

SMR over time (year and day/night)

Amber Lee

December 2020

Set up (from 1 Set up SMR)

```
library(RMySQL)

library(tidyverse)
library(tidyr)
library(broom)
library(lubridate)
library(stringr)
library(kableExtra)
library(data.table) # for transposing coordinates

library(XML)
library(RCurl)
library(lutz)
library(suncalc)

dataset_names <- dbGetQuery(con, "SHOW TABLES")[[1]]

# remove datasets with "_" in the name
dataset_names <- dataset_names[str_detect(dataset_names, "_", negate = TRUE)]

query_sample <- function(dataset_str, percent){
  # input is dataset_str (str) with dataset name, and percent (dbl) for the random sample %
  # output is the dataframe with a column added for the name of dataset and character NA's
  # replaced with NA's
  # global variable con is the SQL connection

  command <- paste("SELECT * FROM", dataset_str, "WHERE rand() <=", percent,
                  "# in SQL, filter for vehicular stops",
                  "# AND type = 'vehicular'", sep = " ")

  df <- dbGetQuery(con, command) %>% mutate(dataset_name = dataset_str)

  # do not consider empty datasets
  if (dim(df)[1] == 0){
    return(NULL)
  }

  # replace character NA's with NA
```

```

if (sum(is.na(df) == 0)) {
  df[df == "NA"] = NA
}

return(df %>% dplyr::select(-type))
}

dataset_lst <- lapply(dataset_names, query_sample, 0.15)

# remove empty datasets through logical indexing
dataset_lst <- dataset_lst[sapply(dataset_lst, function(x) isTRUE(nrow(x) > 0))]

check_nonempty <- function(var, dataset, n_obs){ 
  # helper function for removing empty columns

  # the function environment has the parameter dataset
  col_str <- paste("dataset$", var, sep = "") 
  col <- eval(parse(text = col_str))
  isCollected <- sum(is.na(col)) < n_obs

  return(isCollected)
}

remove_empty_col <- function(dataset){
  # a variable is 'collected' if there is a column for it in the dataset
  # but being collected doesn't imply nonempty

  collected_var <- names(dataset)
  n_obs <- dim(dataset)[1]

  nonempty_bools <- unlist(lapply(collected_var, check_nonempty, dataset, n_obs))

  # use logial indexing!
  nonempty_var <- collected_var[nonempty_bools]

  ## in case i need this information
  # empty_var <- collected_var[!nonempty_bools]

  return(dataset %>% dplyr::select({{ nonempty_var }}))
}

dataset_lst <- lapply(dataset_lst, remove_empty_col)

myfilter_for <- function(dataset, var_vect, need_containment){
  # if need_containment is true, then function only returns
  # datasets containing ALL variables specified in var_vect

  # need_containment = TRUE results in more restrictive filtering
}

```

```

dataset_var <- names(dataset)
intersection <- var_vect[var_vect %in% dataset_var]

if (need_containment){

  if (length(intersection) == length(var_vect)) {
    # embrace syntax from dplyr programming
    return(dataset %>% dplyr::select({{ var_vect }}))
  } else {
    return(NULL)
  }

} else if (!need_containment){

  if (length(intersection) > 0) {
    # embrace syntax from dplyr programming
    return(dataset %>% dplyr::select({{ intersection }}))
  } else{
    return(NULL)
  }

}

}

datasetContaining <- function(dataset, var_vect){
  # var_vect is str with the variables we WANT
  # returns the whole dataset

  if(var_vect %in% names(dataset)){
    return(dataset)

  } else {
    return(NULL)
  }

}

find_dataset <- function(dataset, name_str){

  if(dim(dataset)[1] <= 1){
    return(NULL)
  }

  if(dataset$dataset_name[1] == name_str){
    return(dataset)
  }

}

mysearch_dataset <- function(dataset_list, name_str){

  df <- lapply(dataset_list, find_dataset, name_str)
}

```

```

df <- df[sapply(df, function(x) isTRUE(nrow(x) > 0))]

return(df[[1]])

}

check_missing <- function(n_threshold, df){

  col <- df %>%
    mutate("isMissing_{n_threshold}" := case_when(missing >= n_threshold ~ TRUE,
                                                   TRUE ~ FALSE)) %>%
    select(starts_with("isMissing"))

  return(col)

}

countMissing <- function(dataset, n_threshold, exclude_bool, exclude_var){

  # <n_threshold> is used to classify the observations with
  # at least n_threshold missing values as completely missing

  # <exclude_var> is str specifying which variables we don't count for NA's

  n_var <- dim(dataset)[2]

  if (exclude_bool){
    missing <- list(missing = rowSums(is.na(dataset %>% select(-all_of(exclude_var)))))

  } else {
    missing <- list(missing = rowSums(is.na(dataset)))

  }

  dataset <- dataset %>%
    bind_cols(list(missing), .id = NULL) %>%
    mutate(stop_missing_rate = missing/n_var)

  dataset <- dataset %>%
    # check_missing operates on dataset with missingness already counted
    bind_cols(lapply(1:n_threshold, check_missing, dataset))

  return(as.data.frame(dataset))

}

missing_lst <- lapply(dataset_lst, countMissing, 1, FALSE)

# 15 most frequently recorded variables
freq_var <- data.frame("var" = unlist(lapply(dataset_lst, function(dataset) names(dataset)))) %>%
  group_by(var) %>%
  summarize(count = n(), .groups = "drop") %>%
  # drop type (either pedestrian or vehicular)
  filter(var != "type" & str_detect(var, "row", negate = TRUE)) %>%
  # could make this function that returns top n var

```

```

slice_max(count, n = 16) %>%
pull(var)

```

SMR by week (throughout years)

We do *not* require containment when we use `myfilter_for`.

```

date_lst <- lapply(dataset_lst, myfilter_for, freq_var, FALSE)

# filtering OUT datasets that don't record date
date_lst <- lapply(date_lst, datasetContaining, "date")
date_lst <- date_lst[sapply(date_lst, function(x) isTRUE(nrow(x) > 0))]

# helper function
summarizeMissingTemporally <- function(dataset){

  dataset <- dataset %>%
    mutate(nice_date = ymd(date),
          nice_week = ymd(cut(nice_date, "week")))) %>%
    select(dataset_name, nice_week, stop_missing_rate) %>%
    ungroup() %>%
    group_by(nice_week, dataset_name) %>%
    summarize(avg_SMR = mean(stop_missing_rate), stop_count = n(), .groups = "drop")

  return(dataset)
}

makeDF.SMRByDate <- function(date_list){

  # count NA
  date_list <- lapply(date_list, countMissing, 1, TRUE, "date")

  # summarize
  date_list <- lapply(date_list, summarizeMissingTemporally)

  state_df <- data.frame(state = state.abb, region = state.region)

  southern_str <- state_df$state[state_df$region == "South"]
  west_str <- state_df$state[state_df$region == "West"]
  northeast_str <- state_df$state[state_df$region == "Northeast"]
  northcent_str <- state_df$state[state_df$region == "North Central"]

  nvar_per_dataset <- data.frame(dataset_name = sapply(dataset_lst,
                                                        function(dataset) dataset$dataset_name[1]),
                                    n_var = sapply(dataset_lst,
                                                  function(dataset) length(names(dataset)))))

  # make df just include date and SMR
  date_df <- bind_rows(date_list) %>%
    mutate(state_abb = substr(dataset_name, 1, 2),
          region = case_when(state_abb %in% southern_str ~ "South",

```

```

    state_abb %in% west_str ~ "West",
    state_abb %in% northeast_str ~ "Northeast",
    state_abb %in% northcent_str ~ "North Central",
    TRUE ~ "No regioned (error!))") %>%
#something off about this left join
left_join(nvar_per_dataset, by = "dataset_name")

return(date_df)
}

date_df <- makeDF.SMRByDate(date_lst)

ggplot_date <- function(date_dataset, region_or_dataset){

plot_region <- function(name){

p <- date_dataset %>%
  filter(region == name) %>%
  ggplot() +
  geom_point(aes(x = nice_week, y = avg_SMR,
                 color = dataset_name), alpha = .8) +
  ggtitle(paste(name, "weekly SMR over time"))

ggsave(paste(name, "weekly SMR.png"), p)
return(p)
}

plot_dataset <- function(name){

p <- date_dataset %>%
  filter(dataset_name == name) %>%
  ggplot() +
  geom_point(aes(x = nice_week, y = avg_SMR), alpha = .8) +
  ggtitle(paste(name, "weekly SMR over time"))

ggsave(paste(name, "weekly SMR.png"), p)
return(p)
}

if (region_or_dataset == "region"){
  region_names <- date_dataset %>% distinct(region) %>% pull (region)
  plots <- lapply(region_names, plot_region)

} else if (region_or_dataset == "dataset"){
  dataset_names <- date_dataset %>% distinct(dataset_name) %>% pull(dataset_name)
  plots <- lapply(dataset_names, plot_dataset)

} else {return(stop("typo in region_or_dataset"))}

return(plots)
}

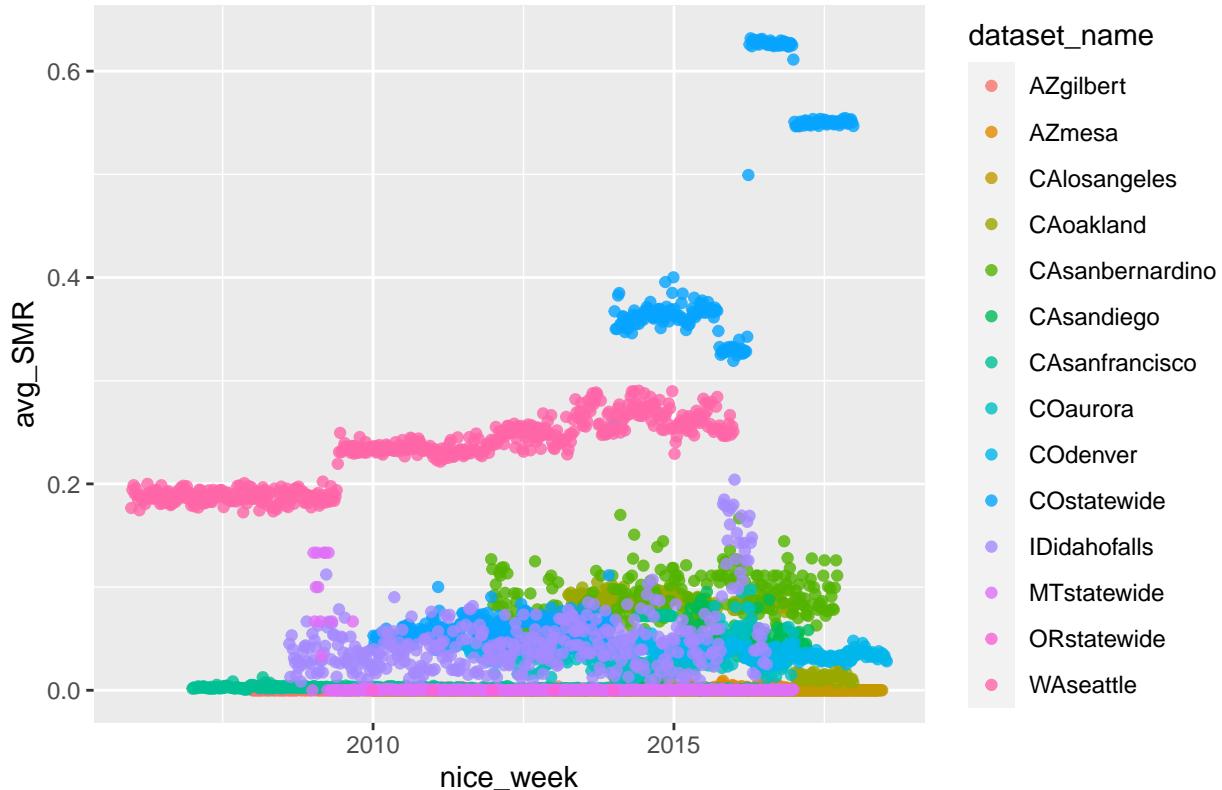
```

```
}
```

```
ggplot_date(date_df, "region")
```

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

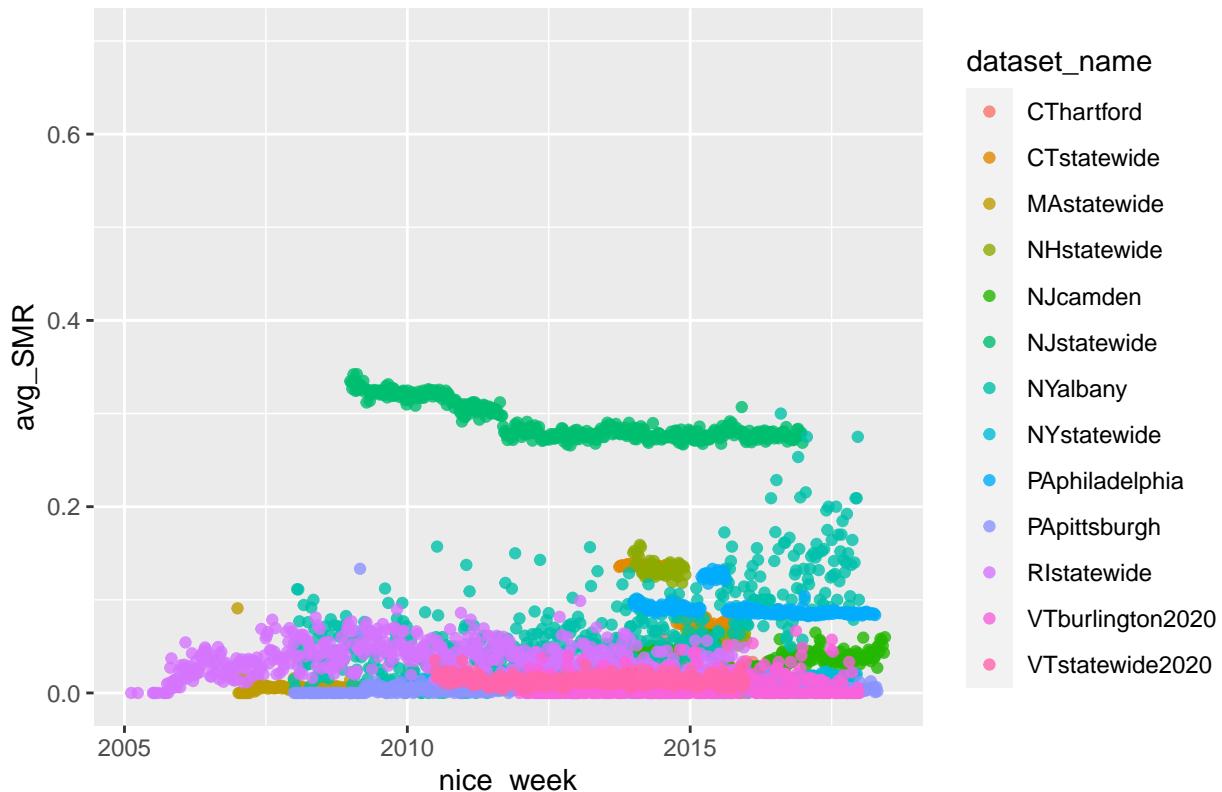
West weekly SMR over time



```
##
```

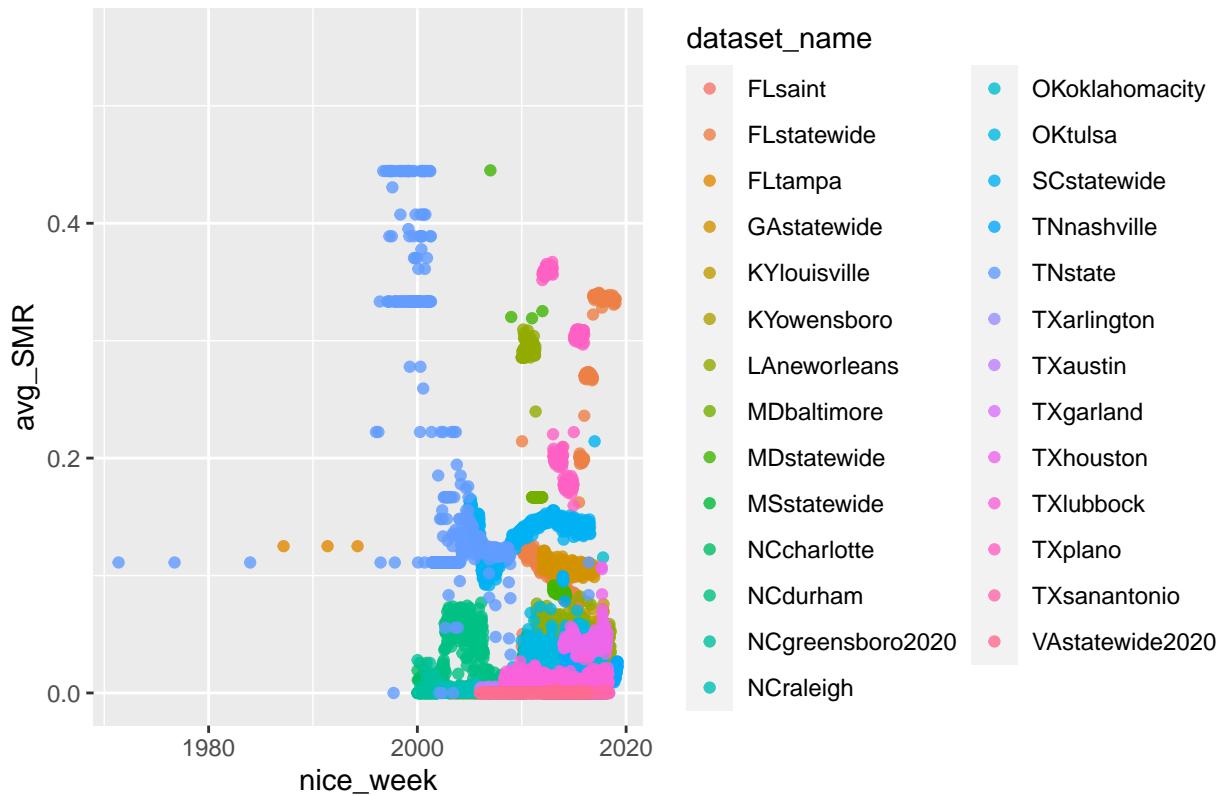
```
## [[2]]
```

Northeast weekly SMR over time



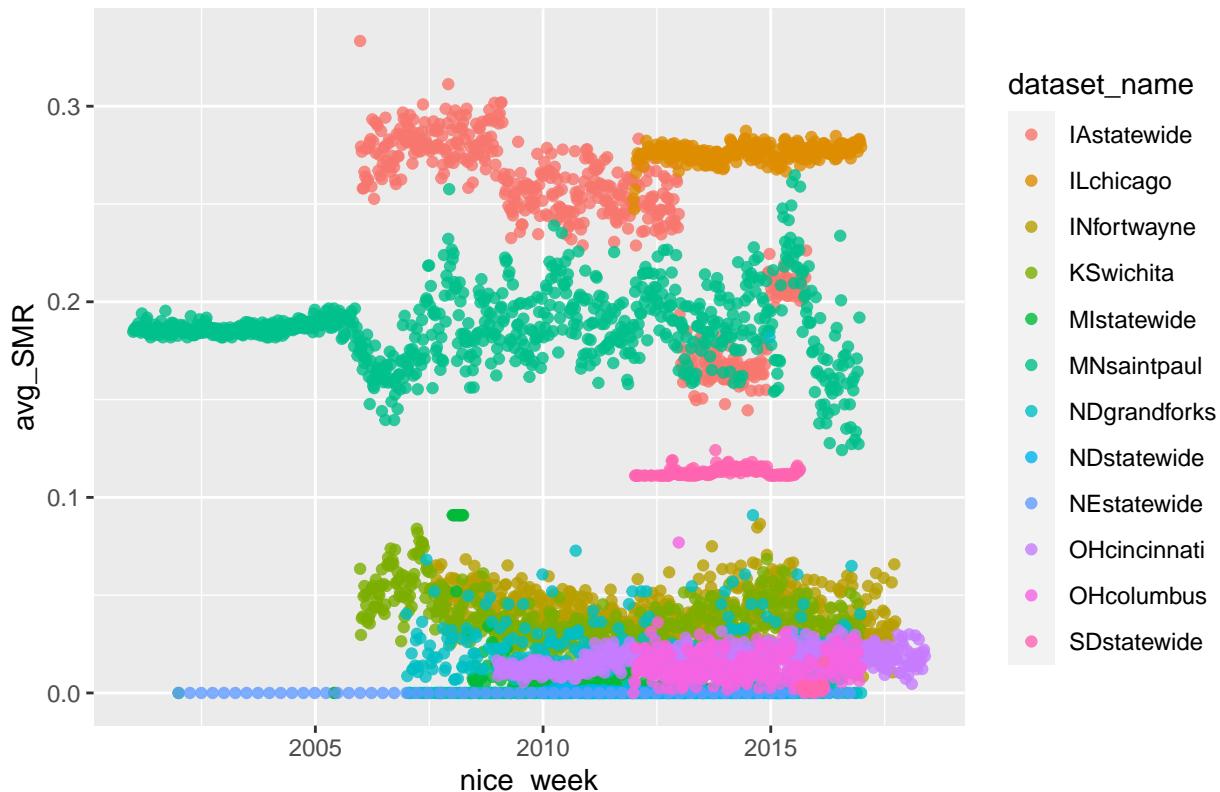
```
##  
## [[3]]
```

South weekly SMR over time



```
##  
## [[4]]
```

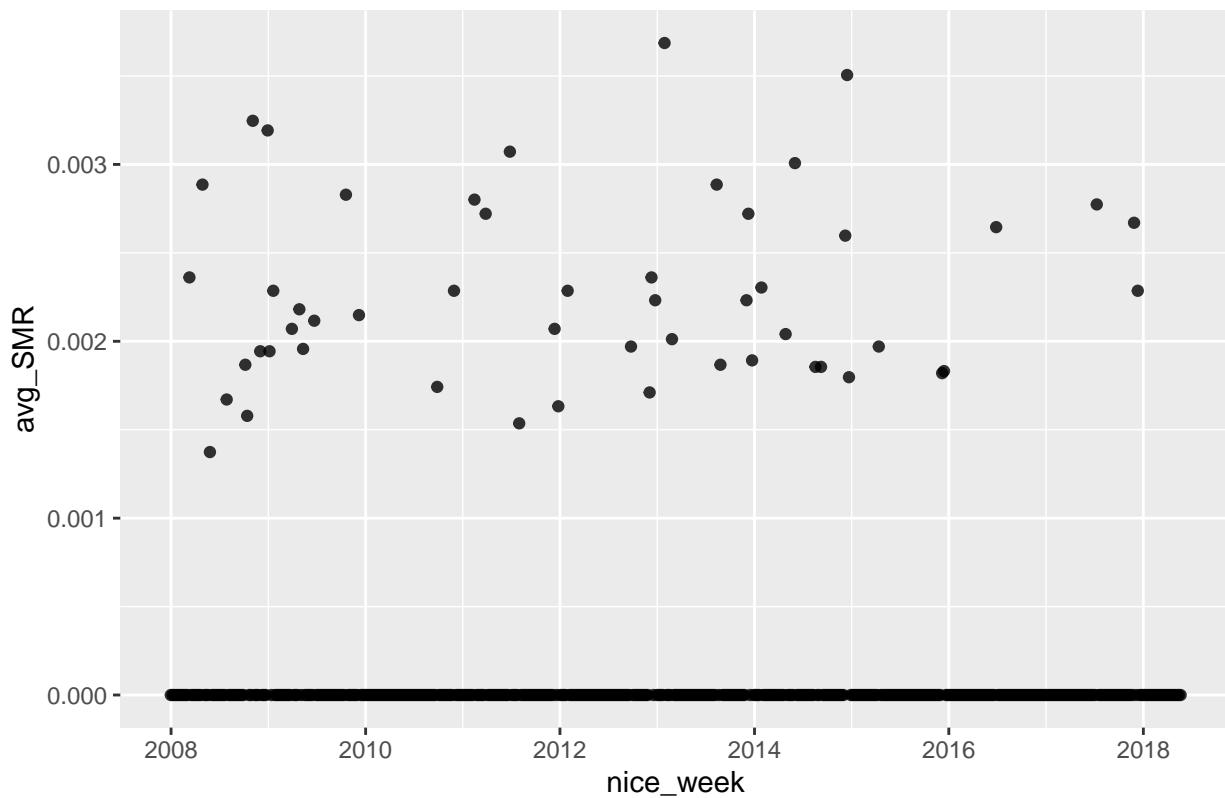
North Central weekly SMR over time



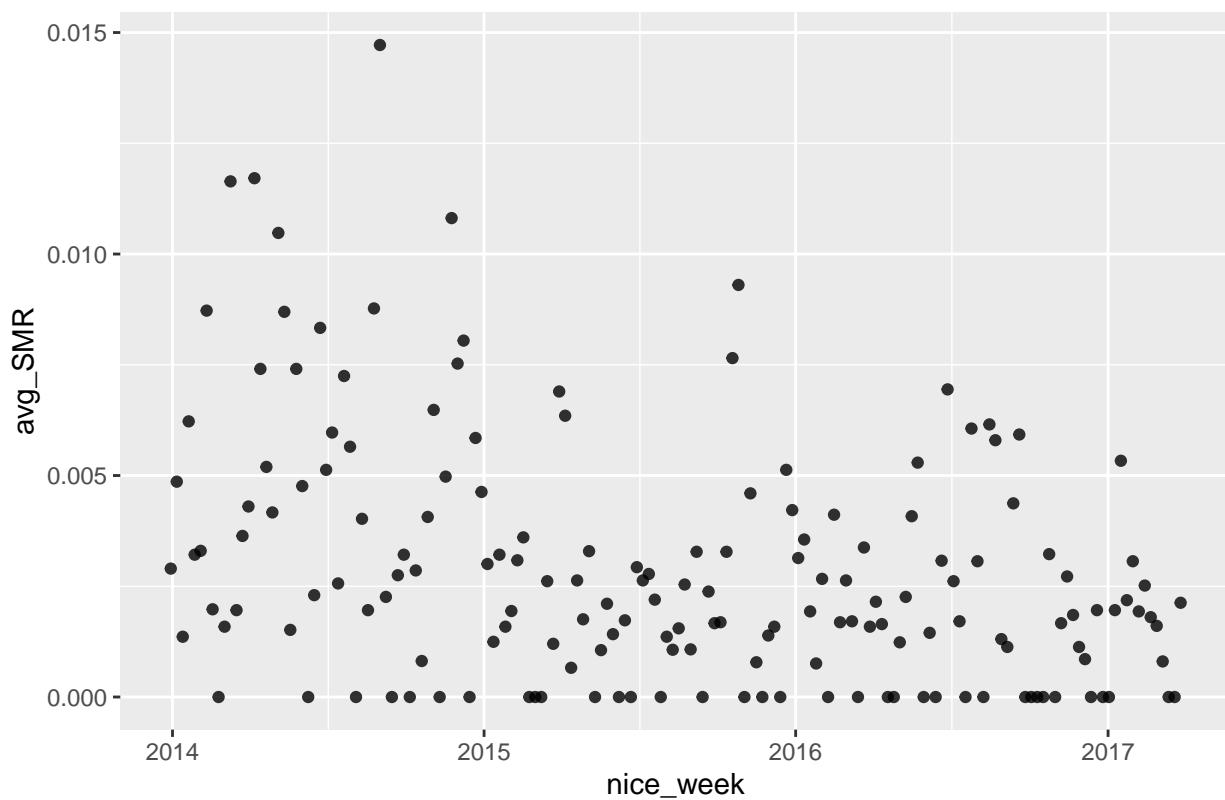
```
ggplot_date(date_df, "dataset")
```

```
## [1]
```

AZgilbert weekly SMR over time

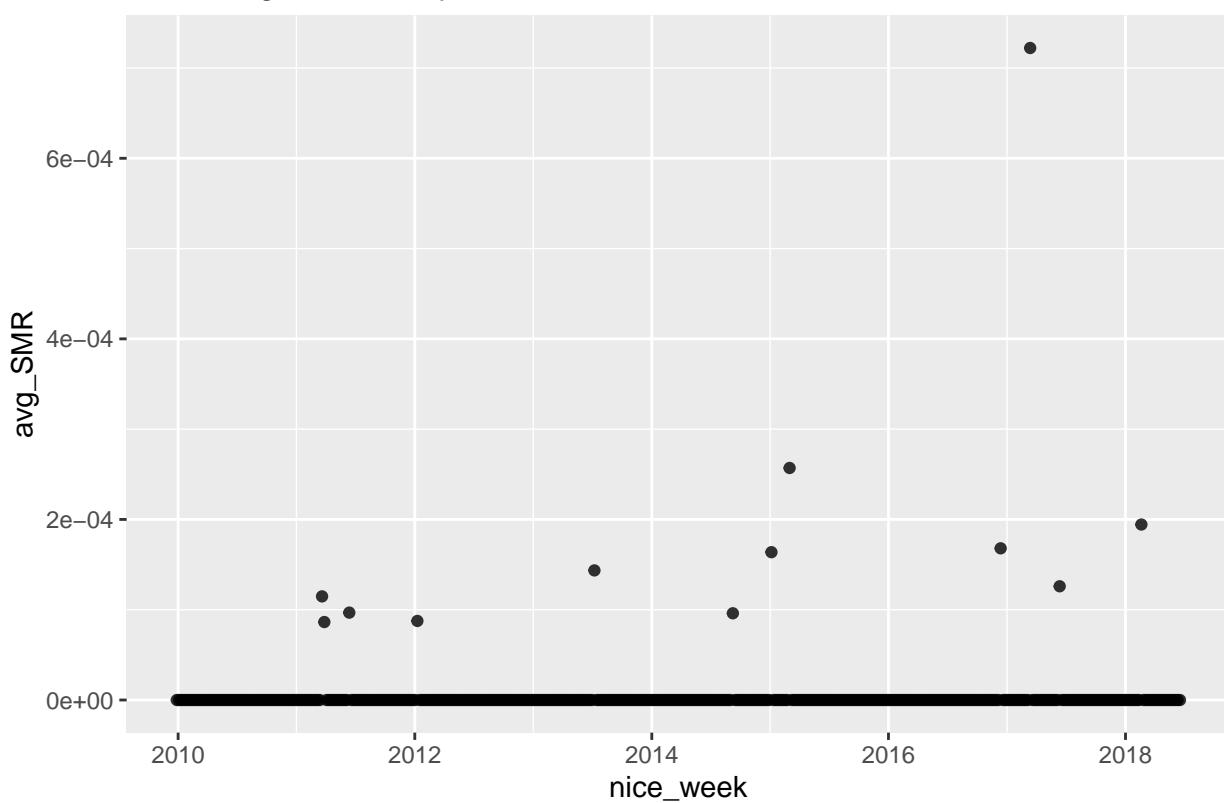


AZmesa weekly SMR over time



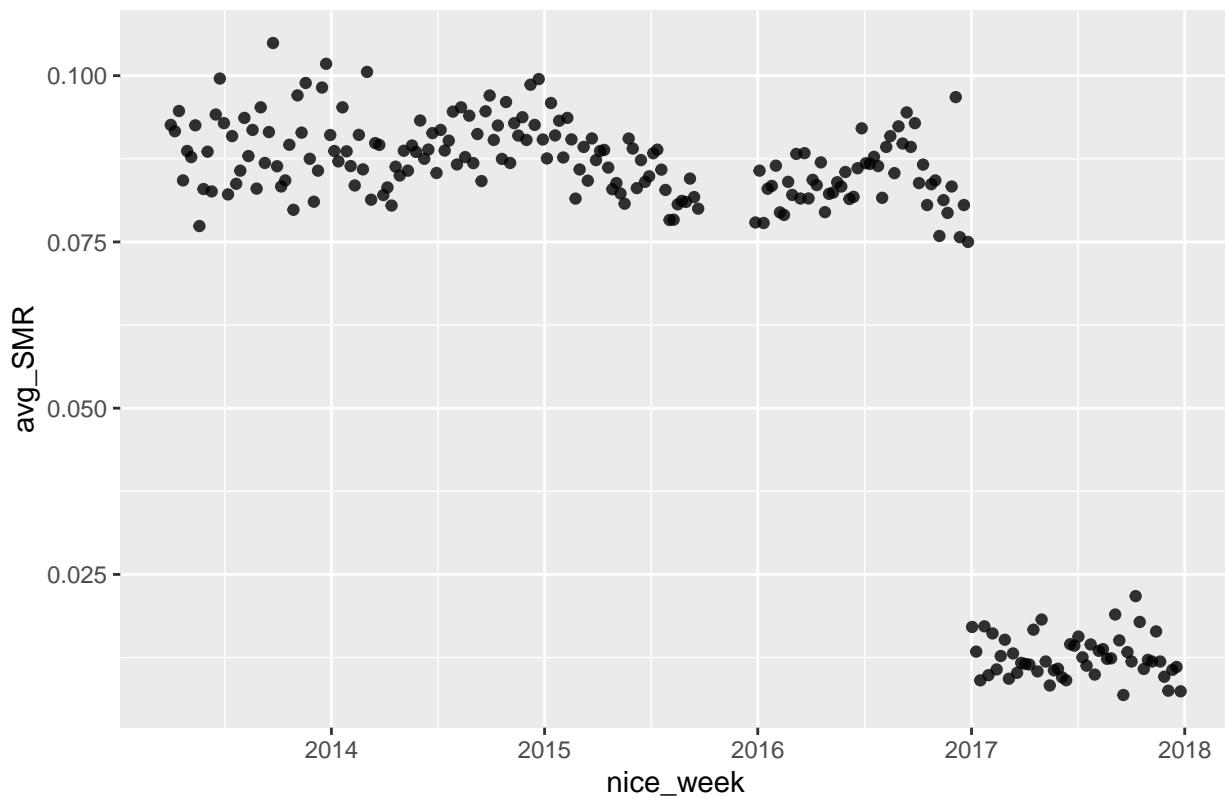
```
##  
## [[3]]
```

CAlosangeles weekly SMR over time



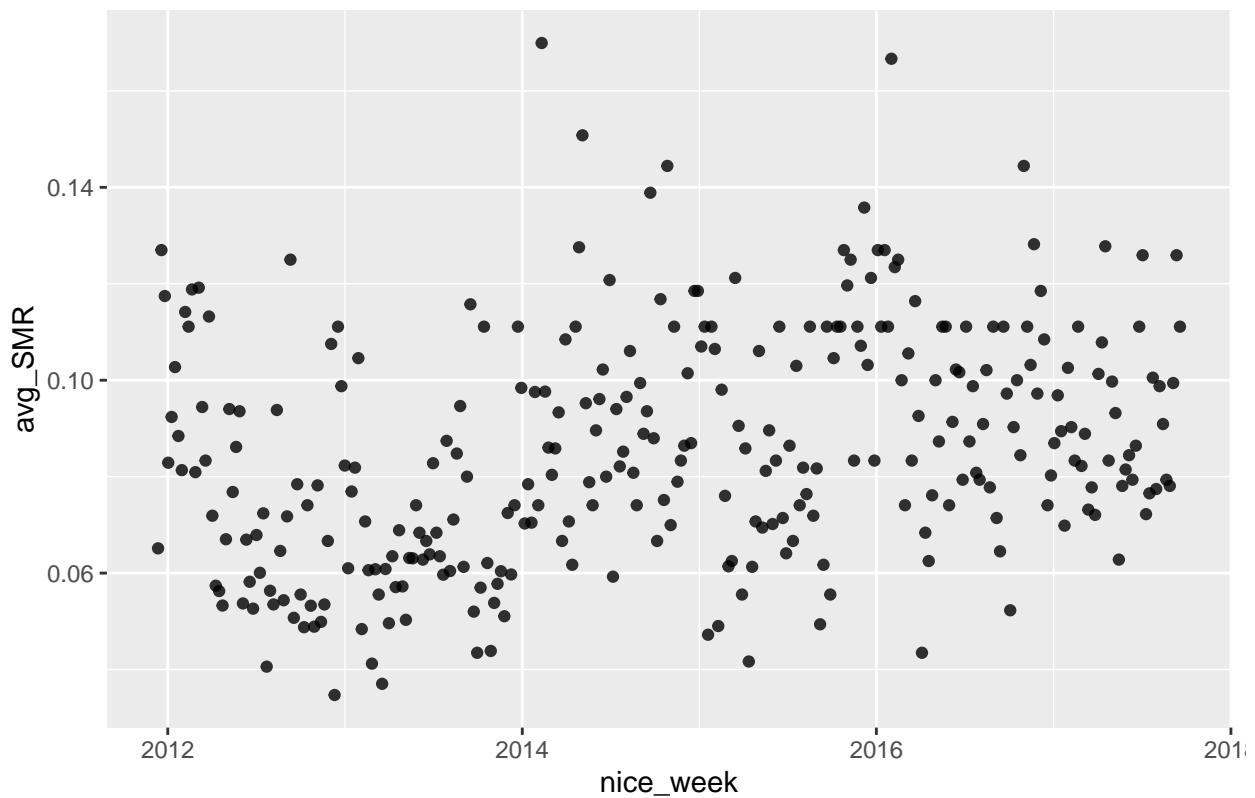
```
##  
## [[4]]
```

CAoakland weekly SMR over time



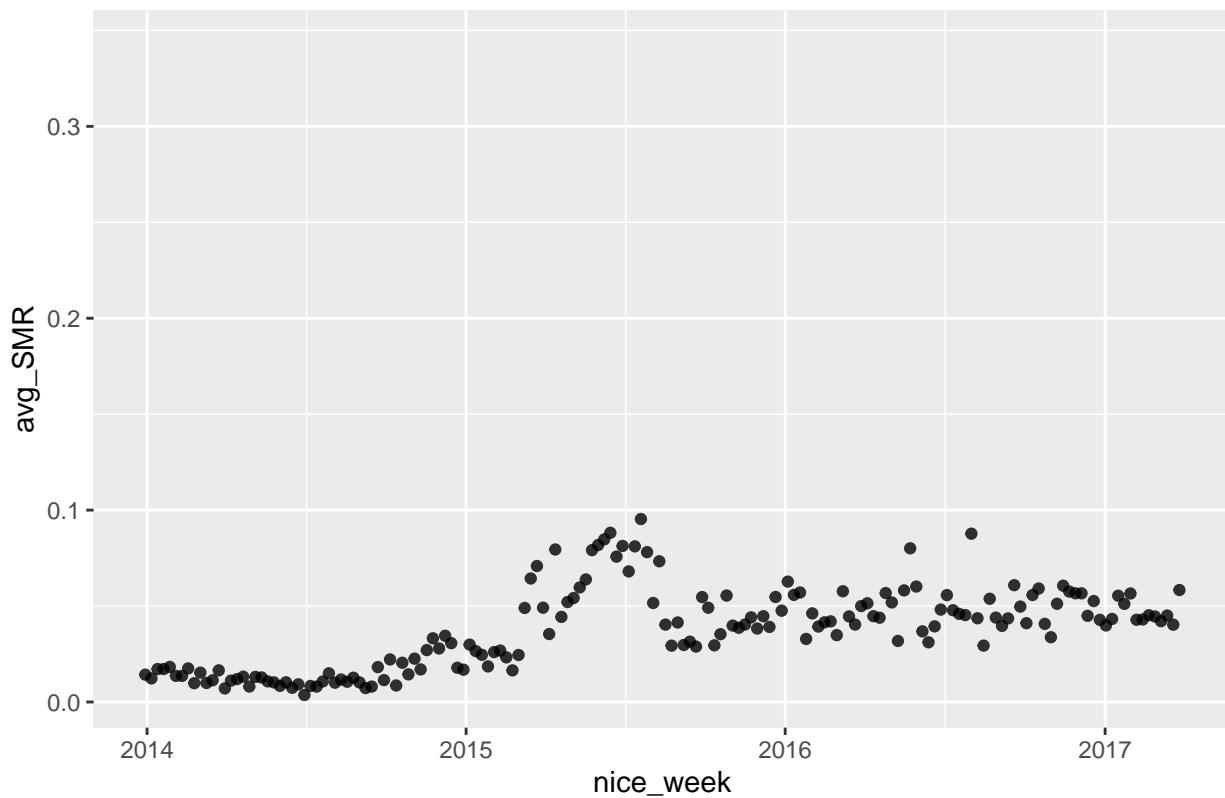
```
##  
## [[5]]
```

CAsanbernardino weekly SMR over time



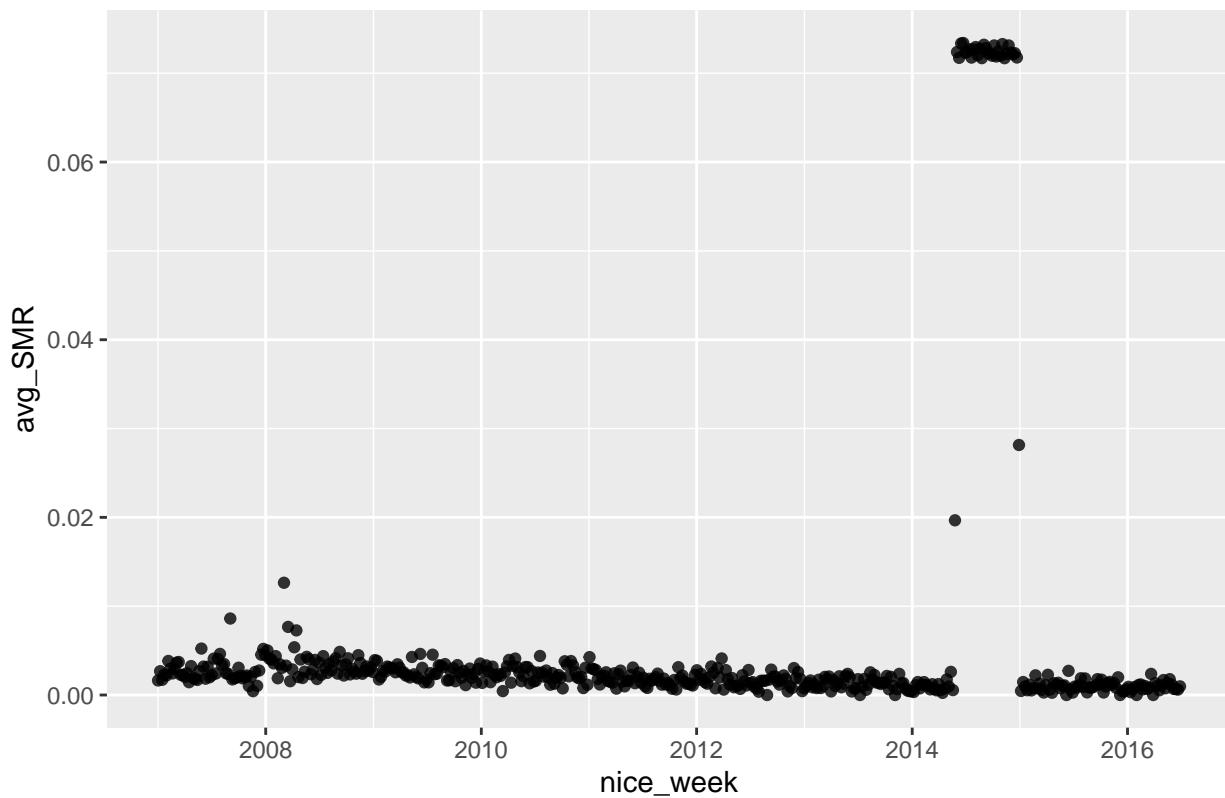
```
##  
## [[6]]
```

CAsandiego weekly SMR over time



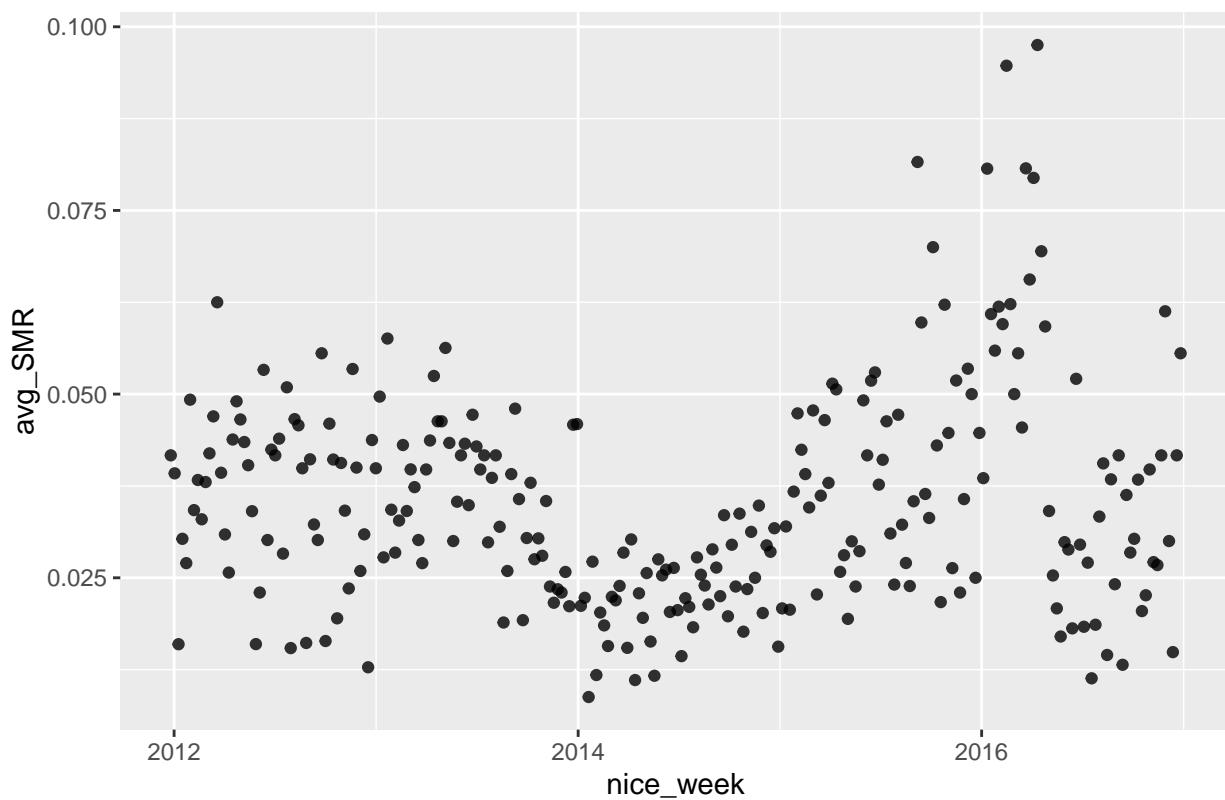
```
##  
## [[7]]
```

CAsanfrancisco weekly SMR over time



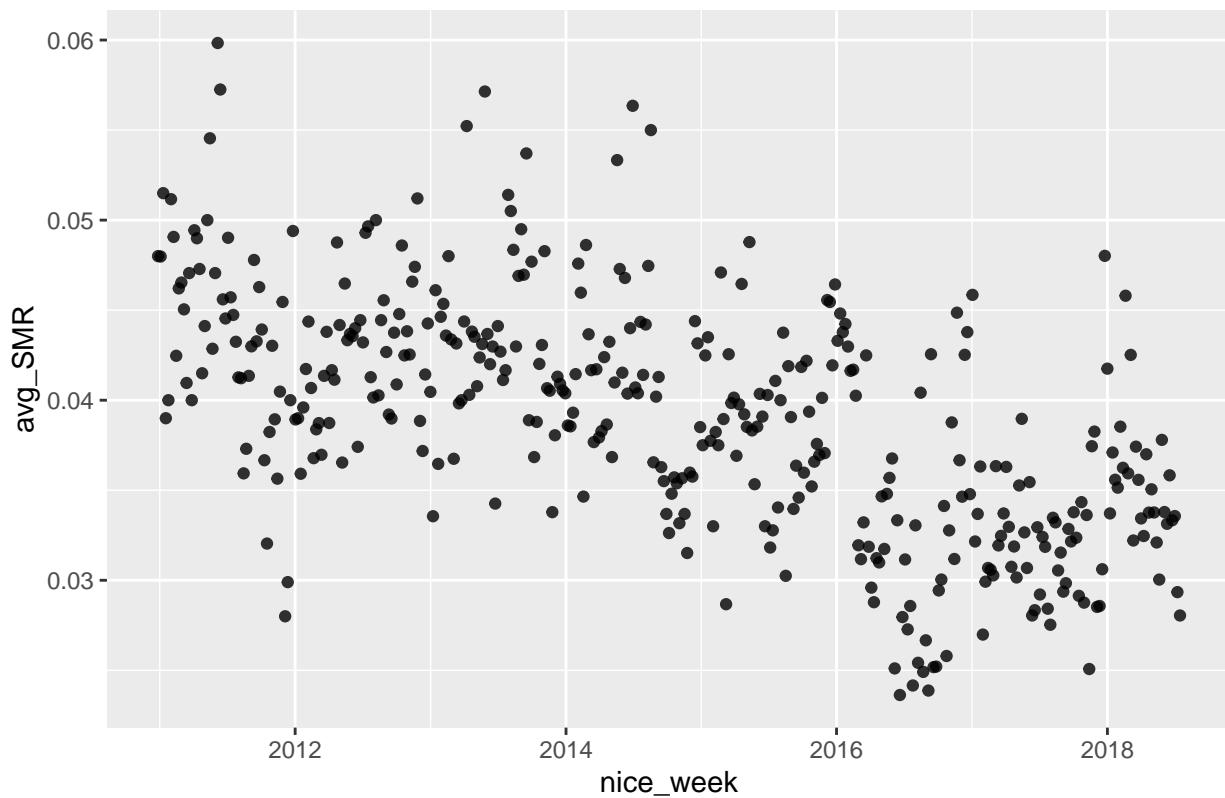
```
##  
## [[8]]
```

COaurora weekly SMR over time



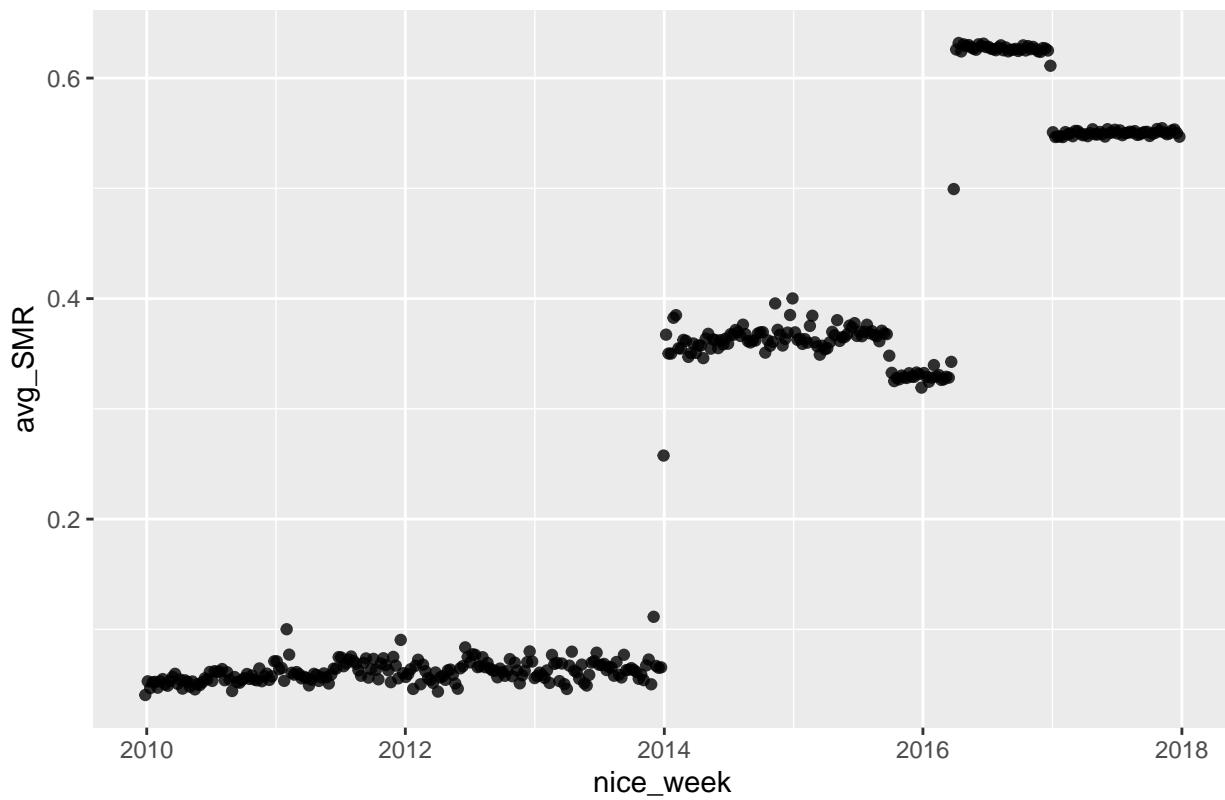
```
##  
## [[9]]
```

COdenver weekly SMR over time



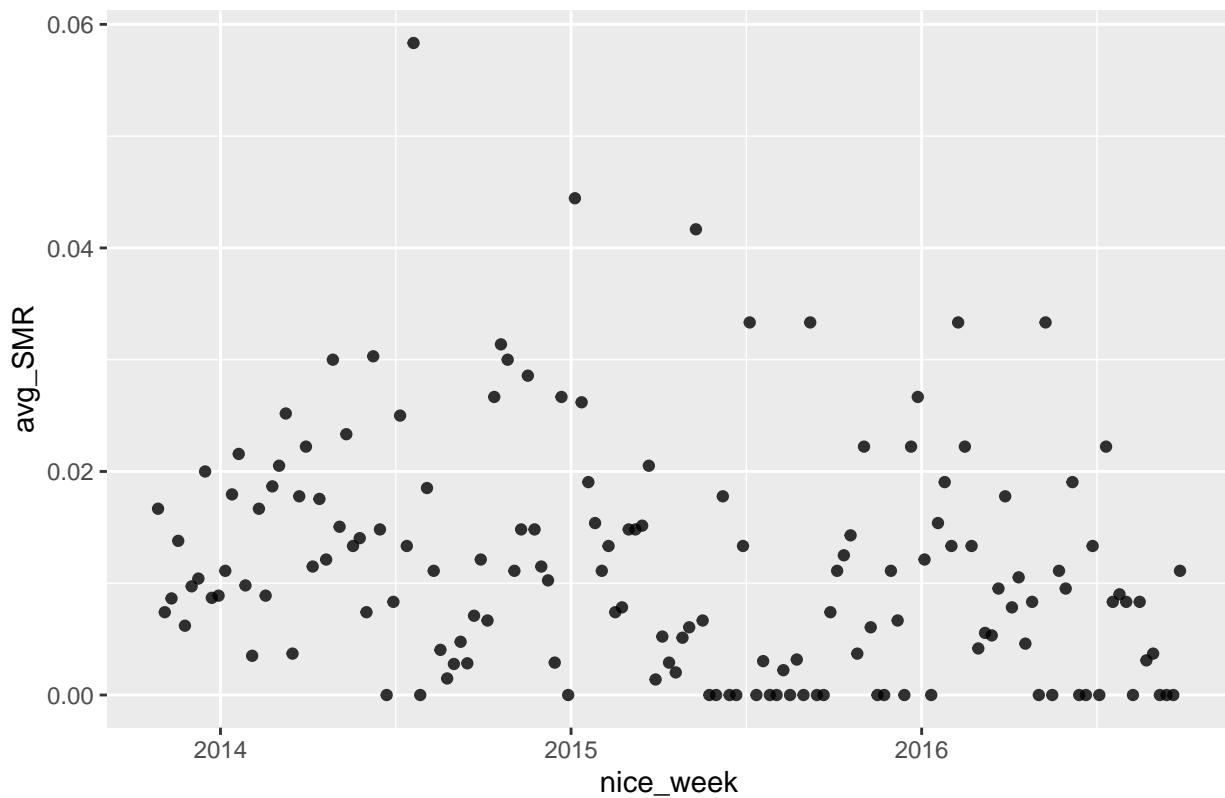
```
##  
## [[10]]
```

COstatewide weekly SMR over time



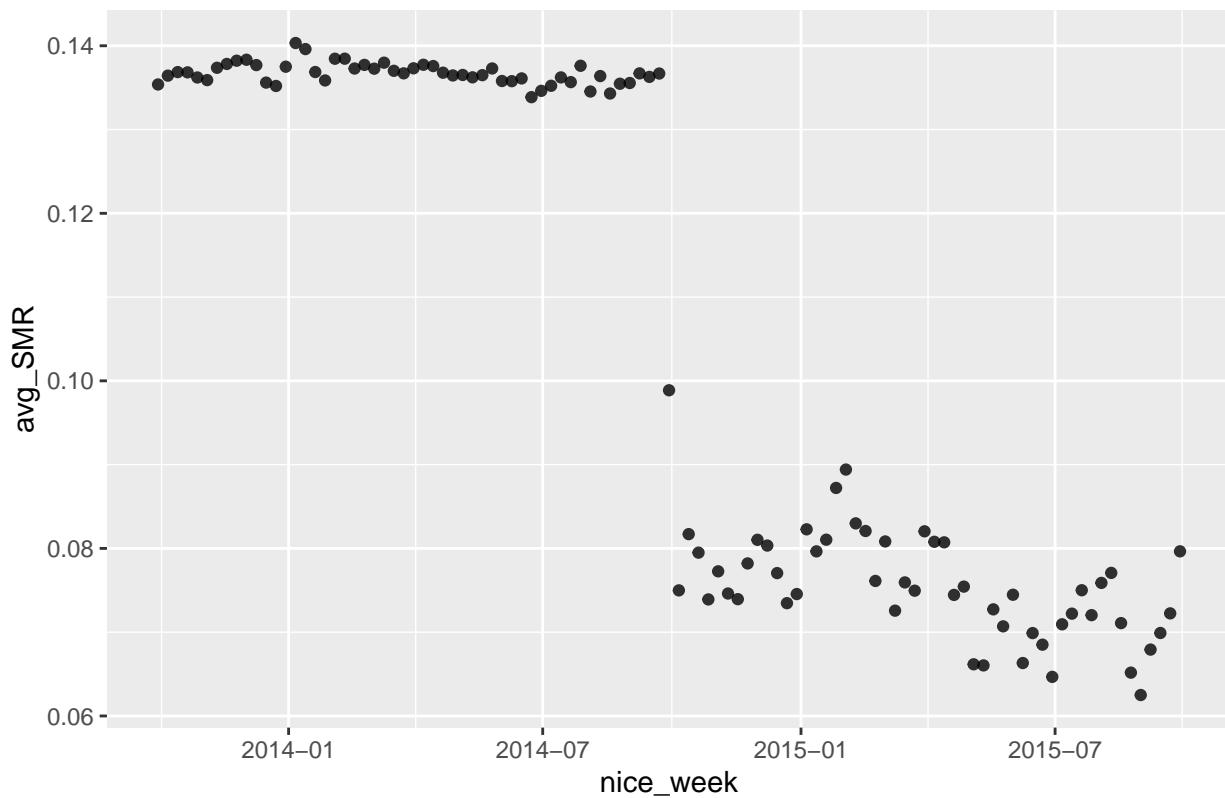
```
##  
## [[11]]
```

CThartford weekly SMR over time



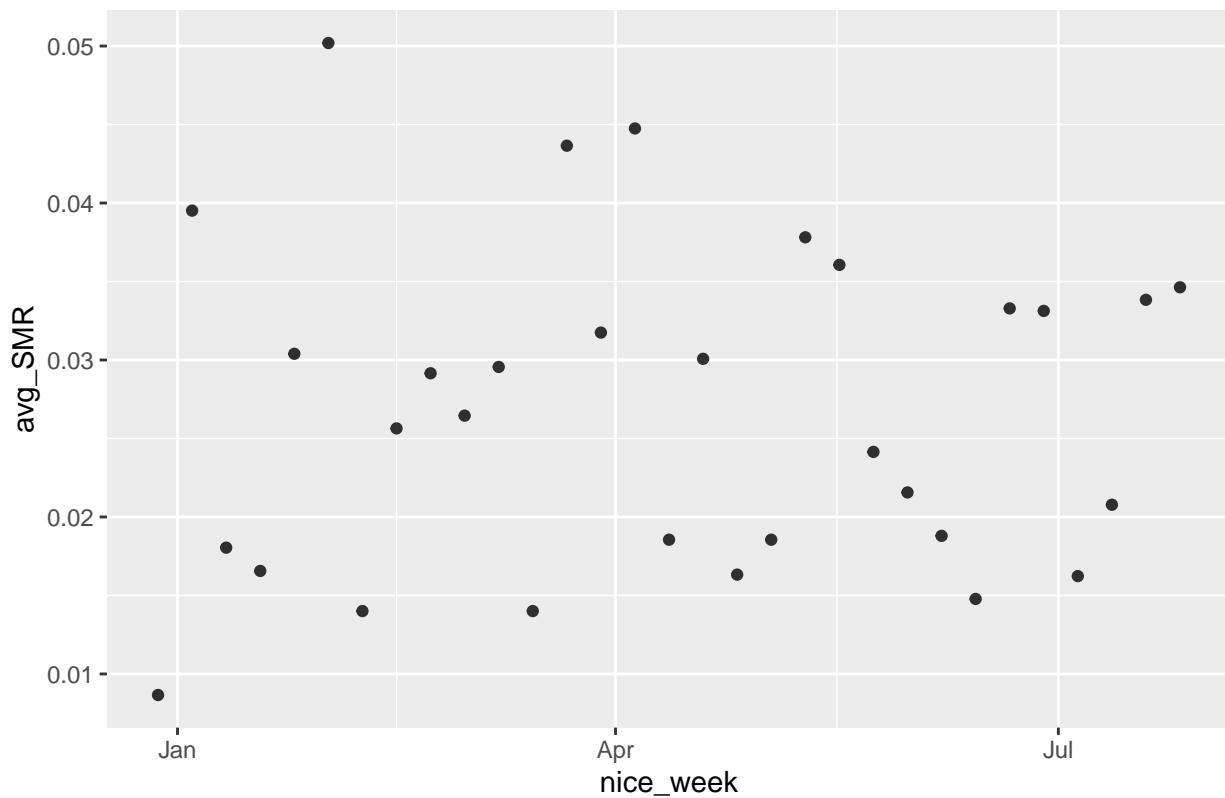
```
##  
## [[12]]
```

CT statewide weekly SMR over time



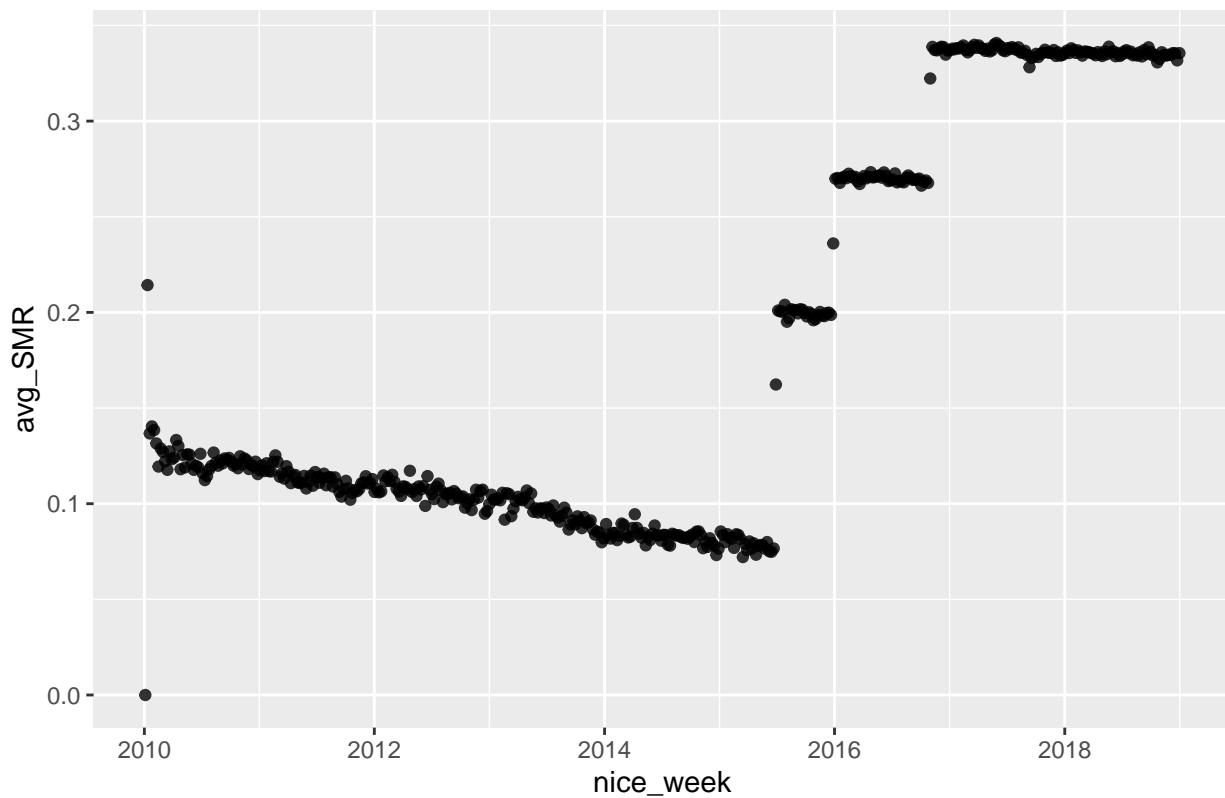
```
##  
## [[13]]
```

FLsaint weekly SMR over time



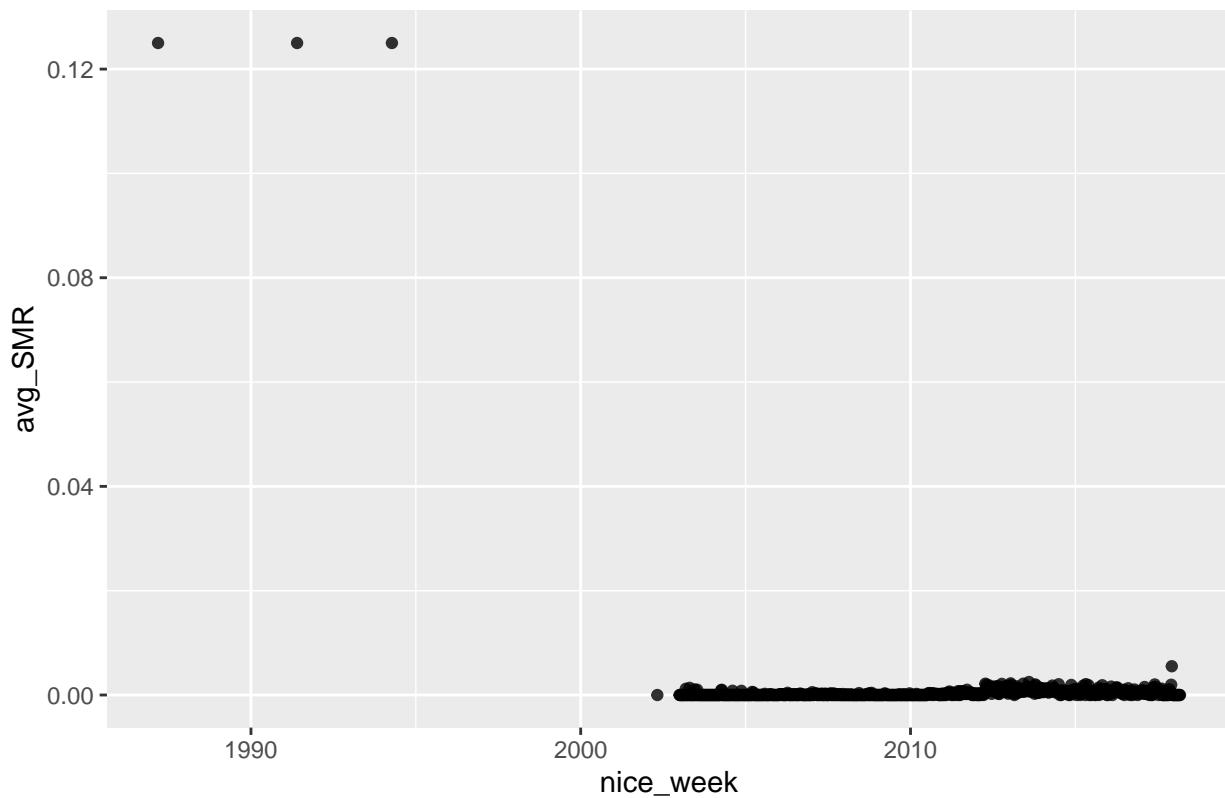
```
##  
## [[14]]
```

FLstatewide weekly SMR over time



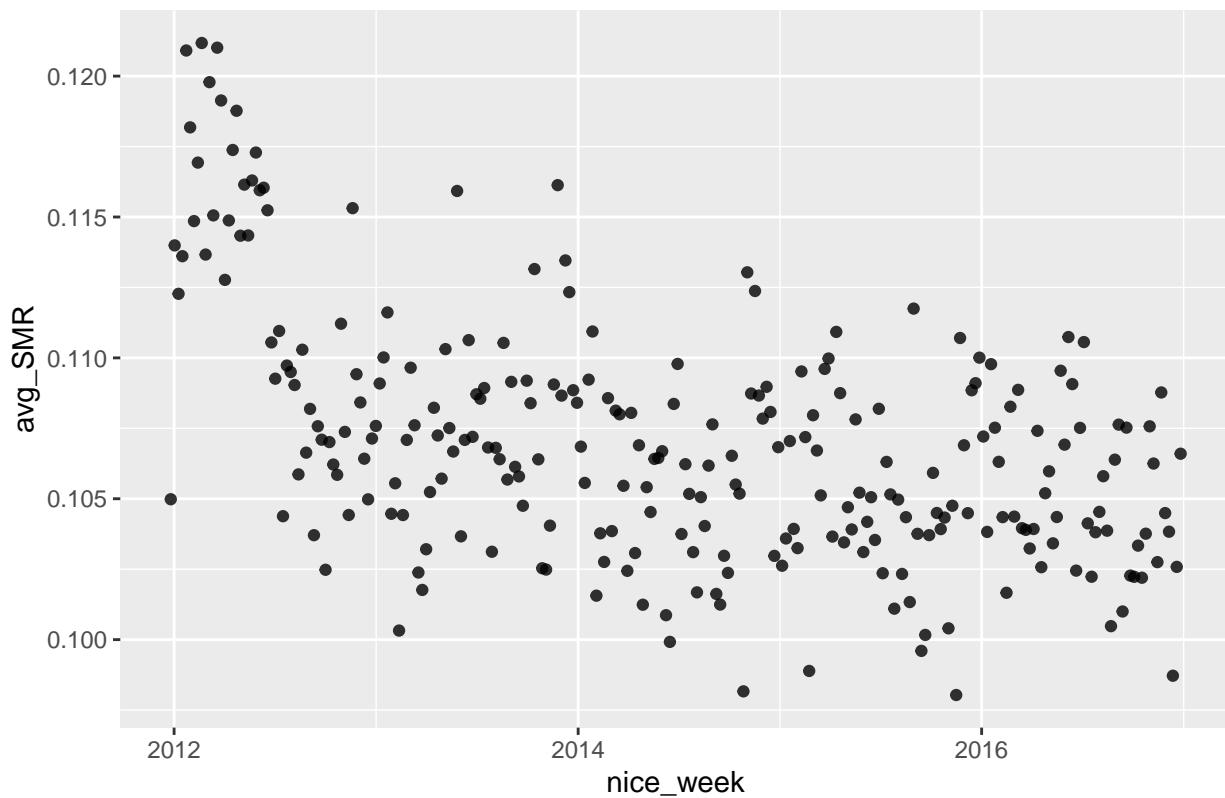
```
##  
## [[15]]
```

FLtampa weekly SMR over time



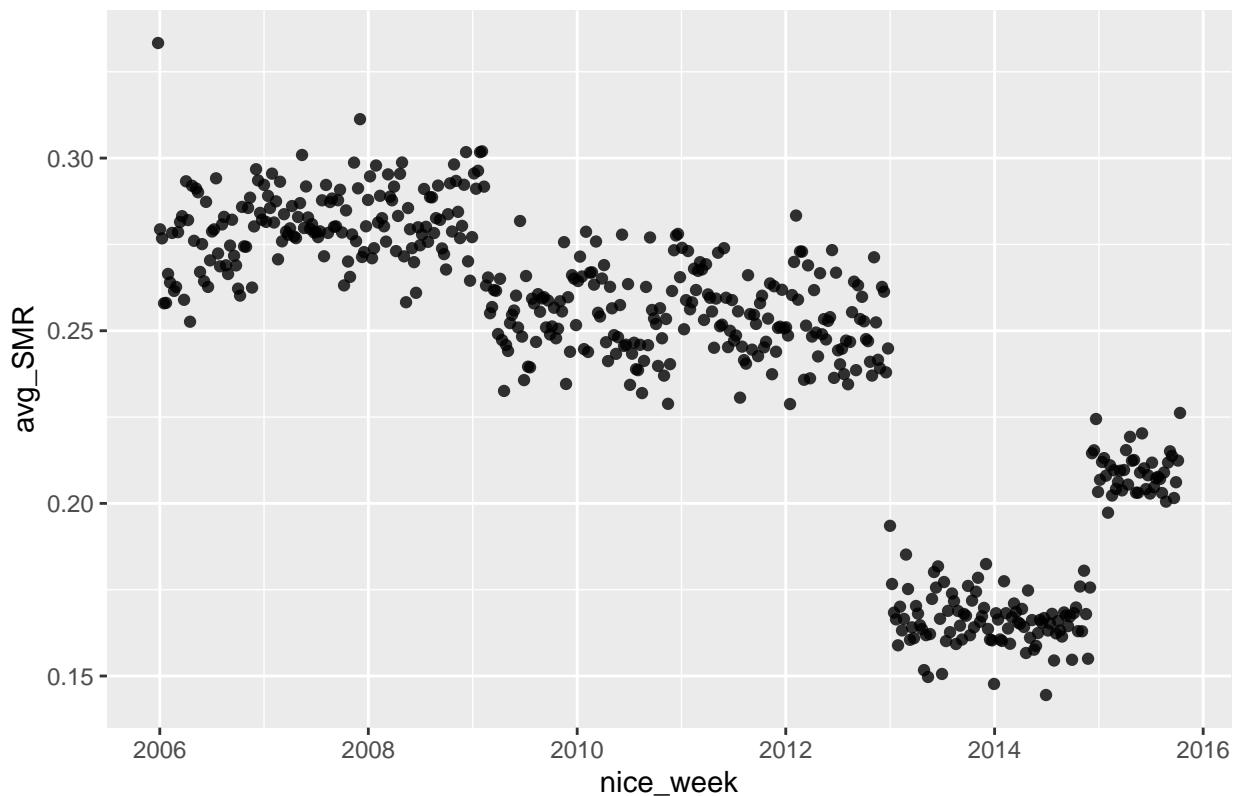
```
##  
## [[16]]
```

GA statewide weekly SMR over time



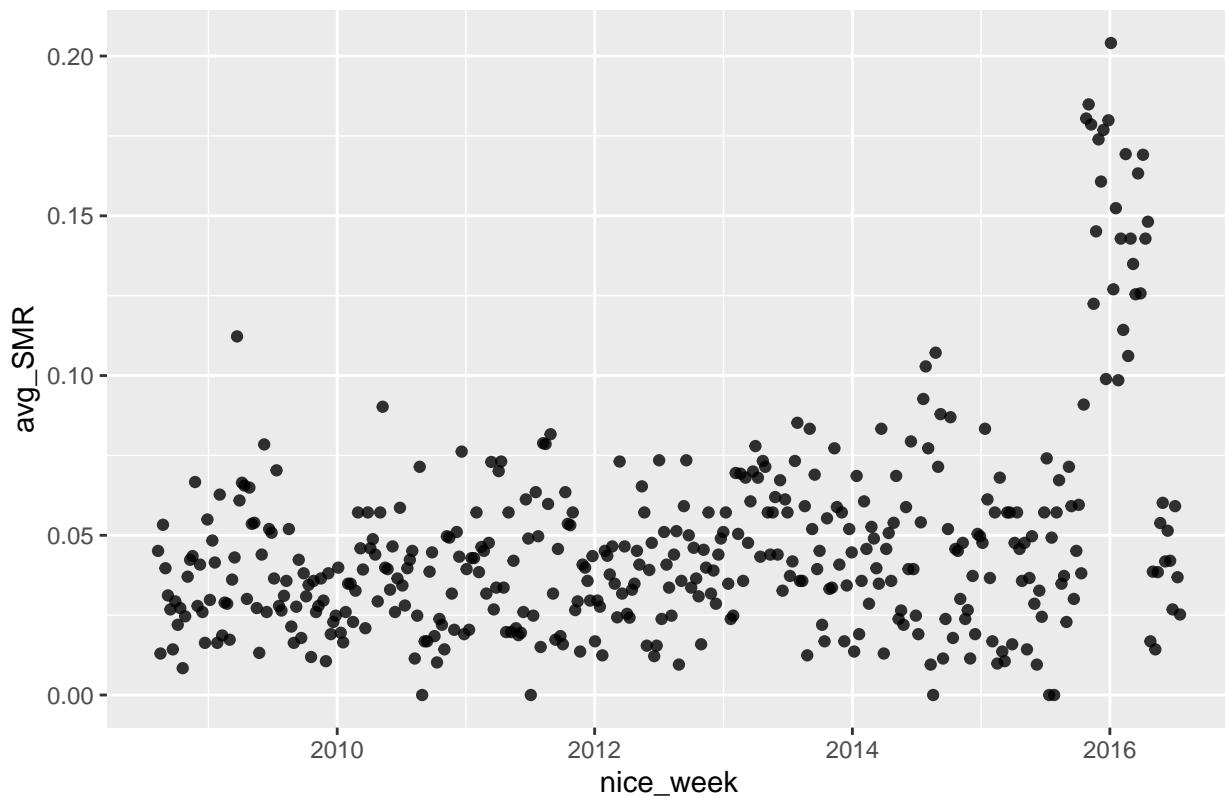
```
##  
## [[17]]
```

I Statewide weekly SMR over time



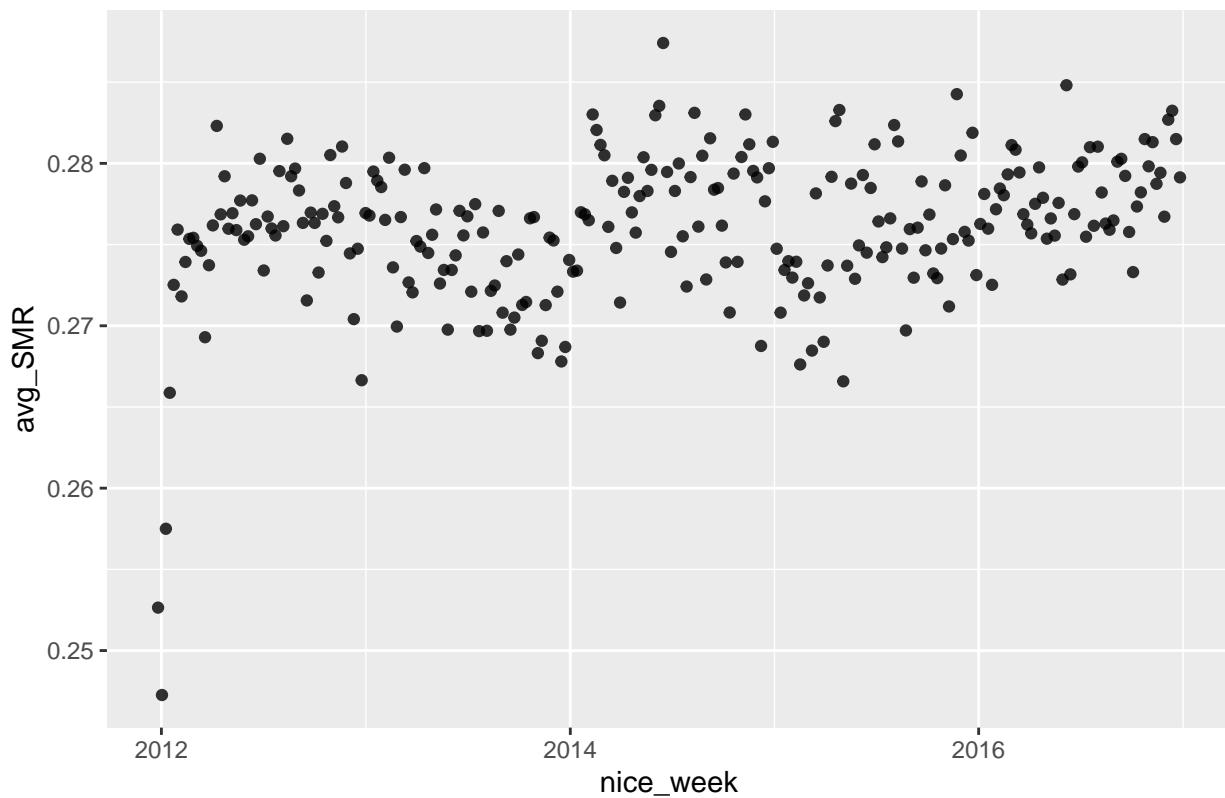
```
##  
## [[18]]
```

IDidahofalls weekly SMR over time



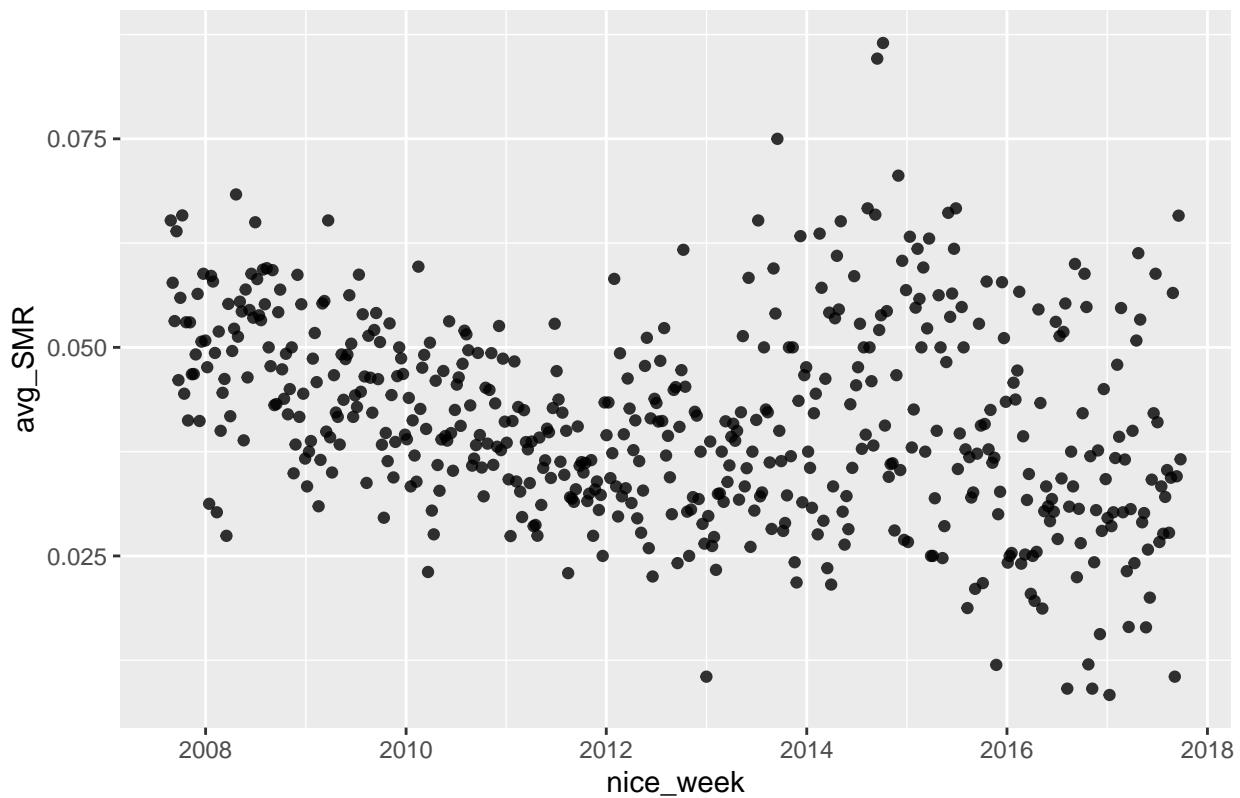
```
##  
## [[19]]
```

ILchicago weekly SMR over time



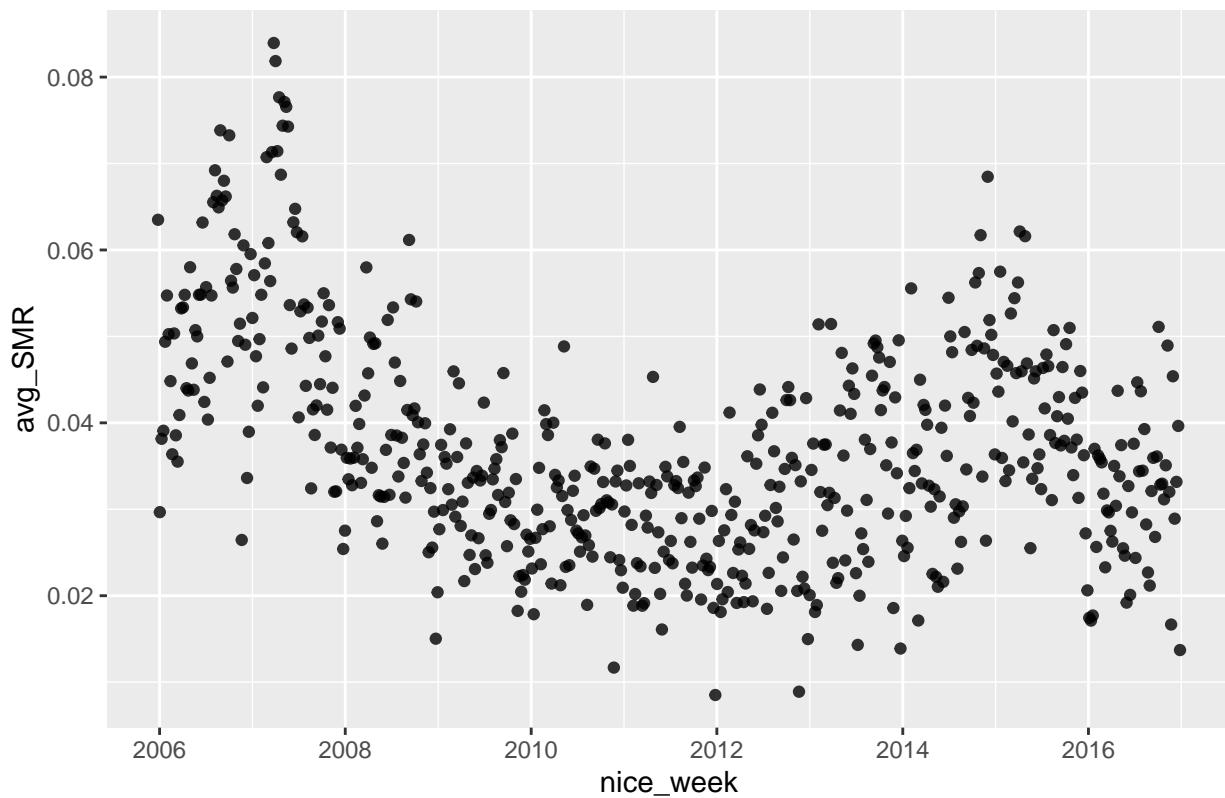
```
##  
## [[20]]
```

INfortwayne weekly SMR over time



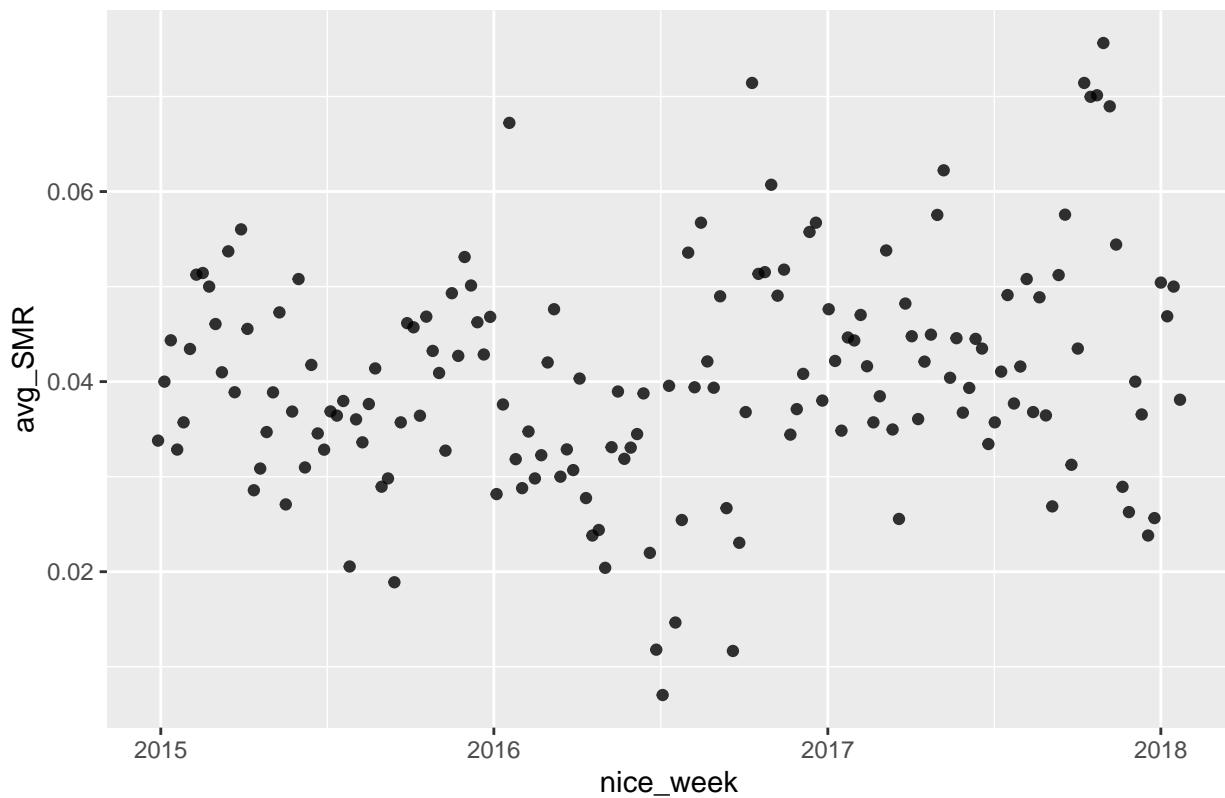
```
##  
## [[21]]
```

KSwitchita weekly SMR over time



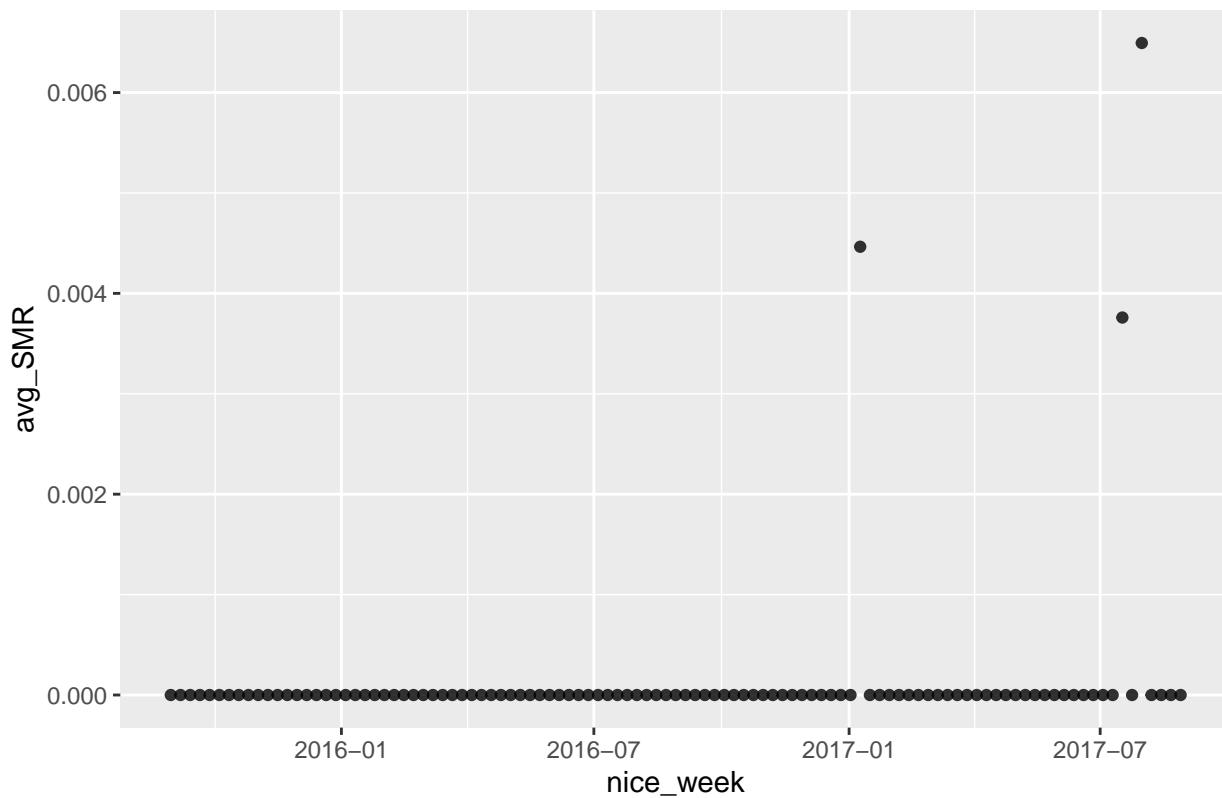
```
##  
## [[22]]
```

KYlouisville weekly SMR over time

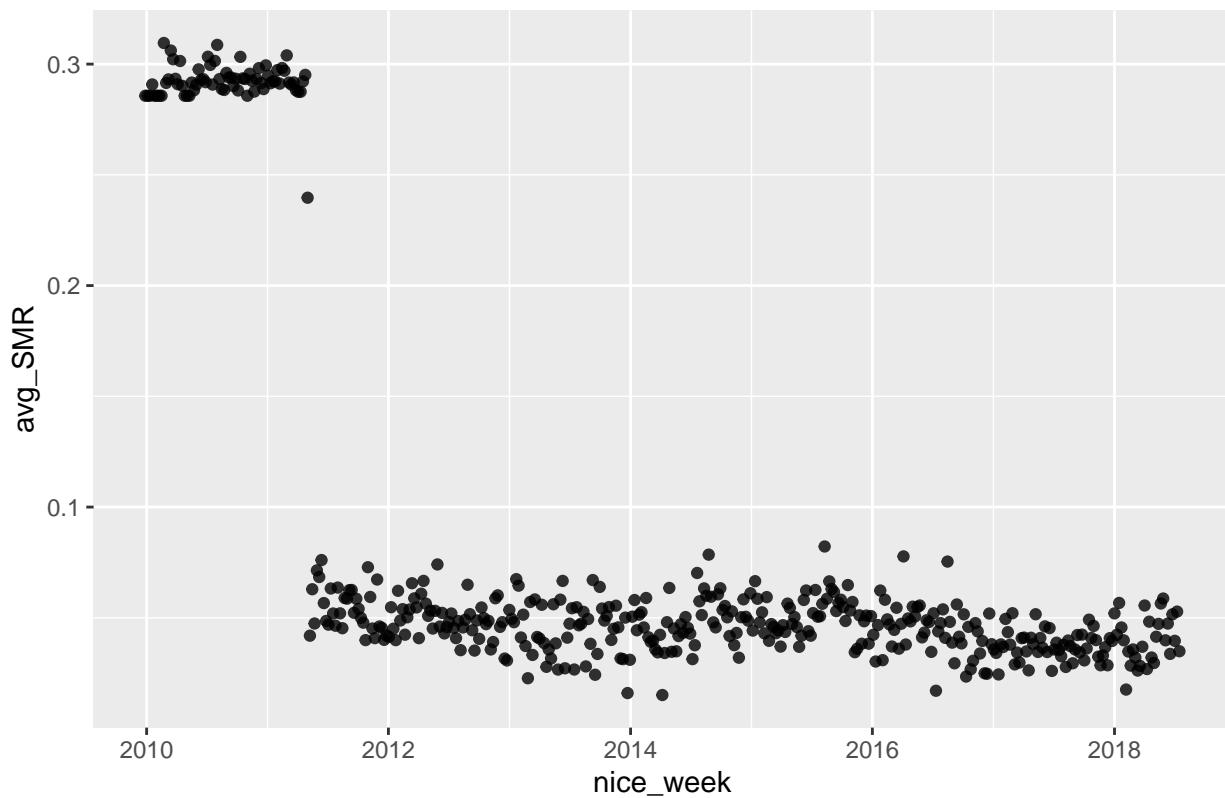


```
##  
## [[23]]
```

KYowensboro weekly SMR over time

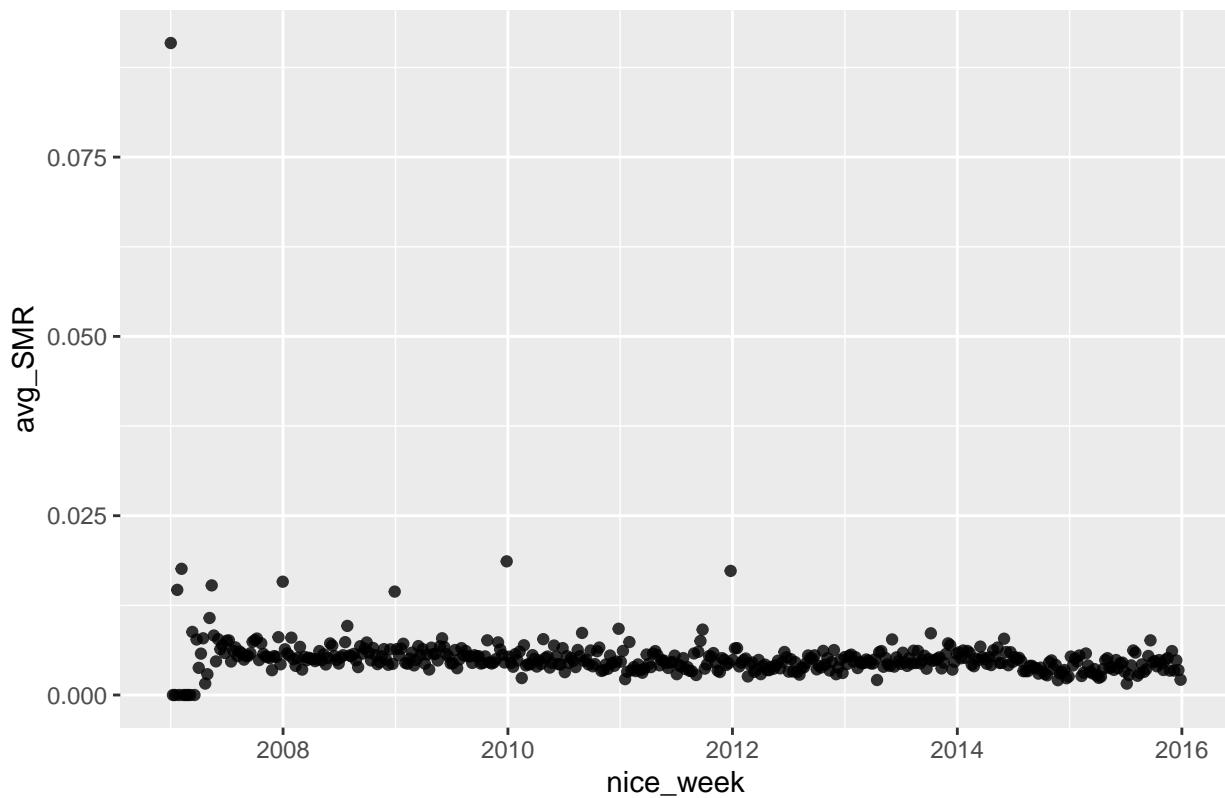


LAneworleans weekly SMR over time



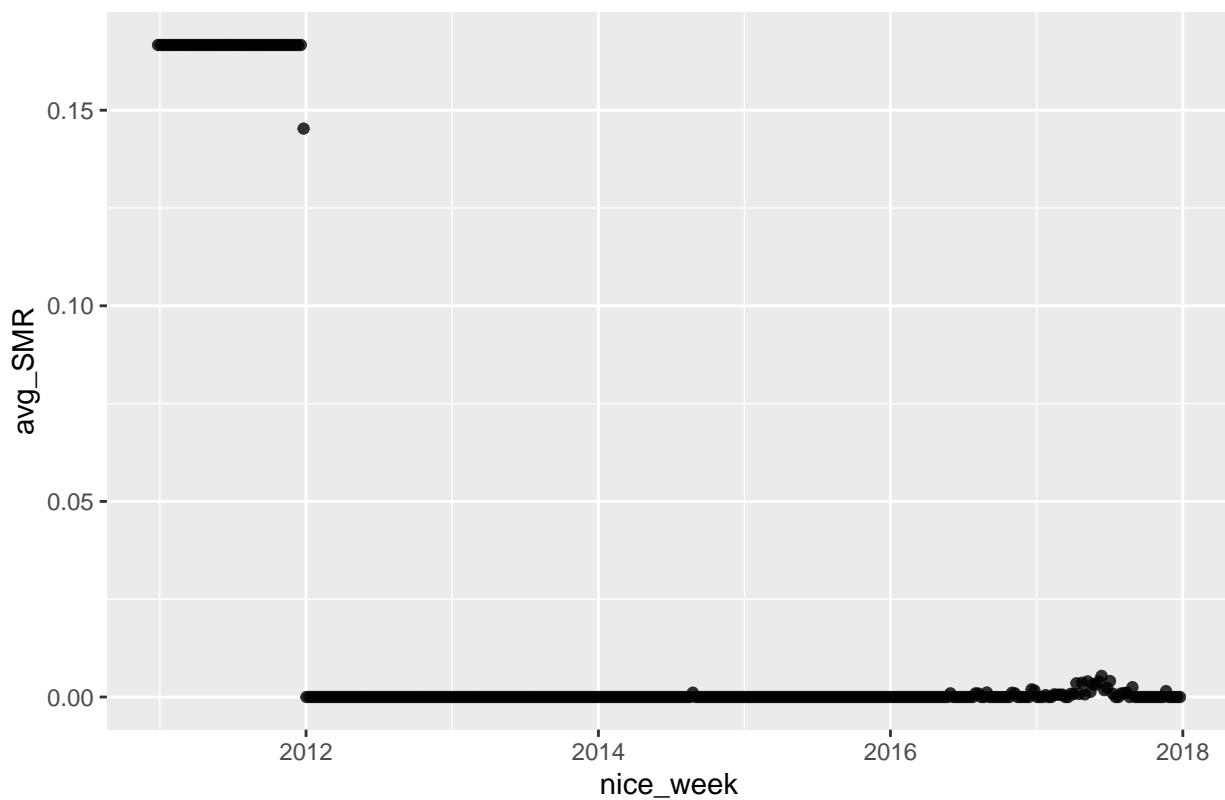
```
##  
## [[25]]
```

MAstatewide weekly SMR over time



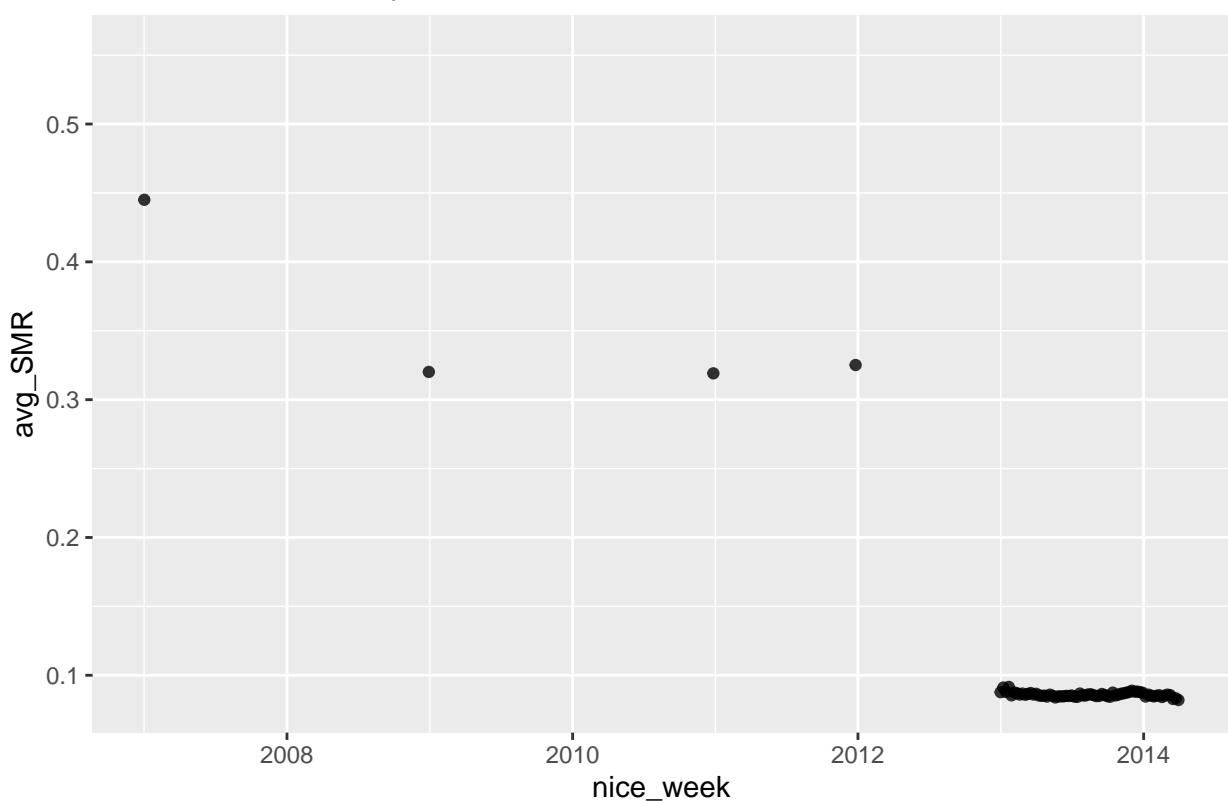
```
##  
## [[26]]
```

MDbaltimore weekly SMR over time



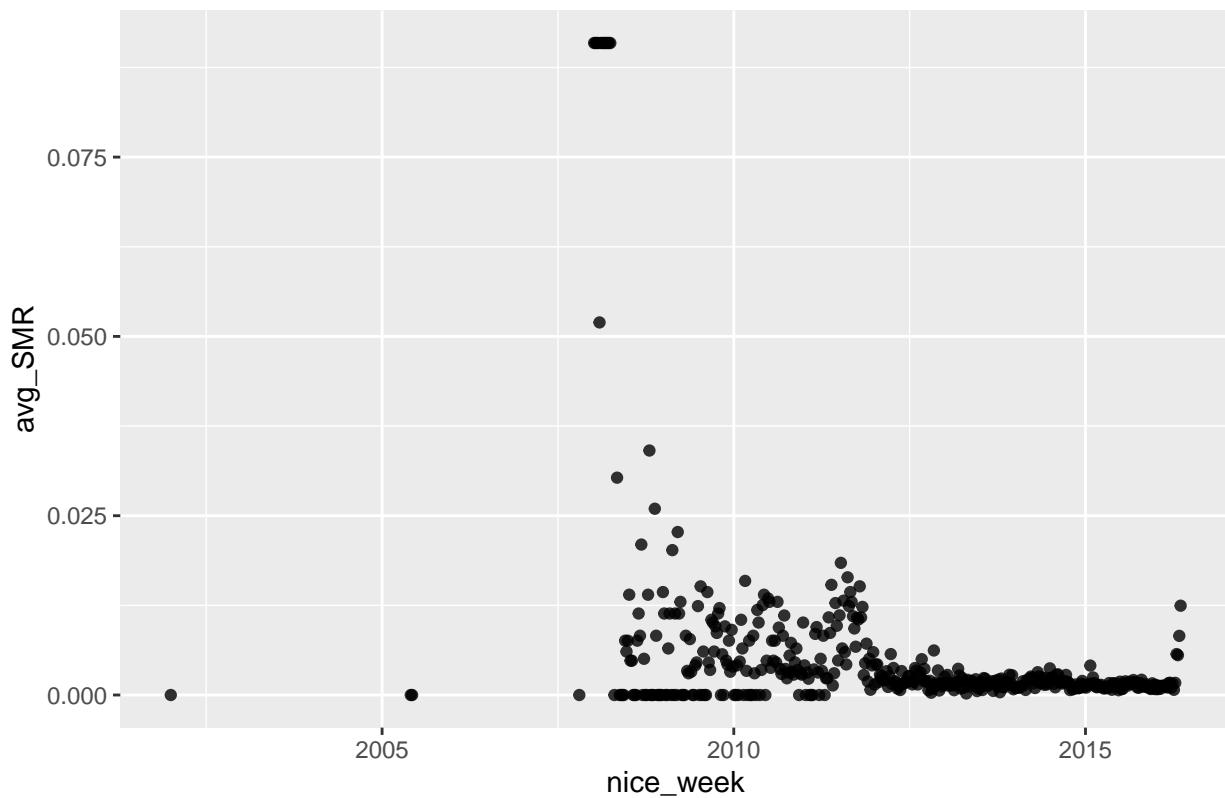
```
##  
## [[27]]
```

MDstatewide weekly SMR over time



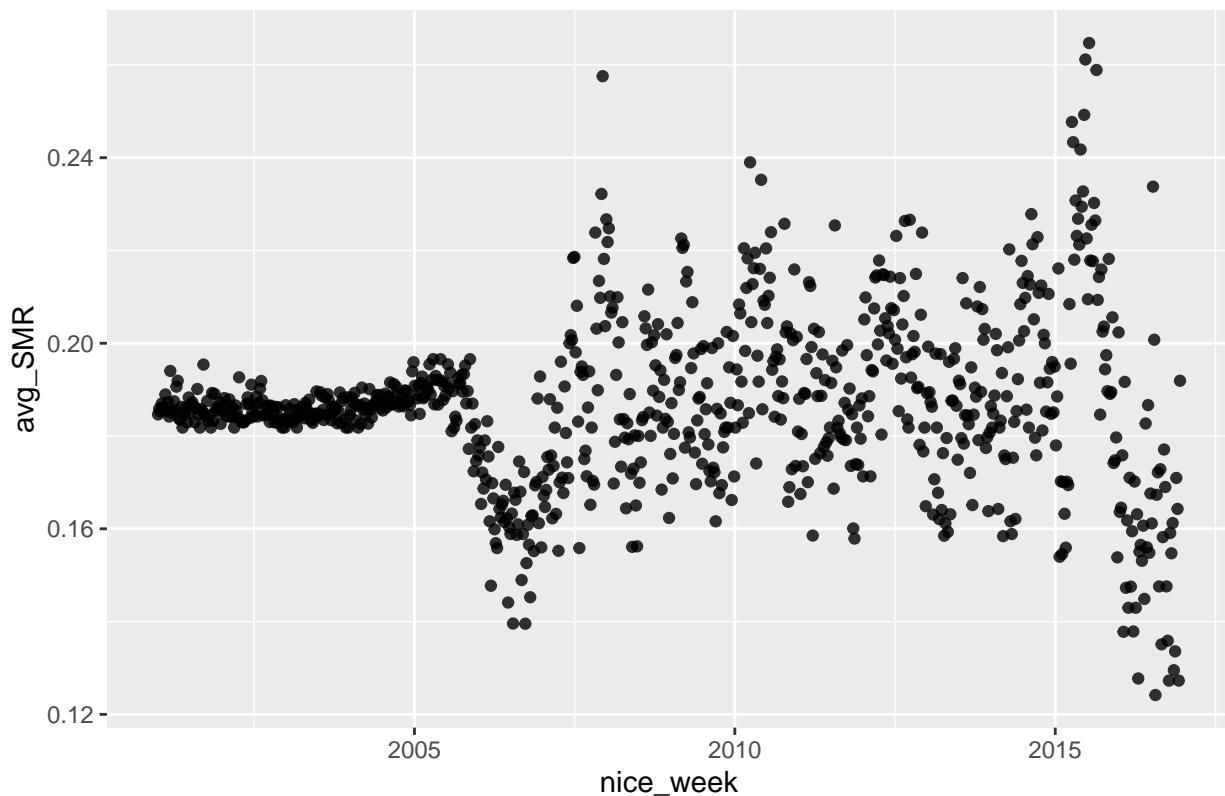
```
##  
## [ [28] ]
```

MI statewide weekly SMR over time



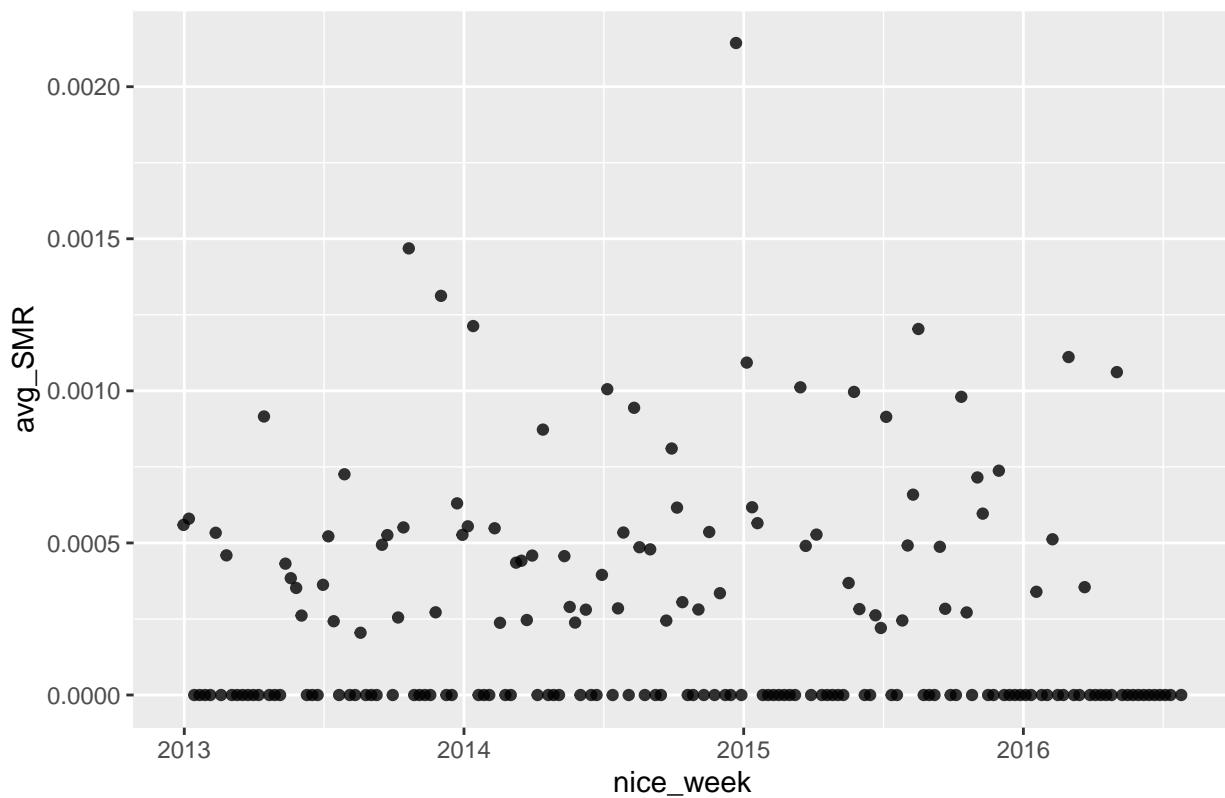
```
##  
## [[29]]
```

MNsaintpaul weekly SMR over time



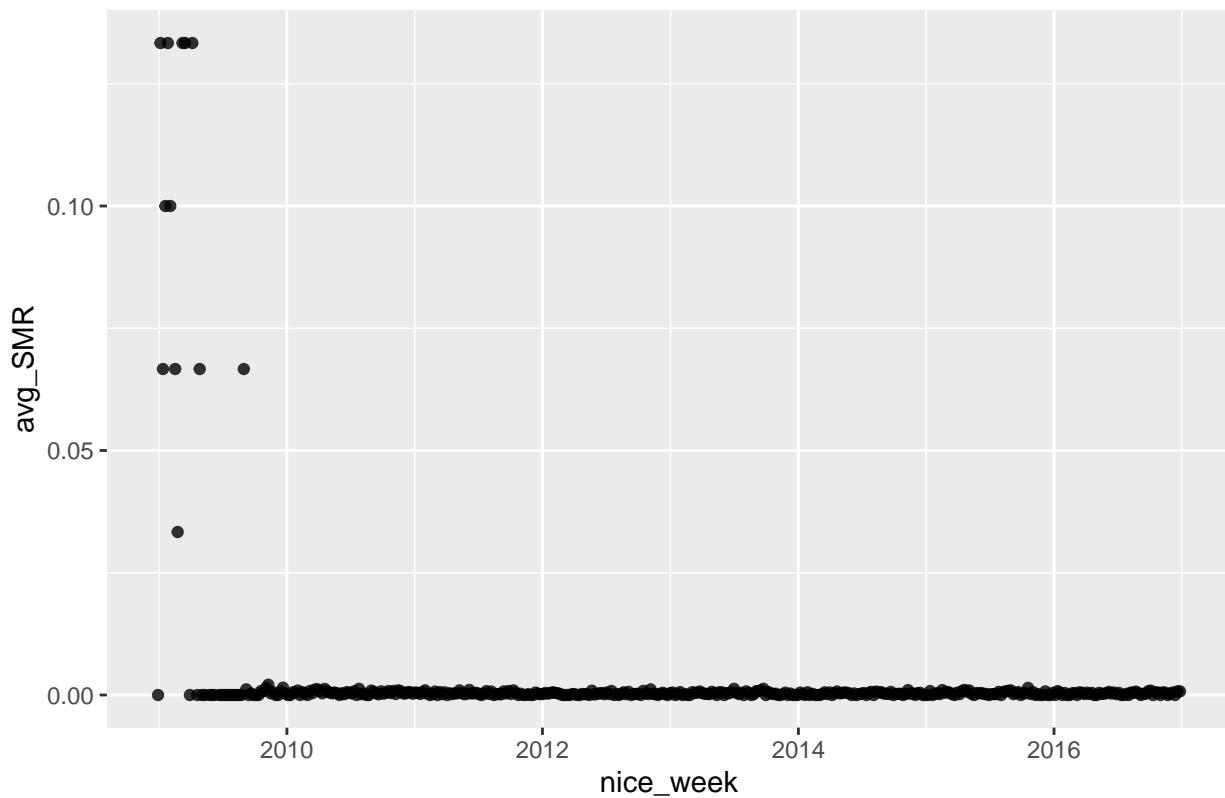
```
##  
## [[30]]
```

MSstatewide weekly SMR over time

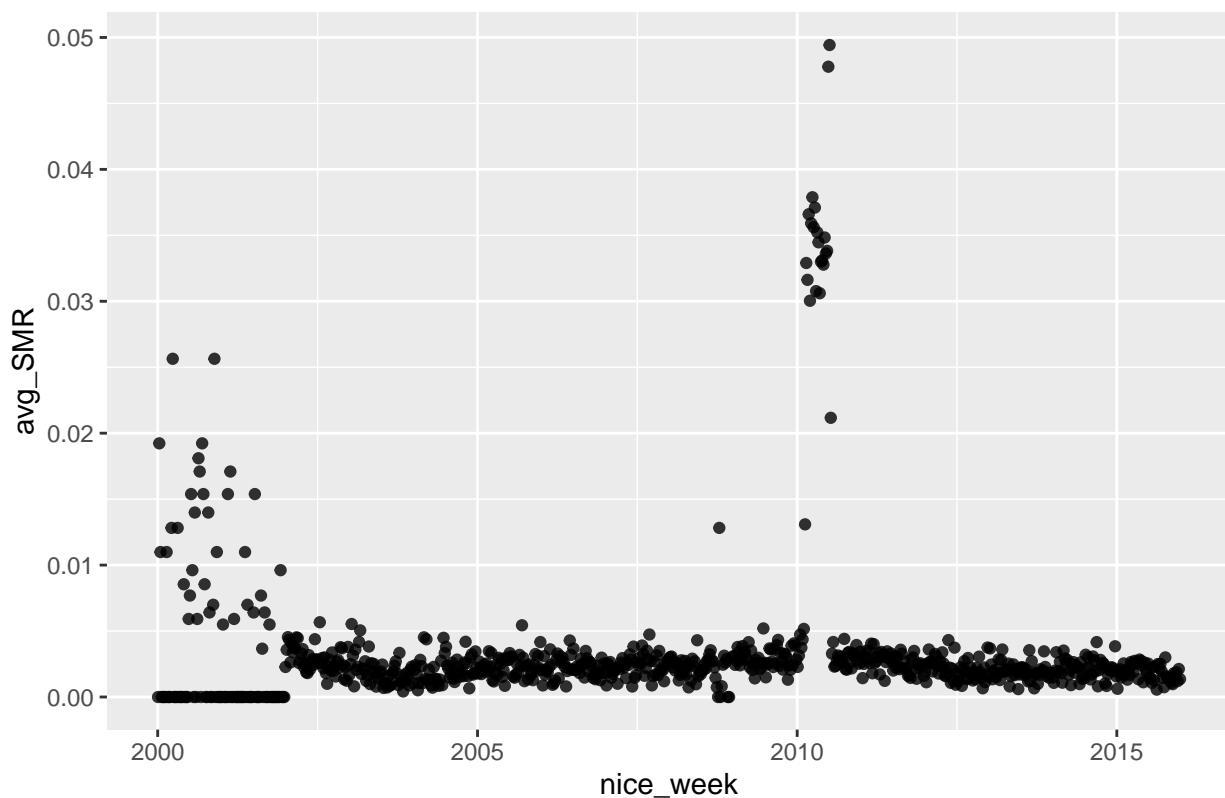


```
##  
## [[31]]
```

MTstatewide weekly SMR over time

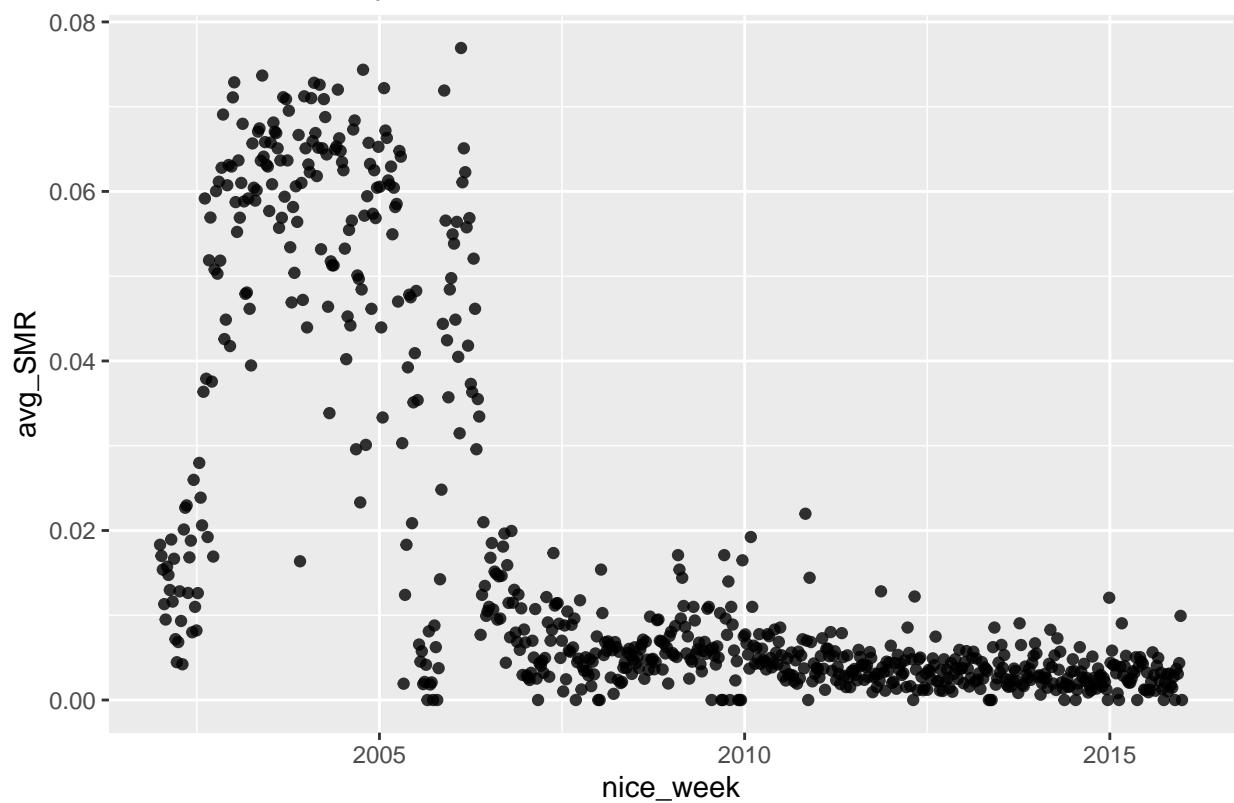


NCcharlotte weekly SMR over time



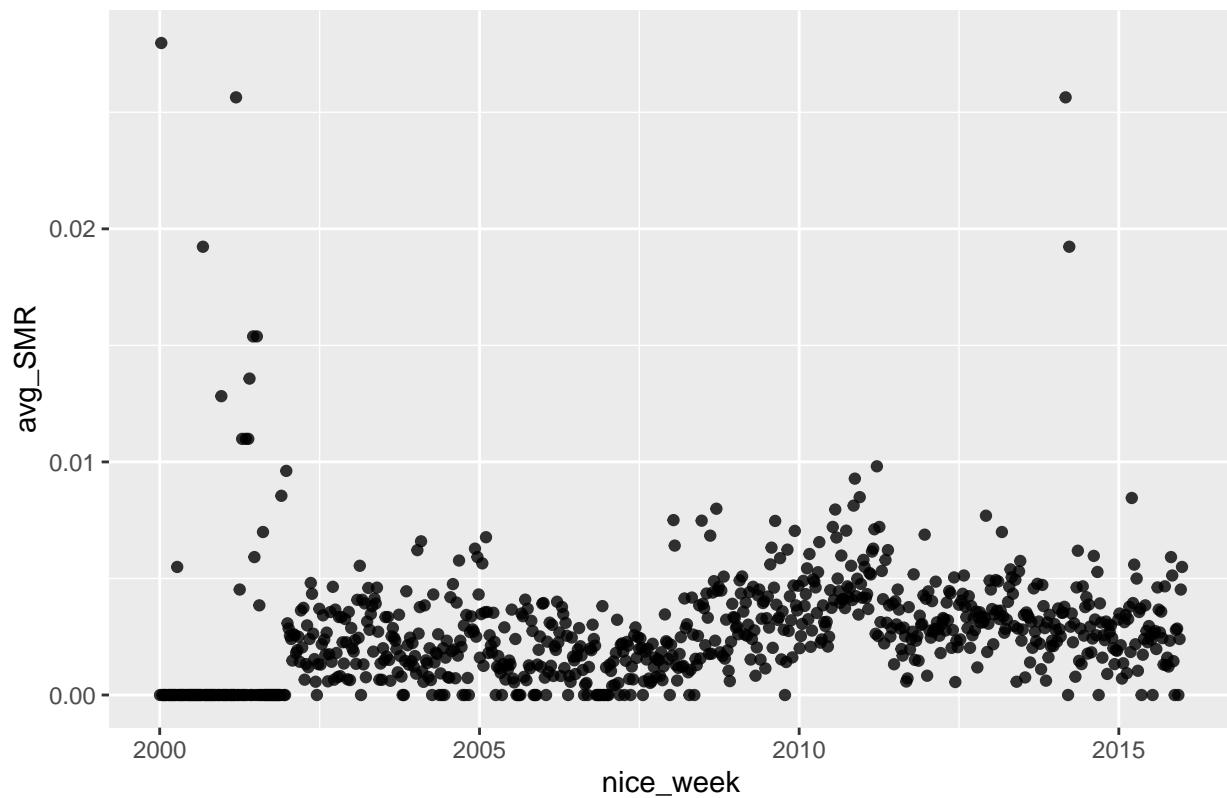
```
##  
## [[33]]
```

NCdurham weekly SMR over time



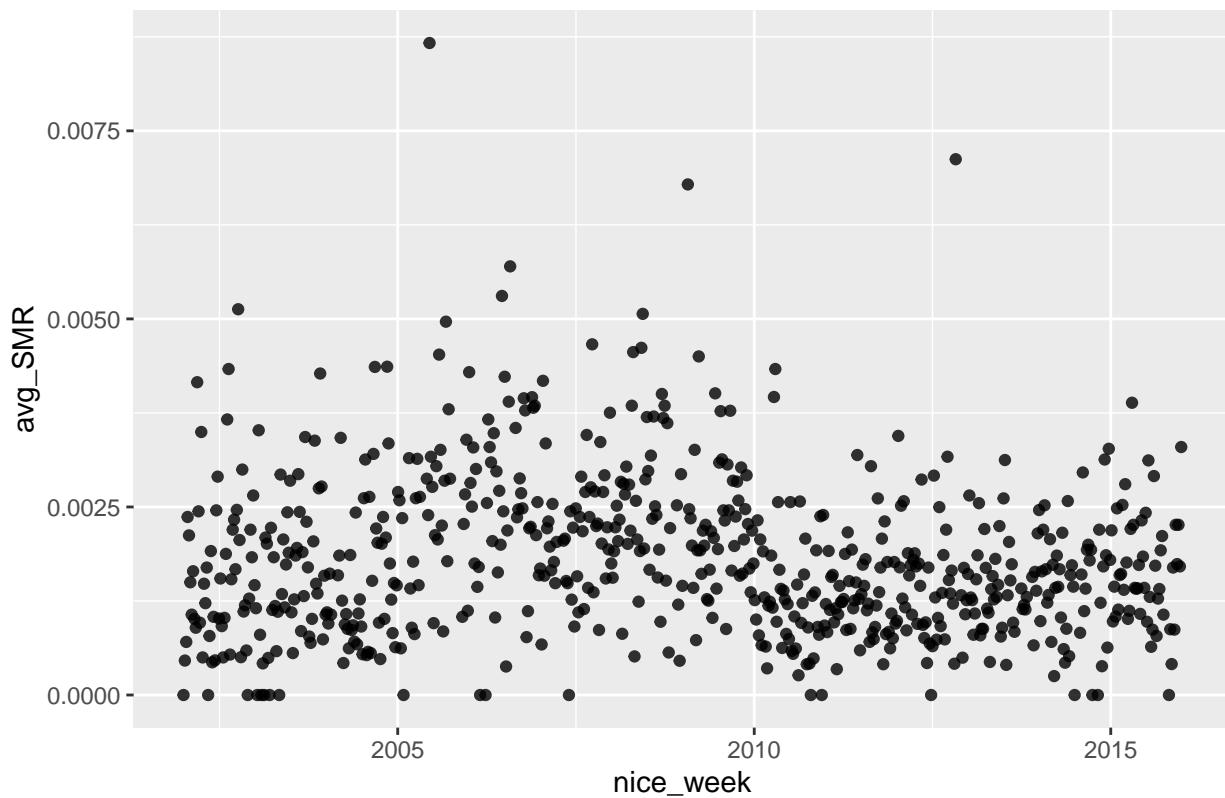
```
##  
## [[34]]
```

NCgreensboro2020 weekly SMR over time



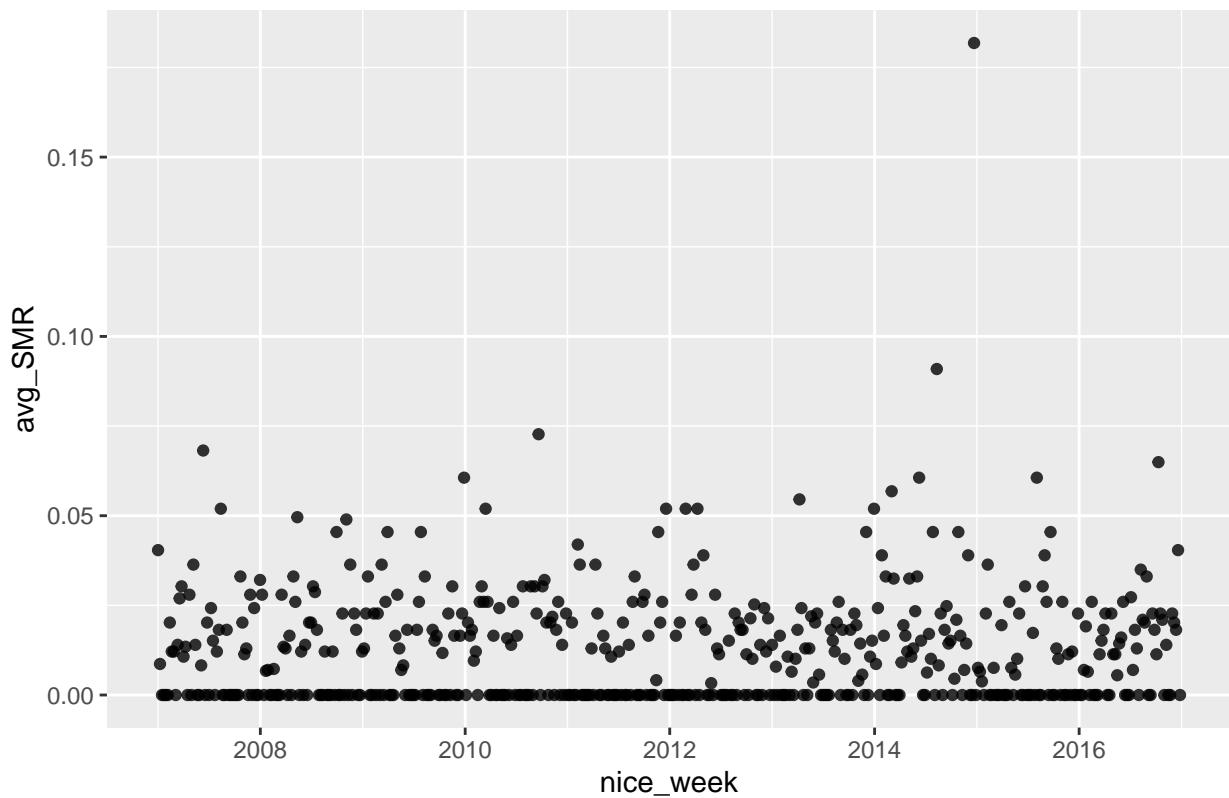
```
##  
## [[35]]
```

NCraleigh weekly SMR over time



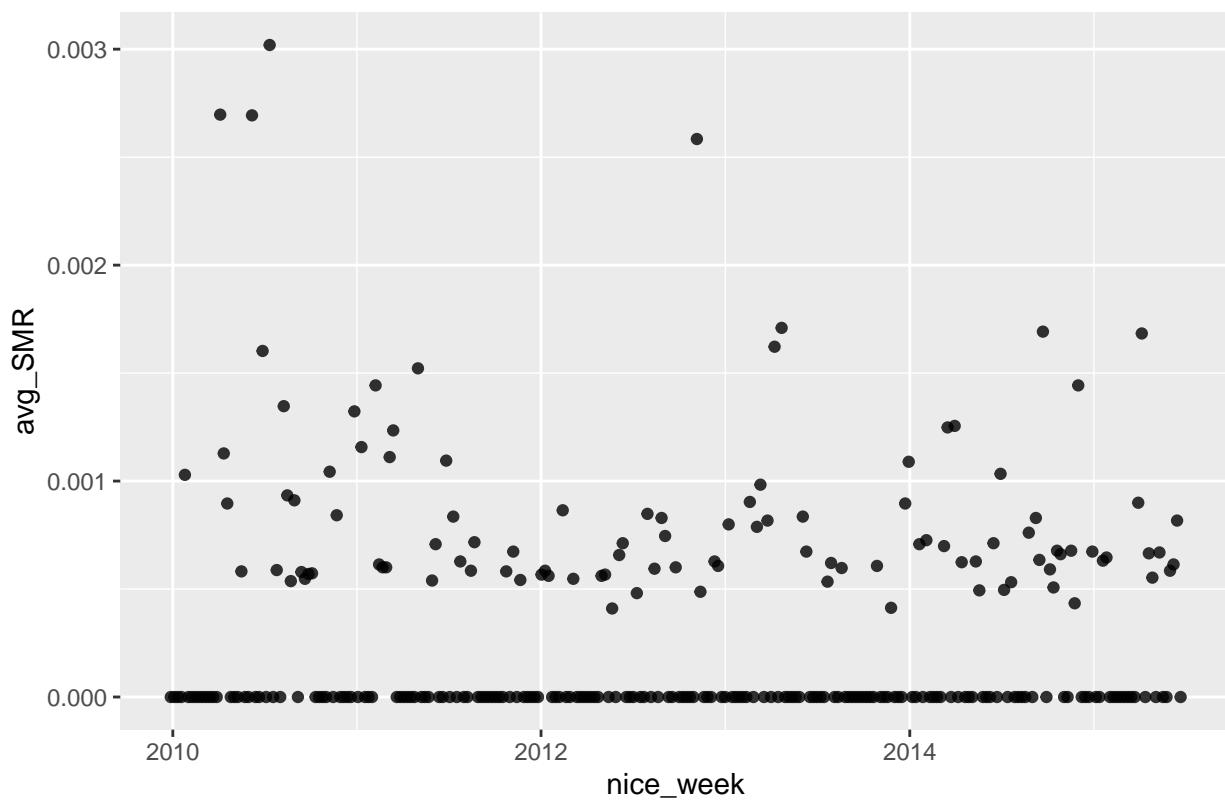
```
##  
## [[36]]
```

NDgrandforks weekly SMR over time



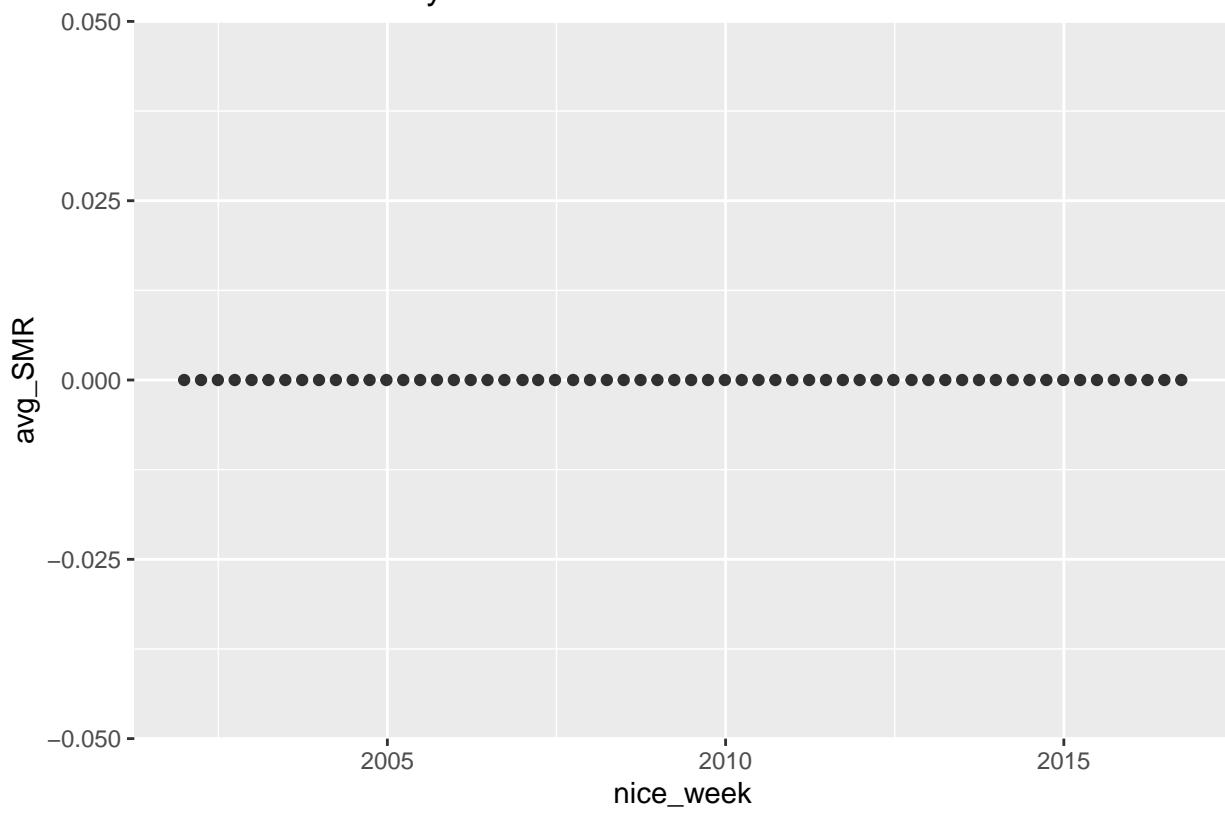
```
##  
## [[37]]
```

NDstatewide weekly SMR over time



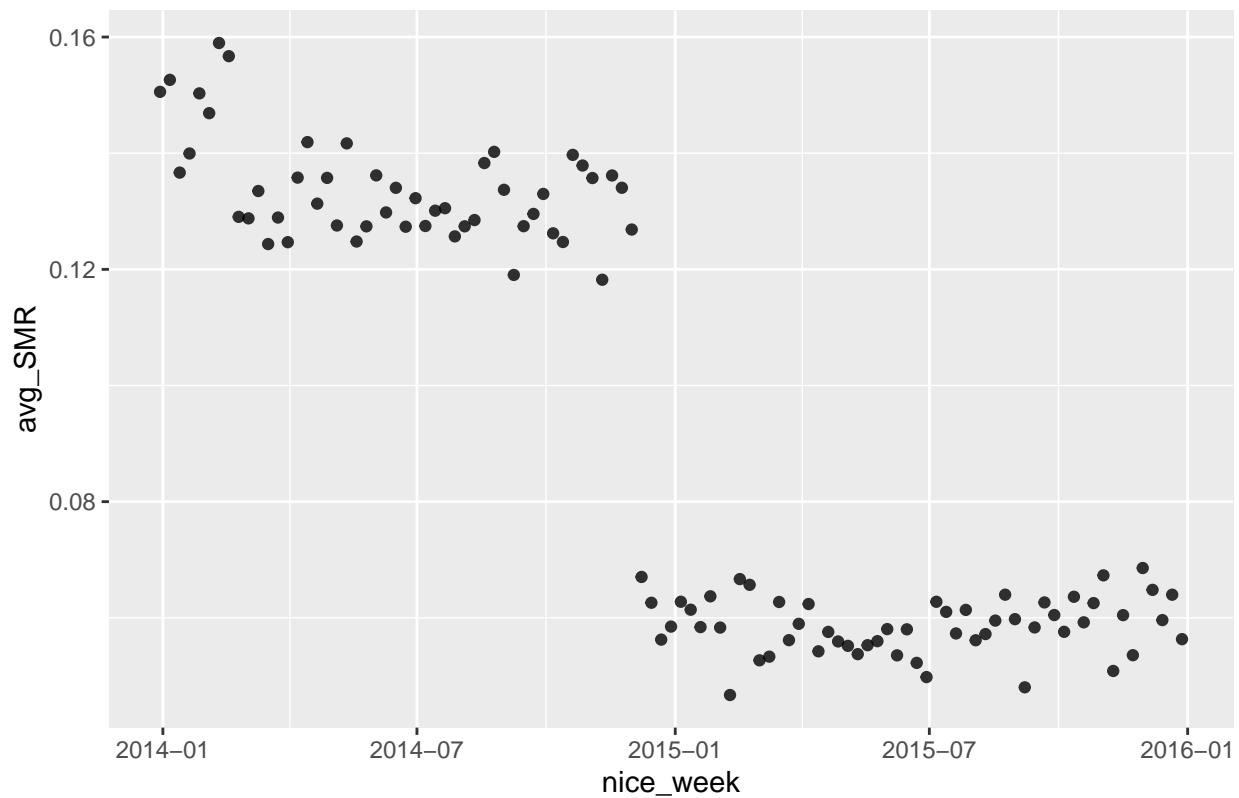
```
##  
## [38]
```

NEstatewide weekly SMR over time



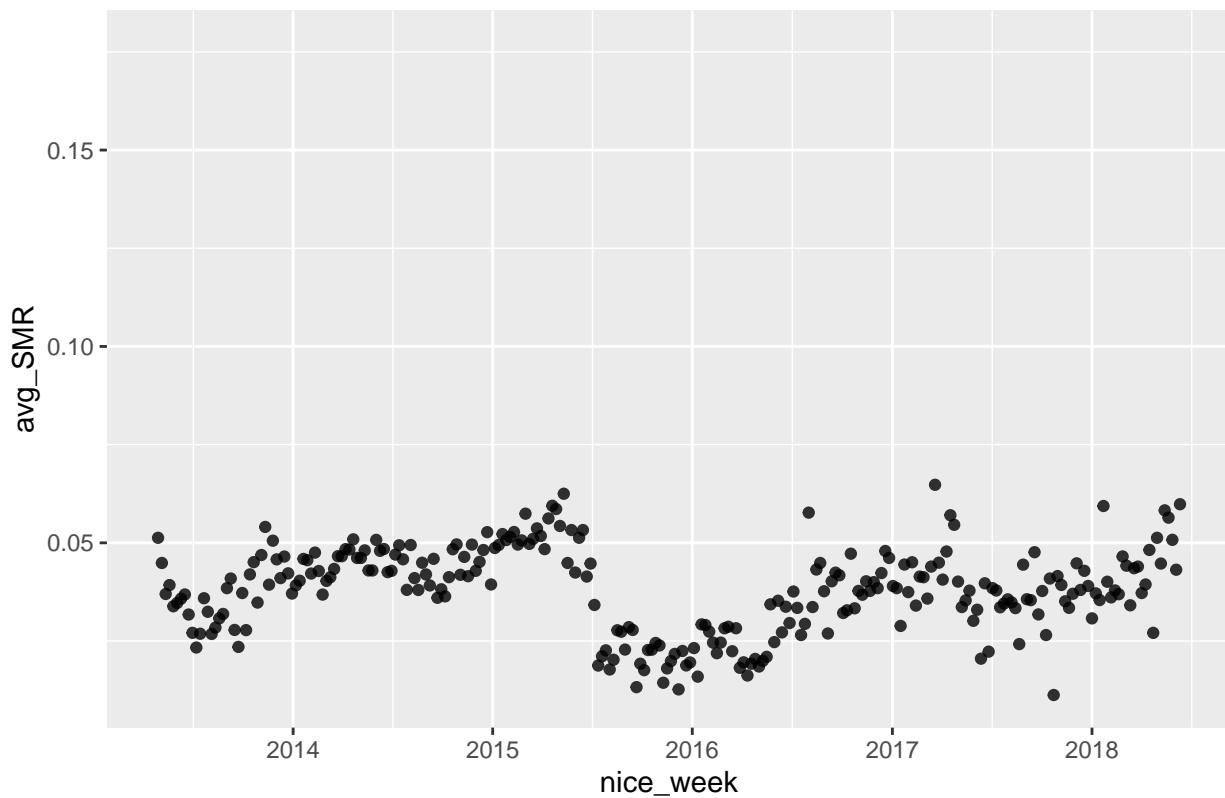
```
##  
## [[39]]
```

NH statewide weekly SMR over time



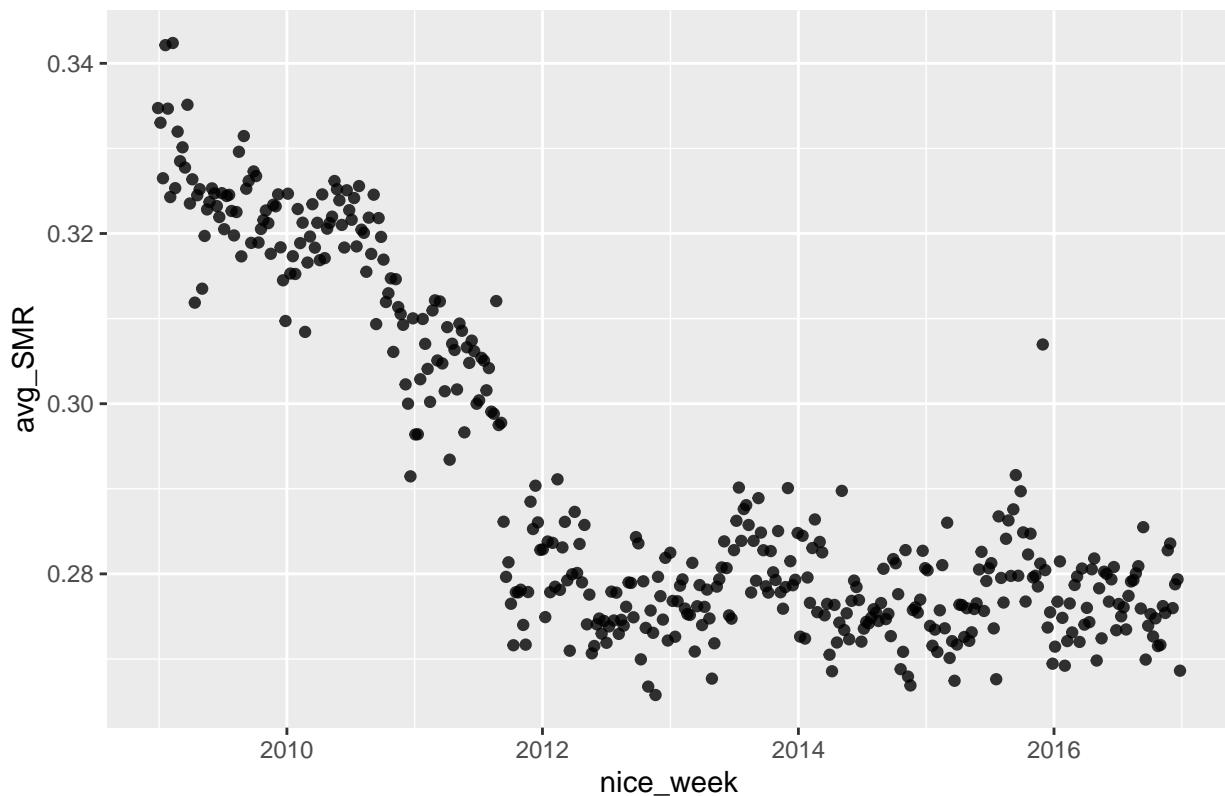
```
##  
## [[40]]
```

NJcamden weekly SMR over time



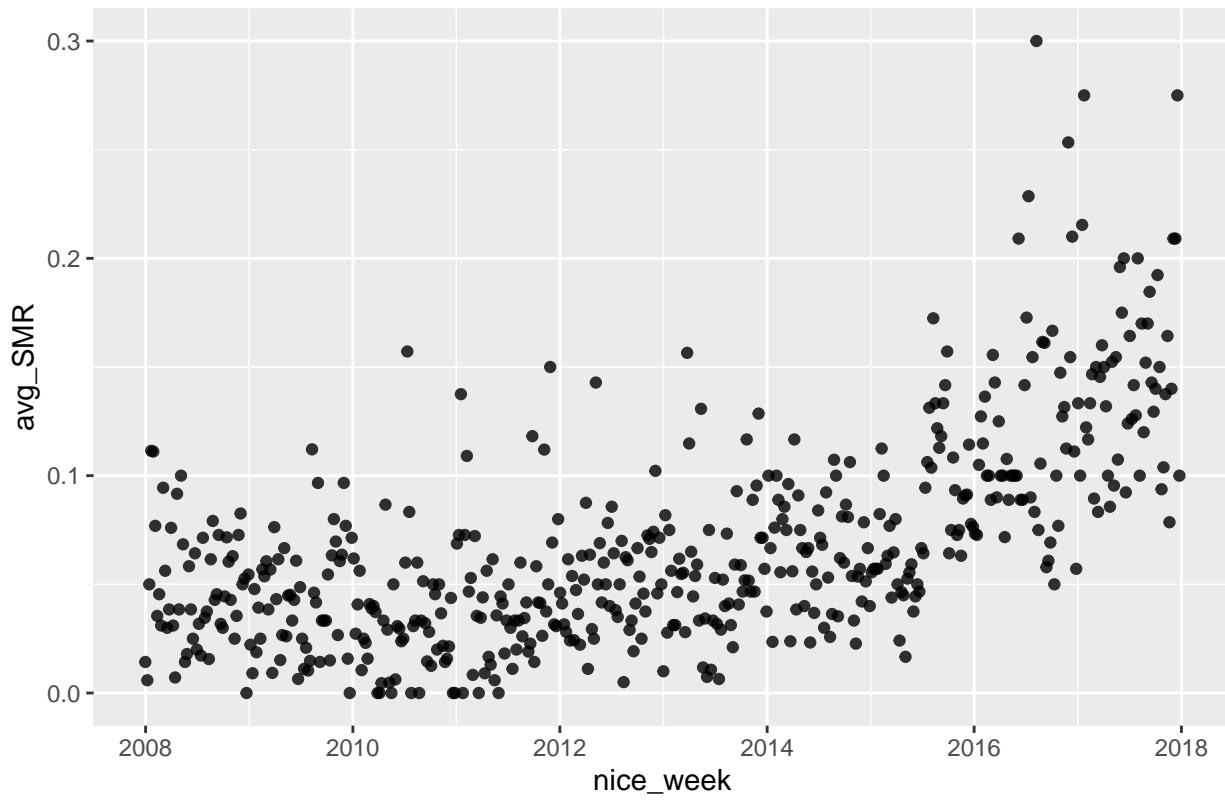
```
##  
## [[41]]
```

NJ statewide weekly SMR over time



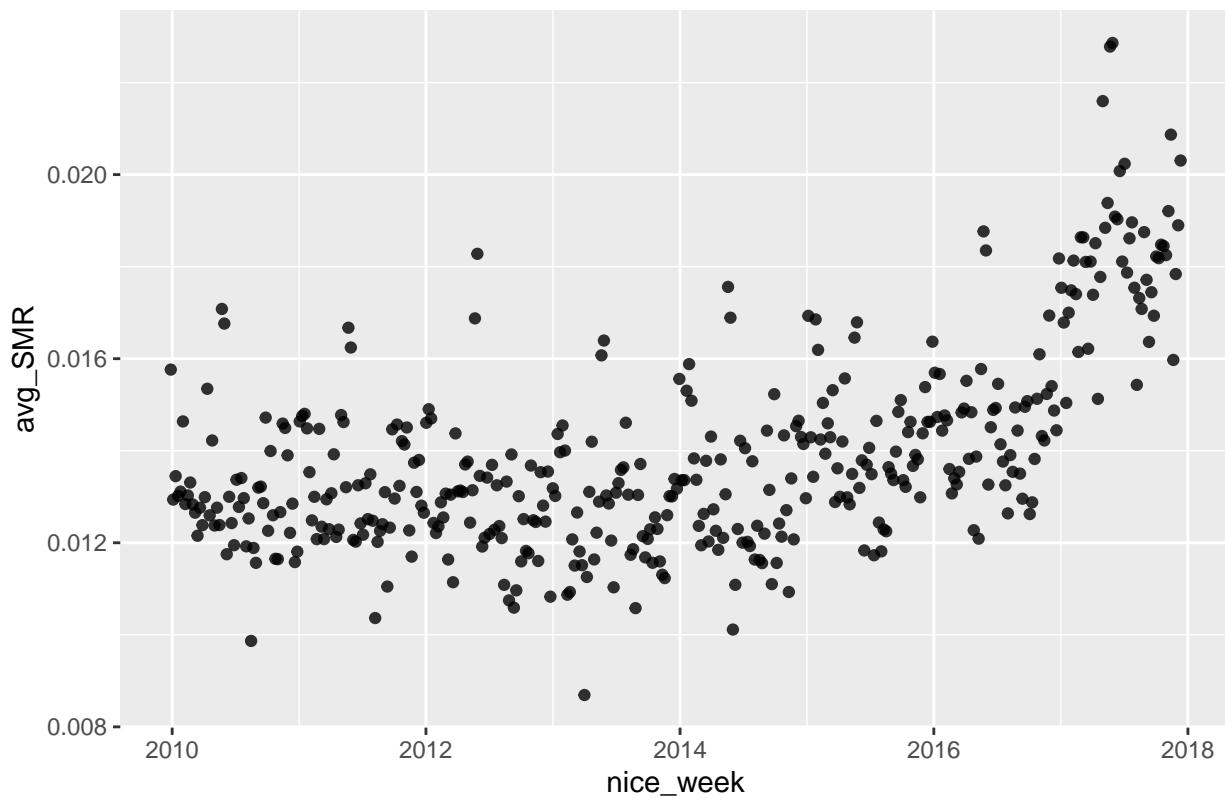
```
##  
## [[42]]
```

NYalbany weekly SMR over time



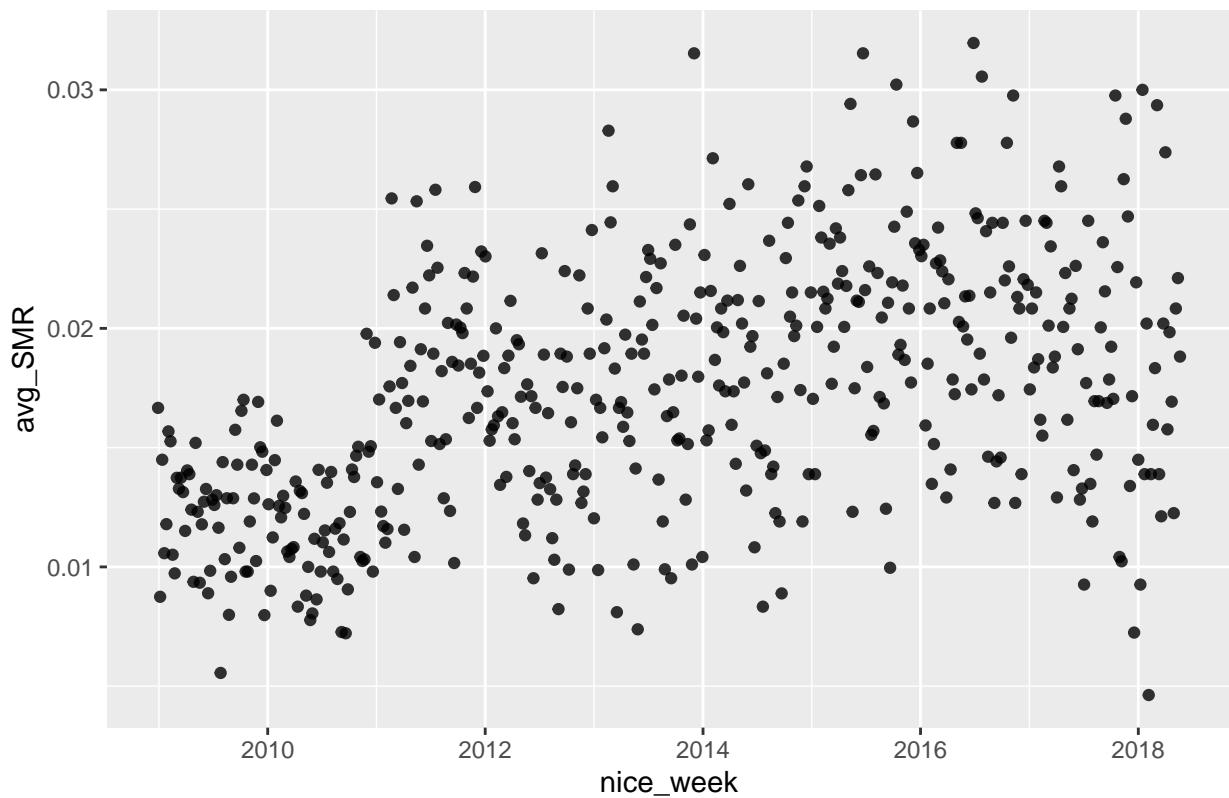
```
##  
## [[43]]
```

NYstatewide weekly SMR over time



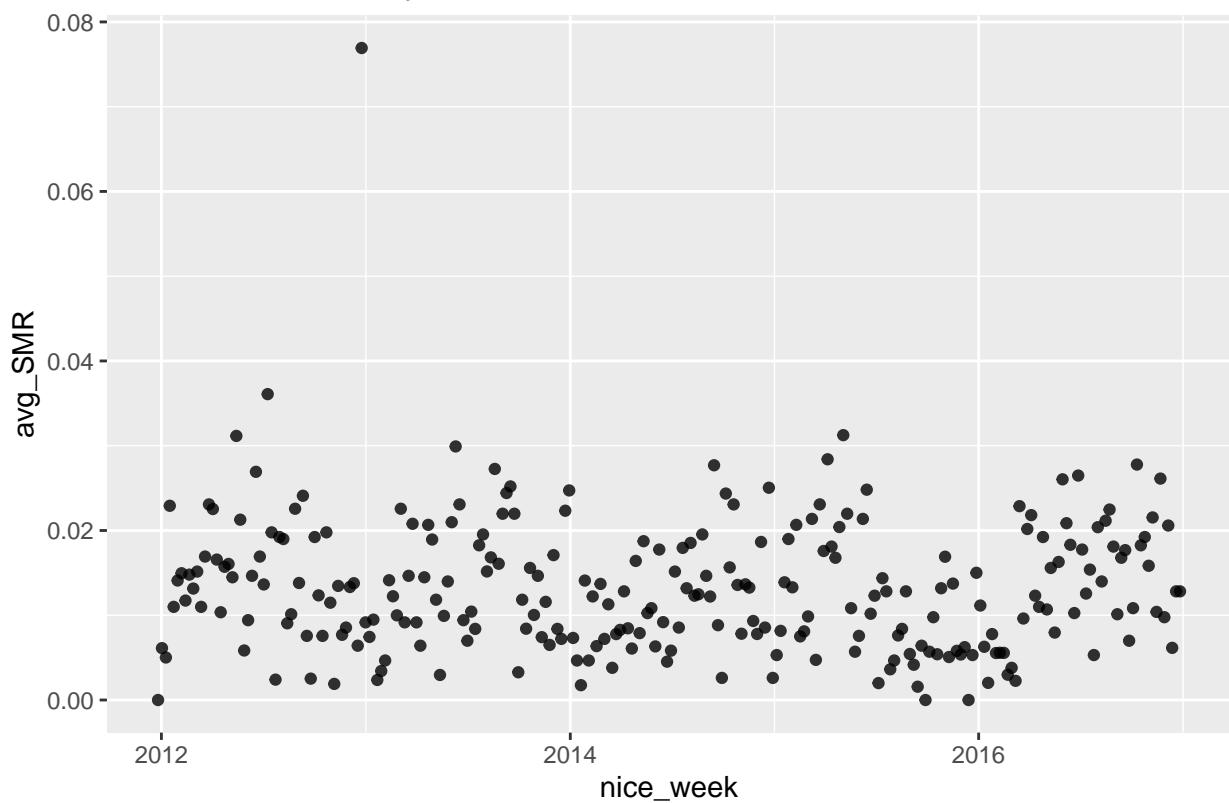
```
##  
## [[44]]
```

OHcincinnati weekly SMR over time



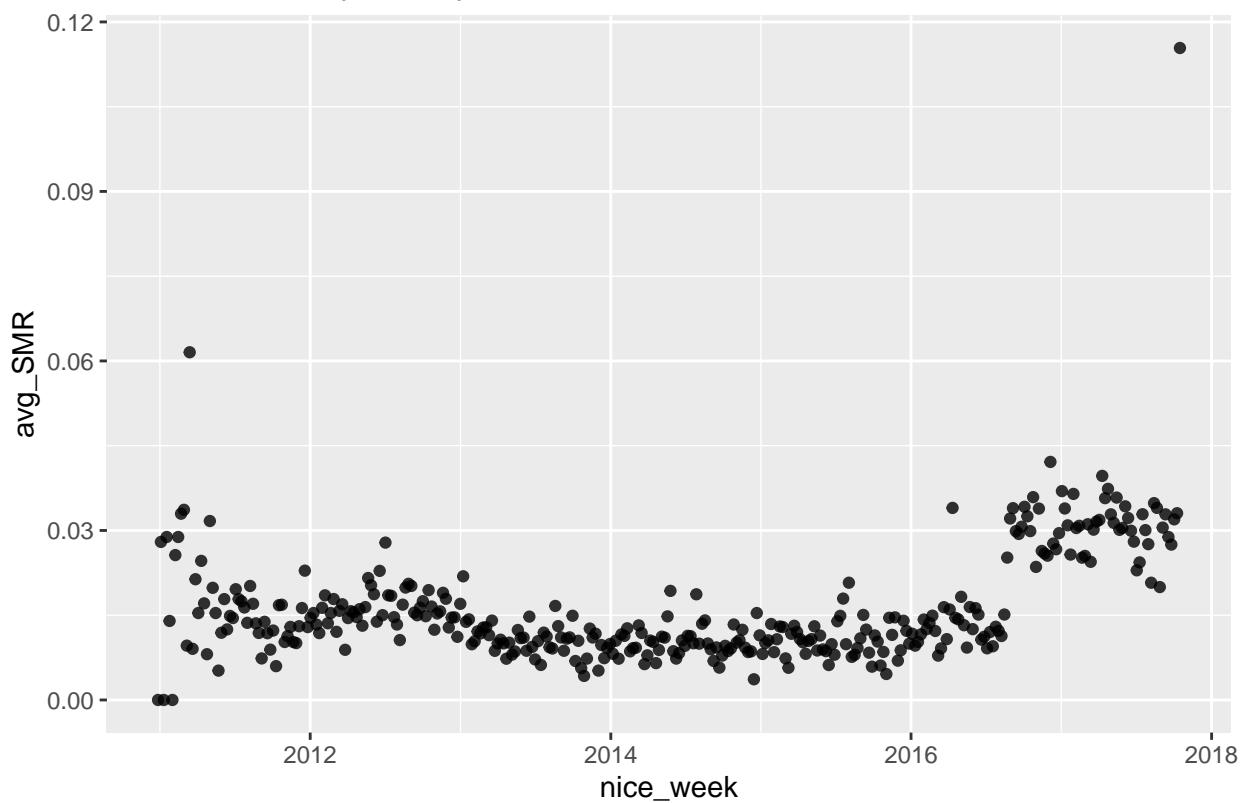
```
##  
## [[45]]
```

OHcolumbus weekly SMR over time



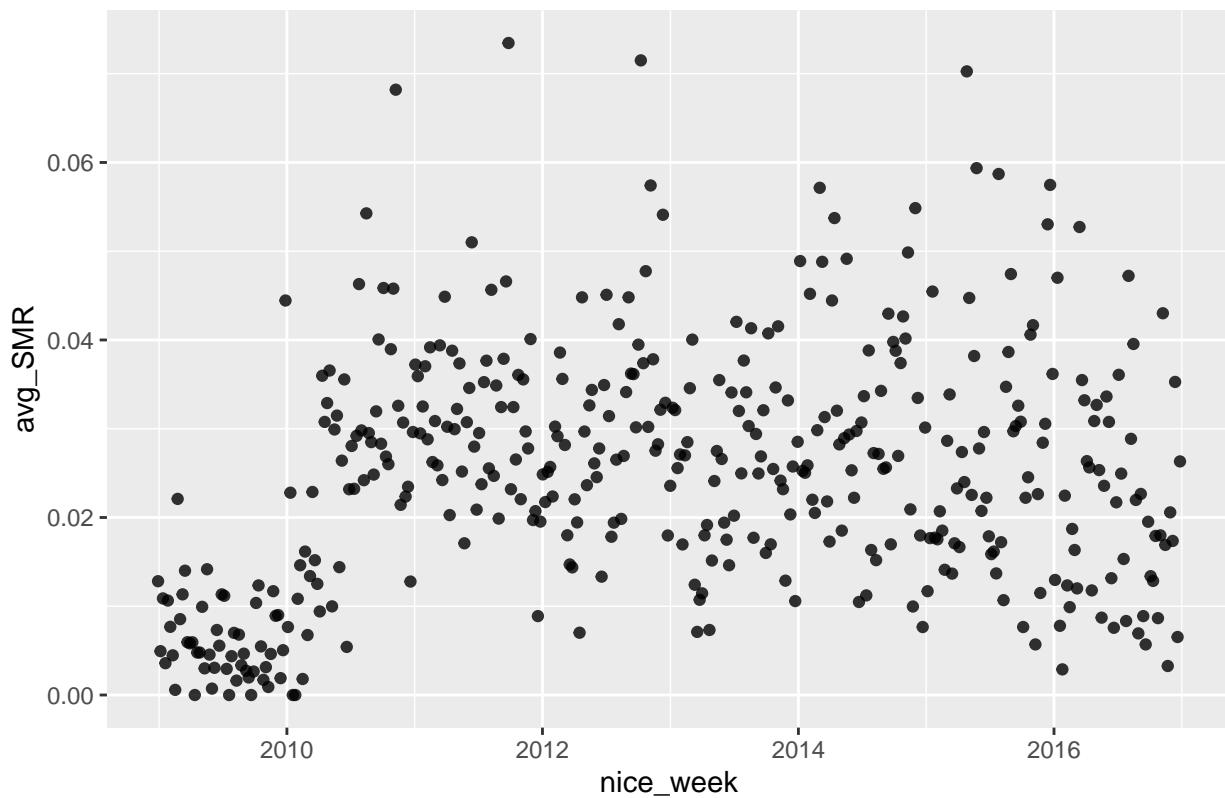
```
##  
## [[46]]
```

OKoklahomacity weekly SMR over time



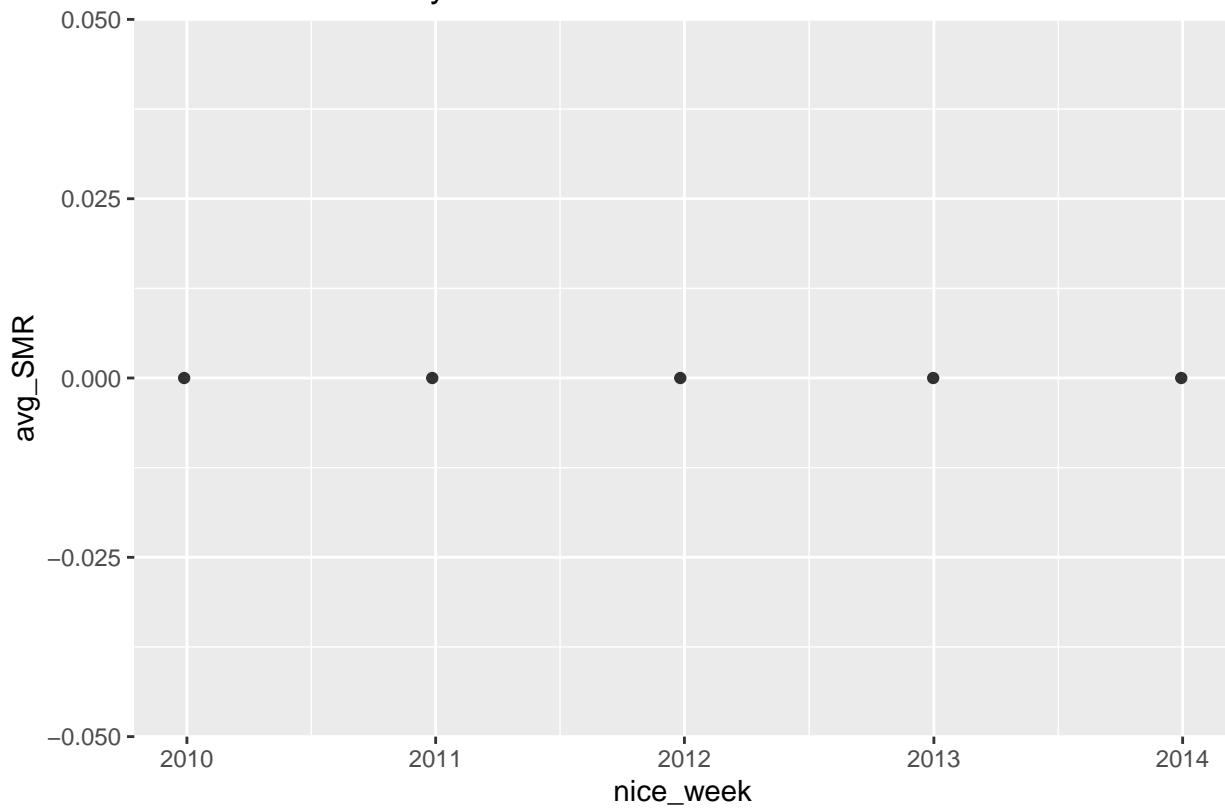
```
##  
## [[47]]
```

OKtulsa weekly SMR over time



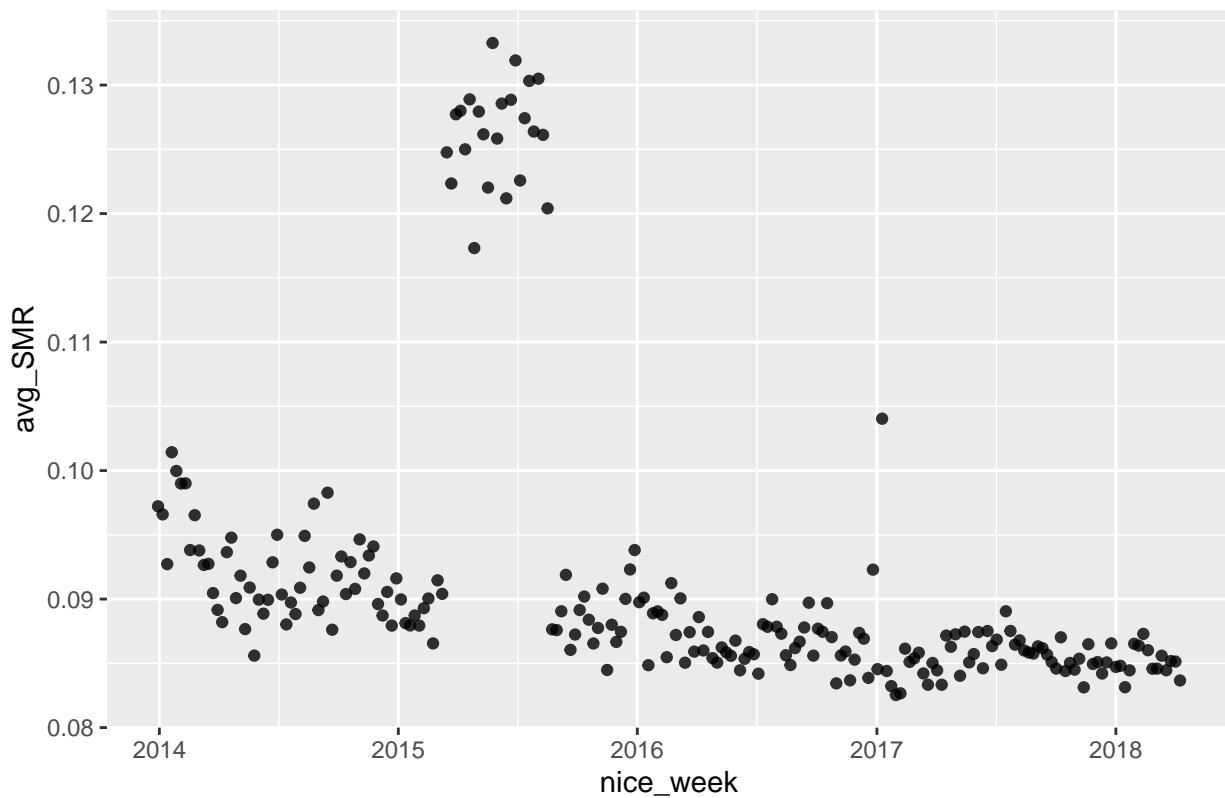
```
##  
## [[48]]
```

ORstatewide weekly SMR over time



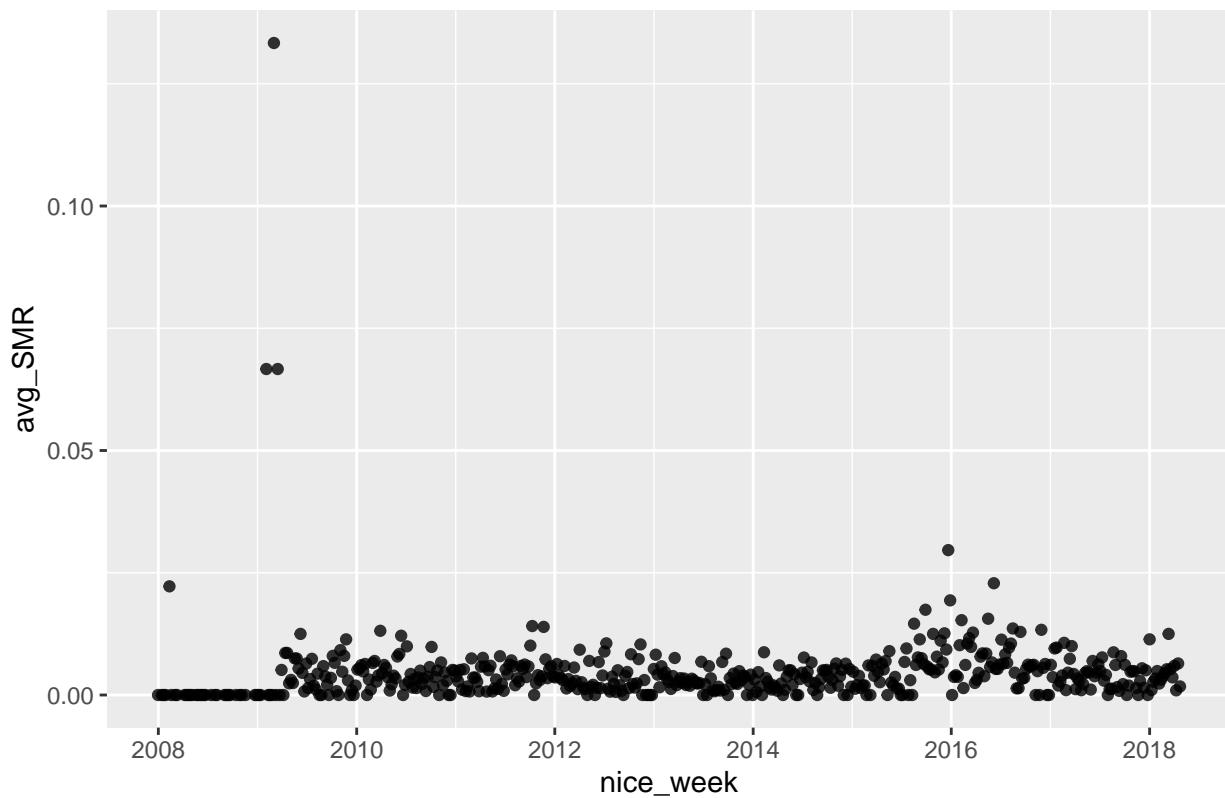
```
##  
## [[49]]
```

PAphiladelphia weekly SMR over time



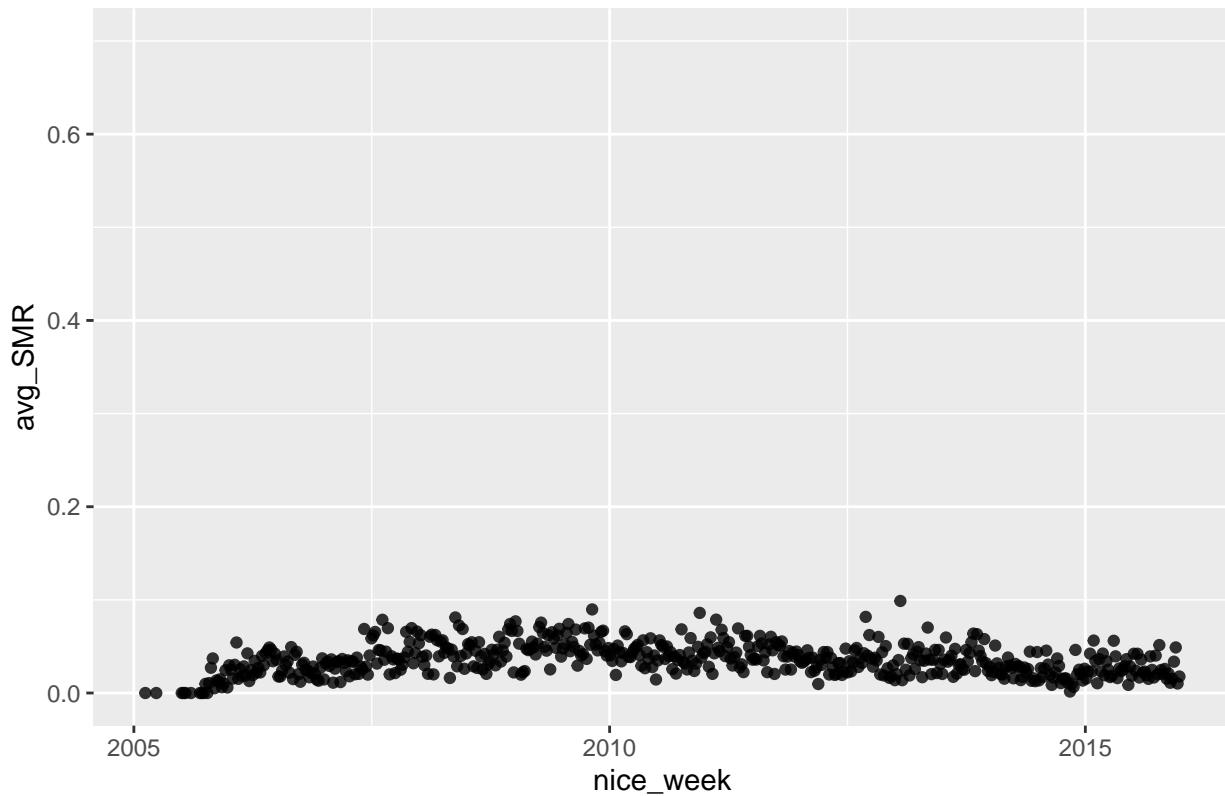
```
##  
## [[50]]
```

PApittsburgh weekly SMR over time



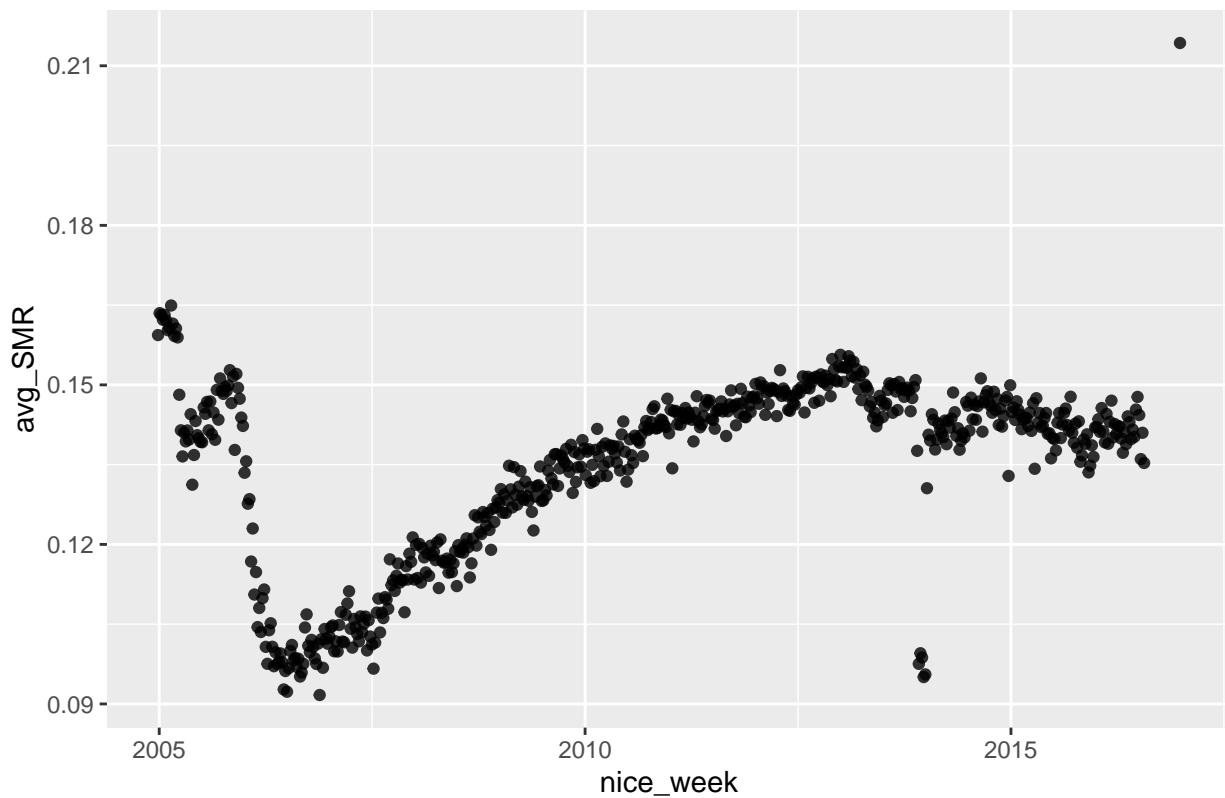
```
##  
## [[51]]
```

R1 statewide weekly SMR over time



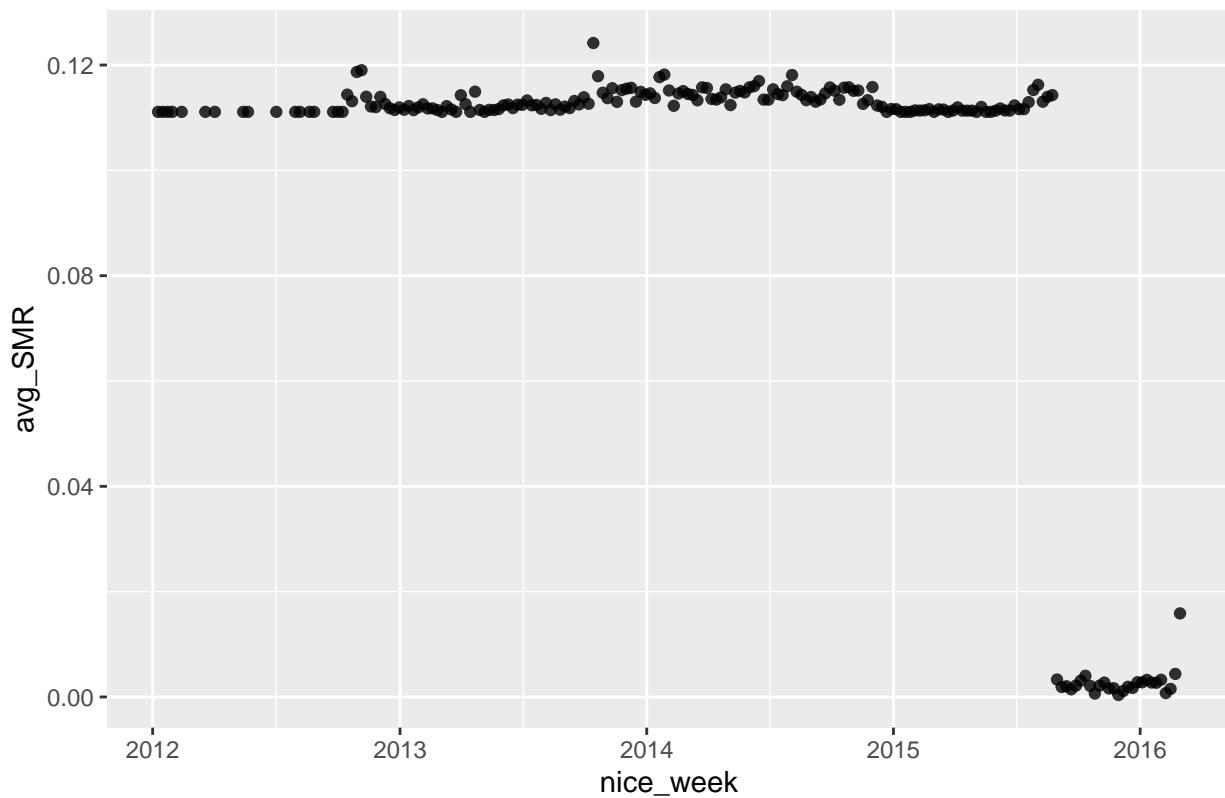
```
##  
## [[52]]
```

SC statewide weekly SMR over time



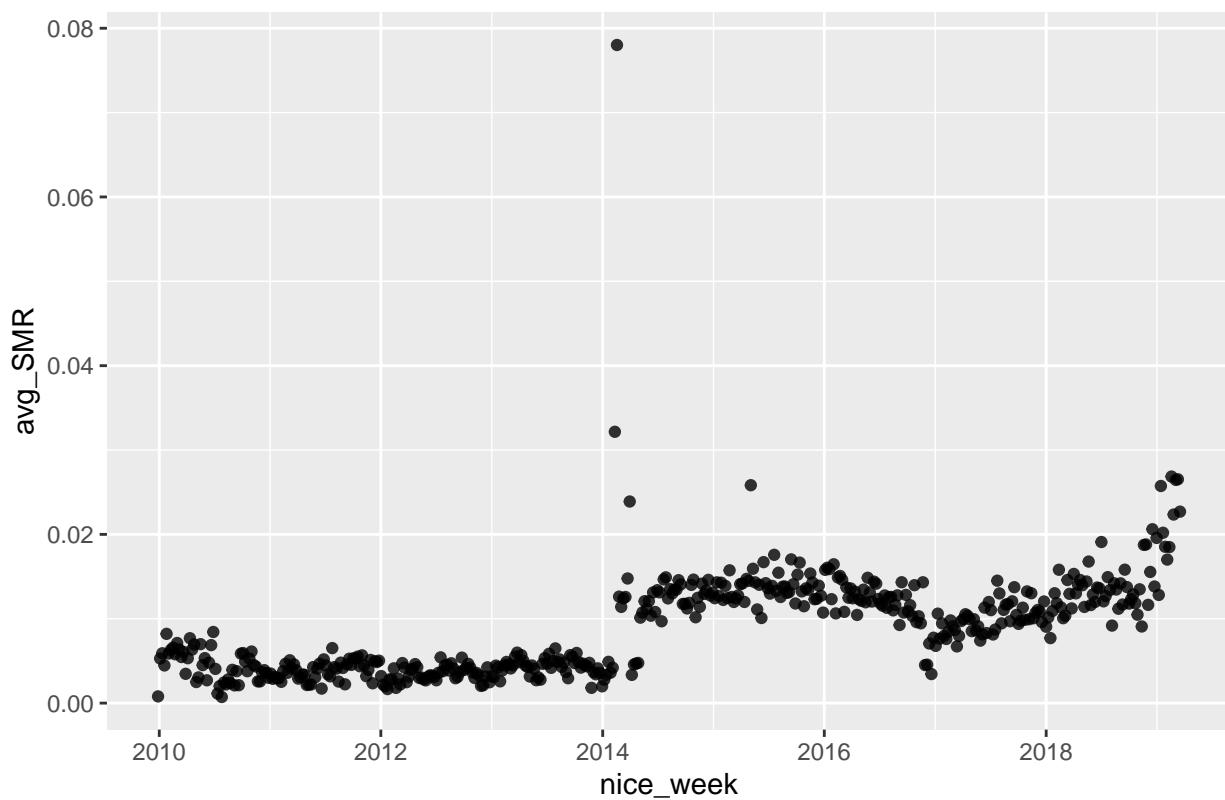
```
##  
## [[53]]
```

SD statewide weekly SMR over time



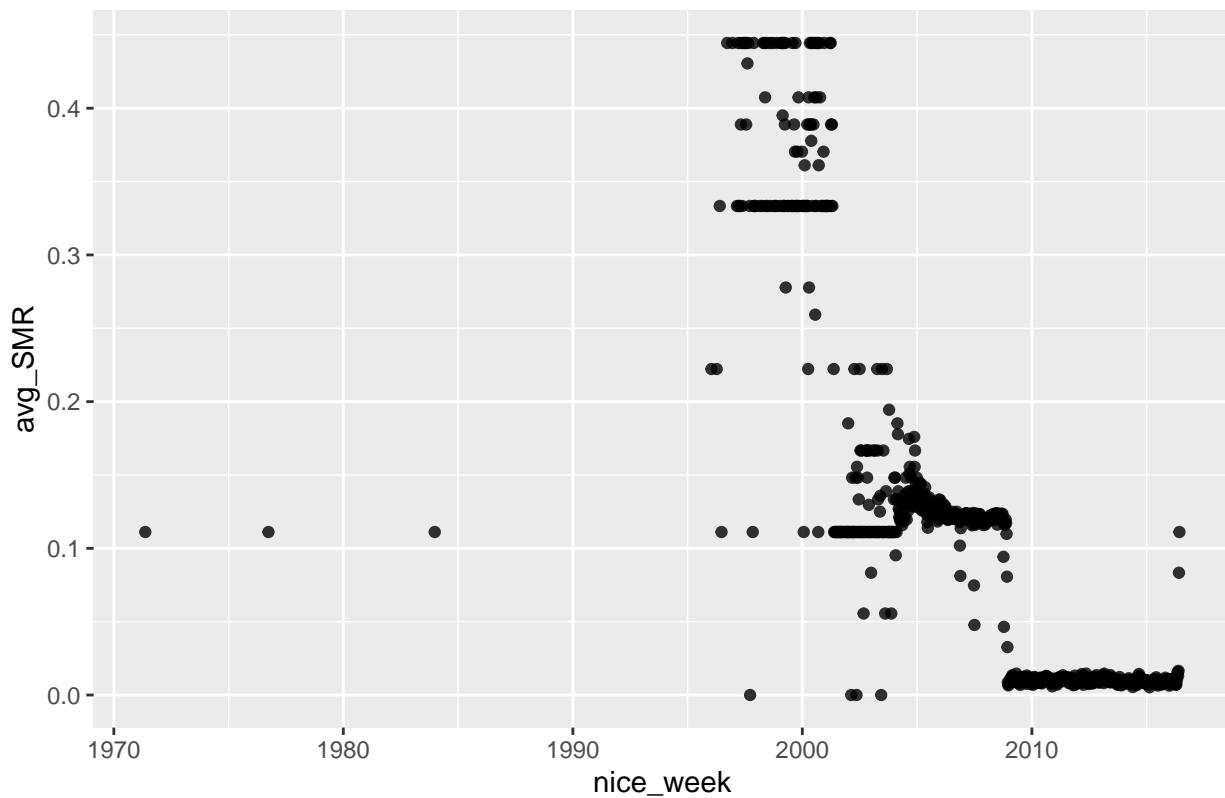
```
##  
## [[54]]
```

TNnashville weekly SMR over time



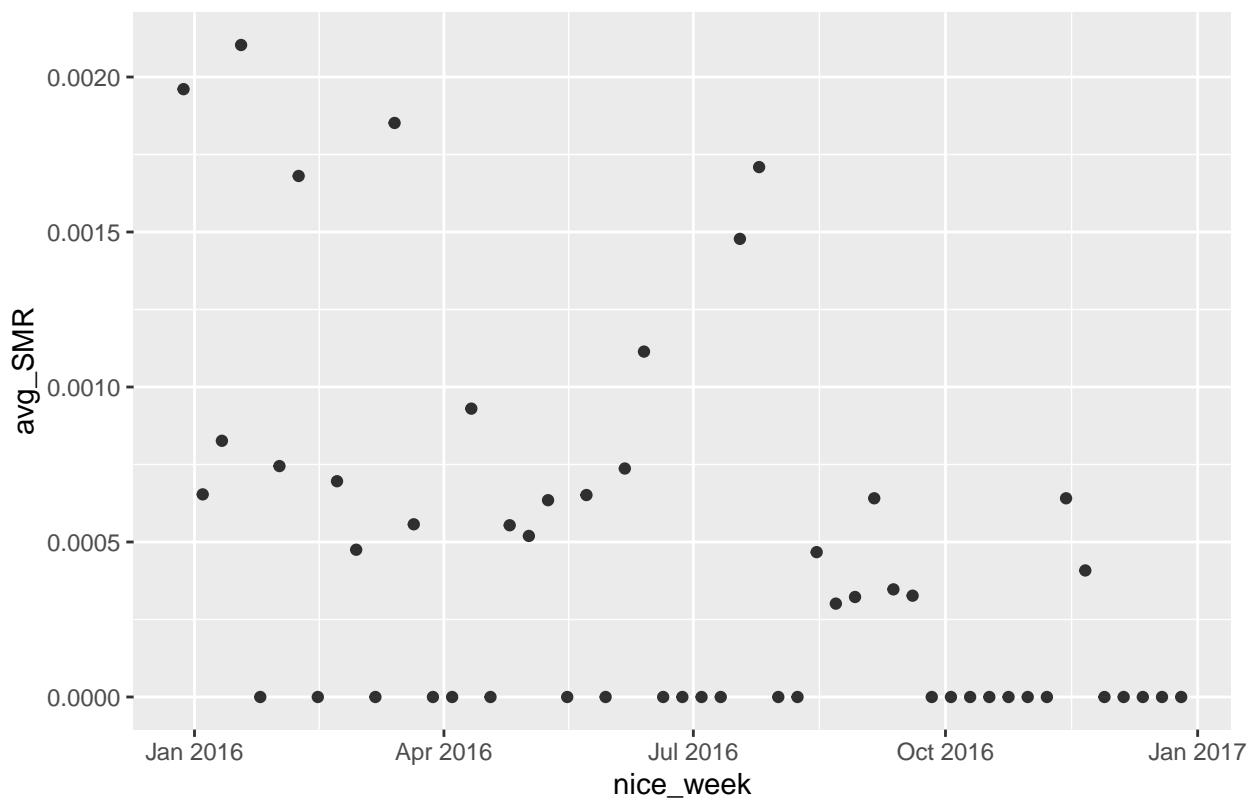
```
##  
## [[55]]
```

TNstate weekly SMR over time



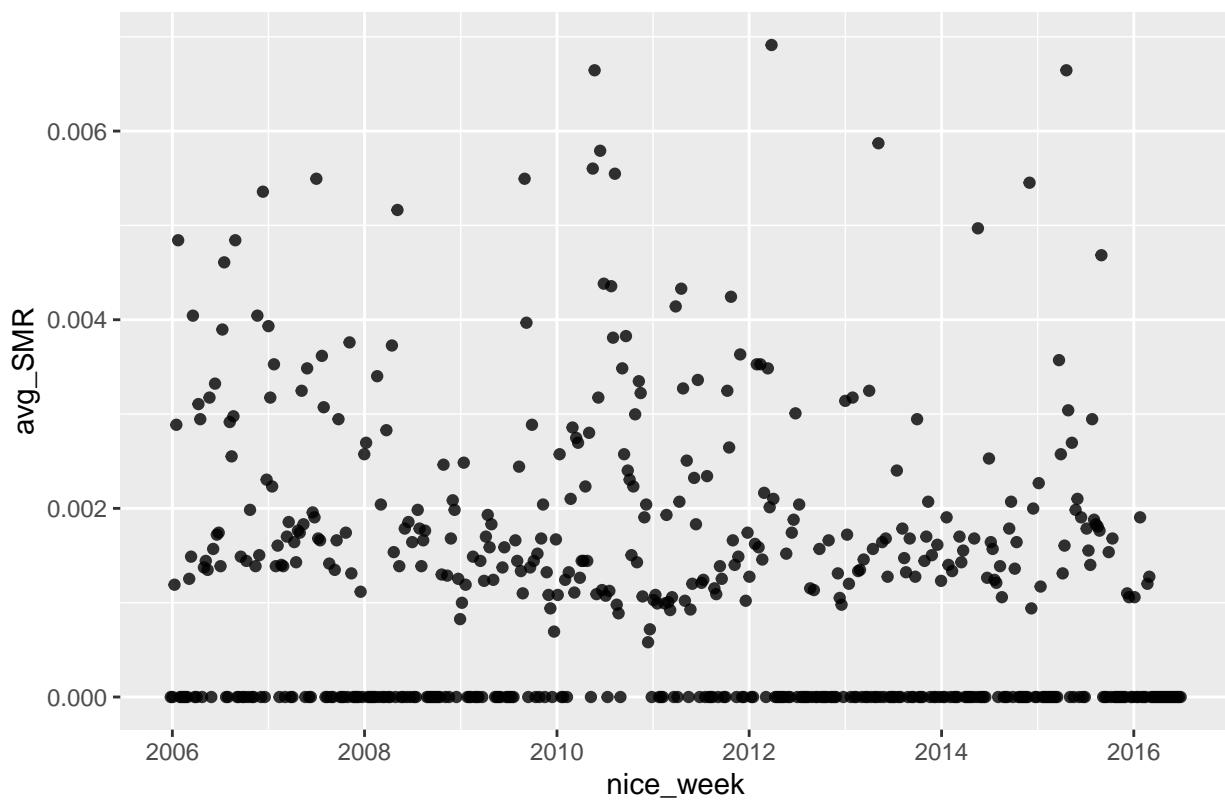
```
##  
## [[56]]
```

TXarlington weekly SMR over time



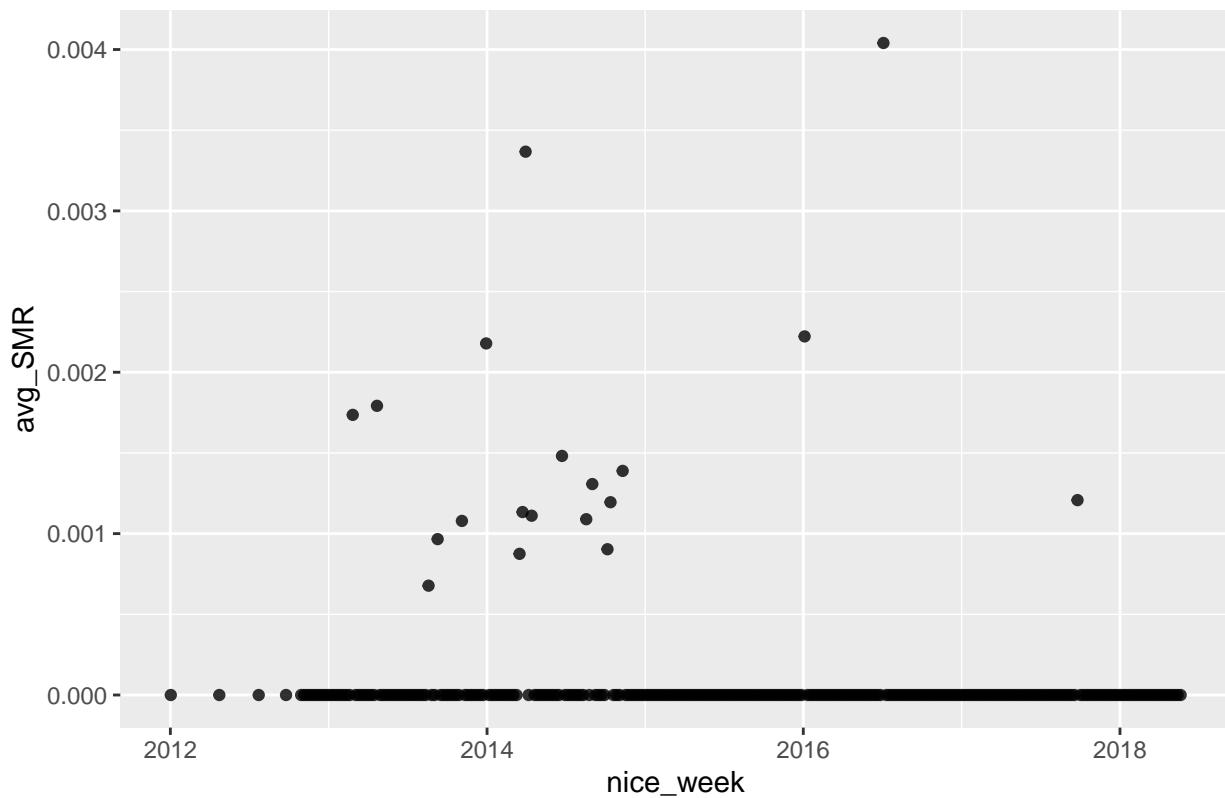
```
##  
## [[57]]
```

TXaustin weekly SMR over time



```
##  
## [[58]]
```

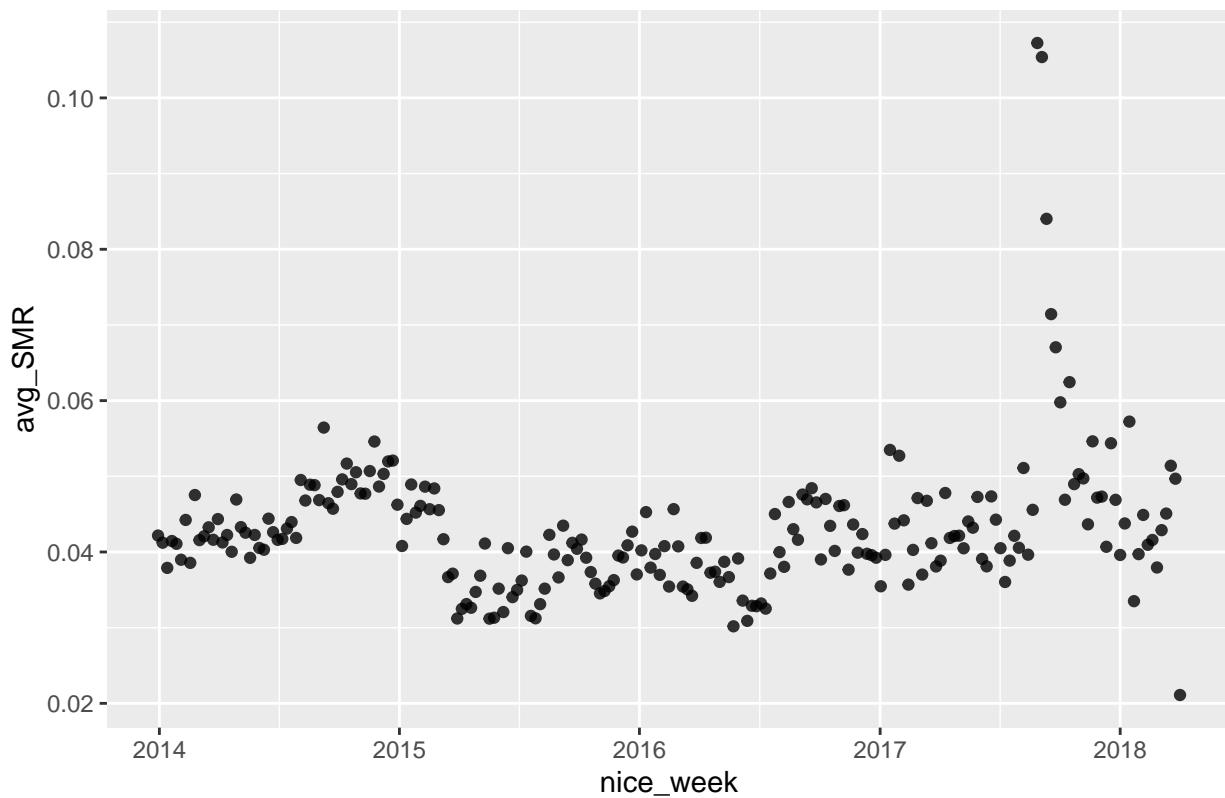
TXgarland weekly SMR over time



##

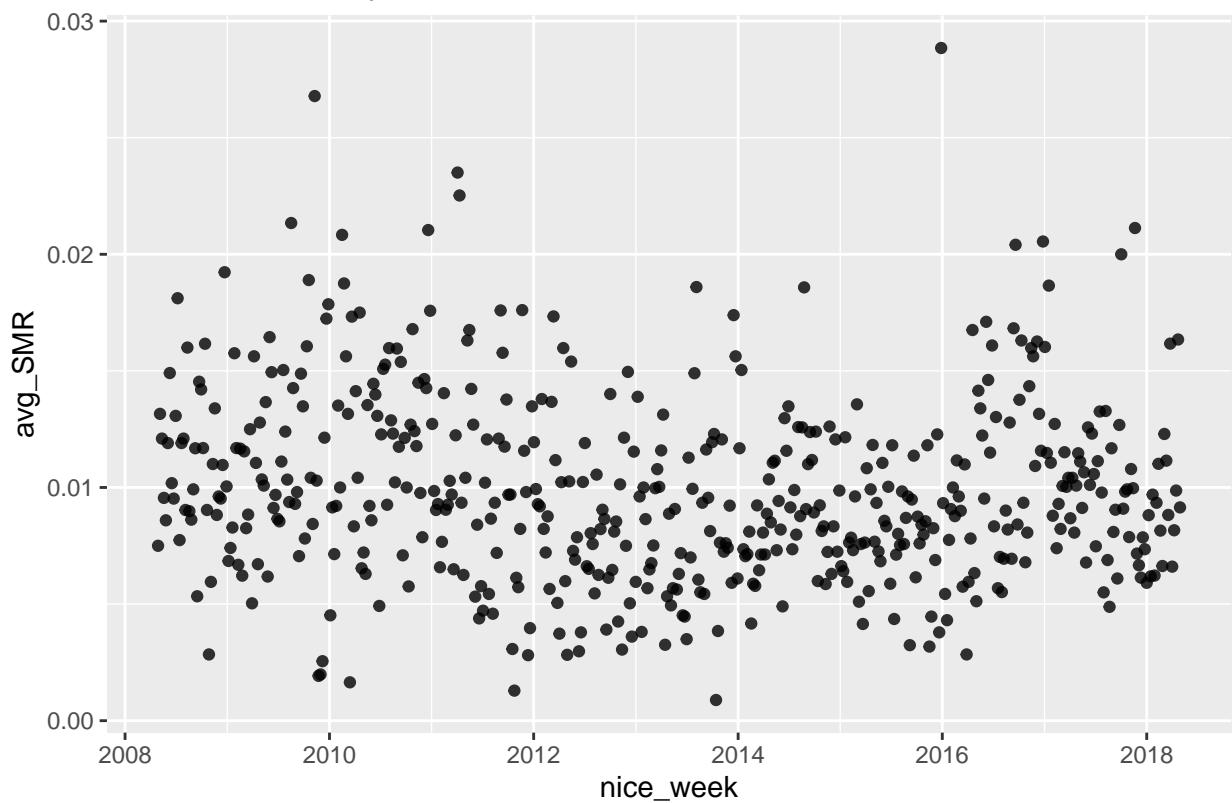
[[59]]

TXhouston weekly SMR over time



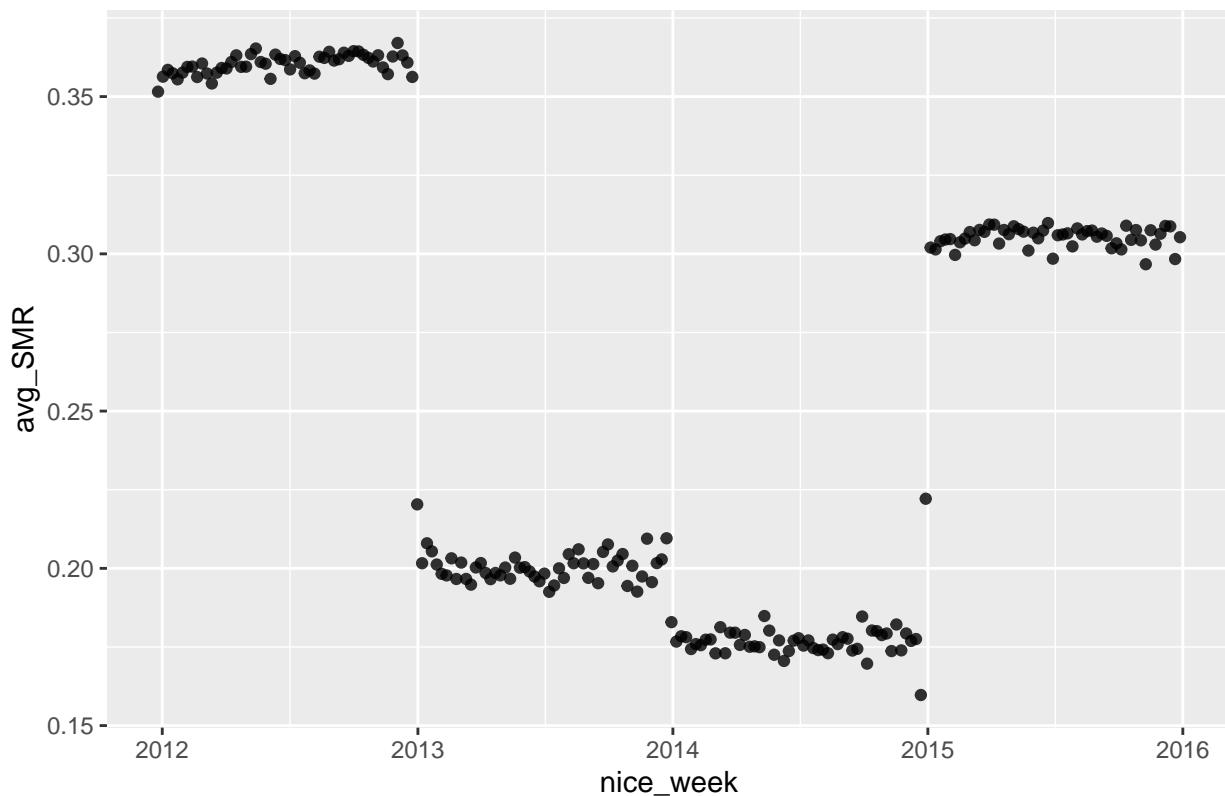
```
##  
## [[60]]
```

TXlubbock weekly SMR over time



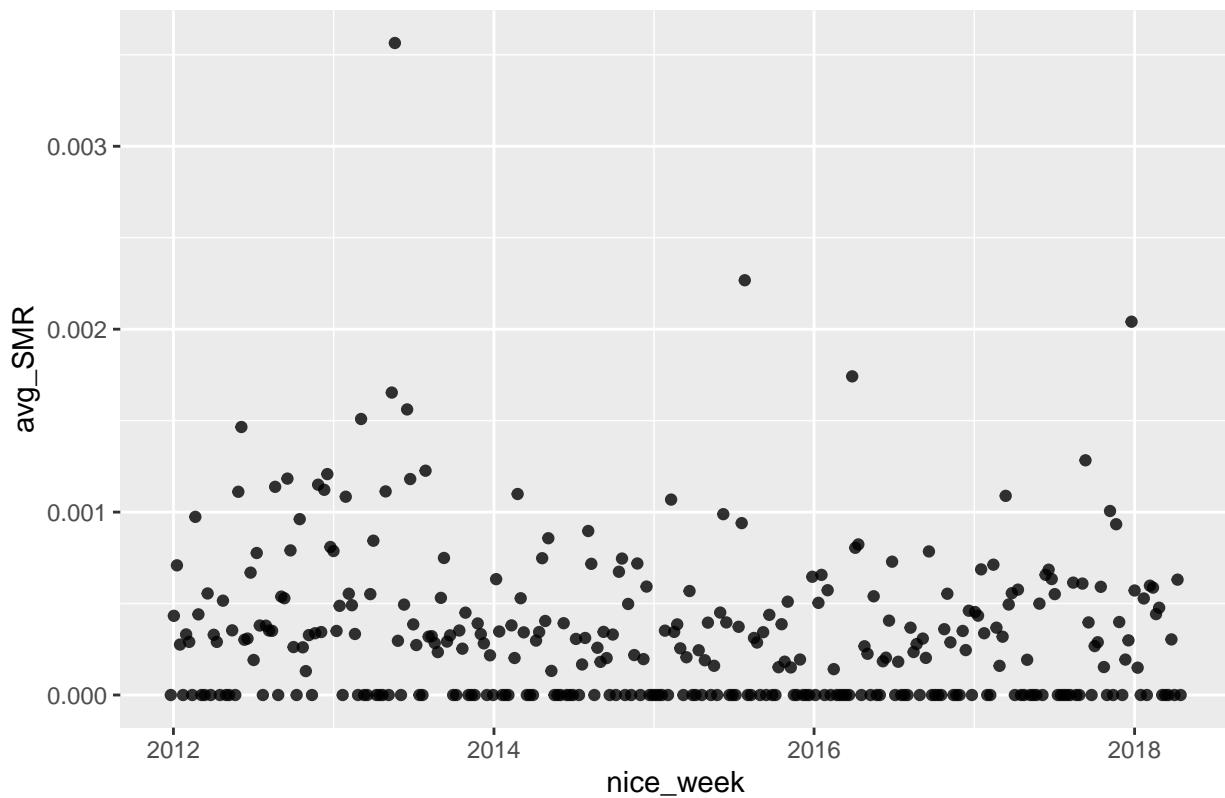
```
##  
## [[61]]
```

TXplano weekly SMR over time



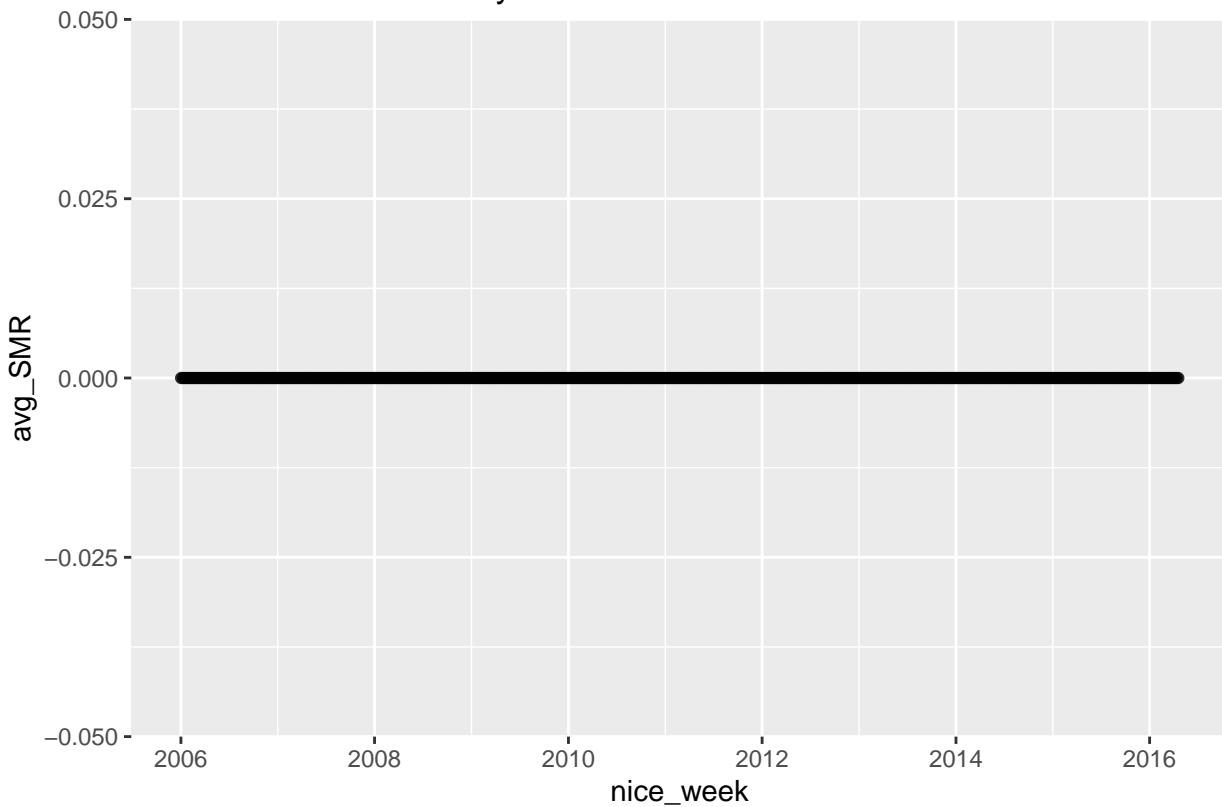
```
##  
## [[62]]
```

TXsanantonio weekly SMR over time



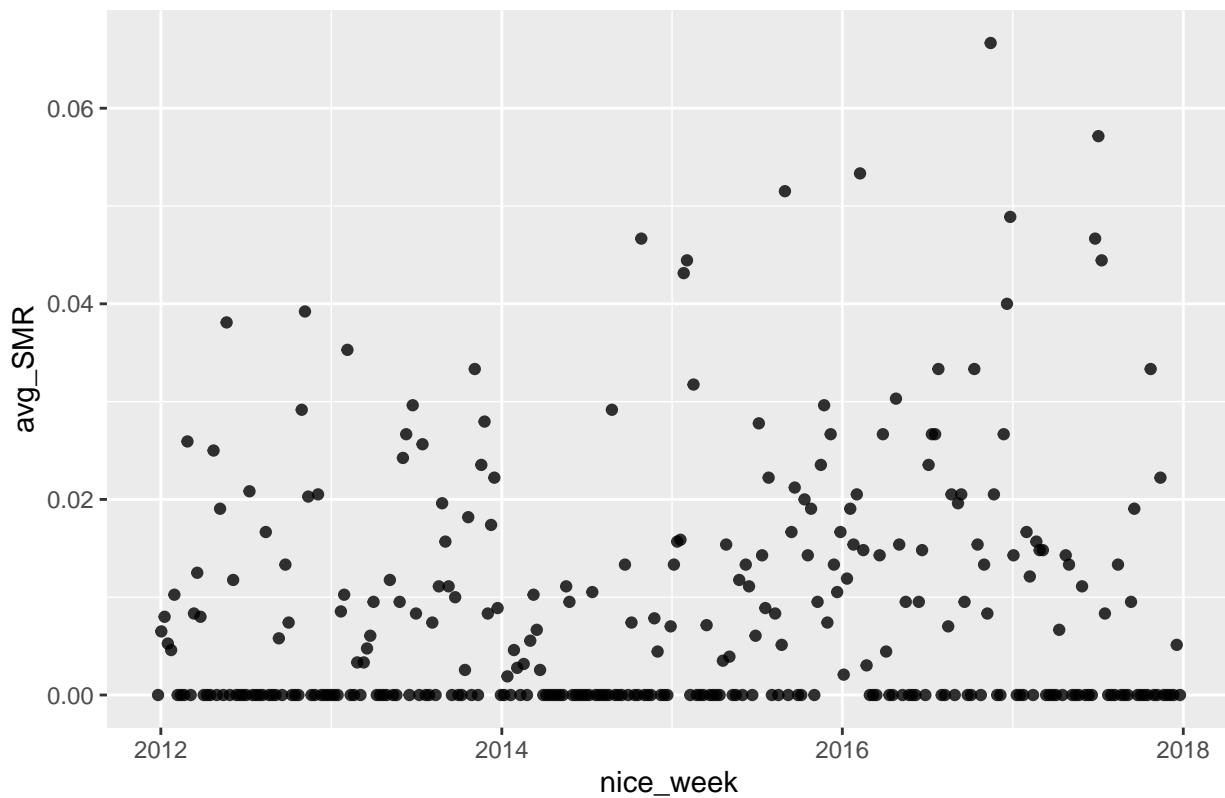
```
##  
## [[63]]
```

VAstatewide2020 weekly SMR over time



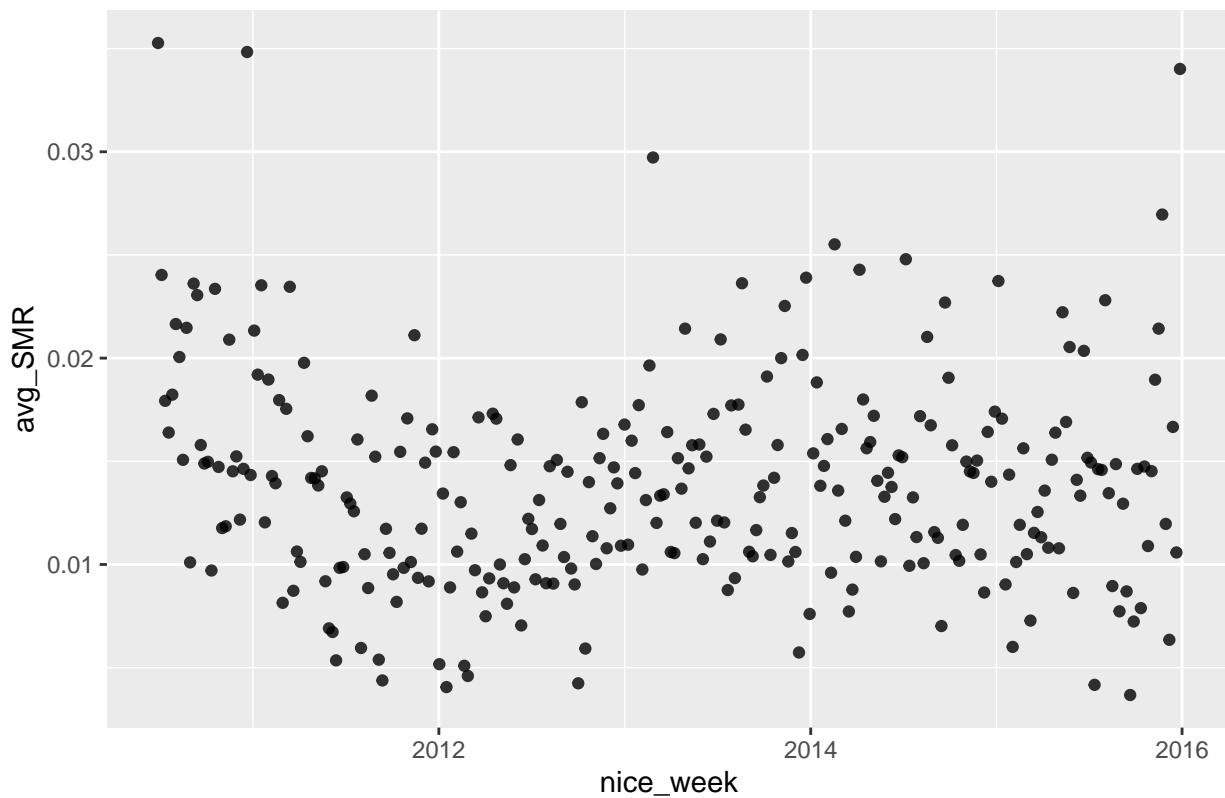
```
##  
## [[64]]
```

VTburlington2020 weekly SMR over time



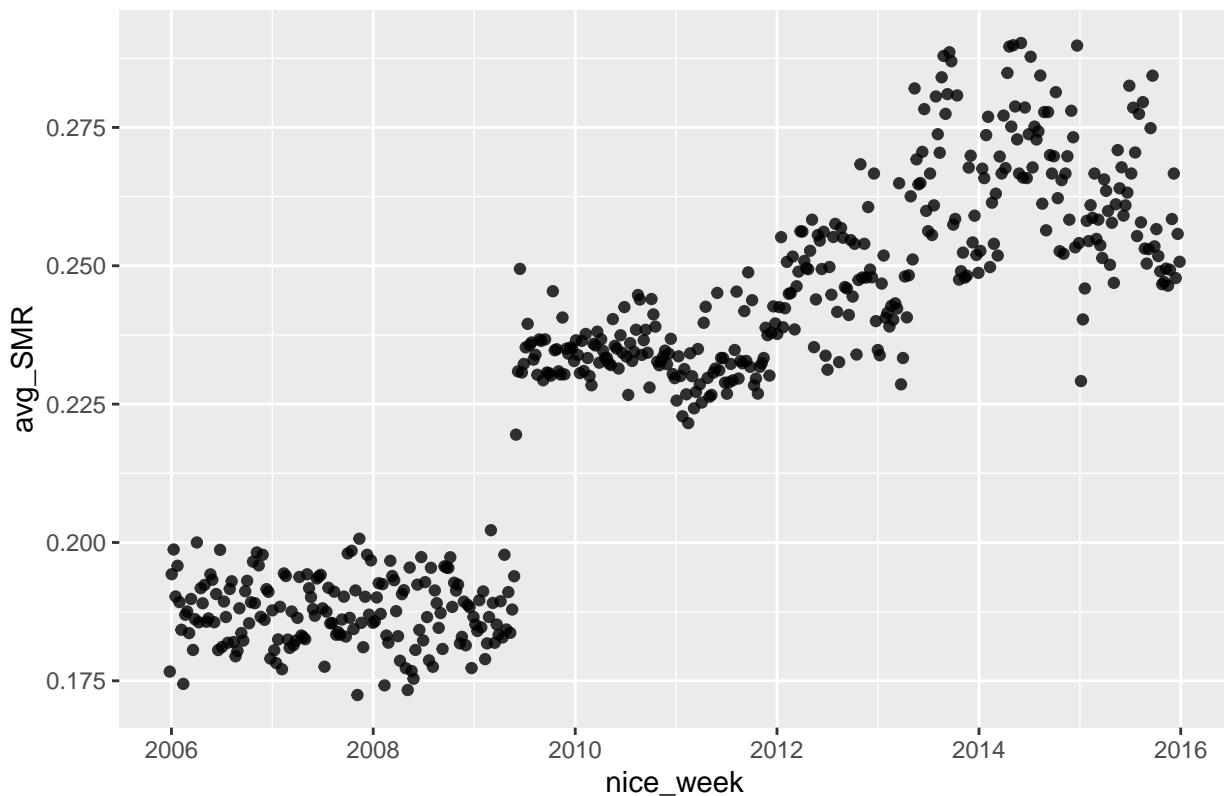
```
##  
## [[65]]
```

VT statewide 2020 weekly SMR over time



```
##  
## [[66]]
```

WAseattle weekly SMR over time



- SMR and its spread is different throughout the years
- the roughly-discernible average SMR can have level changes (Iowa statewide) or have gradual sloping changes
- a low missingness rate isn't indicative of a *good* data set. example is CA los angeles – the dataset records very few variable with a consistently near-0 SMR. but no discrimination analysis tests can be done on it because it doesn't record subject_race!
- some datasets put in all their data by the month, so the dates are actually all inaccurate because the date of the traffic stop is actually when the data are entered, not when the stop takes place!

What is wrong with ORstatewide?

Oregon records its stops by the year. So there are only 5 data points for the 5 years 2010 to 2014.

```
mysearch_dataset(dataset_lst, "ORstatewide") %>% distinct(date)
```

SMR by night/day, throughout years

Set up

To add the variable `day_night`, we find the sunset, sunrise, and twilight times and categorize each traffic stop as having occurred in the darkness, in the sunlight, or during an ambiguous twilight time.

Filter datasets For *time of day* analysis, we need variables `date` and `time`. We build on the filtering from the previous section.

```
# filtering OUT datasets that don't record time
time_lst <- lapply(dataset_lst, datasetContaining, c("time", "date"))
time_lst <- time_lst[sapply(time_lst, function(x) isTRUE(nrow(x) > 0))]
```

```
makeDF.Coordinates <- function(dataset_lst){

  pull_location_str <- function(dataset, for_URL){
    # for_URL is a boolean indicating if pulling the location_str
    # for a URL or not

    name <- dataset$dataset_name[1]

    if(for_URL){

      # extract lowercase letters of name
      check <- str_extract(name, "[a-z]+")
      state <- state.name[grep(str_sub(name, 1, 2), state.abb)]

      if(check == "statewide" | check == "state"){

        url_name <- gsub(" ", "+", state.name[grep(str_sub(name, 1, 2), state.abb)])
        # return full name of the state

        return(url_name)

      } else {

        url_name <- gsub(" ", "+", paste(check, state))
        return(url_name)

      }

    }

    # otherwise just turn dataset_name, as it is formatted in the dataset
    return(name)

  }

  location_str_URL <- lapply(time_lst, pull_location_str, TRUE)

  # scraper function to fetch coordinates from city string
  get_coordinates <- function(city){

    # city is a string

    url_str <- paste("http://www.google.com/search?q=latitude+and+longitude+of+", 
                    city, sep = "")

    doc <- htmlParse(getURL(url_str))

    # class = BNeawe iBp4i AP7Wnd retrieves the coordinates

  }

}
```

```

coordinates <- xpathSApply(doc, "//div[@class='BNeawe iBp4i AP7Wnd']", xmlValue)[1]

clean_coordinates <- str_split(coordinates, ", ")[[1]]

# use regular expressions to extract lat and lng
lat <- as.numeric(str_extract(clean_coordinates[1], "\\\d+\\.\\.*\\\\d*"))
# multiple long by -1 b/c...?
long <- -1*as.numeric(str_extract(clean_coordinates[2], "\\\d+\\.\\.*\\\\d*"))

final_coordinates <- c(lat, long)
return(final_coordinates)

}

# find lat/long information using get_coordinates function and relevant_dataset_names df
coordinates_lst <- lapply(location_str_URL, get_coordinates)

# bind lat/long information with relevant_dataset_names
coordinates_df <- data.table::transpose(data.frame(coordinates_lst)) %>%
  # join with dataset_names
  bind_cols(data.table::transpose(data.frame(lapply(time_lst, pull_location_str, FALSE)))) %>%
  rename(lat = V1..., lng = V2, dataset_name = V1..., 3)

return(coordinates_df)

}

coordinates_df <- makeDF.Coordinates(time_lst)

```

Find the lat/lng for each dataset (city and/or state) (can we use lat/lng from other packages like tigris... other geolocators probably exist)

```

# helper function for add_day_night, returns minute of the day
time_to_minute <- function(time){
  # time can be str
  lubridate::hour(hms(time)) * 60 + lubridate::minute(hms(time))
}

# calculate sunset, sunrise, dusk, and dawn times
oursunriseset <- function(latitude, longitude, date, timezone, direction) {

  date.lat.long <- data.frame(date = date, lat = latitude, lon = longitude)

  if(direction == "sunset"){
    # call getSunlightTimes from the lutz package
    getSunlightTimes(data = date.lat.long, keep=direction, tz=timezone)$sunset

  } else if(direction == "sunrise"){
    getSunlightTimes(data = date.lat.long, keep=direction, tz=timezone)$sunri

  } else if (direction == "dusk"){


```

```

getSunlightTimes(data = date.lat.long, keep=direction, tz=timezone)$dusk

} else if (direction == "dawn"){
  getSunlightTimes(data = date.lat.long, keep=direction, tz=timezone)$dawn
}
}

add_day_night <- function(dataset, coord_df){

  dataset_name_str <- dataset$dataset_name[1]

  # filter for lat/long information of dataset
  coord_df <- coord_df %>% filter(dataset_name == dataset_name_str)

  # initialize lat, long, tz information
  lat <- coord_df$lat[1]
  lng <- coord_df$lng[1]
  time_zone <- lutz::tz_lookup_coords(lat, lng, warn = F)

  # use lubridate to clean date type in dataset
  dataset <- dataset %>%
    filter(!is.na(time) & !is.na(date)) %>%
    mutate(date = as.Date(ymd(date, tz = time_zone)),
      stop_minute = time_to_minute(time))

  # df for dawn, sunrise, sunset, and dusk times for distinct dates
  sunriseset_times <- dataset %>%
    filter(!is.na(date)) %>%
    distinct(date) %>%
    mutate(dawn_hms = format(oursunriseset(lat, lng, date,
                                              time_zone, direction = "dawn"),
                             "%H:%M:%S"),
      sunrise_hms = format(oursunriseset(lat, lng, date,
                                              time_zone, direction = "sunrise"),
                             "%H:%M:%S"),
      sunset_hms = format(oursunriseset(lat, lng, date,
                                              time_zone, direction = "sunset"),
                             "%H:%M:%S"),
      dusk_hms = format(oursunriseset(lat, lng, date,
                                              time_zone, direction = "dusk"),
                             "%H:%M:%S"),
      dawn = time_to_minute(dawn_hms),
      sunrise = time_to_minute(sunrise_hms),
      sunset = time_to_minute(sunset_hms),
      dusk = time_to_minute(dusk_hms))

  # filter dataset for stops occurring during window_of_stop
  dataset <- dataset %>%
    left_join(sunriseset_times, by = "date") %>%
    mutate(day_night = case_when(stop_minute >= dawn &
                                stop_minute < sunrise ~ "twilight morning",
                                stop_minute >= sunrise &
                                stop_minute < sunset ~ "day",

```

```

        stop_minute >= sunset &
        stop_minute < dusk ~ "twilight evening",
        stop_minute >= dusk |
        stop_minute < dawn ~ "night",
        TRUE ~ "missing"))

}

sunriseset_lst <- lapply(time_lst, add_day_night, coordinates_df)

```

Add day/night/twilight variable We can visualize the newly added day and night variables by plotting the stop minute over the date and color code by the labels day, night, and twilight. The following plot is for San Bernadino.

```
ggplot(data = sunriseset_lst[[1]]) +
  geom_point(aes(x = date, y = stop_minute, color = day_night), alpha = .1)
```

Time analysis

```

summarizeSMRDayNight <- function(dataset){
  # calculate average SMR per month for day and night
  # find_diff_bool indicates if we want to SPREAD the dataset

  dataset <- dataset %>%
    mutate(nice_month = ymd(cut(ymd(date), "month")))) %>%
    group_by(dataset_name, nice_month, day_night) %>%
    # filter out stops occurring during twilight
    filter(str_detect(day_night, "twilight", negate = TRUE)) %>%
    summarize(avg_SMR = mean(stop_missing_rate), .groups = "drop")

  return(dataset)
}

makeDF.SMRTIME <- function(sunriseset_list, find_diff_bool){

  # filter away the helper columns like stop_minute and dusk_hms
  sunriseset_list <- lapply(sunriseset_list, myfilter_for, c(freq_var, "day_night"), FALSE)

  # count NA
  sunriseset_list <- lapply(sunriseset_list, countMissing, 1, TRUE,
                            c("time", "day_night", "date"))

  # summarize SMR per month by day and night
  sunriseset_list <- lapply(sunriseset_list, summarizeSMRDayNight)

  sunriseset_df <- bind_rows(sunriseset_list)

  if (find_diff_bool){

    sunriseset_df <- sunriseset_df %>%
      spread(key = day_night, value = avg_SMR) %>%
      mutate(diff_SMR = day - night)
  }
}
```

```

}

return(sunriseset_df)

}

sunriseset_df <- makeDF.SMRTIME(sunriseset_lst, FALSE)
sunrisesetdiff_df <- makeDF.SMRTIME(sunriseset_lst, TRUE)

ggplot_sunriseset <- function(sunriseset_dataset, variable_str){

  plot_avg <- function(name){

    p <- sunriseset_dataset %>%
      filter(dataset_name == name) %>%
      ggplot(aes(x = nice_month, y = avg_SMR)) +
      geom_point() +
      facet_wrap(~ day_night) +
      ggtitle(paste("Average monthly SMR of", name)) +
      labs(x = "Date", y = "Average SMR")

    ggsave(paste(name, "SMR by day and night.png"), p)

    return(p)
  }

  plot_diff <- function(name){

    p <- sunriseset_dataset %>%
      filter(dataset_name == name) %>%
      ggplot(aes(x = nice_month, y = diff_SMR)) +
      geom_point() +
      ggtitle(paste("Difference in SMR of", name,
                    "Between Day and Night per month")) +
      labs(x = "Date", y = "Difference in SMR")

    ggsave(paste(name, "SMR day and night.png"), p)

    return(p)
  }

  dataset_name <- sunriseset_dataset %>% distinct(dataset_name) %>% pull(dataset_name)

  if (variable_str == "avg_SMR"){
    plots <- lapply(dataset_name, plot_avg)

  } else if (variable_str == "diff_SMR") {
    plots <- lapply(dataset_name, plot_diff)

  } else {stop("typo in variable_str!")}
}

```

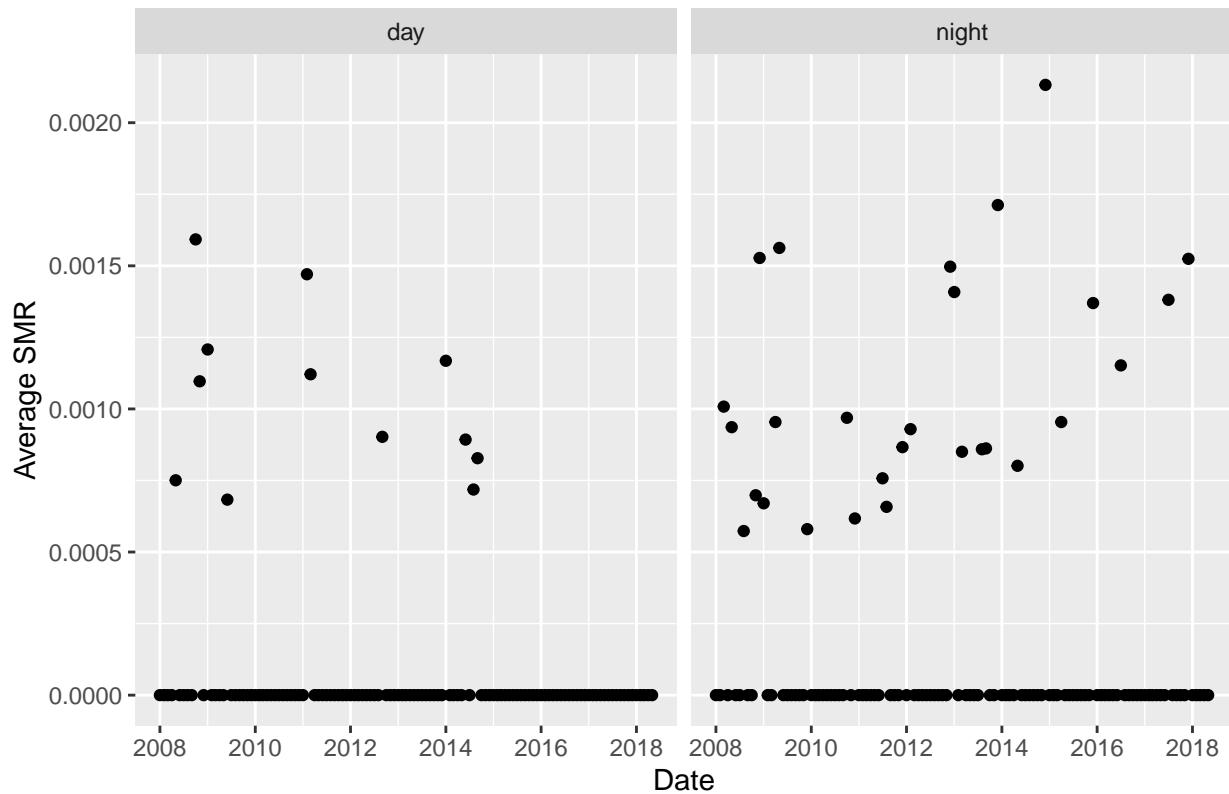
```

    return(plots)
}

ggplot_sunriseset(sunriseset_df, "avg_SMR")
## [[1]]

```

Average monthly SMR of AZgilbert

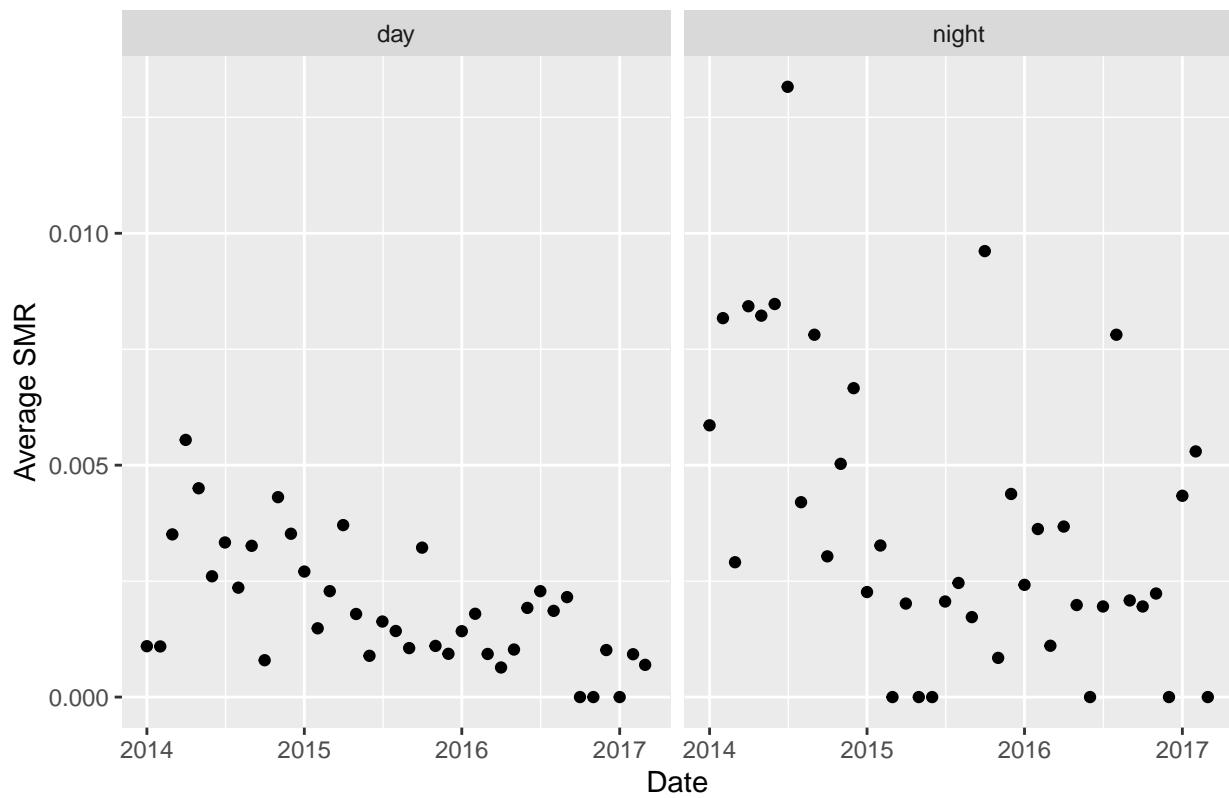


```

## 
## [[2]]

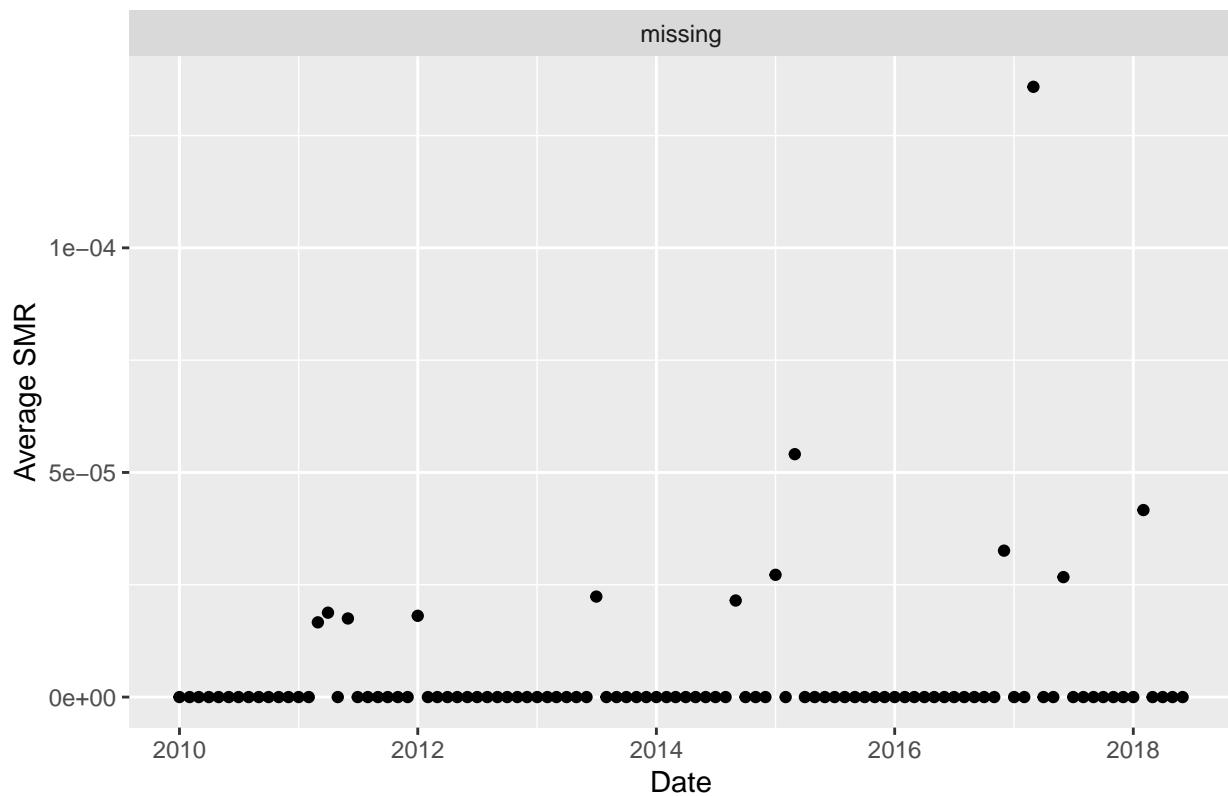
```

Average monthly SMR of AZmesa



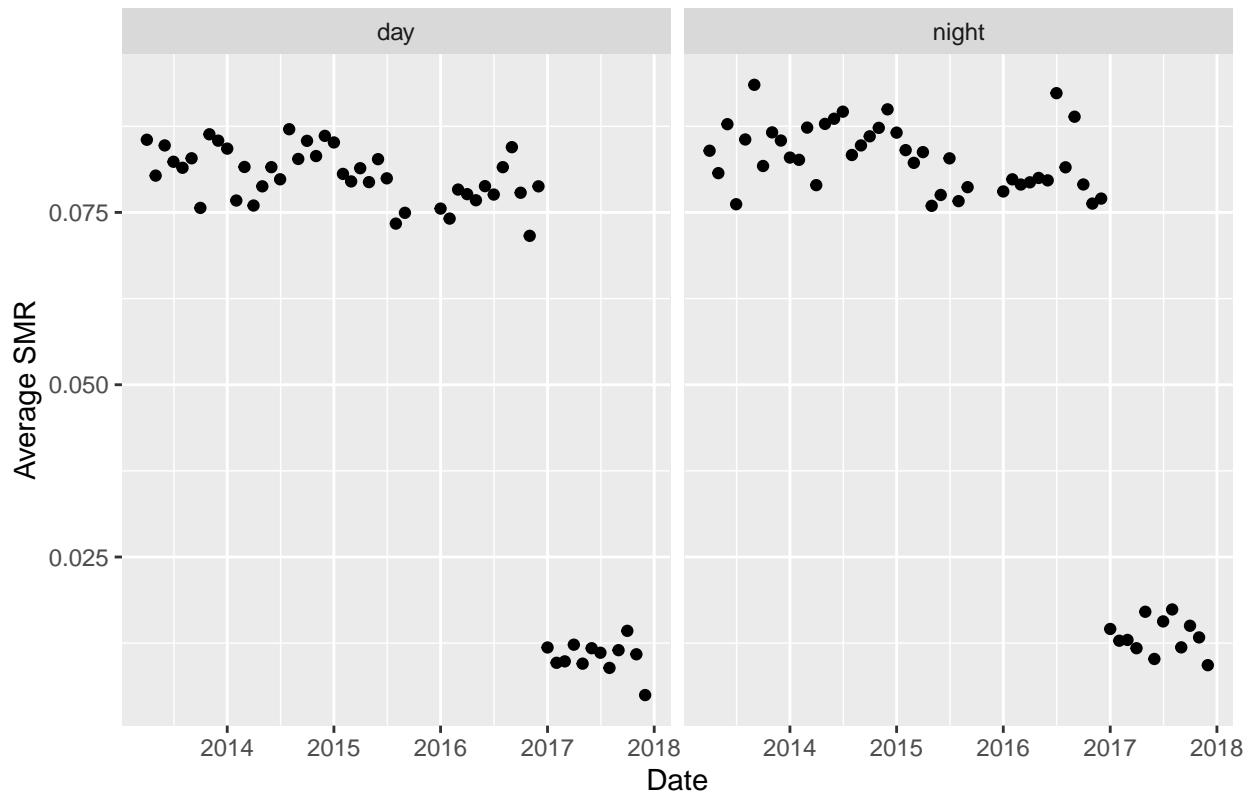
```
##  
## [[3]]
```

Average monthly SMR of CAlosangeles



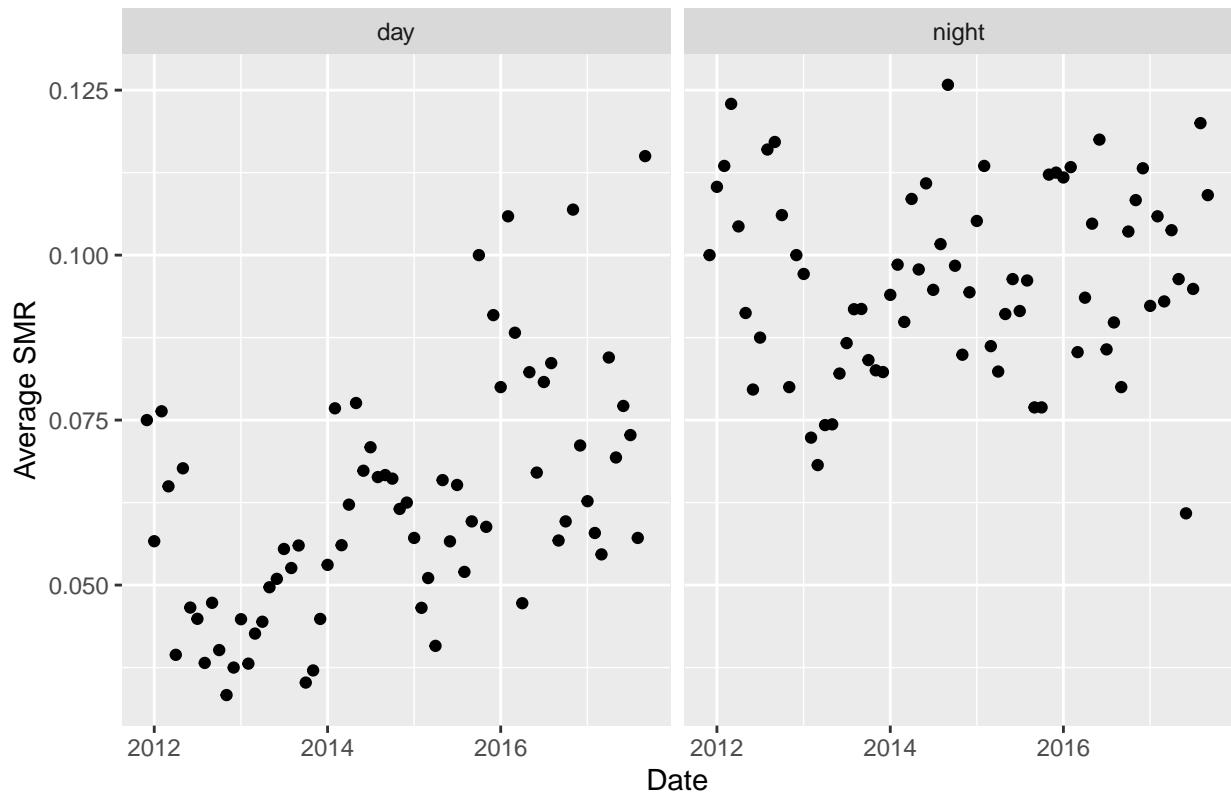
```
##  
## [[4]]
```

Average monthly SMR of CAoakland



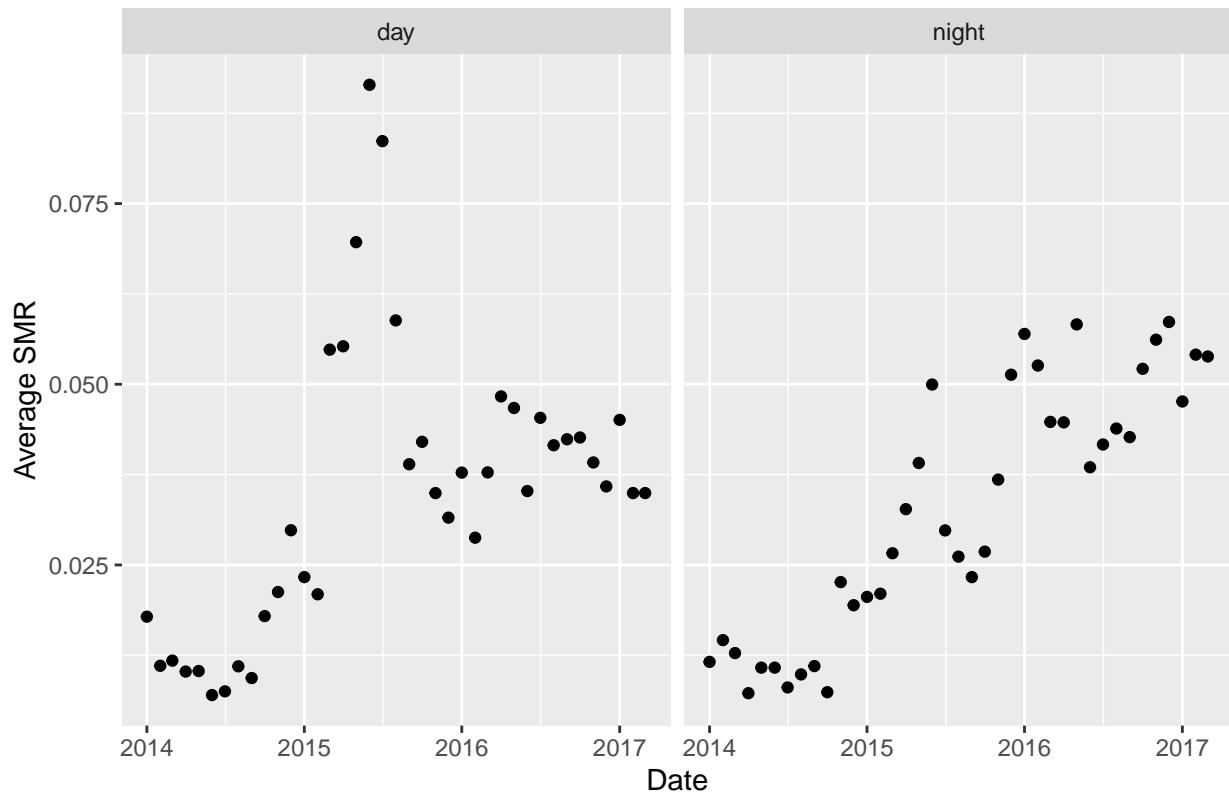
```
##  
## [[5]]
```

Average monthly SMR of CAsanbernardino



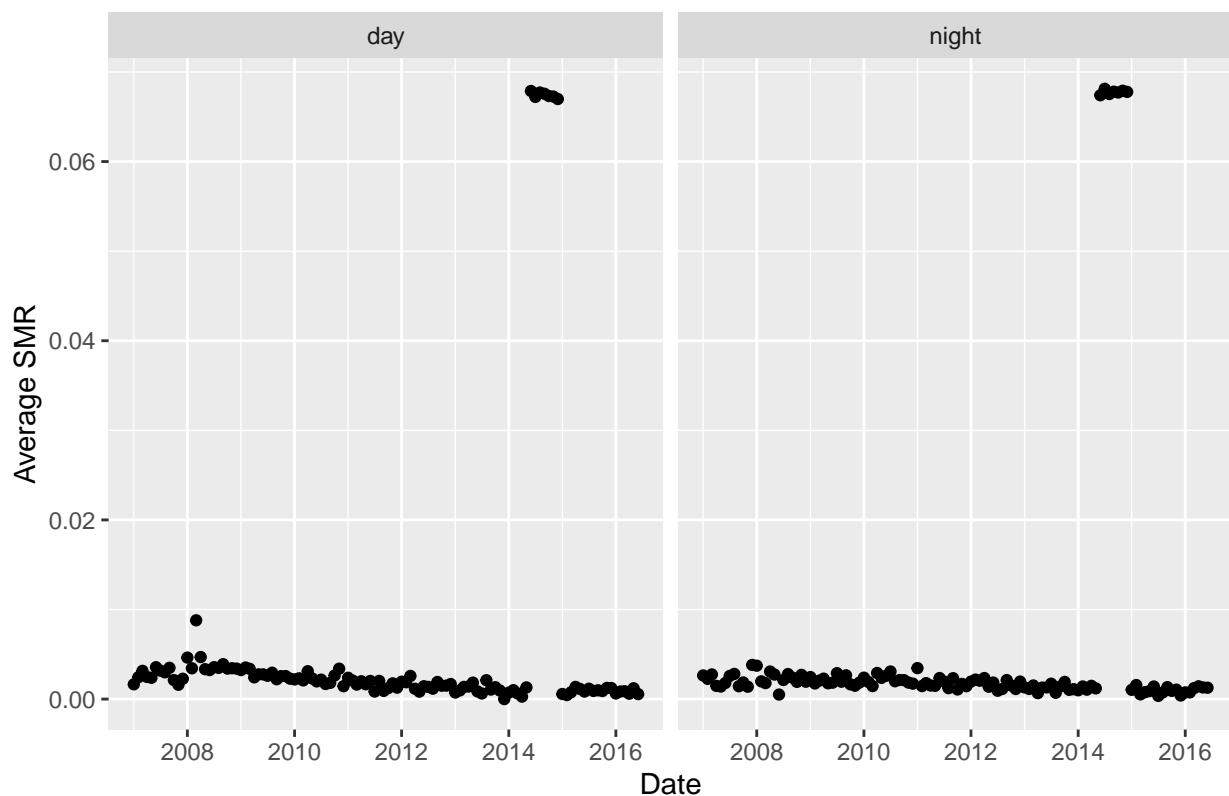
```
##  
## [[6]]
```

Average monthly SMR of CAsandiego



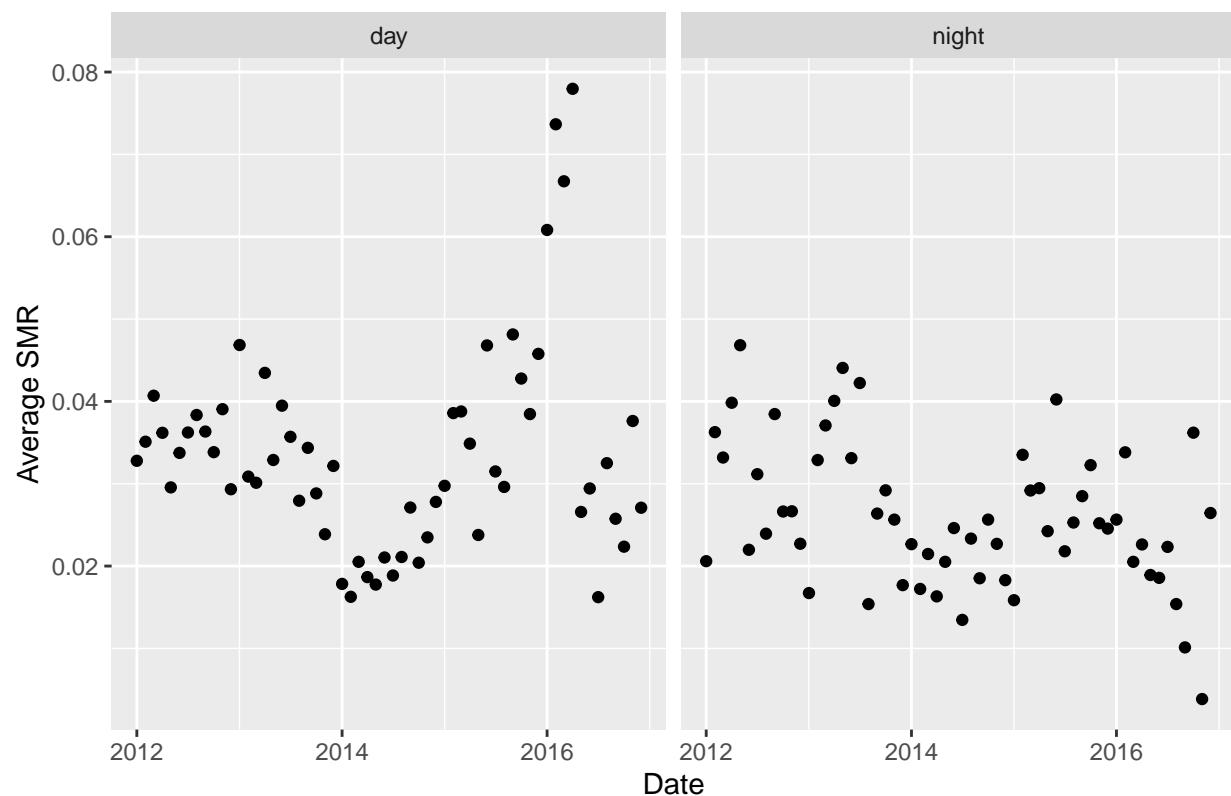
```
##  
## [[7]]
```

Average monthly SMR of CAsanfrancisco



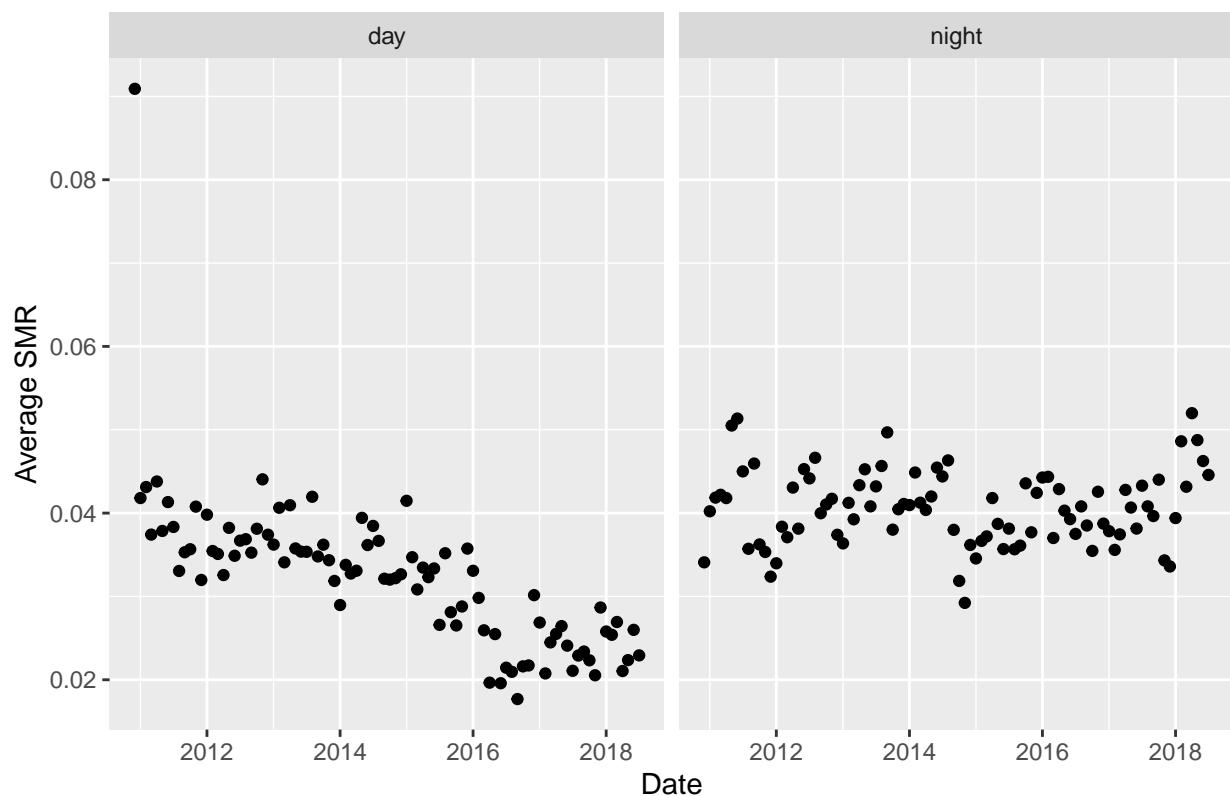
```
##  
## [[8]]
```

Average monthly SMR of COaurora



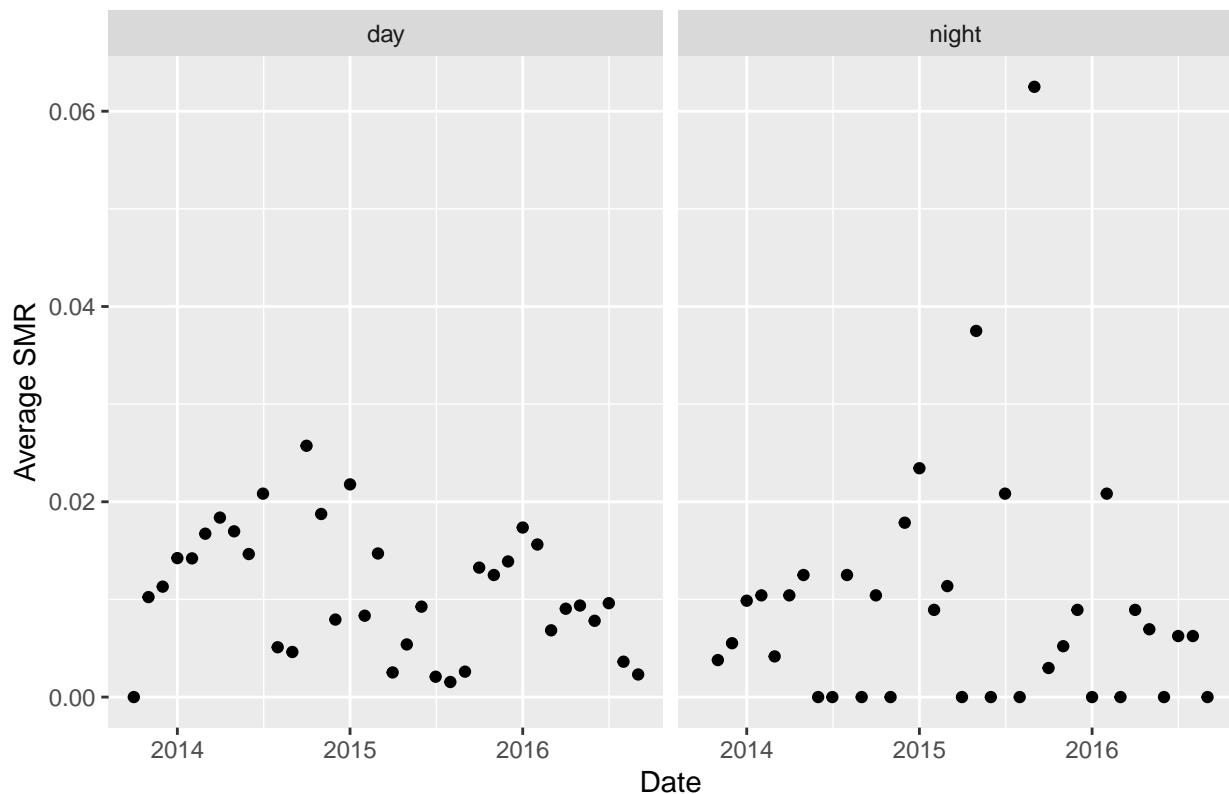
```
##  
## [[9]]
```

Average monthly SMR of COdenver



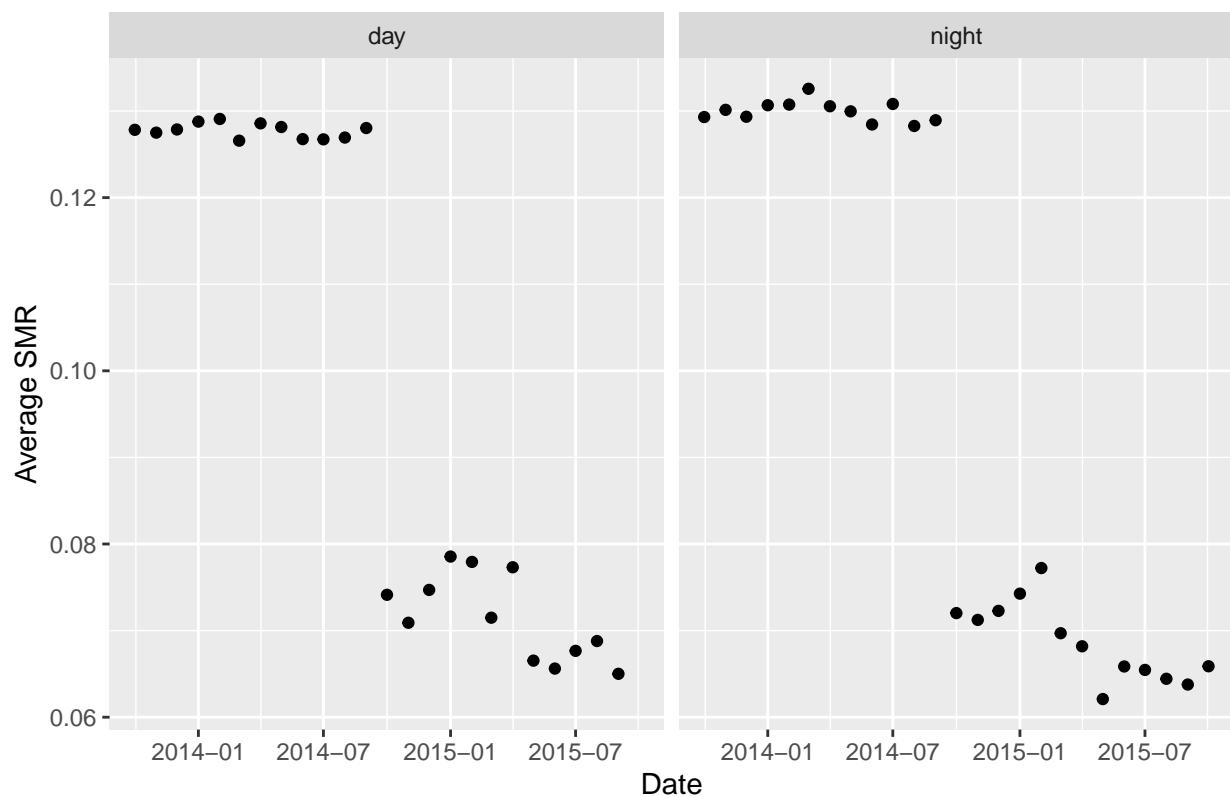
```
##  
## [[10]]
```

Average monthly SMR of CThartford



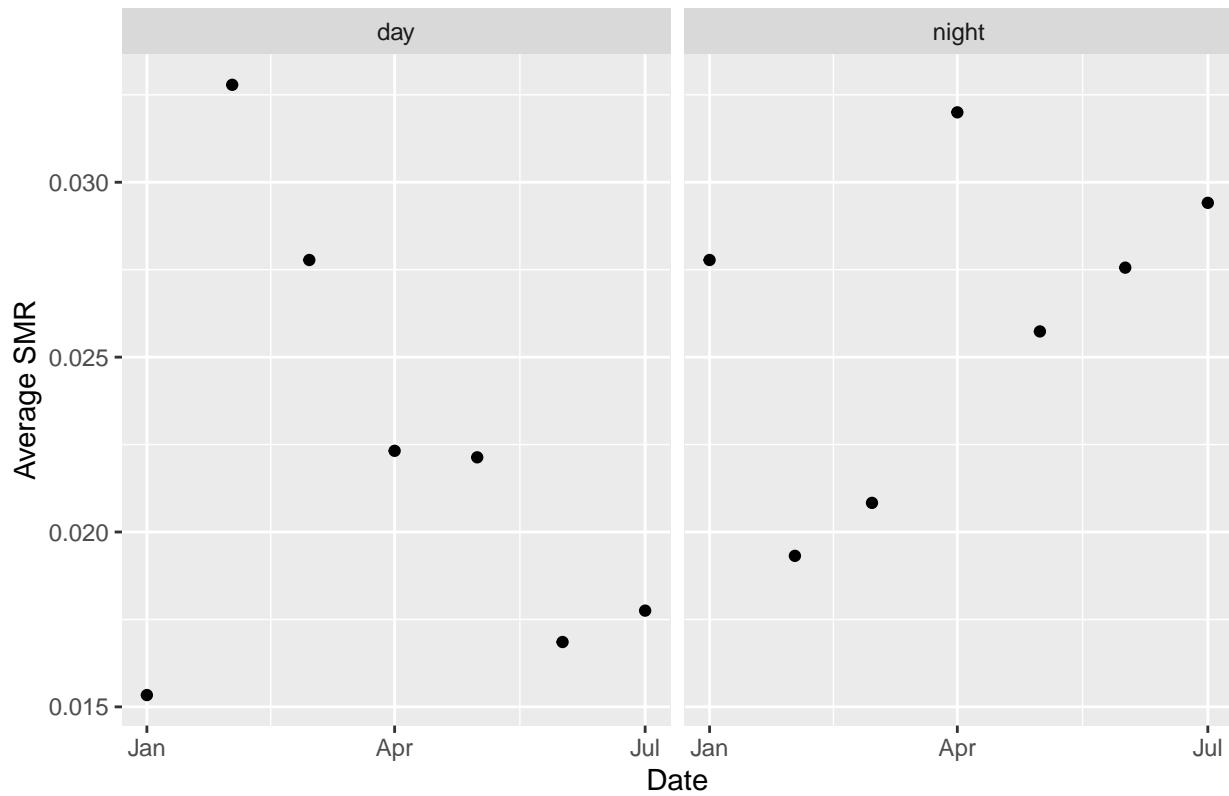
```
##  
## [[11]]
```

Average monthly SMR of CT statewide



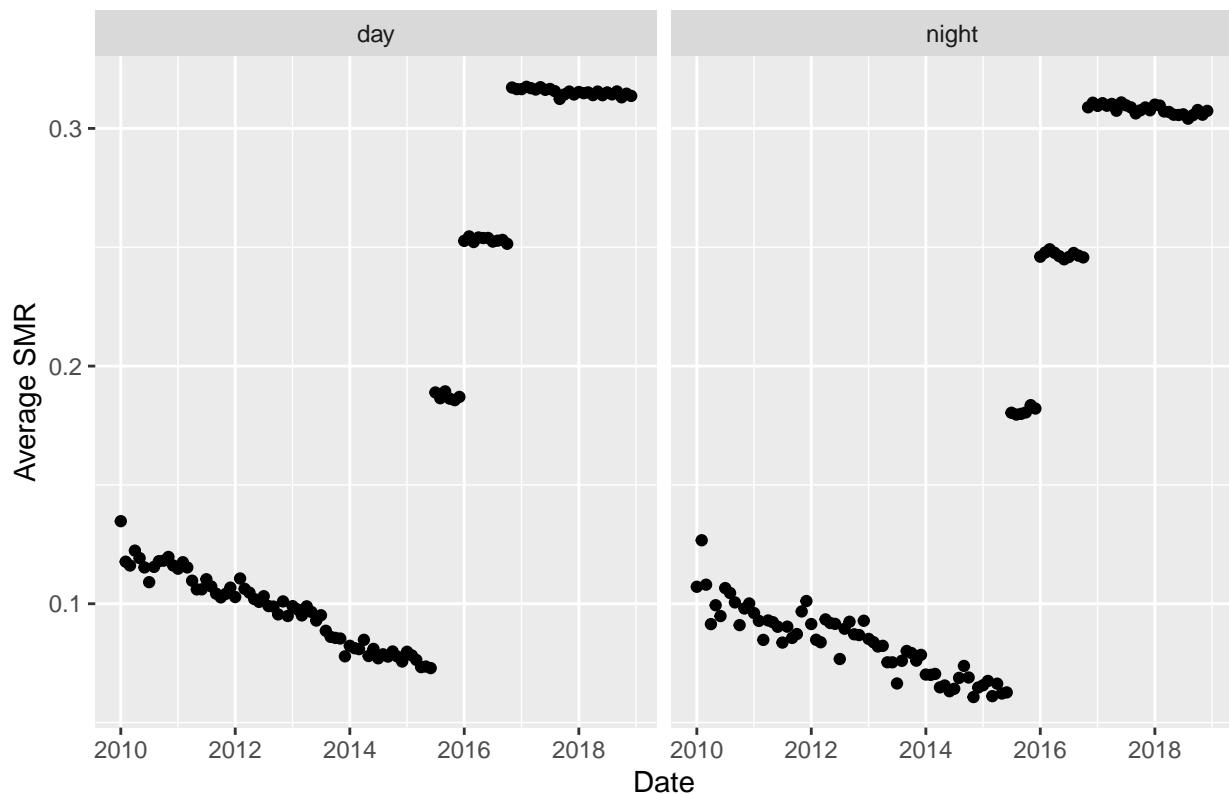
```
##  
## [[12]]
```

Average monthly SMR of FLsaint



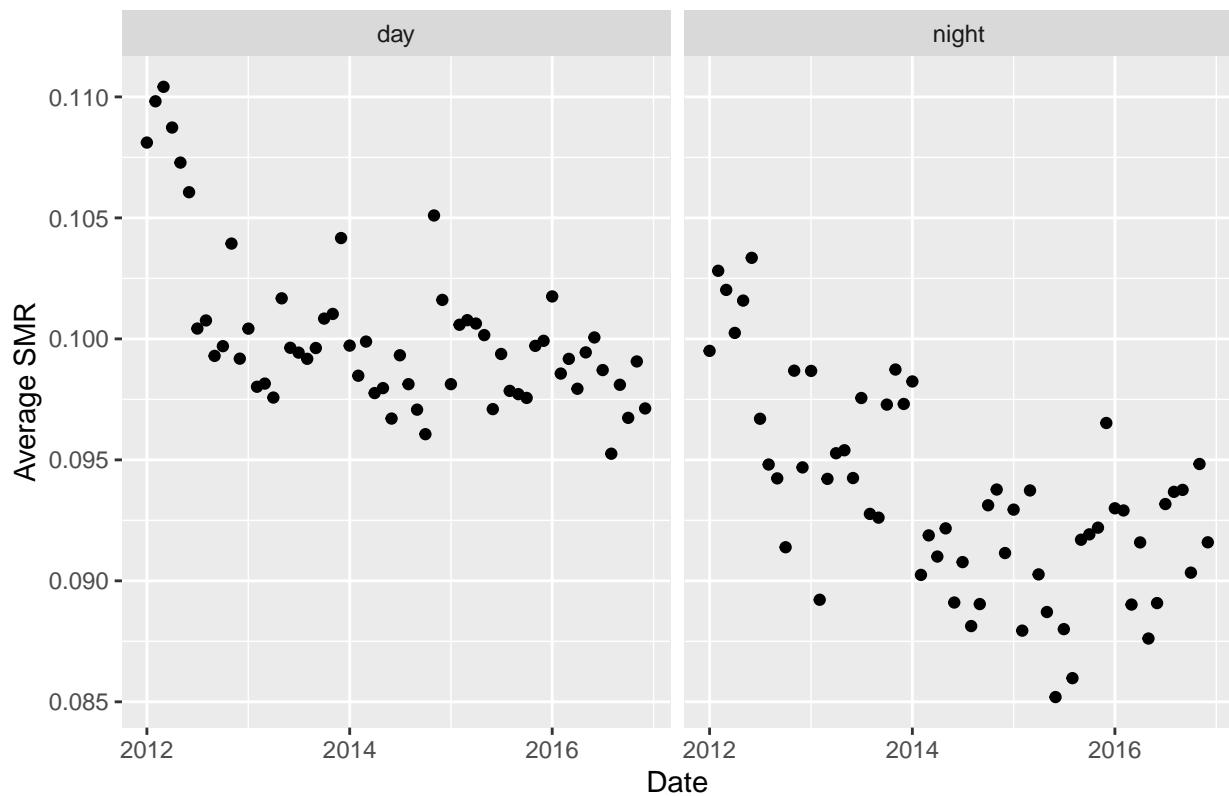
```
##  
## [[13]]
```

Average monthly SMR of FLstatewide



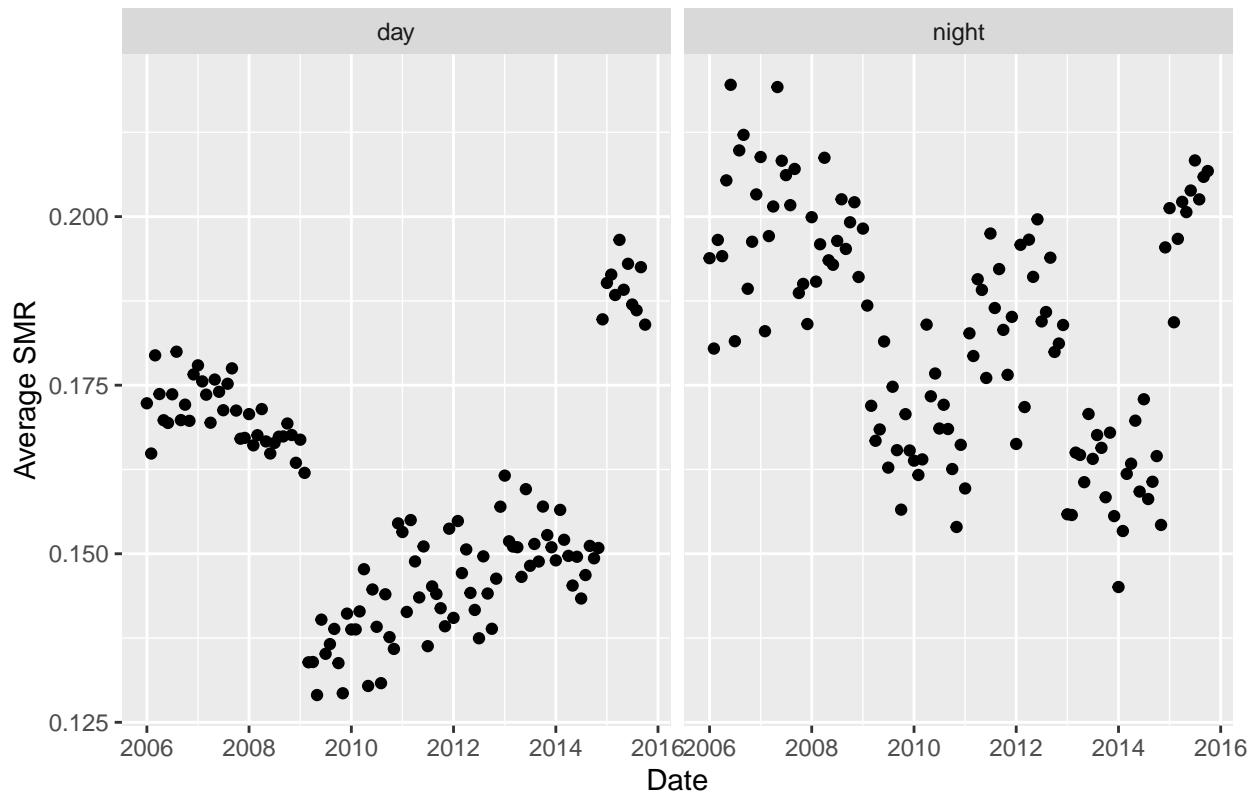
```
##  
## [[14]]
```

Average monthly SMR of GA statewide



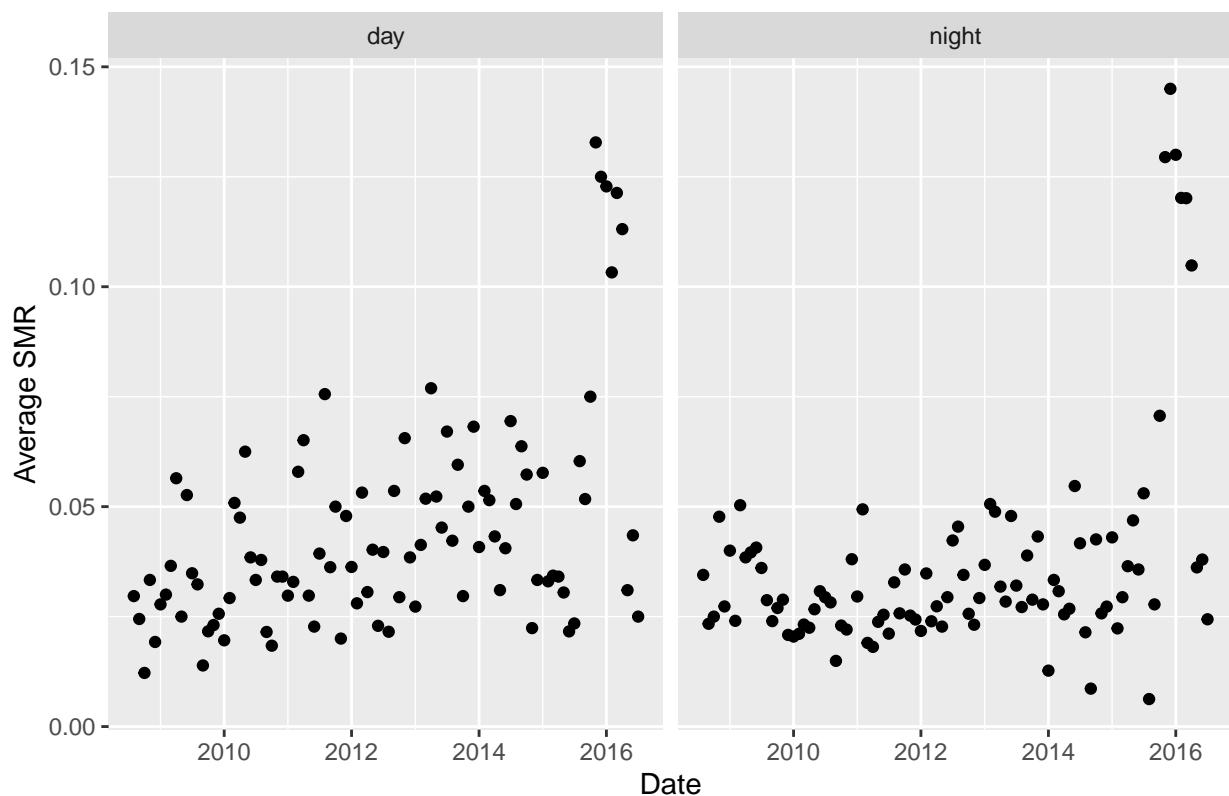
```
##  
## [[15]]
```

Average monthly SMR of IA statewide



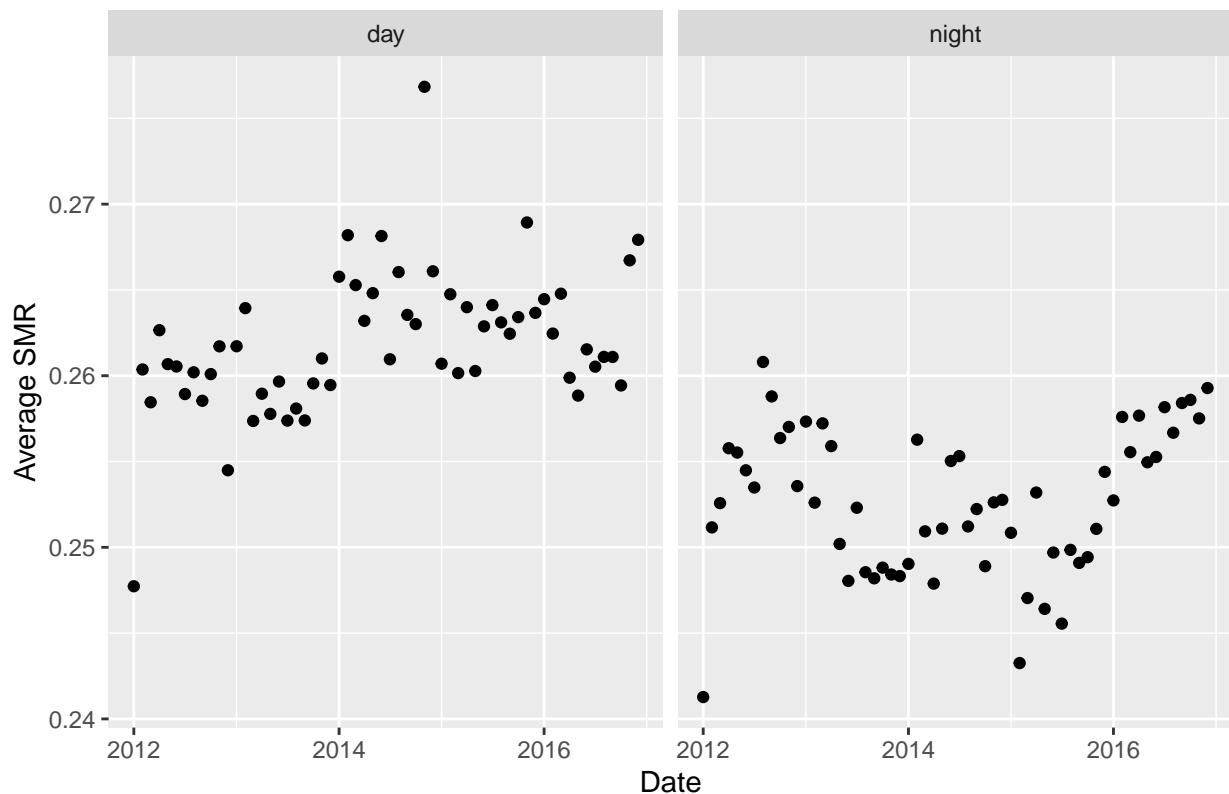
```
##  
## [[16]]
```

Average monthly SMR of IDidahofalls



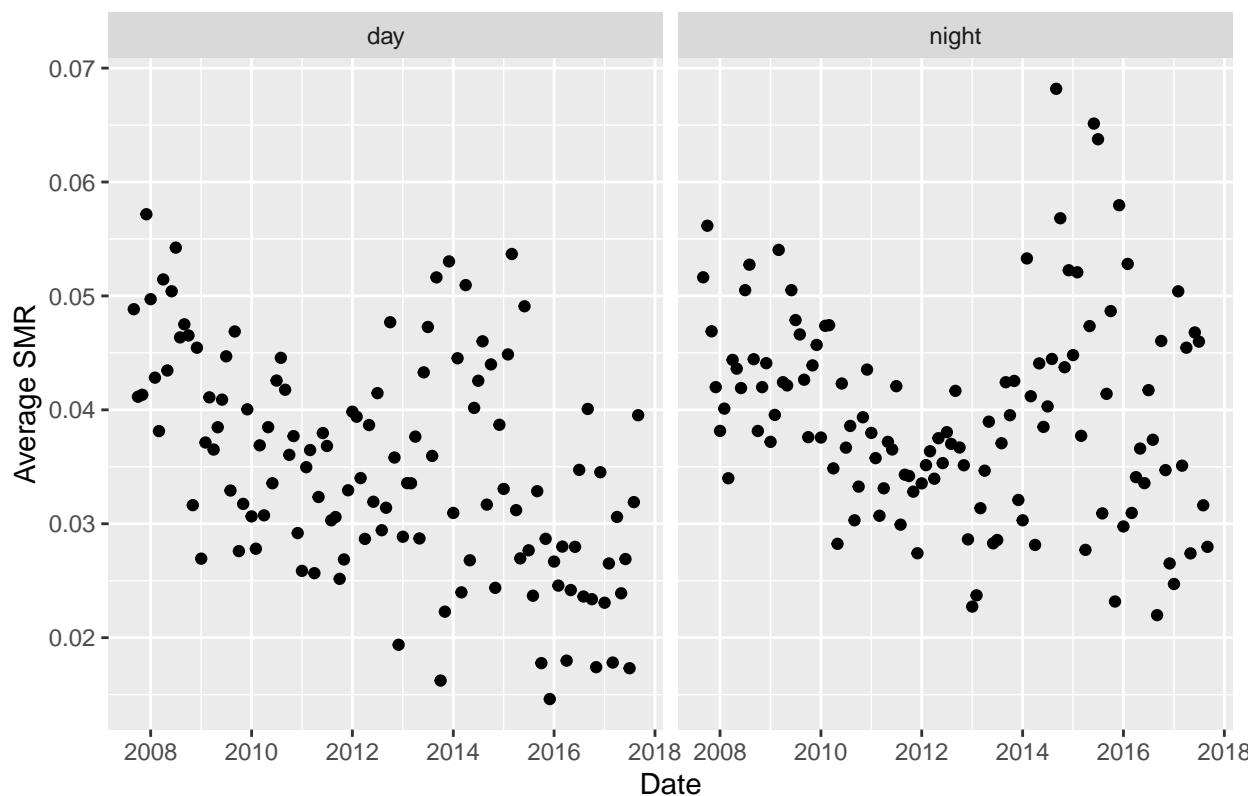
```
##  
## [[17]]
```

Average monthly SMR of ILchicago



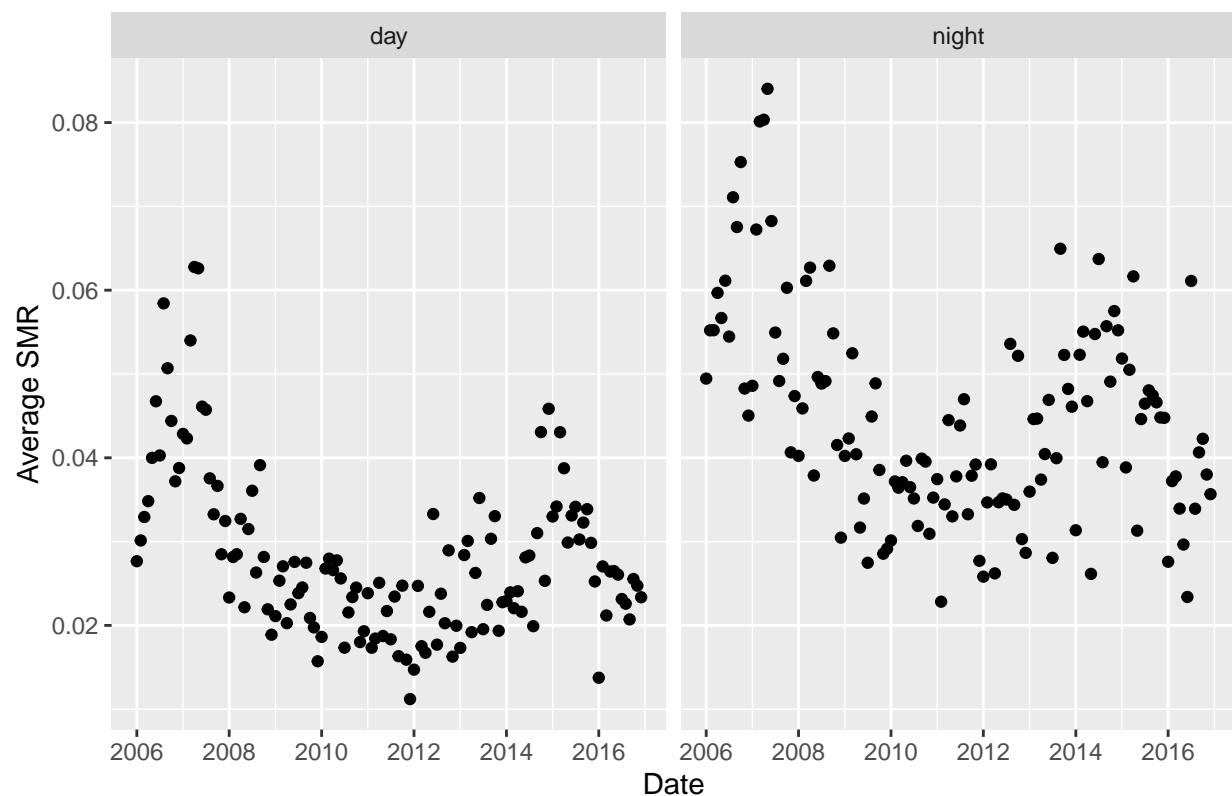
```
##  
## [[18]]
```

Average monthly SMR of INfortwayne



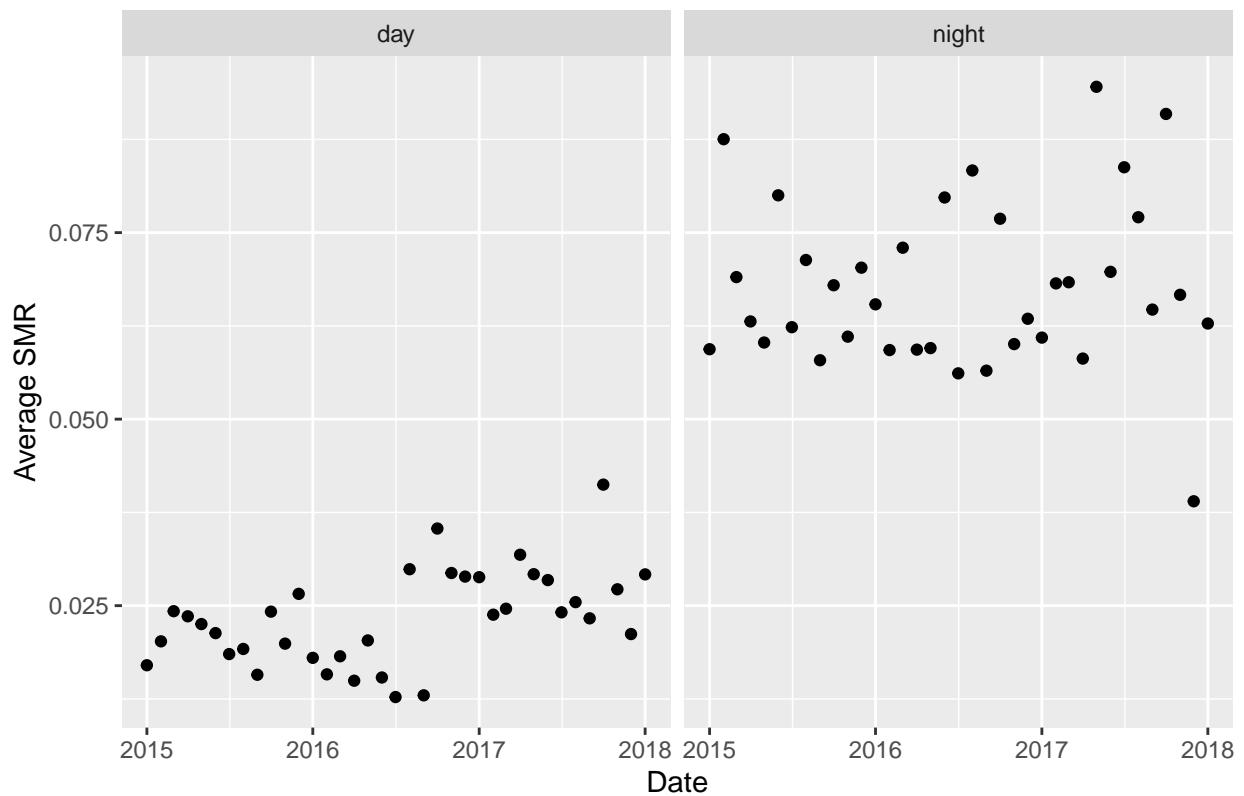
```
##  
## [[19]]
```

Average monthly SMR of KSwichita



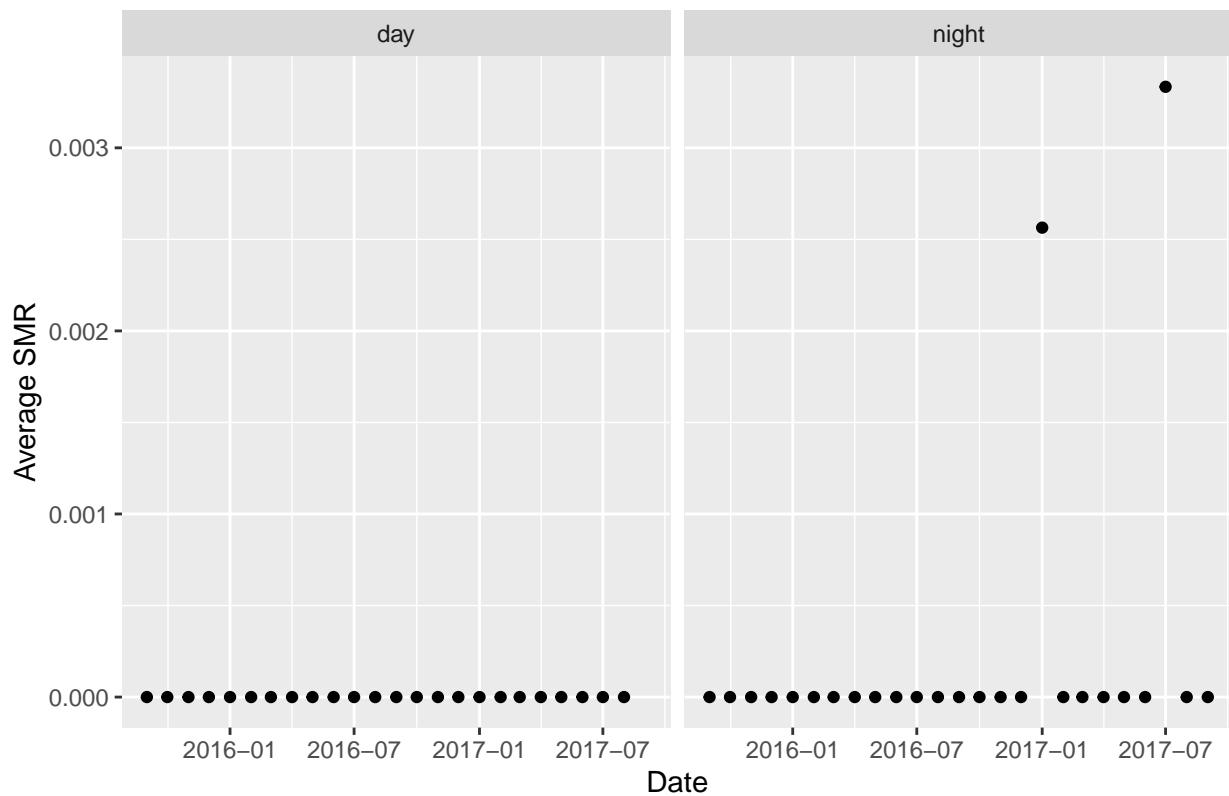
```
##  
## [[20]]
```

Average monthly SMR of KYlouisville



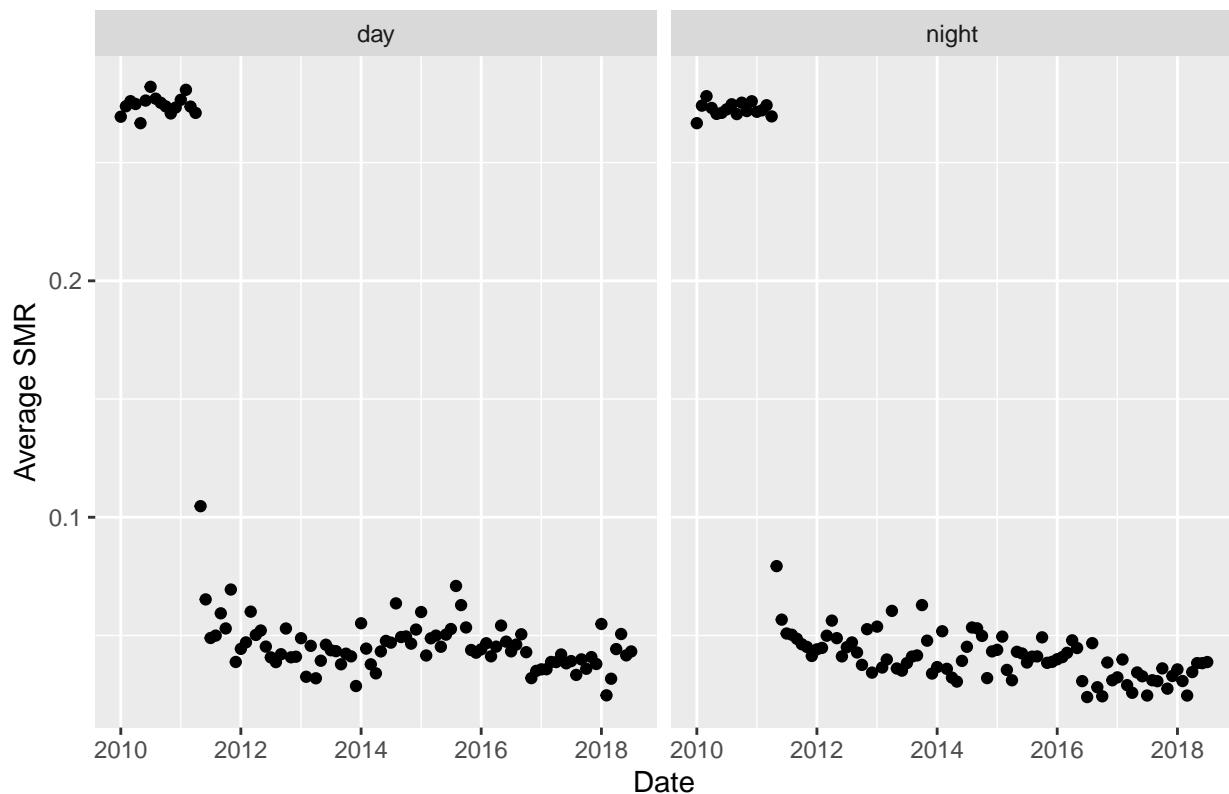
```
##  
## [[21]]
```

Average monthly SMR of KYowensboro



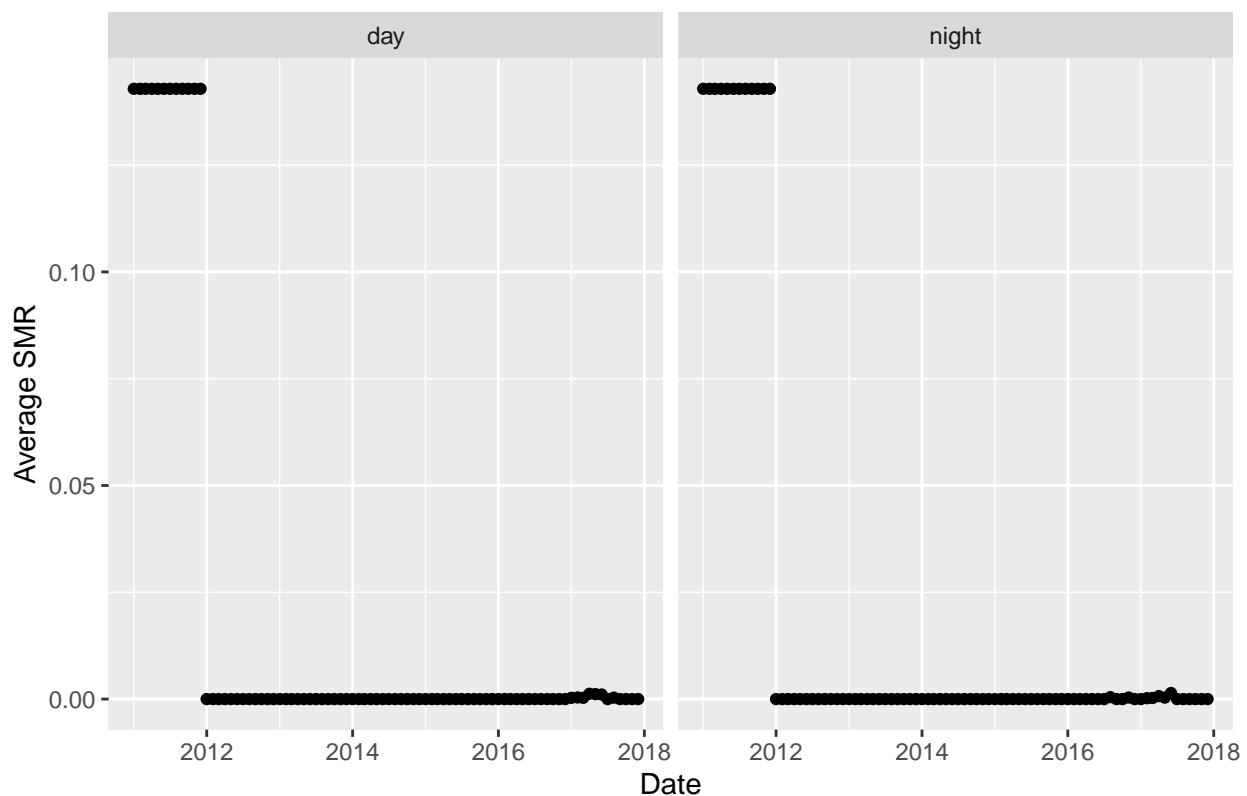
```
##  
## [[22]]
```

Average monthly SMR of LAneworleans



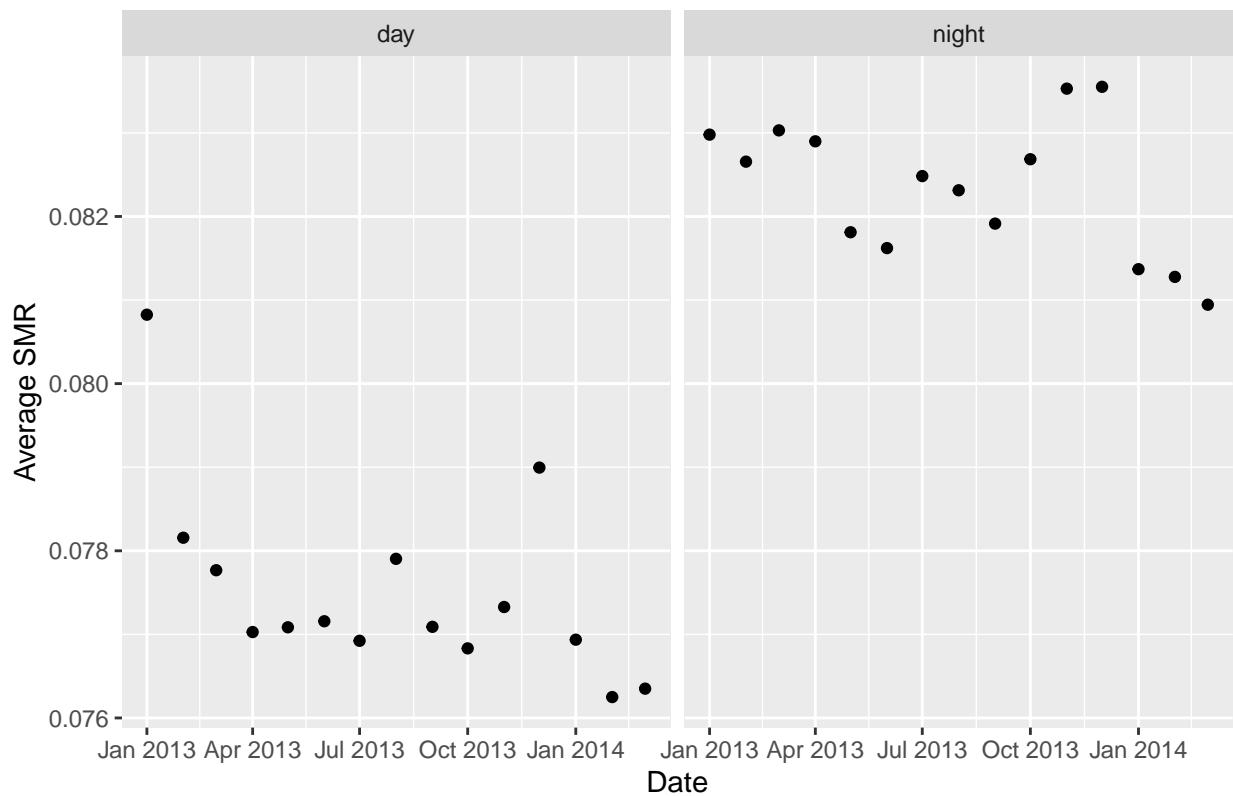
```
##  
## [[23]]
```

Average monthly SMR of MDbaltimore



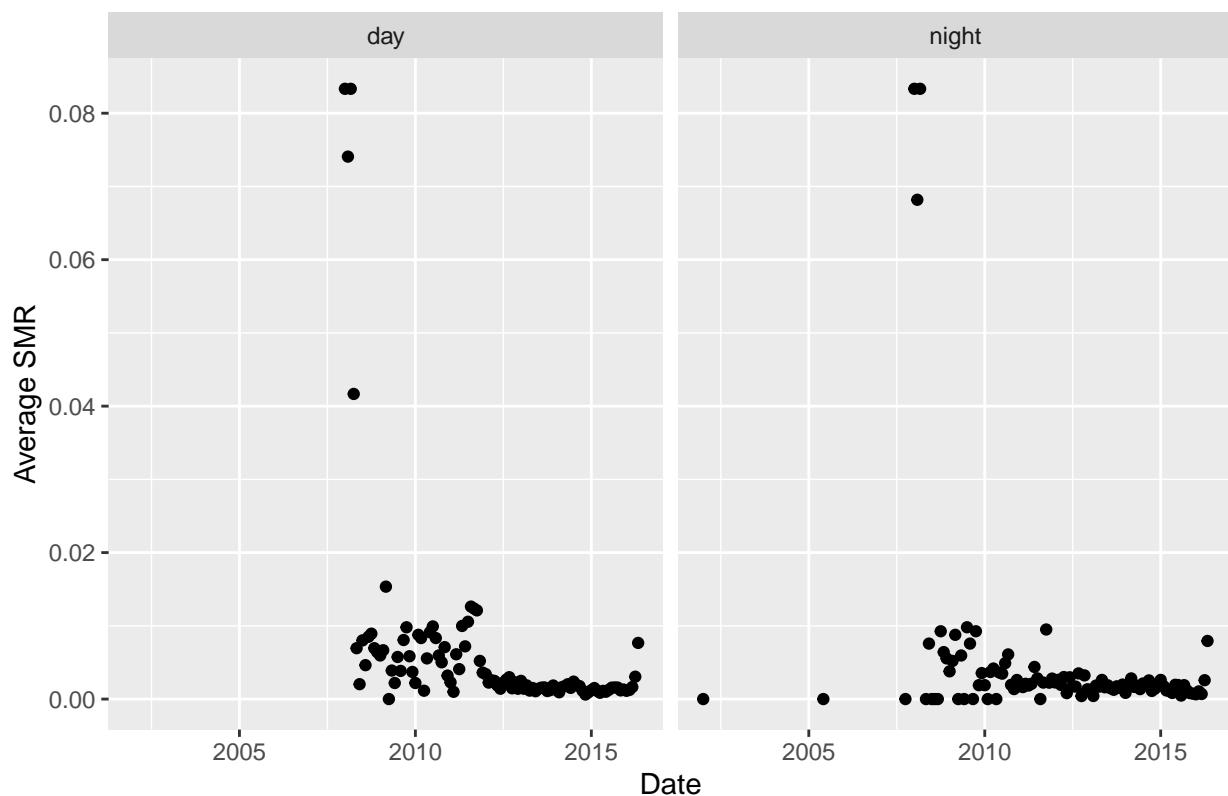
```
##  
## [[24]]
```

Average monthly SMR of MDstatewide



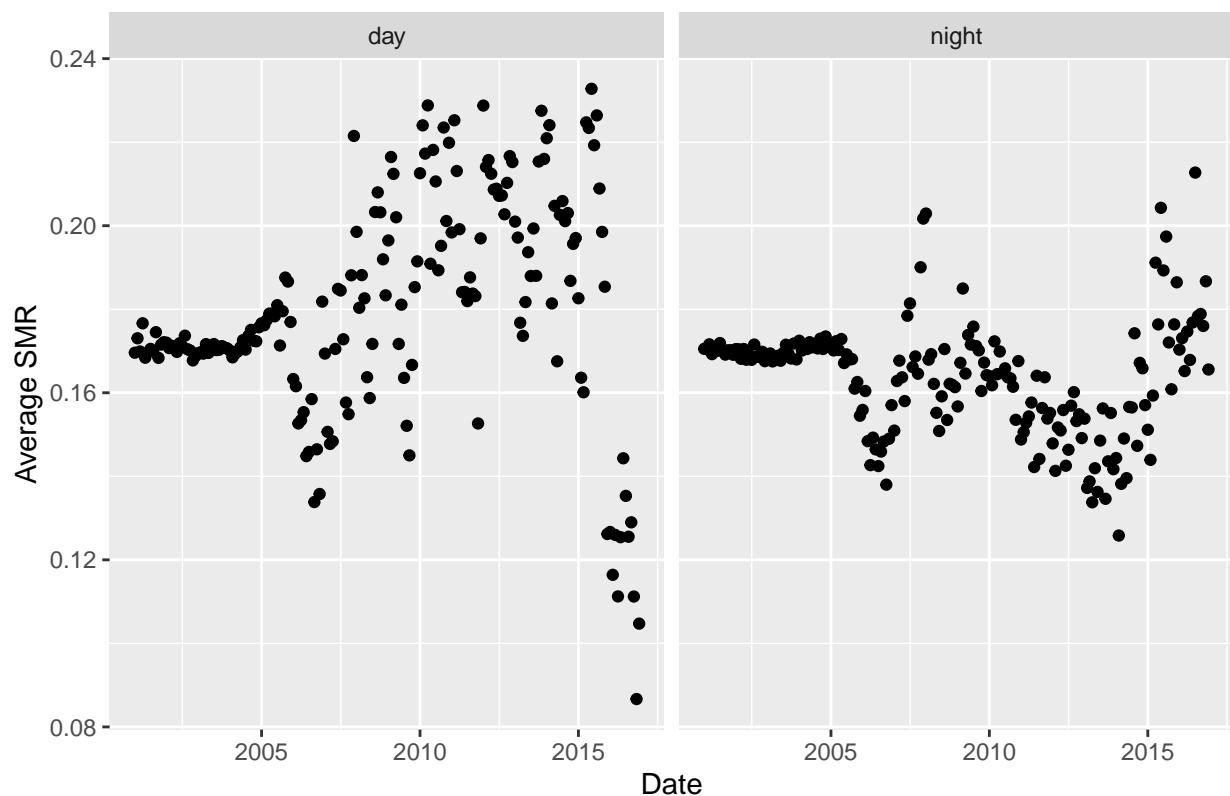
```
##  
## [25]
```

Average monthly SMR of MI statewide



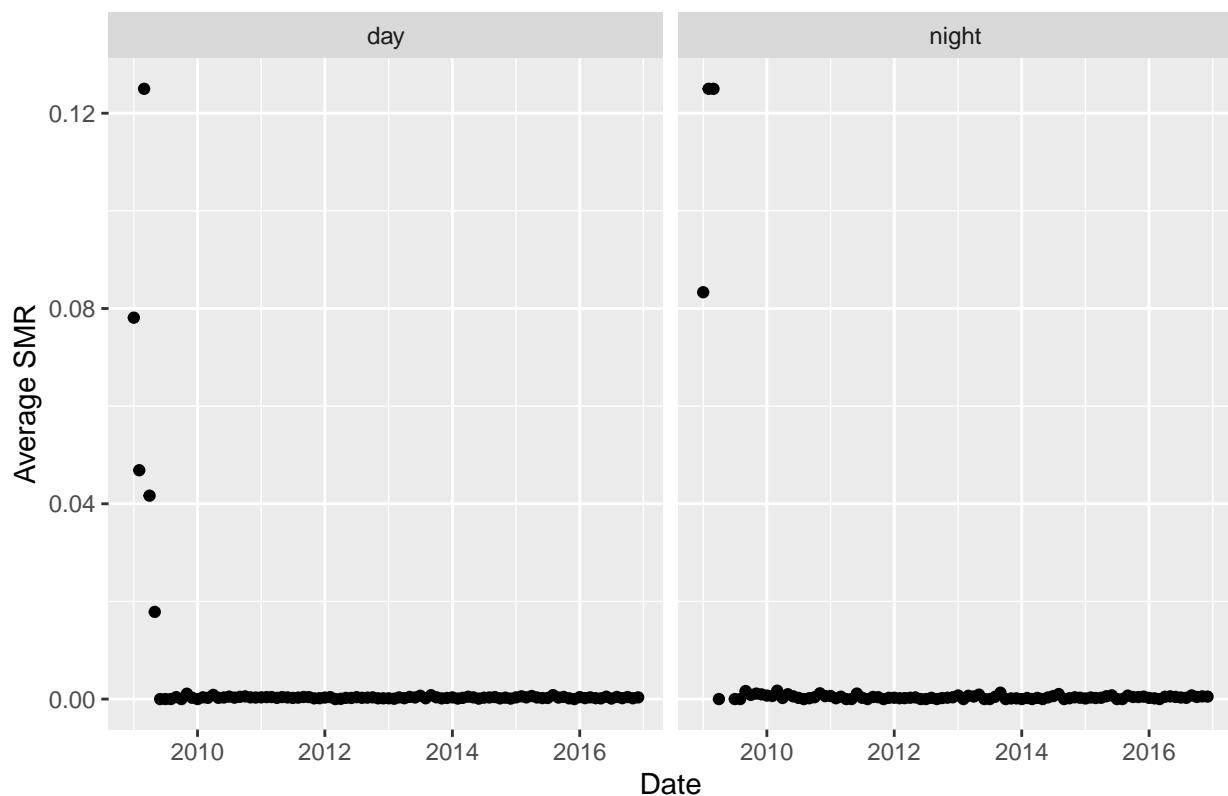
```
##  
## [[26]]
```

Average monthly SMR of MNsaintpaul



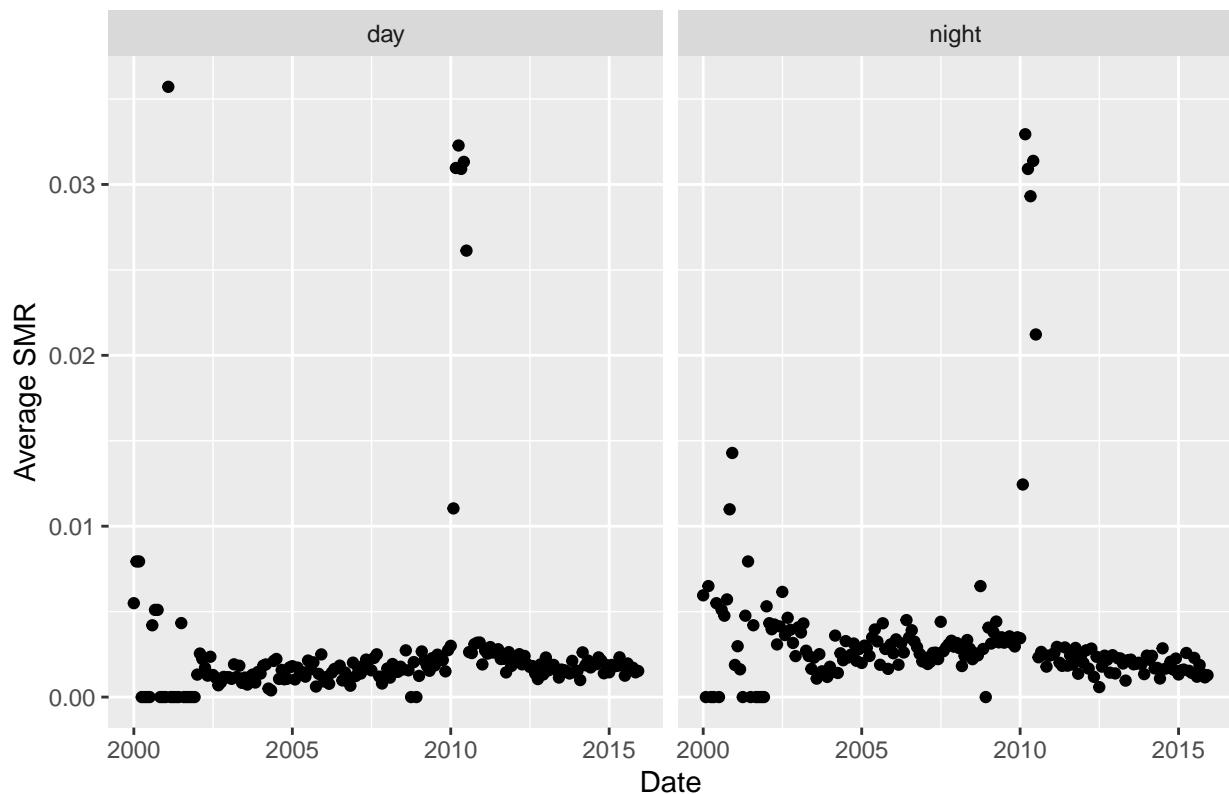
```
##  
## [[27]]
```

Average monthly SMR of MT statewide



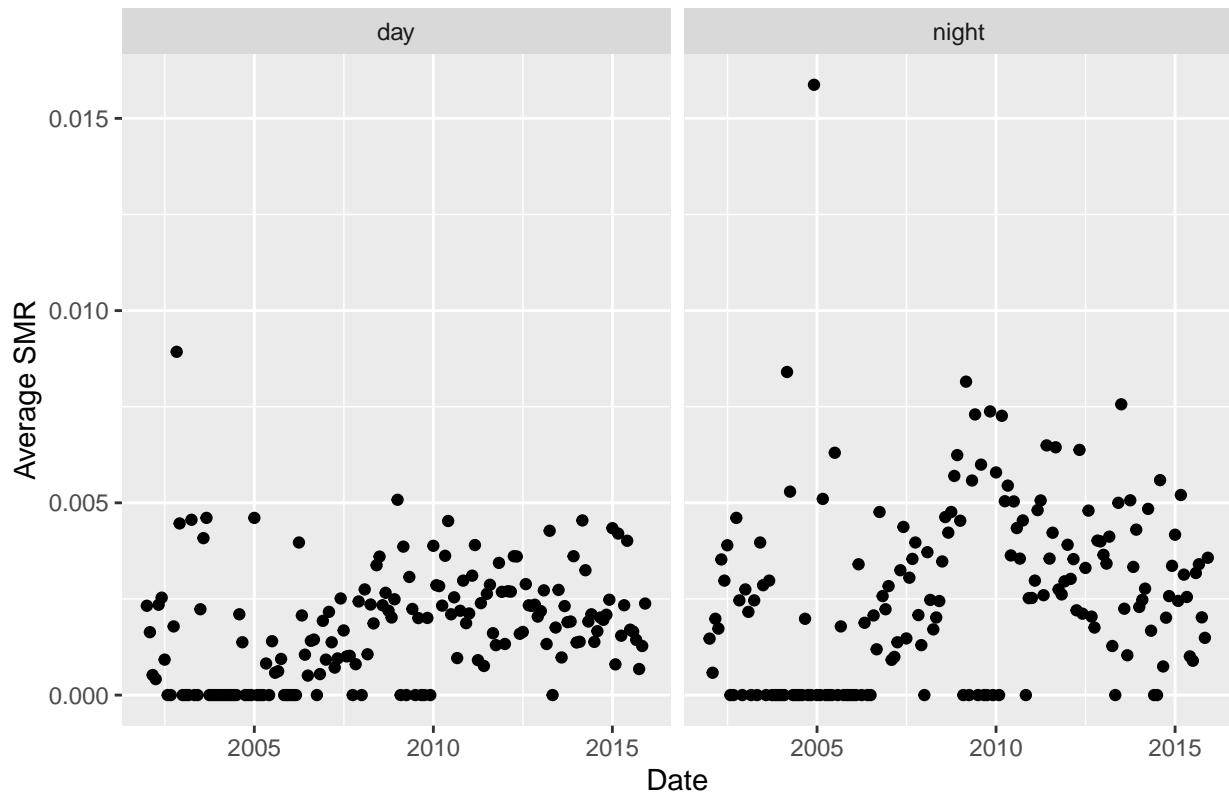
```
##  
## [[28]]
```

Average monthly SMR of NCcharlotte



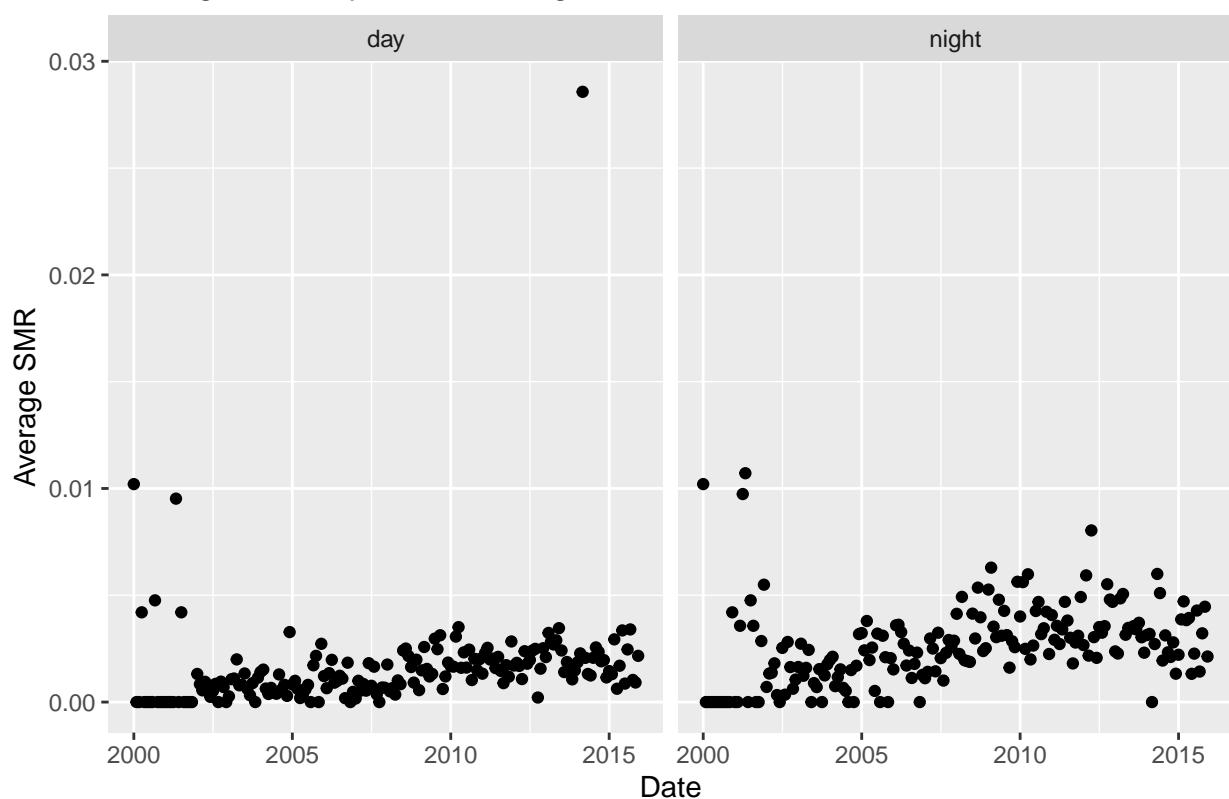
```
##  
## [29]
```

Average monthly SMR of NCdurham



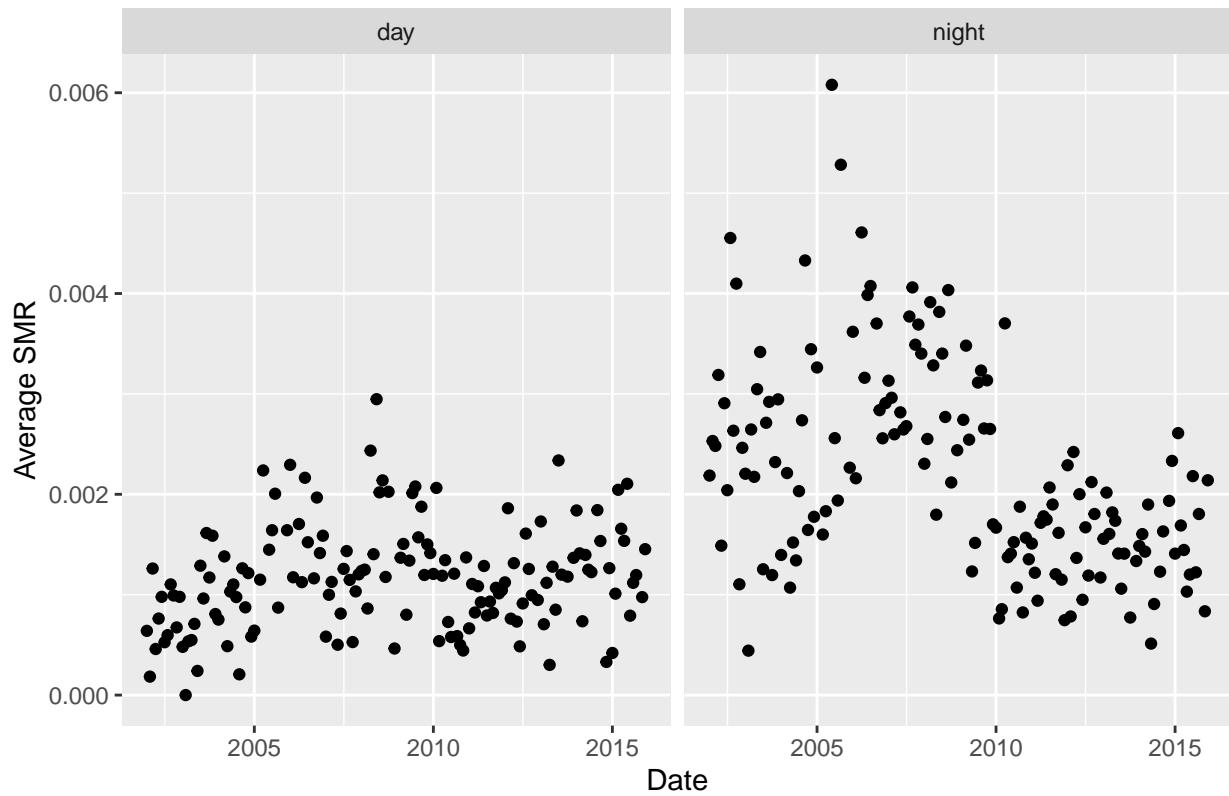
```
##  
## [[30]]
```

Average monthly SMR of NCgreensboro2020



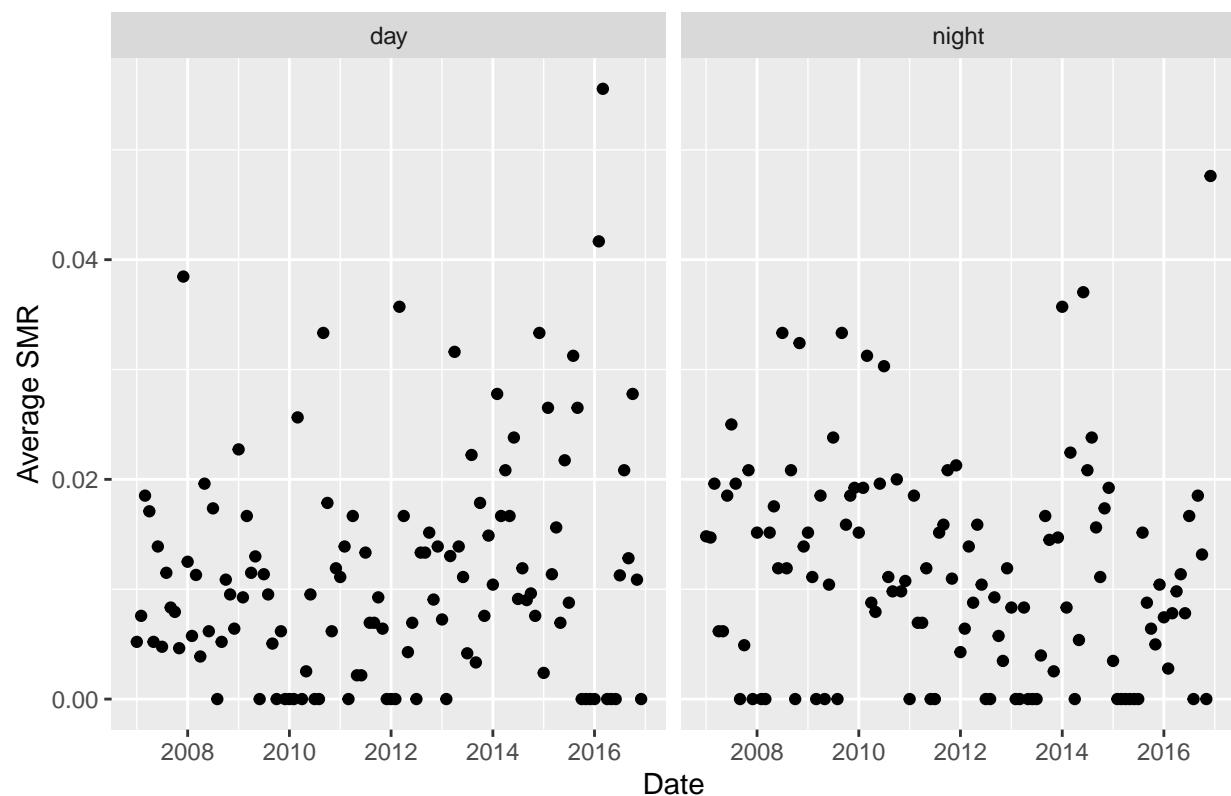
```
##  
## [[31]]
```

Average monthly SMR of NCraleigh



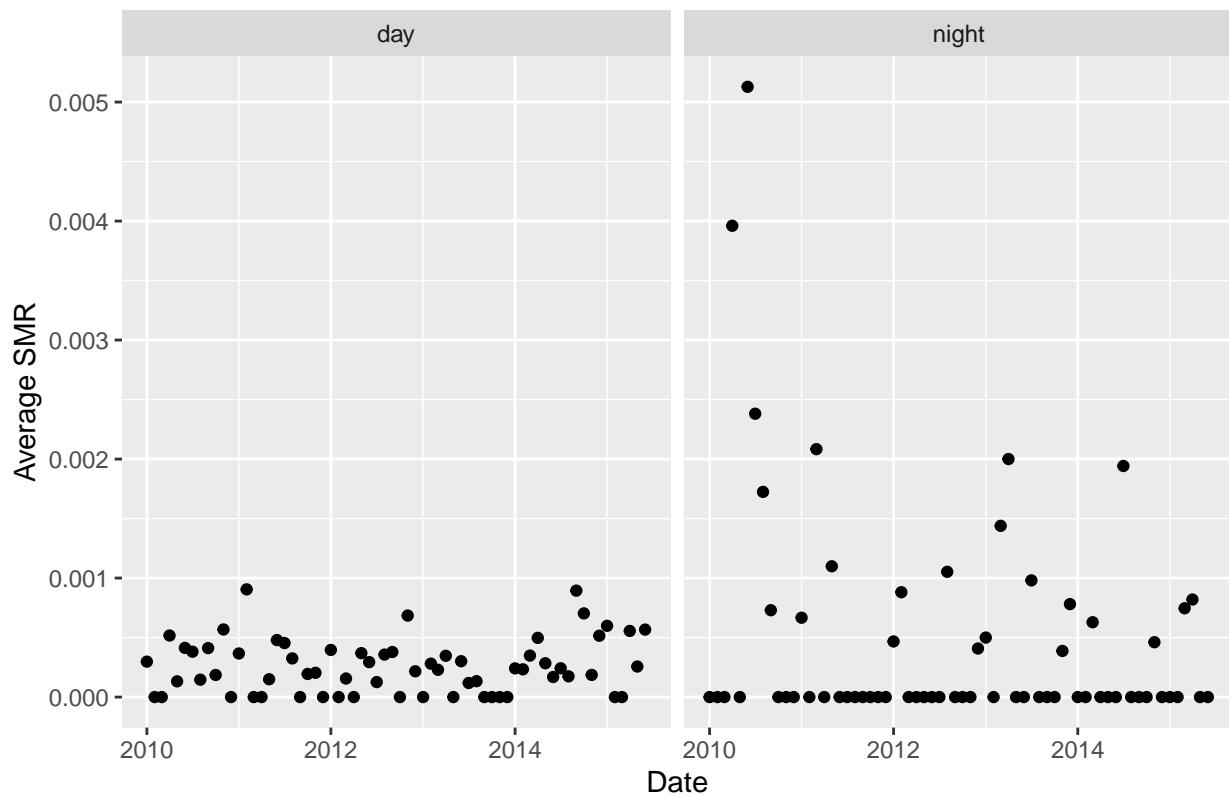
```
##  
## [[32]]
```

Average monthly SMR of NDgrandforks



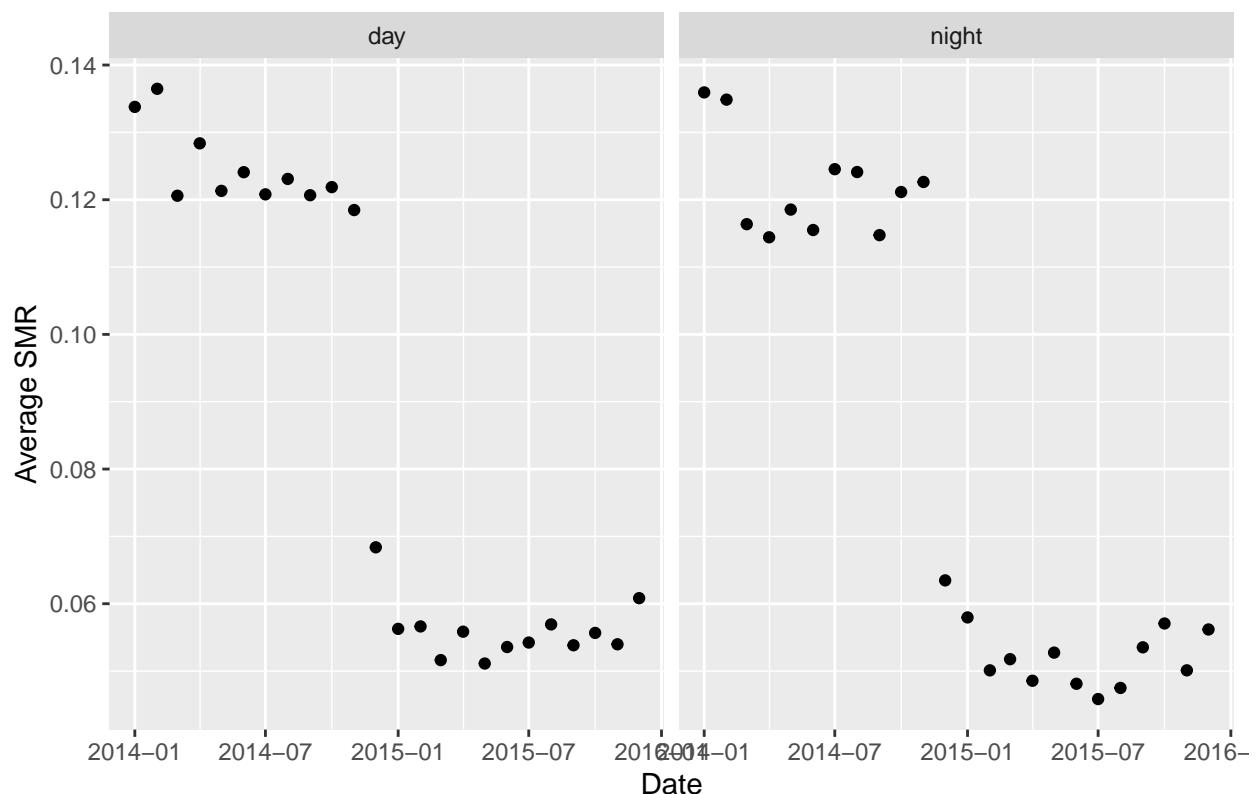
```
##  
## [[33]]
```

Average monthly SMR of ND statewide



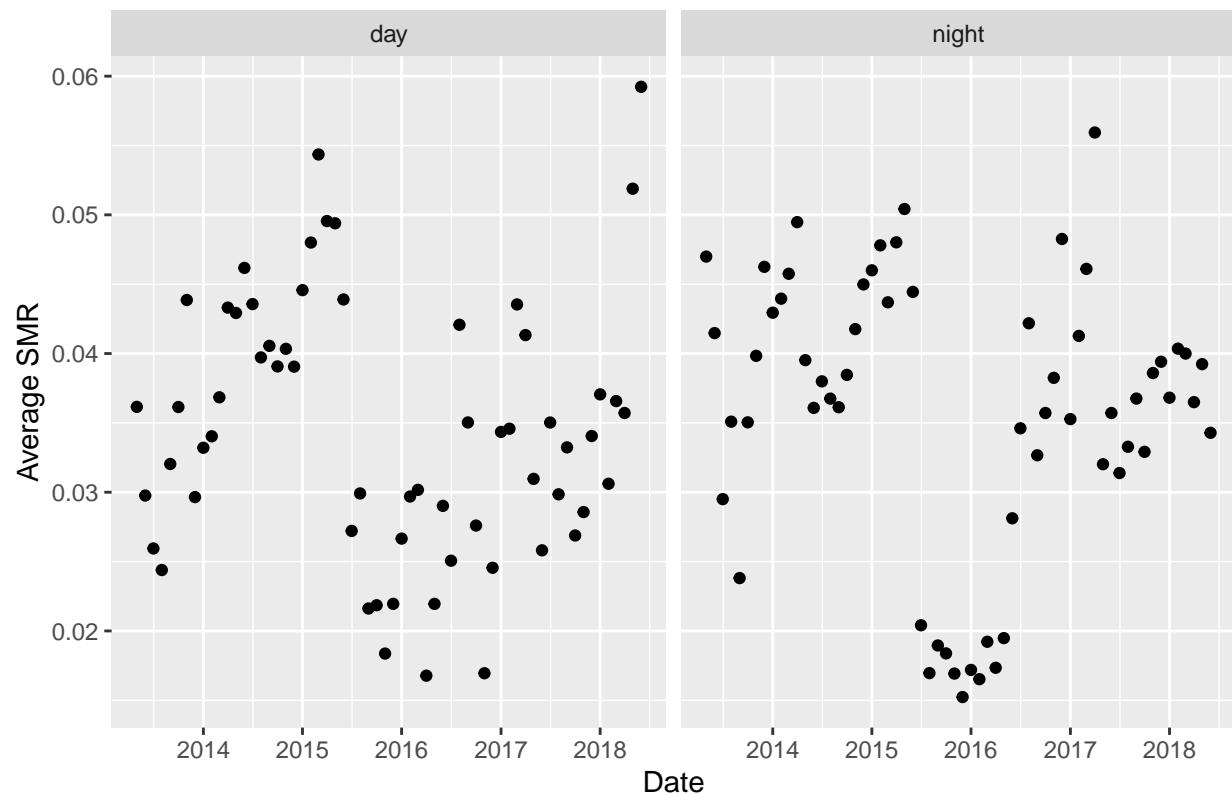
```
##  
## [[34]]
```

Average monthly SMR of NH statewide



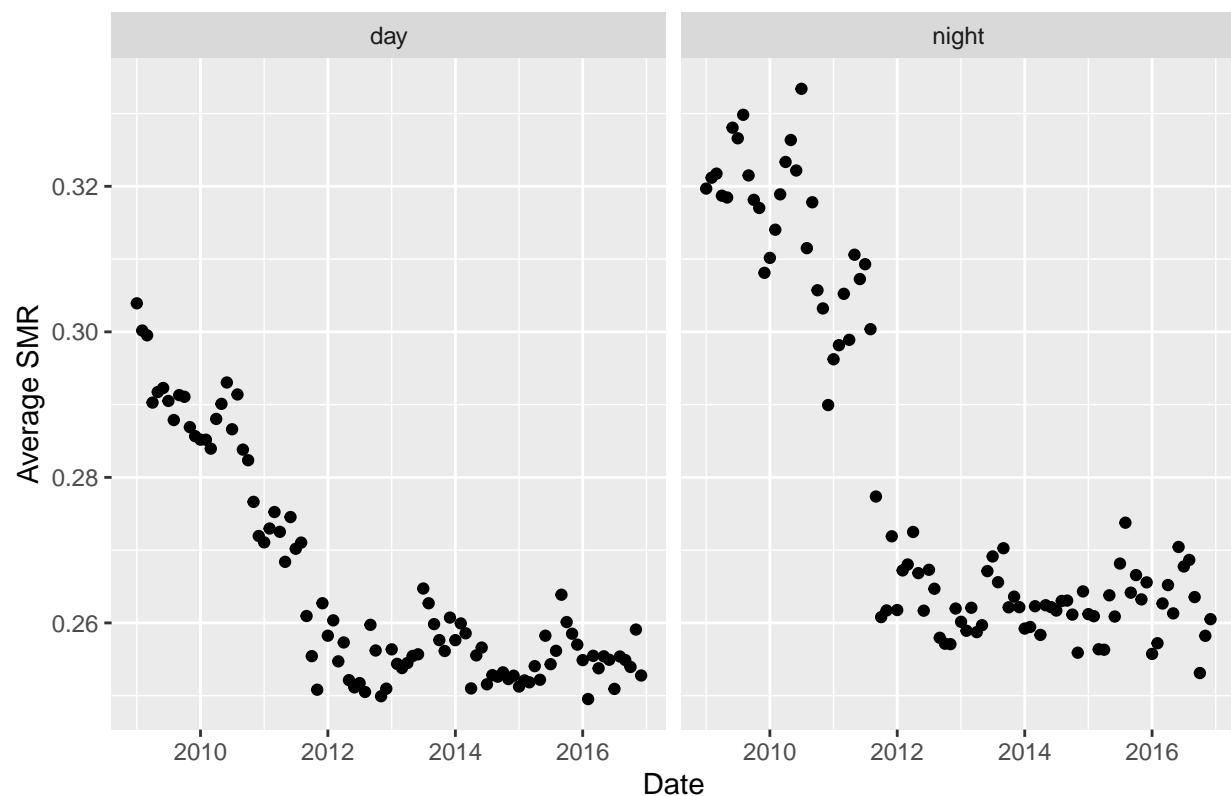
```
##  
## [[35]]
```

Average monthly SMR of NJcamden



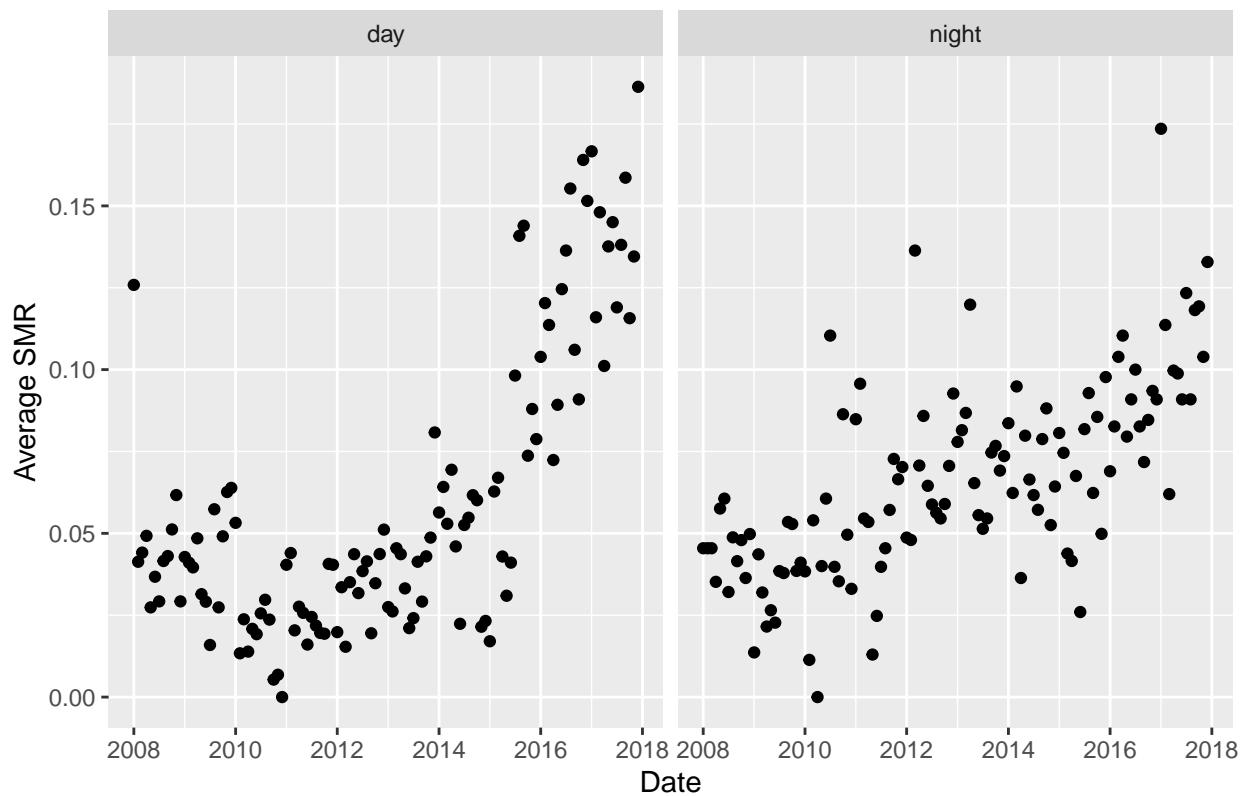
```
##  
## [[36]]
```

Average monthly SMR of NJstatewide



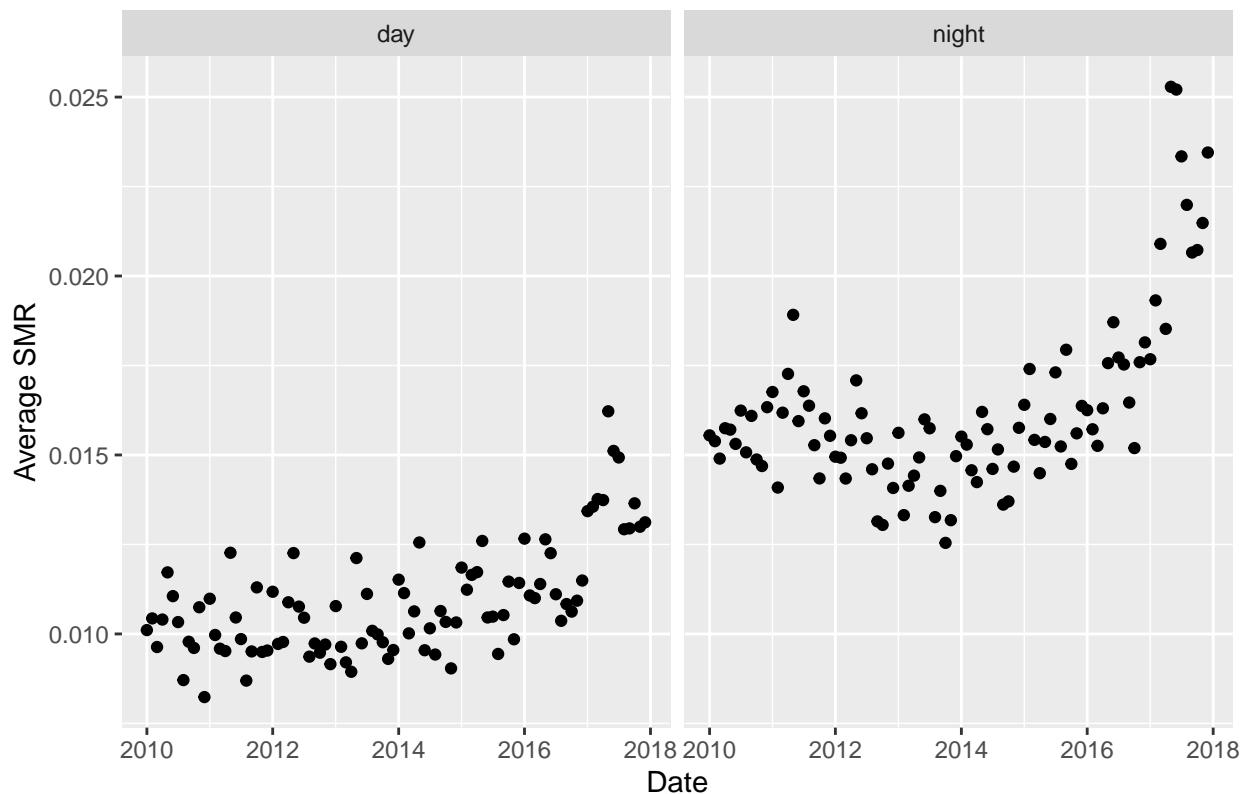
```
##  
## [[37]]
```

Average monthly SMR of NYalbany



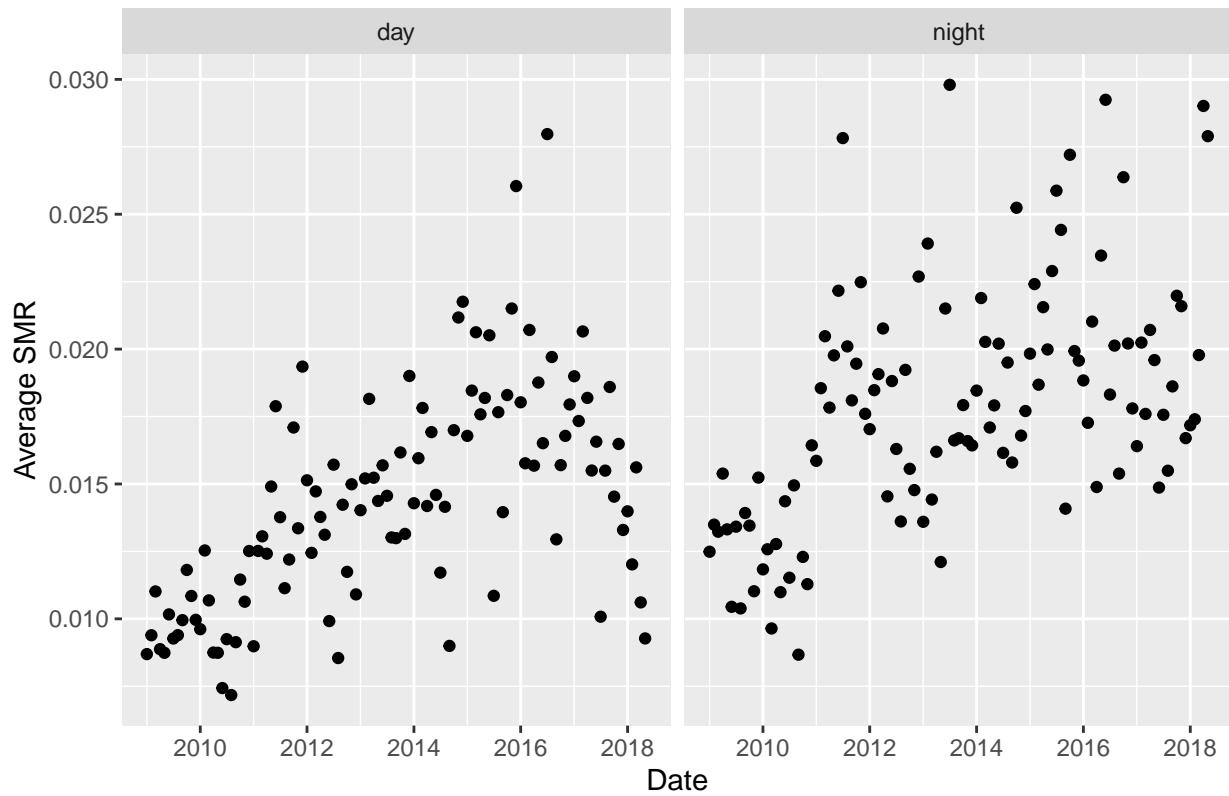
```
##  
## [38]
```

Average monthly SMR of NY statewide



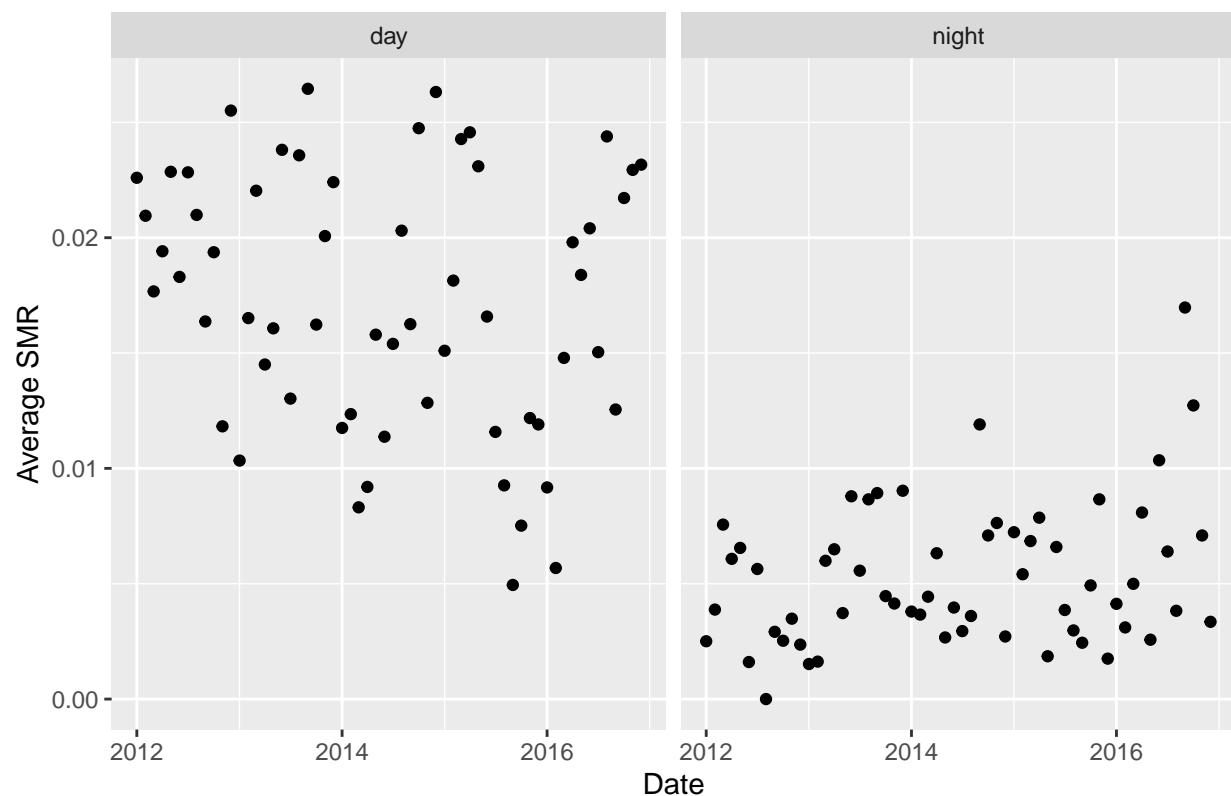
```
##  
## [[39]]
```

Average monthly SMR of OHcincinnati



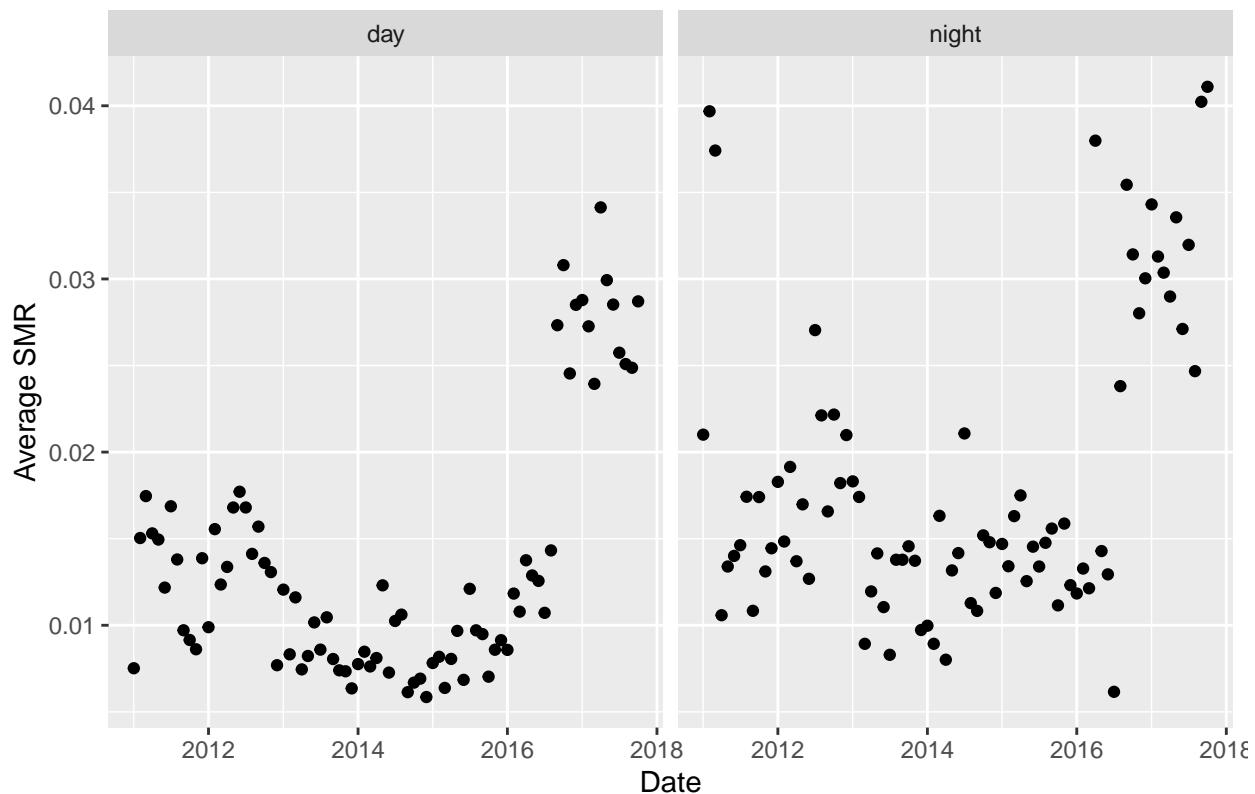
```
##  
## [[40]]
```

Average monthly SMR of OHcolumbus



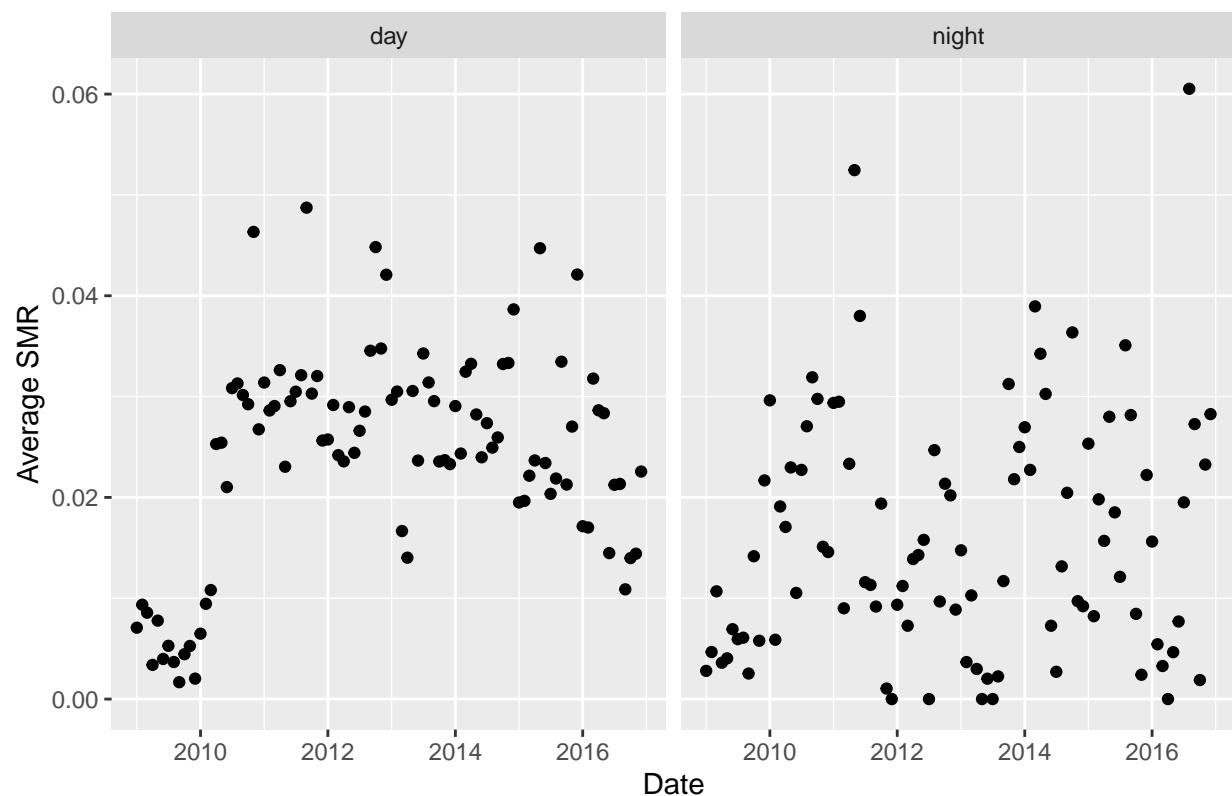
```
##  
## [[41]]
```

Average monthly SMR of OKkoklahomacity



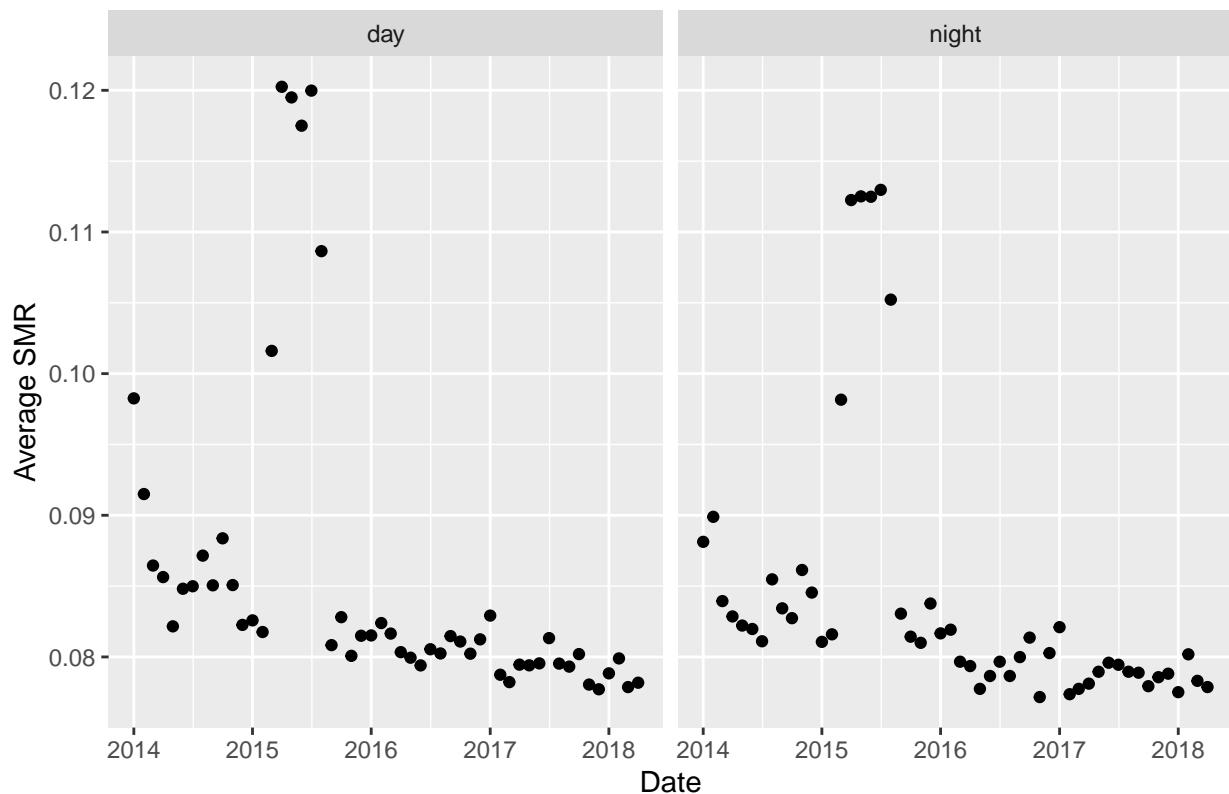
```
##  
## [42]
```

Average monthly SMR of OKtulsa



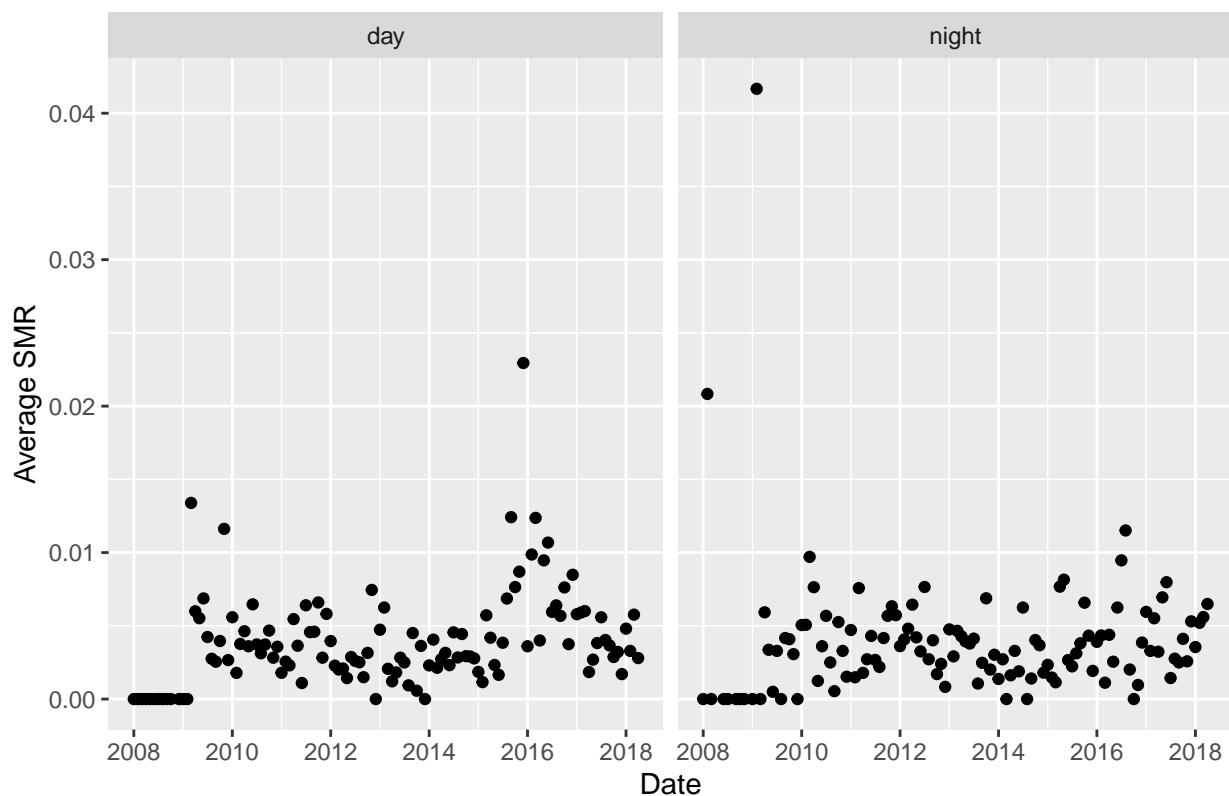
```
##  
## [43]
```

Average monthly SMR of PPhiladelphia



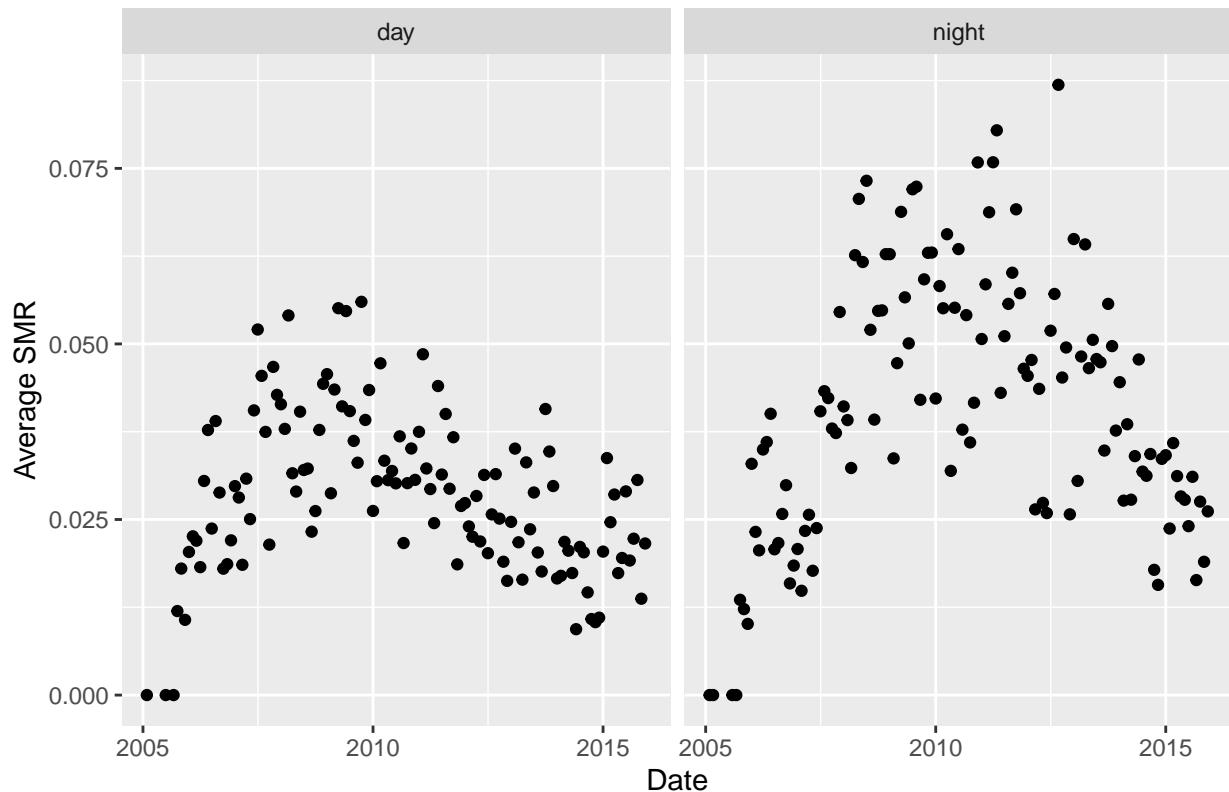
```
##  
## [[44]]
```

Average monthly SMR of PApittsburgh



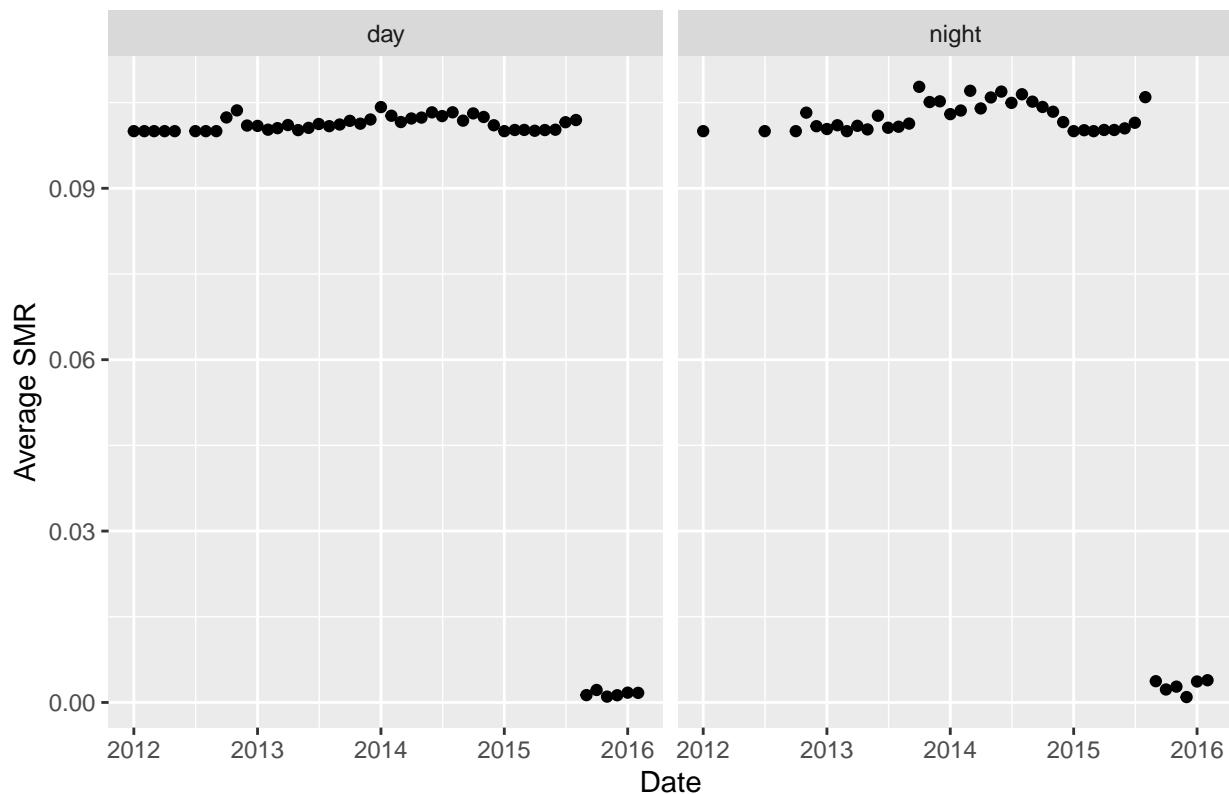
```
##  
## [45]
```

Average monthly SMR of RI statewide



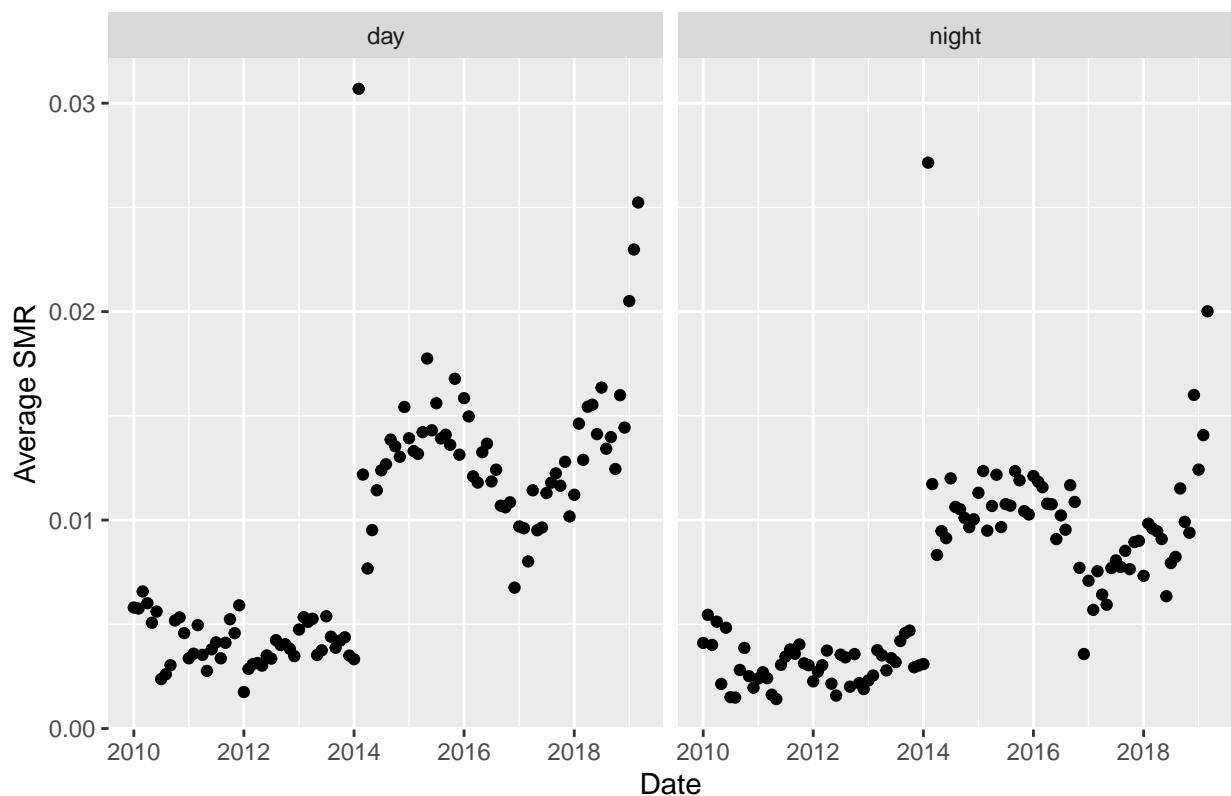
```
##  
## [46]
```

Average monthly SMR of SD statewide



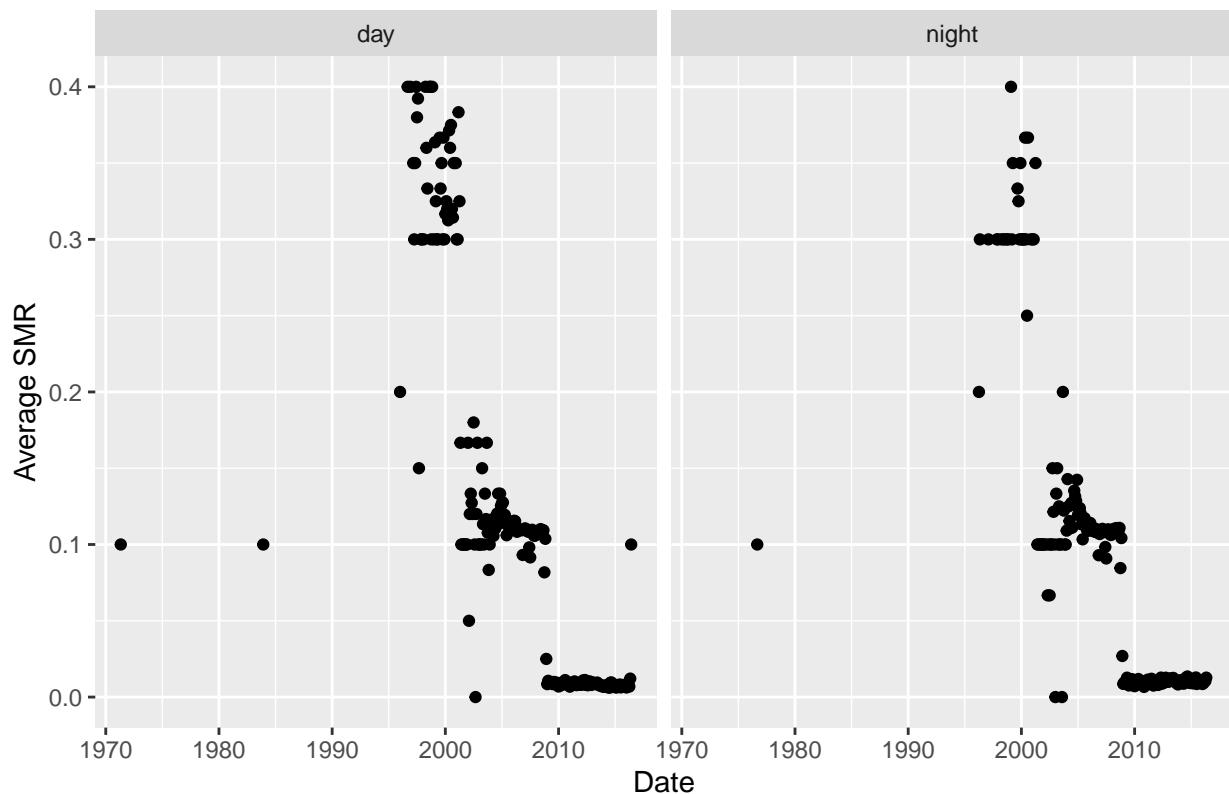
```
##  
## [47]
```

Average monthly SMR of TNnashville



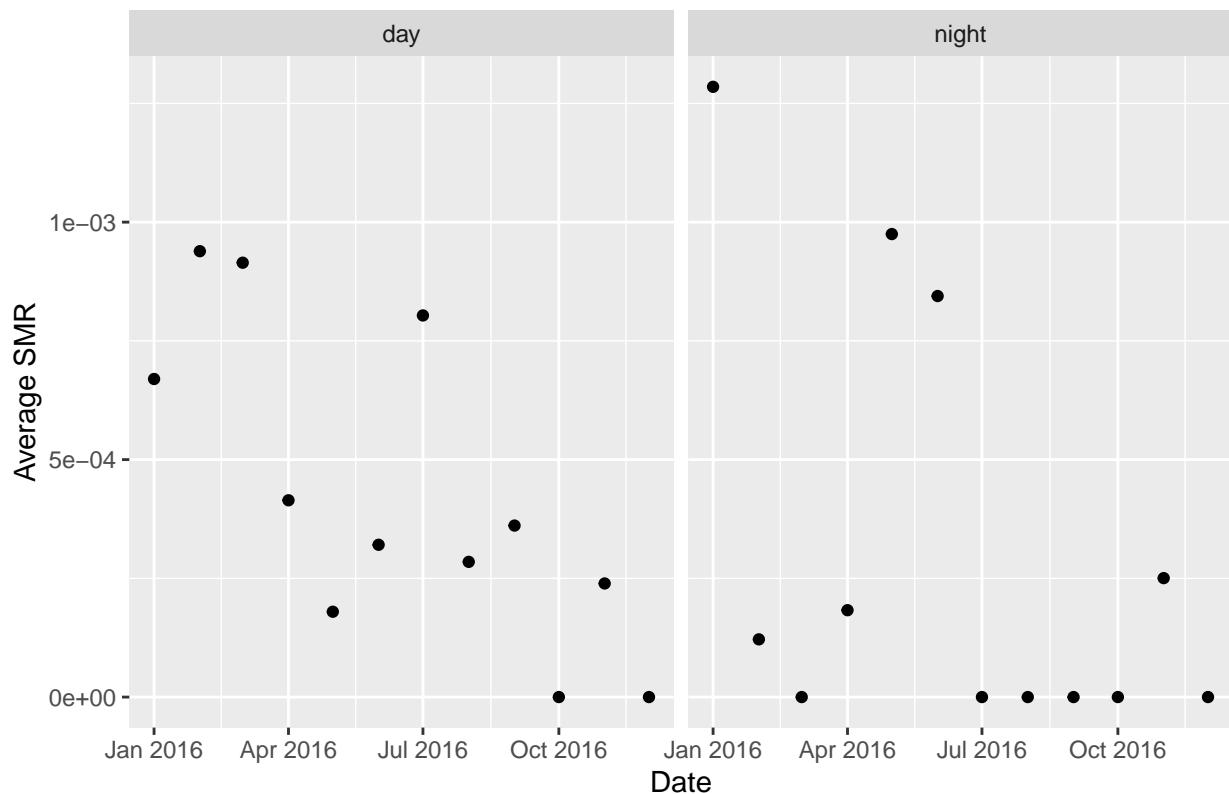
```
##  
## [48]
```

Average monthly SMR of TNstate



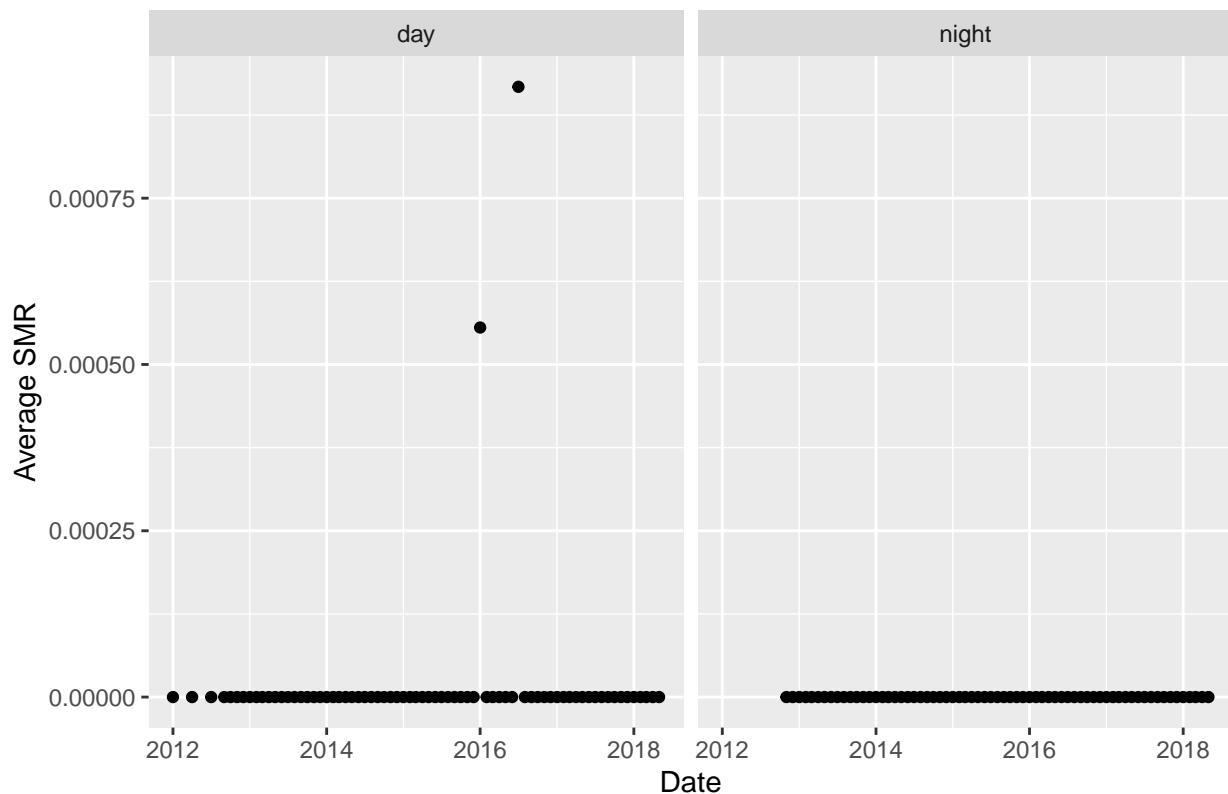
```
##  
## [49]
```

Average monthly SMR of TXarlington



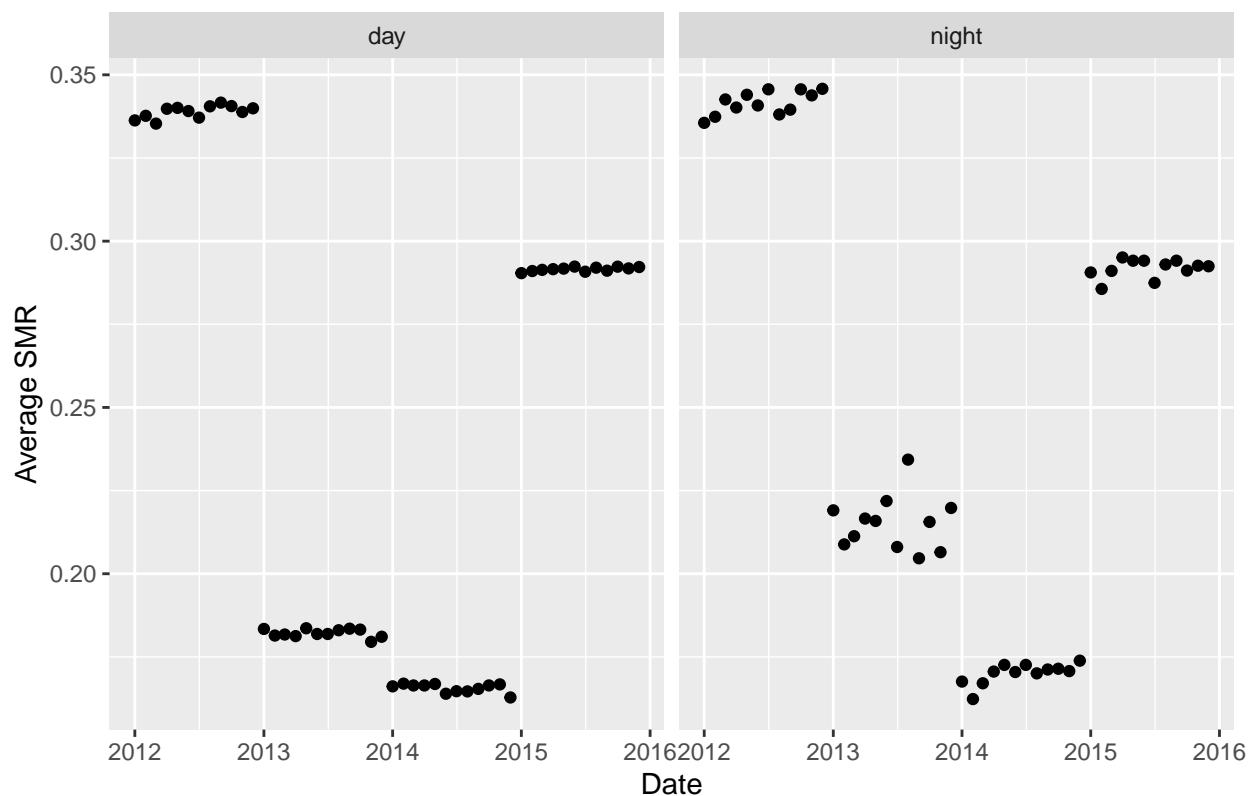
```
##  
## [[50]]
```

Average monthly SMR of TXgarland



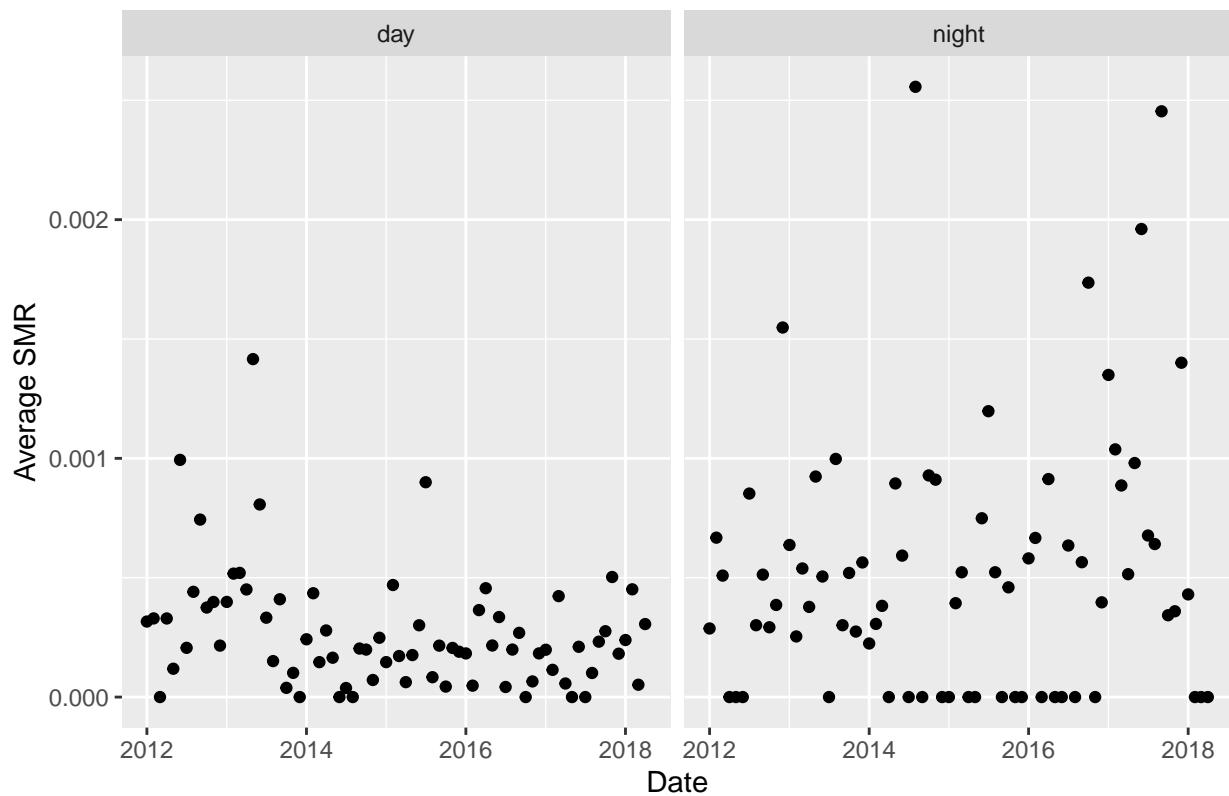
```
##  
## [[51]]
```

Average monthly SMR of TXplano



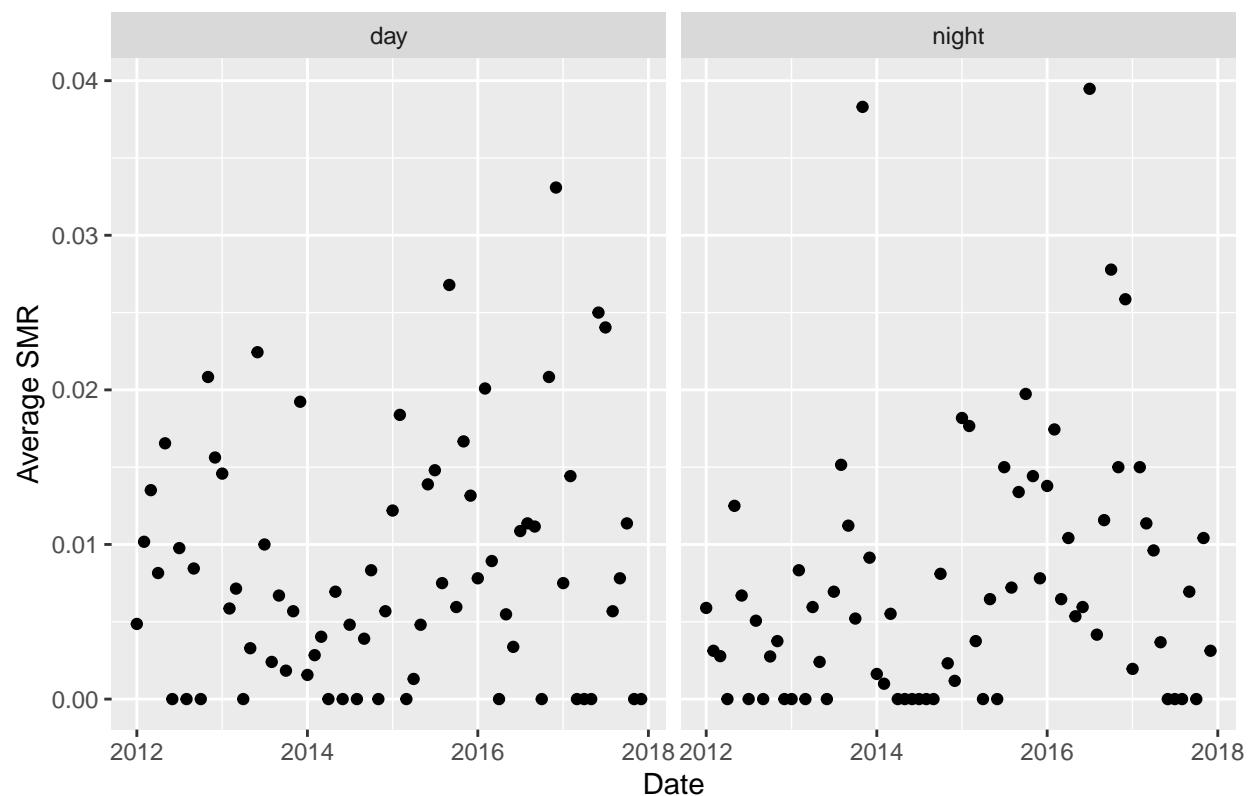
```
##  
## [52]
```

Average monthly SMR of TXsanantonio



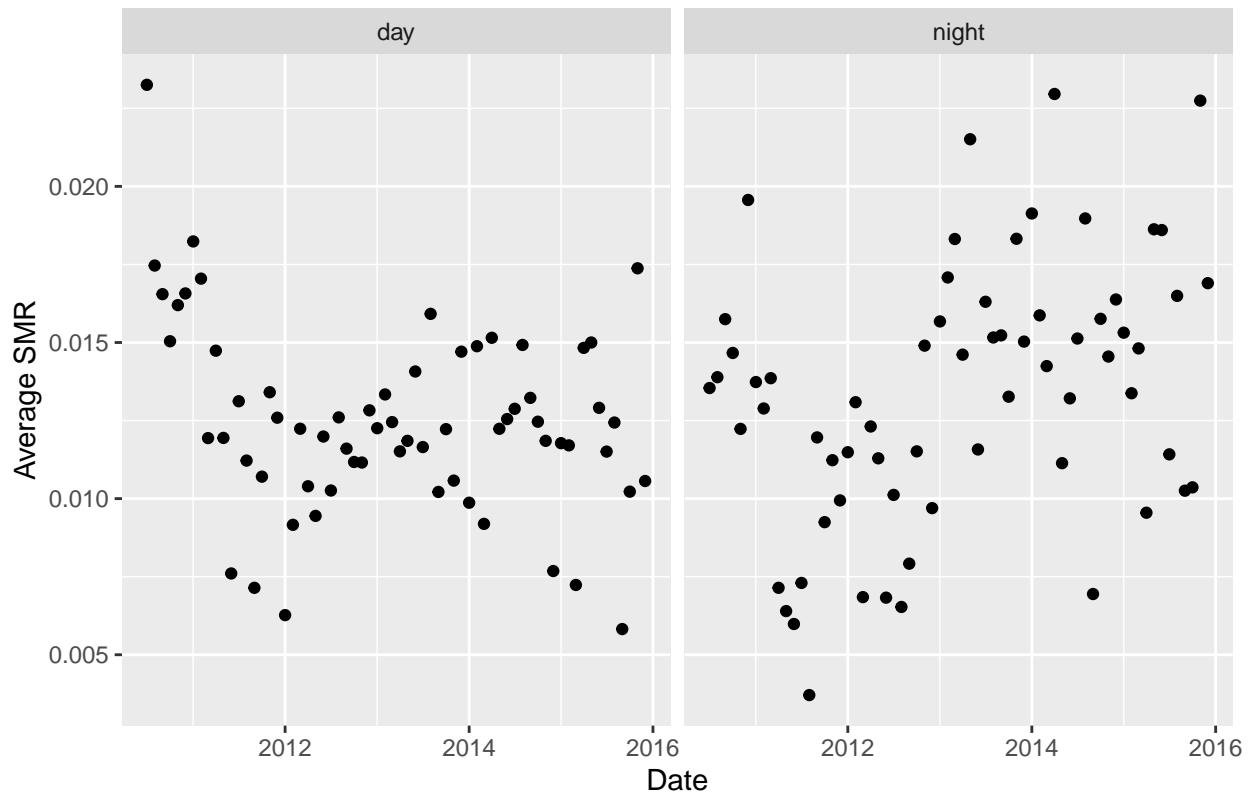
```
##  
## [53]
```

Average monthly SMR of VTburlington2020



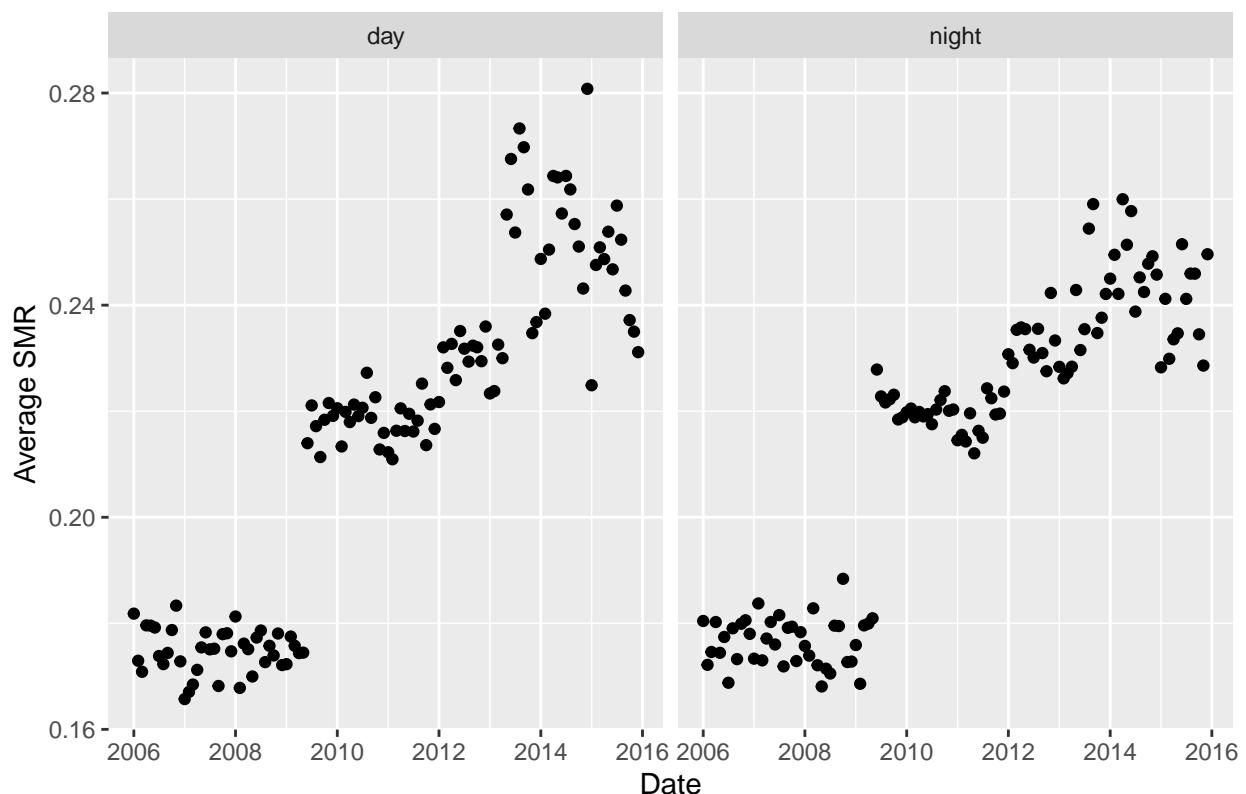
```
##  
## [[54]]
```

Average monthly SMR of VT statewide2020



```
##  
## [55]
```

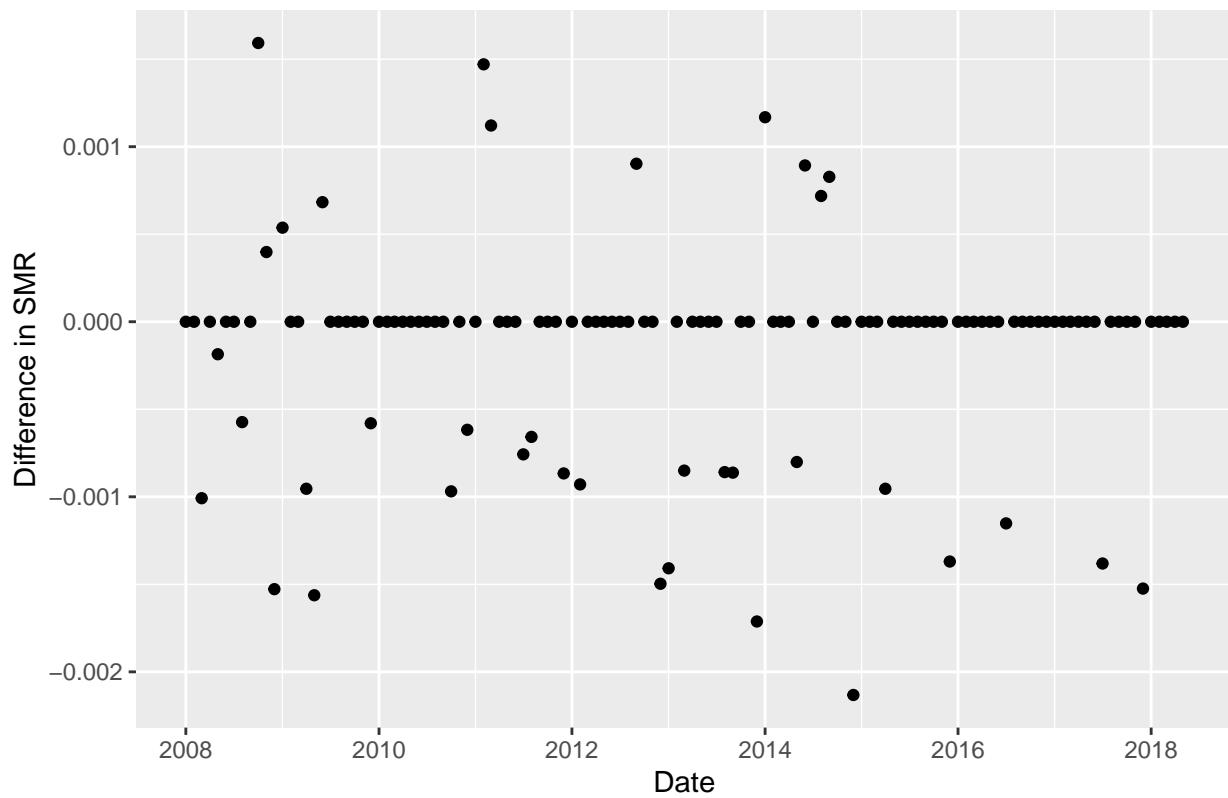
Average monthly SMR of WAseattle



```
ggplot_sunriseset(sunrisesetdiff_df, "diff_SMR")
```

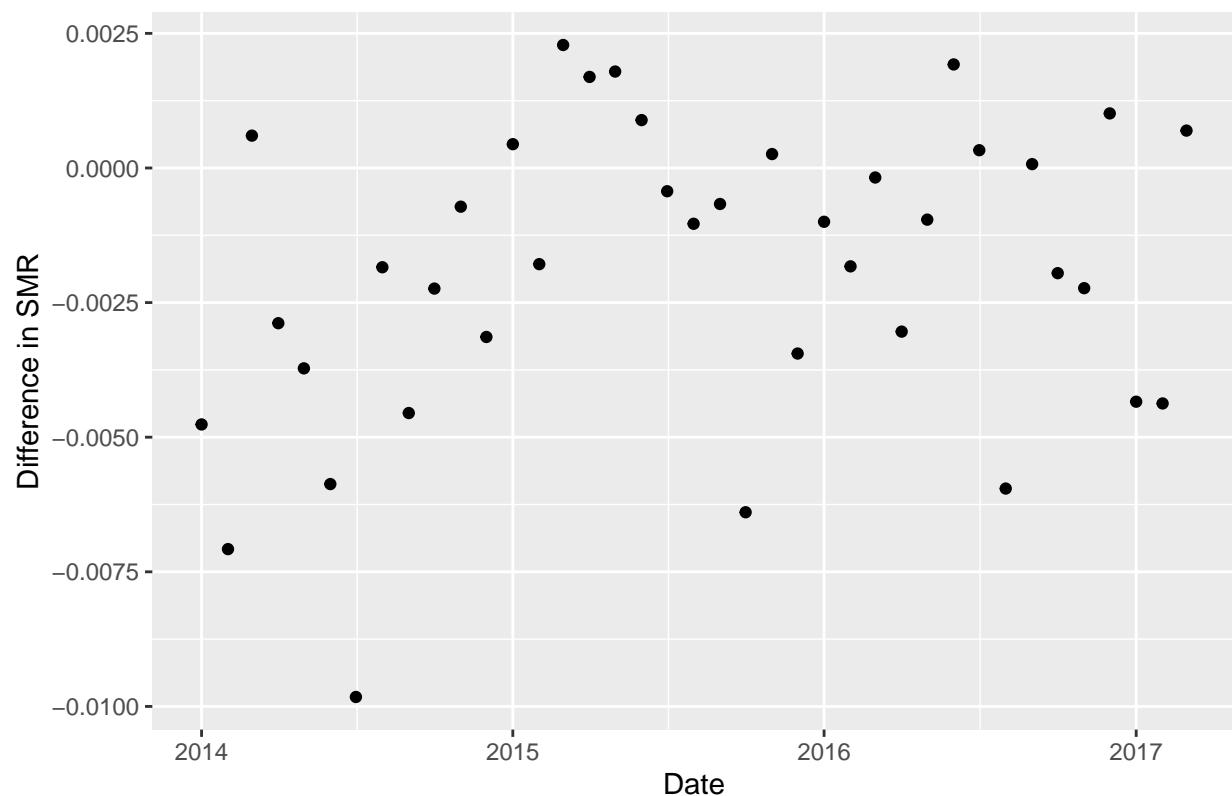
```
## [1]
```

Difference in SMR of AZgilbert Between Day and Night per month



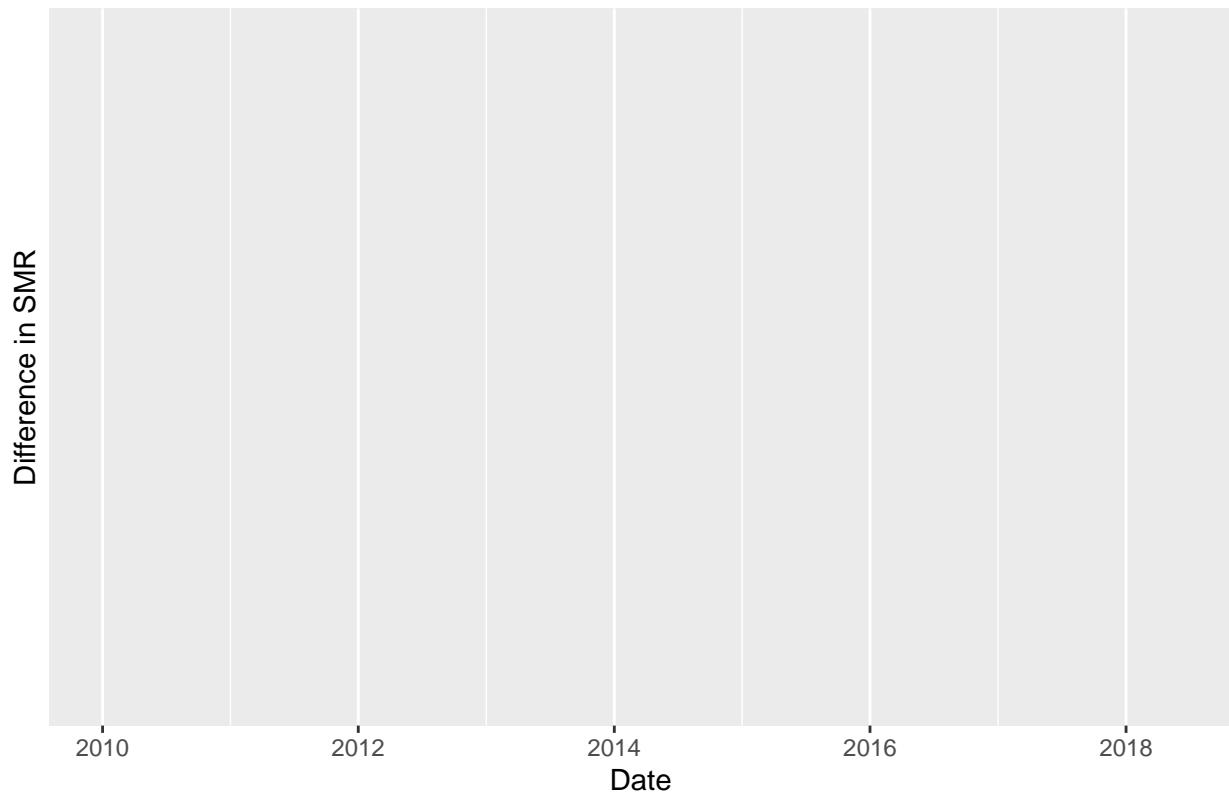
```
##  
## [[2]]
```

Difference in SMR of AZmesa Between Day and Night per month



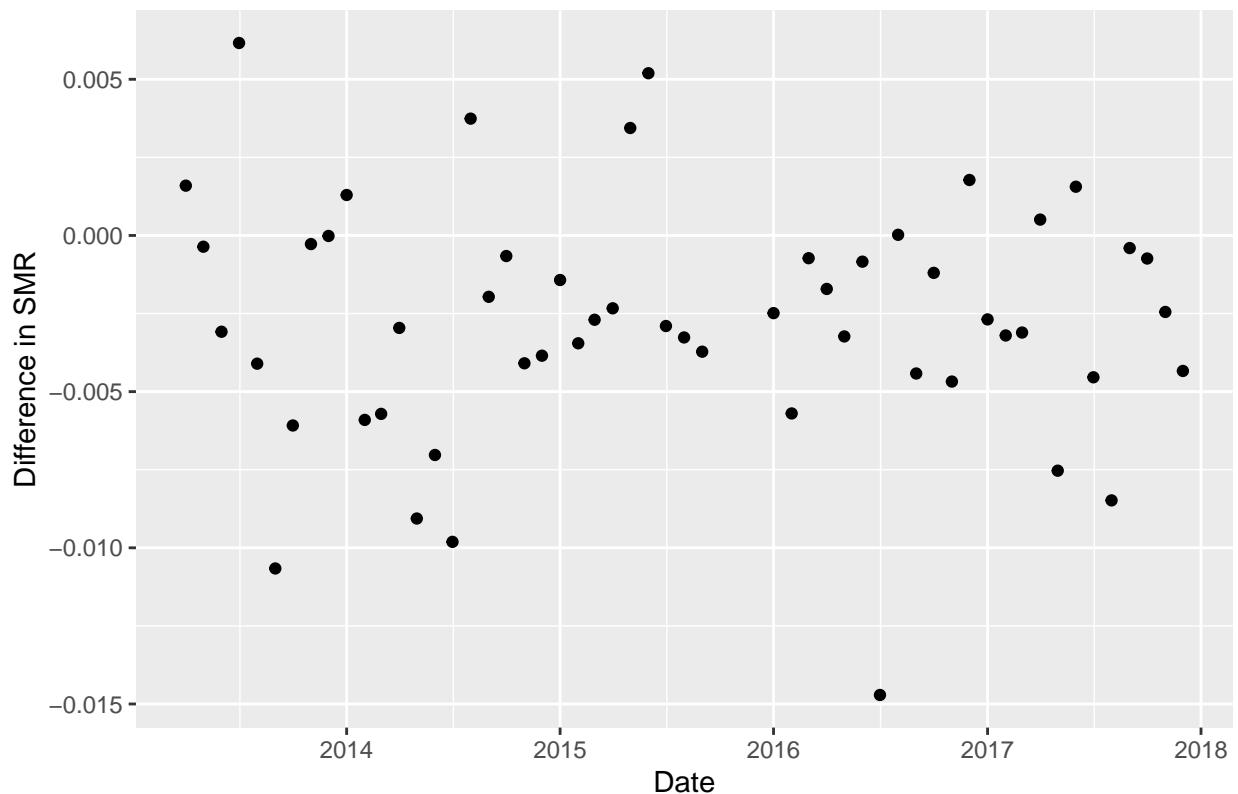
```
##  
## [[3]]
```

Difference in SMR of CAlosangeles Between Day and Night per month



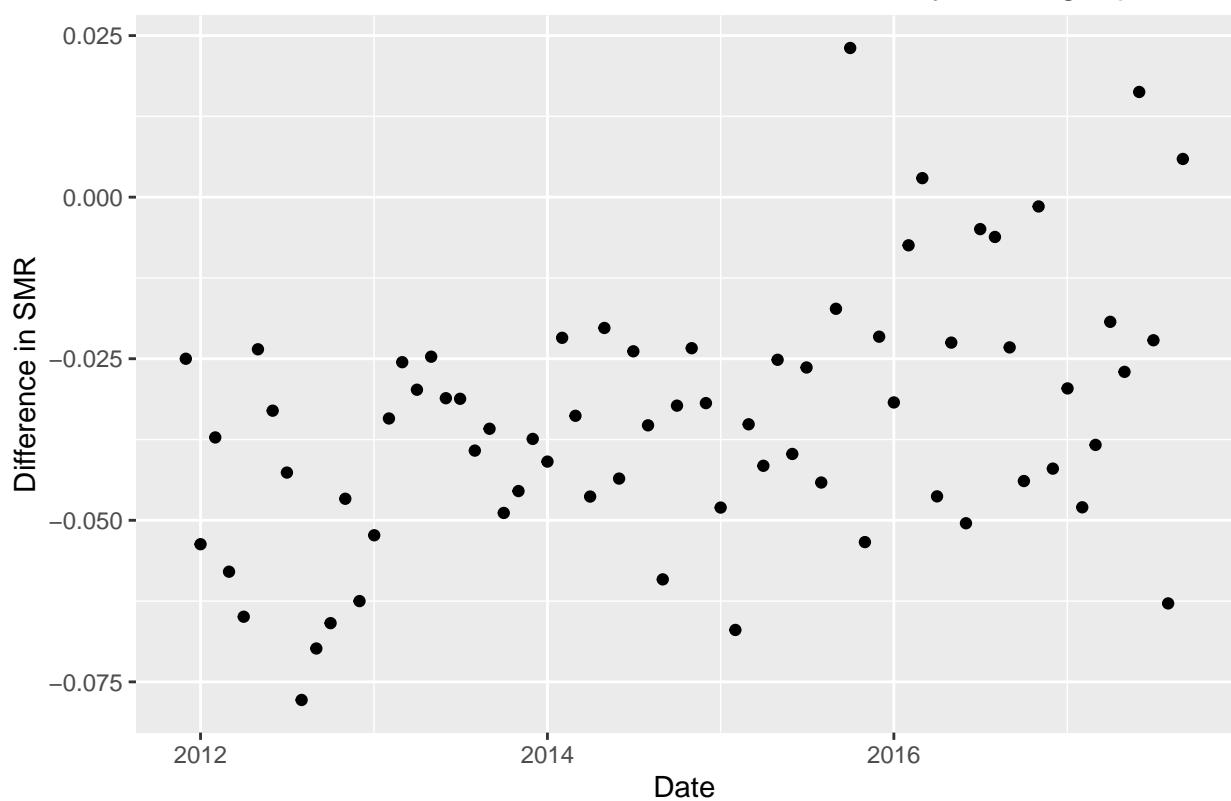
```
##  
## [[4]]
```

Difference in SMR of CAoakland Between Day and Night per month



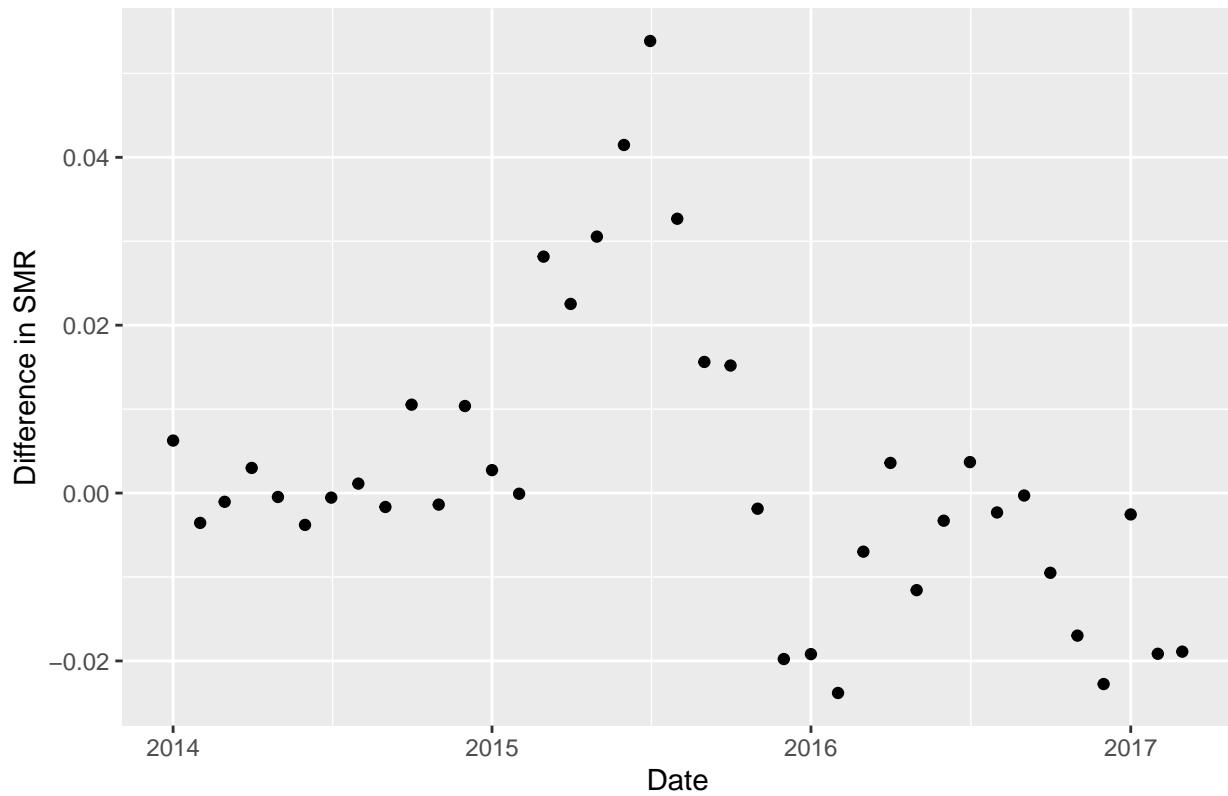
```
##  
## [[5]]
```

Difference in SMR of CAsanbernardino Between Day and Night per month



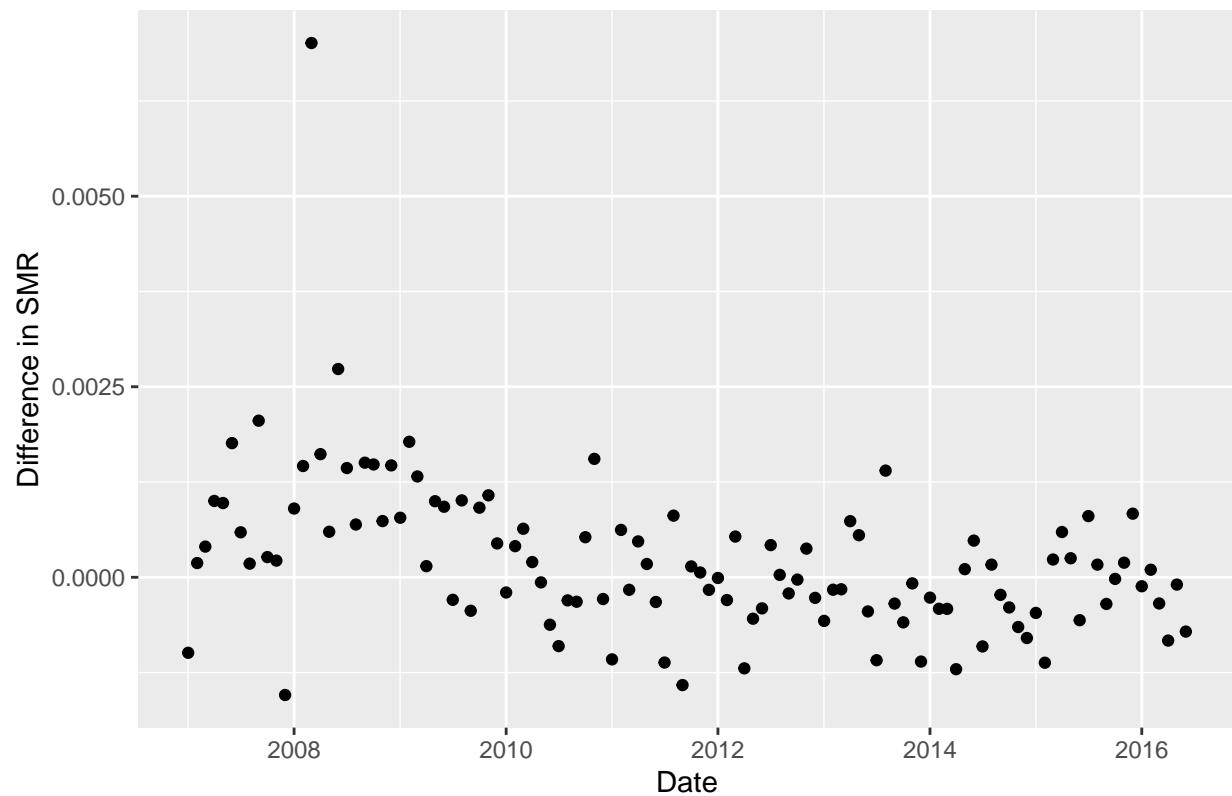
```
##  
## [[6]]
```

Difference in SMR of CAandiego Between Day and Night per month



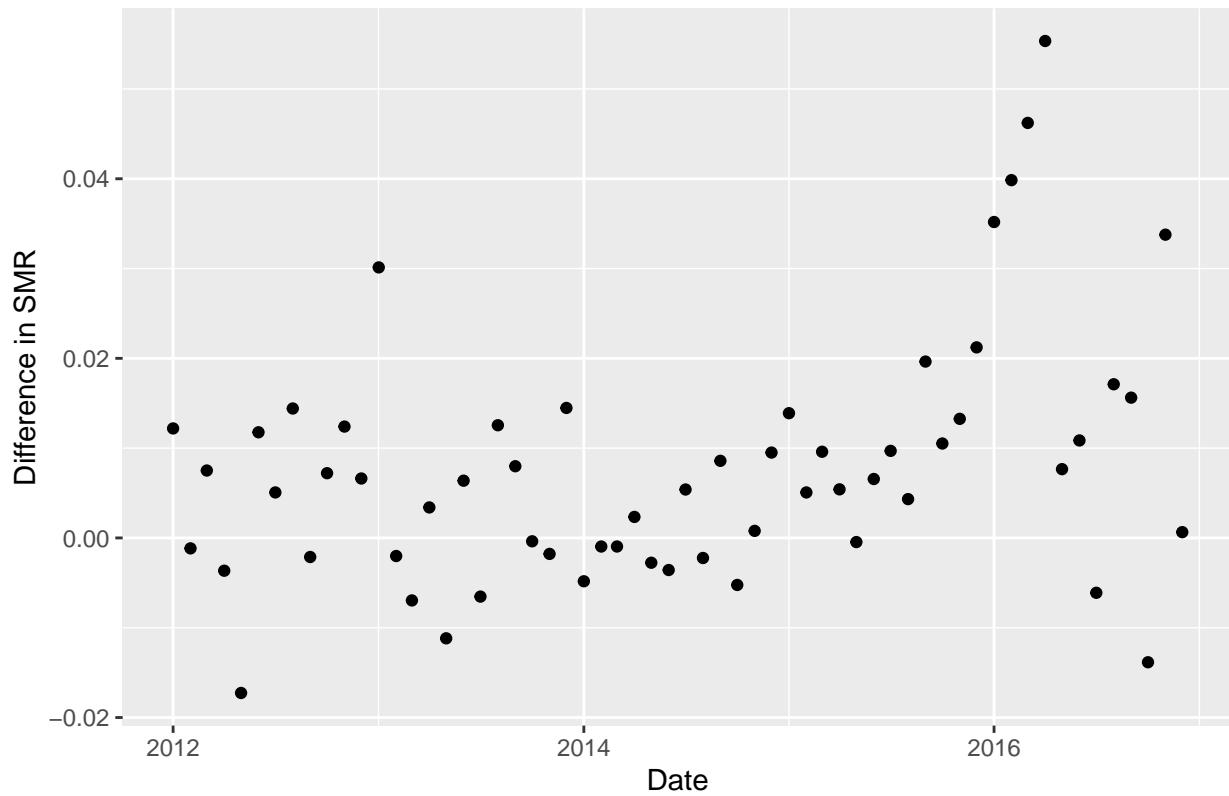
```
##  
## [[7]]
```

Difference in SMR of CAsanfrancisco Between Day and Night per month



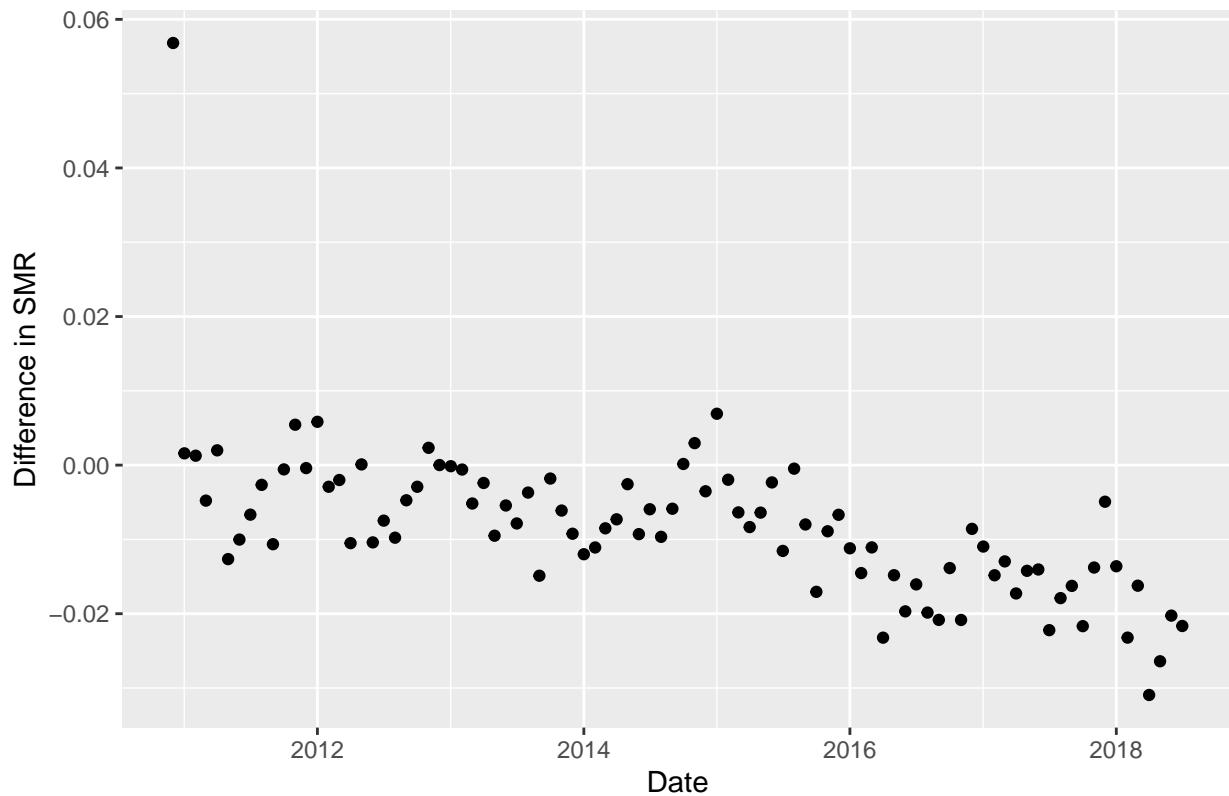
```
##  
## [[8]]
```

Difference in SMR of COaurora Between Day and Night per month



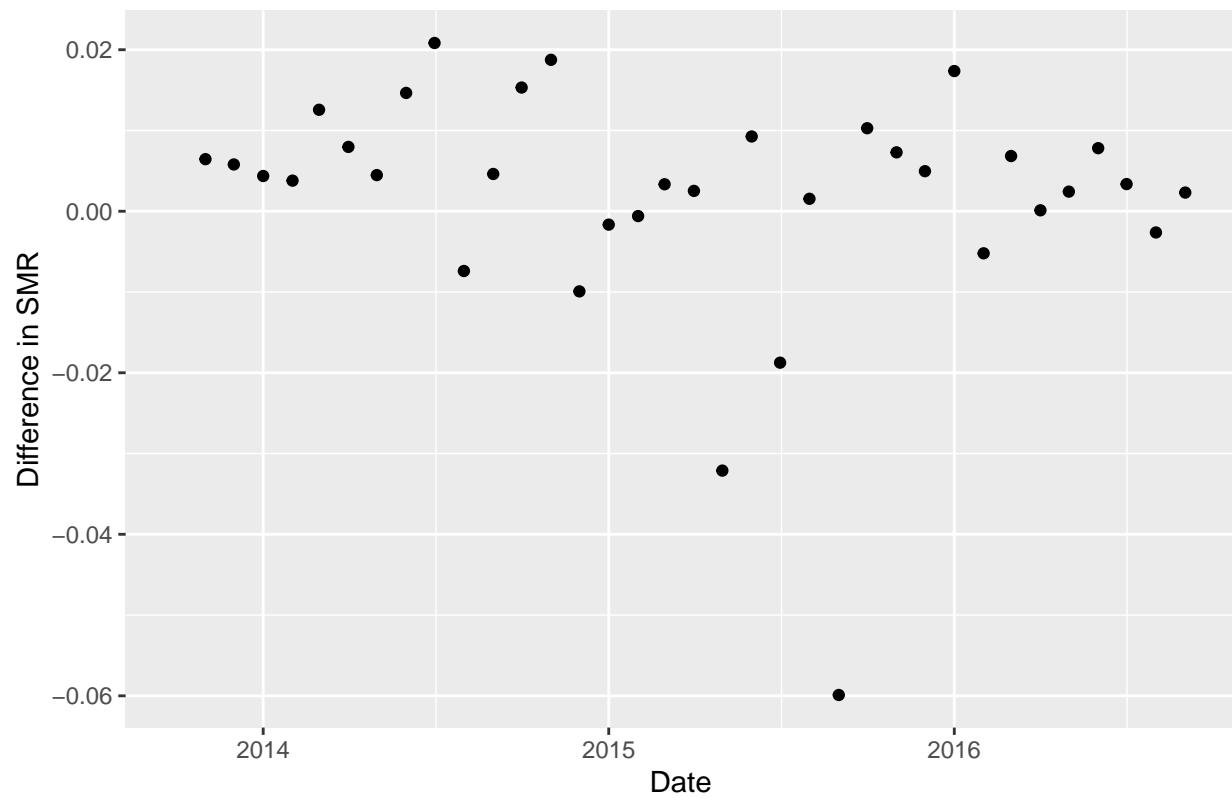
```
##  
## [[9]]
```

Difference in SMR of COdenver Between Day and Night per month



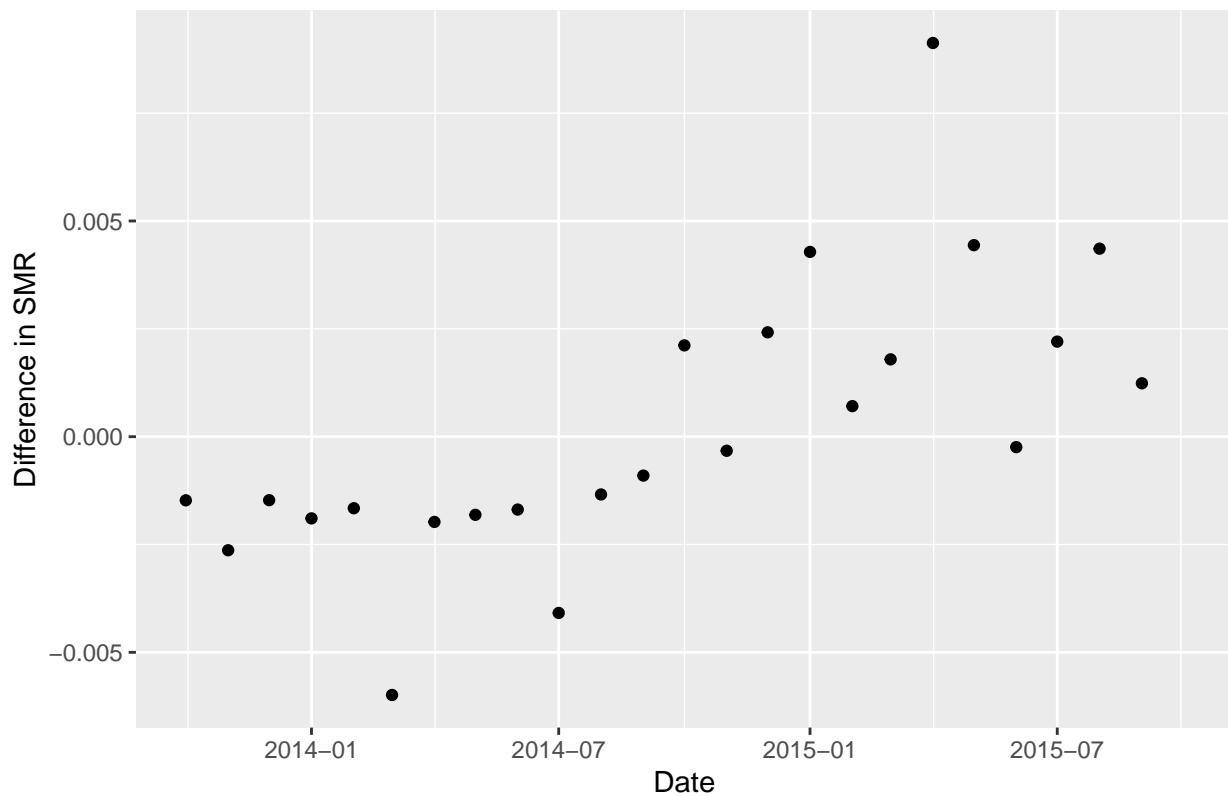
```
##  
## [[10]]
```

Difference in SMR of CThartford Between Day and Night per month



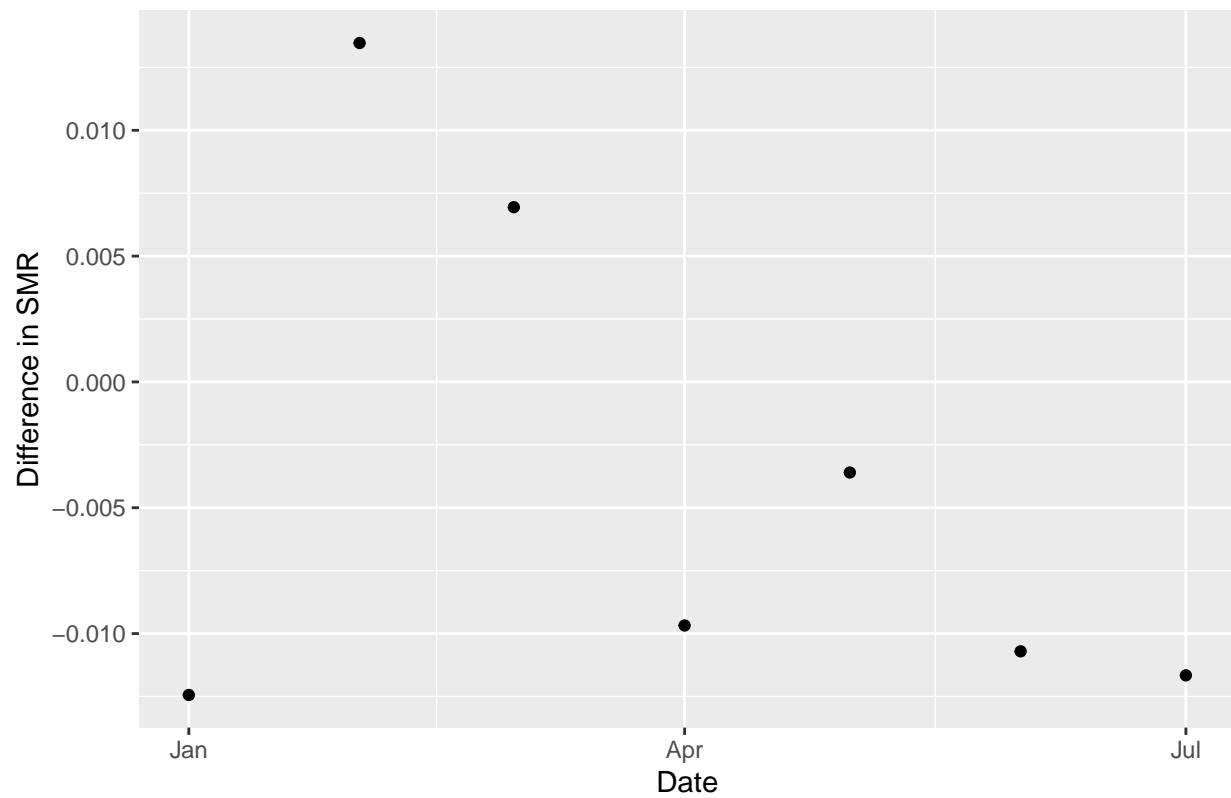
```
##  
## [[11]]
```

Difference in SMR of CT statewide Between Day and Night per month



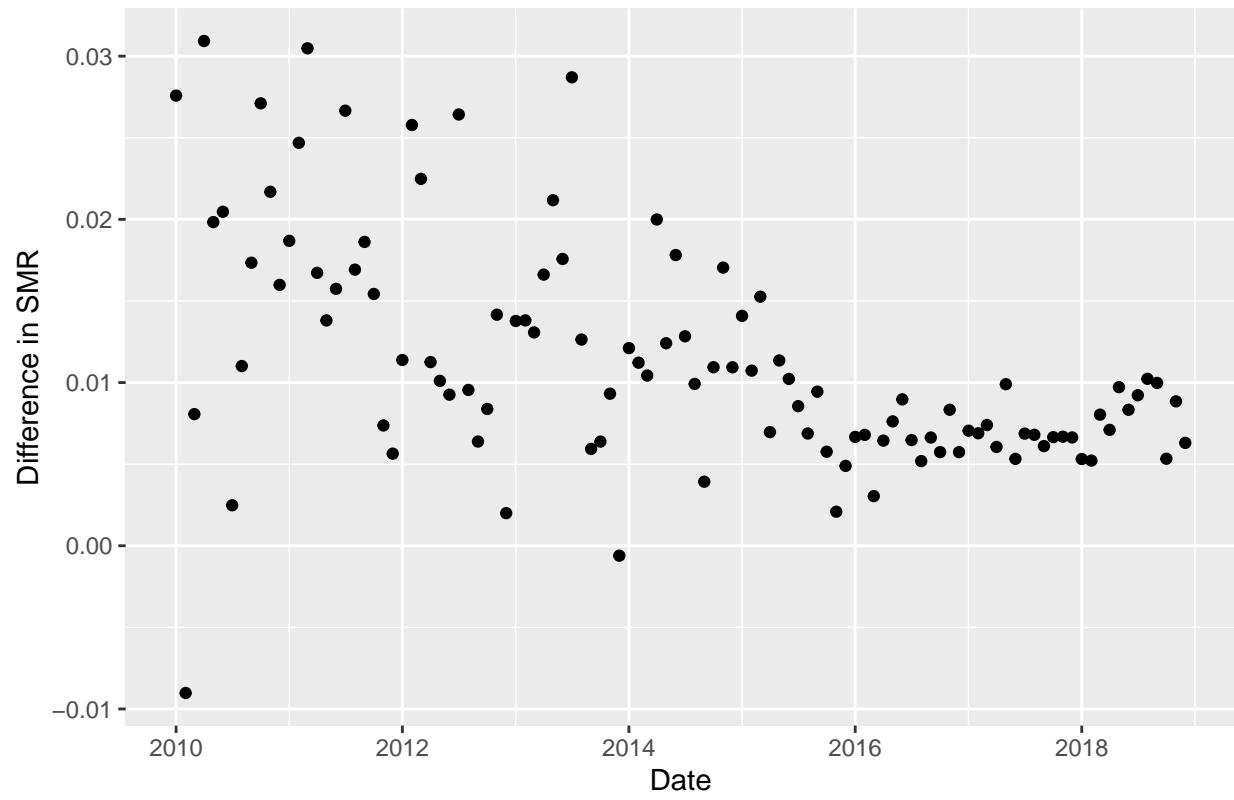
```
##  
## [[12]]
```

Difference in SMR of FLsaint Between Day and Night per month



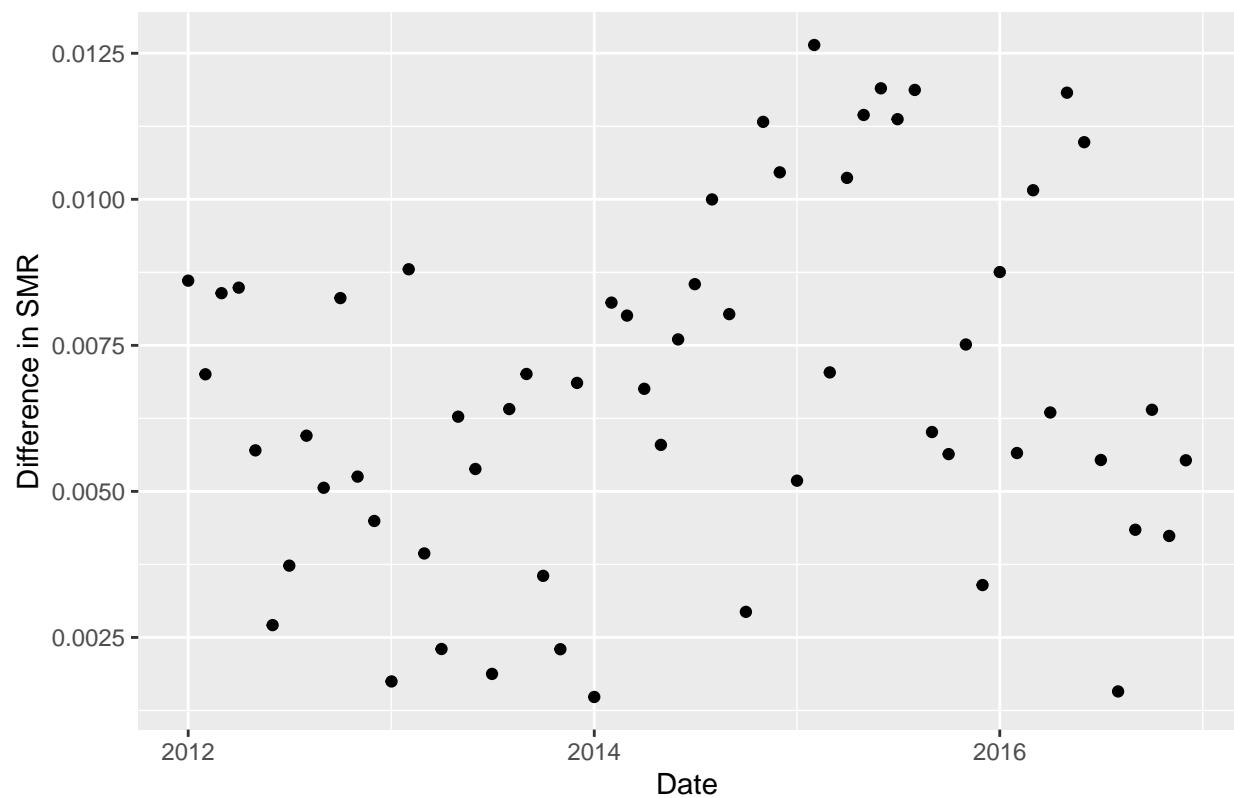
```
##  
## [[13]]
```

Difference in SMR of FL statewide Between Day and Night per month



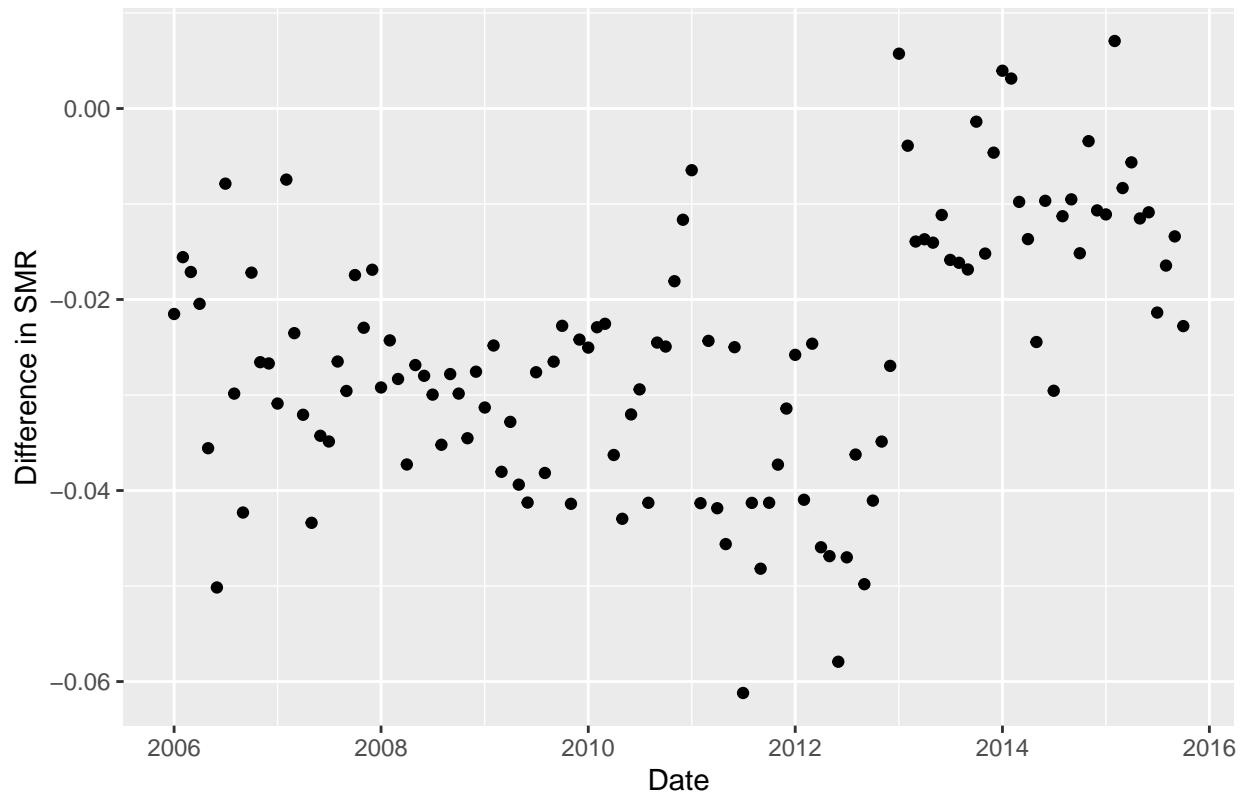
```
##  
## [[14]]
```

Difference in SMR of GA statewide Between Day and Night per month



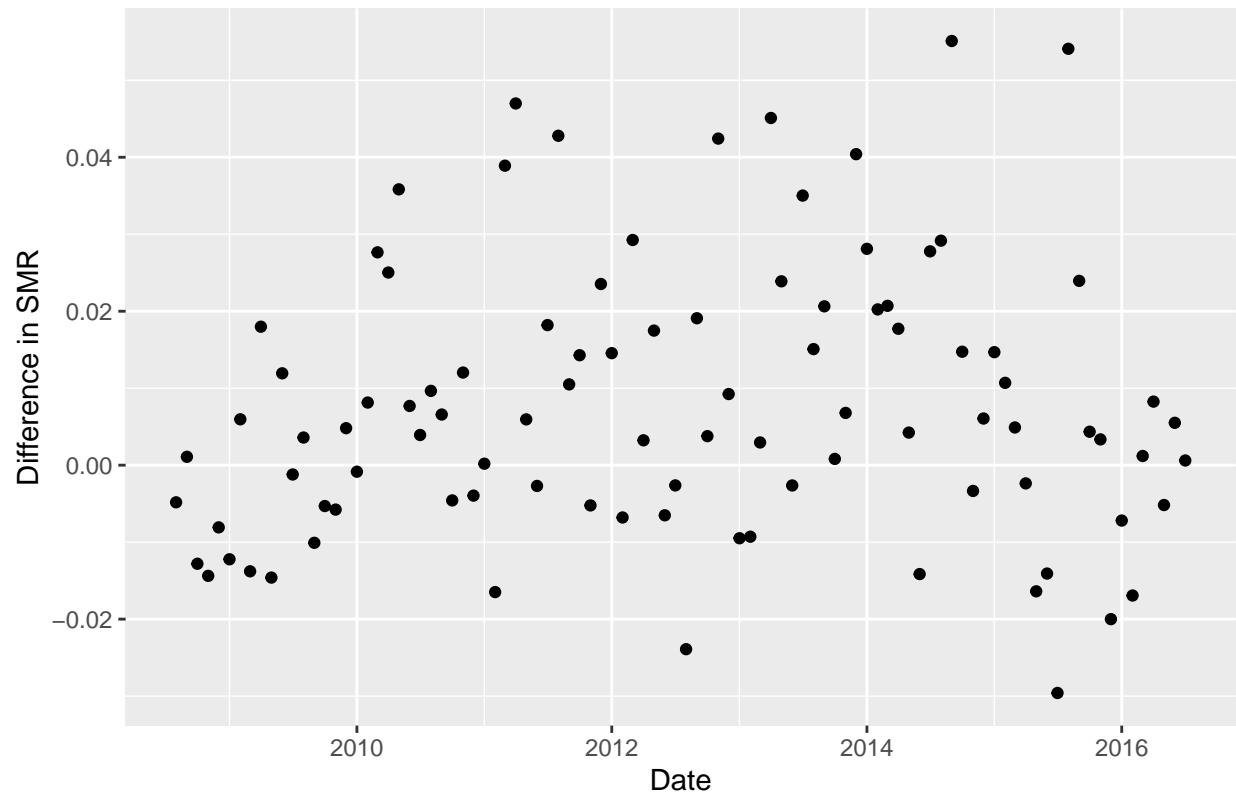
```
##  
## [[15]]
```

Difference in SMR of IAstatewide Between Day and Night per month



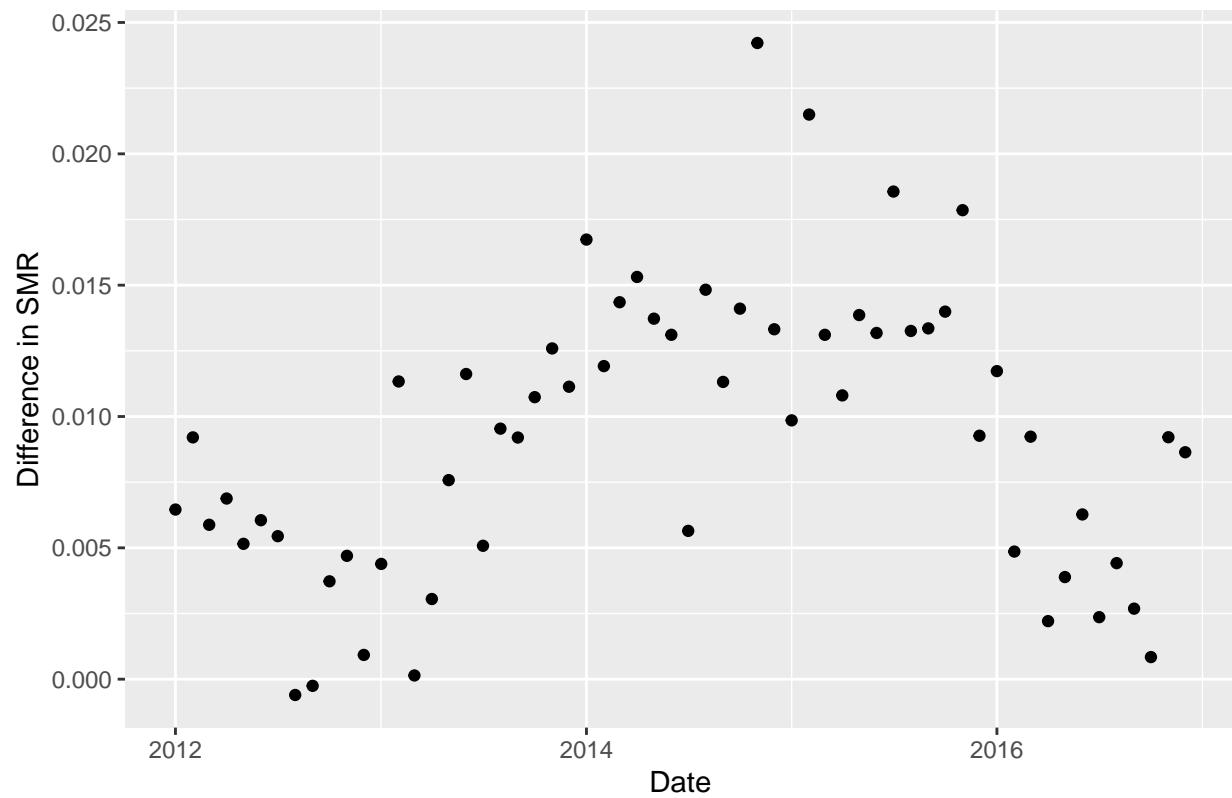
```
##  
## [[16]]
```

Difference in SMR of IDidahofalls Between Day and Night per month



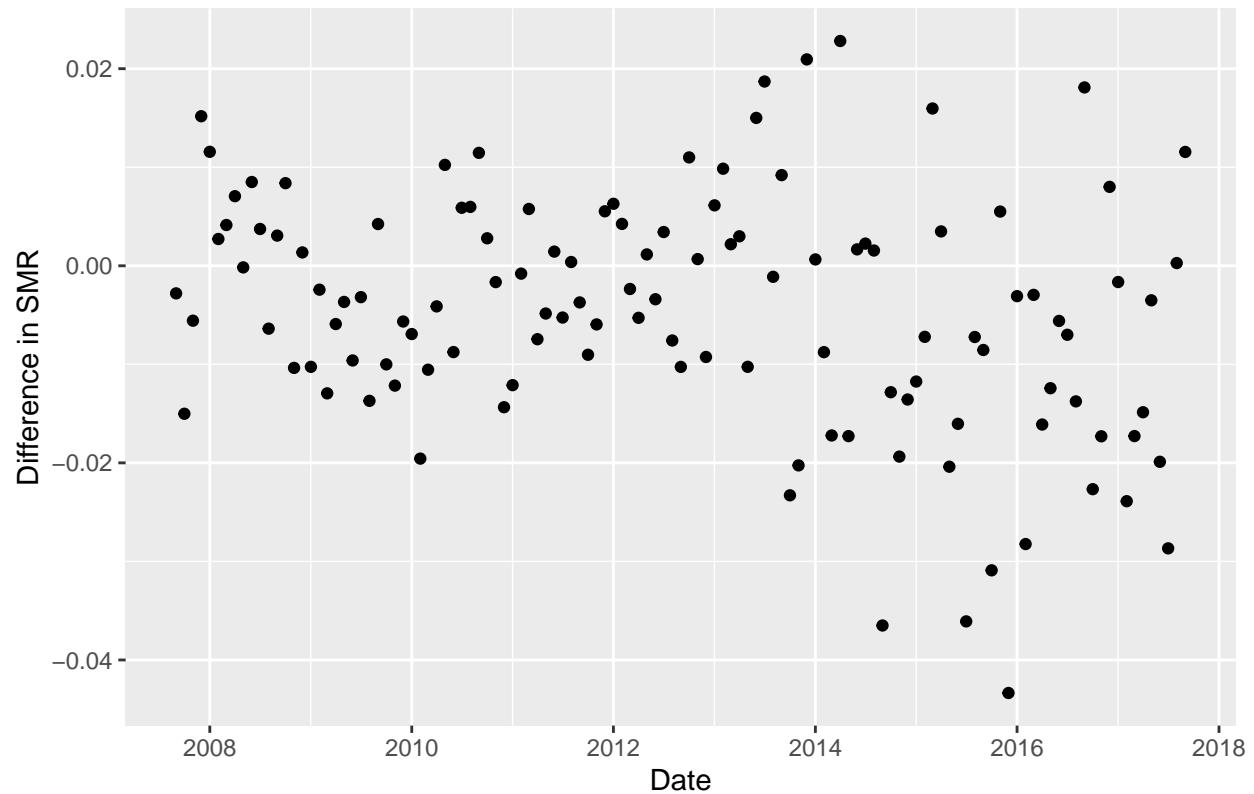
```
##  
## [[17]]
```

Difference in SMR of ILchicago Between Day and Night per month



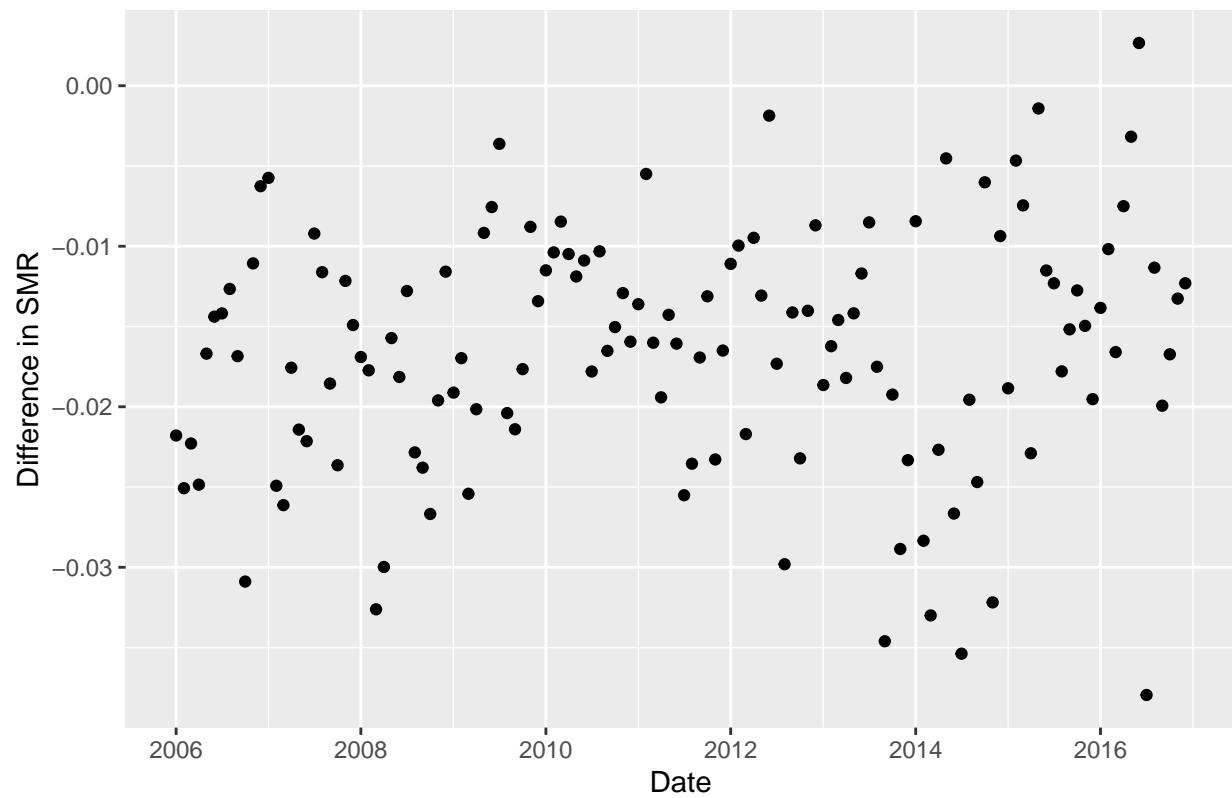
```
##  
## [[18]]
```

Difference in SMR of INfortwayne Between Day and Night per month



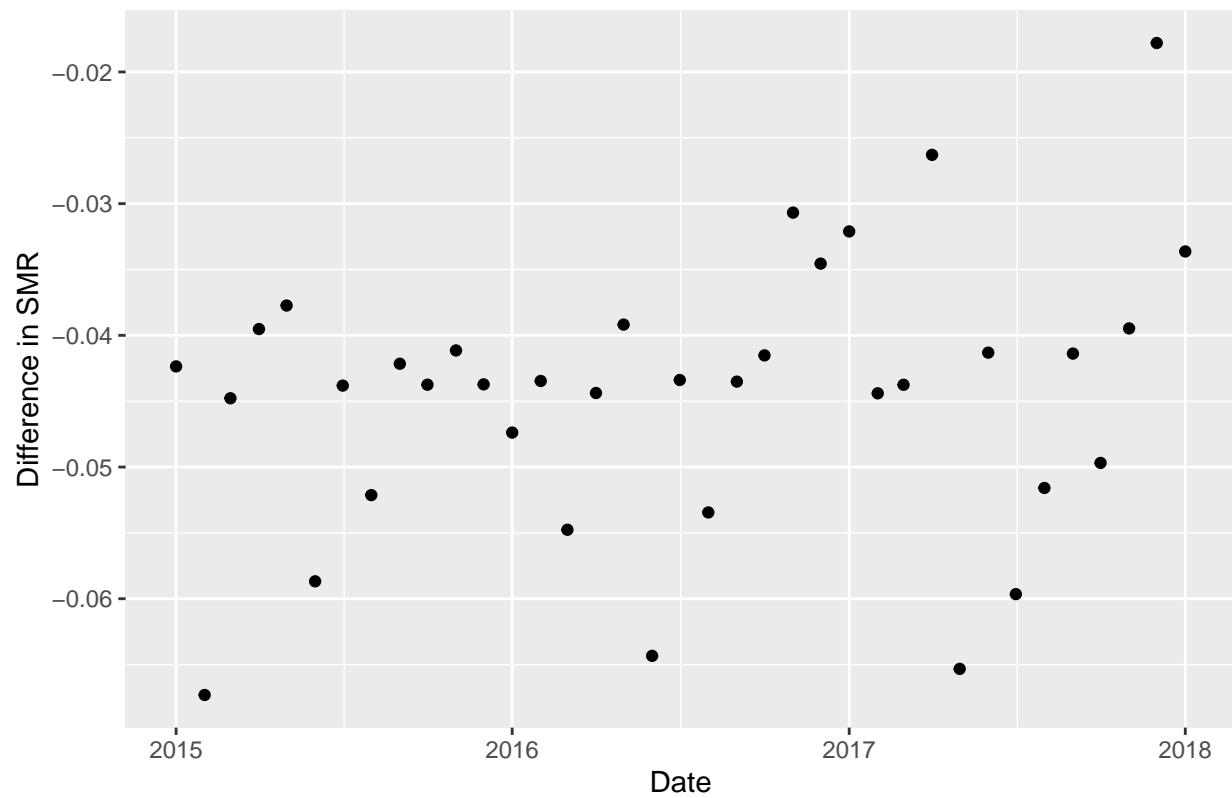
```
##  
## [[19]]
```

Difference in SMR of KSwichita Between Day and Night per month



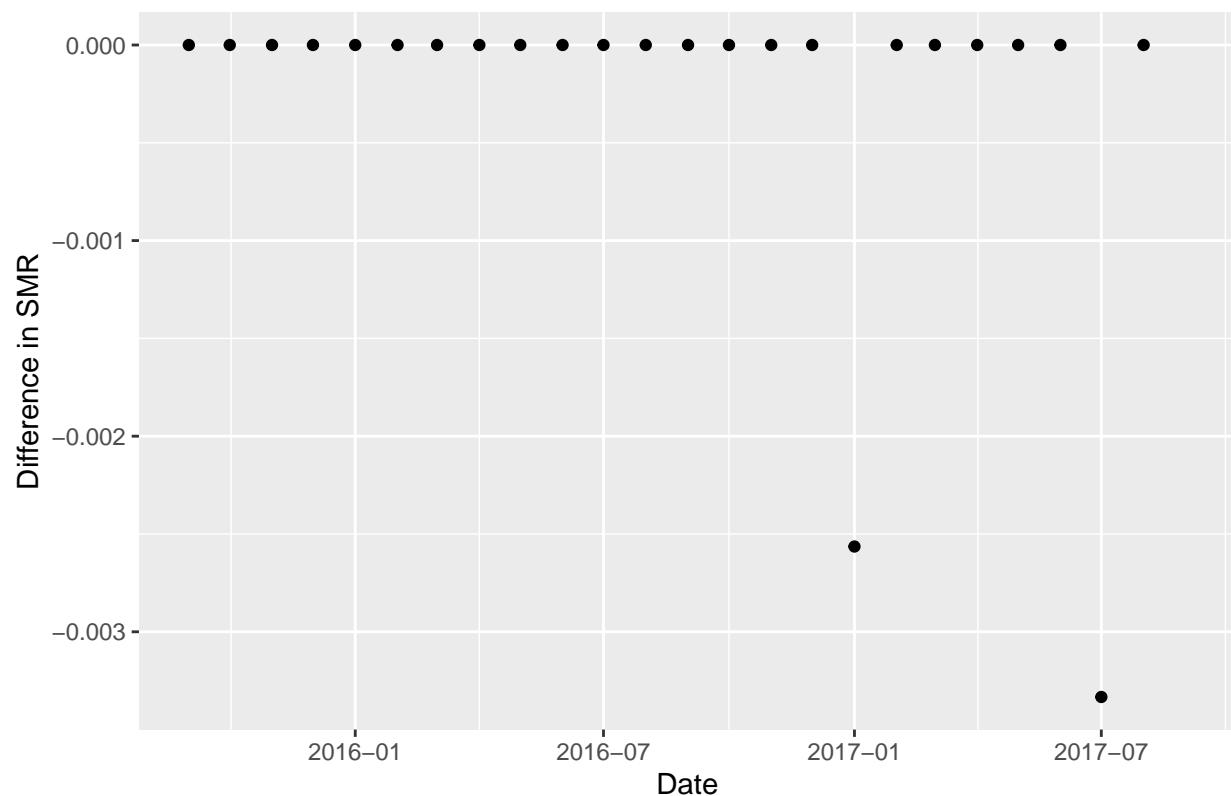
```
##  
## [[20]]
```

Difference in SMR of KYlouisville Between Day and Night per month



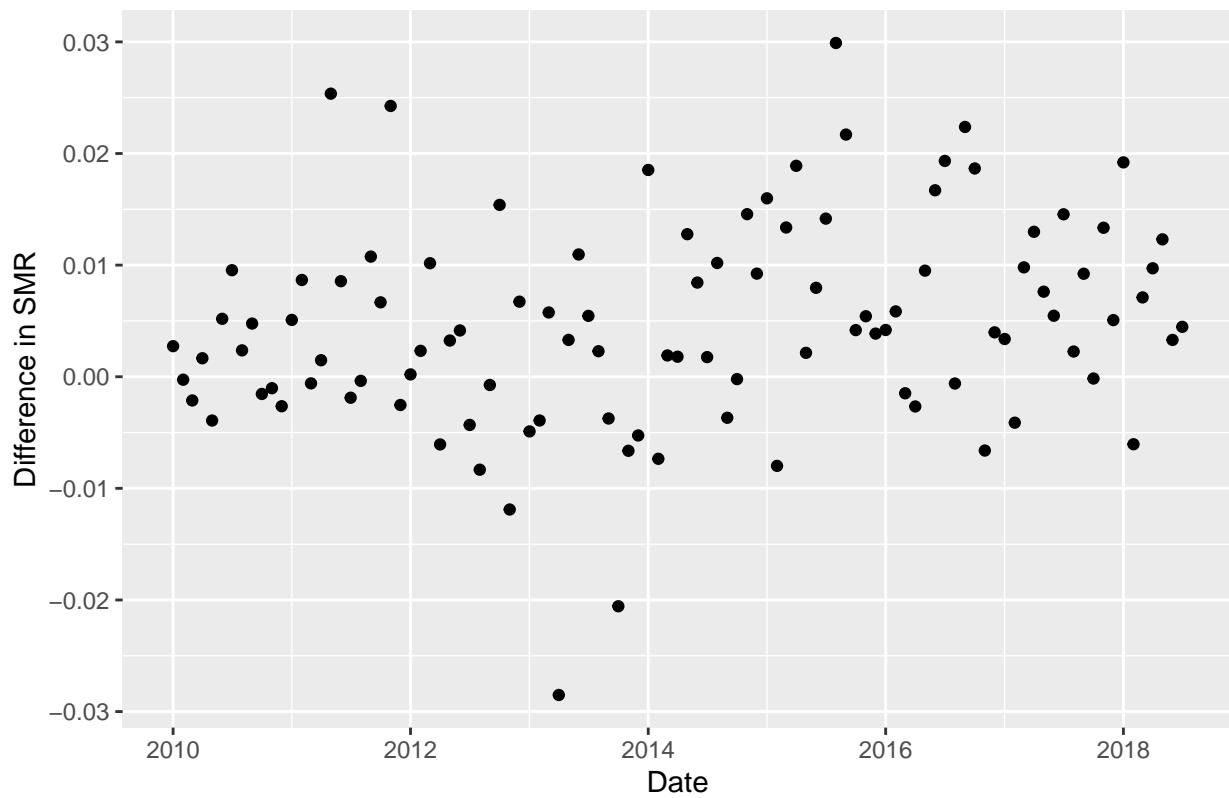
```
##  
## [[21]]
```

Difference in SMR of KYowensboro Between Day and Night per month



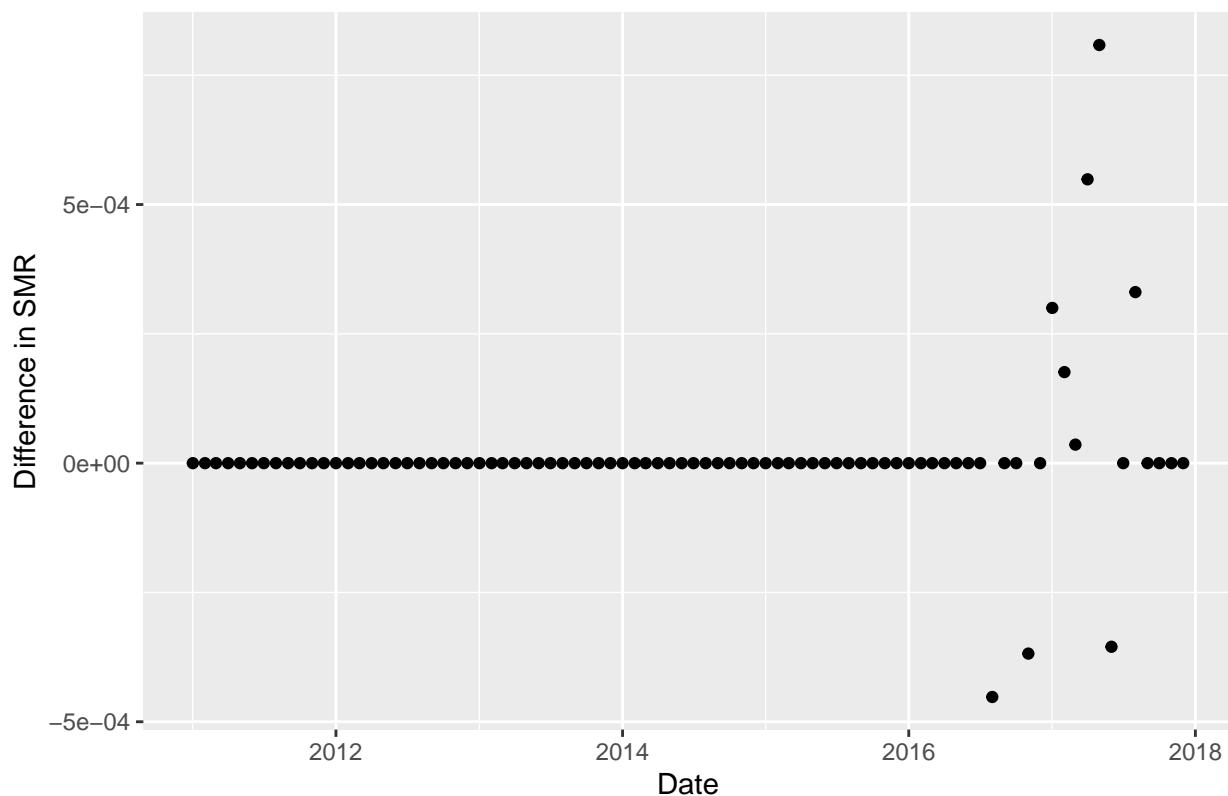
```
##  
## [[22]]
```

Difference in SMR of LAneworleans Between Day and Night per month

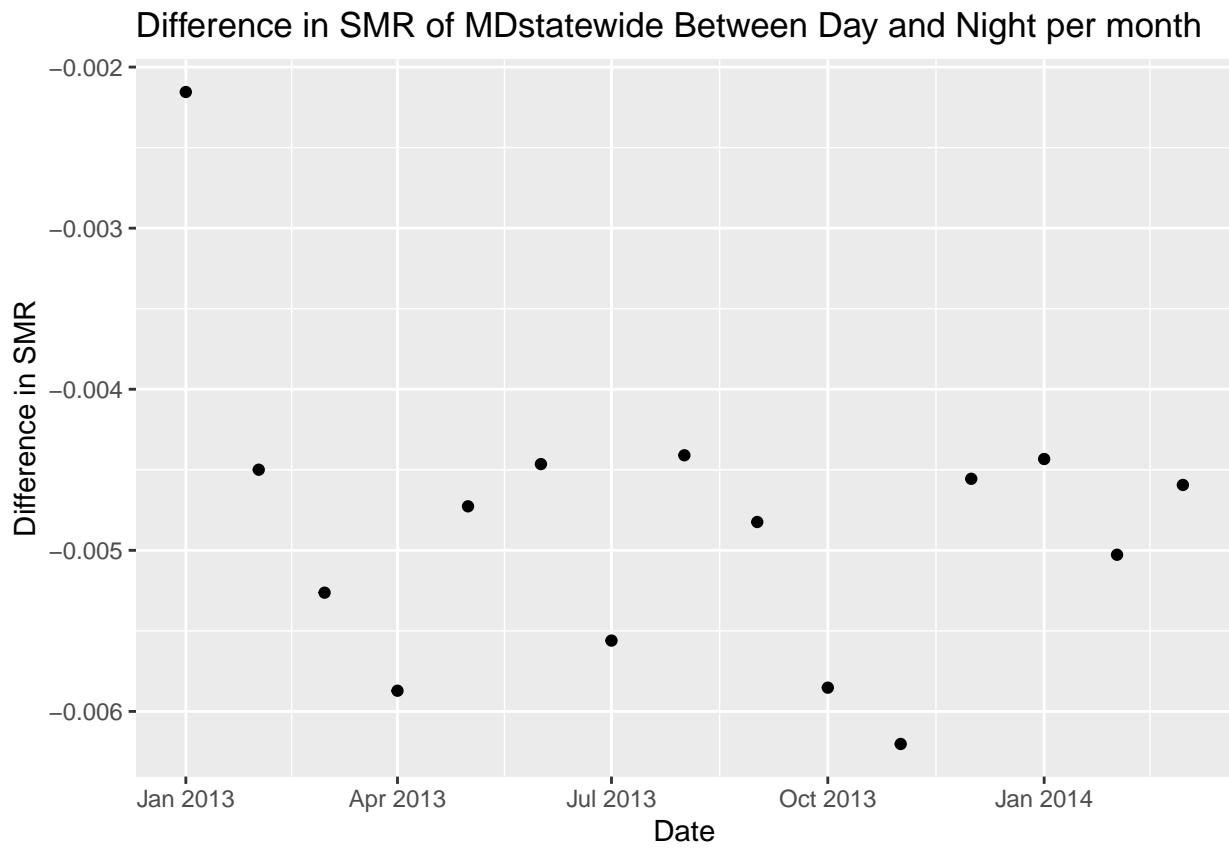


```
##  
## [[23]]
```

Difference in SMR of MDbaltimore Between Day and Night per month

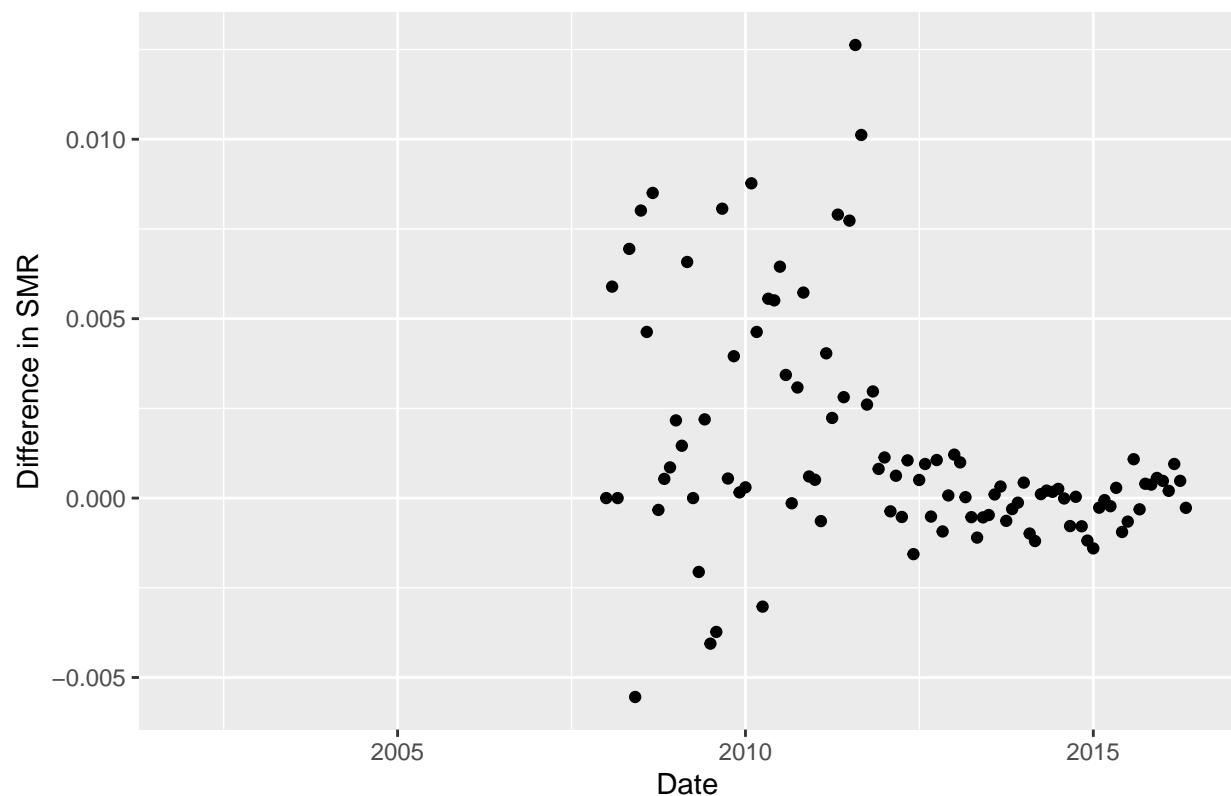


```
##  
## [[24]]
```



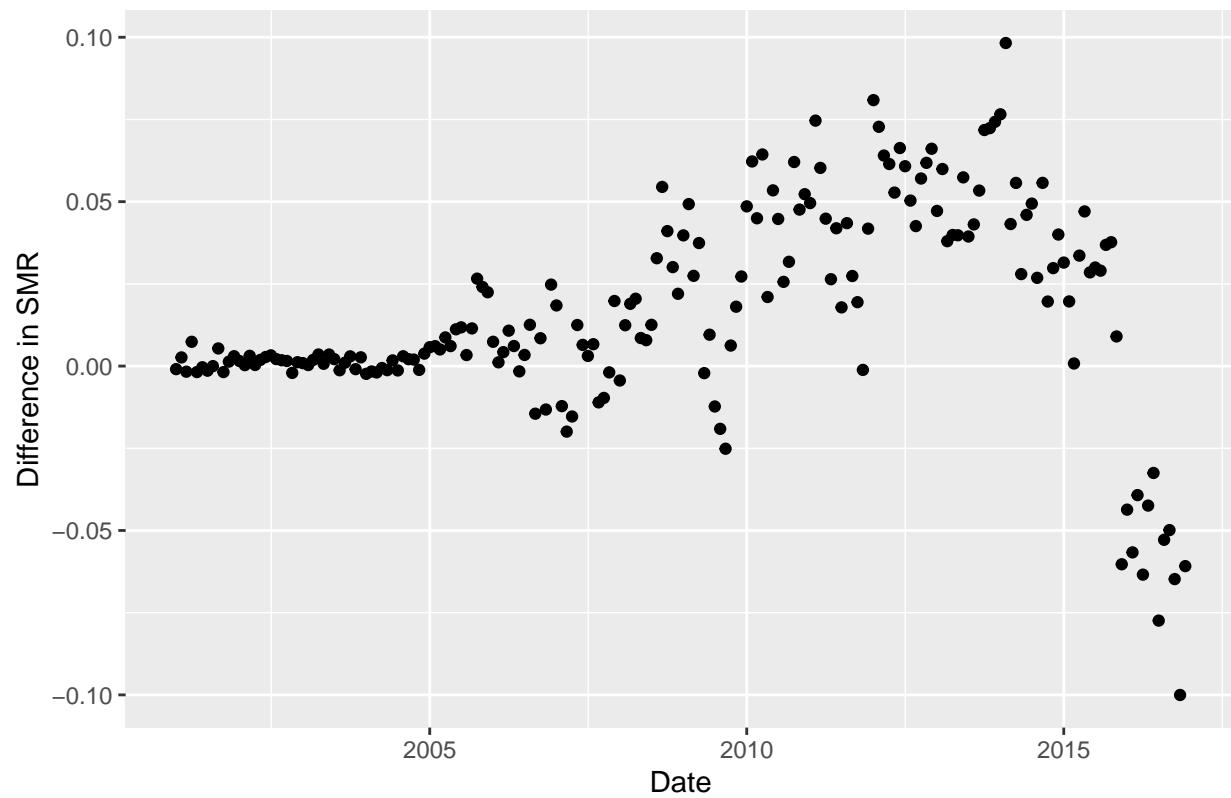
```
##  
## [[25]]
```

Difference in SMR of MI statewide Between Day and Night per month



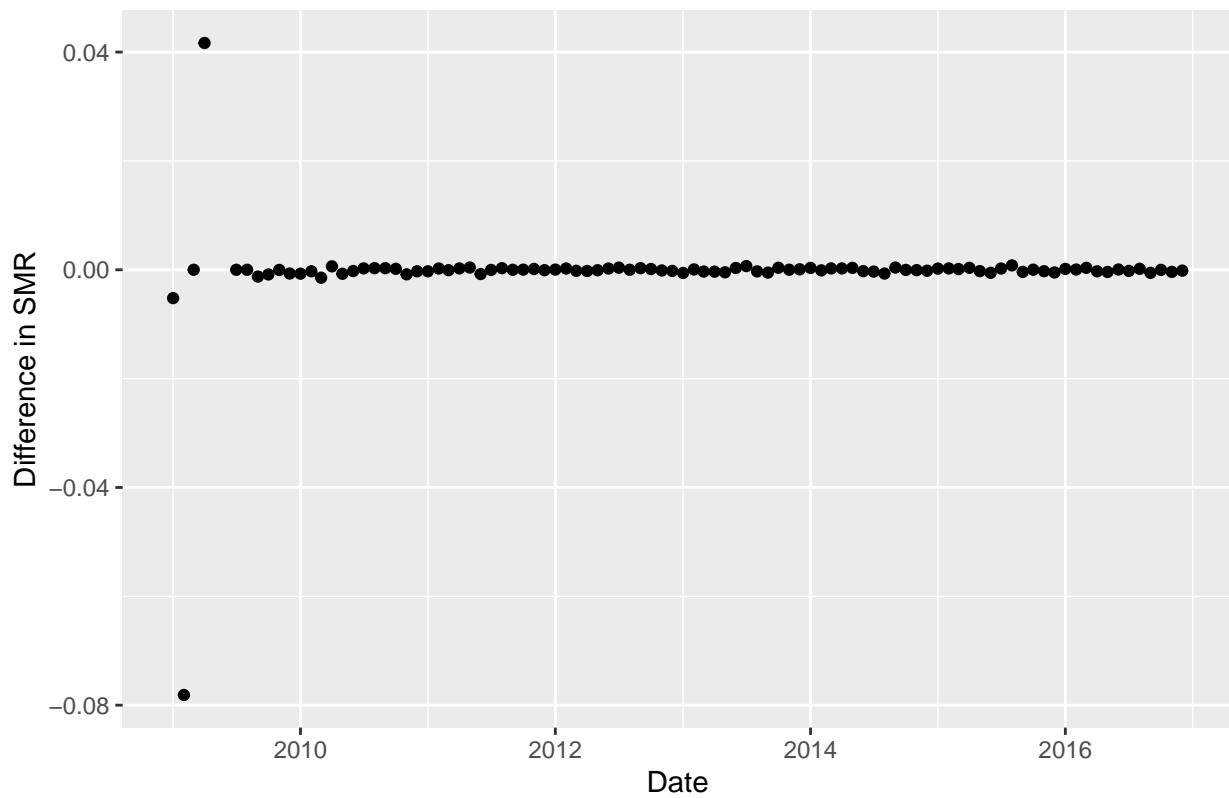
```
##  
## [[26]]
```

Difference in SMR of MNsaintpaul Between Day and Night per month



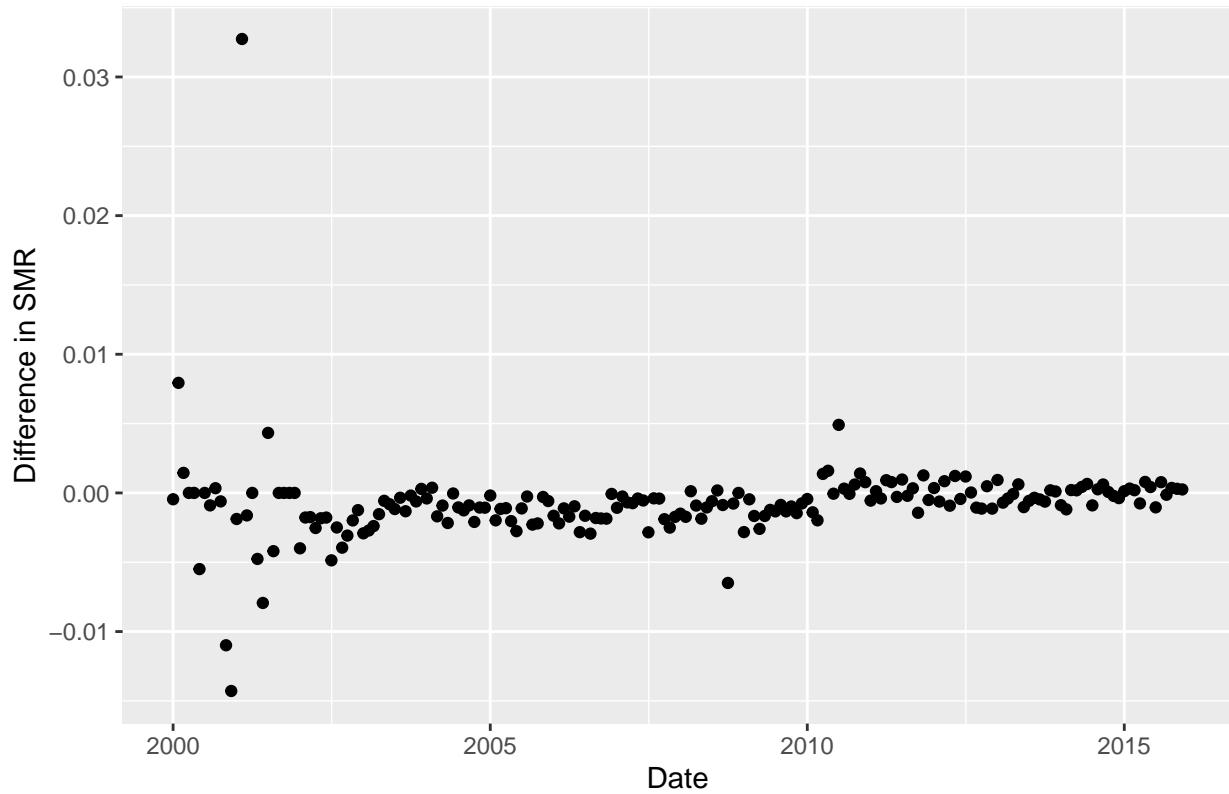
```
##  
## [[27]]
```

Difference in SMR of MT statewide Between Day and Night per month



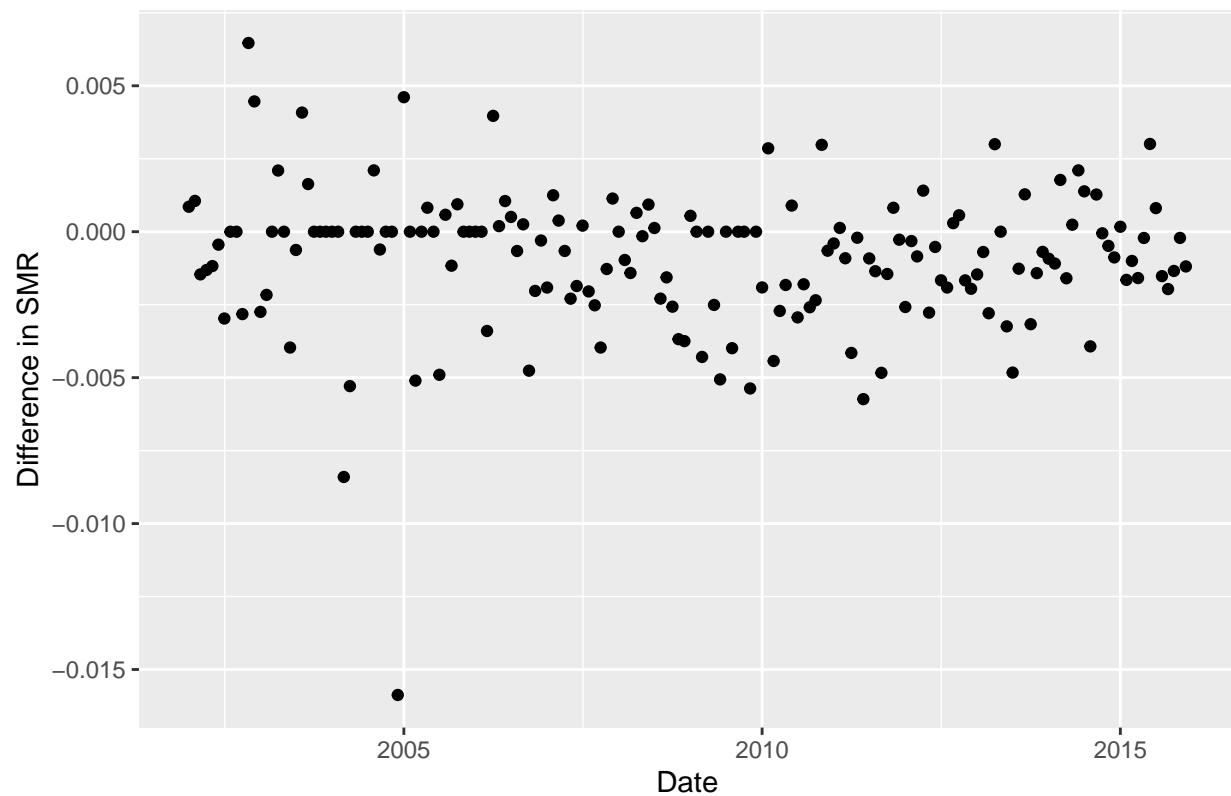
```
##  
## [[28]]
```

Difference in SMR of NCcharlotte Between Day and Night per month



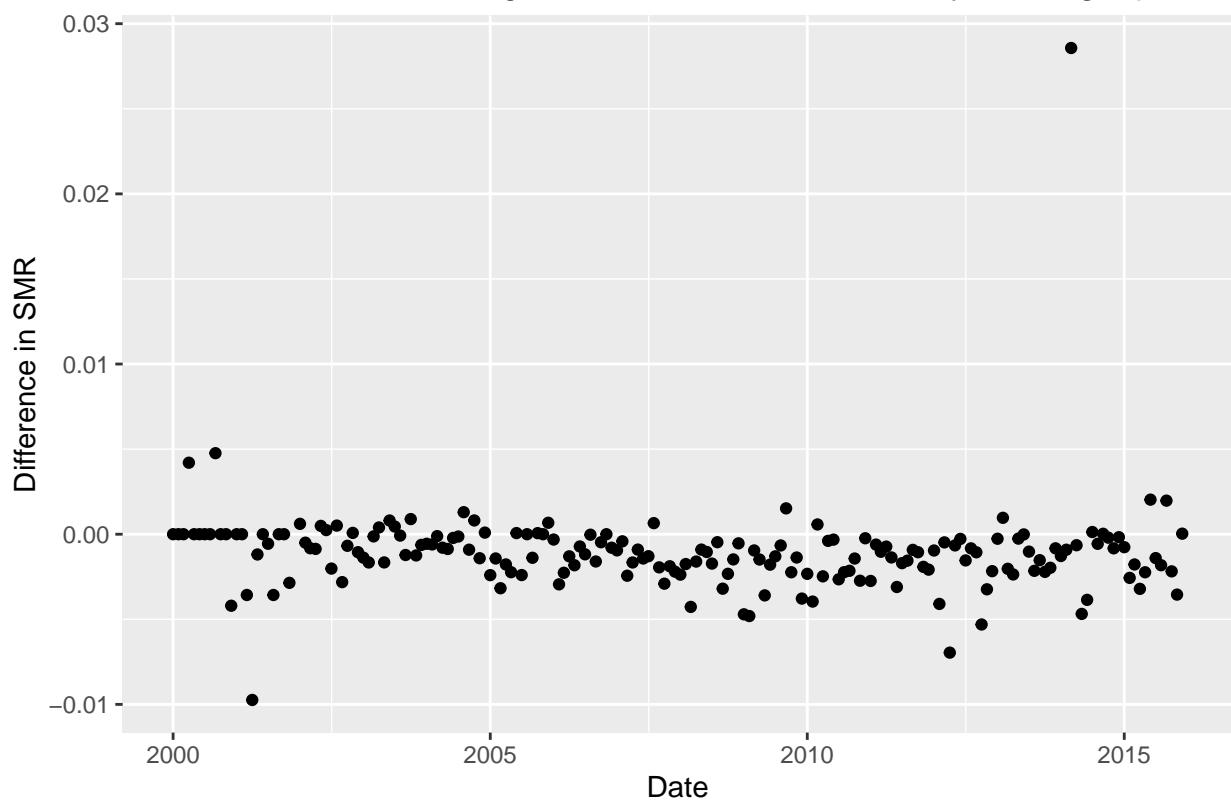
```
##  
## [[29]]
```

Difference in SMR of NCdurham Between Day and Night per month



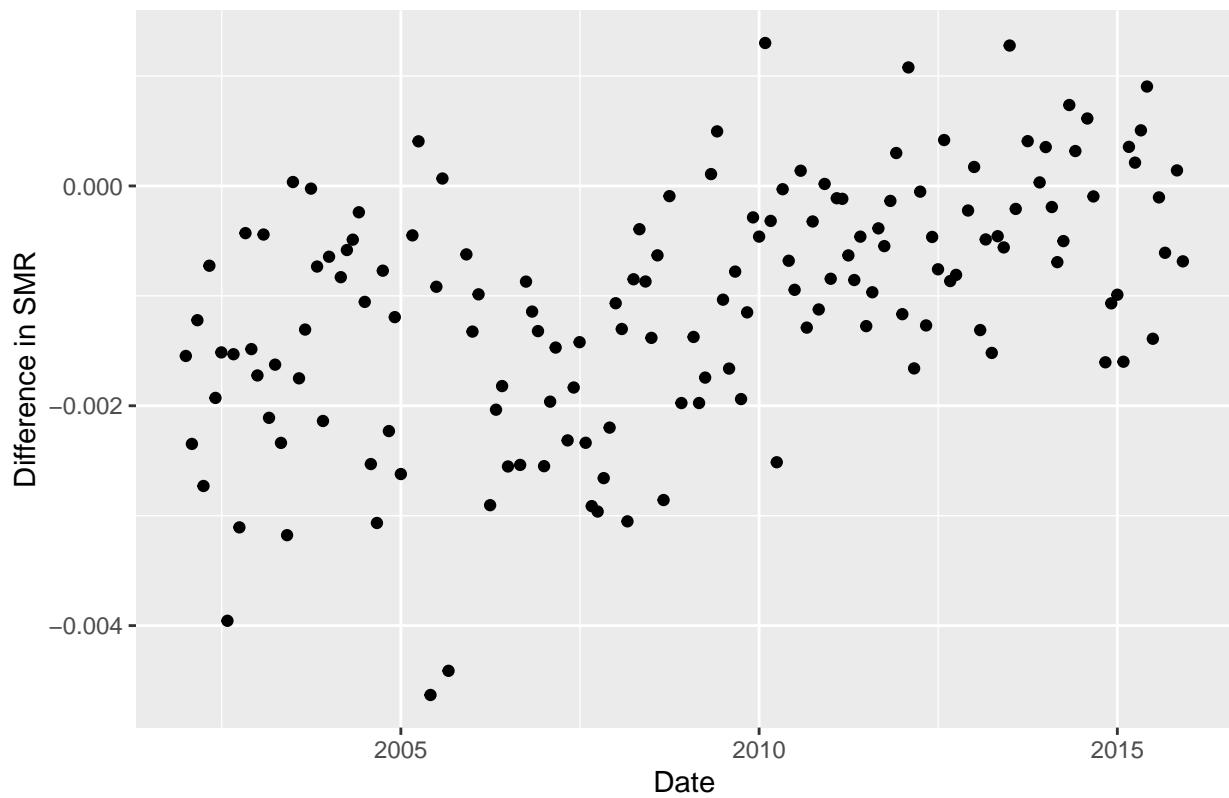
```
##  
## [[30]]
```

Difference in SMR of NCgreensboro2020 Between Day and Night per moi



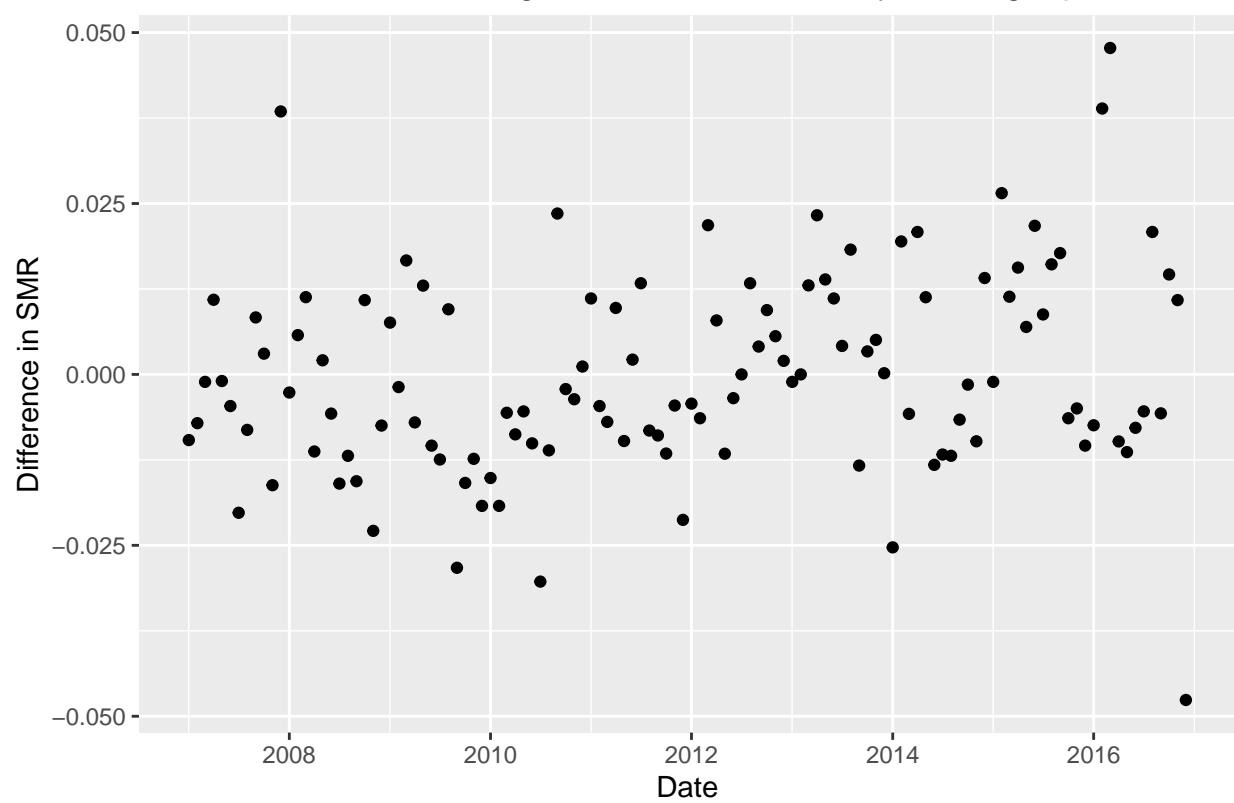
```
##  
## [[31]]
```

Difference in SMR of NCraleigh Between Day and Night per month



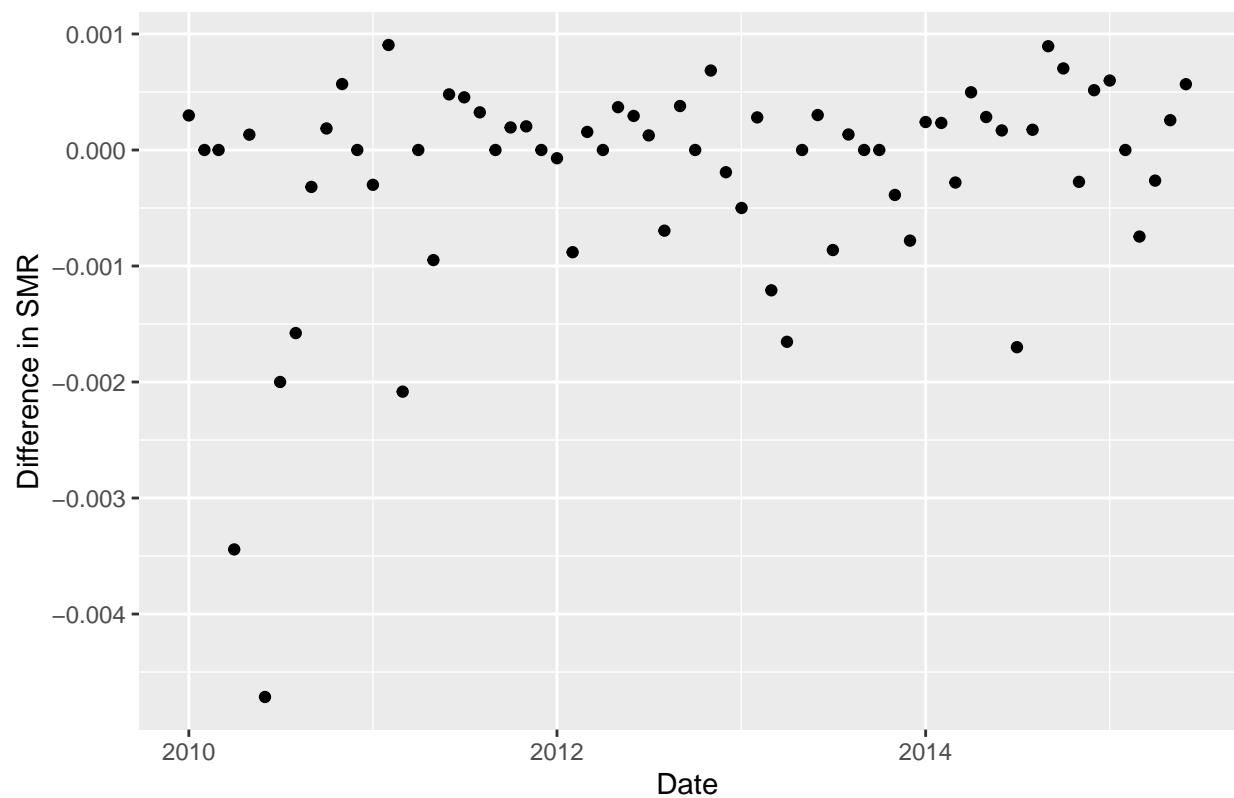
```
##  
## [[32]]
```

Difference in SMR of NDgrandforks Between Day and Night per month



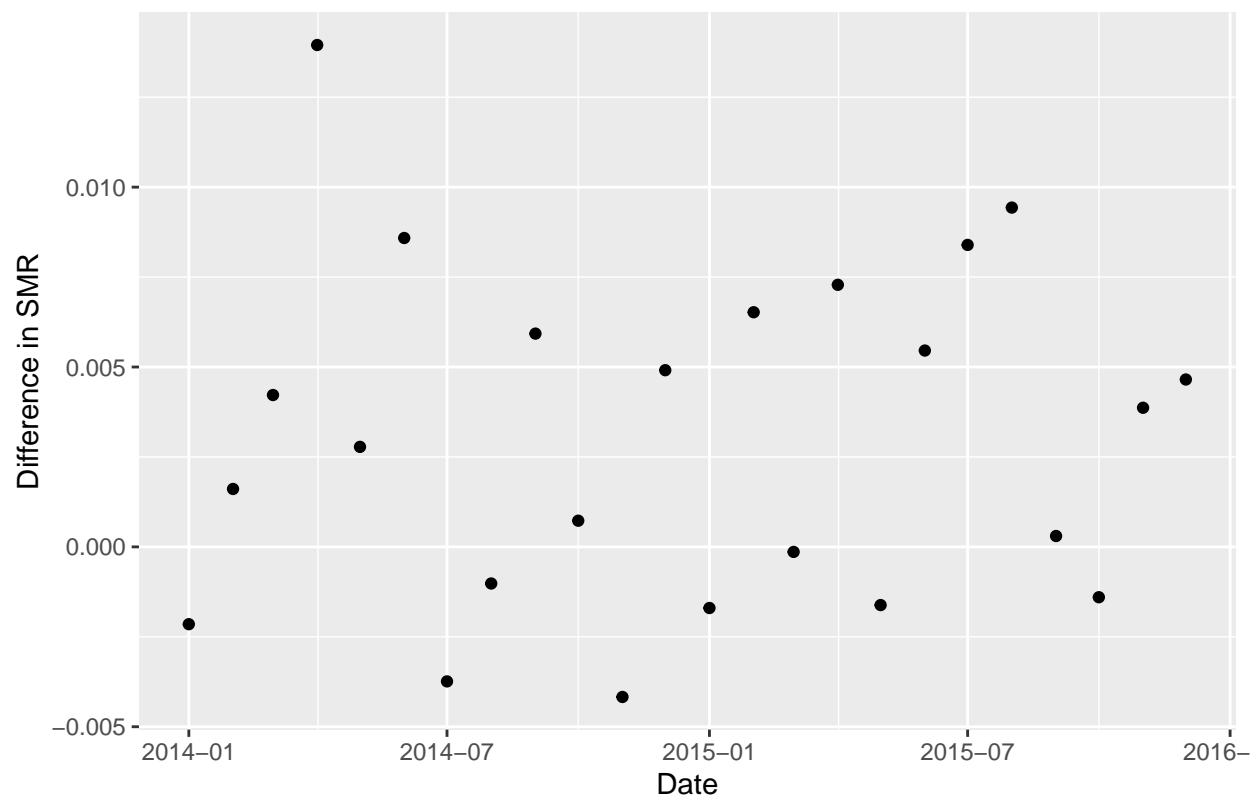
```
##  
## [[33]]
```

Difference in SMR of ND statewide Between Day and Night per month



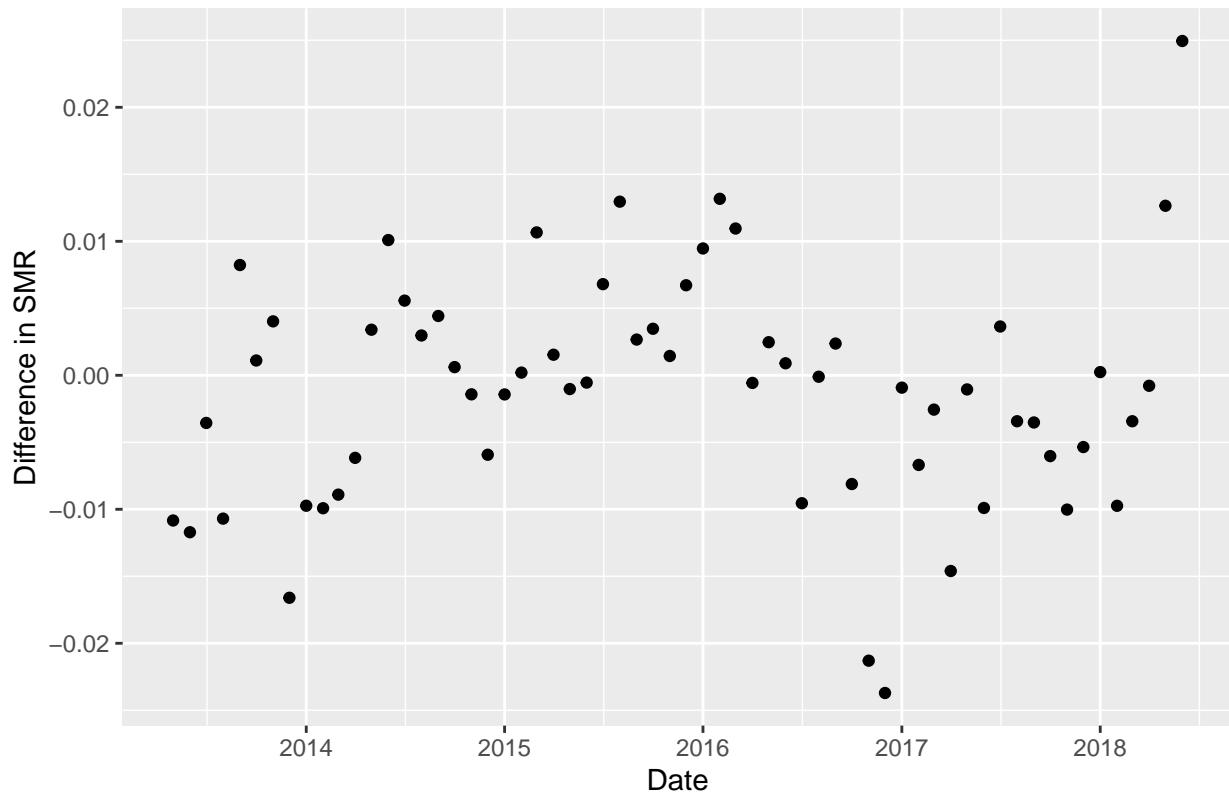
```
##  
## [[34]]
```

Difference in SMR of NH statewide Between Day and Night per month



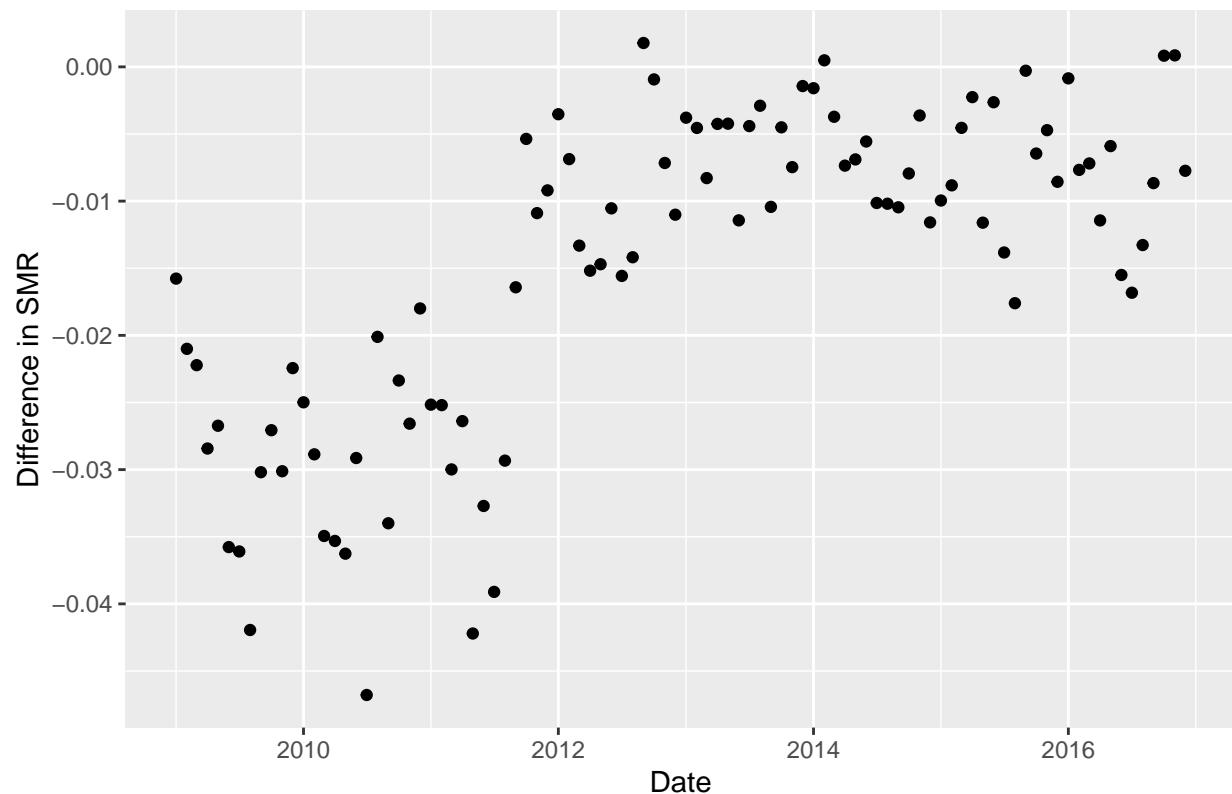
```
##  
## [[35]]
```

Difference in SMR of NJcamden Between Day and Night per month



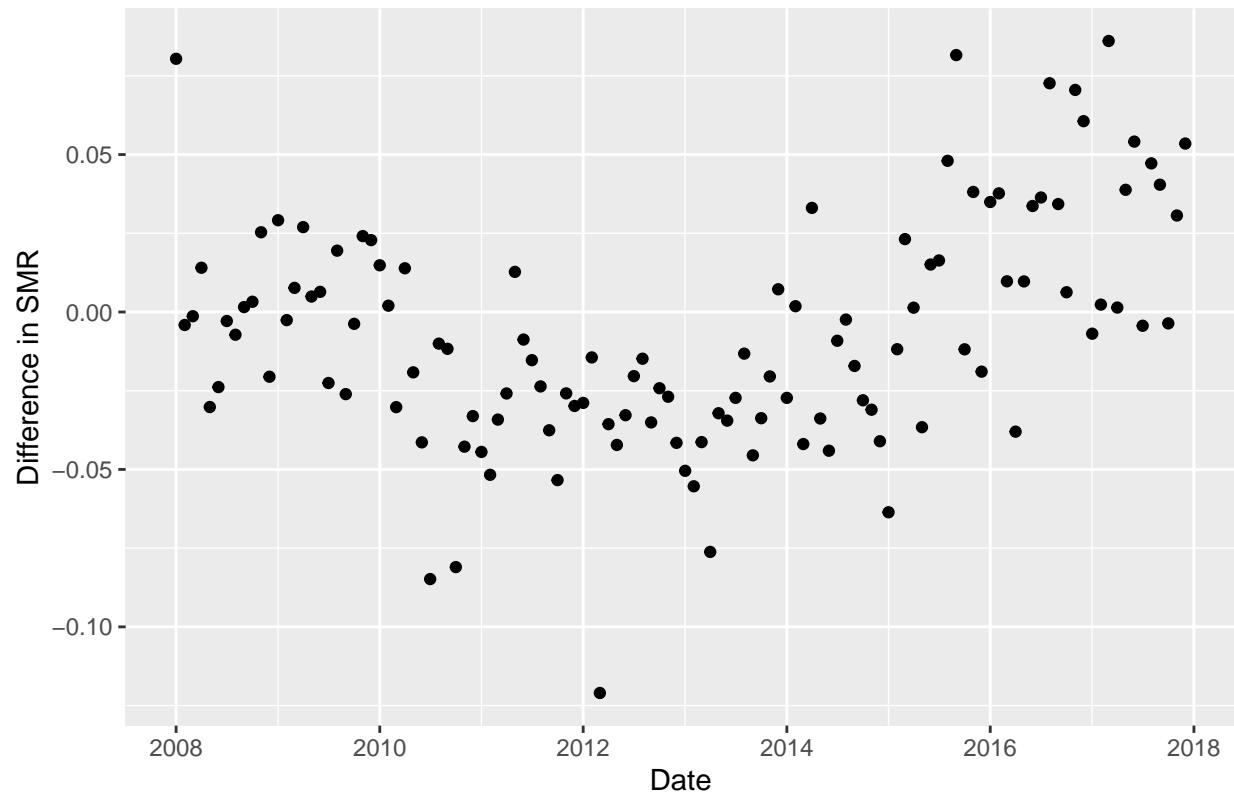
```
##  
## [[36]]
```

Difference in SMR of NJstatewide Between Day and Night per month



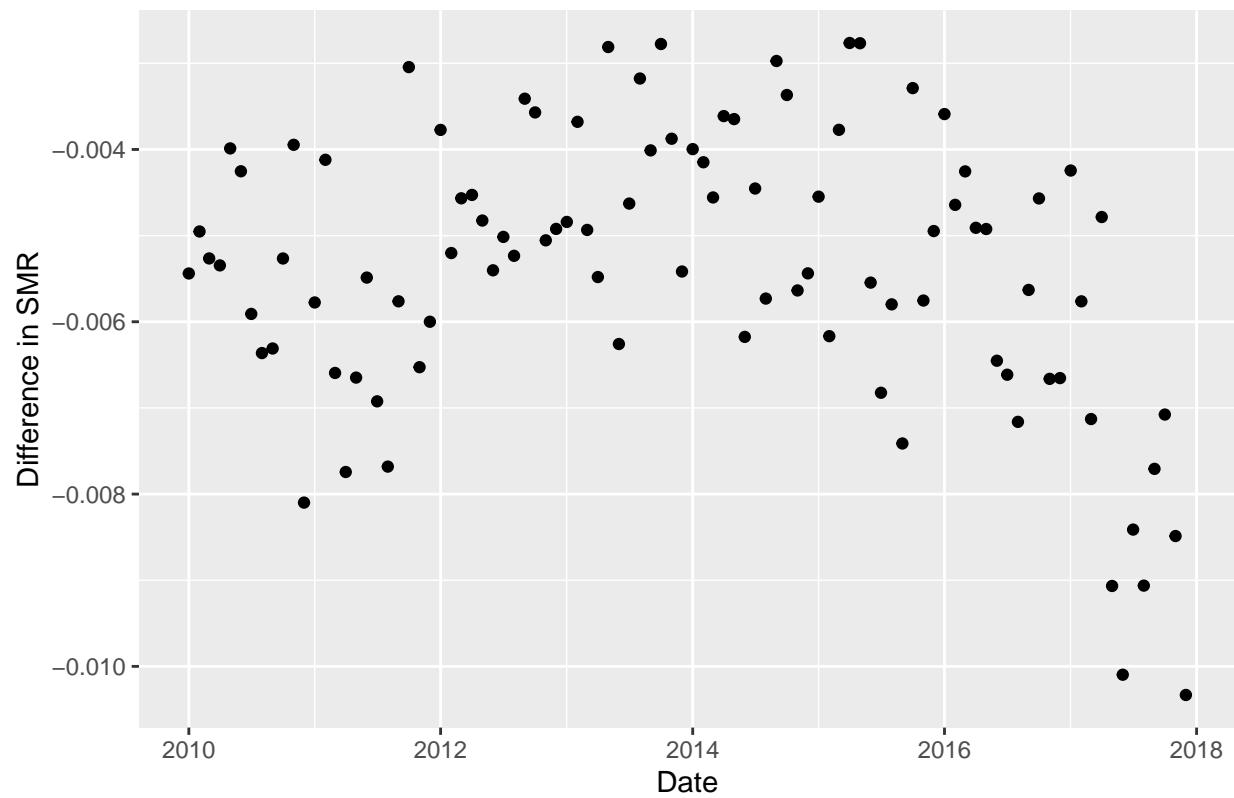
```
##  
## [[37]]
```

Difference in SMR of NYalbany Between Day and Night per month



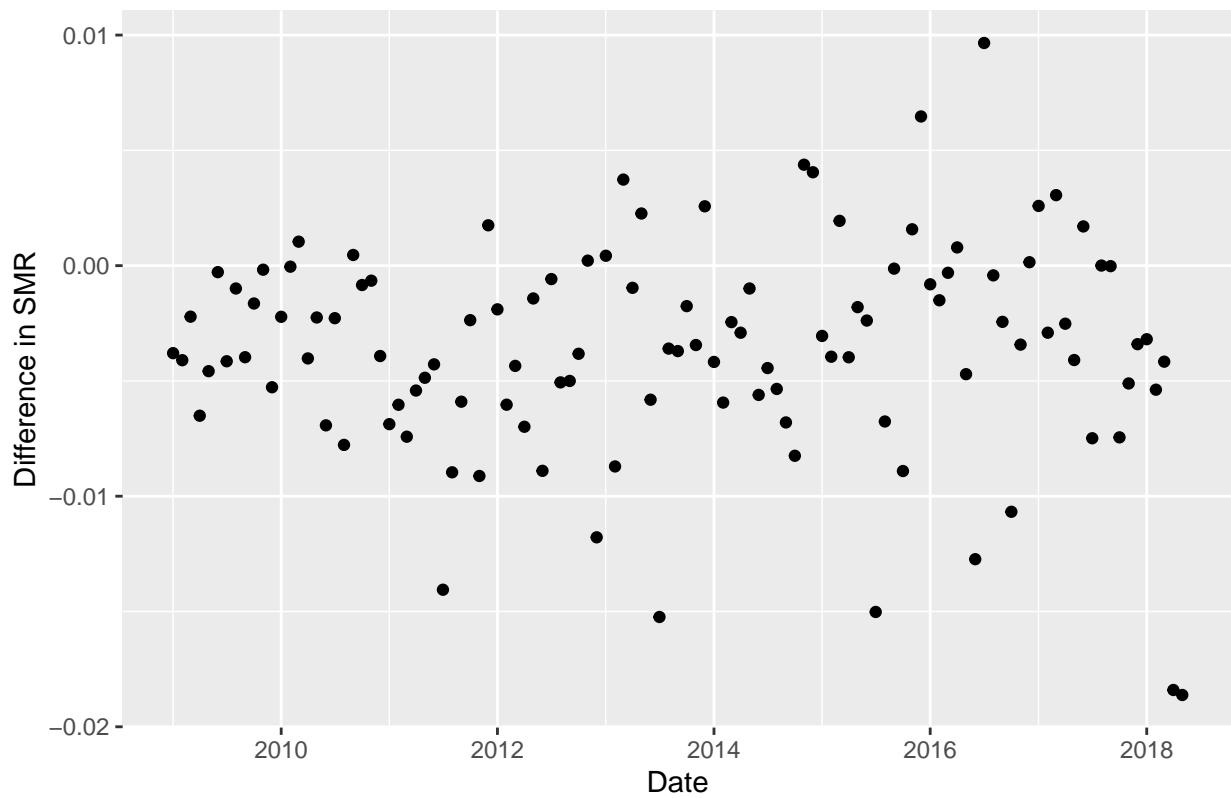
```
##  
## [[38]]
```

Difference in SMR of NY statewide Between Day and Night per month

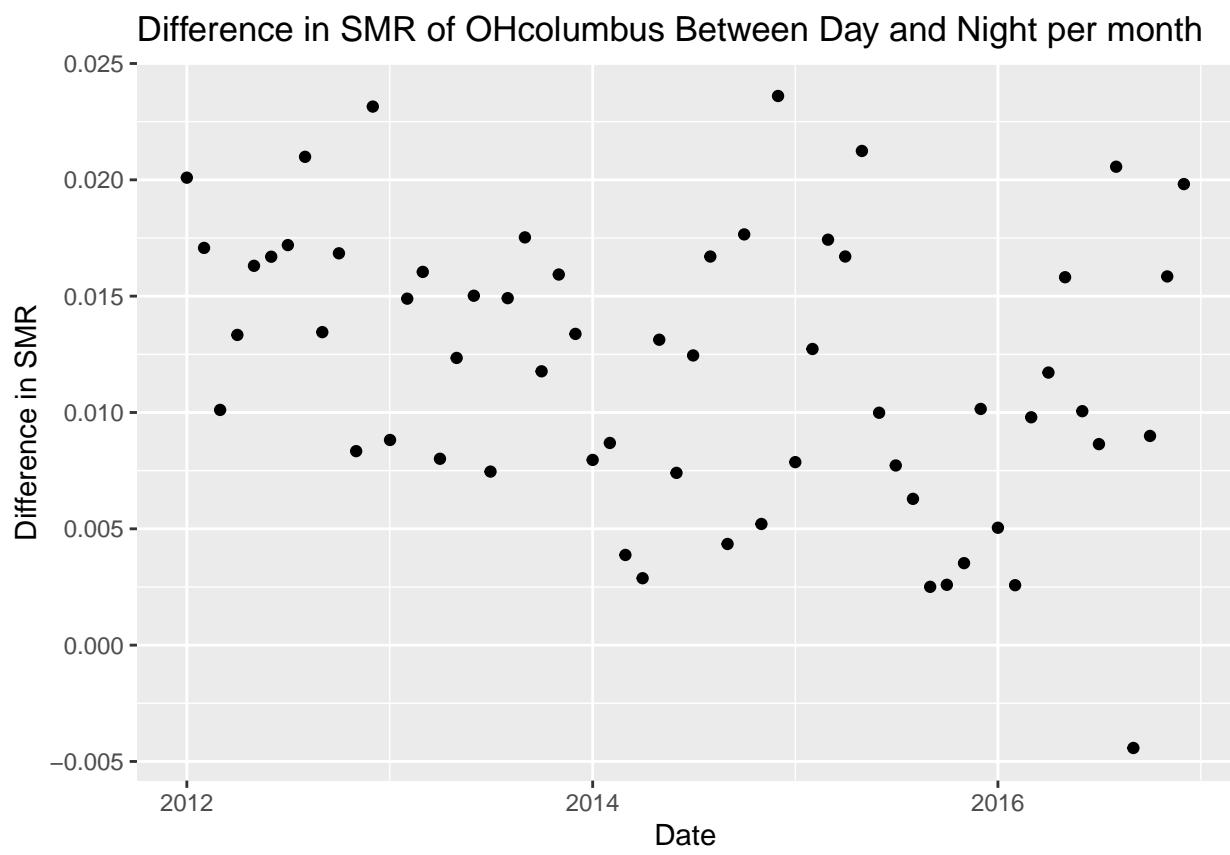


```
##  
## [[39]]
```

Difference in SMR of OHcincinnati Between Day and Night per month

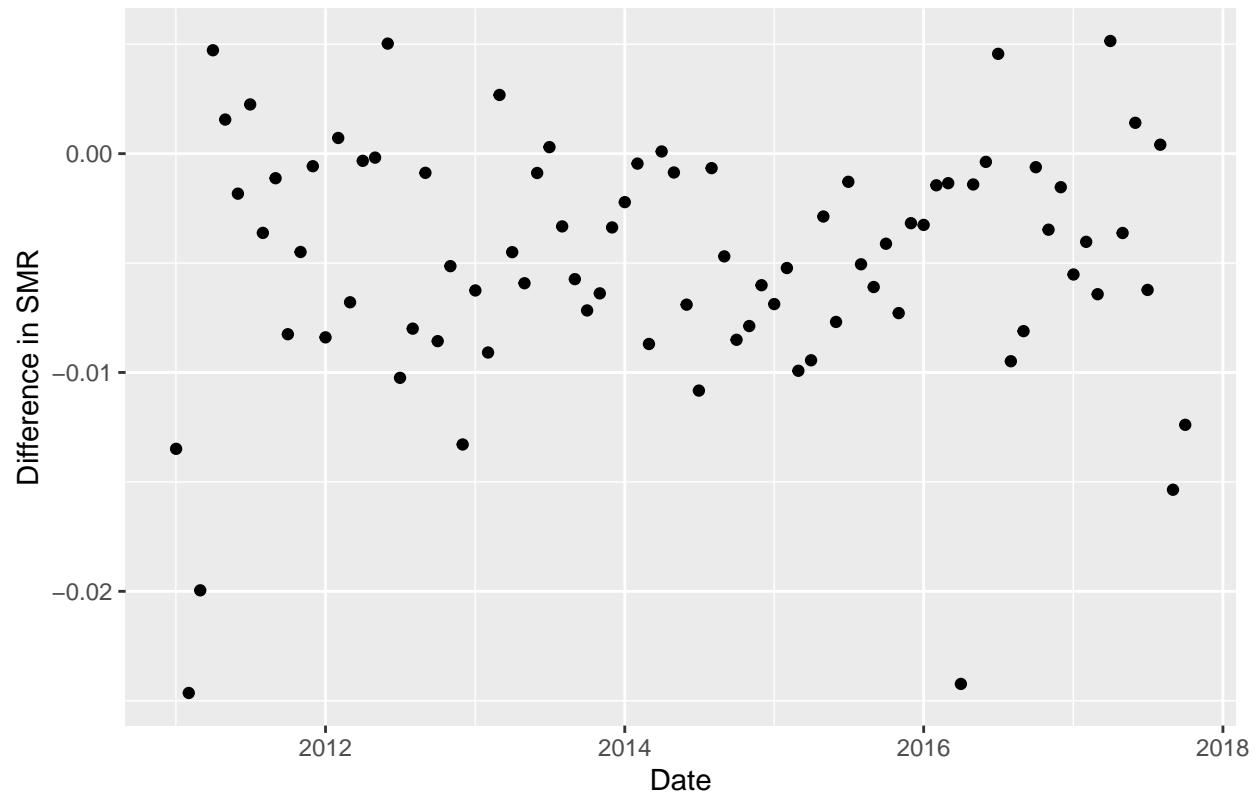


```
##  
## [[40]]
```



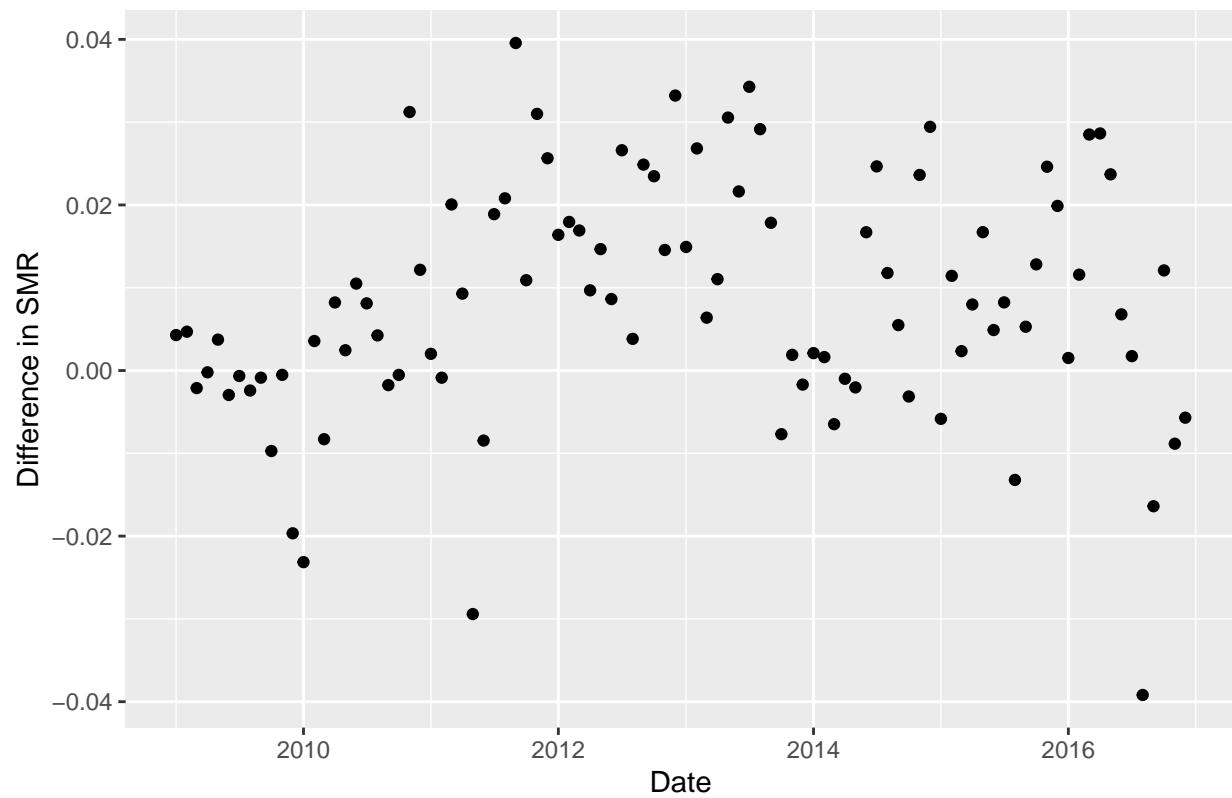
```
##  
## [[41]]
```

Difference in SMR of OKoklahomacity Between Day and Night per month



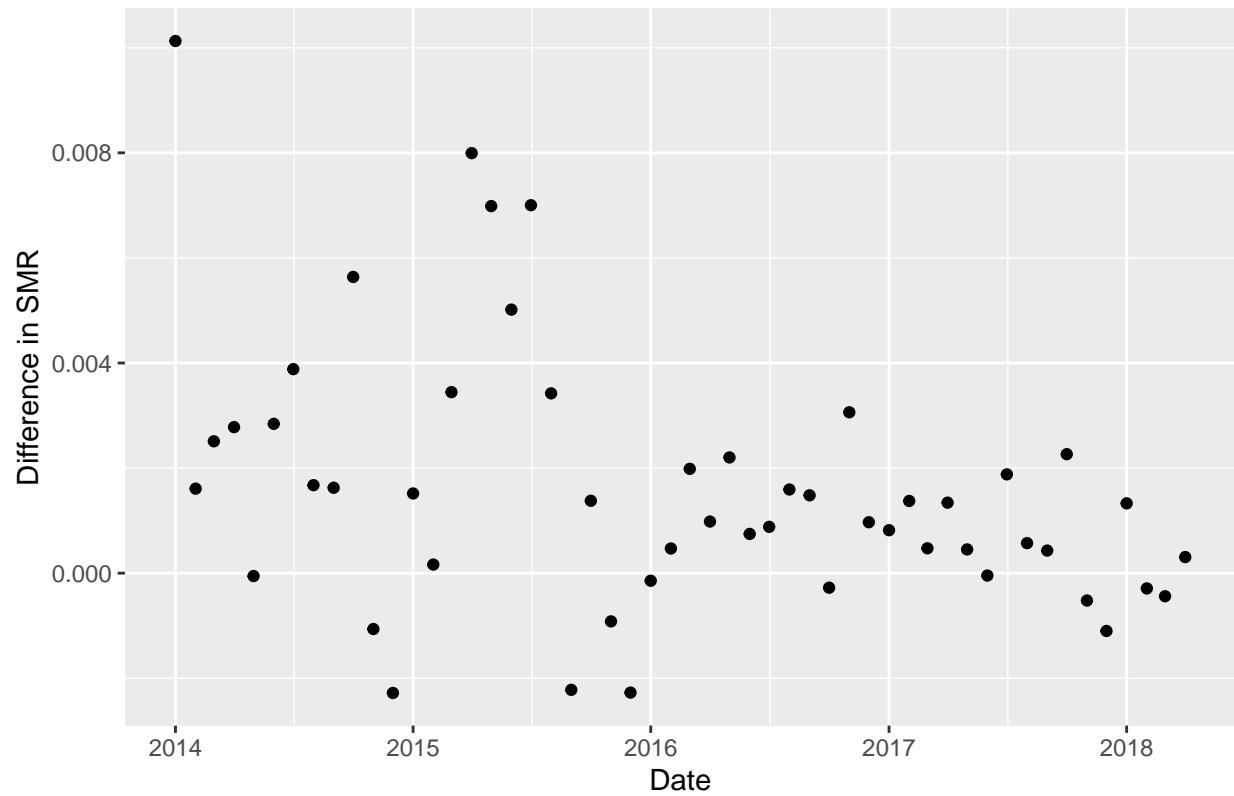
```
##  
## [[42]]
```

Difference in SMR of OKtulsa Between Day and Night per month



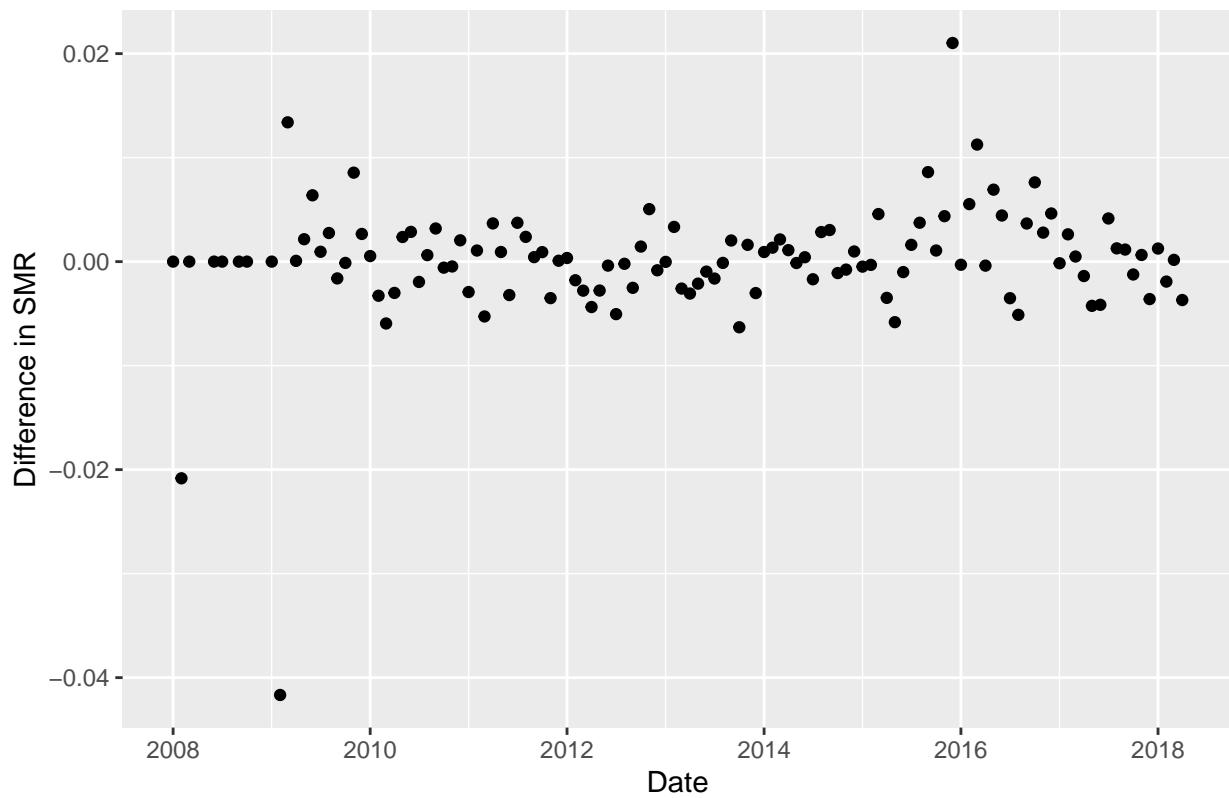
```
##  
## [[43]]
```

Difference in SMR of PPhiladelphia Between Day and Night per month



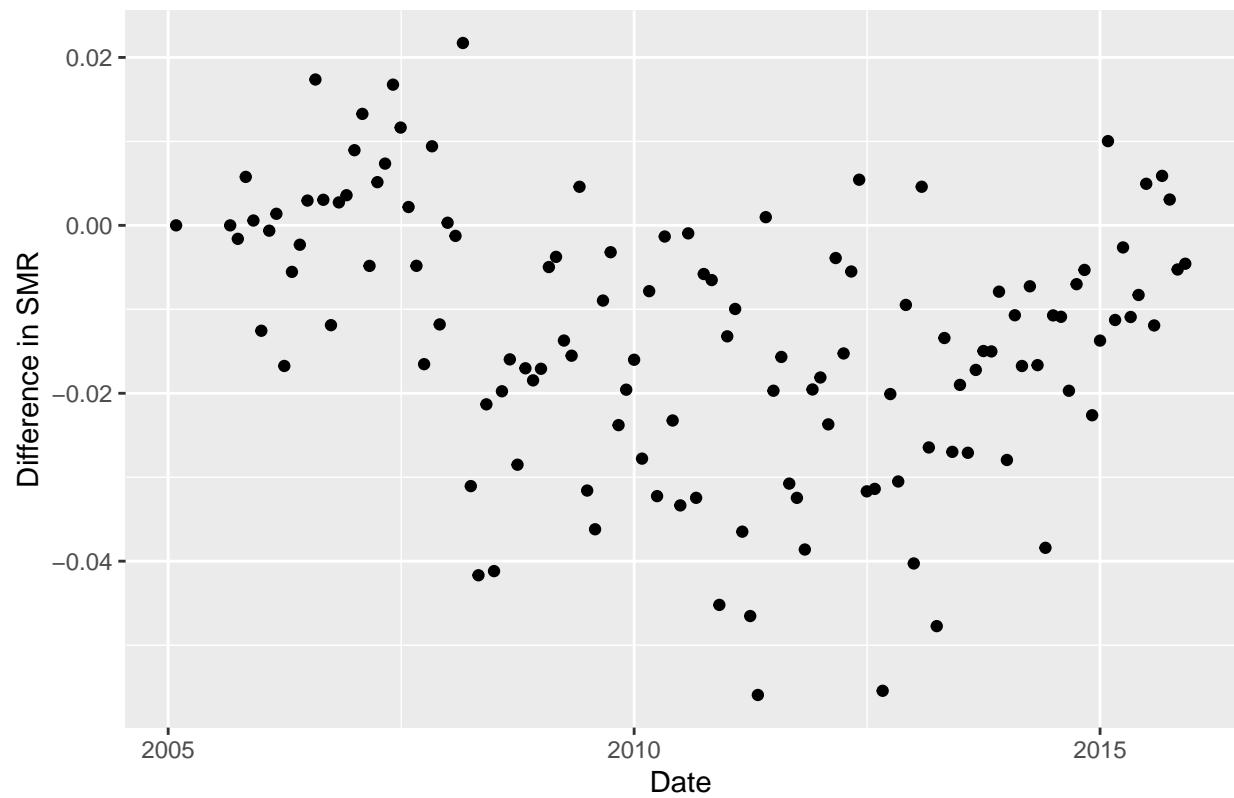
```
##  
## [[44]]
```

Difference in SMR of PApittsburgh Between Day and Night per month



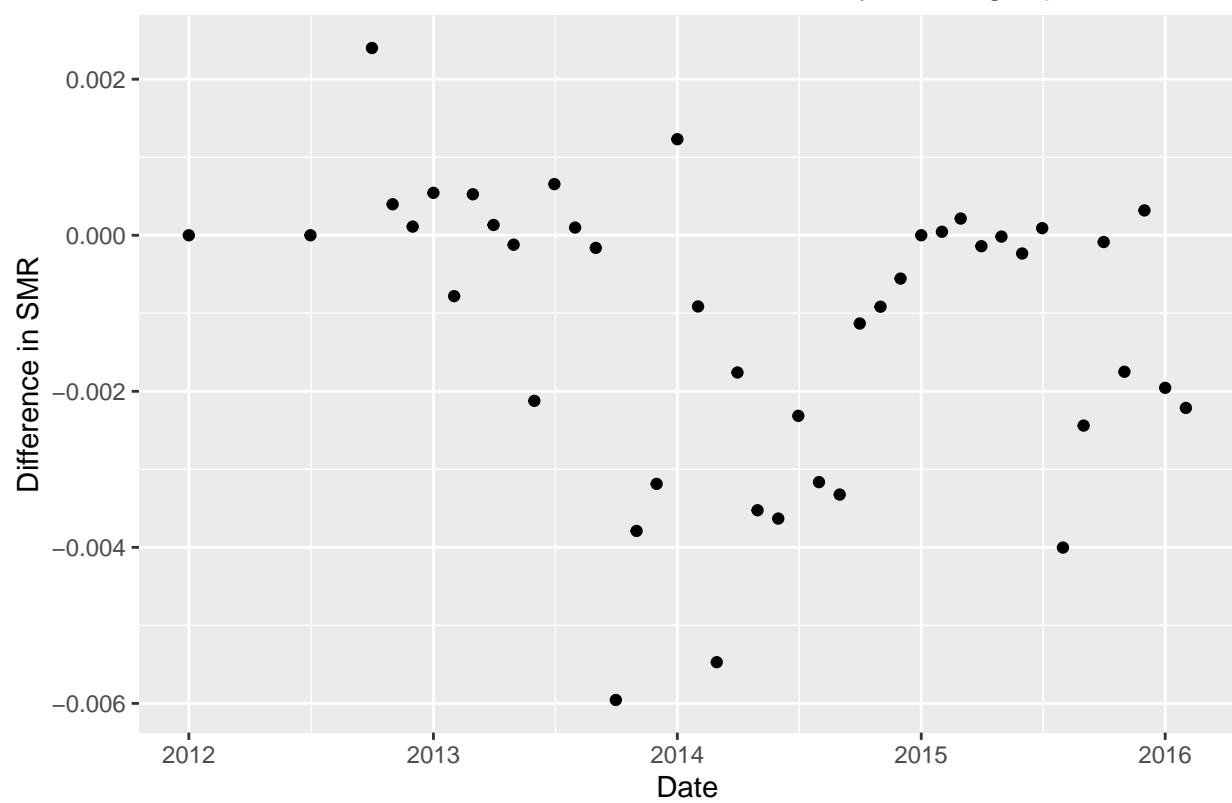
```
##  
## [[45]]
```

Difference in SMR of RI statewide Between Day and Night per month



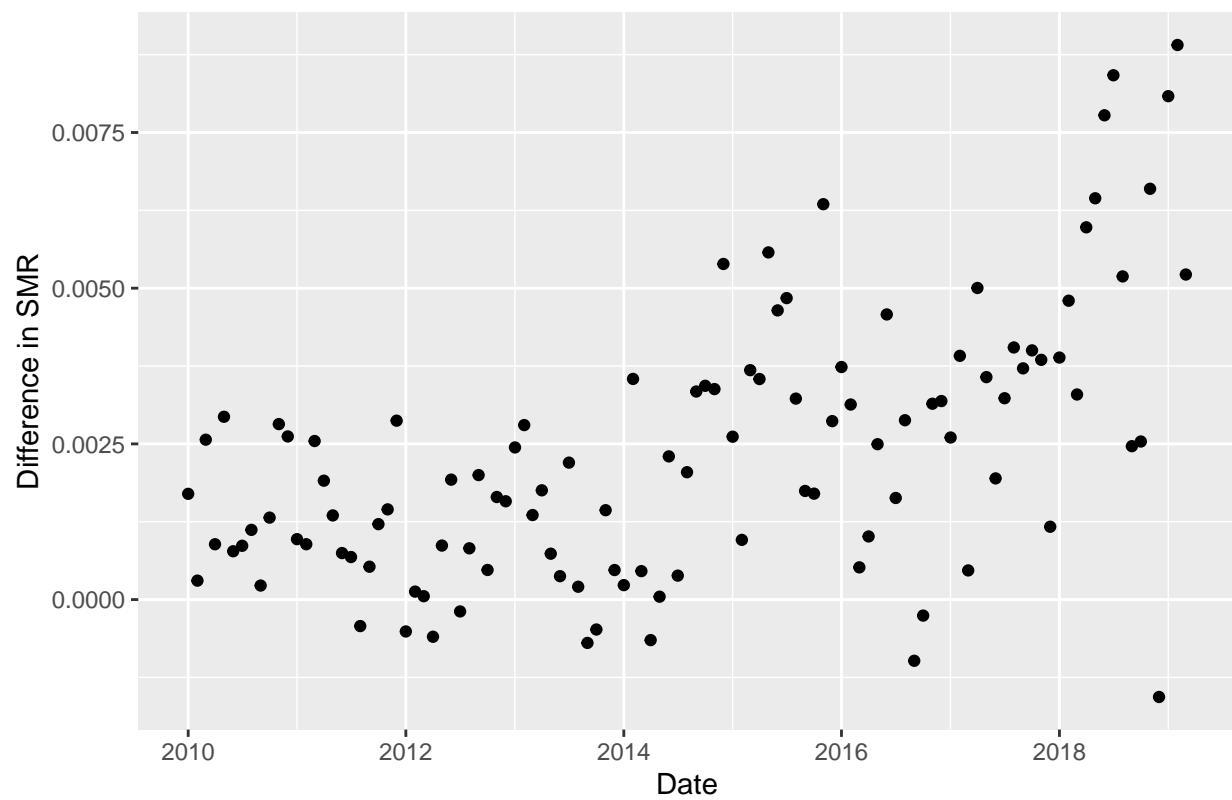
```
##  
## [[46]]
```

Difference in SMR of SD statewide Between Day and Night per month



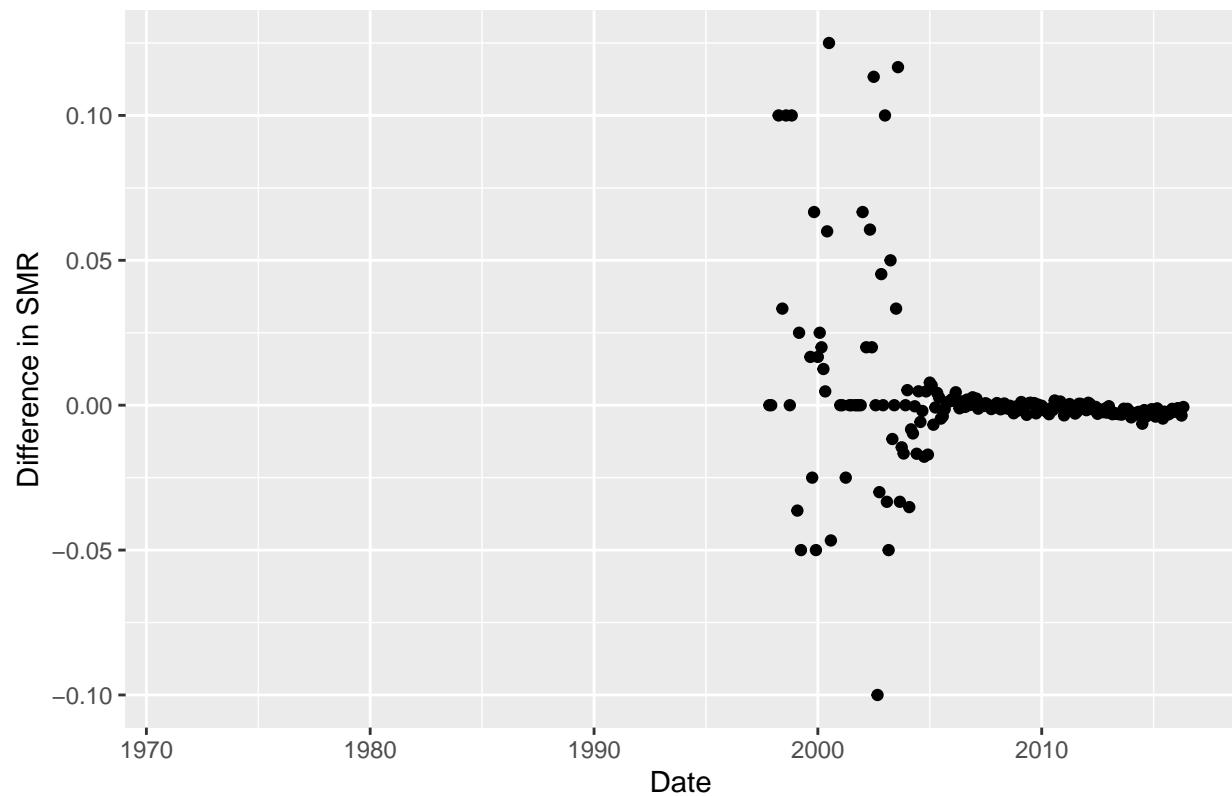
```
##  
## [[47]]
```

Difference in SMR of TNnashville Between Day and Night per month

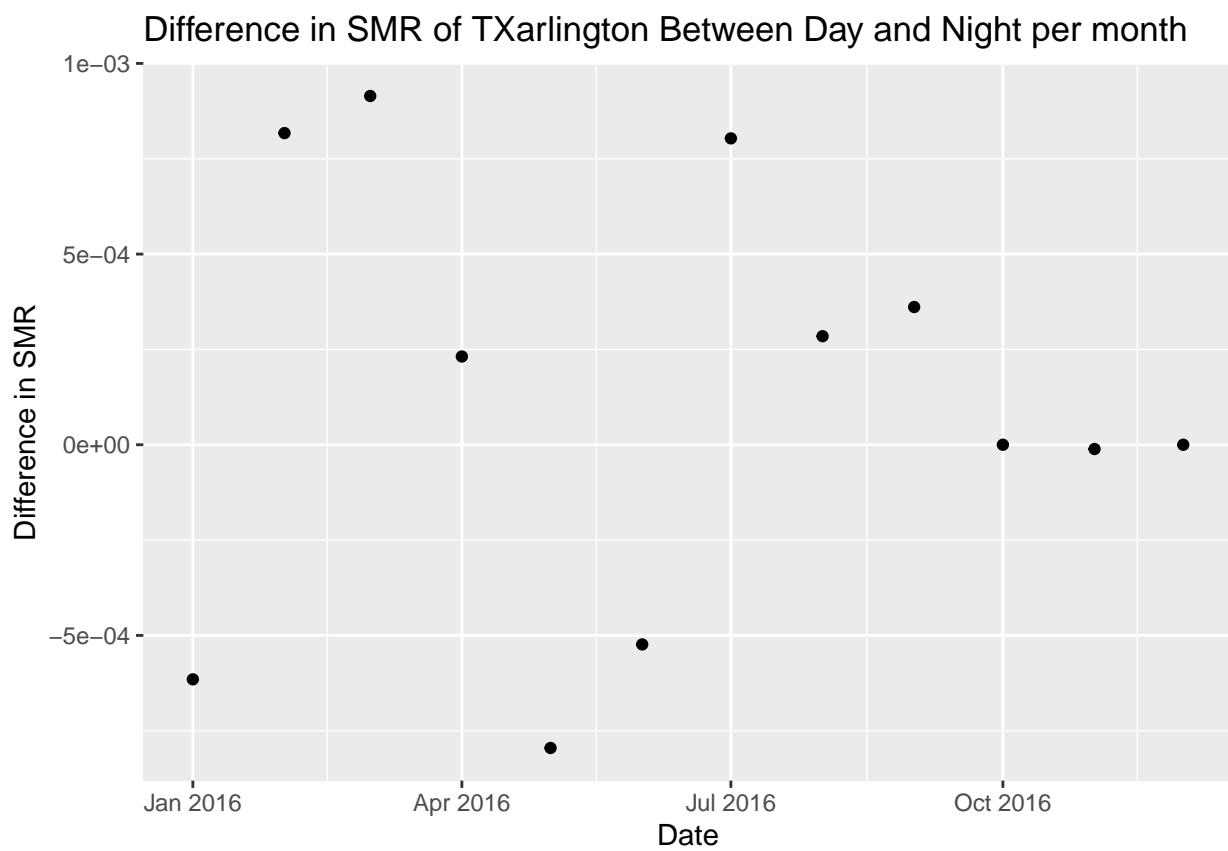


```
##  
## [48]
```

Difference in SMR of TNstate Between Day and Night per month

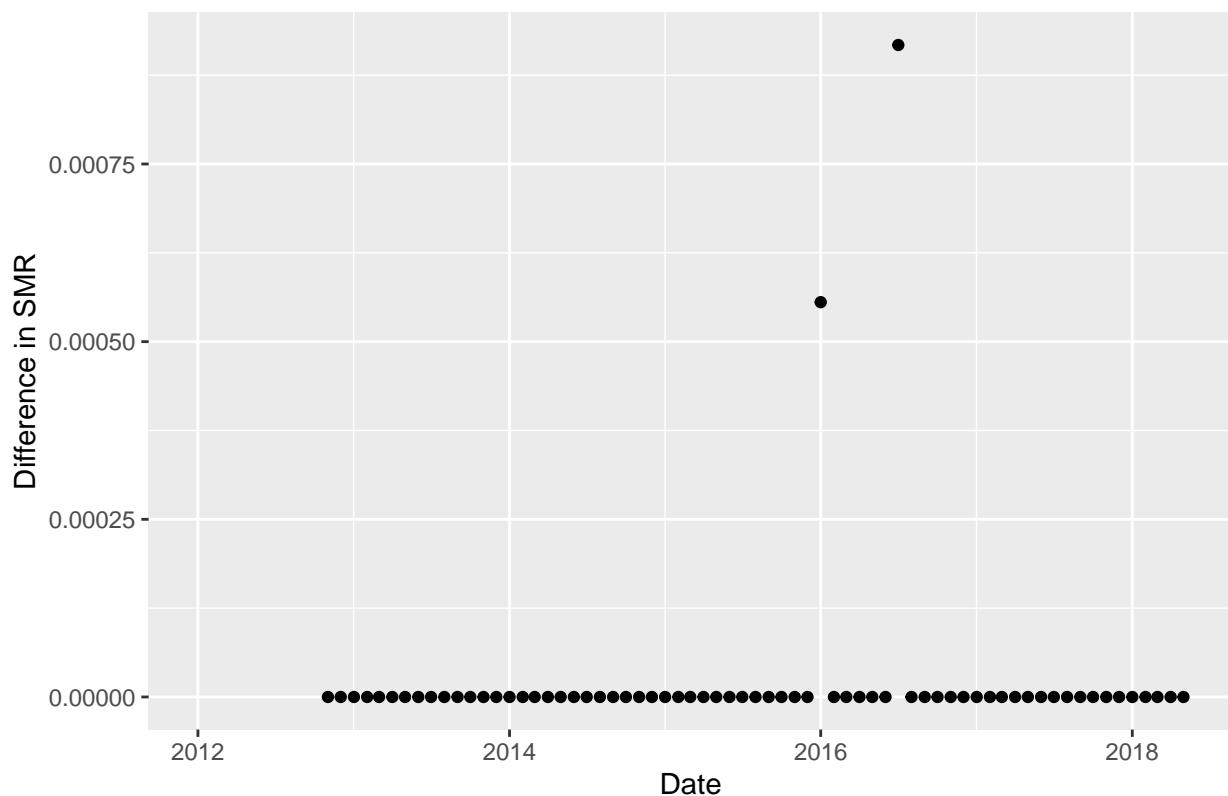


```
##  
## [[49]]
```



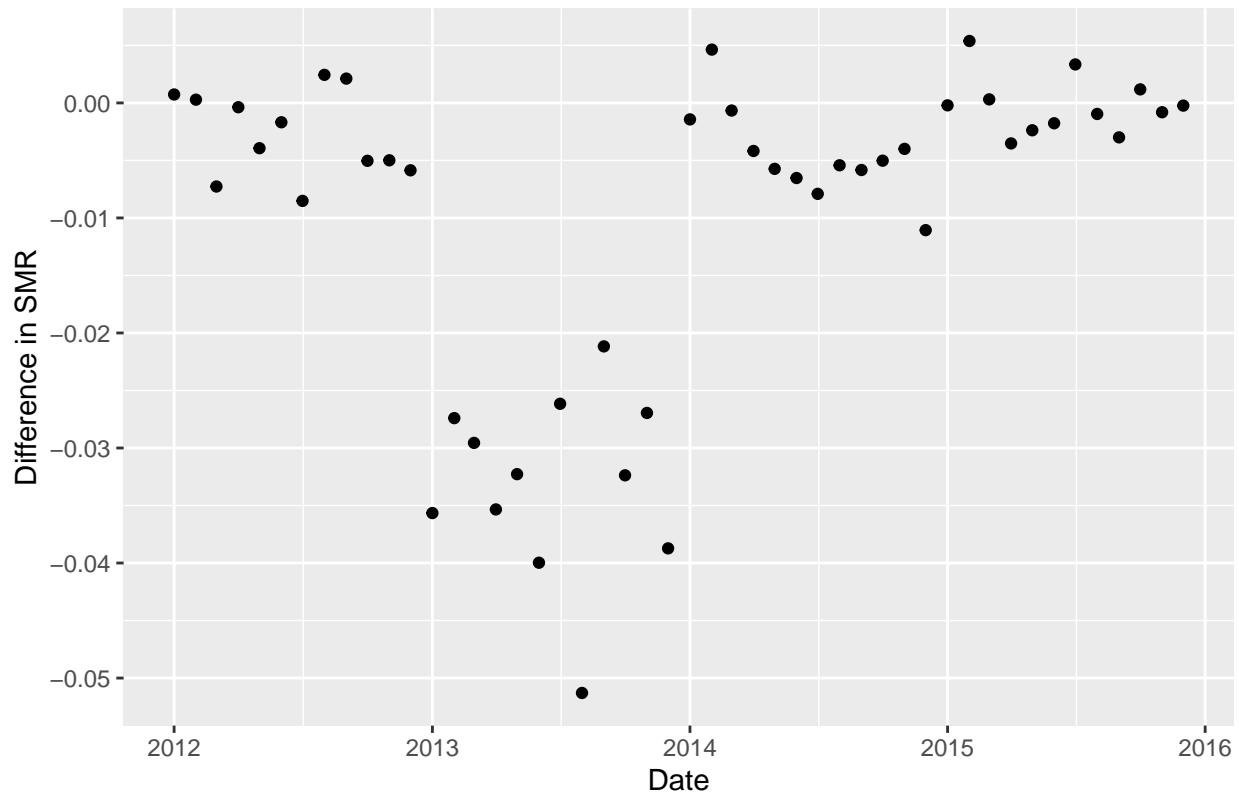
```
##  
## [[50]]
```

Difference in SMR of TXgarland Between Day and Night per month



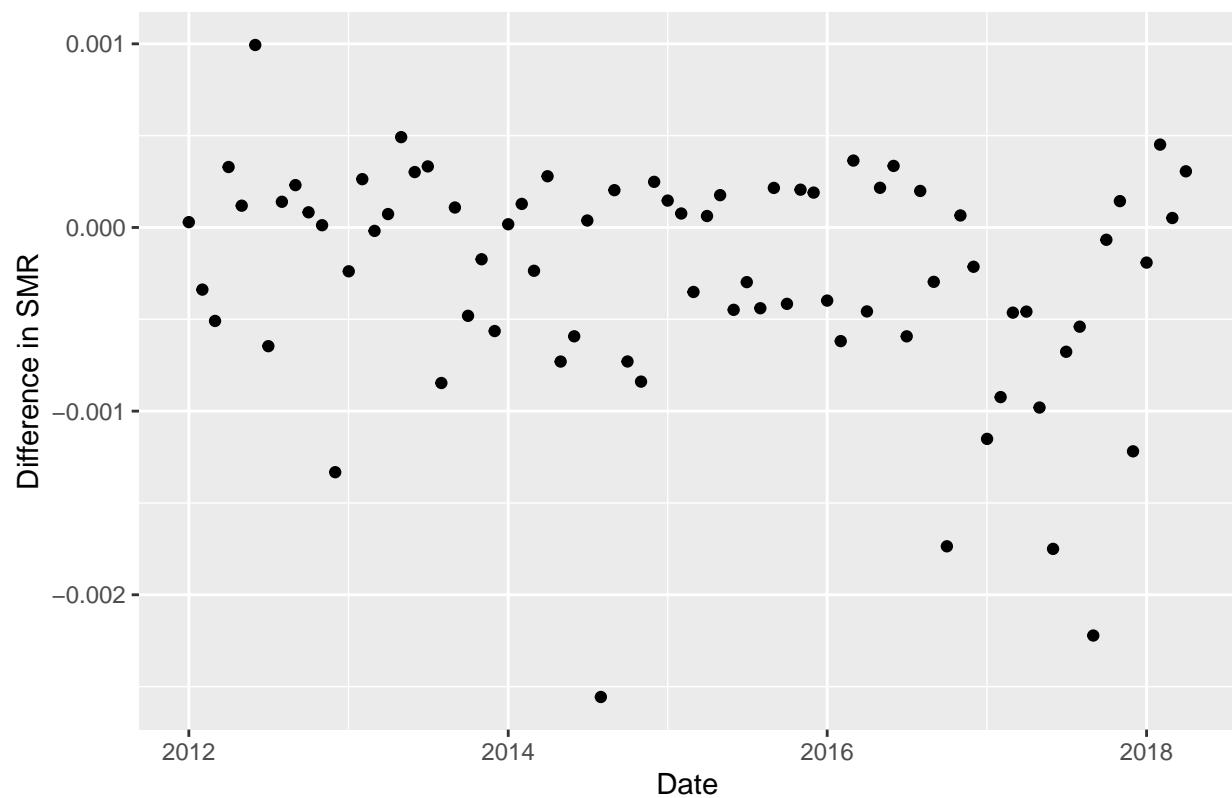
```
##  
## [[51]]
```

Difference in SMR of TXplano Between Day and Night per month



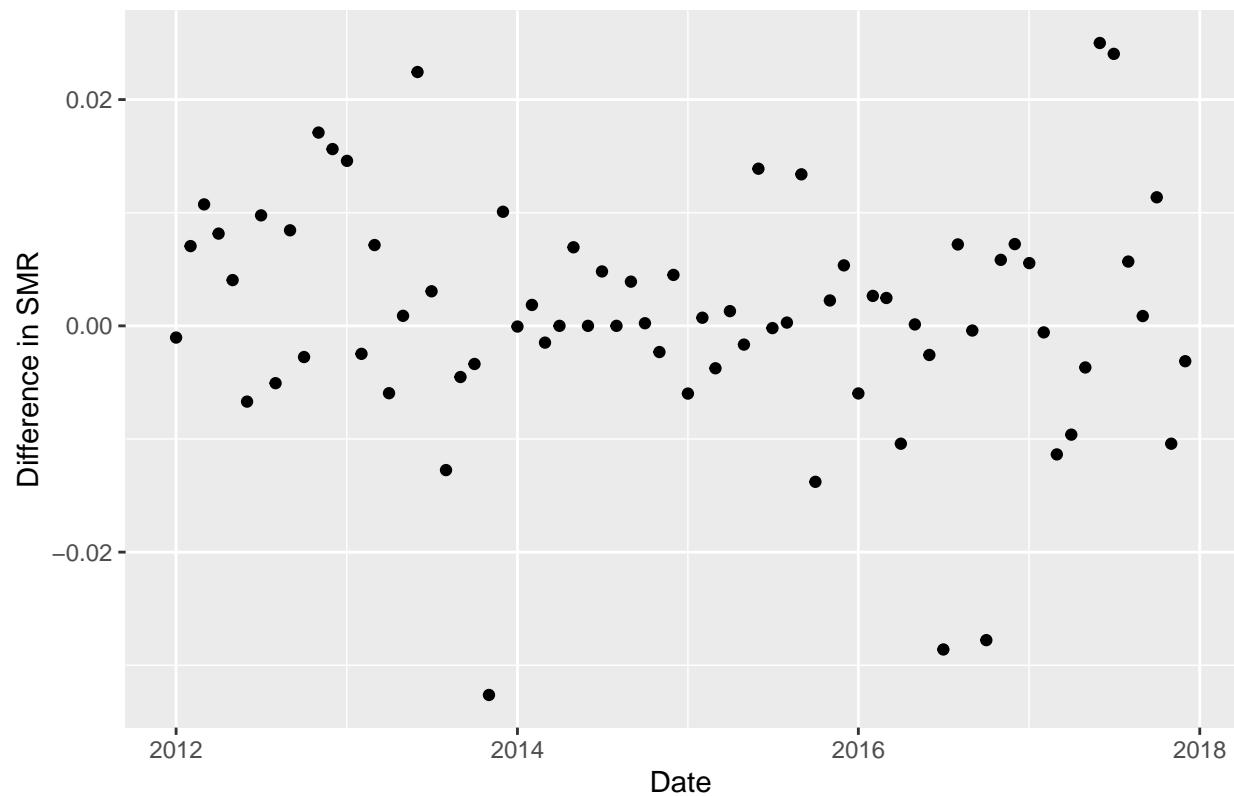
```
##  
## [[52]]
```

Difference in SMR of TXsanantonio Between Day and Night per month



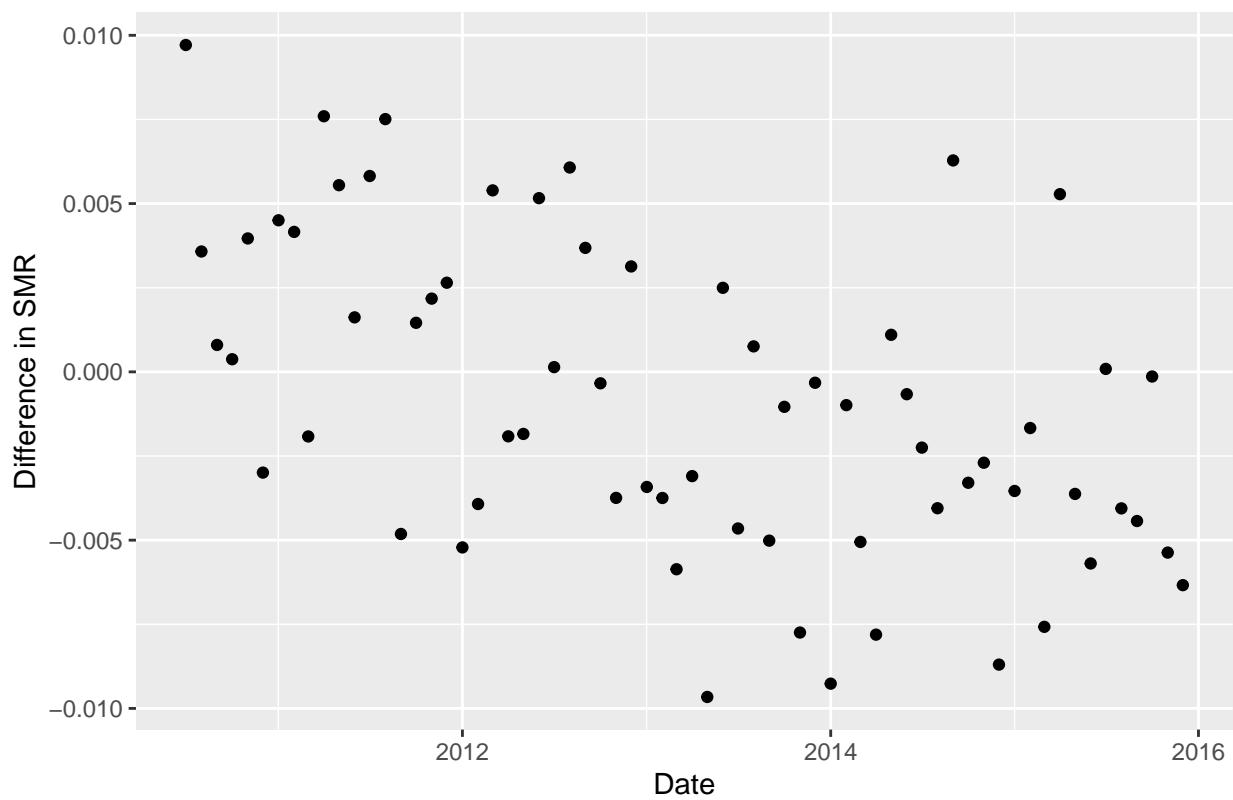
```
##  
## [[53]]
```

Difference in SMR of VTburlington2020 Between Day and Night per month



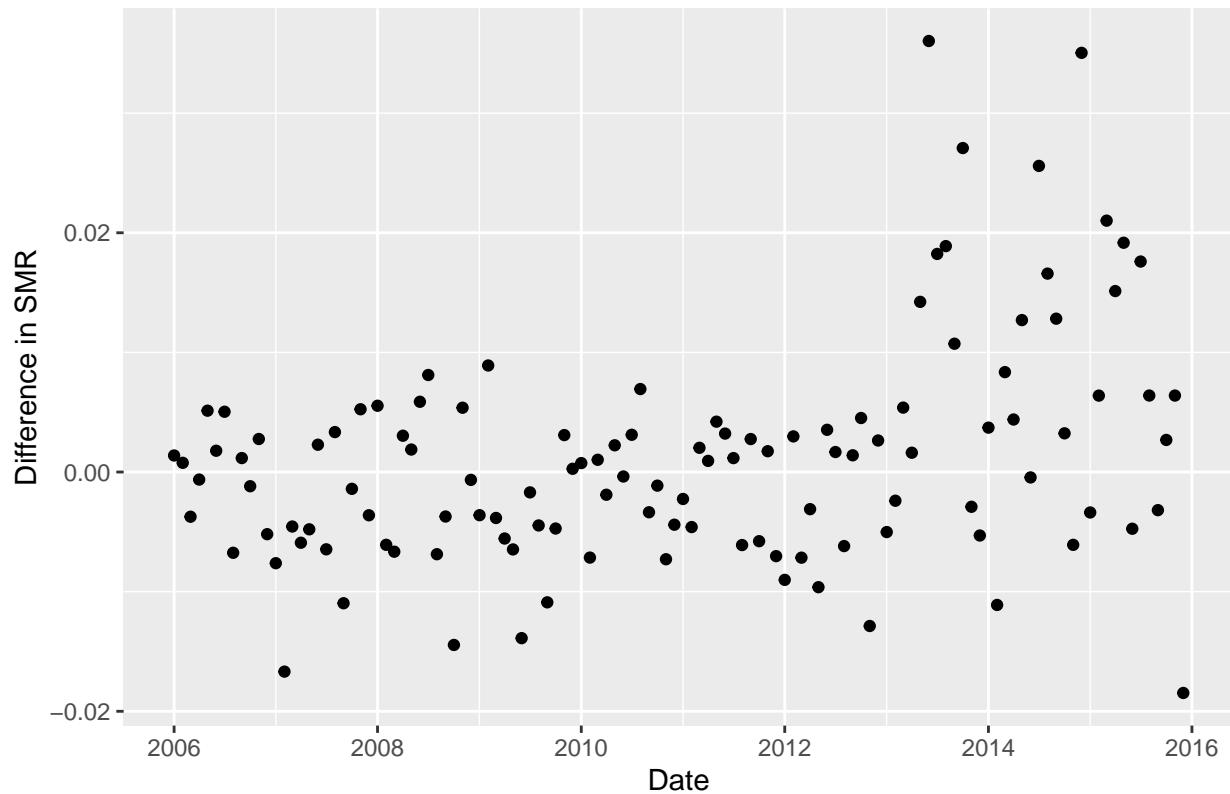
```
##  
## [[54]]
```

Difference in SMR of VT statewide 2020 Between Day and Night per month



```
##  
## [[55]]
```

Difference in SMR of WAseattle Between Day and Night per month



Toy log reg

```

outcome = race*age + gender + 4-hour period of the day + day of the week + year ???
sc_lst <- lapply(sunriseset_lst, myfilter_for, c("subject_race", "dataset_name", "time", "date", "search_conducted"))

sc_lst <- sc_lst[sapply(sc_lst, function(x) isTRUE(nrow(x) > 0))]

sc_lst <- lapply(sc_lst, outcome_clean, "search_conducted")

sapply(sc_lst, function(x) x$dataset_name[1])

am_lst <- lapply(sunriseset_lst, myfilter_for, c("subject_race", "dataset_name", "time", "date", "arrest_made"))

am_lst <- am_lst[sapply(am_lst, function(x) isTRUE(nrow(x) > 0))]

am_lst <- lapply(am_lst, outcome_clean, "arrest_made")

sapply(am_lst, function(x) x$dataset_name[1])

# pre 2014-02-10 0% to 1%
# lat/lng

TNash <- mysearch_dataset(sunriseset_lst, "TNashville") %>%
  mutate(day_week = as.factor(wday(ymd(date))),
        hour = lubridate::hour(hms(time)),
        month = lubridate::month(ymd(date)))
  
```

```
hour = case_when(hour <= 4 ~ 1,
                  hour <= 8 & hour > 4 ~ 2,
                  hour <= 12 & hour > 8 ~ 3,
                  hour <= 16 & hour > 12 ~ 4,
                  hour <= 20 & hour > 16 ~ 5,
                  hour > 20 ~ 6),
hour = as.factor(hour),
subject_age = as.numeric(subject_age),
subject_race = as.factor(subject_race))

TNnash <- outcome_clean(TNnash, "arrest_made")

TNpre <- TNnash %>%
  filter(date < ymd("2014-02-10"))

TNpost <- TNnash %>%
  filter(date > ymd("2014-02-10"))
```

```
fit_TNpre <- glm(arrest_made ~ subject_race + subject_age + subject_sex + day_week + hour, data = TNpre
tidy(fit_TNpre) %>%
  kbl(booktabs = T) %>%
  kableExtra::landscape()
```

```
fit_TNpost <- glm(arrest_made ~ subject_race +subject_age + subject_sex + day_week + hour, data = TNpost)
tidy(fit_TNpost) %>%
  tbl(booktabs = T) %>%
  kableExtra::landscape()
```

```

fit_TN <- glm(arrest_made ~ subject_race +subject_age + subject_sex + day_week + hour, data = TNnash, f
tidy(fit_TN) %>%
  kbl(booktabs = T) %>%
  kableExtra::landscape()

NCraleigh <- mysearch_dataset(sunriseset_lst, "NCraleigh") %>%
  mutate(day_week = as.factor(wday(ymd(date))),
        hour = lubridate::hour(hms(time)),
        hour = case_when(hour <= 4 ~ 1,
                         hour <= 8 & hour > 4 ~ 2,
                         hour <= 12 & hour > 8 ~ 3,
                         hour <= 16 & hour > 12 ~ 4,
                         hour <= 20 & hour > 16 ~ 5,
                         hour > 20 ~ 6),
        hour = as.factor(hour),
        subject_age = as.numeric(subject_age),
        subject_race = as.factor(subject_race))

NCraleigh <- outcome_clean(NCraleigh, "arrest_made")
NCraleigh <- outcome_clean(NCraleigh, "search_conducted")

NCday <- NCraleigh %>% filter(day_night == "day")

NCnight <- NCraleigh %>% filter(day_night == "night")

```

```
fit_NCday <- glm(arrest_made ~ subject_race +subject_age + subject_sex + day_week, data = NCday, family = binomial)
tidy(fit_NCday) %>%
  kbl(booktabs = T) %>%
  kableExtra::landscape()
```

```
fit_NCnight <- glm(arrest_made ~ subject_race + subject_age + subject_sex + day_week, data = NCnight, f  
tidy(fit_NCnight) %>%  
  kbl(booktabs = T) %>%  
  kableExtra::landscape()
```

```
fit_NC <- glm(arrest_made ~ subject_race + subject_age + subject_sex + day_week, data = NCraleigh, family = binomial)
tidy(fit_NC) %>%
  kbl(booktabs = T) %>%
  kableExtra::landscape()
```

```
now for search_conducted
```

```
fit_NCday <- glm(search_conducted ~ subject_race + subject_age + subject_sex + day_week, data = NCday, family = binomial)
tidy(fit_NCday) %>%
  kbl(booktabs = T) %>%
  kableExtra::landscape()
```

```
fit_NCnight <- glm(search_conducted ~ subject_race + subject_age + subject_sex + day_week, data = NCnight)
tidy(fit_NCnight) %>%
  tbl(booktabs = T) %>%
    kableExtra::landscape()
```

```

fit_NC <- glm(search_conducted ~ subject_race + subject_age + subject_sex + day_week, data = NCraleigh,
tidy(fit_NC) %>%
  kbl(booktabs = T) %>%
  kableExtra::landscape()

WAseattle <- mysearch_dataset(sunriseset_lst, "WAseattle") %>%
  mutate(day_week = as.factor(wday(ymd(date))),
        hour = lubridate::hour(hms(time)),
        hour = case_when(hour <= 4 ~ 1,
                          hour <= 8 & hour > 4 ~ 2,
                          hour <= 12 & hour > 8 ~ 3,
                          hour <= 16 & hour > 12 ~ 4,
                          hour <= 20 & hour > 16 ~ 5,
                          hour > 20 ~ 6),
        hour = as.factor(hour),
        subject_age = as.numeric(subject_age),
        subject_race = as.factor(subject_race))

WAseattle %>% group_by(arrest_made, subject_race) %>% summarize(count = n())

```

scraps

```

#year
SDstatewide <- mysearch_dataset(date_lst, "SDstatewide") # 2015-08-31 14% to ~ 0%
  # location is different
NHstatewide <- mysearch_dataset(date_lst, "NHstatewide") # 2014-12-08 15% to 7%
  # subject_age
LAneworleans <- mysearch_dataset(date_lst, "LAneworleans") # 2011-05-09 30% to ~10%
  # location, lat, lng, outcome
FLstatewide <- mysearch_dataset(date_lst, "FLstatewide") # pre 2015-06-15, post 2016-10-31, 10% to 40%
  # location, subject_sex, subject_age
CAoakland <- mysearch_dataset(date_lst, "CAoakland") # pre 2016-12-26      10% to 1%
  # subject_age

#daynight
OHcolumbus <- mysearch_dataset(sunriseset_lst, "OHcolumbus")
KYlouisville <- mysearch_dataset(sunriseset_lst, "KYlouisville")

# finding the differences here:
CAoakland %>% filter(str_detect(date, "2017"))

date_df %>% filter(dataset_name == "CAoakland")

# 2014-12-08 15% to 7%

NHpre2014 <- NHstatewide %>%
  mutate(date = lubridate::ymd(date)) %>%
  filter(date < ymd("2014-12-08"))

NHpost2014 <- NHstatewide %>%
  mutate(date = lubridate::ymd(date)) %>%

```

```

filter(date > ymd("2014-12-08"))

fit_NHpre2014 <- glm(citation_issued ~ as.numeric(subject_age)*as.factor(subject_race) + subject_sex, data = NH)
summary(fit_NHpre2014)

fit_NHpost2014 <- glm(citation_issued ~ as.numeric(subject_age)*as.factor(subject_race) + subject_sex, data = NH)
summary(fit_NHpost2014)

fit_NH <- glm(citation_issued ~ as.numeric(subject_age)*as.factor(subject_race) + subject_sex, data = NH)
summary(fit_NH)

LANeworleans <- DBI::dbGetQuery(con, "SELECT * FROM LANeworleans WHERE type = 'vehicular'")

# 2011-05-09 30% to ~10%
# location, lat, lng, outcome

LApre <- LANeworleans %>%
  mutate(date = lubridate::ymd(date)) %>%
  filter(date < ymd("2011-05-09")) %>%
  select(subject_age, subject_race, subject_sex, search_conducted, officer_assignment, district, zone)

LApost <- LANeworleans %>%
  mutate(date = lubridate::ymd(date)) %>%
  filter(date > ymd("2011-05-09")) %>%
  select(subject_age, subject_race, subject_sex, search_conducted, officer_assignment, district, zone)

fit_LApre2011 <- glm(search_conducted ~ as.numeric(subject_age)*as.factor(subject_race) + subject_sex + officer_assignment, data = LApre)
summary(fit_LApre2011)

fit_LApost2011 <- glm(search_conducted ~ as.numeric(subject_age)*as.factor(subject_race) + subject_sex + officer_assignment, data = LApost)
summary(fit_LApost2011)

fit_LA <- glm(search_conducted ~ as.numeric(subject_age)*as.factor(subject_race) + subject_sex + officer_assignment, data = LANeworleans)
summary(fit_LA)

# pre 2015-06-15, post 2016-10-31, 10% to 40%
# location, subject_sex, subject_age

FLpre <- FLstatewide %>%
  mutate(date = lubridate::ymd(date)) %>%
  filter(date < ymd("2015-06-15"))

FLpost <- FLstatewide %>%
  mutate(date = lubridate::ymd(date)) %>%
  filter(date > ymd("2016-10-31"))
# post doesn't collect sex, age, ...
# so the pre and the aggregate models all have the same ... results?

fit_FLpre <- glm(search_conducted ~ as.numeric(subject_age)*as.factor(subject_race) + subject_sex, data = FLpre)
summary(fit_FLpre)

fit_FLpost <- glm(search_conducted ~ as.numeric(subject_age)*as.factor(subject_race), data = FLpost, family = binomial)
summary(fit_FLpost)

```

```

fit_FL <- glm(search_conducted ~ as.numeric(subject_age)*as.factor(subject_race) + subject_sex, data = CAoakland)
summary(fit_FL)

# pre 2016-12-26      10% to 1%
# subject_age

CApre <- CAoakland %>%
  mutate(date = lubridate::ymd(date)) %>%
  filter(date < ymd("2016-12-26"))

CApost <- CAoakland %>%
  mutate(date = lubridate::ymd(date)) %>%
  filter(date > ymd("2016-12-26"))

fit_CApre <- glm(search_conducted ~ as.factor(subject_race) + subject_sex, data = CApre, family = "binomial")
summary(fit_CApre)

fit_CApost <- glm(search_conducted ~ as.numeric(subject_age) + as.factor(subject_race) + subject_sex, data = CApost)
summary(fit_CApost)

fit_LA <- glm(search_conducted ~ as.numeric(subject_age) + as.factor(subject_race) + subject_sex, data = CApost)
summary(fit_LA)

OHday <- OHcolumbus %>% filter(day_night == "day")
OHnight <- OHcolumbus %>% filter(day_night == "night")

fit_OHday <- glm(search_conducted ~ as.factor(subject_race) + subject_sex, data = OHday, family = "binomial")
fit_OHnight <- glm(search_conducted ~ as.factor(subject_race) + subject_sex, data = OHnight, family = "binomial")
fit_OH <- glm(search_conducted ~ as.factor(subject_race) + subject_sex, data = OHcolumbus, family = "binomial")

summary(fit_OHday)
summary(fit_OHnight)
summary(fit_OH)

KYday <- KYlouisville %>% filter(day_night == "day")
KYnight <- KYlouisville %>% filter(day_night == "night")

fit_KYday <- glm(search_conducted ~ as.numeric(subject_age)*as.factor(subject_race) + subject_sex, data = KYlouisville)
summary(fit_KYday)

fit_KYnight <- glm(search_conducted ~ as.numeric(subject_age)*as.factor(subject_race) + subject_sex, data = KYlouisville)
summary(fit_KYnight)

fit_KY <- glm(search_conducted ~ as.numeric(subject_age)*as.factor(subject_race) + subject_sex, data = KYlouisville)
summary(fit_KY)

```