

Massachusetts_Case

Karl Grindal

6/8/2020

Case 1: Massachusetts Data Security Law

```
library(plyr)
library(here)

## here() starts at C:/Users/karl_000/Documents/SpiderOak Hive/Dissertation/Dissertation_Code
##
## Attaching package: 'here'

## The following object is masked from 'package:plyr':
##
##     here
library(tidyr)
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarise
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
library(lubridate)

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
library(tseries)

## Registered S3 method overwritten by 'quantmod':
##   method           from
```

```

##   as.zoo.data.frame zoo
library(TTR)

AllStateClean <- read.table(here::here("Data", "Other_data", "AllStateClean.txt"), sep=";")

AllStateClean$MA_affected_residents <- gsub("\\,.*","",AllStateClean$MA_affected_residents) # this sele
AllStateClean$NC_affected_residents <- gsub("\\,.*","",AllStateClean$NC_affected_residents) # this sele

AllStateClean$Massachusetts[!is.na(AllStateClean$Massachusetts)]<-1
AllStateClean$New_Hampshire[!is.na(AllStateClean$New_Hampshire)]<-1
AllStateClean$North_Carolina[!is.na(AllStateClean$North_Carolina)]<-1
AllStateClean$Massachusetts[is.na(AllStateClean$Massachusetts)]<-0
AllStateClean$New_Hampshire[is.na(AllStateClean$New_Hampshire)]<-0
AllStateClean$North_Carolina[is.na(AllStateClean$North_Carolina)]<-0

```

Identifying treatment and control options

```

# Case 1: Experiment 1
massachusetts <- dplyr::filter(AllStateClean, Massachusetts==1)
massachusetts$reported_date <- substr(massachusetts$reported_date, start=1, stop=10)

new_hampshire <- dplyr::filter(AllStateClean, New_Hampshire==1)
new_hampshire$reported_date <- substr(new_hampshire$reported_date, start=1, stop=10)

# Case 1: Experiment 2
massachusetts$MA_affected_residents <- as.numeric(massachusetts$MA_affected_residents)
massachusetts1000 <- subset(massachusetts, massachusetts$MA_affected_residents > 1000)

n_carolina <- dplyr::filter(AllStateClean, North_Carolina==1)
n_carolina$NC_affected_residents <- as.numeric(n_carolina$NC_affected_residents)
n_carolina1000 <- subset(n_carolina, n_carolina$NC_affected_residents > 1000)

```

Specifying Treatment and Control Variables

```

Treat_Name <- 'Massachusetts'
Ctrl_Name <- 'New Hampshire'

treatment <- massachusetts # This Must Be Filled in to Work Properly!
control <- new_hampshire # This Must Be Filled in to Work Properly!

```

Specifying Time Variables

```

first_breach_date <- as.Date("2006-06-01")
data_start <- c(2006,6)
data_range <- c(2006,2020)
exp_start <- c(2008,4)
exp_range <- c(2008,2011)

```

```

first_pop_date <- as.Date("2000-04-01")
no_months_out <- 174

months_prior <- 5 # months before treatment start
months_after <- 5 # months after treatment start

# Legislation H.B. 4144 signed into law August 3, 2007
# Legislation H.B. 4144 becomes effective on October 31, 2007
# OCABR finalized the regulation on September 22, 2008

treatment_start <- as.Date("09/22/2008", "%m/%d/%Y") # Legislation H.B. 4144 becomes effective
treatment_end <- as.Date("03/01/2010", "%m/%d/%Y") # post 6 months after enforcement

```

Produce Tables on Overlapping Co-occurrence of Breach Incidents

```

CoveredDays <- AllStateClean

CoveredDays$reported_date <- substr(CoveredDays$reported_date, start=1, stop=10)

CoveredDays <- subset(CoveredDays, reported_date > as.Date("2008-03-22") )
CoveredDays <- subset(CoveredDays, reported_date < as.Date("2010-09-01") )

table(CoveredDays$Massachusetts) # 0 = 313, 1 = 858

##
##    0    1
## 313 858
table(CoveredDays$New_Hampshire) # 0 = 949, 1 = 222

##
##    0    1
## 949 222
table(CoveredDays$North_Carolina) # 0 = 819, 1 = 352

##
##    0    1
## 819 352
table(CoveredDays$Massachusetts,CoveredDays$New_Hampshire) # 0 = 205 + 108 ; 1 = 744 + 114

##
##          0    1
##    0 205 108
##    1 744 114
table(CoveredDays$Massachusetts,CoveredDays$North_Carolina) # 0 = 99 + 214 ; 1 = 720 + 138

##
##          0    1
##    0 99 214
##    1 720 138

```

```

CoveredDays$MA_affected_residents <- as.numeric(CoveredDays$MA_affected_residents)
CoveredDays$NC_affected_residents <- as.numeric(CoveredDays$NC_affected_residents)

CoveredDays$MA_BigBreach <- CoveredDays$MA_affected_residents > 1000
CoveredDays$NC_BigBreach <- CoveredDays$NC_affected_residents > 1000

# Need to resolve this R issue to get to a fixed solution

# table(CoveredDays$MA_BigBreach,CoveredDays$NC_BigBreach)

```

Create Population Time Series for Matching with Incident Frequency

```

# Population
pop <- read.csv(here::here("Data","Other_data","populations.csv")) # starts at 2000.04.01

datforpop <- data.frame(seq(first_pop_date, by="1 month", length.out=(length(pop)-1)))
names(datforpop) <- "yearmonth"
datforpop <- format(datforpop, "%Y/%m")
datforpop <- rbind("yearmonth",datforpop)
row.names(datforpop) <- 1:nrow(datforpop)

pop <- cbind(datforpop,t(pop))
colnames(pop) <- pop[1,]
pop <- pop[-1,]
rownames(pop) <- seq(1:nrow(pop))
pop <- as.data.frame(pop)

```

Experiment 1: Create control and treatment populations with Total Incidents

```

# Format treatment dates into months
treatment$date_formatted <- format(as.Date(treatment$reported_date, "%Y-%m-%d"), "%Y/%m") # Alternative
treatment_freq <- treatment %>%
  dplyr::group_by(treatment$date_formatted) %>%
  dplyr::summarise(frequency = n(),)

## `summarise()` ungrouping output (override with ` `.groups` argument)
names(treatment_freq)[1] <- "yearmonth"
treatment_freq$frequency[is.na(treatment_freq$frequency)]<-0

# Format control dates into months
control$date_formatted <- format(as.Date(control$reported_date, "%Y-%m-%d"), "%Y/%m") # Alternative is
control_freq <- control %>%
  dplyr::group_by(control$date_formatted) %>%
  dplyr::summarise(frequency = n(),)

## `summarise()` ungrouping output (override with ` `.groups` argument)

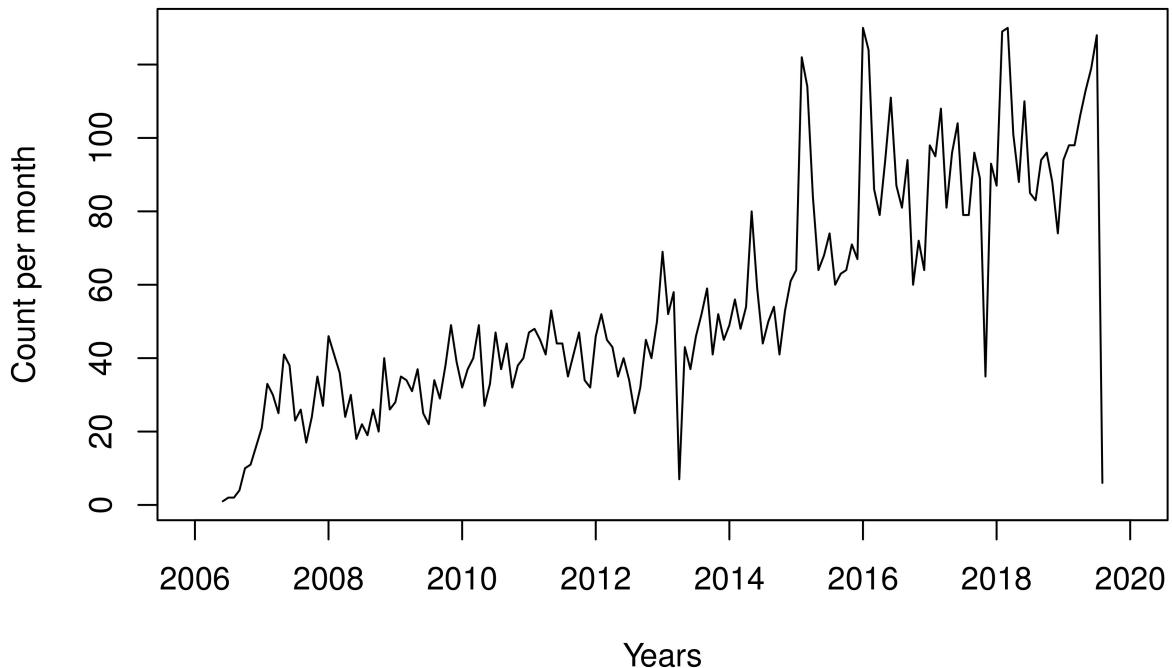
```

```
names(control_freq)[1] <- "yearmonth"
control_freq$frequency[is.na(control_freq$frequency)]<-0

treatment_ts <- ts(treatment_freq$frequency, frequency = 12, start = data_start)
control_ts <- ts(control_freq$frequency, frequency = 12, start = data_start)

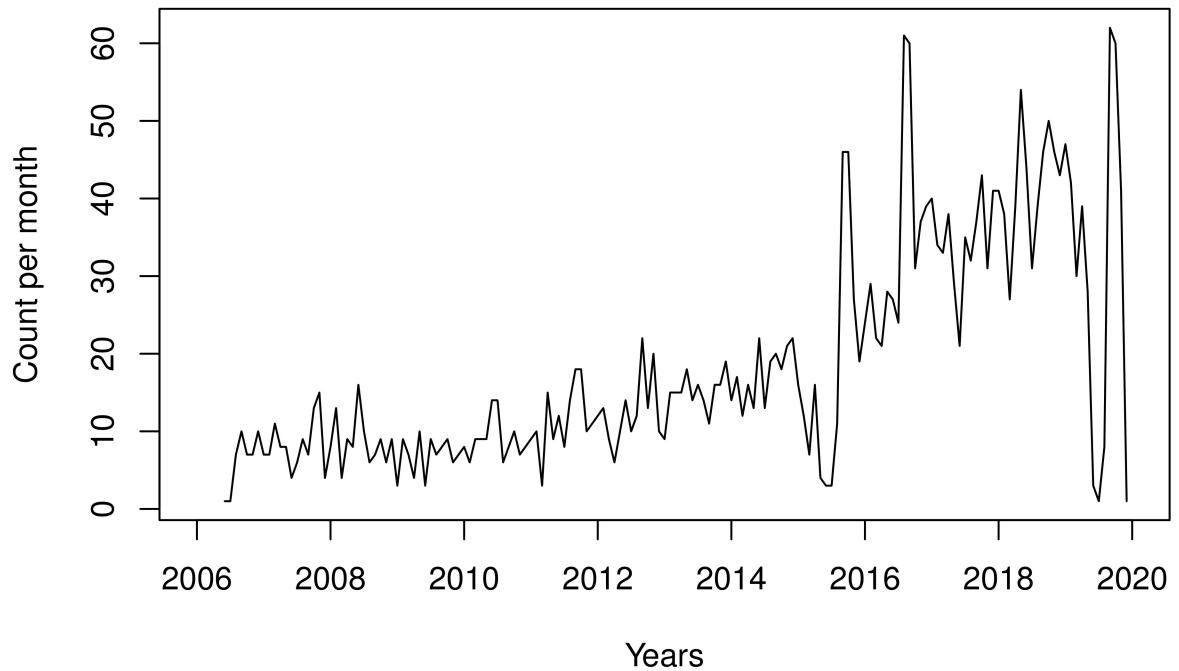
plot.ts(treatment_ts, main = "Breaches over time", xlim=data_range, xlab = "Years", ylab = "Count per month")
```

Breaches over time



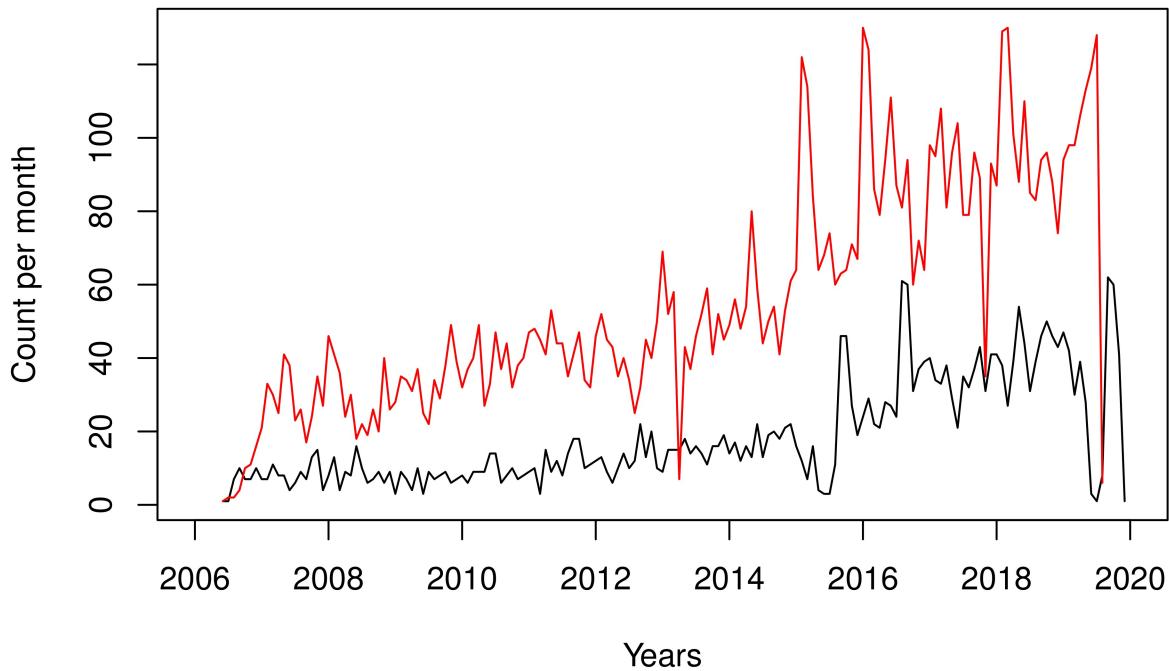
```
plot.ts(control_ts, main = "Breaches over time", xlim=data_range, xlab = "Years", ylab = "Count per month")
```

Breaches over time



```
ts.plot(control_ts, treatment_ts, main = "Breaches over time", xlim=data_range, gpars = list(col = c("b
```

Breaches over time



```
summary(treatment_ts)

##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##     1.00   34.00  46.00    55.19  79.00  130.00

summary(control_ts)

##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##     1.00   8.00  13.00   18.39  27.00  62.00
```

Create charts with breaches per million residents

```
# Merge Treatment and Control Together
comb_ts <- merge(treatment_freq, control_freq, by="yearmonth", all=TRUE)

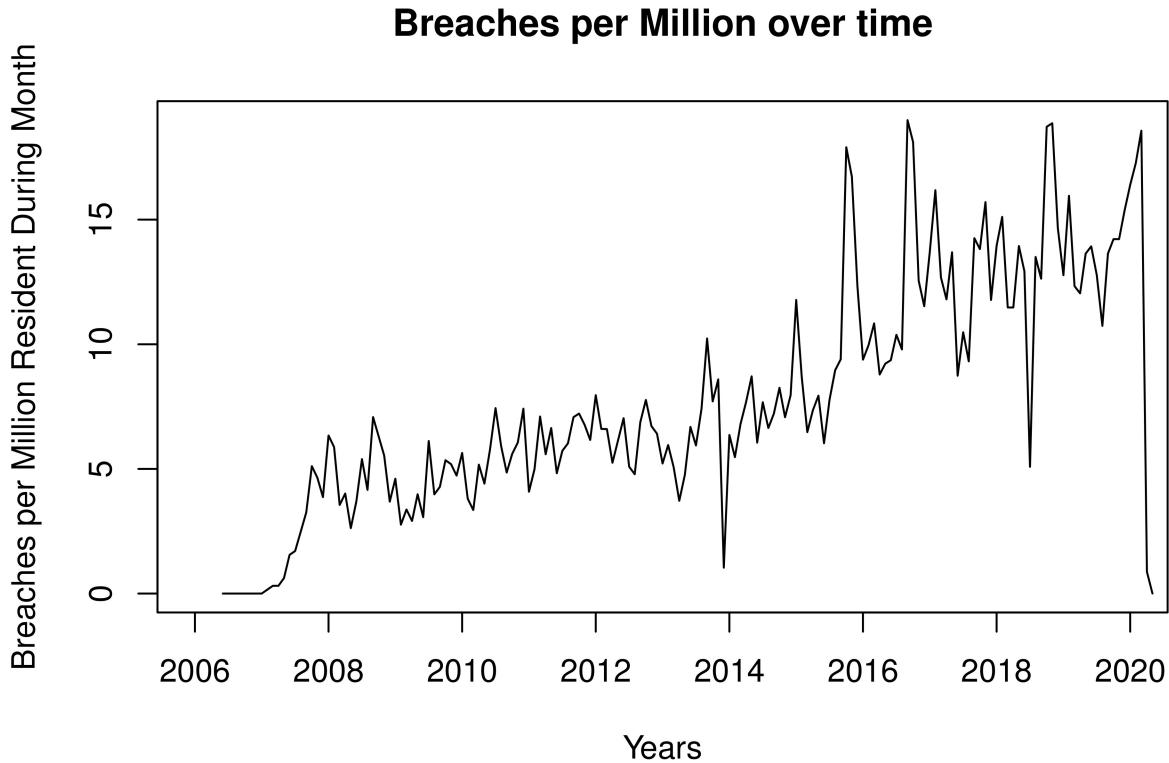
# Merge Combined Treatment and Control Together with Population Statistics
comb_ts <- merge(comb_ts, pop, by='yearmonth', all.x = TRUE)
comb_ts$frequency.x[is.na(comb_ts$frequency.x)]<-0
comb_ts$frequency.y[is.na(comb_ts$frequency.y)]<-0

# change class of columns to numeric
comb_ts[2:ncol(comb_ts)] <- sapply(comb_ts[2:ncol(comb_ts)],as.numeric)

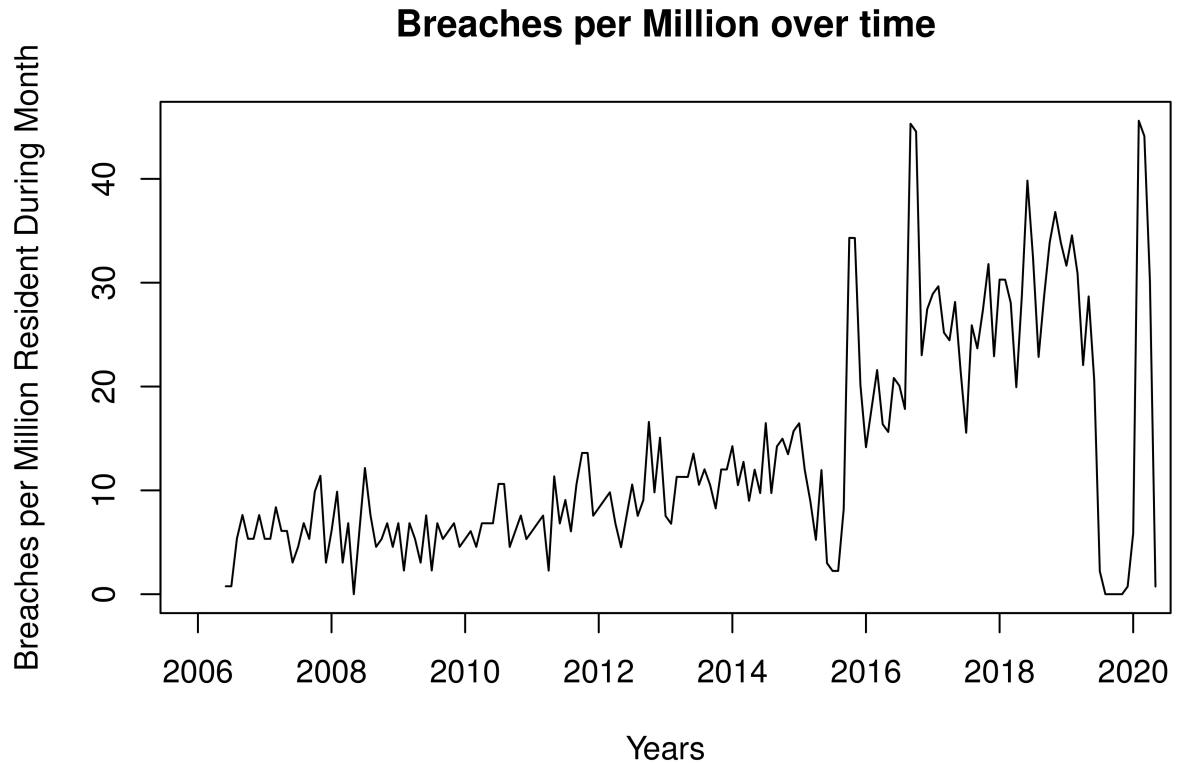
comb_ts$treatpermil <- comb_ts$frequency.x/(comb_ts[,c(Treat_Name)]/1000000)
comb_ts$controlpermil <- comb_ts$frequency.y/(comb_ts[,c(Ctrl_Name)]/1000000)
```

```
treatment_tsM <- ts(comb_ts$treatpermil, frequency = 12, start = data_start)
control_tsM <- ts(comb_ts$controlpermil, frequency = 12, start = data_start)

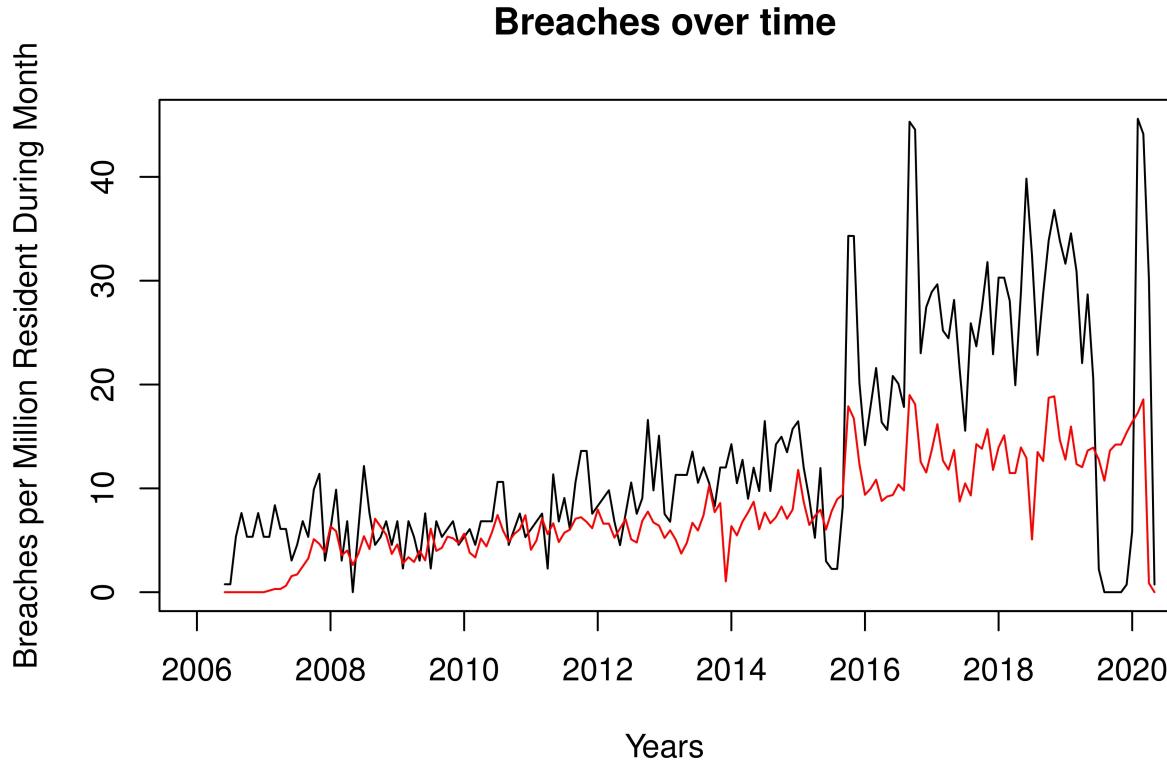
plot.ts(treatment_tsM, main = "Breaches per Million over time", xlim=data_range, xlab = "Years", ylab = "Breaches per Million Resident During Month")
```



```
plot.ts(control_tsM, main = "Breaches per Million over time", xlim=data_range, xlab = "Years", ylab = "Breaches per Million Resident During Month")
```



```
ts.plot(control_tsM, treatment_tsM, main = "Breaches over time", xlim=data_range, gpars = list(col = c(
```



Identifying and subsetting relevant dates

```
treatment_start<- format(as.Date(as.character(treatment_start), origin = "1970-01-01"), "%Y/%m")
treatment_end<- format(as.Date(as.character(treatment_end), origin = "1970-01-01"), "%Y/%m")

pretreat <- comb_ts[(which(comb_ts$yearmonth==treatment_start)-months_prior):which(comb_ts$yearmonth==treatment_end)]
pretreat$type <- "pretest"

posttreat <- comb_ts[(which(comb_ts$yearmonth==treatment_end):(which(comb_ts$yearmonth==treatment_end)+months_post))]
posttreat$type <- "posttest"

mean(posttreat$treatpermil) - mean(pretreat$treatpermil)

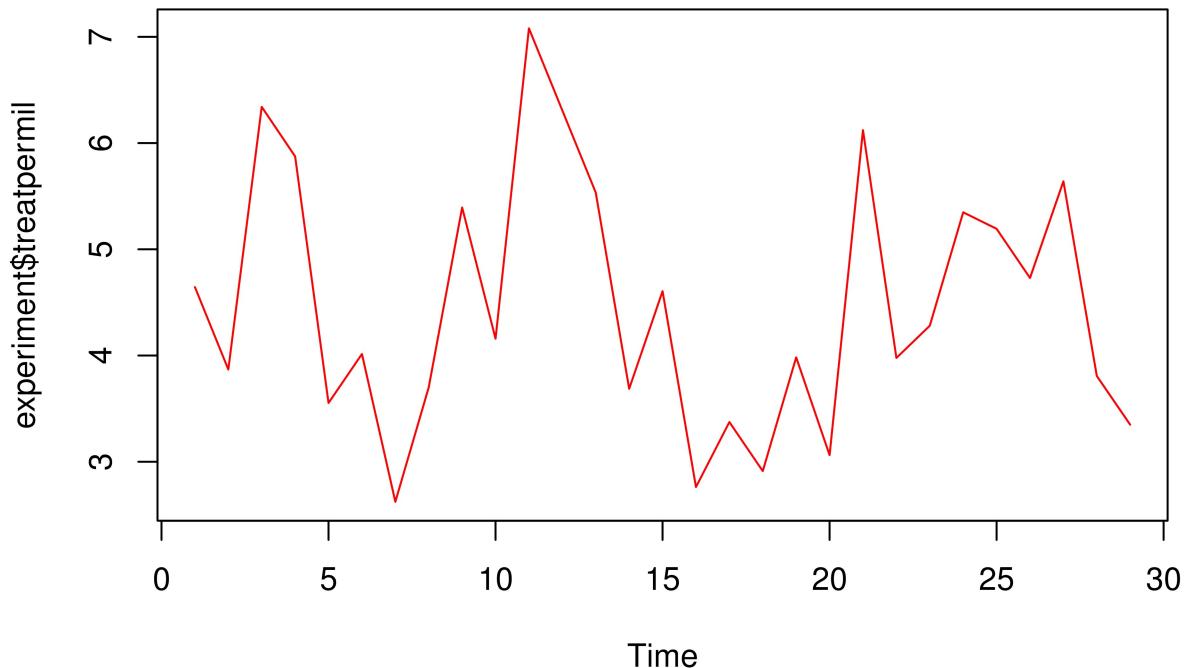
## [1] -0.03804966
mean(posttreat$controlpermil) - mean(pretreat$controlpermil)

## [1] -1.144614

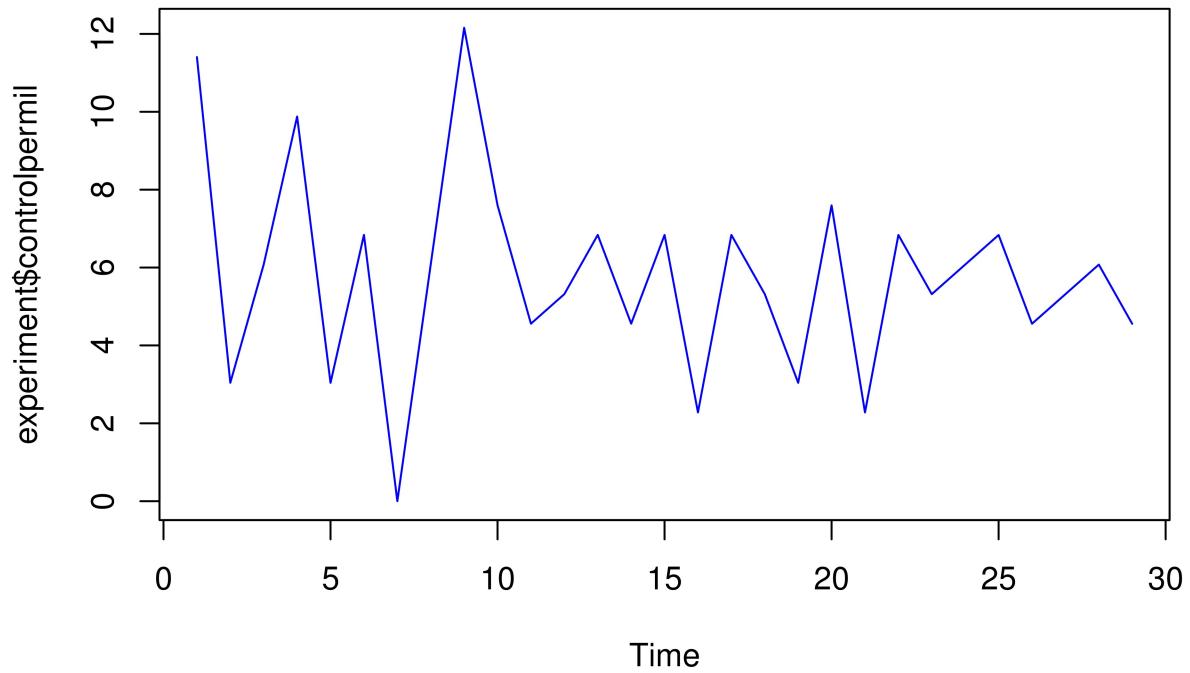
treatment_range <- comb_ts[(which(comb_ts$yearmonth==treatment_start)+1):(which(comb_ts$yearmonth==treatment_end))]
treatment_range$type <- "test"

experiment <- rbind(pretreat,treatment_range,posttreat)
experiment$treatpermil[is.na(experiment$treatpermil)]<-0
experiment$controlpermil[is.na(experiment$controlpermil)]<-0
```

```
ts.plot(experiment$treatpermil, col = "red")
```



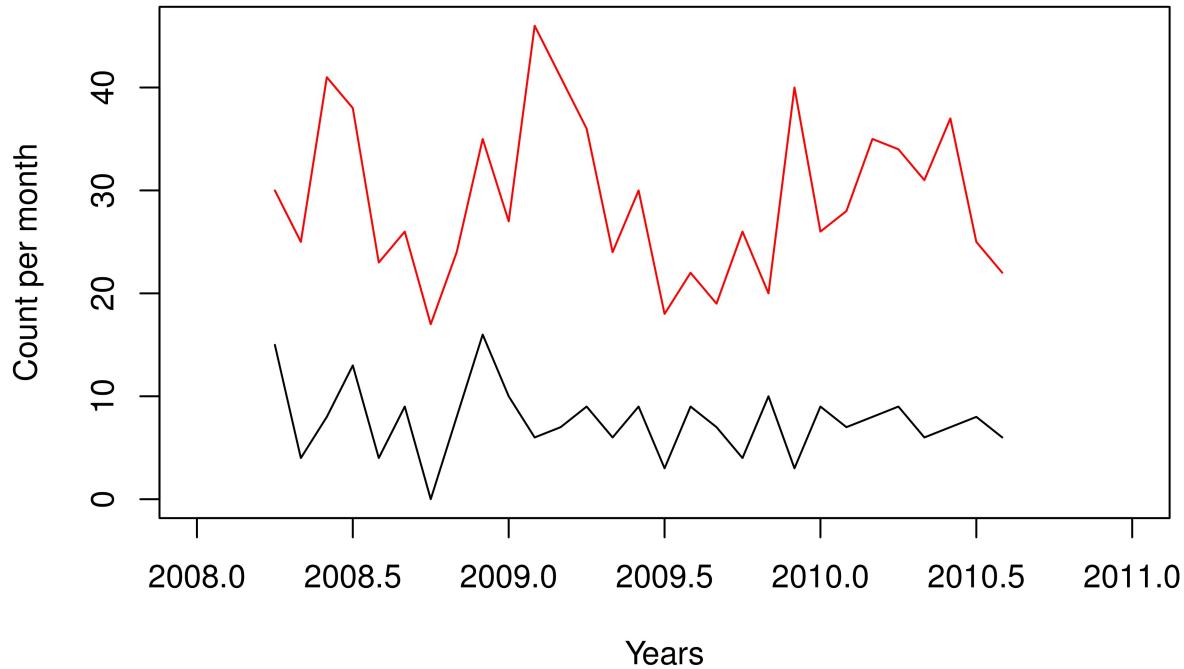
```
ts.plot(experiment$controlpermil, col = "blue")
```



```
# Look at Raw Frequency Counts
```

```
treatment_ts <- ts(experiment$frequency.x, frequency = 12, start = exp_start)
control_ts <- ts(experiment$frequency.y, frequency = 12, start = exp_start)
ts.plot(control_ts, treatment_ts, main = "Breaches over time", xlim=exp_range, gpars = list(col = c("blue", "red")))
```

Breaches over time



```
# Look at Treatment and Control per Million

treatment_tsM <- ts(experiment$treatpermil, frequency = 12, start = exp_start)
mean(treatment_tsM)

## [1] 4.480084
sd(treatment_tsM)

## [1] 1.19537

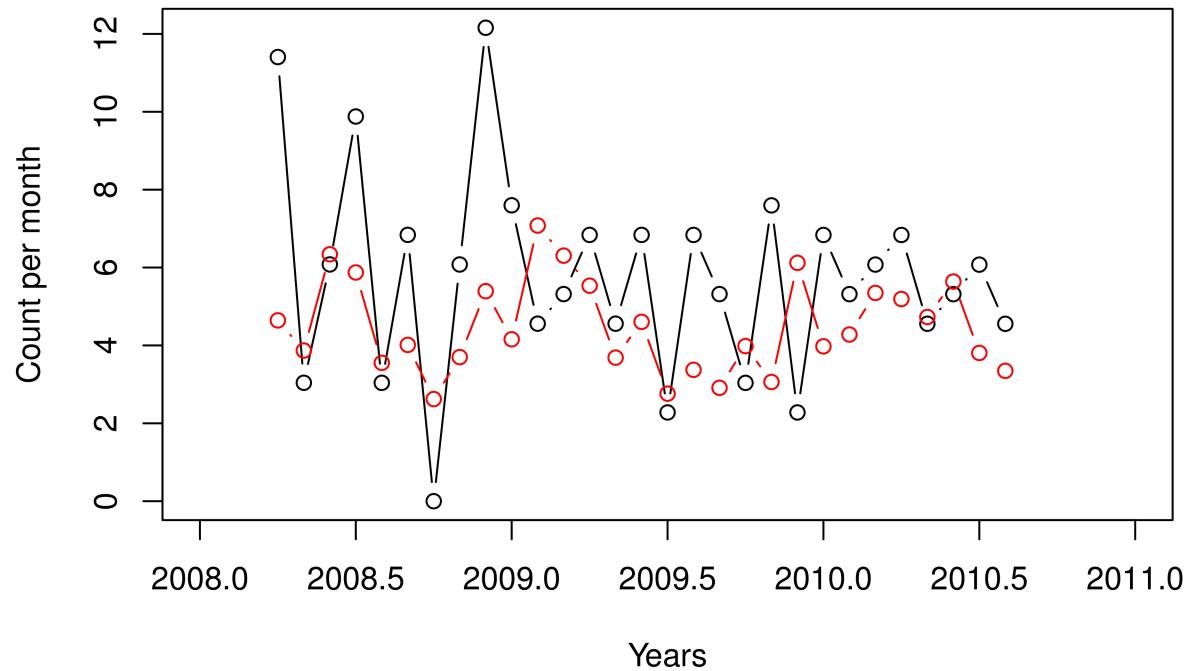
control_tsM <- ts(experiment$controlpermil, frequency = 12, start = exp_start)
mean(control_tsM)

## [1] 5.764173
sd(control_tsM)

## [1] 2.60553

ts.plot(control_tsM, treatment_tsM, main = "Breaches over time", xlim=exp_range,
        gpars = list(col = c("black", "red")), type = "b", xlab = "Years", ylab = "Count pe
```

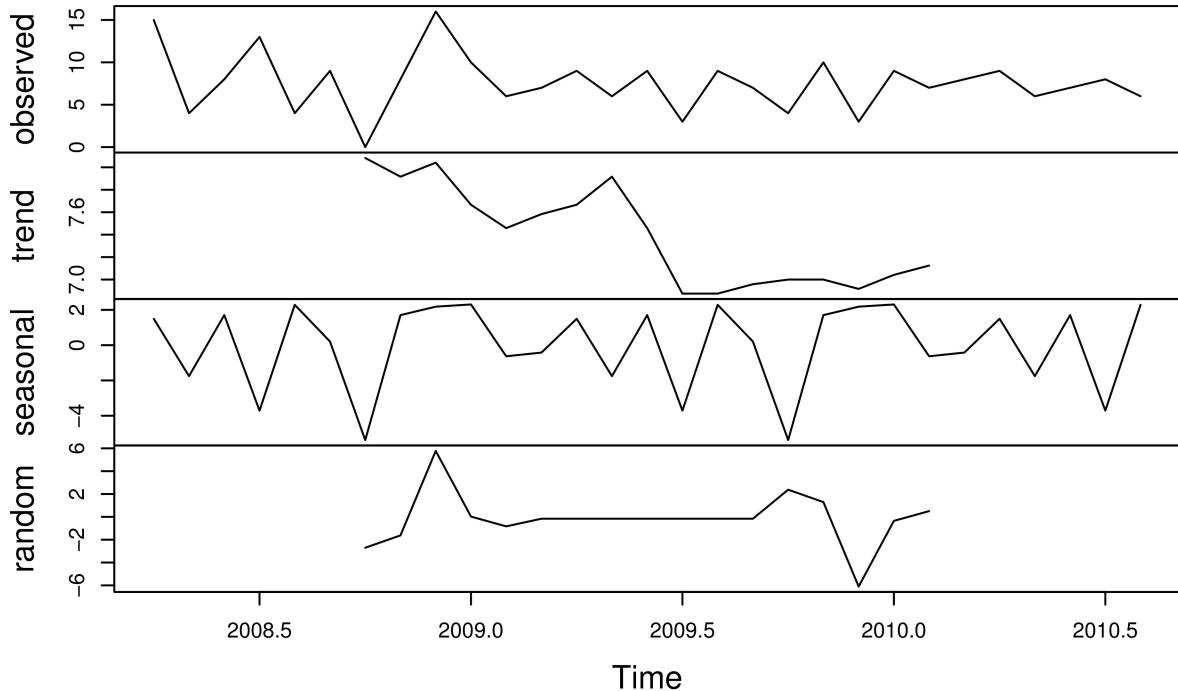
Breaches over time



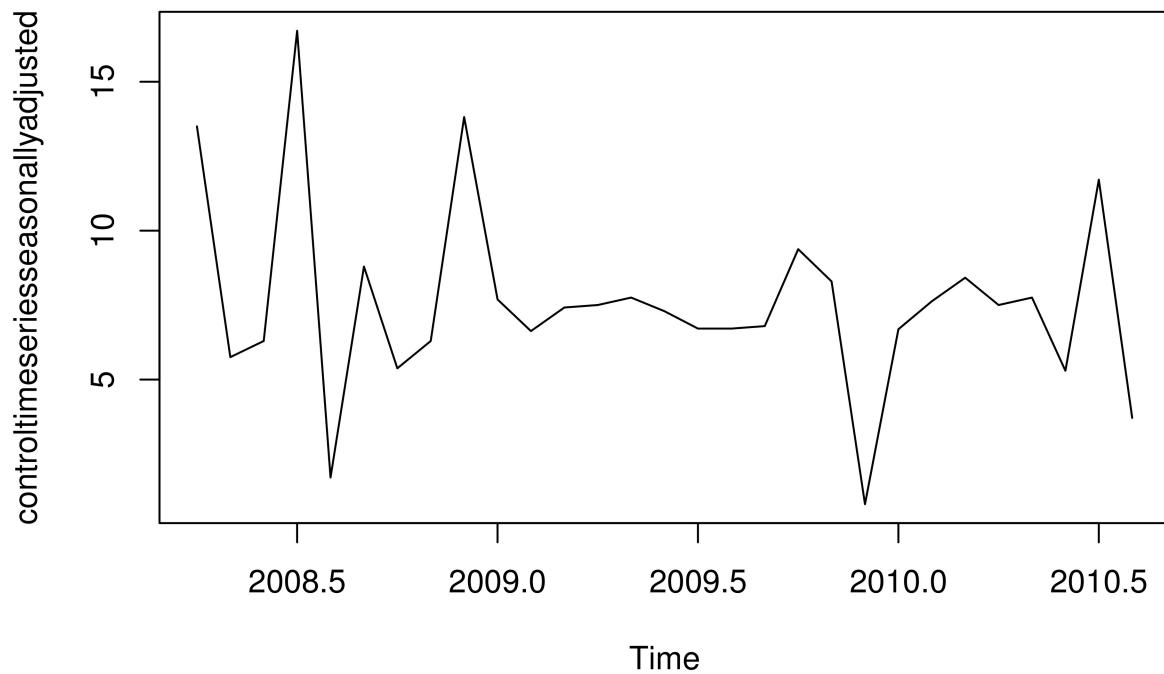
Decompose the data to find seasonal patterns

```
controltimeseriescomponents <- decompose(control_ts)
plot(controltimeseriescomponents)
```

Decomposition of additive time series



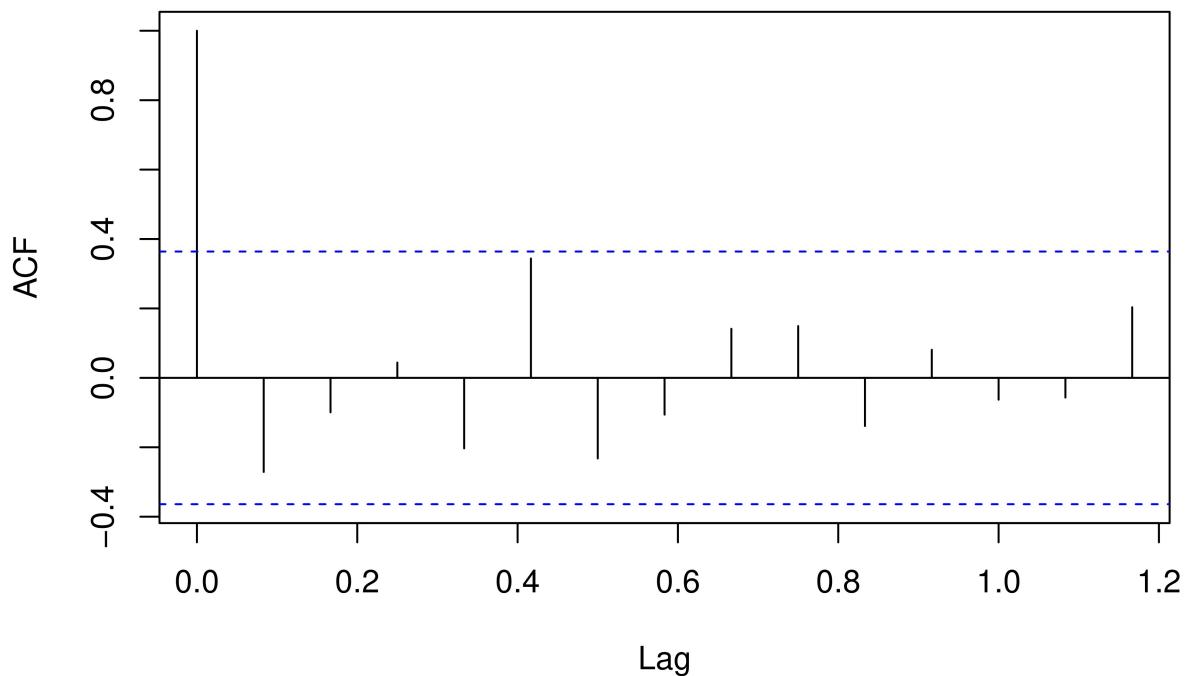
```
controltimeseriesseasonallyadjusted <- control_ts - controltimeseriescomponents$seasonal  
plot(controltimeseriesseasonallyadjusted)
```



Run Statistical Tests on Time Series for Stationarity

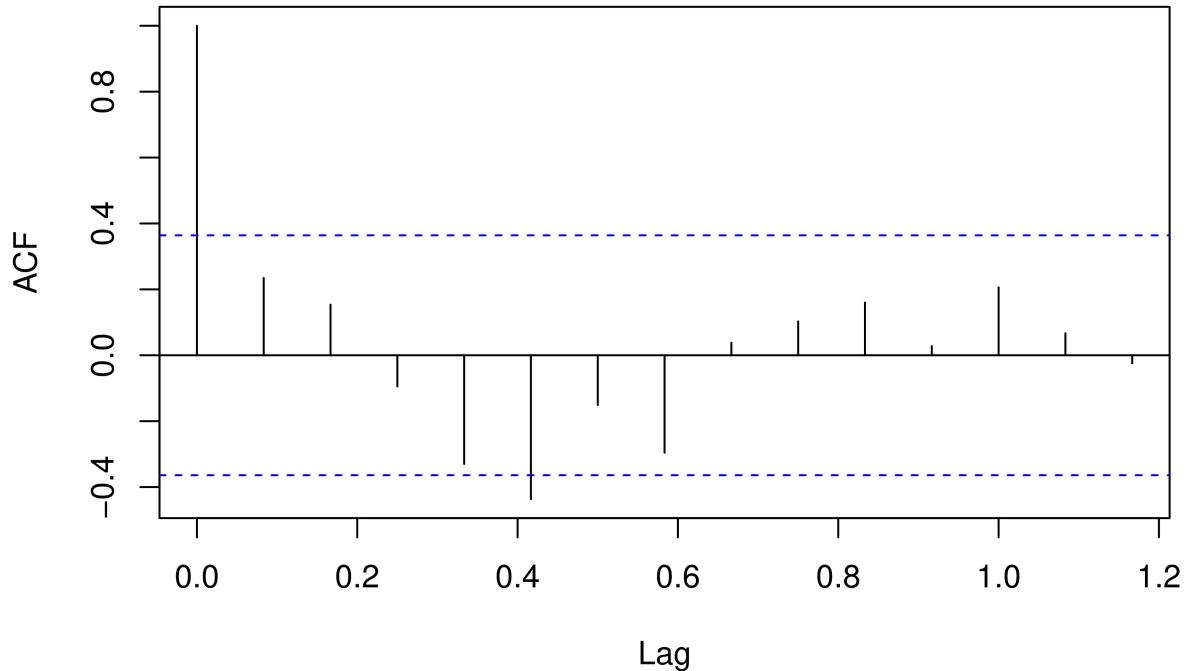
```
# source of statistical tests http://r-statistics.co/Time-Series-Analysis-With-R.html
acfcontrol <- acf(control_ts) # autocorrelation (i.e. a Time Series with lags of itself)
```

Series control_ts



```
acfcontrol <- acf(control_ts)
```

Series treatment_ts



```
# shows that the control time series is a "stationary time series"

png(here::here("Output","acfcontrol.png"))
plot(acfcontrol)

png(here::here("Output","acftreatmentNH.png"))
plot(acftreatment)

pacfcontrolNH <- pacf(control_ts) # partial autocorrelation (i.e. correlation of the time series with
pacftreatmentMA <- pacf(treatment_ts) # partial autocorrelation (i.e. correlation of the time series w

png(here::here("Output","pacfcontrolNH.png"))
plot(pacfcontrolNH)

png(here::here("Output","pacftreatmentMA.png"))
plot(pacftreatmentMA)

ccfRes <- ccf(control_ts, treatment_ts, ylab = "cross-correlation")
ccfRes

##
## Autocorrelations of series 'X', by lag
##
## -0.9167 -0.8333 -0.7500 -0.6667 -0.5833 -0.5000 -0.4167 -0.3333 -0.2500 -0.1667
##  0.123   0.069  -0.052  -0.004  -0.078  -0.264  -0.145  -0.162   0.224   0.313
## -0.0833  0.0000  0.0833  0.1667  0.2500  0.3333  0.4167  0.5000  0.5833  0.6667
##  -0.016   0.275   0.110  -0.153  -0.093  -0.182   0.072   0.132  -0.055  -0.139
```

```

##  0.7500  0.8333  0.9167
##  0.116 -0.098   0.123
# adf test is an Augmented Dickey-Fuller Test
adf.test(control_ts) # p-value < 0.05 indicates the TS is stationary

## Warning in adf.test(control_ts): p-value smaller than printed p-value
##
## Augmented Dickey-Fuller Test
##
## data: control_ts
## Dickey-Fuller = -4.3878, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
adf.test(treatment_ts)

##
## Augmented Dickey-Fuller Test
##
## data: treatment_ts
## Dickey-Fuller = -3.4088, Lag order = 3, p-value = 0.07523
## alternative hypothesis: stationary
kpss.test(control_ts) # Kwiatkowski-Phillips-Schmidt-Shin (KPSS) testz

## Warning in kpss.test(control_ts): p-value greater than printed p-value
##
## KPSS Test for Level Stationarity
##
## data: control_ts
## KPSS Level = 0.20559, Truncation lag parameter = 2, p-value = 0.1
kpss.test(treatment_ts)

## Warning in kpss.test(treatment_ts): p-value greater than printed p-value
##
## KPSS Test for Level Stationarity
##
## data: treatment_ts
## KPSS Level = 0.06372, Truncation lag parameter = 2, p-value = 0.1
# https://www.sas.com/content/dam/SAS/en\_ca/User%20Group%20Presentations/Health-User-Groups/ITS\_SAS.pdf
```

ITS analyses use regression-based techniques

```

quasiexp <- experiment[experiment$type != "test",]

# Added dummy variables for ITS
control <- as.data.frame(t(rbind(quasiexp$yearmonth, quasiexp$controlpermil)))
control$treat <- as.vector(rep(0, nrow(control)))                                # Create example vector
time <- 1:nrow(control)
control$time <- as.vector(time)
control$z <- c(rep(0, 6), 1:(nrow(control)-6))
```

```

treatment <- as.data.frame(t(rbind(quasiexp$yearmonth, quasiexp$treatpermil)))
treatment$treat <- as.vector(rep(1,nrow(control)))                                # Create example vector
time <- 1:nrow(control)
treatment$time <- as.vector(time)
treatment$z <- c(rep(0,6),1:(nrow(control)-6))
treatment

##          V1            V2 treat time z
## 1 2008/04 4.64423963347661     1    1 0
## 2 2008/05 3.86833308266535     1    2 0
## 3 2008/06 6.34100698284062     1    3 0
## 4 2008/07 5.87419908000767     1    4 0
## 5 2008/08 3.55320954466238     1    5 0
## 6 2008/09 4.01415762516259     1    6 0
## 7 2010/03 5.34817048256084     1    7 1
## 8 2010/04 5.19271937979382     1    8 2
## 9 2010/05 4.73004056391238     1    9 3
## 10 2010/06 5.64017426004343    1   10 4
## 11 2010/07 3.80731513162574    1   11 5
## 12 2010/08 3.34842814889608    1   12 6

AppendITS <- rbind(treatment,control)
names(AppendITS) <- c("yearmonth","incident_permil","treat","time","z")
AppendITS$incident_permil <- as.numeric(as.character(AppendITS$incident_permil))
AppendITS$time <- as.numeric(as.character(AppendITS$time))
AppendITS$z <- as.numeric(as.character(AppendITS$z))
AppendITS

##      yearmonth incident_permil treat time z
## 1 2008/04      4.644240     1    1 0
## 2 2008/05      3.868333     1    2 0
## 3 2008/06      6.341007     1    3 0
## 4 2008/07      5.874199     1    4 0
## 5 2008/08      3.553210     1    5 0
## 6 2008/09      4.014158     1    6 0
## 7 2010/03      5.348170     1    7 1
## 8 2010/04      5.192719     1    8 2
## 9 2010/05      4.730041     1    9 3
## 10 2010/06     5.640174     1   10 4
## 11 2010/07     3.807315     1   11 5
## 12 2010/08     3.348428     1   12 6
## 13 2008/04     11.406289    0    1 0
## 14 2008/05     3.041027    0    2 0
## 15 2008/06     6.080755    0    3 0
## 16 2008/07     9.879125    0    4 0
## 17 2008/08     3.039694    0    5 0
## 18 2008/09     6.839223    0    6 0
## 19 2010/03     6.077046    0    7 1
## 20 2010/04     6.836464    0    8 2
## 21 2010/05     4.557307    0    9 3
## 22 2010/06     5.316462    0   10 4
## 23 2010/07     6.075509    0   11 5
## 24 2010/08     4.555639    0   12 6

```

```

factor_cols <- c("treat", "time", "z")

sapply(AppendITS, class)

##      yearmonth incident_permil          treat           time           z
##   "character"     "numeric"     "numeric"     "numeric"     "numeric"
regTest <- lm(incident_permil ~ time + treat + z, AppendITS)
summary(regTest)

##
## Call:
## lm(formula = incident_permil ~ time + treat + z, data = AppendITS)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.8555 -0.8660 -0.0786  0.8076  4.3133
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.28936   1.24607   5.850 1.01e-05 ***
## time        -0.19640   0.27528  -0.713   0.4838
## treat       -1.44521   0.77125  -1.874   0.0756 .
## z            0.07386   0.44694   0.165   0.8704
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.889 on 20 degrees of freedom
## Multiple R-squared:  0.2144, Adjusted R-squared:  0.0966
## F-statistic:  1.82 on 3 and 20 DF,  p-value: 0.176
regTest2 <- lm(incident_permil ~ time + treat + time*treat + z + z*time + z*treat + z*treat*time, AppendITS)
summary(regTest2)

##
## Call:
## lm(formula = incident_permil ~ time + treat + time * treat +
##     z + z * time + z * treat + z * treat * time, data = AppendITS)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.4733 -0.6230 -0.1693  0.8501  3.3960
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 8.50614   1.84537   4.609  0.00029 ***
## time        -0.49590   0.45851  -1.082  0.29549
## treat       -3.37395   2.60974  -1.293  0.21443
## z            1.22177   2.63327   0.464  0.64892
## time:treat  0.38265   0.64844   0.590  0.56336
## time:z       -0.07072   0.20175  -0.351  0.73050
## treat:z      0.55078   3.72401   0.148  0.88427
## time:treat:z -0.08418   0.28531  -0.295  0.77174
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```
##  
## Residual standard error: 2.017 on 16 degrees of freedom  
## Multiple R-squared:  0.2834, Adjusted R-squared: -0.03006  
## F-statistic: 0.9041 on 7 and 16 DF, p-value: 0.5271
```

```
AppendITS
```

```
##   yearmonth incident_permil treat time z  
## 1 2008/04      4.644240    1   1 0  
## 2 2008/05      3.868333    1   2 0  
## 3 2008/06      6.341007    1   3 0  
## 4 2008/07      5.874199    1   4 0  
## 5 2008/08      3.553210    1   5 0  
## 6 2008/09      4.014158    1   6 0  
## 7 2010/03      5.348170    1   7 1  
## 8 2010/04      5.192719    1   8 2  
## 9 2010/05      4.730041    1   9 3  
## 10 2010/06     5.640174    1  10 4  
## 11 2010/07     3.807315    1  11 5  
## 12 2010/08     3.348428    1  12 6  
## 13 2008/04     11.406289   0   1 0  
## 14 2008/05     3.041027   0   2 0  
## 15 2008/06     6.080755   0   3 0  
## 16 2008/07     9.879125   0   4 0  
## 17 2008/08     3.039694   0   5 0  
## 18 2008/09     6.839223   0   6 0  
## 19 2010/03     6.077046   0   7 1  
## 20 2010/04     6.836464   0   8 2  
## 21 2010/05     4.557307   0   9 3  
## 22 2010/06     5.316462   0  10 4  
## 23 2010/07     6.075509   0  11 5  
## 24 2010/08     4.555639   0  12 6
```

```
# View(AppendITS)
```