost
if( nrow(recs_mtmp_rep) != nrow(weights_long) ) {
  stop("DOEID mismatch!")
}

recs_mtmp_med <- recs_mtmp %>%
  gather(key = 'tmp_type', value = 'tmp', day, night)  %>%
  arrange(tmp) %>%
  group_by(bhvr, tmp_type) %>%
  summarize(tmp_med = tmp[min(which(cumsum(weight) > 0.5*sum(weight)))] )


recs_mtmp_med_rep <- recs_mtmp_rep %>%
  gather(key = 'tmp_type', value = 'tmp', day, night) %>%
  group_by(bhvr, tmp_type, repl) %>%
  summarize(tmp_med_r = tmp[min(which(cumsum(w) > 0.5*sum(w)))] )

recs_mtmp_med_rep <- recs_mtmp_med_rep %>%
  left_join(recs_mtmp_med, by = c ('bhvr', 'tmp_type'))


recs_mtmp_med <- recs_mtmp_med_rep %>%
  group_by(bhvr, tmp_type) %>%
  summarize(tmp_med = mean(tmp_med),
            stderr = 2*sqrt(mean({tmp_med_r - tmp_med}^2))) %>%
  group_by(bhvr, tmp_type) %>%
  mutate(lwr = tmp_med - qnorm(.975)*stderr,
         upr = tmp_med + qnorm(.975)*stderr
  )



recs_mtmp_med_tib <- recs_mtmp_med %>%
  tidyr::pivot_wider(names_from = tmp_type, values_from = c(tmp_med, stderr,
                                                            lwr, upr))
```

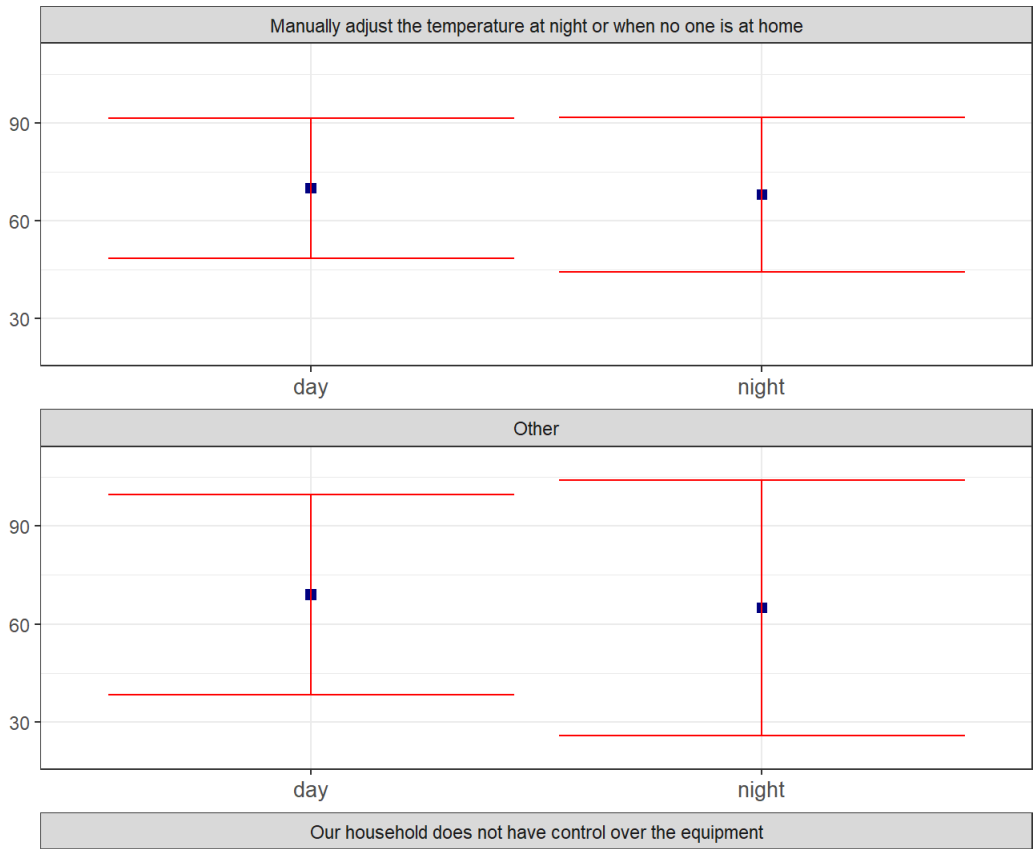
```
# Output a markdown table: ------------------------------------------------------
cap_title <- '**Table 2.** *The (national) median difference between the daytime (with someone home) and
nighttime temperatures".*'
cap_text0 <- 'Each row shows each level of "main heating equipment household behavior".'
cap <- paste(cap_title, cap_text0)

cols <- c('equipment behavior', 'tmp_med_day', 'tmp_med_night',
          'stderr_day', 'stderr_night', 'lwr_day', 'lwr_night', 'upr_day', 'upr_night')

knitr::kable(recs_mtmp_med_tib, digits=4, caption=cap, col.names = cols)
```

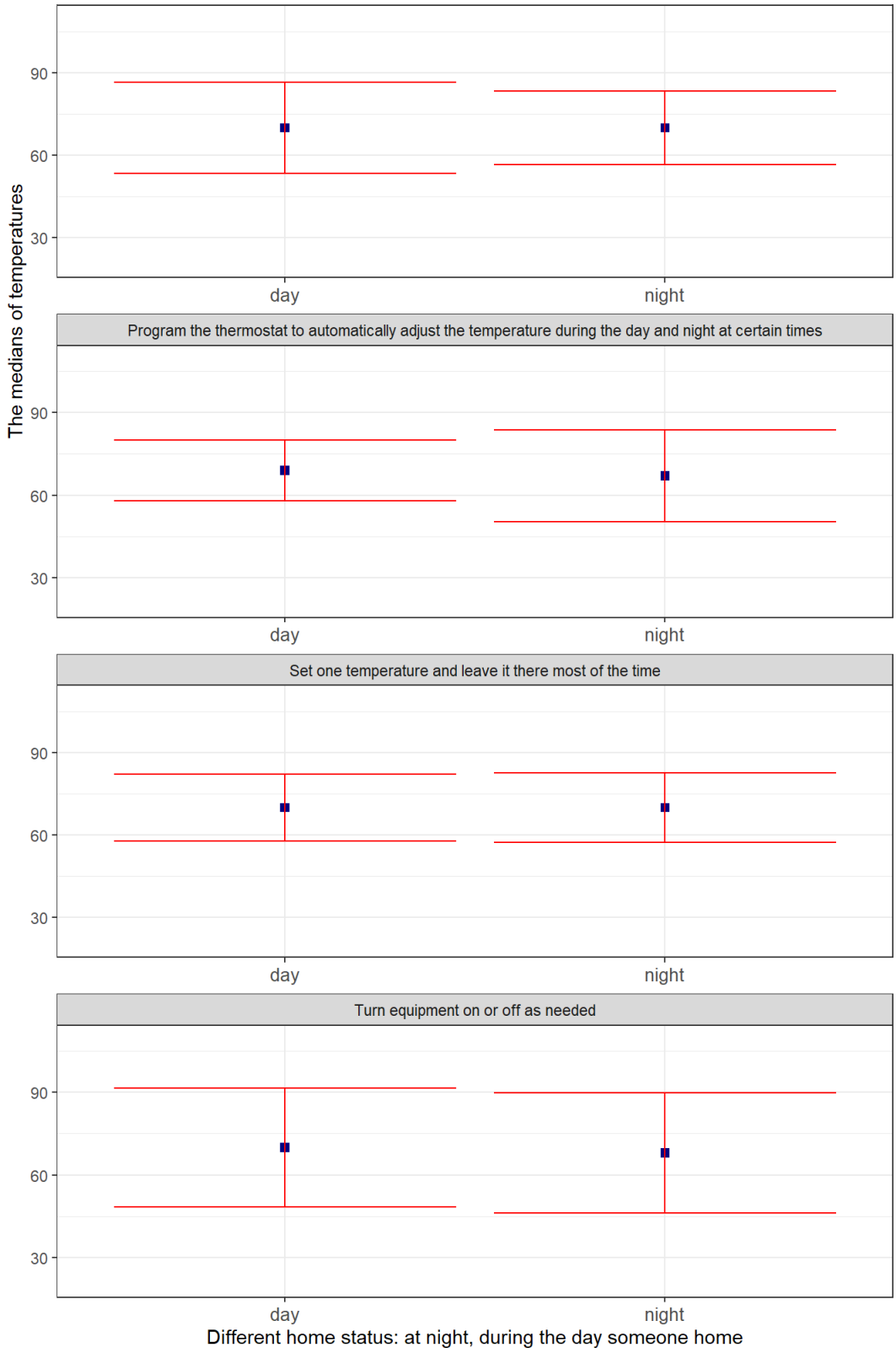
Table 2. *The (national) median difference between the daytime (with someone home) and nighttime temperatures".* Each row shows each level of "main heating equipment household behavior".

equipment behavior	tmp_med_day	tmp_med_night	stderr_day	stderr_night	lwr_day	lwr_night	upr_day	upr_night

equipment behavior	tmp_med_day	tmp_med_night	stderr_day	stderr_night	lwr_day	lwr_night	upr_day	upr_night
Manually adjust the temperature at night or when no one is at home	70	68	10.9221	12.0899	48.5931	44.3042	91.4069	91.6958
Other	69	65	15.6418	19.9875	38.3426	25.8252	99.6574	104.1748
Our household does not have control over the equipment	70	70	8.4508	6.7915	53.4367	56.6888	86.5633	83.3112
Program the thermostat to automatically adjust the temperature during the day and night at certain times	69	67	5.6384	8.5147	57.9489	50.3115	80.0511	83.6885
Set one temperature and leave it there most of the time	70	70	6.1981	6.4872	57.8519	57.2854	82.1481	82.7146
Turn equipment on or off as needed	70	68	11.0491	11.1878	48.3441	46.0723	91.6559	89.9277

```
p_med <- recs_mtmp_med %>%
  ungroup() %>%
  ggplot( aes( x = tmp_type, y = tmp_med) ) +
  geom_point( col='navy', pch = 15, cex=2) +
  geom_errorbar( aes(ymin=lwr, ymax=upr), col='red' ) +
  facet_wrap(~bhvr, scales='free_x', nrow=6) +
  theme_bw() +
  theme( axis.text.x =
           element_text(
             angle = 0,
             size = 10
           ), plot.margin = margin(0.2,0.2,0.2,0.2,"cm") ) +
  ylim( c(20, 110) ) +
  xlab('Different home status: at night, during the day someone home') +
  ylab('The medians of temperatures')
p_med
```





Among all “main heating equipment household behaviors”, the median temperatures during the daytime (with someone home) are always greater than or equal to the ones at night. The behaviors “Manually adjust the temperature at night or when no one is at home” and “Program the thermostat to automatically adjust the temperature during the day and night at certain times” and “Other” result in the confidence intervals of median night temperatures become wider than the daytime ones. The “Set one temperature and leave it there most of the time” results in the confidence interval of median night temperature become more narrow than the daytime one. The other two behavior have no influnce on the confidence interval.

Problem Set 2

Source the codes used in PS 1 Question 3

```
# libraries including ------------------------------------------------------
source('./ps2_q2_funcs.R', encoding = 'UTF-8')
library(tidyverse)
library(mousetrap)
```

```
## Welcome to mousetrap 3.1.3!
```

```
## Summary of recent changes: http://pascalkieslich.github.io/mousetrap/news
```

```
## Forum for questions: http://forum.cogsci.nl/index.php?p=/categories/mousetrap
```

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

Load the data and examine the columns

```
# read in the  data: -------------------------------------------------------

factor_to_numeric_list = function(x) {
  tmp = str_replace(x, '(\\[)', '') %>%
    str_replace('(\\])', '') %>%
    str_split(', ')
  return(sapply(tmp, as.numeric))
}

data_prep = function(x) {
  return(
    x %>%
    transmute(subject = subject_nr, trial = count_trial,
              x = factor_to_numeric_list(xpos_get_response),
              y = factor_to_numeric_list(ypos_get_response),
              tm = factor_to_numeric_list(timestamps_get_response),
              correct)
  )
}

sakh <- data_prep(mousetrap::KH2017_raw)
lapply(sakh, class)
```

```
## $subject
## [1] "integer"
##
## $trial
## [1] "integer"
##
## $x
## [1] "list"
##
## $y
## [1] "list"
##
## $tm
## [1] "list"
##
## $correct
## [1] "integer"
```

In this part we use “list” columns for each component.

Apply to measure curvature

```
normalize_dim3 = function(x,y,t) {
  a = unname(unlist(x))
  b = unname(unlist(y))
  c = unname(unlist(t))
  tmp = unname(cbind(a,b,c))
  return(list(normalize(tmp)) )
}

measure_curv_list = function(x) {
  tmp = (matrix(unlist(x), ncol = 3))
  return(list(measure_curvature(tmp)) )
}

unlist_curv = function(x,f) {
  tmp = unname(unlist(x))
  if(f == 'tot_dist') return(tmp[1])
  if(f == 'max_abs_dev') return(tmp[2])
  if(f == 'avg_abs_dev') return(tmp[3])
  if(f == 'AUC') return(tmp[4])
}

norm_sakh <- sakh %>%
  filter(correct == 1) %>%
  group_by(subject, trial) %>%
  summarize(norm_xyt = normalize_dim3(x,y,tm))

meas <- norm_sakh %>%
  group_by(subject,trial) %>%
  summarize(curv = measure_curv_list(norm_xyt) ) %>%
  group_by(subject,trial) %>%
  summarize( tot_dist = unlist_curv(curv, 'tot_dist'),
         max_abs_dev = unlist_curv(curv, 'max_abs_dev'),
         avg_abs_dev = unlist_curv(curv, 'avg_abs_dev'),
         AUC = unlist_curv(curv, 'AUC')
         )

con_expl <- mousetrap::KH2017_raw %>%
  transmute(subject = subject_nr, trial = count_trial,
            Condition = Condition, Exemplar = Exemplar,correct) %>%
  filter(correct == 1) %>%
  select(-correct)
options(digits = 7)
meas <- meas %>%
  left_join(con_expl, by=c('subject', 'trial'))
head(meas)
```

```
## # A tibble: 6 x 8
## # Groups:   subject [1]
##   subject trial tot_dist max_abs_dev avg_abs_dev     AUC Condition Exemplar
##     <int> <int>    <dbl>       <dbl>       <dbl>   <dbl> <fct>     <fct>
## 1       1     2    1063.        85.1        6.03  46527. Typical   Loewe
## 2       1     3    1033.        65.4       14.6   16242. Typical   Kaninch~
## 3       1     4    1153.        99.2       18.2   54516. Atypical  Seeloewe
## 4       1     5    1050.        64.3       20.7   32054. Typical   Alligat~
## 5       1     6    1962.       624.        92.5  228193. Atypical  Pinguin
## 6       1     7    1587.       598.        90.9  315864. Typical   Katze
```

The meas data frame contains the curvature measurements for all correct responses and the head of it is shown in the above table.

Fit linear mixed models exploring how each curvature

```
# fit linear mixed model ------------------------------------------------

meas_fitting <- meas %>%
  filter(AUC > 0) %>%
  group_by(subject,trial) %>%
  summarize(tot_dist_log = log(tot_dist),
            max_abs_dev_log = log(max_abs_dev),
            avg_abs_dev_log = log(avg_abs_dev),
            AUC_log = log(AUC),
            Condition = factor(Condition, levels = c('Typical','Atypical'),
                               labels = c(0,1) ),
            Exemplar = factor(Exemplar, levels = c('Aal','Alligator', 'Chamaeleon',
                                                   'Falke', 'Fledermaus',
                                                   'Goldfisch','Hai','Hund',
                                                   'Kaninchen','Katze',
                                                   'Klapperschlange','Lachs',
                                                   'Loewe','Pferd','Pinguin',
                                                   'Schmetterling','Seeloewe',
                                                   'Spatz','Wal'),
                              labels = c(1:19)))

m1 <- lmer(tot_dist_log~Condition+(1|Exemplar)+(1|subject), data = meas_fitting )
m2 <- lmer(max_abs_dev_log~Condition+(1|Exemplar)+(1|subject), data = meas_fitting )
m3 <- lmer(avg_abs_dev_log~Condition+(1|Exemplar)+(1|subject), data = meas_fitting )
m4 <- lmer(AUC_log~Condition+(1|Exemplar)+(1|subject), data = meas_fitting )

re1 <- cbind( matrix(lme4::fixef(m1)[2]), matrix(confint(m1)['Condition1',],nrow=1), mean(fitted.values(m
1)) )
```

```
## Computing profile confidence intervals ...
```

```
re2 <- cbind( matrix(lme4::fixef(m2)[2]), matrix(confint(m2)['Condition1',],nrow=1), mean(fitted.values(m
2)) )
```

```
## Computing profile confidence intervals ...
```

```
re3 <- cbind( matrix(lme4::fixef(m3)[2]), matrix(confint(m3)['Condition1',],nrow=1), mean(fitted.values(m
3)) )
```

```
## Computing profile confidence intervals ...
```

```
re4 <- cbind( matrix(lme4::fixef(m4)[2]), matrix(confint(m4)['Condition1',],nrow=1), mean(fitted.values(m
4)) )
```

```
## Computing profile confidence intervals ...
```

```
re <- data.frame(rbind(re1,re2,re3,re4), row.names = c('tot_dist_log', 'max_abs_dev_log',
                                                       'avg_abs_dev_log','AUC_log'))
re <- cbind(re, exp(re[,1:4]) )
re <- cbind(re, re[5] / re[8])
names(re) <- c('Coef of Condition log', '2.5% lwr log', '97.5% upr log', 'mean log',
               'Coef of Condition', '2.5% lwr', '97.5% upr', 'mean', 'Coef / mean')
```

```
# Output a markdown table: ----------------------------------------------------
cap_title <- '**Table 3.** *The effect Condition have on the curvature measurements.*'
cap <- paste(cap_title, cap_text0)

cols <- c('Coef of Condition log', '2.5% lwr log', '97.5% upr log', 'mean log',
               'Coef of Condition', '2.5% lwr', '97.5% upr', 'mean', 'Coef / mean')

knitr::kable(re, digits=4, caption=cap, col.names = cols)
```

Table 3. *The effect Condition have on the curvature measurements.* Each row shows each level of “main heating equipment household behavior”.

	Coef of Condition log	2.5% lwr log	97.5% upr log	mean log	Coef of Condition	2.5% lwr	97.5% upr	mean	Coef / mean
tot_dist_log	0.1579	0.0838	0.2320	7.1935	1.1710	1.0874	1.2611	1330.7576	0.0009
max_abs_dev_log	0.5048	0.2736	0.7362	4.9026	1.6567	1.3147	2.0880	134.6441	0.0123
avg_abs_dev_log	0.6460	0.3815	0.9109	3.2563	1.9080	1.4645	2.4865	25.9540	0.0735
AUC_log	0.4058	0.2057	0.6063	11.1280	1.5005	1.2284	1.8337	68049.3931	0.0000

From the above table, we find the coefficient of Condition on “the average absolute deviation of the observed trajectory from the direct path” is the maximum. Moreover, the expectation of average absolute deviation is also the smallest, which means the unit change in Condition variable is more influential in changing the distribution of average absolute deviation. Considering the two facts in average absolute deviation, we can conclude that the Condition has largest effect on the average absolute deviation.