

# HITECH\_Act\_Case

Karl Grindal

6/19/2020

```
HITECH <- read.csv(here::here("Data","Other_data","HITECH.csv"))
DUNS <- read.csv(here::here("Data","Other_data","DUNS_HITECH.csv"))

NAICS <- merge(HITECH, DUNS, by="clean_name", all=TRUE)

NAICS$DUNS_2 <- substr(NAICS$Primary.NAICS.Code, start=1, stop=2)
# colSums(!is.na(NAICS)) # 1740 resolved for Duns_2
# colSums(is.na(NAICS)) # 248 not resolved for Duns_2
NAICS <- NAICS[complete.cases(NAICS$firstdate),]

248/1740 # = 14.2% are not included

## [1] 0.1425287
1740/(1740+248) # = 87.5% of incidents are resolved

## [1] 0.8752515
write.csv(NAICS, here::here("Data","Other_data","NAICS_Clean.csv"))
```

## Specifying Time Variables

```
first_breach_date <- as.Date("2006-06-01") # doesn't change
data_start <- c(2006,6) # doesn't change
data_range <- c(2006,2011)
exp_start <- c(2008,4)
exp_range <- c(2008,2011)
first_pop_date <- as.Date("2000-04-01")
no_months_out <- 174

months_prior <- 6 # months before treatment start
months_after <- 6 # months after treatment start

# HIGH TECH Act regulations become effective on February 17, 2009
# Enforcement of HIGH TECH Act implement on May 27, 2009

treatment_start <- as.Date("02/17/2009", "%m/%d/%Y")-180 # Legislation H.B. 4144 becomes effective
treatment_end <- as.Date("05/27/2009", "%m/%d/%Y")+180 # moving avg uses 30 days post treatment
```

## 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)

datfull <- data.frame(seq(as.Date(first_breach_date), by="1 month", length.out=54))
names(datfull) <- "yearmonth"
datfull <- format(datfull, "%Y/%m")

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

pop[2:ncol(pop)] <- sapply(pop[2:ncol(pop)],as.numeric)

# Create a new column for the total population across collecting states
pop$totpop <- rowSums(pop[,c('Massachusetts','South Carolina','North Carolina','New Hampshire','Hawaii')])
```

## Create Monthly Frequency Accross Full Collected Time

```
NAICS$date_formatted <- format(as.Date(NAICS$firstdate, "%Y-%m-%d"), "%Y/%m") # Alternative is "%m/%d/%Y"

NAICS_ts <- NAICS %>%
  dplyr::group_by(NAICS$date_formatted) %>%
  dplyr::summarise(frequency = n())

## `summarise()` ungrouping output (override with ` `.groups` argument)
NAICS_ts # Feb 2005 and Dec 2010

## # A tibble: 62 x 2
##   `NAICS$date_formatted` frequency
##   <chr>                  <int>
## 1 2005/02                  1
## 2 2005/12                  2
## 3 2006/01                  3
## 4 2006/02                  3
## 5 2006/03                  3
## 6 2006/04                  3
## 7 2006/05                 14
## 8 2006/06                 11
```

```

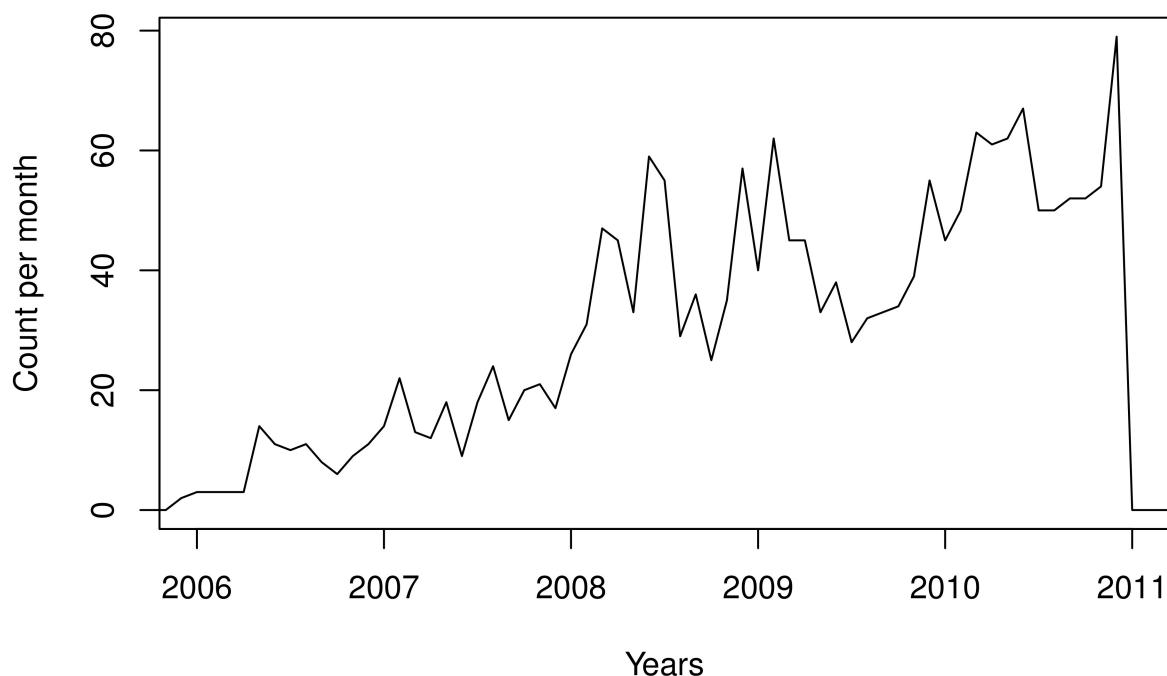
##  9 2006/07          10
## 10 2006/08          11
## # ... with 52 more rows

names(NAICS_ts)[1] <- "yearmonth"
NAICS_ts <- merge(NAICS_ts, pop, by="yearmonth", all.y=TRUE)
NAICS_ts$frequency[is.na(NAICS_ts$frequency)]<-0
# Look at Raw Frequency Counts

NAICS_ts <- ts(NAICS_ts$frequency, frequency = 12, start = c(2000,4))
ts.plot(NAICS_ts, main = "Breaches over time", xlim=data_range, gpars = list(col = c("black", "red"))),

```

## Breaches over time



## Designate Treatment and Control groups

```

# Format control dates into months
NAICS$date_formatted <- format(as.Date(NAICS$firstdate, "%Y-%m-%d"), "%Y/%m") # Alternative is "%m/%d/%Y"
NAICS <- NAICS[!is.na(NAICS$date_formatted),]

Health <- NAICS[grep1( "62", NAICS$DUNS_2), ]
Finance <- NAICS[grep1( "52", NAICS$DUNS_2), ]
Education <- NAICS[grep1( "61", NAICS$DUNS_2), ]
Information <- NAICS[grep1( "51", NAICS$DUNS_2), ]

Non_Health <- NAICS %>%

```

```

filter(DUNS_2 != 62)

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

treatment <- Health # This Must Be Filled in to Work Properly!
control <- Non_Health # This Must Be Filled in to Work Properly!

```

## 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<- merge(datfull,treatment_freq, by="yearmonth", all.x=TRUE) # WHAT IS THIS THING!!!!
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<- merge(datfull,control_freq, by="yearmonth", all.x=TRUE) # WHAT IS THIS THING!!!!
control_freq$frequency[is.na(control_freq$frequency)]<-0

ts(treatment_freq$frequency, frequency = 12, start = data_start)

##          Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2006              0   0   1   0   0   1   0   0   1   0
## 2007    0   0   0   1   0   2   0   1   1   1   2   2
## 2008    2   1   7   3   2   6   4   1   2   4   6   9
## 2009    4   4   2   5   3   4   3   4   1   5   2   5
## 2010    3   5   9   8   9  10   7   8   9   6   8

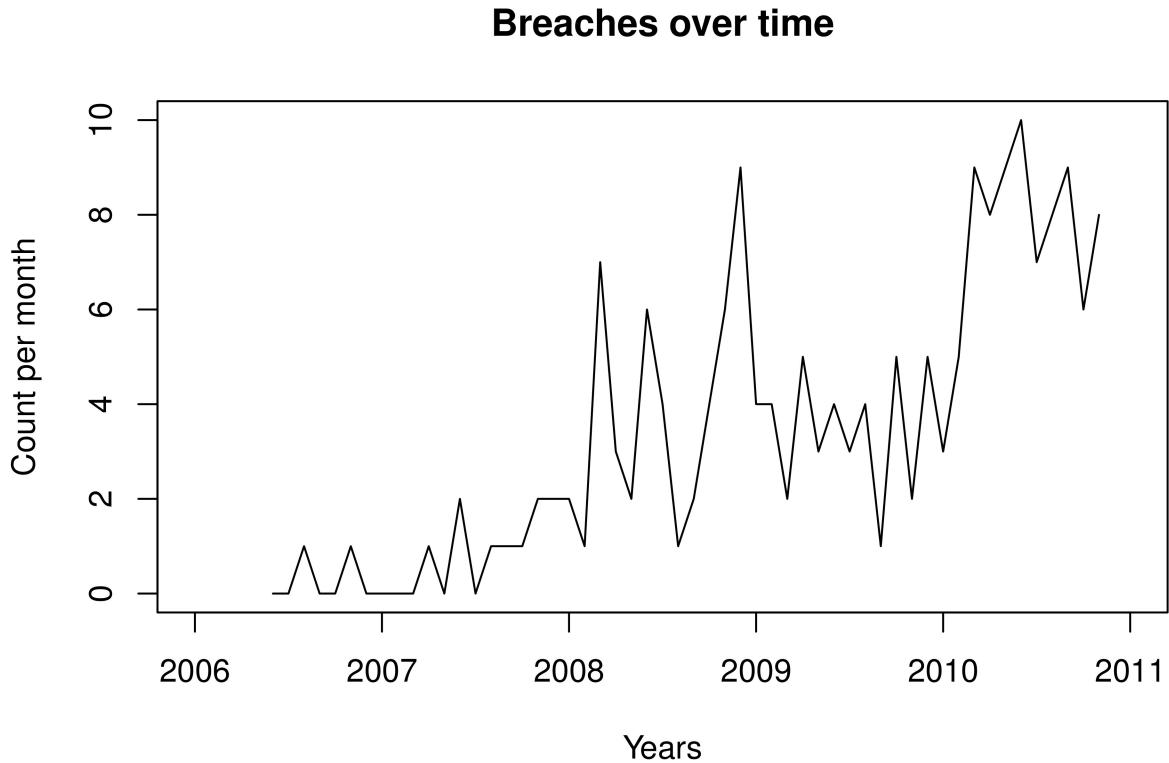
ts(control_freq$frequency, frequency = 12, start = data_start)

##          Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2006              8   9   9   6   6   7   8
## 2007   11  14   8  10  11   6  17  20  12  17  14  13
## 2008   23  24  34  37  29  50  48  23  32  19  24  43
## 2009   30  52  36  36  27  29  21  26  29  22  33  41
## 2010   40  39  43  43  45  49  37  37  33  38  41

```

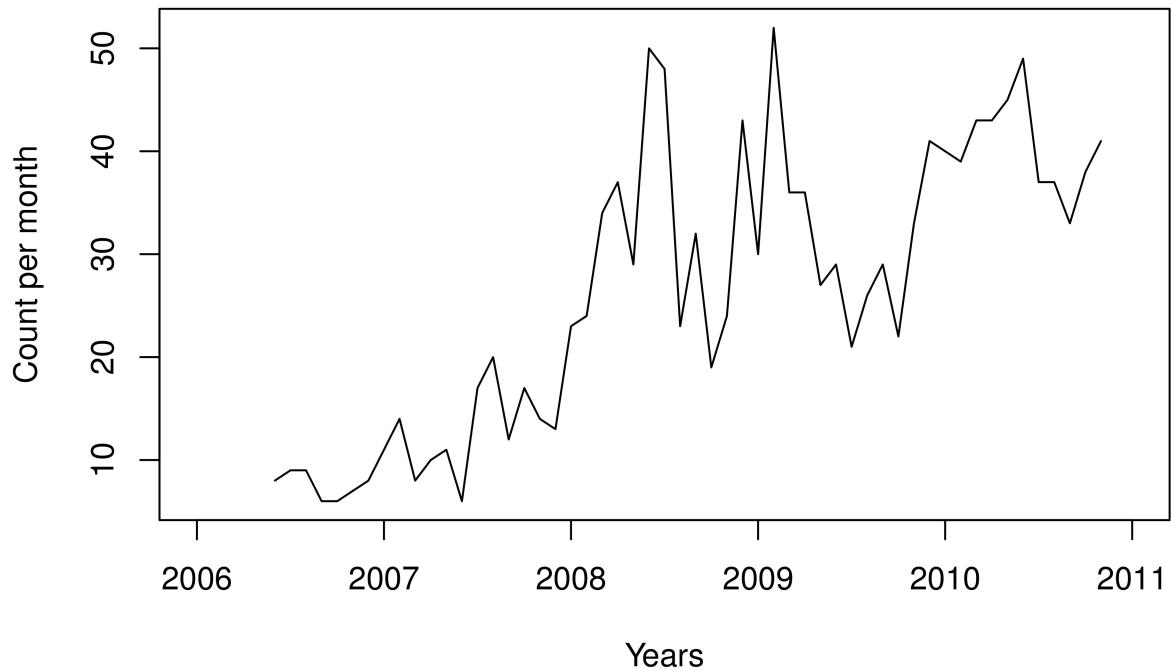
```
treatment_ts <- ts(treatment_freq$frequency, frequency = 12, start = data_start)
control_ts <- ts(control_freq$frequency, frequency = 12, start = data_start)

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



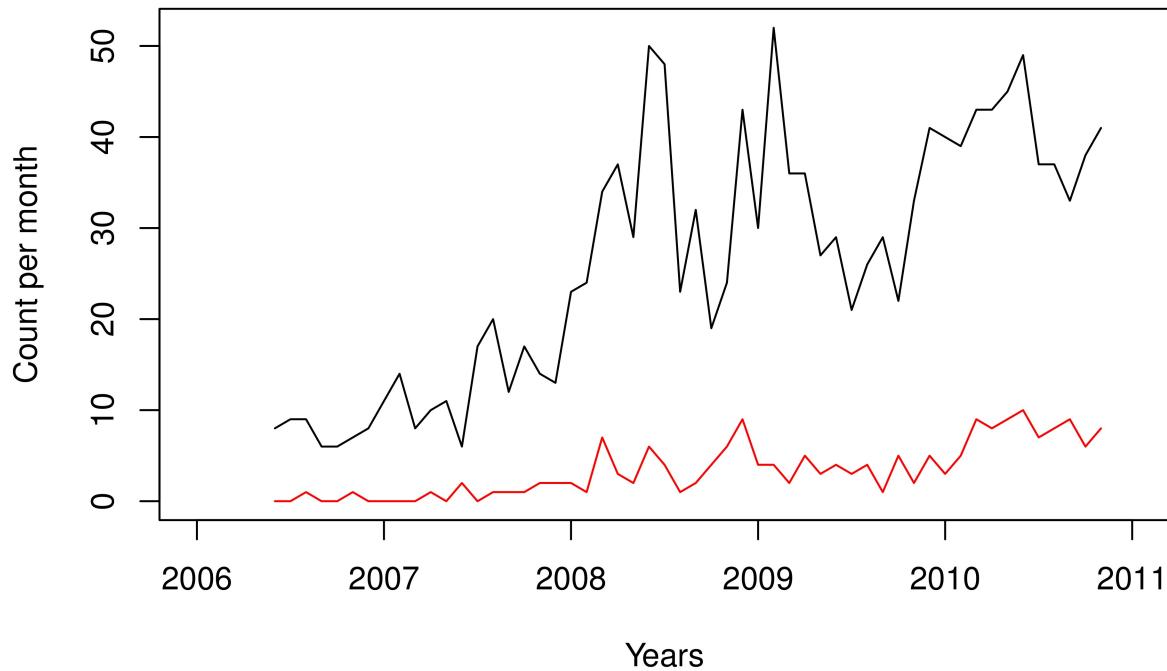
```
plot.ts(control_ts, main = "Breaches over time", xlim=data_range, xlab = "Years", ylab = "Count per mont
```

## 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.
## 0.000   1.000  2.500   3.389  5.000 10.000

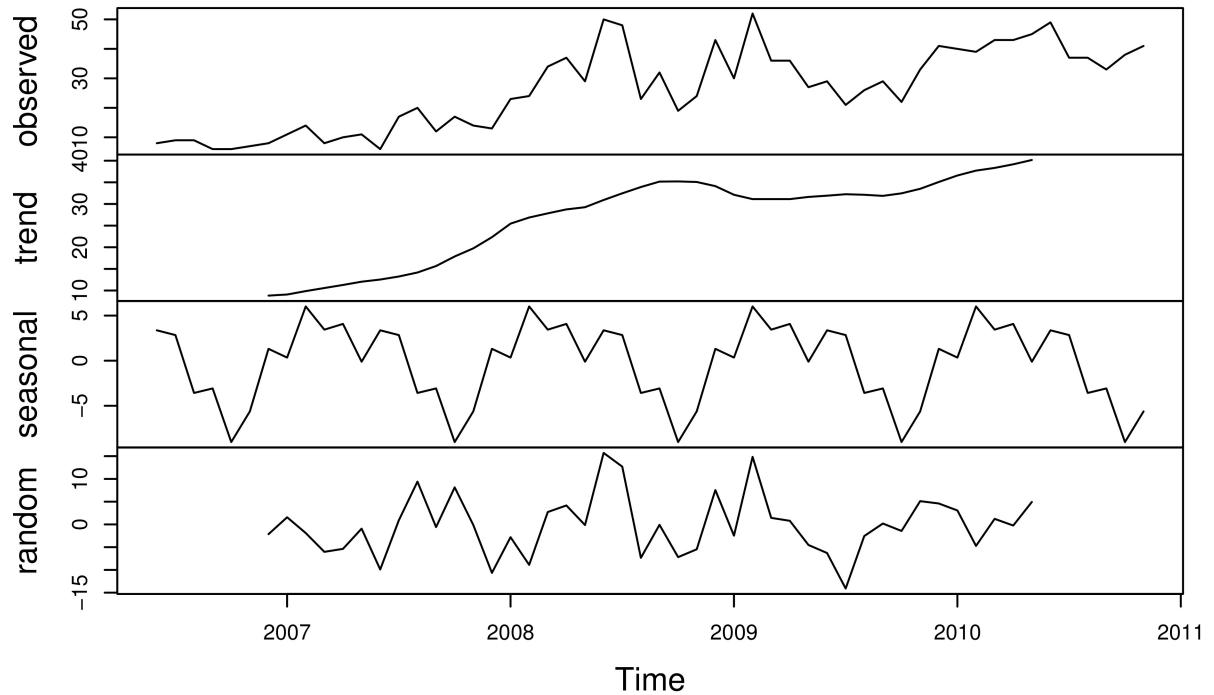
summary(control_ts)

##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
## 6.00   13.25  26.50  26.28  37.00  52.00
```

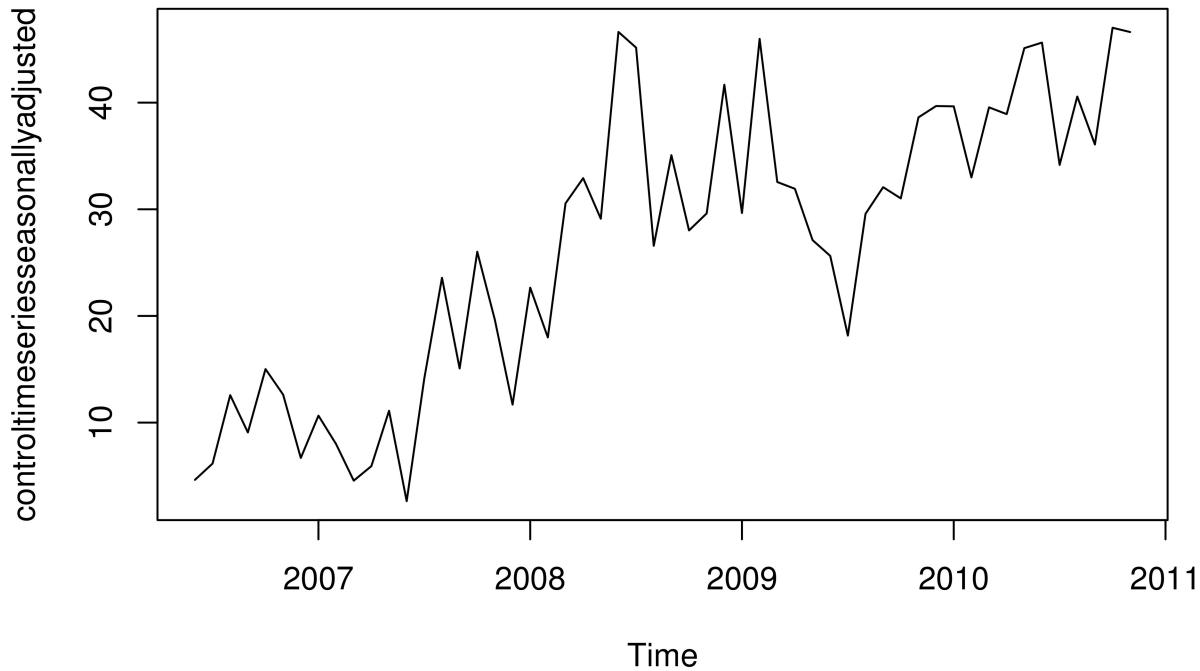
## Decompose the Control to Find Seasonal Patterns

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

## Decomposition of additive time series



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



## 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

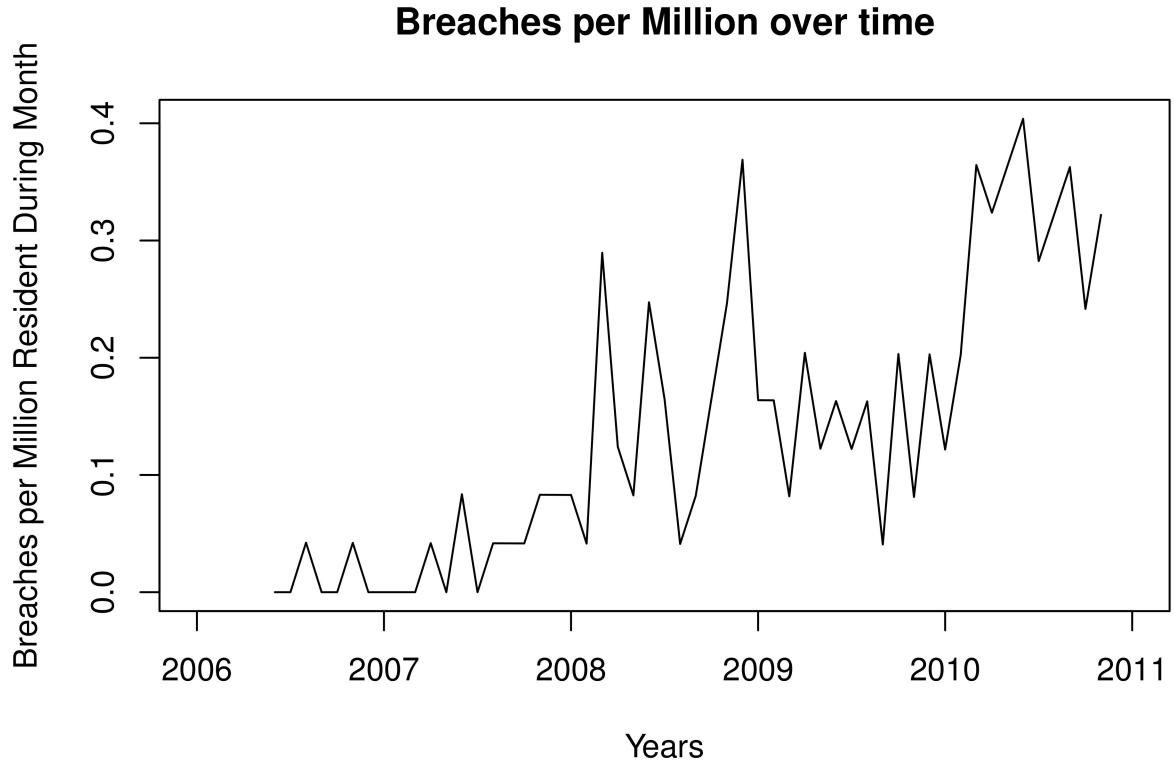
# change class of columns to numeric
comb_ts[2:ncol(comb_ts)] <- sapply(comb_ts[2:ncol(comb_ts)],as.numeric)

comb_ts$totpop <- as.numeric(as.character(comb_ts$totpop))
comb_ts$treatpermil <- comb_ts$frequency.x/(comb_ts$totpop/1000000)
comb_ts$controlpermil <- comb_ts$frequency.y/(comb_ts$totpop/1000000)

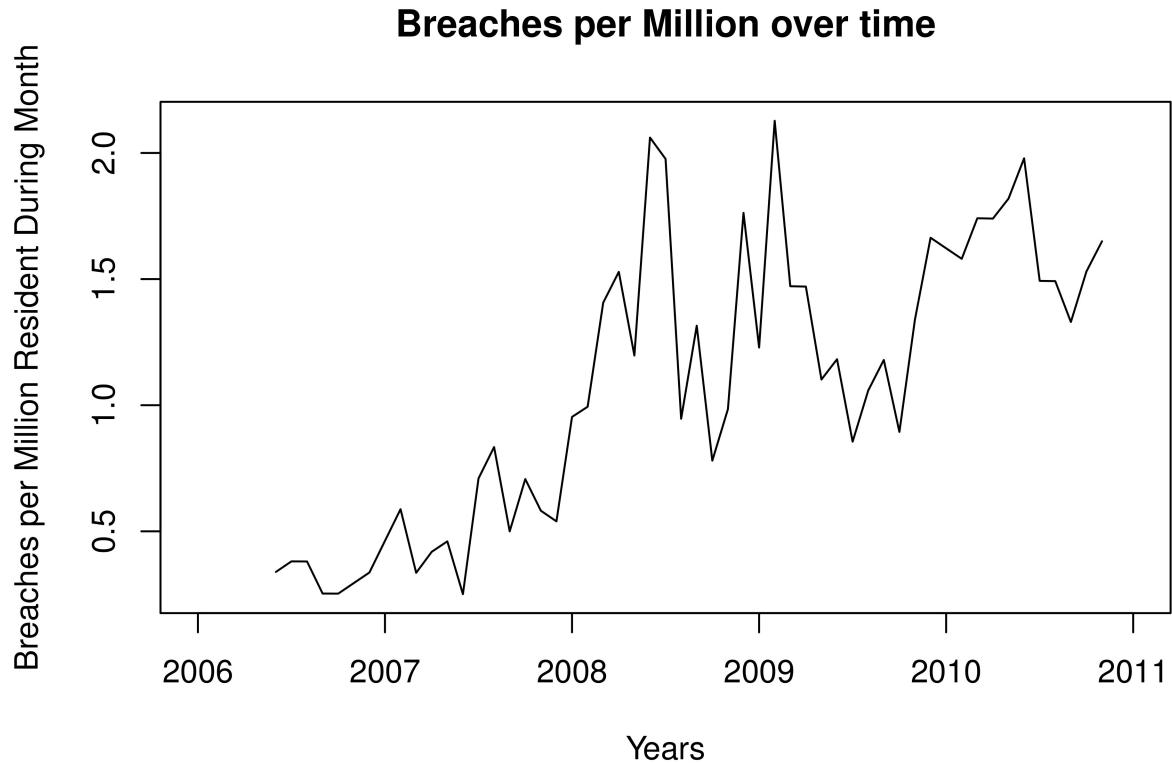
treatment_tsM <- ts(comb_ts$treatpermil, frequency = 12, start = data_start)
control_tsM <- ts(comb_ts$controlpermil, frequency = 12, start = data_start)

plot.ts(treatment_tsM, main = "Breaches per Million over time", xlim=data_range, xlab = "Years", ylab =

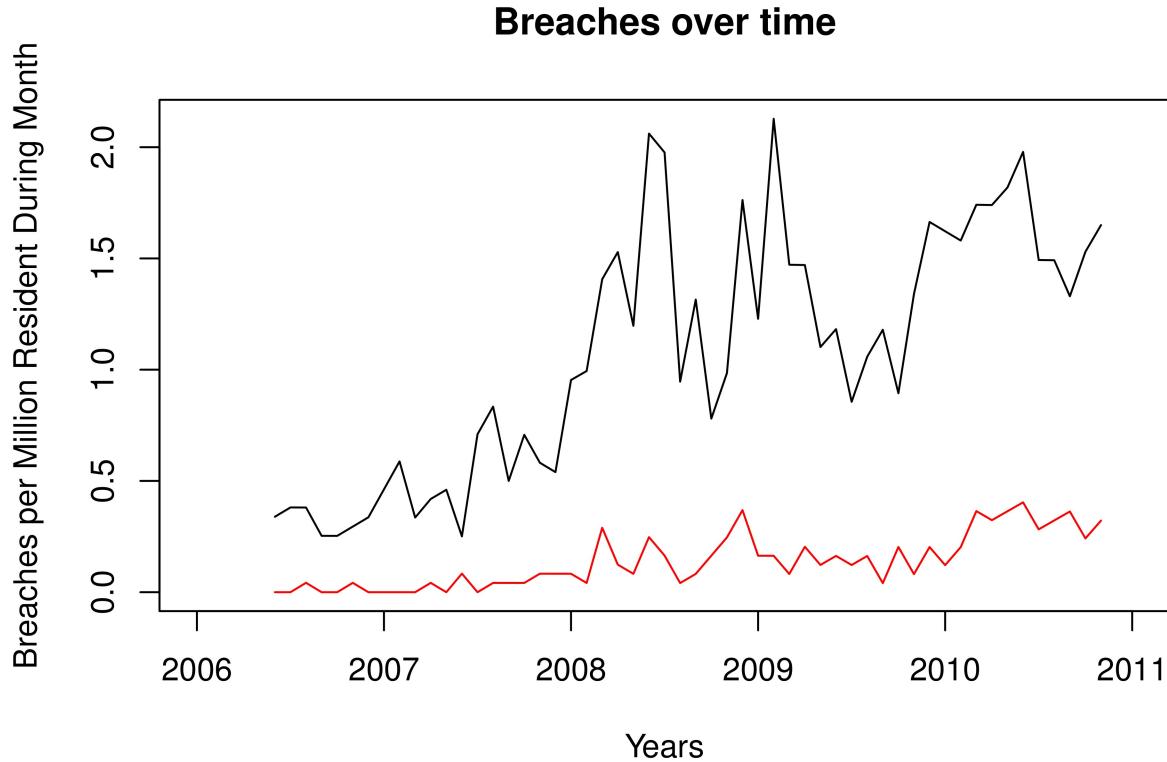
```



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

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

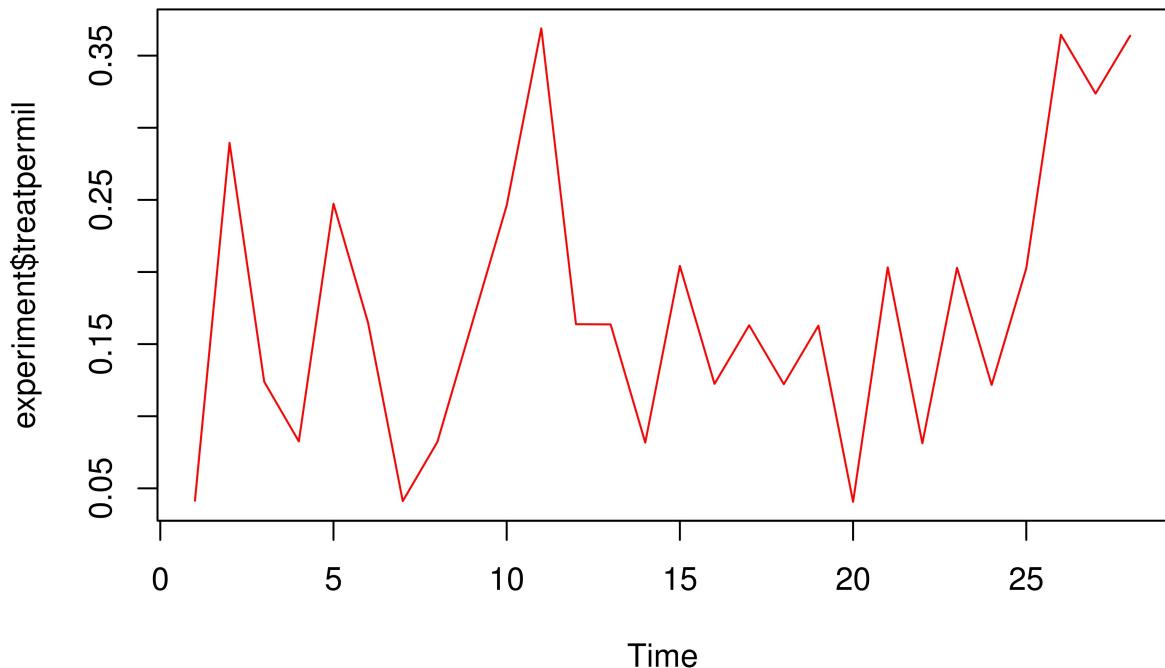
## [1] 0.09567449
mean(posttreat$controlpermil) - mean(pretreat$controlpermil)

## [1] 0.199573

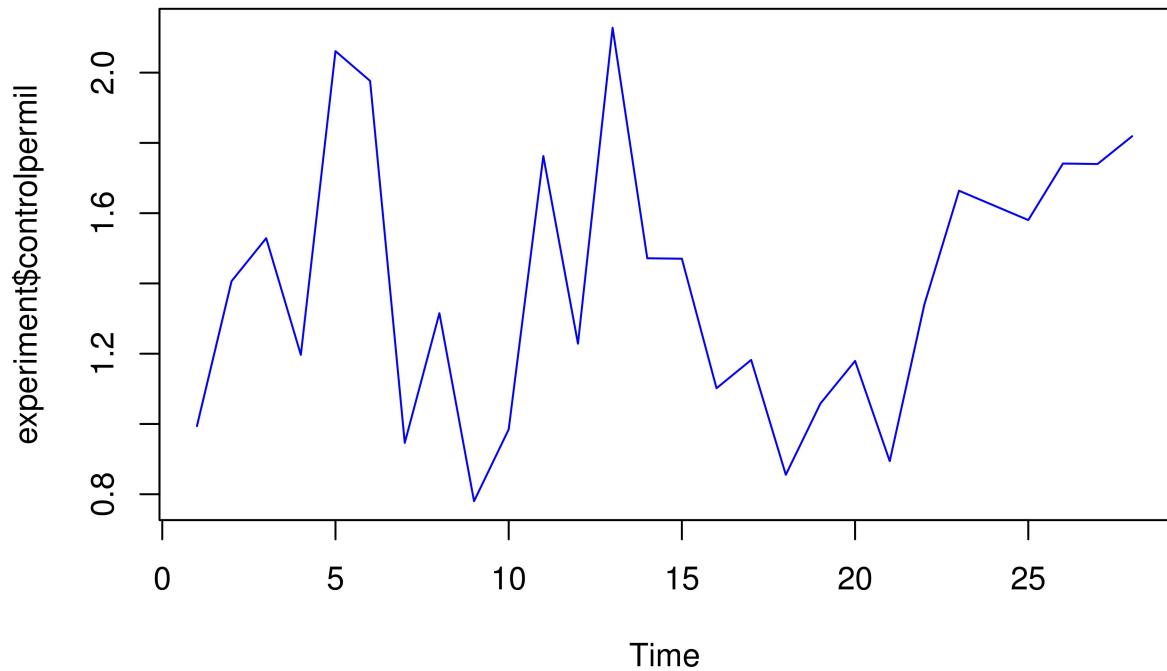
treatment_range <- comb_ts[(which(comb_ts$yearmonth==treatment_start)+1):(which(comb_ts$yearmonth==treatment_end))]
treatment_range$type <- "test"

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")
```

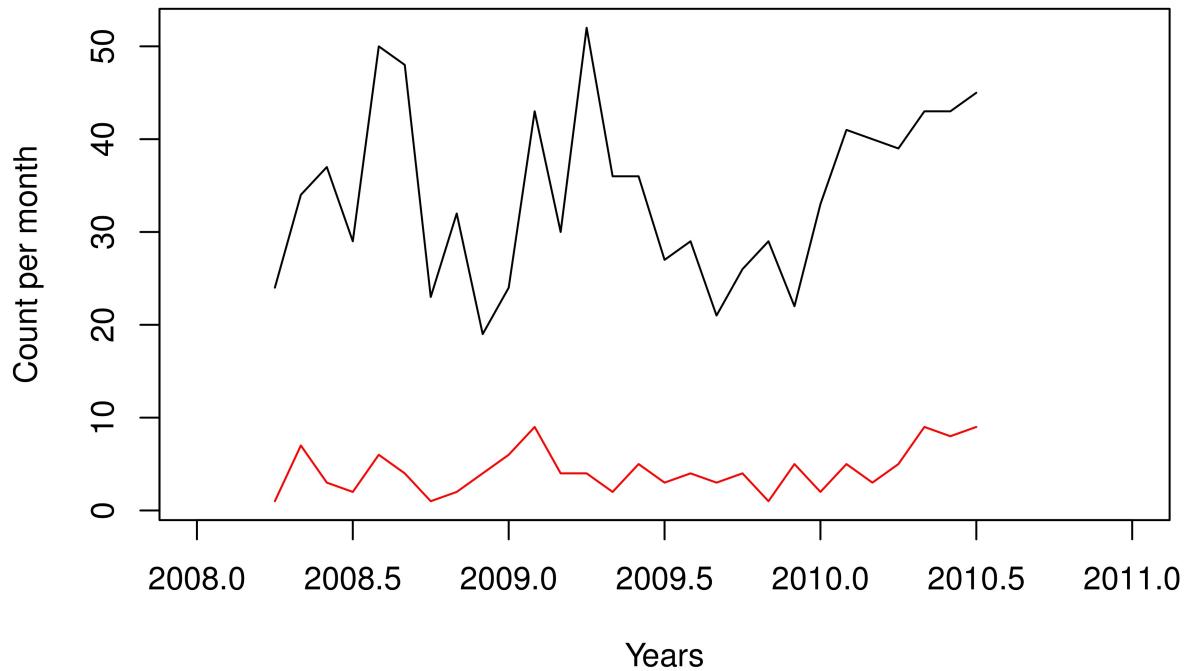


```
ts.plot(experiment$controlpermil, col = "blue")
```



```
# 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"), lty = c(1, 2)))
```

## Breaches over time



```
# Look at Treatment and Control per Million
treatment_tsM <- ts(experiment$treatpermil, frequency = 12, start = exp_start)
mean(treatment_tsM)

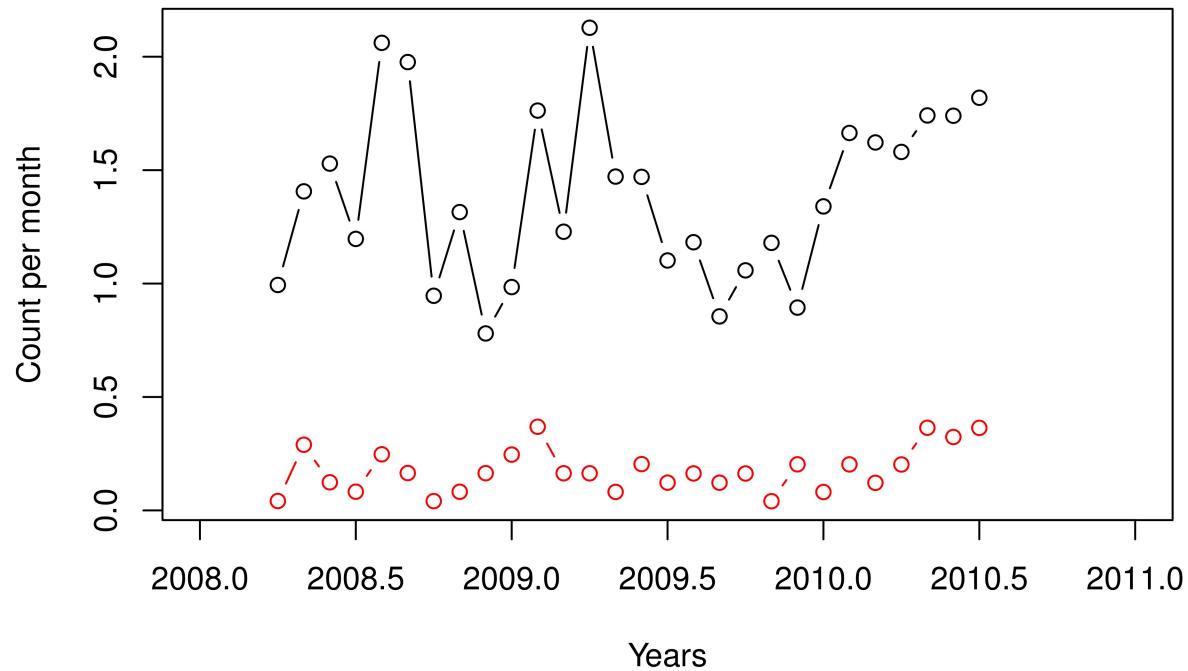
## [1] 0.1764458
sd(treatment_tsM)

## [1] 0.09775996
control_tsM <- ts(experiment$controlpermil, frequency = 12, start = exp_start)
mean(control_tsM)

## [1] 1.393913
sd(control_tsM)

## [1] 0.3789974
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
```

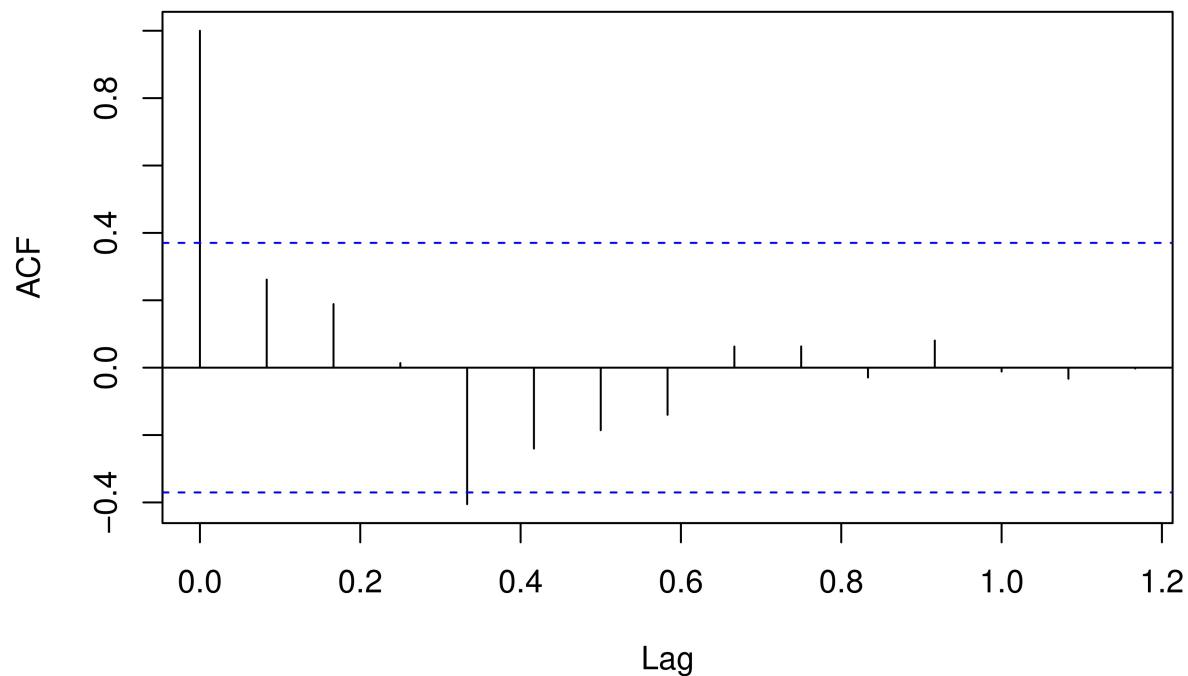
## Breaches over time



## Run Statistical Tests on Time Series for Stationarity

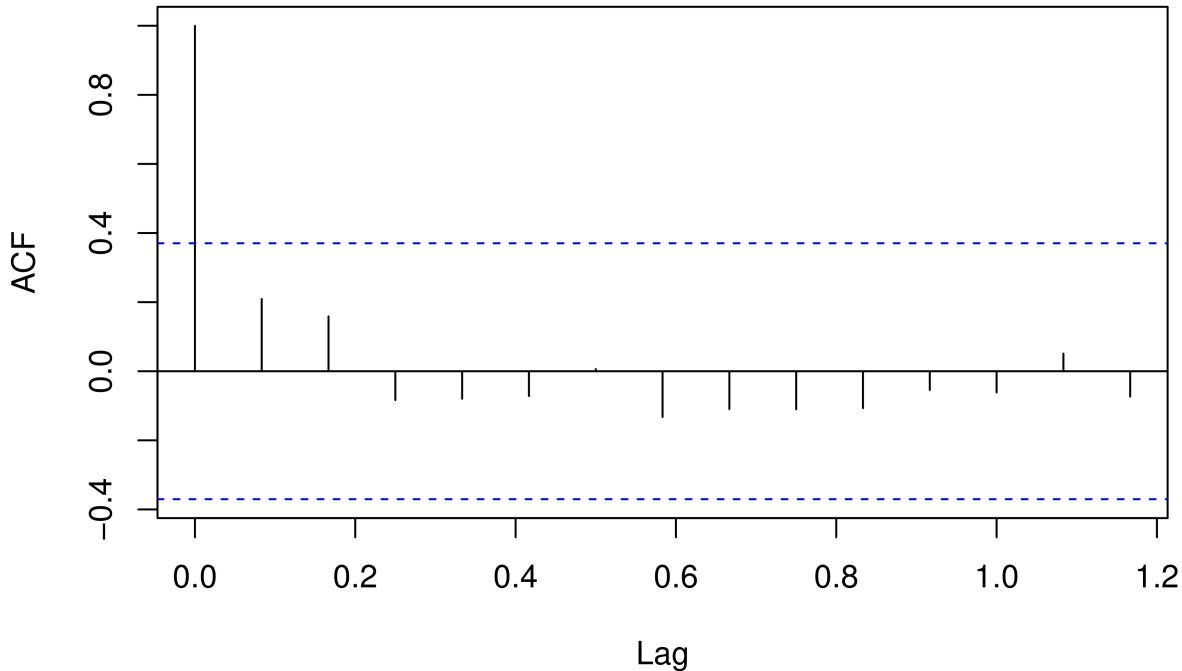
```
# source of statistical tests http://r-statistics.co/Time-Series-Analysis-With-R.html
acfcontrol <- acf(control_ts) # autocorrelation (i.e. a Time Series with lags of itself)
```

### **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
## -0.100 -0.142 -0.324 -0.157 -0.250 0.081 0.097 0.123 0.075 0.049
## -0.0833 0.0000 0.0833 0.1667 0.2500 0.3333 0.4167 0.5000 0.5833 0.6667
## -0.046 0.489 0.269 0.230 0.209 -0.064 -0.188 -0.096 -0.236 -0.075
```

```

##  0.7500  0.8333  0.9167
## -0.010  -0.198   0.096
# adf test is an Augmented Dickey-Fuller Test
adf.test(control_ts) # p-value < 0.05 indicates the TS is stationary

##
##  Augmented Dickey-Fuller Test
##
## data: control_ts
## Dickey-Fuller = -3.087, Lag order = 3, p-value = 0.1567
## alternative hypothesis: stationary
adf.test(treatment_ts)

##
##  Augmented Dickey-Fuller Test
##
## data: treatment_ts
## Dickey-Fuller = -1.6401, Lag order = 3, p-value = 0.7102
## 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 = 0.12812, 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 = 0.26387, 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 2008/02 0.0414154285972955    1   1 0
## 2 2008/03 0.289571124894053    1   2 0
## 3 2008/04 0.123957880930109    1   3 0
## 4 2008/05 0.0825427798592315    1   4 0
## 5 2008/06 0.247341582972115    1   5 0
## 6 2008/07 0.164703673386028    1   6 0
## 7 2008/08 0.041138734994595    1   7 1
## 8 2009/11 0.0812268041630687    1   8 2
## 9 2009/12 0.202916692065073    1   9 3
## 10 2010/01 0.121659967528144   1  10 4
## 11 2010/02 0.20261675486574   1  11 5
## 12 2010/03 0.364440783839238   1  12 6
## 13 2010/04 0.323708298683058   1  13 7
## 14 2010/05 0.363823302664242   1  14 8

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 2008/02      0.04141543    1   1 0
## 2 2008/03      0.28957112    1   2 0
## 3 2008/04      0.12395788    1   3 0
## 4 2008/05      0.08254278    1   4 0
## 5 2008/06      0.24734158    1   5 0
## 6 2008/07      0.16470367    1   6 0
## 7 2008/08      0.04113873    1   7 1
## 8 2009/11      0.08122680    1   8 2
## 9 2009/12      0.20291669    1   9 3
## 10 2010/01     0.12165997    1  10 4
## 11 2010/02     0.20261675    1  11 5
## 12 2010/03     0.36444078    1  12 6
## 13 2010/04     0.32370830    1  13 7
## 14 2010/05     0.36382330    1  14 8
## 15 2008/02     0.99397029    0   1 0
## 16 2008/03     1.40648832    0   2 0
## 17 2008/04     1.52881386    0   3 0
## 18 2008/05     1.19687031    0   4 0
## 19 2008/06     2.06117986    0   5 0
## 20 2008/07     1.97644408    0   6 0
## 21 2008/08     0.94619090    0   7 1
## 22 2009/11     1.34024227    0   8 2
## 23 2009/12     1.66391687    0   9 3
## 24 2010/01     1.62213290    0  10 4
## 25 2010/02     1.58041069    0  11 5
## 26 2010/03     1.74121708    0  12 6

```

```

## 27 2010/04      1.73993211      0   13 7
## 28 2010/05      1.81911651      0   14 8
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
## -0.59681 -0.08893  0.01269  0.07855  0.57252
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.329135  0.152996  8.687 7.12e-09 ***
## time        0.031905  0.033011  0.967  0.343
## treat       -1.354704  0.089237 -15.181 8.36e-14 ***
## z           -0.009472  0.047168 -0.201   0.843
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2361 on 24 degrees of freedom
## Multiple R-squared:  0.9076, Adjusted R-squared:  0.8961
## F-statistic: 78.63 on 3 and 24 DF,  p-value: 1.483e-12

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
## -0.66286 -0.06992 -0.00515  0.12269  0.44173
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.03763   0.20664   5.021 6.54e-05 ***
## time        0.11636   0.05040   2.309  0.03177 *
## treat       -0.89704   0.29224  -3.070  0.00605 **
## z           -0.39415   0.19508  -2.020  0.05694 .
## time:treat -0.11520   0.07128  -1.616  0.12173
## time:z       0.02157   0.01190   1.813  0.08489 .
## treat:z      0.33064   0.27589   1.198  0.24476
## time:treat:z -0.01480   0.01683  -0.879  0.38975

```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.228 on 20 degrees of freedom
## Multiple R-squared:  0.9283, Adjusted R-squared:  0.9031
## F-statistic: 36.96 on 7 and 20 DF,  p-value: 4.333e-10
```