John Hopkins COVID-19

Edouard Robeyns

2022-04-25

Introduction

Being a foreigner living in China, the recent lockdowns in **Guangdong** (where I live) and **Shanghai** are of particular interest to me. During this short analysis, I will try to identify which of the two areas is preferable.

Library preparation

In order to access convenient functions, some packages must be imported.

```
# use tidyverse
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# use lubridate
library(lubridate)
```

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

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

Import data

First let's import the data. I will use the same method displayed during the course, because I intend to keep this file as reference for later works.

The files are available here at the time of writing (April 2022): link

```r
# save url
url_in <- "https://github.com/CSSEGISandData/COVID-19/raw/master/csse_covid_19_data/csse_covid_19_time_s
# save file names
file_names <- c(
  "time_series_covid19_confirmed_US.csv",
  "time_series_covid19_confirmed_global.csv",
  "time_series_covid19_deaths_US.csv",
  "time_series_covid19_deaths_global.csv",
  "time_series_covid19_recovered_global.csv"
)
# concate url and file names
urls <- str_c(url_in, file_names)
# download the file content into dataframes
confirmed_us <- read_csv(urls[1])
```

```
## Rows: 3342 Columns: 835
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (6): iso2, iso3, Admin2, Province_State, Country_Region, Combined_Key
## dbl (829): UID, code3, FIPS, Lat, Long_, 1/22/20, 1/23/20, 1/24/20, 1/25/20,...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
confirmed_global <- read_csv(urls[2])
```

```
## Rows: 284 Columns: 828
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (2): Province/State, Country/Region
## dbl (826): Lat, Long, 1/22/20, 1/23/20, 1/24/20, 1/25/20, 1/26/20, 1/27/20, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
deaths_us <- read_csv(urls[3])
```

```
## Rows: 3342 Columns: 836
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (6): iso2, iso3, Admin2, Province_State, Country_Region, Combined_Key
## dbl (830): UID, code3, FIPS, Lat, Long_, Population, 1/22/20, 1/23/20, 1/24/...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
deaths_global <- read_csv(urls[4])
```

```
## Rows: 284 Columns: 828
## -- Column specification --------------------------------------------------------
## Delimiter: ","
```

```
## chr   (2): Province/State, Country/Region
## dbl (826): Lat, Long, 1/22/20, 1/23/20, 1/24/20, 1/25/20, 1/26/20, 1/27/20, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
recovered_global <- read_csv(urls[5])
```

```
## Rows: 269 Columns: 828
## -- Column specification ------------------------------------------------------
## Delimiter: ","
## chr   (2): Province/State, Country/Region
## dbl (826): Lat, Long, 1/22/20, 1/23/20, 1/24/20, 1/25/20, 1/26/20, 1/27/20, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Data summary

I always start with a summary of the data, to get an idea of the number of lines/columns, and to start identifying which columns may be relevant to the chosen areas of inquiry.

I intentionally cut the dates columns out to shorten the document.

```
# display the summary
summary(confirmed_global[1:4])
```

```
##  Province/State     Country/Region          Lat              Long        
##  Length:284         Length:284         Min.   :-71.950   Min.   :-178.12  
##  Class :character   Class :character   1st Qu.:  4.643   1st Qu.: -22.04  
##  Mode  :character   Mode  :character   Median : 21.608   Median :  20.92  
##                                        Mean   : 20.106   Mean   :  21.96  
##                                        3rd Qu.: 40.951   3rd Qu.:  84.99  
##                                        Max.   : 71.707   Max.   : 178.06  
##                                        NA's   :2         NA's   :2        
```

Analysis

First, I will clean the global cases files:

- pivot the *date*/*case* to have one *case* number per *date*
- rename *Country* and *Province* columns
- transform the *date* column to *R type date*
- remove *Latitude* and *Longitude* which are not relevant

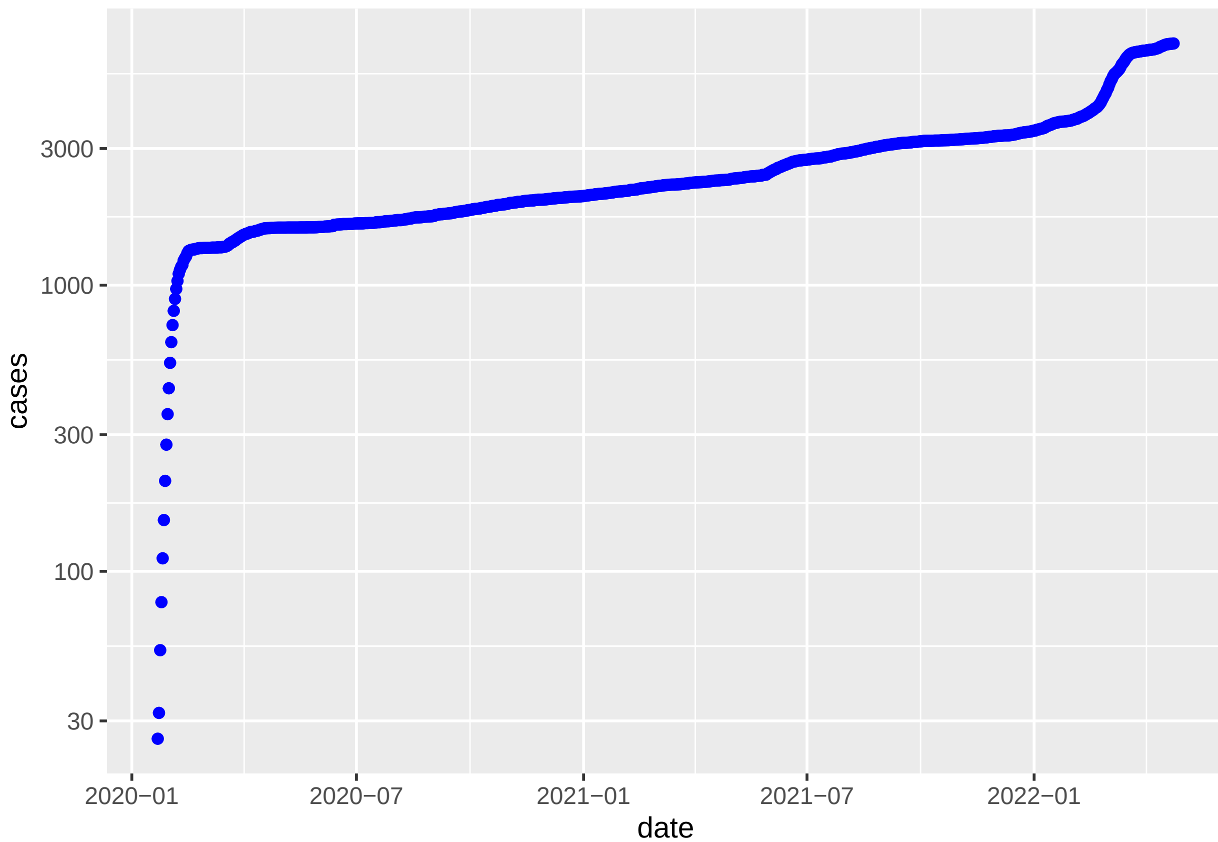
```
# global cases
global_cases <- confirmed_global %>%
  pivot_longer(cols = -c("Province/State", "Country/Region", Lat, Long),
               names_to = "date",
                values_to = "cases") %>%
  rename(Country_region = "Country/Region",
```

```
         Province_state = "Province/State") %>%
  mutate(date = mdy(date)) %>%
  select(-c(Lat, Long))
```

Guangdong region cases

First, let's isolate the cases for **Guangdong**. Once filtered, I remove the *Province* column to unify the data with subsequent dataframes. I will use the color blue, and a log10 scale for better readability.

```
# filter data for Guangdong proving only
guangdong_cases <- global_cases %>% filter(global_cases$Country_region == "China", global_cases$Province
# remove the province column
guangdong_cases$Province_state <- NULL

# plot the result
ggplot() + geom_point(data=guangdong_cases, aes(x = date, y = cases), color="blue") + scale_y_log10()
```

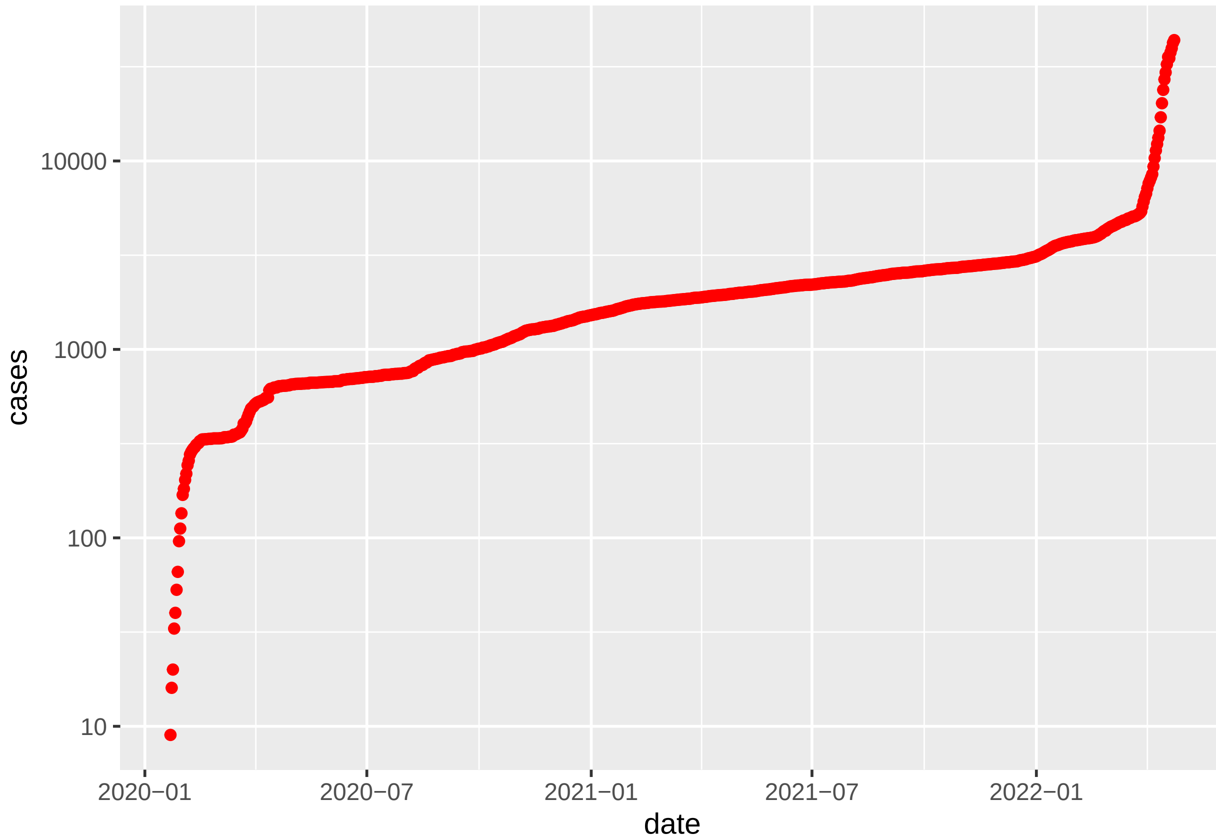


Shanghai region cases

Then, let's do the same with **Shanghai**. I will use the color red.

```
# filter data for Shanghai proving only
shanghai_cases <- global_cases %>% filter(global_cases$Country_region == "China", global_cases$Province_
```

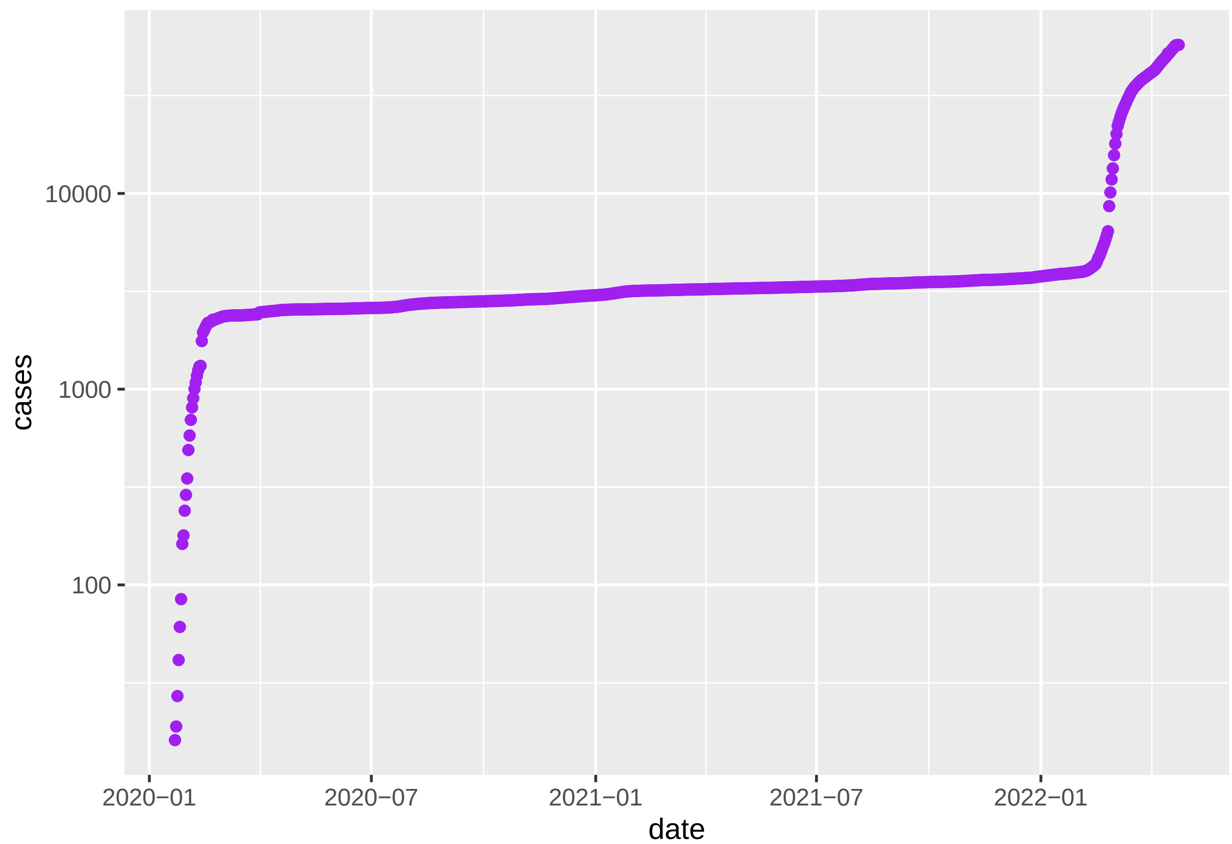
```r
# remove the province column
shanghai_cases$Province_state <- NULL

# plot the result
ggplot() + geom_point(data=shanghai_cases, aes(x = date, y = cases), color="red") + scale_y_log10()
```



China region cases

In order to have a good reference point, let's add the **Chinese** average. I will use the color purple.

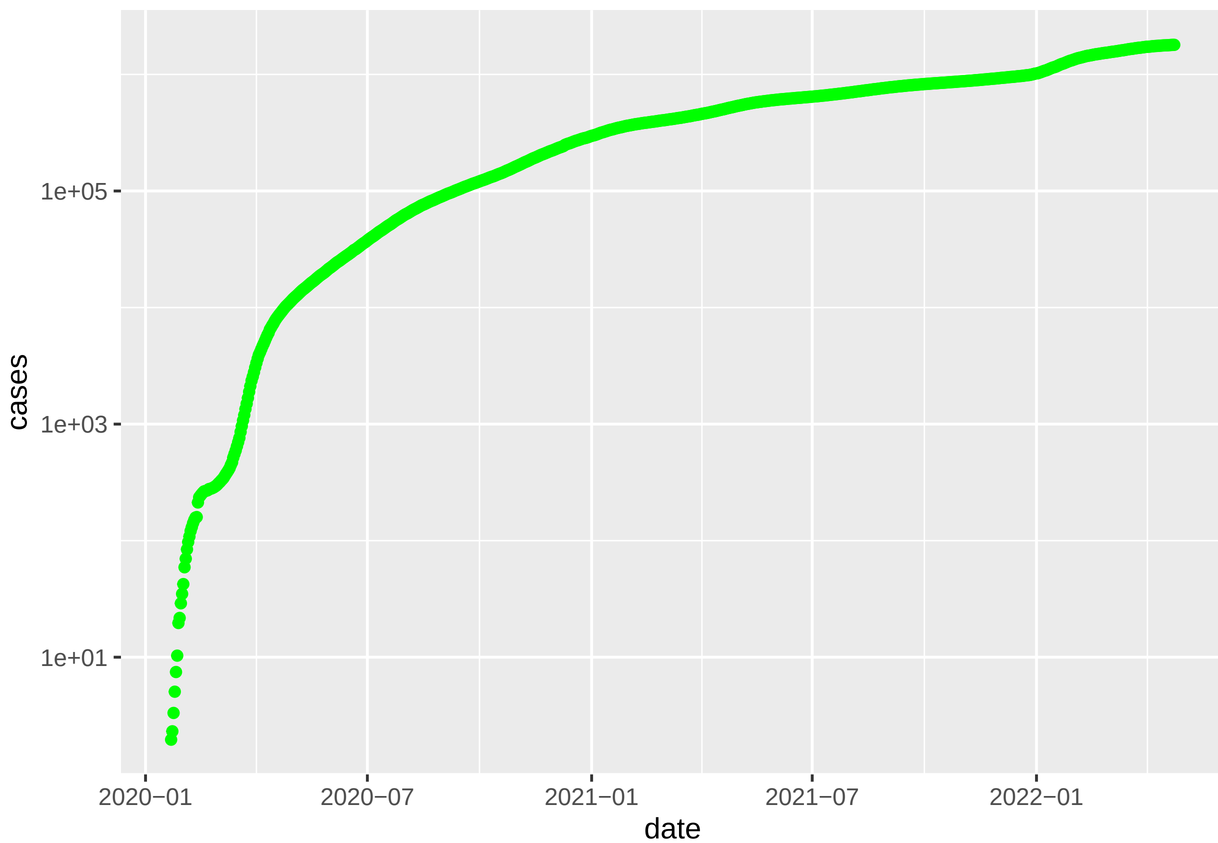
```r
# filter data for all provinces of China
china_cases <- global_cases %>% filter(global_cases$Country_region == "China")
# remove the province column
china_cases$Province_state <- NULL
# average the cases per date
china_cases <- aggregate(cases ~ date, china_cases, mean)

# plot the result
ggplot() + geom_point(data=china_cases, aes(x = date, y = cases), color="purple") + scale_y_log10()
```



World region cases

Finally, let's add the **world** average as well. I will use the color green.

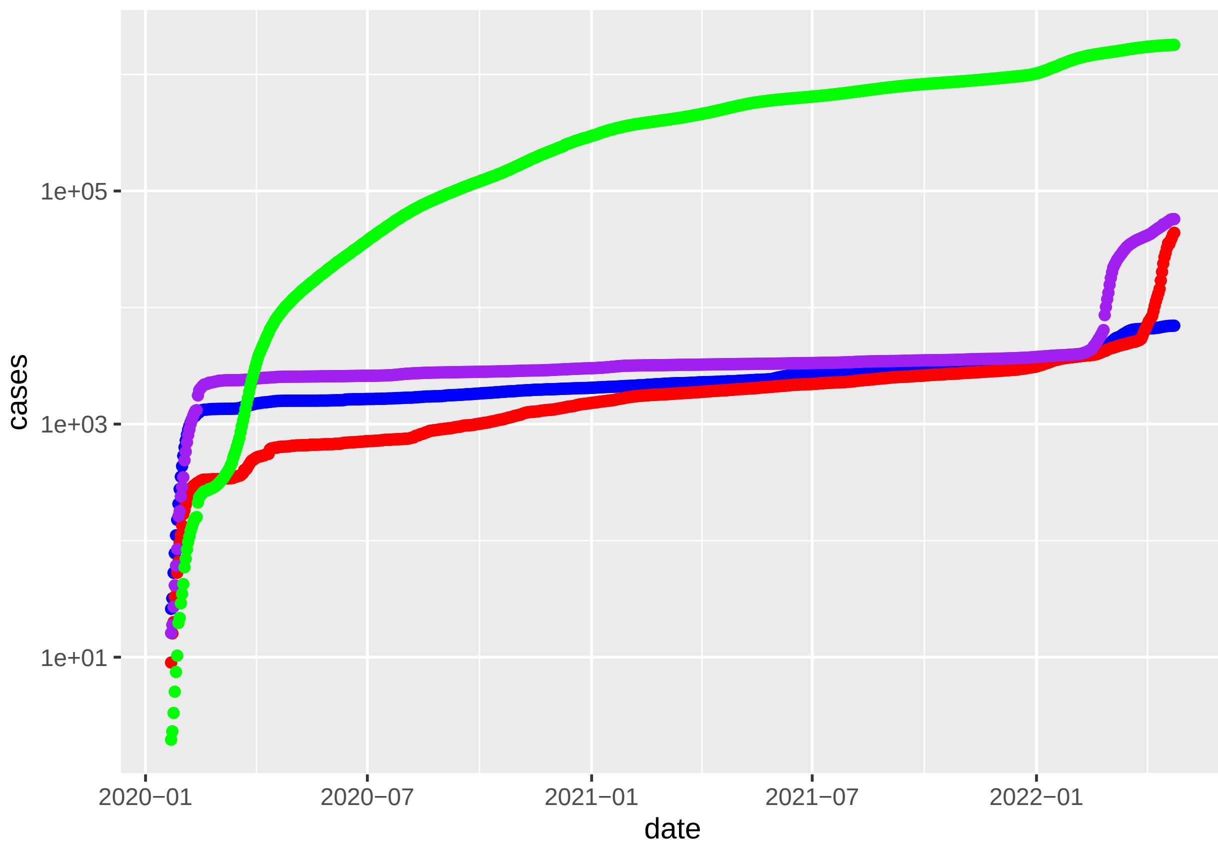
```
# select only only cases and date column for the whole world
world_cases <- global_cases %>% select(3:4)
# average the cases per date
world_cases <- aggregate(cases ~ date, world_cases, mean)

# plot the result
ggplot() + geom_point(data=world_cases, aes(x = date, y = cases), color="green") + scale_y_log10()
```



Combining

Finally, let's display all 4 dataframes onto a graph, keeping the same colors.

```r
# add all 4 dataframes to a single graph
ggplot() +
  geom_point(data=guangdong_cases, aes(x = date, y = cases), color="blue") +
  geom_point(data=shanghai_cases, aes(x = date, y = cases), color="red") +
  geom_point(data=china_cases, aes(x = date, y = cases), color="purple") +
  geom_point(data=world_cases, aes(x = date, y = cases), color="green") +
  scale_y_log10()
```



Several points can be made:

- the **world** line (green) displays a well-defined logarithm curve. Since the graph is using a log10 on the y axis, it means the rate was steadily linear throughout the 2+ years of pandemic
- all 3 curves for **China**, **Guangdong** and **Shanghai** display the overall same pattern: a steep increase at the beginning of 2020 (the initial outbreak of the virus), then a very 'flat' period until early 2022 when a second outbreak happened
- **Shanghai** was doing much better than **Guangdong** during the 2020 outbreak
- while **Guangdong** curve was higher than **Shanghai** curve, **Shanghai** had a steeper curve and reached the same levels by the end of 2021
- the last outbreak of early 2022 started later in **Guangdong** than on average in **China**, and later still in **Shanghai**
- during the last outbreak, while **Guangdong** was doing initially worse than **Shanghai**, the situation clearly was quickly under control, while **Shanghai** situation is quickly catching up to the national average
- while **Shanghai** situation is a lot worse than **Guangdong**'s (this is a logarithmic scale, so around 10 times worse), both are still below the national average, and **China** is still doing a lot better than the **world** average

Model

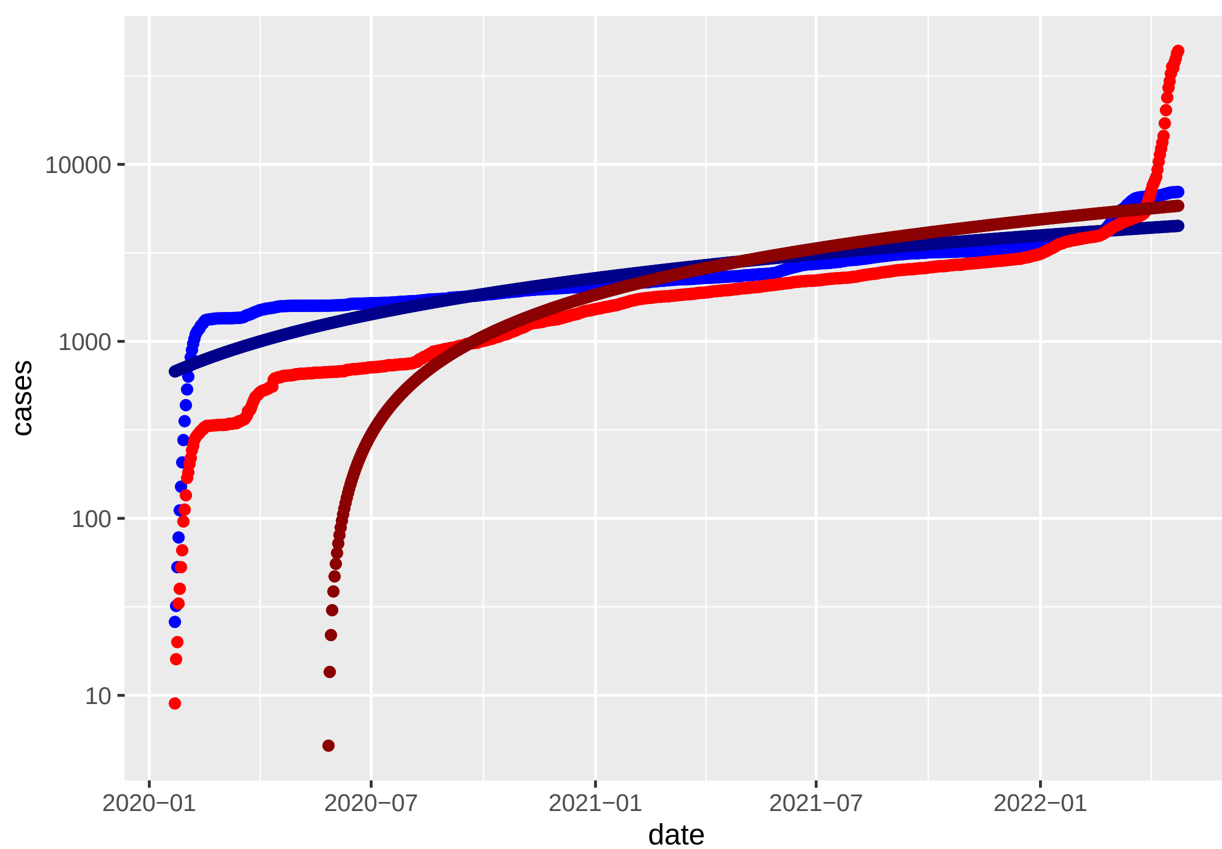
Linear model

Since **Shanghai** was doing better than **Guangdong** during the first outbreak, but worse during the second outbreak, it is hard to define which city is overall safest.

Using a linear regression may help answer this question.

```
# create Guangdong model
guangdong_mod <- lm(as.numeric(cases) ~ as.numeric(date), data=guangdong_cases)
# generate prediction for Guangdong
guangdong_cases_with_model <- guangdong_cases %>% mutate(pred = predict(guangdong_mod))

# create Shanghai model
shanghai_mod <- lm(as.numeric(cases) ~ as.numeric(date), data=shanghai_cases)
# generate predictions for Shanghai
shanghai_cases_with_model <- shanghai_cases %>% mutate(pred = predict(shanghai_mod))

# plot both, models are darker colors
ggplot() +
  geom_point(data=guangdong_cases_with_model, aes(x = date, y = cases), color = "blue") +
  geom_point(data=guangdong_cases_with_model, aes(x = date, y = pred), color = "blue4") +
  geom_point(data=shanghai_cases_with_model, aes(x = date, y = cases), color = "red") +
  geom_point(data=shanghai_cases_with_model, aes(x = date, y = pred), color = "red4") +
  scale_y_log10()
```

```
## Warning in self$trans$transform(x): NaNs produced

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 126 rows containing missing values (geom_point).
```

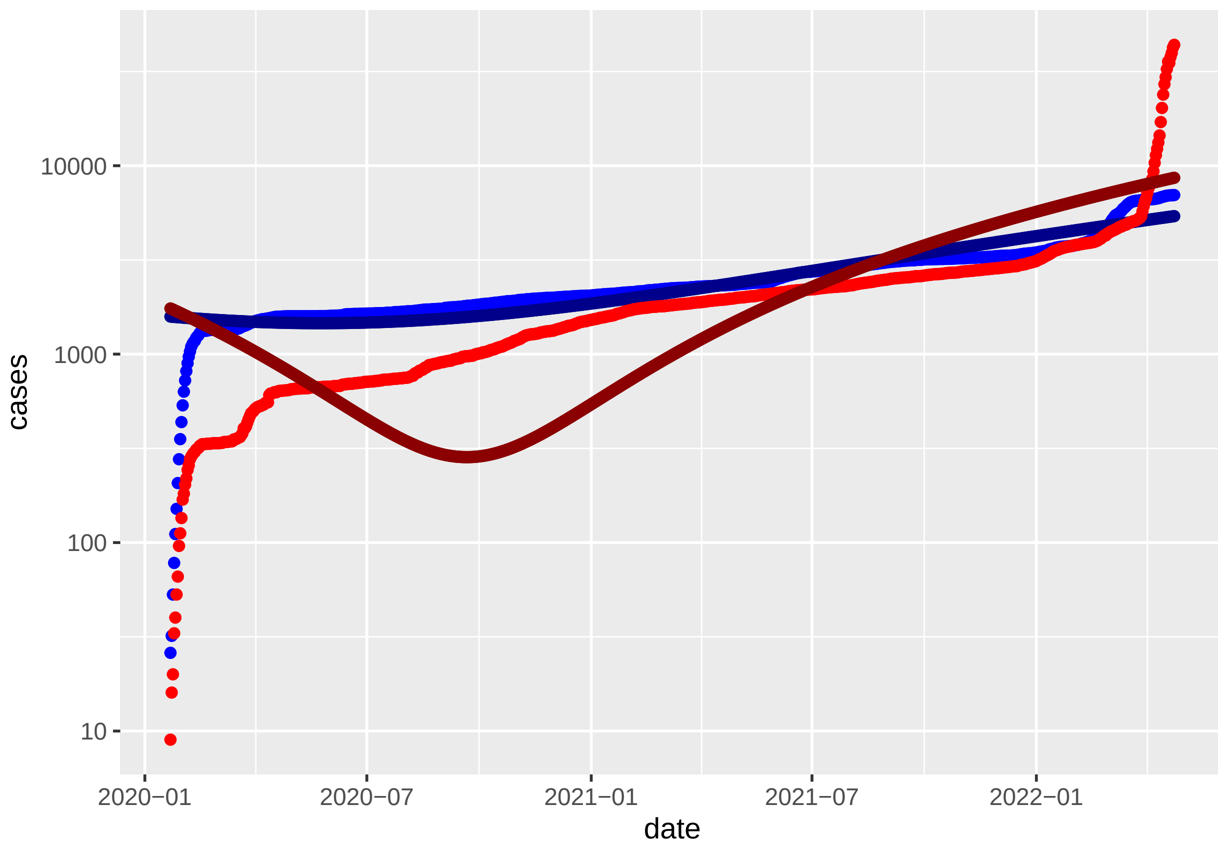


This graph is very to read, the model prediction for **Shanghai** start much later than **Guangdong**'s, and ends up slightly higher. Both curve have the same shape, making conclusion hazardous. Furthermore, R displays 3 warnings, lowering the confidence to use this graph.

Quadratic model

A linear model only show the overall tendency, and may not be the best fit for the data. Next, I will use a quadratic model and see if it provides better insights.

```
# add squared date to Guangdong data
guangdong_cases$date2 <- as.numeric(guangdong_cases$date)^2
# create Guangdong model
guangdong_mod <- lm(as.numeric(cases) ~ as.numeric(date) + date2, data=guangdong_cases)
# generate prediction for Guangdong
guangdong_cases_with_model <- guangdong_cases %>% mutate(pred = predict(guangdong_mod))

# add squared date to Shanghai data
shanghai_cases$date2 <- as.numeric(shanghai_cases$date)^2
# create Shanghai model
shanghai_mod <- lm(as.numeric(cases) ~ as.numeric(date) + date2, data=shanghai_cases)
# generate prediction for Shanghai
shanghai_cases_with_model <- shanghai_cases %>% mutate(pred = predict(shanghai_mod))

# plot both, models are darker colors
ggplot() +
  geom_point(data=guangdong_cases_with_model, aes(x = date, y = cases), color = "blue") +
  geom_point(data=guangdong_cases_with_model, aes(x = date, y = pred), color = "blue4") +
  geom_point(data=shanghai_cases_with_model, aes(x = date, y = cases), color = "red") +
  geom_point(data=shanghai_cases_with_model, aes(x = date, y = pred), color = "red4") +
  scale_y_log10()
```



This is more interesting. First, R does not display any warning. Second, the curves cross at two points. The surface enclosed between the curves before mid-2021 is visually greater than the surface between them after. This would tend to indicate that on average, **Shanghai** was safer than **Guangdong**.

Conclusion

At the time of writing of this document (April 2022), recent events on social media and in the news are painting a dire situation in **Shanghai**. Looking at the data and the above analysis makes me realize to temperate my feeling. Overall, **Shanghai** fared better than **Guangdong** during the whole pandemic. And while the situation is worrisome at the moment, those numbers are still below **Chinese** average and far below the **world** average.

Bias

Since I am living in **Guangdong**, it is obvious I have an inherent bias toward the city I live in. Furthermore, the choice of **Shanghai** was motivated by visibility on social media, and any number of other Chinese city could have been of interest.

The choice to restrict to only two cities, while motivated by a desired to not over-extend this document, is also a source of bias. I added the **Chinese** and **world** averages in an effort to balance any misrepresentation.

Session info

Here are the session info.

```r
# display session info
sessionInfo()
```

```
## R version 4.1.3 (2022-03-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 22000)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] lubridate_1.8.0 forcats_0.5.1   stringr_1.4.0   dplyr_1.0.8
##  [5] purrr_0.3.4     readr_2.1.2     tidyr_1.2.0     tibble_3.1.6
##  [9] ggplot2_3.3.5   tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.1.2 xfun_0.30        haven_2.4.3      colorspace_2.0-3
##  [5] vctrs_0.4.0      generics_0.1.2   htmltools_0.5.2  yaml_2.3.5
##  [9] utf8_1.2.2       rlang_1.0.2      pillar_1.7.0     glue_1.6.2
## [13] withr_2.5.0      DBI_1.1.2        bit64_4.0.5      dbplyr_2.1.1
## [17] modelr_0.1.8     readxl_1.4.0     lifecycle_1.0.1  munsell_0.5.0
## [21] gtable_0.3.0     cellranger_1.1.0 rvest_1.0.2      evaluate_0.15
## [25] knitr_1.38       tzdb_0.3.0       fastmap_1.1.0    curl_4.3.2
## [29] parallel_4.1.3   fansi_1.0.3      highr_0.9        broom_0.8.0
## [33] backports_1.4.1  scales_1.2.0     vroom_1.5.7      jsonlite_1.8.0
## [37] farver_2.1.0     bit_4.0.4        fs_1.5.2         hms_1.1.1
## [41] digest_0.6.29    stringi_1.7.6    grid_4.1.3       cli_3.2.0
## [45] tools_4.1.3      magrittr_2.0.3   crayon_1.5.1     pkgconfig_2.0.3
## [49] ellipsis_0.3.2   xml2_1.3.3       reprex_2.0.1     assertthat_0.2.1
## [53] rmarkdown_2.13   httr_1.4.2       rstudioapi_0.13  R6_2.5.1
## [57] compiler_4.1.3
```