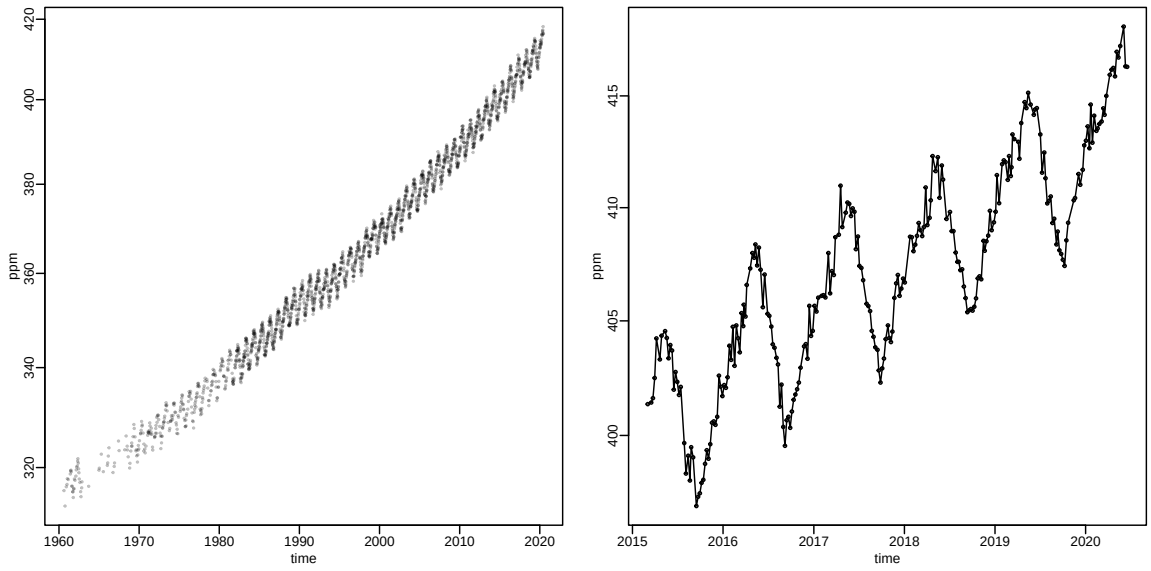
Homework 2, Smoothing

STA442 Methods of Applied Statistics

Due 29 Oct 2021

1 CO2

Figure 1 shows atmoshperic Carbon Dioxide concentrations from an observatory in Haiwaii, made available by the Scripps CO_2 Program at scrippsco2.ucsd.edu. The figure was produced with code in the appendix.



(a) all (b) recent

Figure 1: CO2 at Mauna Loa Observatory, Hawaii

Write a short consulting report (roughly a page of writing) discussing if the CO2 data appears to be impacted by the following events:

- the fall of the Berlin wall in November 1989 years ago, preceding a dramatic fall in industrial production in the Soviet Union and Eastern Europe;
- the global lockdown during the COVID-19 pandemic starting in February 2020, shutting down much of the global economy.

You should

- explain fully the model you are using and why you have chosen to use it
- make your graphs look nice
- at a minimum, plot the estimated smoothed trend of CO2 and discuss whether it appears shallower or steeper after the events listed above.
- visual investigation is sufficient, you aren't expected for formally test for effects.

2 Death

Daily cause-specific mortality counts by province are available from Statistics Canada at

https://open.canada.ca/data/en/dataset/aed00edc-26ad-414c-8aa3-82212059ef8a

Consider the following four outcomes: Malignant neoplasms (cancers), Diseases of the heart (heart attacks), Influenza and pneumonia (the flu), and chronic lower respiratory diseases. The first two are hypothesized to have increased during the covid-19 lockdown in 2020 as susceptible individuals had less access to healthcare. The last two are hypothesized to have decreased, as masks, social distancing, and decreased air pollution made the population less susceptible to respiratory problems.

Your task is to quantify the excess (or deficit of) mortality in Ontario for each of the four types from March to November 2020. You should consider both total mortality for the period and whether there were particular months or weeks with excess deaths. The code in the appendix accomplishes this by fitting a model to the pre-COVID data and using the model to create an ensemble of 100 different forecasts for March-November under normal (no COVID) circumstances.

Some notes:

- the data from December 2020 onwards looks incomplete (it has a suspicious drop in deaths), so I suggest excluding it
- write a short report explaining your model clearly and concluding if and when there was excess mortality from either of the four types listed. Roughly two pages of writing, excluding tables and figures
- notice that the death counts are all multiples of 5, Stats Can rounds the data to preserve confidentiality. Ignore this (pretend the data are exact and not rounded), although there are ways of dealing with the rounding.

Appendix

CO2

```
cUrl = paste0("http://scrippsco2.ucsd.edu/assets/data/atmospheric/",
  "stations/flask_co2/daily/daily_flask_co2_mlo.csv")
cFile = basename(cUrl)
if (!file.exists(cFile)) download.file(cUrl, cFile)
co2s = read.table(cFile, header = FALSE, sep = ",",
  skip = 69, stringsAsFactors = FALSE, col.names = c("day",
    "time", "junk1", "junk2", "Nflasks", "quality",
    "co2"))
co2s$date = strptime(paste(co2s$day, co2s$time), format = "%Y-%m-%d %H:%M",
  tz = "UTC")
# remove low-quality measurements
co2s = co2s[co2s$quality == 0, ]

plot(co2s$date, co2s$co2, log = "y", cex = 0.3, col = "#00000040",
  xlab = "time", ylab = "ppm")
plot(co2s[co2s$date > ISOdate(2015, 3, 1, tz = "UTC"),
  c("date", "co2")], log = "y", type = "o", xlab = "time",
  ylab = "ppm", cex = 0.5)
```

The code below might prove useful.

```
co2s$day = as.Date(co2s$date)
toAdd = data.frame(day = seq(max(co2s$day) + 3, as.Date("2025/1/1"),
  by = "10 days"), co2 = NA)
co2ext = rbind(co2s[, colnames(toAdd)], toAdd)
timeOrigin = as.Date("2000/1/1")
```

```
co2ext$timeYears = round(as.numeric(co2ext$day - timeOrigin)/365.25,
  2)
co2ext$cos12 = cos(2 * pi * co2ext$timeYears)
co2ext$sin12 = sin(2 * pi * co2ext$timeYears)
co2ext$cos6 = cos(2 * 2 * pi * co2ext$timeYears)
co2ext$sin6 = sin(2 * 2 * pi * co2ext$timeYears)
allDays = seq(from = min(co2ext$day), to = max(co2ext$day),
  by = "1 day")
co2ext$dayInt = as.integer(co2ext$day)
```

```
library('INLA', verbose=FALSE)
# disable some error checking in INLA
mm = get("inla.models", INLA:::inla.get.inlaEnv())
if(class(mm) == 'function') mm = mm()
mm$latent$rw2$min.diff = NULL
assign("inla.models", mm, INLA:::inla.get.inlaEnv())

co2res = inla(co2 ~ sin12 + cos12 + sin6 + cos6 +
  f(dayInt, model = 'rw2', values = as.integer(allDays),
    prior='pc.prec', param = c(0.001, 0.5), scale.model=FALSE),
  data = co2ext, family='gaussian',
  control.family = list(hyper=list(prec=list(
    prior='pc.prec', param=c(1, 0.5)))),
  # add this line if your computer has trouble
  control.inla = list(strategy='gaussian'),
  control.predictor = list(compute=TRUE, link=1),
  control.compute = list(config=TRUE),
  control.mode = list(theta = c(1, 20), restart=TRUE),
  verbose=TRUE)
qCols = c('0.5quant','0.025quant','0.975quant')
1/sqrt(co2res$summary.hyperpar[,qCols])
```

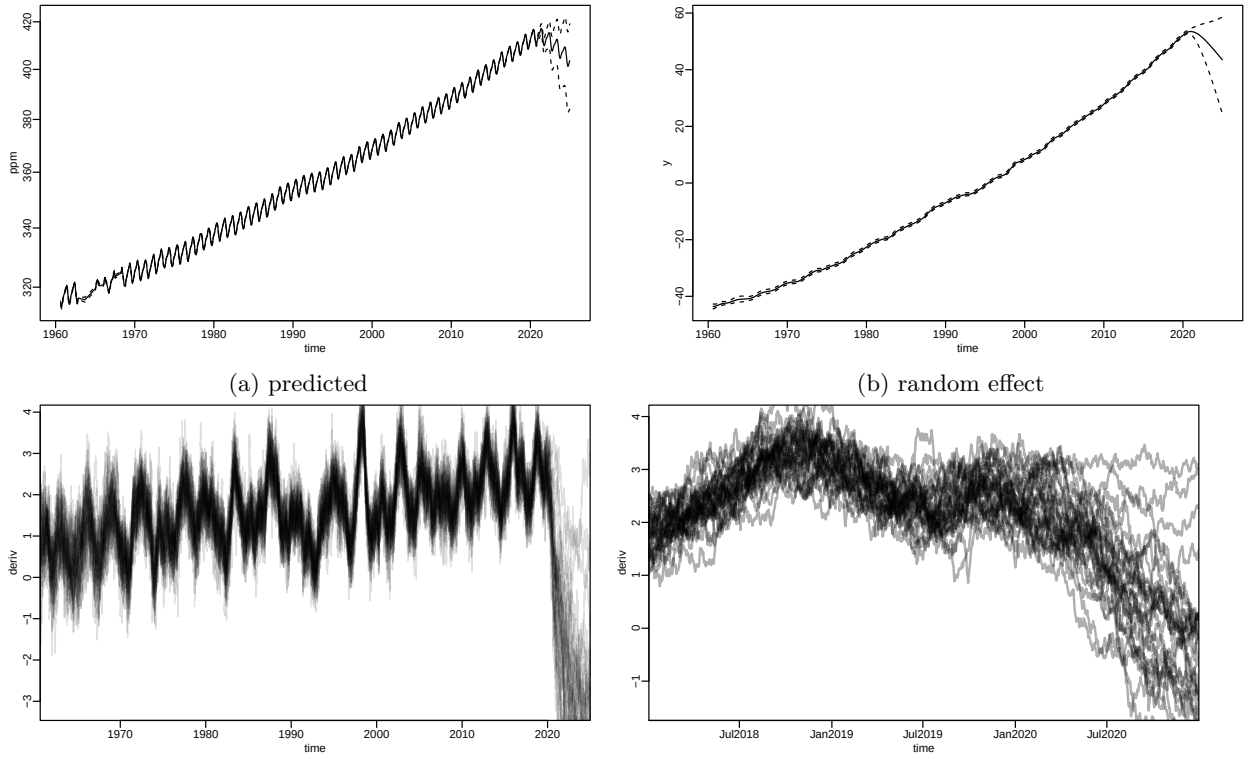
```
                                          0.5quant   0.025quant   0.975quant
Precision for the Gaussian observations 0.6081403116 0.6363325915 0.5885084062
Precision for dayInt                    0.0002118331 0.0002698735 0.0001649478
```

```
# source('https://bioconductor.org/biocLite.R')
# biocLite('Biobase')
```

```
sampleList = INLA::inla.posterior.sample(30, co2res,
  selection = list(dayInt = 0))
sampleMean = do.call(cbind, Biobase::subListExtract(sampleList,
  "latent"))
sampleDeriv = apply(sampleMean, 2, diff) * 365.25
```

```
matplot(co2ext$day, co2res$summary.fitted.values[,
  qCols], type = "l", col = "black", lty = c(1, 2,
  2), log = "y", xlab = "time", ylab = "ppm")
matplot(allDays, co2res$summary.random$dayInt[, qCols],
  type = "l", col = "black", lty = c(1, 2, 2), xlab = "time",
  ylab = "y")
matplot(allDays[-1], sampleDeriv, type = "l", lty = 1,
  xaxs = "i", col = "#00000020", xlab = "time", ylab = "deriv",
  ylim = quantile(sampleDeriv, c(0.025, 0.995)))
forX = as.Date(c("2018/1/1", "2021/1/1"))
forX = seq(forX[1], forX[2], by = "6 months")
```

```
toPlot = which(allDays > min(forX) & allDays < max(forX))
matplot(allDays[toPlot], sampleDeriv[toPlot, ], type = "l",
  lty = 1, lwd = 2, xaxs = "i", col = "#00000050",
  xlab = "time", ylab = "deriv", xaxt = "n", ylim = quantile(sampleDeriv[toPlot,
    ], c(0.01, 0.995)))
axis(1, as.numeric(forX), format(forX, "%b%Y"))
```



(a) predicted

(b) random effect

(c) derivative

(d) derivative, detail

Figure 2: INLA results

Covid

```
deadFile = Pmisc::downloadIfOld("https://www150.statcan.gc.ca/n1/tbl/csv/13100810-eng.zip",
  path = "../data")
```

```
Loading required namespace: R.utils
```

```
(deadFileCsv = deadFile[which.max(file.info(deadFile)$size)])
```

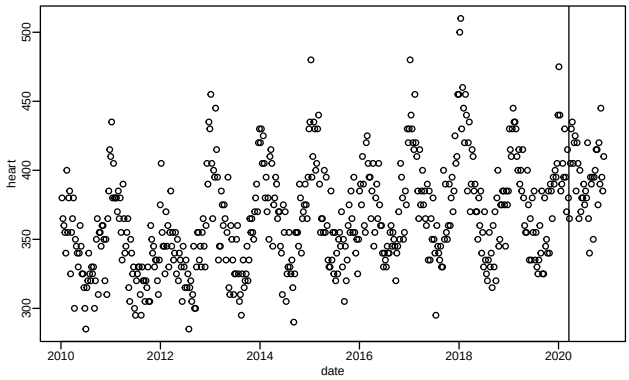
```
[1] "../data/13100810.csv"
```

```
x = read.csv(deadFileCsv)
x[1:2, ]
```

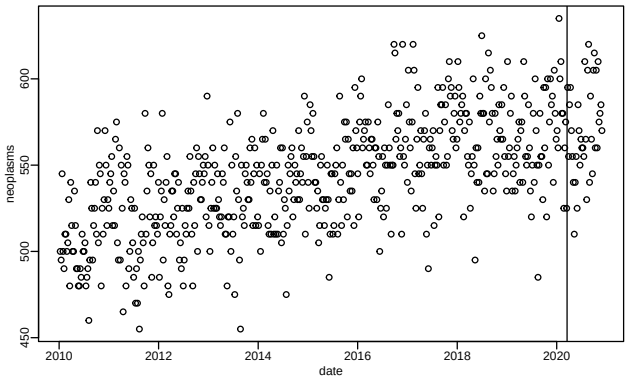
```
    REF_DATE                          GEO          DGUID
1 2010-01-09 Canada, place of occurrence 2016A000011124
2 2010-01-09 Canada, place of occurrence 2016A000011124
              Cause.of.death..ICD.10..  Characteristics    UOM UOM_ID
1 Total, all causes of death [A00-Y89] Number of deaths Number    223
2        Malignant neoplasms [C00-C97] Number of deaths Number    223
  SCALAR_FACTOR SCALAR_ID      VECTOR COORDINATE VALUE STATUS SYMBOL TERMINATED
```

```
1         units         0 v1234858180      1.1.1  4955            NA         NA
2         units         0 v1234858181      1.2.1  1320            NA         NA
  DECIMALS
1        0
2        0
```

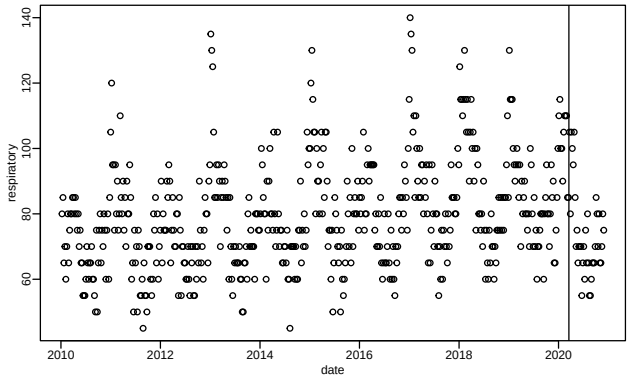
```
x$date = as.Date(as.character(x[[grep("DATE", names(x))]]))
x$province = gsub("[,].*", "", x$GEO)
# remove 2021 data, which appears incomplete
x = x[x$date < as.Date("2020/12/01") & x$province ==
  "Ontario", ]
for (D in c("heart", "neoplasms", "respiratory", "Influenza")) {
  plot(x[grep(D, x$Cause), c("date", "VALUE")], ylab = D)
  abline(v = as.Date("2020/03/17"))
}
```



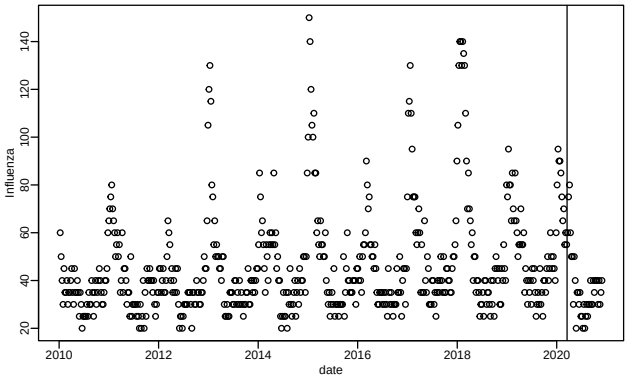
(a) 1



(b) 2



(c) 3



(d) 4

Figure 3: raw data

```
dateSeq = sort(unique(x$date))
table(diff(dateSeq))
```

```
  7
568
```

```
dateSeqInt = as.integer(dateSeq)
x$dateInt = x$dateIid = as.integer(x$date)
x$cos12 = cos(2 * pi * x$dateInt/365.25)
x$cos6 = cos(2 * pi * x$dateInt * 2/365.25)
```

```
x$sin12 = sin(2 * pi * x$dateInt/365.25)
x$sin6 = sin(2 * pi * x$dateInt * 2/365.25)
xInfluenza = x[grepl("Influenza", x$Cause) & x$province ==
  "Ontario", ]
xPreCovid = xInfluenza[xInfluenza$date < as.Date("2020/03/01"),
  ]
library("INLA")
```

```
Loading required package: Matrix
```

```
Loading required package: foreach
```

```
Loading required package: parallel
```

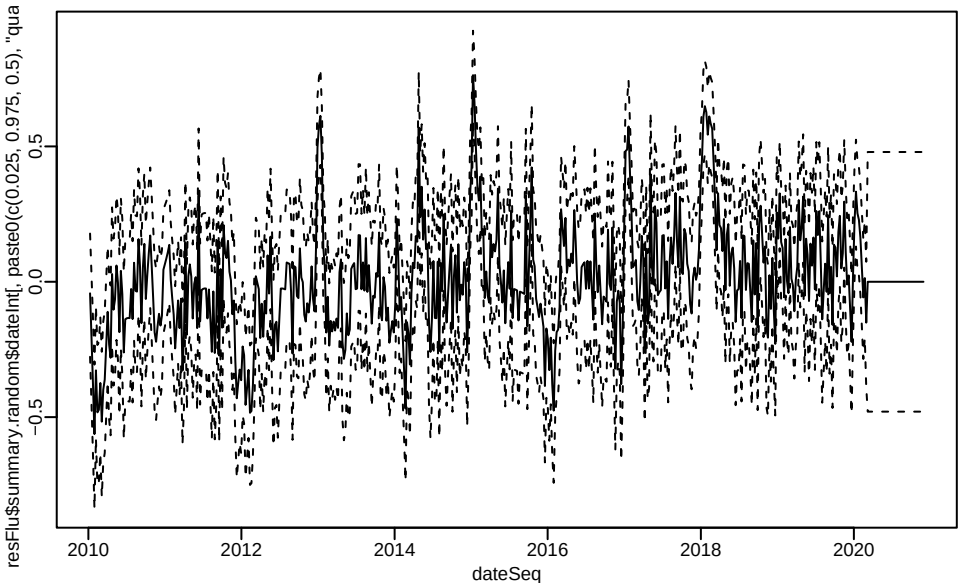
```
Loading required package: sp
```

```
This is INLA_21.02.23 built 2021-07-19 18:35:03 UTC.
 - See www.r-inla.org/contact-us for how to get help.
 - To enable PARDISO sparse library; see inla.pardiso()
 - Save 350.5Mb of storage running 'inla.prune()'
```

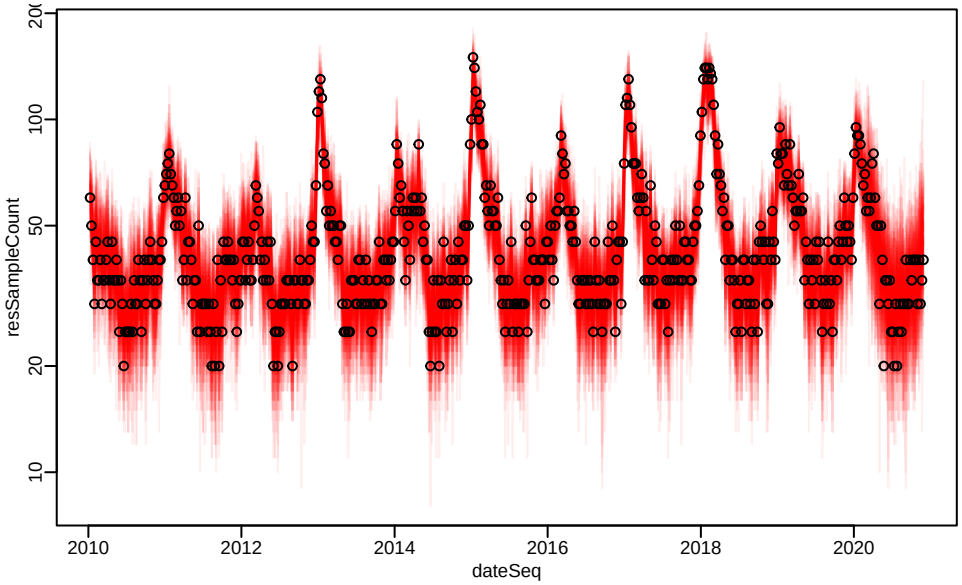
```
resFlu = inla(VALUE ~ cos12 + cos6 + sin12 + sin6 +
  f(dateInt, values = dateSeqInt, prior = "pc.prec",
    param = c(1, 0.5)) + f(dateIid, values = dateSeqInt),
  data = xPreCovid, family = "poisson", control.compute = list(config = TRUE))
matplot(dateSeq, resFlu$summary.random$dateInt[, paste0(c(0.025,
  0.975, 0.5), "quant")], type = "l", lty = c(2,
  2, 1), col = "black")
toPredict = cbind(`(Intercept):1` = 1, `cos12:1` = cos(2 *
  pi * dateSeqInt/365.25), `cos6:1` = cos(2 * pi *
  dateSeqInt * 2/365.25), `sin12:1` = sin(2 * pi *
  dateSeqInt/365.25), `sin6:1` = sin(2 * pi * dateSeqInt *
  2/365.25))
dateIntSeq = paste0("dateInt:", 1:length(dateSeqInt))
dateIidSeq = paste0("dateIid:", 1:length(dateSeqInt))
resSample = inla.posterior.sample(n = 100, resFlu)
resSampleFitted = lapply(resSample, function(xx) {
  toPredict %*% xx$latent[colnames(toPredict), ] +
    xx$latent[dateIntSeq, ] + xx$latent[dateIidSeq,
    ]
})
resSampleFitted = do.call(cbind, resSampleFitted)
resSampleLambda = exp(resSampleFitted)
resSampleCount = matrix(rpois(length(resSampleLambda),
  resSampleLambda), nrow(resSampleLambda), ncol(resSampleLambda))
matplot(dateSeq, resSampleCount, col = "#FF000010",
  type = "l", lty = 1, log = "y")
points(xInfluenza[, c("date", "VALUE")])
is2020 = dateSeq[dateSeq >= as.Date("2020/3/1")]
sample2020 = resSampleCount[match(is2020, dateSeq),
  ]
count2020 = xInfluenza[match(is2020, xInfluenza$date),
  "VALUE"]
excess2020 = sample2020 - count2020
matplot(is2020, excess2020, type = "l", lty = 1, col = "#0000FF10",
  ylim = quantile(excess2020, c(0.001, 0.999)))
matlines(is2020, t(apply(excess2020, 1, quantile, prob = c(0.05,
```

```
  0.95))), col = "black", lty = 2)
abline(h = 0)
quantile(apply(excess2020, 1, sum))
```

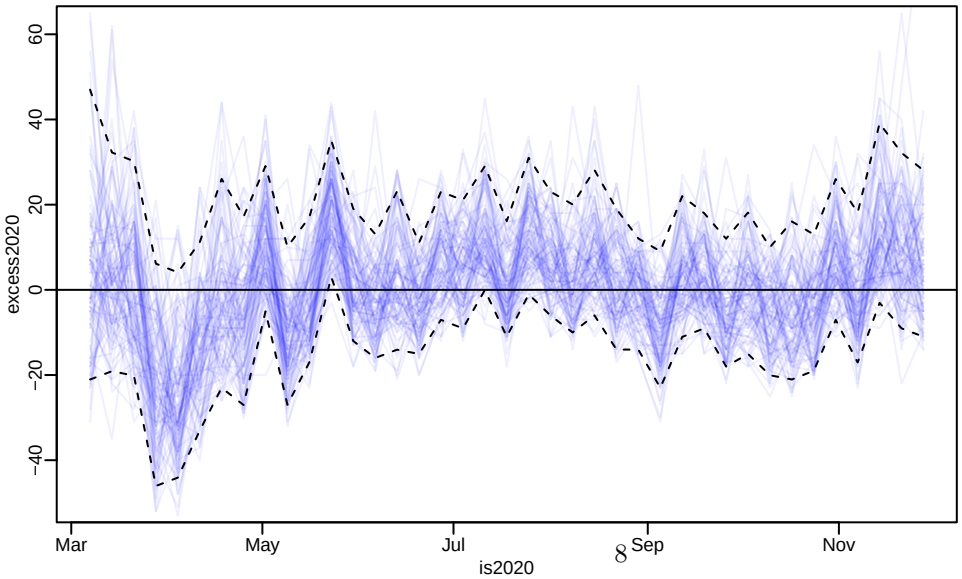
```
     0%     25%     50%     75%    100%
-2455.0  -315.5   192.0   684.5  1832.0
```



(a) 1



(b) 2



(c) 3

Figure 4: results