StateCoronavirusAnalysis

Stephen R. Proulx

3/14/2020

##Gather and process the data

```
confirmed_sheet<-read.csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_
  select(-Lat, -Long)
deaths_sheet <- read.csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19
  select(-Lat, -Long)
recovered_sheet <- read.csv("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covi
  select(-Lat, -Long)



confirmed_long <- gather(confirmed_sheet, -Province.State , -Country.Region , key="date", value = "cases
  separate(date,c("x","date.longer"),1,remove=TRUE) %>%
  separate(date.longer,c("month","day","year"),remove=TRUE) %>%
  separate(Province.State,c("location","State"),sep=", ",remove=FALSE) %>% #for US data before 3/10/202
  mutate(location = as.character(location)) %>%
  mutate(State = as.character(State)) %>%
  mutate(year=as.character(as.numeric(year)+2000)) %>% #data was in a format with just the last two dig
  unite(comb.date, c(month,day,year) , sep=".")%>%
  mutate(date = parse_date(comb.date , "%m%.%d%.%Y"))%>%
  select(-comb.date , -x) %>%
  mutate(delta.days=period_to_seconds(days(ymd(date) - ymd(20200122)))/(60*60*24)) %>% #calculate days
  as_tibble()
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 14960
## rows [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19,
## 20, ...].
```

Unfortunately the data seems to be grouped into locations that are not consistent throughout the dataset. Before March 10, 2020, the US data is presented by city or county within each state, but after March 10 it is aggregated by state.

For data befor March 10, go through each state to process and combine the data from the different locations.

```
wash.data.substate <- filter(confirmed_long,Country.Region=="US" ,State=="WA") %>%
  filter(delta.days<48) %>%
  group_by(date,State, delta.days, Country.Region) %>% summarise(mean=mean(cases), count=n() ) %>%
  mutate(total.cases = mean*count) %>%
  select(-mean, -count)


wash.data.state <- filter(confirmed_long,Country.Region=="US" , Province.State=="Washington") %>%
  filter(delta.days>47) %>%
  mutate(State = "WA" , total.cases=cases) %>%
  select(-Province.State,-location,-cases)


WA.data.state <- bind_rows(wash.data.state,wash.data.substate)
```

```
CA.data.substate <- filter(confirmed_long,Country.Region=="US" ,State=="CA", Province.State!="Diamond Pr
  filter(delta.days<48) %>%
  group_by(date,State, delta.days, Country.Region) %>% summarise(mean=mean(cases), count=n() ) %>%
  mutate(total.cases = mean*count) %>%
  select(-mean, -count)


CA.data.state <- filter(confirmed_long,Country.Region=="US" , Province.State=="California") %>%
  filter(delta.days>47) %>%
  mutate(State = "CA" , total.cases=cases) %>%
  select(-Province.State,-location,-cases)


CA.data.state <- bind_rows(CA.data.state,CA.data.substate)
```

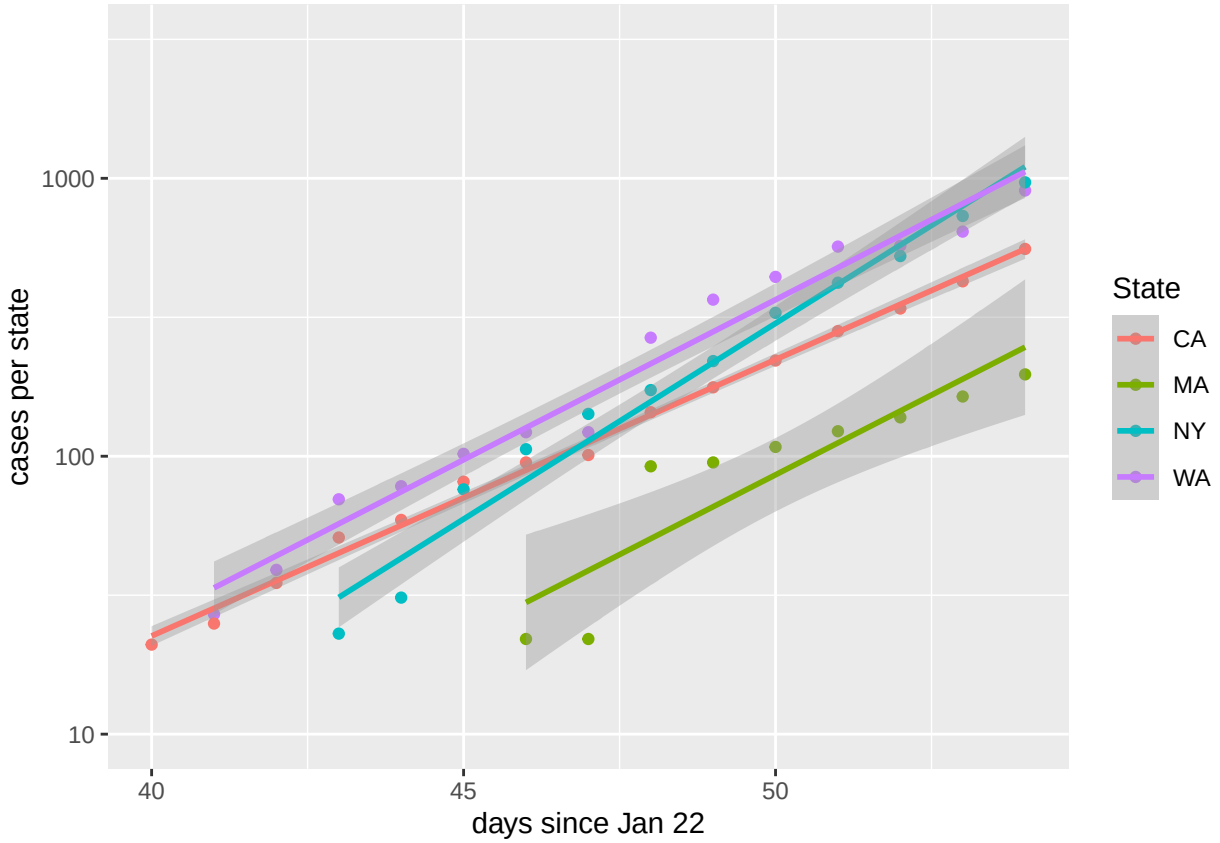
```
NY.data.substate <- filter(confirmed_long,Country.Region=="US" ,State=="NY") %>%
  filter(delta.days<48) %>%
  group_by(date,State, delta.days, Country.Region) %>% summarise(mean=mean(cases), count=n() ) %>%
  mutate(total.cases = mean*count) %>%
  select(-mean, -count)

NY.data.state <- filter(confirmed_long,Country.Region=="US" , Province.State=="New York") %>%
  filter(delta.days>47) %>%
  mutate(State = "NY" , total.cases=cases) %>%
  select(-Province.State,-location,-cases)


NY.data.state <- bind_rows(NY.data.state,NY.data.substate)
```

```
MA.data.substate <- filter(confirmed_long,Country.Region=="US" ,State=="MA") %>%
  filter(delta.days<48) %>%
  group_by(date,State, delta.days, Country.Region) %>% summarise(mean=mean(cases), count=n() ) %>%
  mutate(total.cases = mean*count) %>%
  select(-mean, -count)

MA.data.state <- filter(confirmed_long,Country.Region=="US" , Province.State=="Massachusetts") %>%
  filter(delta.days>47) %>%
  mutate(State = "MA" , total.cases=cases) %>%
  select(-Province.State,-location,-cases)


MA.data.state <- bind_rows(MA.data.state,MA.data.substate)
```

###Plot the stae specific data

```
State.Totals <- bind_rows(MA.data.state,WA.data.state,CA.data.state,NY.data.state) %>% mutate(wday=wday


ggplot(data=filter(State.Totals,total.cases>20) , aes(x=delta.days,y=log(total.cases, base=10) ,color=St
  geom_point()+
    geom_smooth( method = "lm" )+
  scale_y_continuous( limits=c(1,3.5),breaks=c(1,2,3), labels=c(10,100,1000))+
  labs( x="days since Jan 22" , y="cases per state")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



###Plot the country specific data

The US and China data are split by within-country region, so we must sum up the Province.State entries to get the total. The new tibble confirmed_long2 has the non-aggregated US and China data removed and has them replaced by the aggregated version.

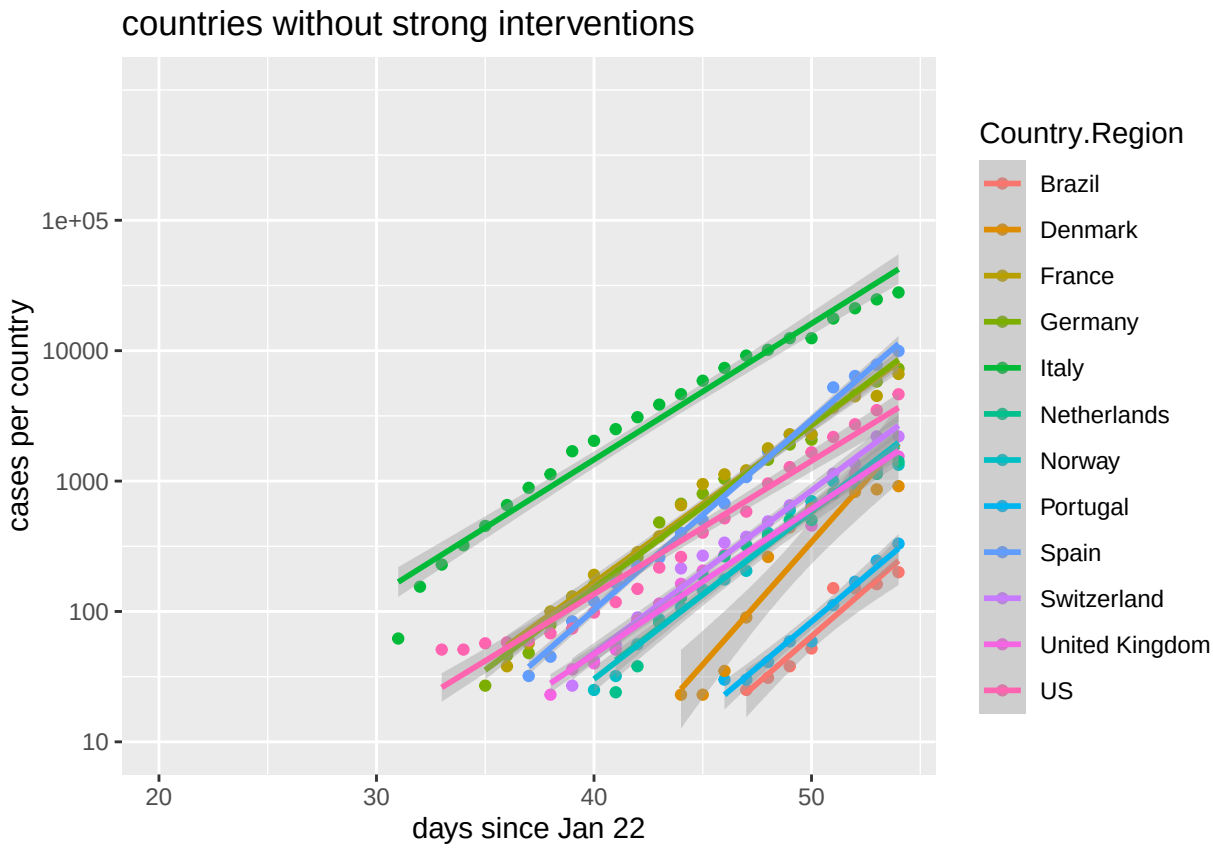
```
USTotals <- filter(confirmed_long,Country.Region=="US" ) %>%
  group_by(date, delta.days, Country.Region) %>% summarise(mean=mean(cases), count=n() ) %>%
  mutate(total.cases = mean*count) %>%
  select(-mean, -count)  %>%
#  mutate(Country.Region = "USA") %>%
  rename(cases=total.cases)

ChinaTotals <- filter(confirmed_long,Country.Region=="China" ) %>%
  group_by(date, delta.days, Country.Region) %>% summarise(mean=mean(cases), count=n() ) %>%
  mutate(total.cases = mean*count) %>%
  select(-mean, -count)  %>%
  mutate(Country.Region = "China") %>%
  rename(cases=total.cases)

confirmed_long2 <- bind_rows(filter(confirmed_long,Country.Region!="China",Country.Region!="US"),USTotal
```

```
## Warning in bind_rows_(x, .id): binding factor and character vector,
## coercing into character vector
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

Plot data from countries that did not have strong interventions. Use "lm" fitting to get linear fits. This is ok for the most part, the US data is poorly fit by a line up until about day 35, maybe becuase those were mostly cases of people returning to the US and not part of the local epidemic.

```
ggplot(data=filter(confirmed_long2,  Country.Region=="France"  | Country.Region=="Italy"  |  Country.Reg
  geom_point(aes(color= Country.Region))+
    geom_smooth(method = "lm", formula = y ~ x) +
  scale_y_continuous( limits=c(1,6),breaks=c(1,2,3,4,5), labels=c(10,100,1000,10000,100000))+
  scale_x_continuous( limits=c(20,54))+
  labs( x="days since Jan 22" , y="cases per country", title="countries without strong interventions")
```

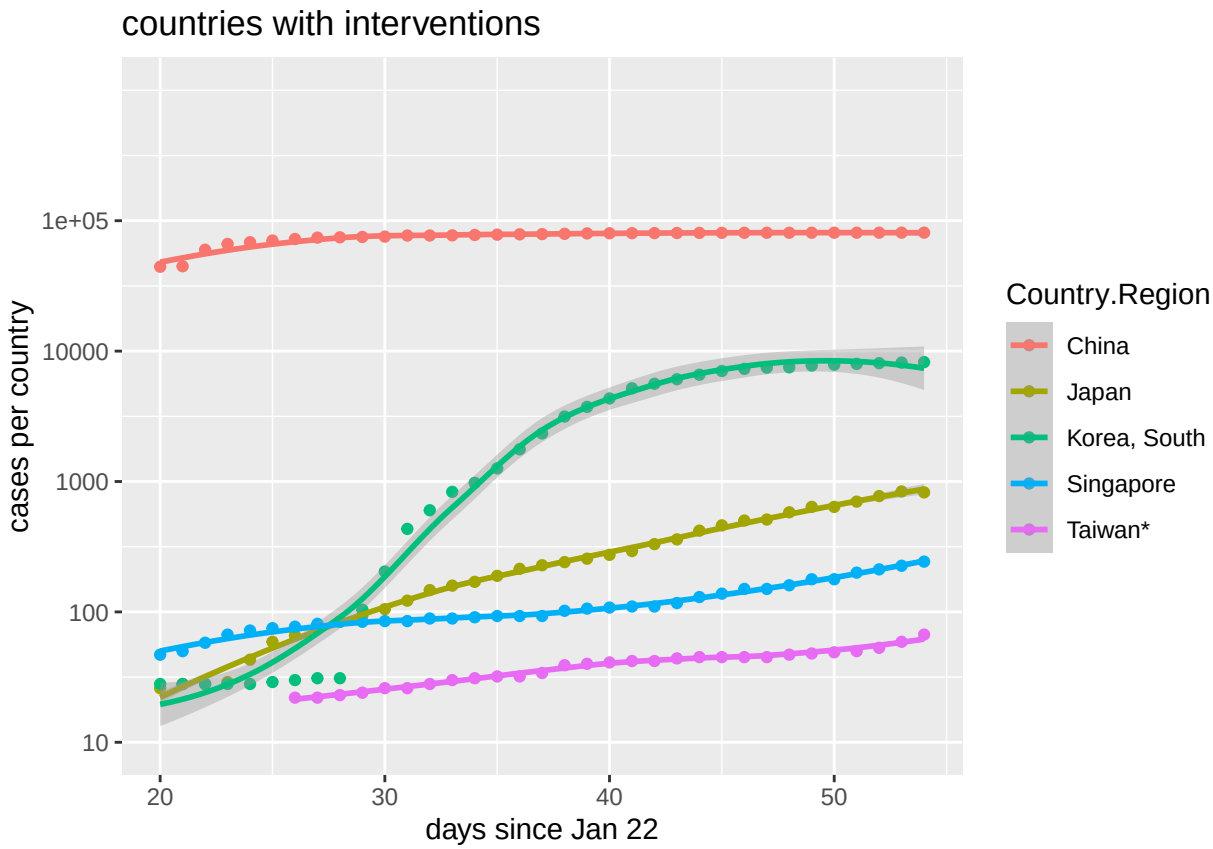


Countries that seemed to have early strong intervention plans. This list is somewhat arbitrary, but these can be more clearly visualized on their own in any case.

```
ggplot(data=filter(confirmed_long2,Country.Region=="China" |Country.Region=="Korea, South" |Country.Regi
  geom_point(aes(color= Country.Region))+
    geom_smooth(method = "loess",formula = y ~ x) +
  scale_y_continuous( limits=c(1,6),breaks=c(1,2,3,4,5), labels=c(10,100,1000,10000,100000))+
   scale_x_continuous( limits=c(20,54))+
  labs( x="days since Jan 22" , y="cases per country" , title="countries with interventions")
```

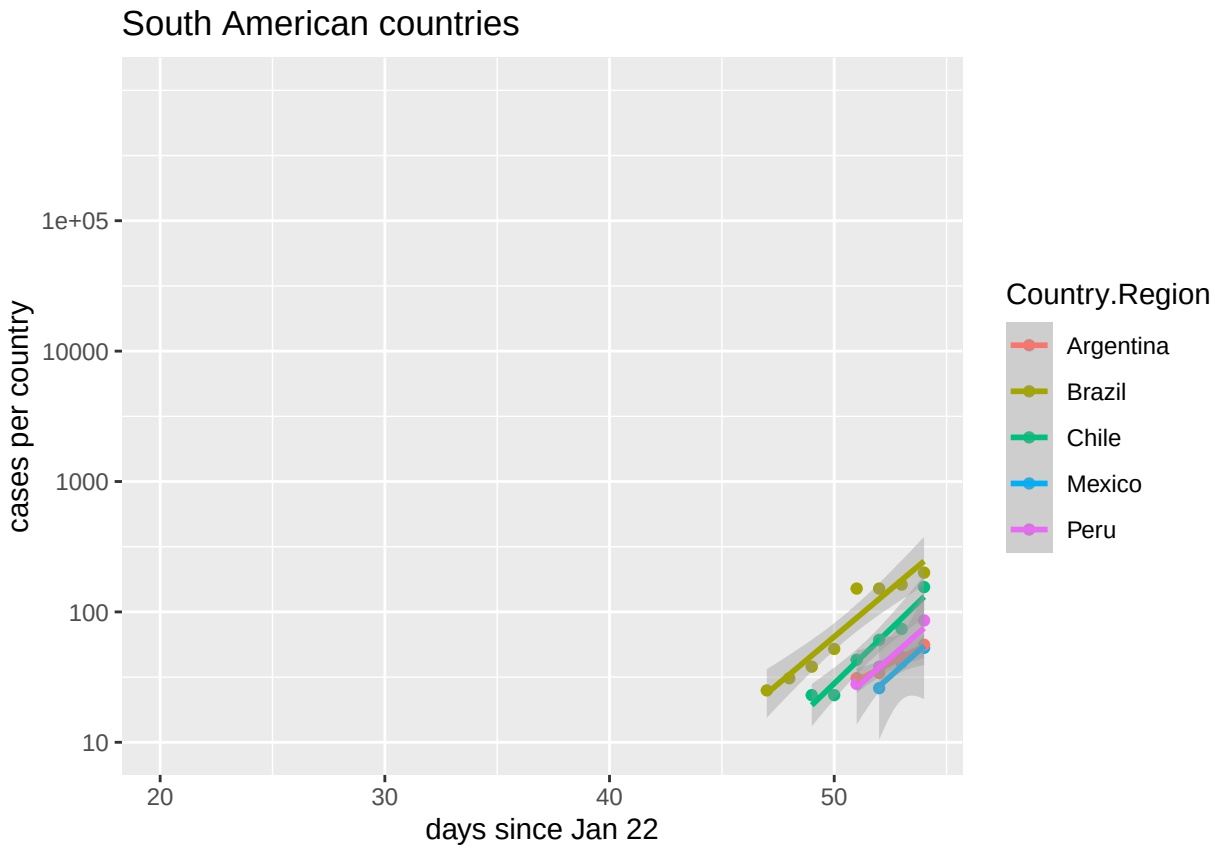
```
## Warning: Removed 39 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 39 rows containing missing values (geom_point).
```



South American countries: Leave the date-range the same so that slopes can be visually compared.

```
ggplot(data=filter(confirmed_long2,Country.Region=="Brazil" |Country.Region=="Argentina" |Country.Regio
  geom_point(aes(color= Country.Region))+
    geom_smooth(method = "lm", formula = y ~ x) +
  scale_y_continuous( limits=c(1,6),breaks=c(1,2,3,4,5), labels=c(10,100,1000,10000,100000))+
  scale_x_continuous( limits=c(20,54))+
  labs( x="days since Jan 22" , y="cases per country", title="South American countries")
```

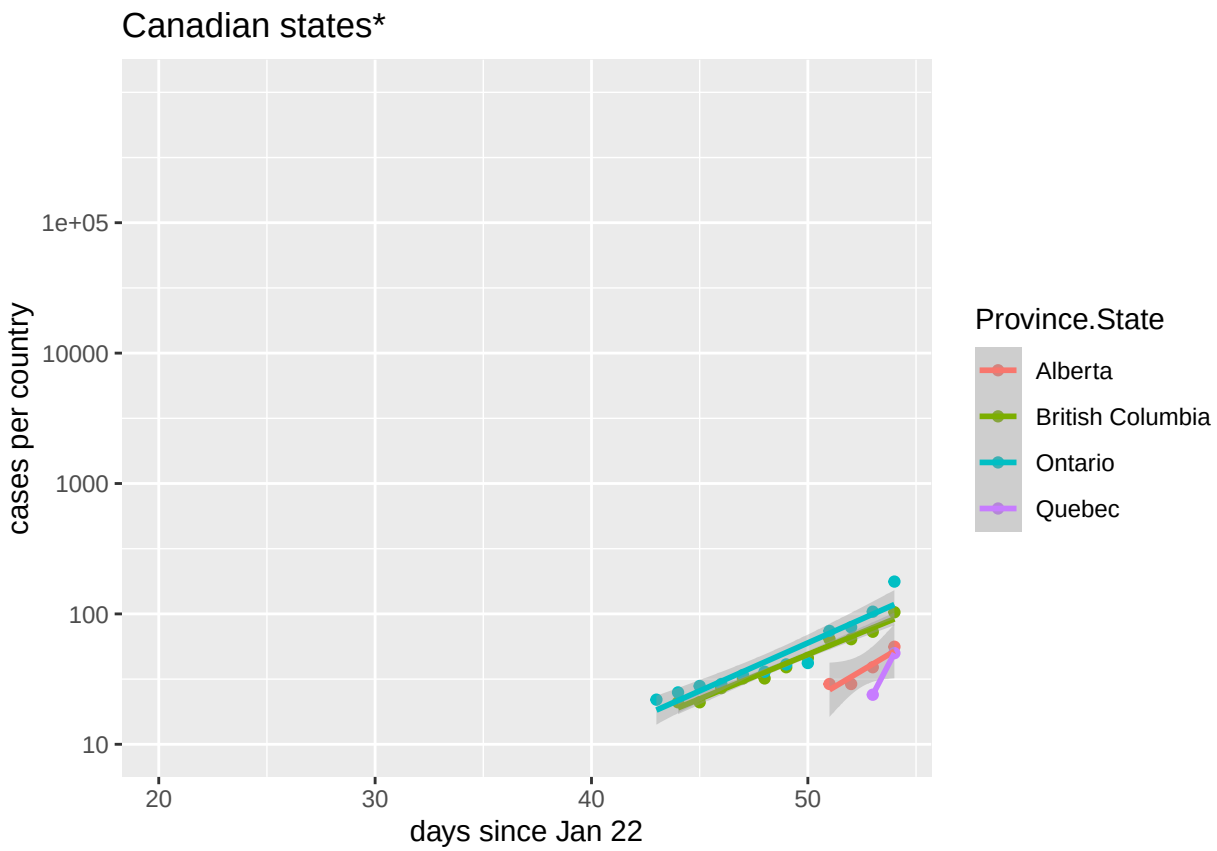


Plot Canada.

```
#Canadians
ggplot(data=filter(confirmed_long2, Country.Region=="Canada"    ,cases>20) , aes(x=delta.days,y=log(cases
  geom_point(aes(color= Province.State))+
    geom_smooth(method = "lm", formula = y ~ x) +
  scale_y_continuous( limits=c(1,6),breaks=c(1,2,3,4,5), labels=c(10,100,1000,10000,100000))+
  scale_x_continuous( limits=c(20,54))+
  labs( x="days since Jan 22" , y="cases per country", title="Canadian states*")
```

```
## Warning in qt((1 - level)/2, df): NaNs produced
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max;
## returning -Inf
```



#Do Bayesian fitting of subsets of the data

This is a very un-sophisiticated method. Just assume that the number of cases on day t+1 is Poisson distributed with lambda = (cases on day t) * lambda_1 where lambda_1 is the per case number of new infections. One thing to think about is that the cases are all supposedly taken out of circulation, so they cannot be literally infecting people on the next day, although they could have infected them before and not been detected. I think a more plausible interpretation of the model is that there are many undetected cases, of which a fraction are detected. If the fraction detected is relatively constant, then the dynamics of the detected cases match those of the undetected cases.

It would be great to build a more complex model including: 1. testing rates and probabilty of being tested given infected, i.e. to build a model of how the medical professionals choose who to test. 2. day of the week effects on testing rate 3. the incubation period 4. variance in the infection rate due to un-measured environmental variance

This writes the stan file. If you have the stan file already can leave this unevaluated

```
sink("model_PoissonOnly.stan")
cat("

    data {
    int<lower=0> n; // number of time points
    int days[n];
    int total_cases[n] ;
    }
    transformed data{
    int new_cases[n] ; //really only need n-1 but keep the indexing the same for simplicity
    new_cases[1]=0;
```

```
    for(i in 2:n){
    new_cases[i]=total_cases[i]-total_cases[i-1];
    }
    }
    parameters {
    real <lower=0 , upper=50.0> lambda ; // Poisson parameter for the growth rate
    }
    model {
    for(i in 2:n){
    new_cases[i] ~ poisson(total_cases[i-1]*lambda); // update for each time step is the sum of Poisson
    }
    }



    ",fill = TRUE)
sink()
```

Note that the only parameter being fit is lambda, which is the per case Poisson parameter. Thus the daily multiplication factor is this lambda, and so the doubling time is log(2)/(log(1+lambda))

US growth rate inference

```
print(fit_poiss, pars=c("lambda"),digits_summary = 3)
```

```
## Inference for Stan model: model_PoissonOnly.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##         mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff  Rhat
## lambda 0.304       0 0.005 0.295 0.301 0.304 0.307 0.313   590 1.007
##
## Samples were drawn using NUTS(diag_e) at Mon Mar 16 22:19:41 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

For California

```
print(fit_poiss, pars=c("lambda"),digits_summary = 3)
```

```
## Inference for Stan model: model_PoissonOnly.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##         mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff  Rhat
## lambda 0.261   0.001 0.012 0.239 0.253 0.261 0.268 0.284   530 1.003
##
## Samples were drawn using NUTS(diag_e) at Mon Mar 16 22:19:42 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

For New York

```
print(fit_poiss, pars=c("lambda"),digits_summary = 3)
```

```
## Inference for Stan model: model_PoissonOnly.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##         mean se_mean    sd 2.5%   25%  50%   75% 97.5% n_eff  Rhat
## lambda 0.341       0 0.011 0.32 0.333 0.34 0.348 0.362   894 1.002
##
## Samples were drawn using NUTS(diag_e) at Mon Mar 16 22:19:43 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Italy

```
print(fit_poiss, pars=c("lambda"),digits_summary = 3)
```

```
## Inference for Stan model: model_PoissonOnly.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##         mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff  Rhat
## lambda 0.196       0 0.001 0.193 0.195 0.196 0.196 0.198   797 1.001
##
## Samples were drawn using NUTS(diag_e) at Mon Mar 16 22:19:45 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

France

```
print(fit_poiss, pars=c("lambda"),digits_summary = 3)
```

```
## Inference for Stan model: model_PoissonOnly.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##         mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff  Rhat
## lambda 0.272       0 0.003 0.265 0.269 0.272 0.274 0.278   676 1.003
##
## Samples were drawn using NUTS(diag_e) at Mon Mar 16 22:19:45 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Brazil

```
print(fit_poiss, pars=c("lambda"),digits_summary = 3)
```

```
## Inference for Stan model: model_PoissonOnly.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##         mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff  Rhat
## lambda 0.288   0.001 0.021 0.247 0.273 0.288 0.302  0.33   664 1.002
##
## Samples were drawn using NUTS(diag_e) at Mon Mar 16 22:19:46 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
```

```
## convergence, Rhat=1).
```

And for the Canadians

```
mydata.sub <- filter(confirmed_long2,cases>20,Province.State == "Ontario")

mindays=min(mydata.sub$delta.days)


mydata.sub <- mutate(mydata.sub,days=delta.days-mindays) %>% arrange(days)   %>%
  rename(total_cases=cases)

n=max(mydata.sub$days)+1


stan_data <- c(mydata.sub[c("days","total_cases")], list(n=n))

#fit the model
fit_poiss <- stan(file = 'model_PoissonOnly.stan',
                  data =stan_data, chains = 4,iter = 1000, seed = 2131231 )
```

```
print(fit_poiss, pars=c("lambda"),digits_summary = 3)
```

```
## Inference for Stan model: model_PoissonOnly.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##         mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff  Rhat
## lambda 0.288   0.001 0.021 0.247 0.273 0.288 0.302  0.33   664 1.002
##
## Samples were drawn using NUTS(diag_e) at Mon Mar 16 22:19:46 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
mydata.sub <- filter(confirmed_long2,cases>20,Province.State == "British Columbia")

mindays=min(mydata.sub$delta.days)


mydata.sub <- mutate(mydata.sub,days=delta.days-mindays) %>% arrange(days)   %>%
  rename(total_cases=cases)

n=max(mydata.sub$days)+1


stan_data <- c(mydata.sub[c("days","total_cases")], list(n=n))

#fit the model
fit_poiss <- stan(file = 'model_PoissonOnly.stan',
                  data =stan_data, chains = 4,iter = 1000, seed = 2131231 )
```

```
print(fit_poiss, pars=c("lambda"),digits_summary = 3)
```

```
## Inference for Stan model: model_PoissonOnly.
## 4 chains, each with iter=1000; warmup=500; thin=1;
```

```
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##         mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff  Rhat
## lambda 0.288   0.001 0.021 0.247 0.273 0.288 0.302  0.33   664 1.002
##
## Samples were drawn using NUTS(diag_e) at Mon Mar 16 22:19:46 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
mydata.sub <- filter(confirmed_long2,cases>20,Province.State == "Alberta")

mindays=min(mydata.sub$delta.days)


mydata.sub <- mutate(mydata.sub,days=delta.days-mindays) %>% arrange(days)   %>%
  rename(total_cases=cases)

n=max(mydata.sub$days)+1


stan_data <- c(mydata.sub[c("days","total_cases")], list(n=n))

#fit the model
fit_poiss <- stan(file = 'model_PoissonOnly.stan',
                  data =stan_data, chains = 4,iter = 1000, seed = 2131231 )
```

```
print(fit_poiss, pars=c("lambda"),digits_summary = 3)
```

```
## Inference for Stan model: model_PoissonOnly.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##         mean se_mean    sd  2.5%   25%   50%   75% 97.5% n_eff  Rhat
## lambda 0.288   0.001 0.021 0.247 0.273 0.288 0.302  0.33   664 1.002
##
## Samples were drawn using NUTS(diag_e) at Mon Mar 16 22:19:46 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```