# FTC_Wyndham

Karl Grindal

1/28/2021

## Case 1: Massachusetts Data Security Law

```r
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
```

```r
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
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$Massachusetts[!is.na(AllStateClean$Massachusetts)]<- 1
AllStateClean$New_Hampshire[!is.na(AllStateClean$New_Hampshire)]<- 1
AllStateClean$North_Carolina[!is.na(AllStateClean$North_Carolina)]<- 1
AllStateClean$California[!is.na(AllStateClean$California)]<- 1
AllStateClean$South_Carolina[!is.na(AllStateClean$South_Carolina)]<- 1
AllStateClean$Hawaii[!is.na(AllStateClean$Hawaii)]<- 1
AllStateClean$Iowa[!is.na(AllStateClean$Iowa)]<- 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
AllStateClean$California[is.na(AllStateClean$California)]<- 0
AllStateClean$South_Carolina[is.na(AllStateClean$South_Carolina)]<- 0
AllStateClean$Hawaii[is.na(AllStateClean$Hawaii)]<- 0
AllStateClean$Iowa[is.na(AllStateClean$Iowa)]<- 0
```

# Create Population Time Series for Matching with Incident Frequncy

```
# Creating blank frequency starting with earliest date

dat2 <- data.frame(seq(as.Date("2006-06-01"), by="1 month", length.out=174)) # treatment date
names(dat2) <- "yearmonth"
dat2 <- format(dat2,  "%Y/%m")

# Population
pop <- read.csv(here::here("Data","Other_data","populations.csv")) # starts at 2000.04.01
pop <- pop[c(5, 12, 16, 22, 30, 34, 42),]
# California row 5, Hawaii row 12, Iowa row 16, Massachusetts row 22, New Hampshire row 30, North Carol

datforpop <- data.frame(seq(as.Date("2000-04-01"), 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)

pop[,2:ncol(pop)] <- sapply(pop[,2:ncol(pop)],as.numeric)
pop$sevenpop <- rowSums(pop[,2:ncol(pop)])
```

## Identifying treatment and control options

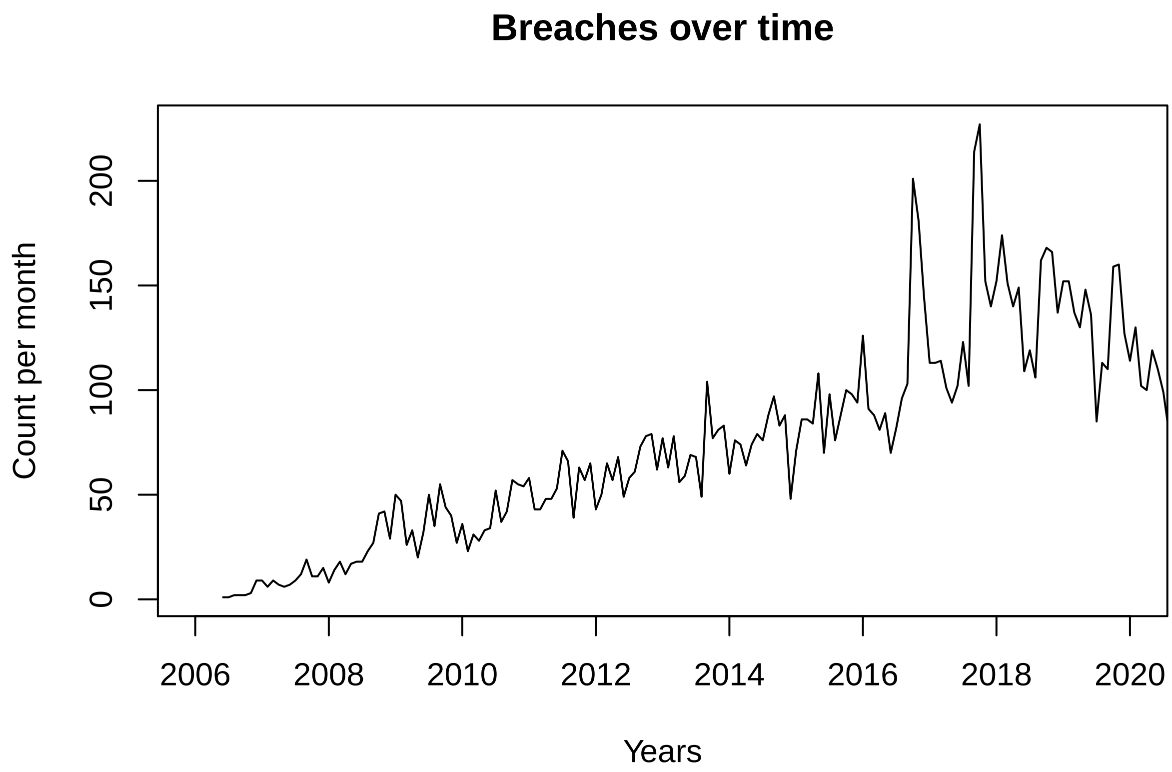
```
# Case 1: Experiment 1
massachusetts <- dplyr::filter(AllStateClean,Massachusetts==1)
massachusetts$reported_date <- substr(massachusetts$reported_date, start=1, stop=10)

#
sevenstates <- AllStateClean %>%
                              dplyr::filter(AllStateClean$Massachusetts == 1 | AllStateClean$New_Hampshi
                                             AllStateClean$California == 1 | AllStateClean$South_Caroli
```

## Experiment 1: Collect all breaches accross relevant collecting states

```
treatment <- sevenstates # This Must Be Filled in to Work Properly!

# 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

treatment_ts <- ts(treatment_freq$frequency, frequency = 12, start = c(2006,6))

plot.ts(treatment_ts, main = "Breaches over time", xlim=c(2006,2020), xlab = "Years", ylab = "Count per
```

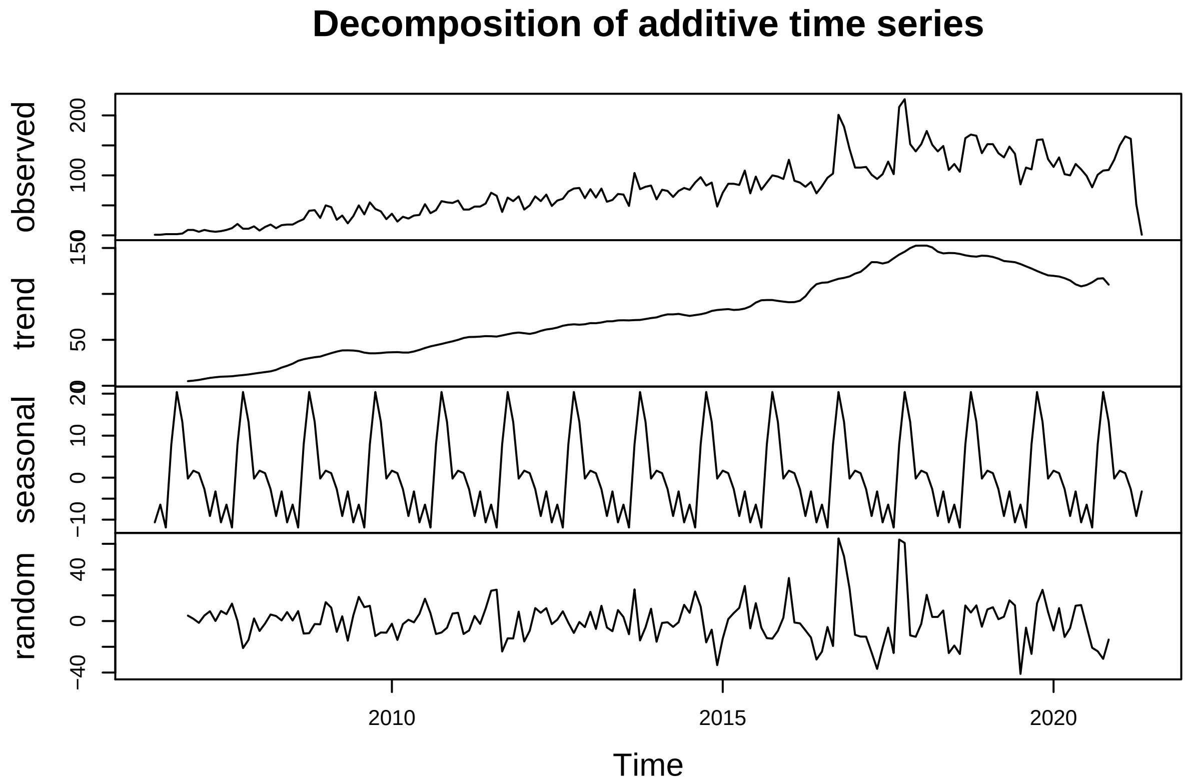



```
summary(treatment_ts)
```

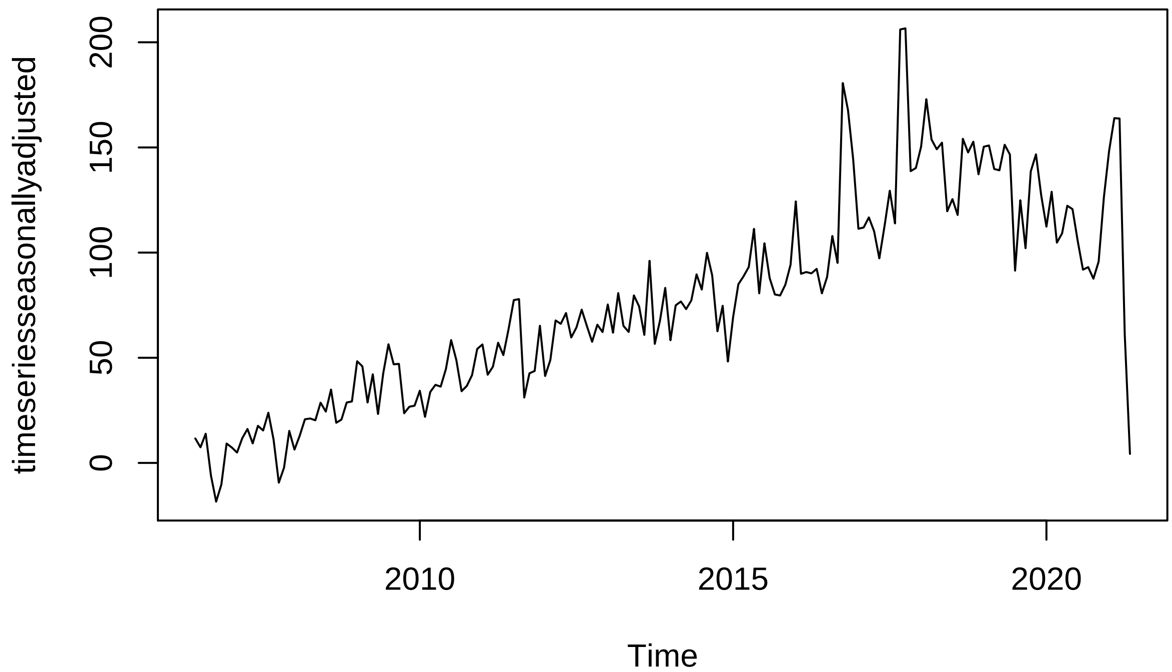
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   40.75   72.00   76.07  106.50  227.00
```

### Decompose the Massachusetts Data to Find Seasonal Patterns

```
timeseriescomponents <- decompose(treatment_ts)
plot(timeseriescomponents)
```




```
timeseriesseasonallyadjusted <- treatment_ts - timeseriescomponents$seasonal
plot(timeseriesseasonallyadjusted)
```




### Create charts with breaches per million residents

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

comb_ts$sevenpop <- as.numeric(as.character(comb_ts$sevenpop))
comb_ts$treatpermil <- comb_ts$frequency/(comb_ts$sevenpop/1000000)
comb_ts$treatpermil
```

```
##   [1] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##   [7] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##  [13] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##  [19] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##  [25] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##  [31] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##  [37] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##  [43] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##  [49] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##  [55] 0.00000000 0.00000000 0.00000000 0.00000000 0.01651349 0.00000000
##  [61] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##  [67] 0.00000000 0.00000000 0.01639629 0.03276818 0.03274383 0.03271952
##  [73] 0.04904285 0.14701946 0.14691053 0.09786784 0.14668925 0.11400426
##  [79] 0.09764316 0.11382991 0.14624092 0.19483902 0.30825977 0.17833013
##  [85] 0.17819429 0.24280725 0.12939871 0.22627563 0.29064977 0.19358283
```

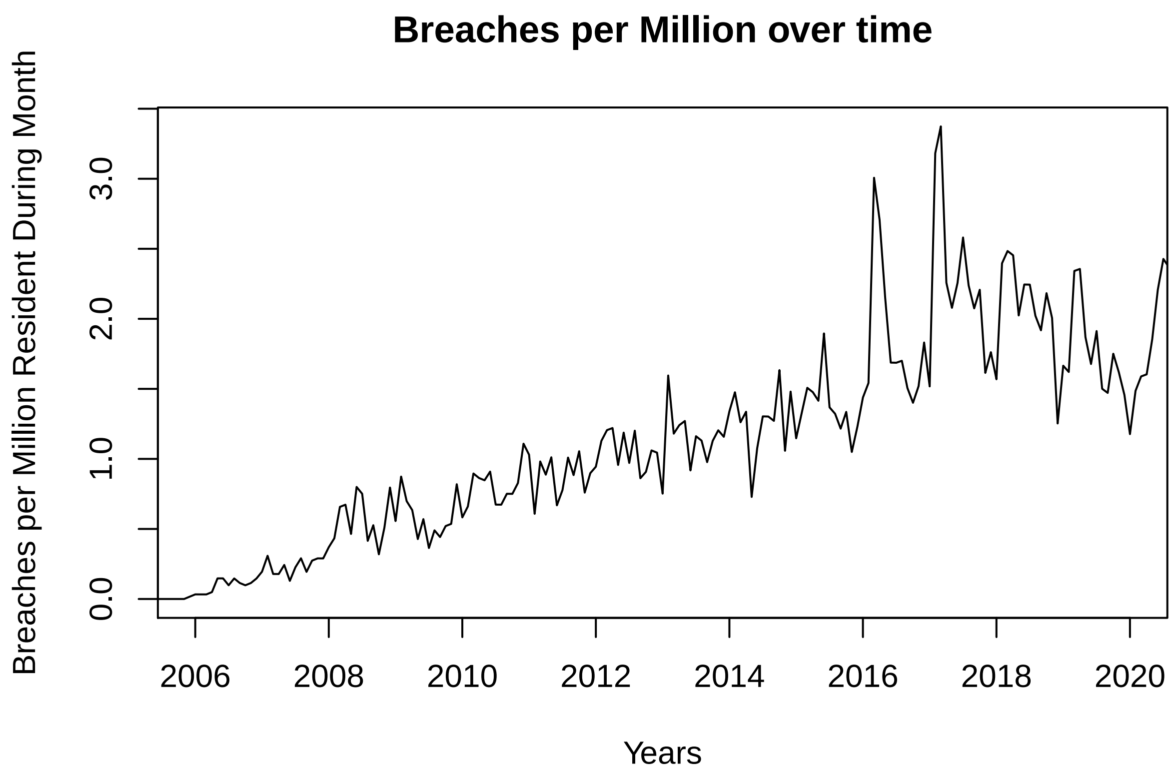
```
##  [91] 0.27398260 0.28982475 0.28955079 0.36963216 0.43350662 0.65766733
##  [97] 0.67307357 0.46430402 0.79977171 0.75107945 0.41513840 0.52645996
## [103] 0.31879652 0.50964294 0.79564406 0.55648050 0.87373150 0.69839593
## [109] 0.63437057 0.42783975 0.56997328 0.36384363 0.48999178 0.44220697
## [115] 0.52074156 0.53607837 0.81920772 0.58241704 0.66057719 0.89575938
## [121] 0.86361809 0.84728757 0.90937569 0.67369344 0.67323622 0.75100985
## [127] 0.75050085 0.82811677 1.10861316 1.02984476 0.60813332 0.98170568
## [133] 0.88761036 1.01150449 0.66869808 0.77703211 1.00945350 0.88461037
## [139] 1.05460660 0.75942042 0.89829501 0.94411706 1.12907834 1.20559440
## [145] 1.22022335 0.95699512 1.18772194 0.97111532 1.20150671 0.86202764
## [151] 0.90758421 1.06068391 1.04459507 0.75220724 1.59542858 1.18042265
## [157] 1.24089438 1.27066501 0.91792592 1.16191286 1.13052397 0.97704903
## [163] 1.12890303 1.20431687 1.15775390 1.33959815 1.47554700 1.26167952
## [169] 1.33672942 0.72860505 1.07696007 1.30355763 1.30266593 1.27150162
## [175] 1.63367107 1.05813802 1.48038264 1.14726921 1.32751201 1.50750938
## [181] 1.47635406 1.41513200 1.89559031 1.36810795 1.32219464 1.21627460
## [187] 1.33558241 1.04981547 1.22903195 1.43798749 1.54189863 3.00711150
## [193] 2.70624495 2.15172230 1.68747571 1.68644841 1.70048915 1.50579166
## [199] 1.40070281 1.51912344 1.83093473 1.51754966 3.18223024 3.37379655
## [205] 2.25793768 2.07860435 2.25560448 2.58074020 2.23874963 2.07486683
## [211] 2.20740564 1.61419542 1.76161245 1.56856726 2.39632780 2.48413091
## [217] 2.45362019 2.02420246 2.24497315 2.24411647 2.02206173 1.91817966
## [223] 2.18313084 2.00552981 1.25308729 1.66537897 1.62068866 2.34194307
## [229] 2.35597970 1.86950945 1.67764946 1.91254725 1.50061399 1.47119019
## [235] 1.75071633 1.61830921 1.45647829 1.17695215 1.48590209 1.58888541
## [241] 1.60359731 1.85369964 2.20678529 2.42746381 2.36861621 0.76501890
## [247] 0.01471190 0.00000000 0.00000000
```

```r
treatment_tsM <- ts(comb_ts$treatpermil, frequency = 12, start = c(2000,4))
treatment_tsM
```

```
##             Jan        Feb        Mar        Apr        May        Jun
## 2000                                  0.00000000 0.00000000 0.00000000
## 2001 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 2002 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 2003 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 2004 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 2005 0.00000000 0.01651349 0.00000000 0.00000000 0.00000000 0.00000000
## 2006 0.03276818 0.03274383 0.03271952 0.04904285 0.14701946 0.14691053
## 2007 0.19483902 0.30825977 0.17833013 0.17819429 0.24280725 0.12939871
## 2008 0.36963216 0.43350662 0.65766733 0.67307357 0.46430402 0.79977171
## 2009 0.55648050 0.87373150 0.69839593 0.63437057 0.42783975 0.56997328
## 2010 0.58241704 0.66057719 0.89575938 0.86361809 0.84728757 0.90937569
## 2011 1.02984476 0.60813332 0.98170568 0.88761036 1.01150449 0.66869808
## 2012 0.94411706 1.12907834 1.20559440 1.22022335 0.95699512 1.18772194
## 2013 0.75220724 1.59542858 1.18042265 1.24089438 1.27066501 0.91792592
## 2014 1.33959815 1.47554700 1.26167952 1.33672942 0.72860505 1.07696007
## 2015 1.14726921 1.32751201 1.50750938 1.47635406 1.41513200 1.89559031
## 2016 1.43798749 1.54189863 3.00711150 2.70624495 2.15172230 1.68747571
## 2017 1.51754966 3.18223024 3.37379655 2.25793768 2.07860435 2.25560448
## 2018 1.56856726 2.39632780 2.48413091 2.45362019 2.02420246 2.24497315
## 2019 1.66537897 1.62068866 2.34194307 2.35597970 1.86950945 1.67764946
## 2020 1.17695215 1.48590209 1.58888541 1.60359731 1.85369964 2.20678529
##             Jul        Aug        Sep        Oct        Nov        Dec
## 2000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
```

```
## 2001 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 2002 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 2003 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 2004 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 2005 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.01639629
## 2006 0.09786784 0.14668925 0.11400426 0.09764316 0.11382991 0.14624092
## 2007 0.22627563 0.29064977 0.19358283 0.27398260 0.28982475 0.28955079
## 2008 0.75107945 0.41513840 0.52645996 0.31879652 0.50964294 0.79564406
## 2009 0.36384363 0.48999178 0.44220697 0.52074156 0.53607837 0.81920772
## 2010 0.67369344 0.67323622 0.75100985 0.75050085 0.82811677 1.10861316
## 2011 0.77703211 1.00945350 0.88461037 1.05460660 0.75942042 0.89829501
## 2012 0.97111532 1.20150671 0.86202764 0.90758421 1.06068391 1.04459507
## 2013 1.16191286 1.13052397 0.97704903 1.12890303 1.20431687 1.15775390
## 2014 1.30355763 1.30266593 1.27150162 1.63367107 1.05813802 1.48038264
## 2015 1.36810795 1.32219464 1.21627460 1.33558241 1.04981547 1.22903195
## 2016 1.68644841 1.70048915 1.50579166 1.40070281 1.51912344 1.83093473
## 2017 2.58074020 2.23874963 2.07486683 2.20740564 1.61419542 1.76161245
## 2018 2.24411647 2.02206173 1.91817966 2.18313084 2.00552981 1.25308729
## 2019 1.91254725 1.50061399 1.47119019 1.75071633 1.61830921 1.45647829
## 2020 2.42746381 2.36861621 0.76501890 0.01471190 0.00000000 0.00000000
```

```
plot.ts(treatment_tsM, main = "Breaches per Million over time", xlim=c(2006,2020), xlab = "Years", ylab
```



### Identifying and subsetting relevant dates

```r
#June 26, 2012

treatment_start <- as.Date("06/26/2012", "%m/%d/%Y")+5 # Legislation H.B. 4144 becomes effective
treatment_start<- format(as.Date(as.character(treatment_start), origin = "1970-01-01"), "%Y/%m")

treatment_end <- as.Date("07/1/2012", "%m/%d/%Y") # post 6 months after enforcement
treatment_end<- format(as.Date(as.character(treatment_end), origin = "1970-01-01"), "%Y/%m")


pretreat <- comb_ts[(which(comb_ts$yearmonth==treatment_start)-6):(which(comb_ts$yearmonth==treatment_s
pretreat$type <- "pretest"

which(comb_ts$yearmonth==treatment_end)
```

```
## [1] 148
```

```r
posttreat <- comb_ts[(which(comb_ts$yearmonth==treatment_end)+1):(which(comb_ts$yearmonth==treatment_en
posttreat$type <- "posttest"

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

```
## [1] -0.1358542
```

```r
mean(posttreat$controlpermil) - mean(pretreat$controlpermil)
```

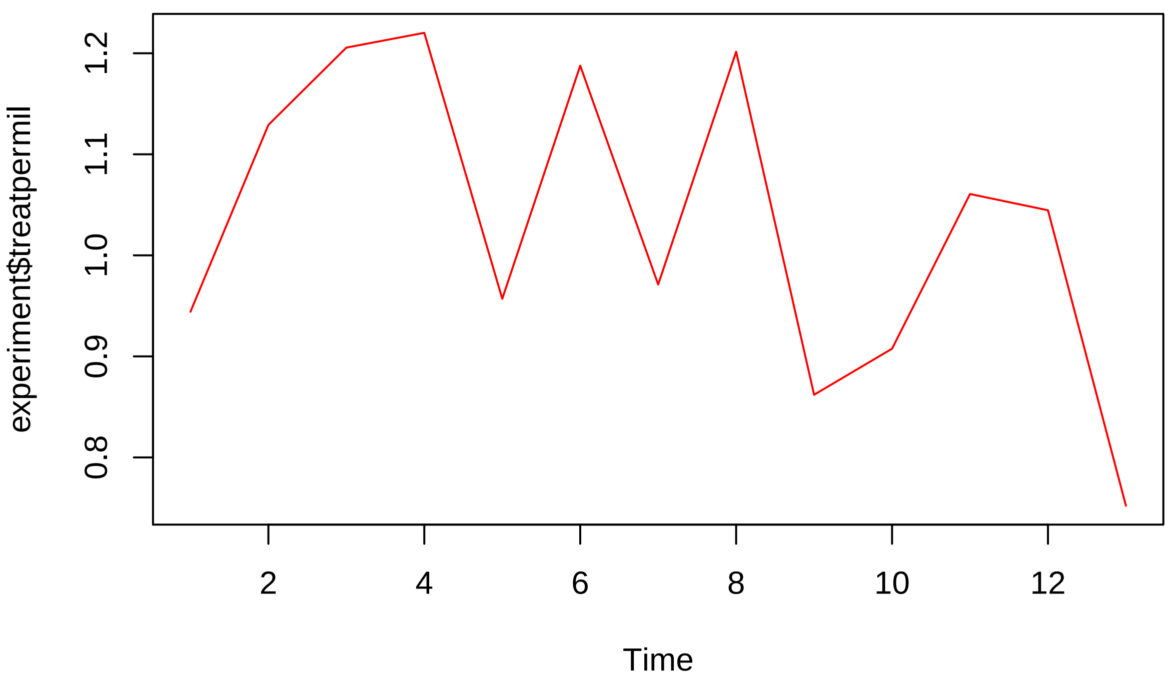
```
## Warning in mean.default(posttreat$controlpermil): argument is not numeric or
## logical: returning NA

## Warning in mean.default(pretreat$controlpermil): argument is not numeric or
## logical: returning NA

## [1] NA
```

```r
treatment_range <- comb_ts[(which(comb_ts$yearmonth==treatment_start)):(which(comb_ts$yearmonth==treatm
treatment_range$type <- "test"

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

```
##     yearmonth frequency California  Hawaii    Iowa Massachusetts New Hampshire
## 142   2012/01        61   37793584 1387066 3071263       6638294       1322217
## 143   2012/02        73   37819454 1388356 3072084       6642412       1322553
## 144   2012/03        78   37845323 1389646 3072905       6646531       1322889
## 145   2012/04        79   37871192 1390935 3073726       6650650       1323224
## 146   2012/05        62   37897062 1392225 3074548       6654768       1323560
## 147   2012/06        77   37922931 1393514 3075369       6658886       1323896
## 148   2012/07        63   37948800 1394804 3076190       6663005       1324232
## 149   2012/08        78   37974799 1395924 3077591       6667198       1324431
## 150   2012/09        56   38000798 1397044 3078991       6671390       1324630
## 151   2012/10        59   38026797 1398164 3080392       6675582       1324830
## 152   2012/11        69   38052796 1399284 3081792       6679775       1325029
## 153   2012/12        68   38078795 1400404 3083193       6683968       1325228
## 154   2013/01        49   38104794 1401524 3084594       6688160       1325427
##     North Carolina South Carolina sevenpop treatpermil     type
```

```
## 142        9703534        4694674 64610632   0.9441171  pretest
## 143        9711191        4698454 64654504   1.1290783  pretest
## 144        9718848        4702234 64698376   1.2055944  pretest
## 145        9726505        4706014 64742246   1.2202233  pretest
## 146        9734162        4709794 64786119   0.9569951  pretest
## 147        9741819        4713574 64829989   1.1877219  pretest
## 148        9749476        4717354 64873861   0.9711153     test
## 149        9757298        4721248 64918489   1.2015067 posttest
## 150        9765119        4725142 64963114   0.8620276 posttest
## 151        9772941        4729036 65007742   0.9075842 posttest
## 152        9780763        4732929 65052368   1.0606839 posttest
## 153        9788584        4736823 65096995   1.0445951 posttest
## 154        9796406        4740717 65141622   0.7522072 posttest
```

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




```
# Look at Raw Frequency Counts

treatment_ts <- ts(experiment$frequency, frequency = 12, start = c(2011,7))
treatment_ts
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2011                          61  73  78  79  62  77
## 2012  63  78  56  59  69  68  49
```

```
# Look at Treatment and Control per Million

treatment_tsM <- ts(experiment$treatpermil, frequency = 12, start = c(2011,7))
```

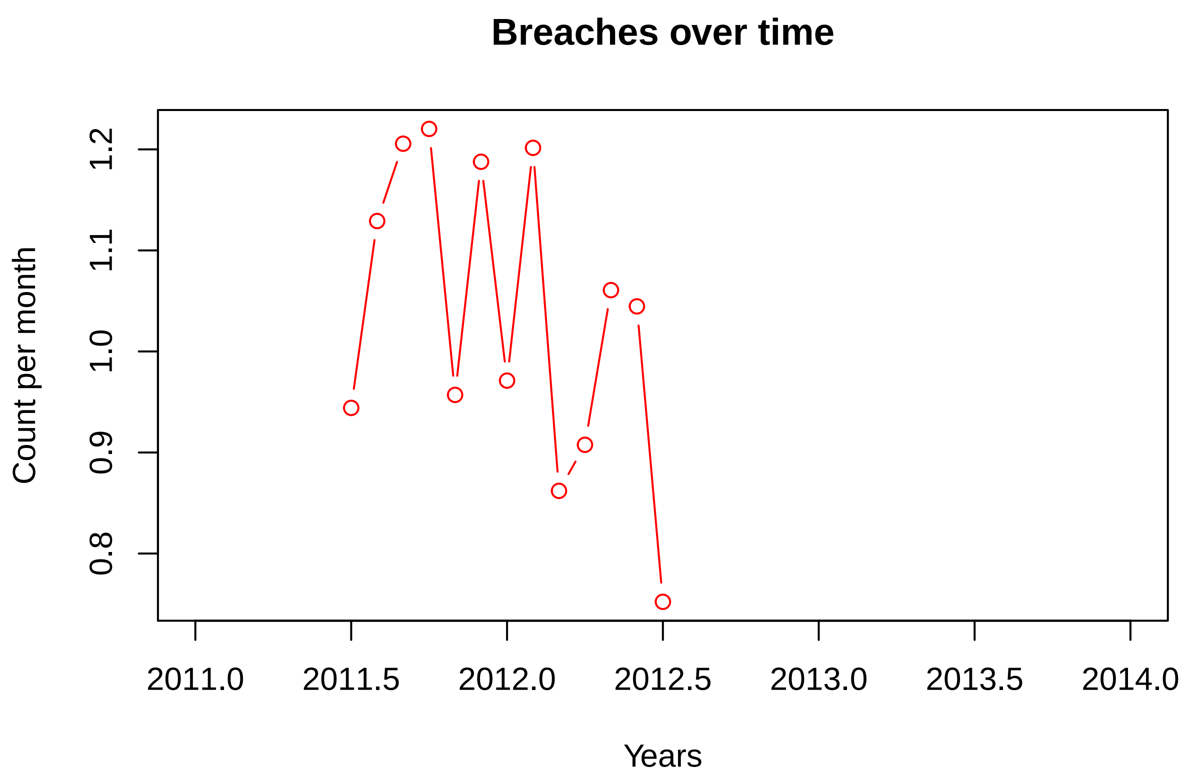
```r
mean(treatment_tsM)
```

```
## [1] 1.034112
```

```r
sd(treatment_tsM)
```

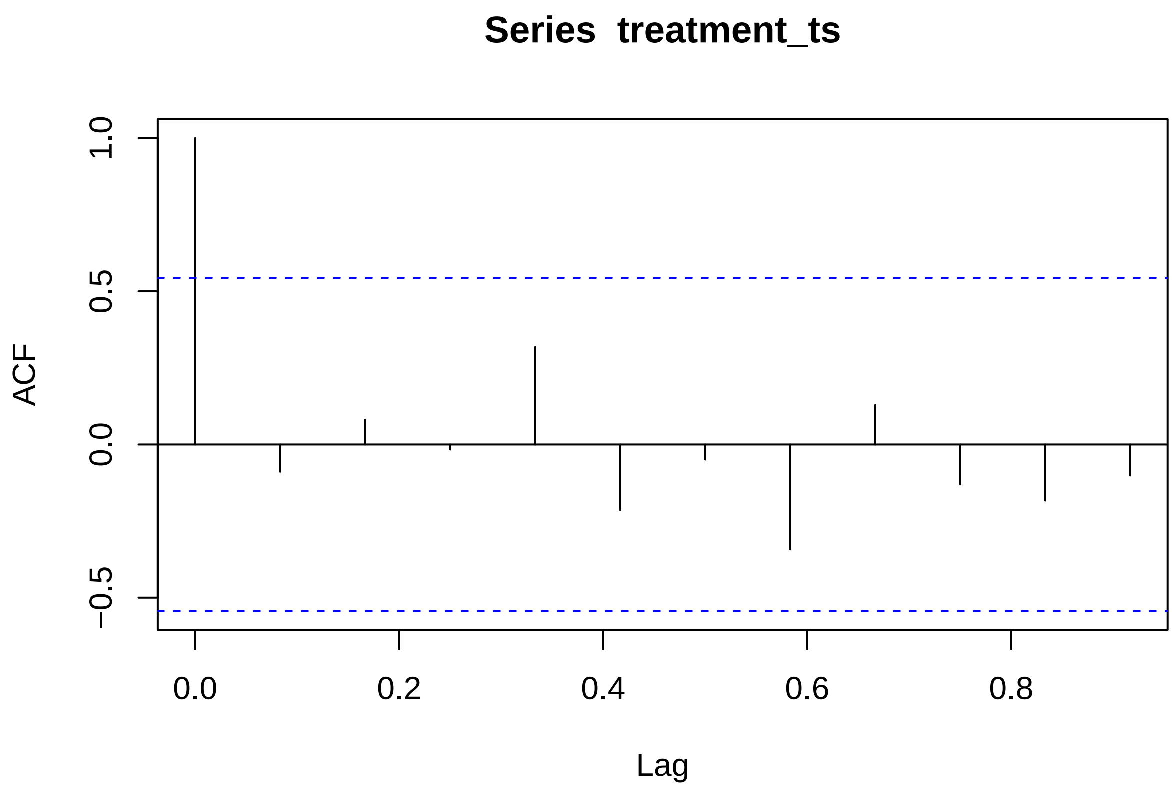
```
## [1] 0.1496113
```

```r
ts.plot(treatment_tsM, main = "Breaches over time", xlim=c(2011,2014),
                    gpars = list(col = c("red")), type = "b", xlab = "Years", ylab = "Count per month")
```

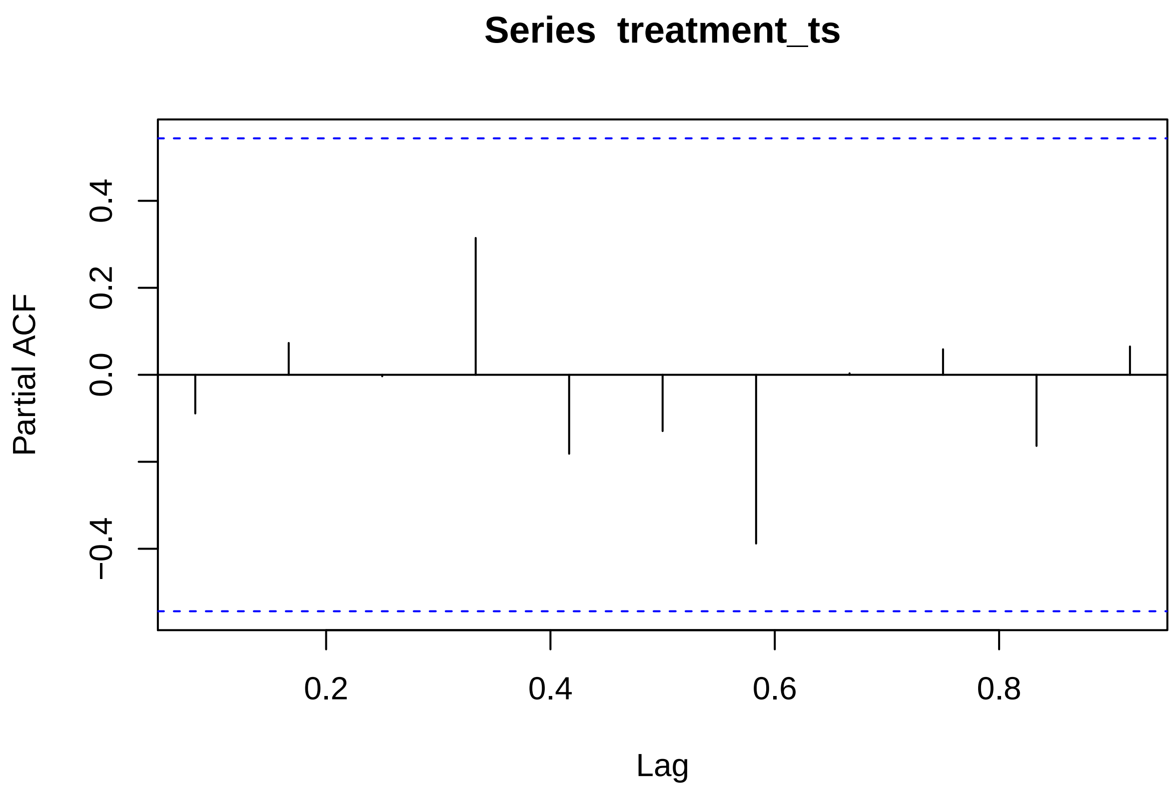



### Run Statistical Tests on Time Series for Stationarity

```r
# source of statistical tests http://r-statistics.co/Time-Series-Analysis-With-R.html

acftreatmentMA <- acf(treatment_ts) # autocorrelation (i.e. a Time Series with lags of itself)
```




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

# png(here::here("Output","acttreatmentMA.png"))
# plot(acftreatmentMA)

pacftreatment <- pacf(treatment_ts)  # partial autocorrelation (i.e. correlation of the time series wit
```




```r
# png(here::here("Output","pacftreatmentNH.png"))
# plot(pacftreatmentMA)

treatment_ts
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2011                             61  73  78  79  62  77
## 2012  63  78  56  59  69  68  49
```

```r
# adf test is an Augmented Dickey-Fuller Test
adf.test(treatment_ts) # p-value < 0.05 indicates the TS is stationary
```

```
## Warning in adf.test(treatment_ts): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  treatment_ts
## Dickey-Fuller = -5.039, Lag order = 2, p-value = 0.01
## alternative hypothesis: stationary
```

```r
kpss.test(treatment_ts) # Kwiatkowski-Phillips-Schmidt-Shin (KPSS) testz
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  treatment_ts
## KPSS Level = 0.35962, Truncation lag parameter = 2, p-value = 0.09456
```

```
# 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",]
quasiexp
```

```
##     yearmonth frequency California  Hawaii    Iowa Massachusetts New Hampshire
## 142   2012/01        61   37793584 1387066 3071263       6638294       1322217
## 143   2012/02        73   37819454 1388356 3072084       6642412       1322553
## 144   2012/03        78   37845323 1389646 3072905       6646531       1322889
## 145   2012/04        79   37871192 1390935 3073726       6650650       1323224
## 146   2012/05        62   37897062 1392225 3074548       6654768       1323560
## 147   2012/06        77   37922931 1393514 3075369       6658886       1323896
## 149   2012/08        78   37974799 1395924 3077591       6667198       1324431
## 150   2012/09        56   38000798 1397044 3078991       6671390       1324630
## 151   2012/10        59   38026797 1398164 3080392       6675582       1324830
## 152   2012/11        69   38052796 1399284 3081792       6679775       1325029
## 153   2012/12        68   38078795 1400404 3083193       6683968       1325228
## 154   2013/01        49   38104794 1401524 3084594       6688160       1325427
##     North Carolina South Carolina sevenpop treatpermil     type
## 142        9703534        4694674 64610632   0.9441171  pretest
## 143        9711191        4698454 64654504   1.1290783  pretest
## 144        9718848        4702234 64698376   1.2055944  pretest
## 145        9726505        4706014 64742246   1.2202233  pretest
## 146        9734162        4709794 64786119   0.9569951  pretest
## 147        9741819        4713574 64829989   1.1877219  pretest
## 149        9757298        4721248 64918489   1.2015067 posttest
## 150        9765119        4725142 64963114   0.8620276 posttest
## 151        9772941        4729036 65007742   0.9075842 posttest
## 152        9780763        4732929 65052368   1.0606839 posttest
## 153        9788584        4736823 65096995   1.0445951 posttest
## 154        9796406        4740717 65141622   0.7522072 posttest
```

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

```
##         V1                V2 time z
## 1  2012/01 0.944117061105361    1 0
## 2  2012/02  1.12907833922908    2 0
## 3  2012/03  1.20559440317327    3 0
## 4  2012/04  1.22022334535629    4 0
## 5  2012/05  0.95699512421789    5 0
## 6  2012/06   1.1877219352914    6 0
## 7  2012/08  1.20150670789642    7 1
## 8  2012/09  0.86202764233254    8 2
## 9  2012/10 0.907584207431786    9 3
## 10 2012/11  1.06068390930827   10 4
## 11 2012/12  1.04459506925012   11 5
```

```
## 12 2013/01 0.752207244701398   12 6
```

```r
AppendITS <- treatment
names(AppendITS) <- c("yearmonth","incident_permil","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 time z
## 1    2012/01       0.9441171    1 0
## 2    2012/02       1.1290783    2 0
## 3    2012/03       1.2055944    3 0
## 4    2012/04       1.2202233    4 0
## 5    2012/05       0.9569951    5 0
## 6    2012/06       1.1877219    6 0
## 7    2012/08       1.2015067    7 1
## 8    2012/09       0.8620276    8 2
## 9    2012/10       0.9075842    9 3
## 10   2012/11       1.0606839   10 4
## 11   2012/12       1.0445951   11 5
## 12   2013/01       0.7522072   12 6
```

```r
factor_cols <- c("time","z")

sapply(AppendITS, class)
```

```
##       yearmonth incident_permil            time               z
##     "character"       "numeric"       "numeric"       "numeric"
```

```r
regTest <- lm(incident_permil ~ time + z + z*time, AppendITS)
summary(regTest)
```

```
##
## Call:
## lm(formula = incident_permil ~ time + z + z * time, data = AppendITS)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.17855 -0.11301  0.04307  0.10723  0.14204
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.038719   0.132480   7.841 5.05e-05 ***
## time         0.019152   0.032917   0.582    0.577
## z           -0.092520   0.189044  -0.489    0.638
## time:z       0.002105   0.014483   0.145    0.888
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1448 on 8 degrees of freedom
## Multiple R-squared:  0.3652, Adjusted R-squared:  0.1271
## F-statistic: 1.534 on 3 and 8 DF,  p-value: 0.2789
```