China Intervention Correlation Analysis

Kylie Ainslie

06 April, 2020

# Read in data

1. New daily confirmed cases from China CDC
2. Exante daily intracity movement data
3. Back calculated new infections

# case data  
new\_case\_dat <- readr::read\_csv("china\_new\_case\_data\_imported.csv")

## Parsed with column specification:  
## cols(  
## date\_confirm = col\_datetime(format = ""),  
## province = col\_character(),  
## new\_cases = col\_double(),  
## imported\_cases = col\_double(),  
## new\_cases\_wo\_imported = col\_double()  
## )

# wrangle into incidence data  
incidence\_dat <- new\_case\_dat %>%   
 group\_by(province) %>%  
 select(province, date\_confirm, new\_cases\_wo\_imported) %>%  
 mutate(date\_confirm = as.Date(date\_confirm),  
 new\_cases\_wo\_imported = ifelse(is.na(new\_cases\_wo\_imported) | new\_cases\_wo\_imported < 0,   
 0, new\_cases\_wo\_imported)) %>%  
 rename("dates" = "date\_confirm", "I" = "new\_cases\_wo\_imported")   
# movement data  
# movement\_dat <- readr::read\_csv("exante\_movement\_data.csv")[,-1]  
movement\_top6 <- readRDS("movement\_province\_level\_subset.rds")

case\_sums <- new\_case\_dat %>%   
 filter(date\_confirm < '2020-02-15') %>%   
 group\_by(province) %>%   
 summarise\_at(.vars = "new\_cases", .funs = sum)  
prop\_hubei <- case\_sums[case\_sums$province == "hubei",2] / sum(case\_sums$new\_cases)

# Estimate R\_t for confirmed case data

#####################################################  
### COVID19 spread with Chinese movement patterns ###  
#####################################################  
# 1. Estimating R\_t over time for each province in China using EpiEstim  
# 2. Determining most highly correlated lag time  
# 3. Determine rolling correlation using runCor (at weekly and biweekly resolution)  
  
### 1. Estimating R\_t with EpiEstim  
si\_mean = 6.48  
si\_std = 3.83  
# serial interval estimate used: mean = 6.48, sd = 3.83  
# from Neil's IBM model  
  
r <- incidence\_dat %>%  
 tidyr::nest(gg = -"province") %>%  
 mutate\_at("gg", purrr::map,function(x) estimate\_R(x, method="parametric\_si",  
 config = make\_config(list(mean\_si = si\_mean,   
 std\_si = si\_std)))  
 )

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.   
## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.   
## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.   
## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.   
## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.   
## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.   
## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.   
## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.   
## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

## Warning in estimate\_R\_func(incid = incid, method = method, si\_sample = si\_sample, : You're estimating R too early in the epidemic to get the desired  
## posterior CV.

## Default config will estimate R on weekly sliding windows.  
## To change this change the t\_start and t\_end arguments.

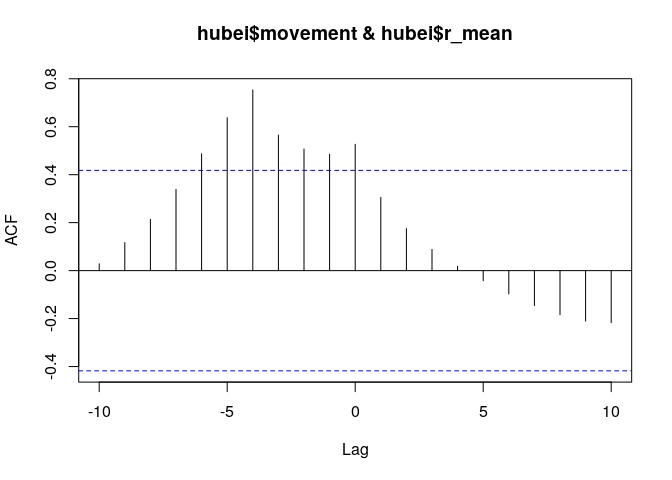
# loop through provinces to concatenate   
R\_t <- list()  
for(i in 1:length(r$province)){  
 R\_t[[i]] <- r$gg[[i]]$R %>%   
 mutate(province = r$province[i],  
 date\_start = r$gg[[i]]$dates[t\_start],  
 date\_end = r$gg[[i]]$dates[t\_start][t\_end]) %>%  
 rename(r\_mean = `Mean(R)`, r\_q2.5 = `Quantile.0.025(R)`,  
 r\_q97.5 = `Quantile.0.975(R)`,  
 r\_median = `Median(R)`) %>%  
 select(date\_start, date\_end, province, r\_mean, r\_q2.5,  
 r\_q97.5, r\_median)  
}  
  
r\_dat <- bind\_rows(R\_t)

# Combine R\_t estimates and pop-weighted avg movement data for top 6 provinces

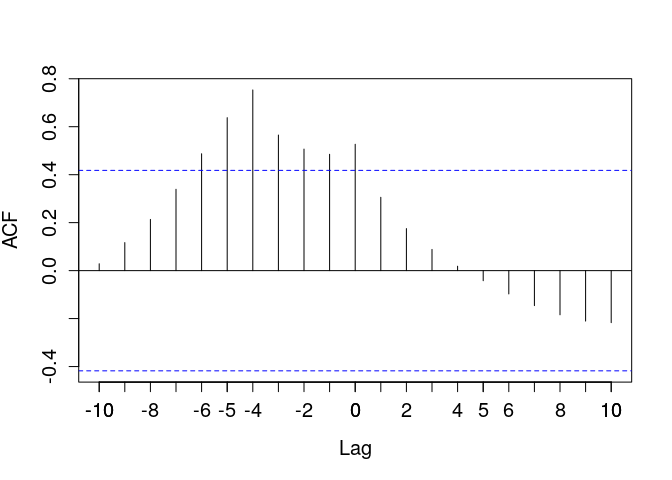
movement\_top6 <- movement\_top6 %>%   
 filter(year == 2020, province != "anhui") %>%  
 mutate(year = as.numeric(year))   
r\_top6 <- r\_dat %>% filter(province %in% c("hubei", "guangdong", "henan", "zhejiang",  
 "hunan", "beijing", "hong\_kong\_sar"))  
dat\_all\_top6 <- left\_join(r\_top6, movement\_top6, by=c("date\_end" = "date", "province")) %>%  
 select(-date\_start, -year, -month\_day, -id)

# Determine cross correlation between R\_t and movement

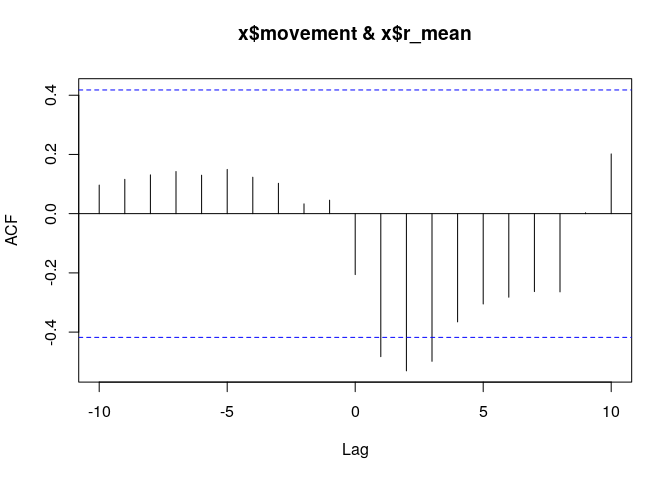
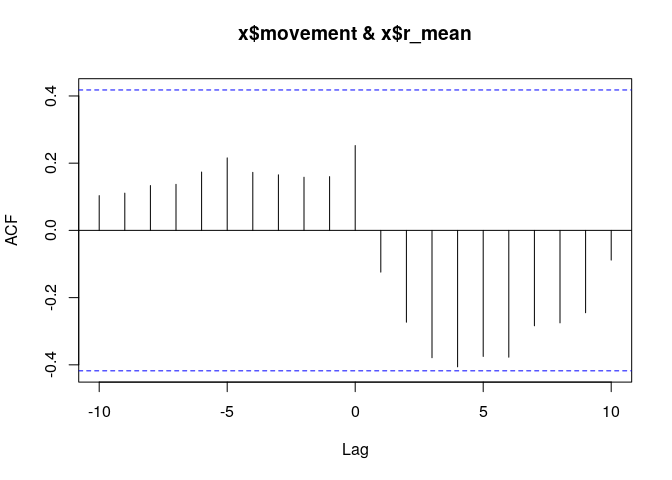
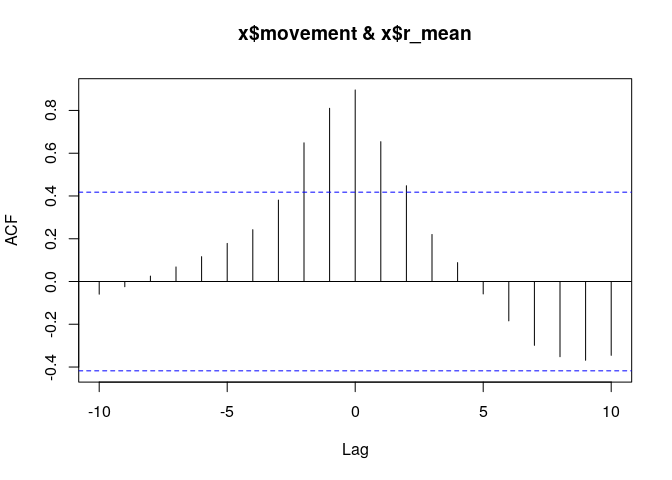
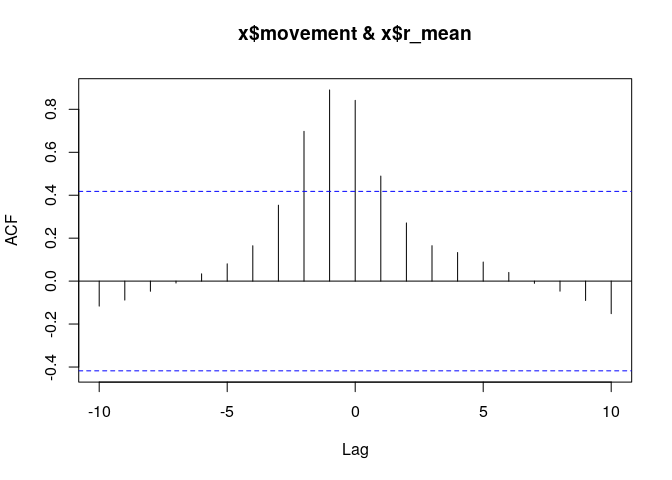
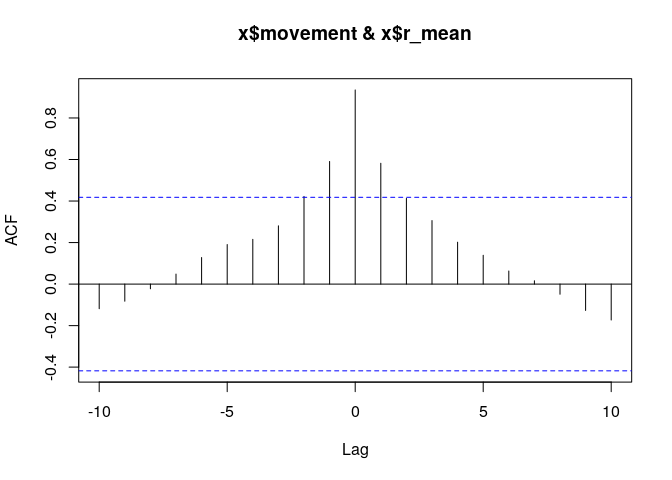
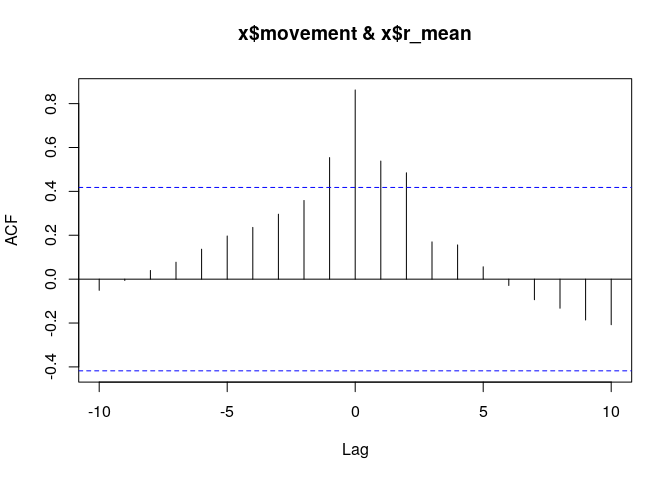
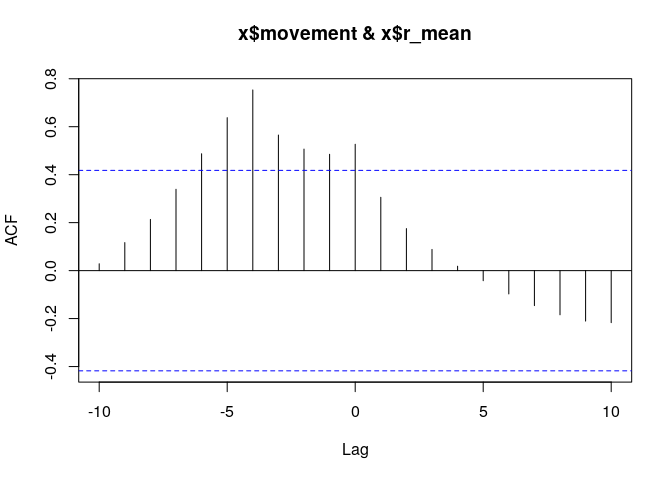
### cross correlation  
# restrict cross correlation to peak of epidemic  
dat\_top6\_peak <- dat\_all\_top6 %>% filter(date\_end < '2020-02-15')  
  
hubei <- dat\_top6\_peak %>% filter(province == "hubei", !is.na(movement))  
hubei\_corr <- ccf(hubei$movement, hubei$r\_mean, lag = 10)



lag\_hubei <- hubei\_corr$lag[which(hubei\_corr$acf == max(hubei\_corr$acf))]   
  
# plot hubei corr  
plot(hubei\_corr, main = "", cex.lab = 1.25, cex.axis = 1.25)  
axis(1, at=c(seq(-10:10)-11),cex.axis=1.25)



### cross correlation sensitivity analysis  
 # look at distribution of lag time with highest corr for all locations  
movement\_2020 <- movement\_top6 %>%   
 filter(year == 2020) %>%  
 mutate(year = as.numeric(year))   
  
rhos <- dat\_top6\_peak %>%  
 filter(!is.na(movement)) %>%  
 tidyr::nest(gg = -"province") %>%  
 mutate\_at("gg",purrr::map,  
 function(x) ccf(x$movement, x$r\_mean, lag.max = 10))



max\_lag <- numeric(nrow(rhos))  
for (p in 1:nrow(rhos)){  
 df <- tibble(province = rhos$province[p],lags = rhos$gg[[p]]$lag[,1,1],cc = rhos$gg[[p]]$acf[,1,1]) %>%  
 filter(lags < 1)  
 max\_lag[p] <- df$lags[which(df$cc == max(df$cc))]  
}  
lag\_df <- tibble(province = rhos$province, lag = max\_lag)  
p\_lags <- ggplot(lag\_df, aes(x=lag)) +   
 geom\_histogram(binwidth = 2) +  
 theme(legend.position = "none",  
 panel.grid.major = element\_blank(),  
 panel.grid.minor = element\_blank(),  
 panel.background = element\_blank()  
 )  
ggsave("figure\_S2.png", p\_lags,  
 width = 4, height = 4,  
 dpi = 500)

# create date\_lag variable using hubei only  
dat\_all\_top6a <- dat\_all\_top6 %>% mutate(date\_lag = date\_end + lag\_hubei) %>%  
 select(-date\_end, -movement)  
dat\_all\_top6b <- left\_join(dat\_all\_top6a, movement\_top6,   
 by = c("date\_lag" = "date", "province")) %>%  
 select(-id, -month\_day, -year)  
  
# create lag variable using lag for each region  
dat\_all\_top6c <- left\_join(dat\_all\_top6, lag\_df, by = "province") %>%   
 group\_by(province) %>%  
 mutate(date\_lag = date\_end + lag) %>%  
 select(-date\_end, -movement) %>%  
 filter(!is.na(date\_lag))  
dat\_all\_top6d <- left\_join(dat\_all\_top6c, movement\_top6,   
 by = c("date\_lag" = "date", "province")) %>%  
 select(-id, -month\_day, -year)

# Determine biweekly rolling correlation between R\_t and movement

### determine rolling biweekly correlation  
# rolling correlation - biweekly  
rolling\_corr\_bi <- dat\_all\_top6b %>%  
 group\_by(province) %>%  
 filter(!is.na(movement)) %>%  
 tq\_transmute\_xy(x = movement,   
 y = r\_mean,  
 mutate\_fun = runCor,  
 n = 14,  
 col\_rename = "rolling.corr.biweekly")  
  
# rolling correlation - Hubei vs R\_t in other provinces  
dat\_all\_top6b <- dat\_all\_top6b %>%  
 group\_by(province) %>%  
 mutate(movement\_hubei = dat\_all\_top6b[dat\_all\_top6b$province == 'hubei',]$movement)  
  
rolling\_corr\_hubei <- dat\_all\_top6b %>%  
 group\_by(province) %>%  
 filter(!is.na(movement\_hubei)) %>%  
 tq\_transmute\_xy(x = movement\_hubei,   
 y = r\_mean,  
 mutate\_fun = runCor,  
 n = 14,  
 col\_rename = "rolling.corr.hubei")  
  
rolling\_corr <- left\_join(rolling\_corr\_bi, rolling\_corr\_hubei,   
 by = c("province", "date\_lag")) %>%  
 mutate(date\_lag = as.Date(date\_lag))  
  
dat\_corr <- left\_join(dat\_all\_top6b, rolling\_corr, by = c("province", "date\_lag"))

### determine rolling biweekly correlation  
# rolling correlation - biweekly  
rolling\_corr\_bi2 <- dat\_all\_top6d %>%  
 group\_by(province) %>%  
 filter(!is.na(movement)) %>%  
 tq\_transmute\_xy(x = movement,   
 y = r\_mean,  
 mutate\_fun = runCor,  
 n = 14,  
 col\_rename = "rolling.corr.biweekly") %>%  
 mutate(date\_lag = as.Date(date\_lag))  
  
# rolling correlation - Hubei vs R\_t in other provinces  
# dat\_all\_top6b <- dat\_all\_top6d %>%  
# group\_by(province) %>%  
# mutate(movement\_hubei = dat\_all\_top6d[dat\_all\_top6b$province == 'hubei',]$movement)  
#   
# rolling\_corr\_hubei <- dat\_all\_top6b %>%  
# group\_by(province) %>%  
# filter(!is.na(movement\_hubei)) %>%  
# tq\_transmute\_xy(x = movement\_hubei,   
# y = r\_mean,  
# mutate\_fun = runCor,  
# n = 14,  
# col\_rename = "rolling.corr.hubei")  
  
# rolling\_corr <- left\_join(rolling\_corr\_bi, rolling\_corr\_hubei,   
# by = c("province", "date\_lag")) %>%  
# mutate(date\_lag = as.Date(date\_lag))  
  
dat\_corr2 <- left\_join(dat\_all\_top6d, rolling\_corr\_bi2, by = c("province", "date\_lag"))

readr::write\_csv(dat\_corr, "dat\_all.csv")  
readr::write\_csv(dat\_corr2, "dat\_all\_lag\_sens\_analysis.csv")