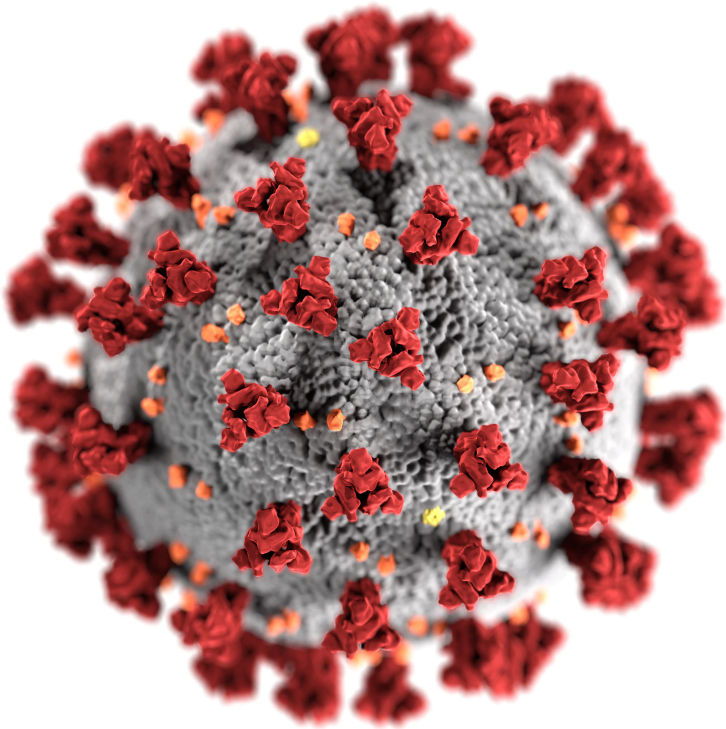
Covid-2019, Analysis, Data Visualization, Prediction

Author: ***Zhuofei, Zhou*** — ***based on global data***

by *using Julia(version: `1.5.2`)*:

Introduction



Novel Coronavirus 2019, on 12 January 2020, WHO officially named it 2019-NCOV. Coronaviruses are a large family of viruses known to cause colds and more serious illnesses such as Middle East Respiratory syndrome (MERS) and severe acute respiratory syndrome (SARS). Novel Coronavirus is a novel coronavirus strain that has never been found in humans before. **More information is available**.

Collect data:

> Data from COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University

you can download the datasets from github.**link**, So I download five CSV files from github.

here is the file name:

- time series covid19 confirmed global.csv
- time series covid19 deaths global.csv
- time series covid19 recovered global.csv
- cases country.csv
- cases time.csv

Analysis

before start, we shold import some *julia packages* to help to process the data and Follow-up work.

```
#import some packages
begin
    using DataFrames
    import CSV
    using DelimitedFiles
    using Dates
    using Gadfly
    using Compose
    using Flux
    using IterTools: ncycle
    using Parameters: @with_kw
    using Query
end
```

First, focus on Global situation. I will read the three CSV files, which include global word in file name, And create a new dataframe named "Global_num" to store the new data

255 rows × 4 columns

	Date Date…	confirmed_num Int64	deaths_num Int64	recovered_num Int64
1	2020-01-22	555	17	28
2	2020-01-23	654	18	30
3	2020-01-24	941	26	36
4	2020-01-25	1434	42	39
5	2020-01-26	2118	56	52
6	2020-01-27	2927	82	61
7	2020-01-28	5578	131	107

	Date	confirmed_num	deaths_num	recovered_num
	Date…	Int64	Int64	Int64
8	2020-01-29	6167	133	126
9	2020-01-30	8235	171	143
10	2020-01-31	9927	213	222
11	2020-02-01	12038	259	284
12	2020-02-02	16787	362	472
13	2020-02-03	19887	426	623
14	2020-02-04	23898	492	852
15	2020-02-05	27643	564	1124
16	2020-02-06	30803	634	1487
17	2020-02-07	34396	719	2011
18	2020-02-08	37130	806	2616
⋮	⋮	⋮	⋮	⋮

```
begin
    #read three CSV files
    dt, Header = readdlm("time_series_covid19_confirmed_global.csv", ',', header=true)
    dt_deaths, Header_deaths = readdlm("time_series_covid19_deaths_global.csv", ',', header=true)
    dt_recover,Header_recov = readdlm("time_series_covid19_recovered_global.csv",',',header=true)
    confirmed_num = []
    deaths_num = []
    recovered_num = []
    for i in 5:259
        push!(confirmed_num, sum(dt[:, i]))
        push!(deaths_num, sum(dt_deaths[:, i]))
        push!(recovered_num, sum(dt_recover[:,i]))
    end
    dates = Date(2020, 1, 22):Day(1):Date(2020, 10, 2)
    Global_num = DataFrame(Date=dates,
        confirmed_num=confirmed_num,
        deaths_num=deaths_num,
        recovered_num=recovered_num)
    Global_num[:confirmed_num] = convert.(Int, Global_num[:confirmed_num])
    Global_num[:deaths_num] = convert.(Int, Global_num[:deaths_num])
    Global_num[:recovered_num] = convert.(Int, Global_num[:recovered_num])
    Global_num

end
```

So, until `2020.10.02` , the global comfirm cases numbers, deaths numbers and recovered numbers.

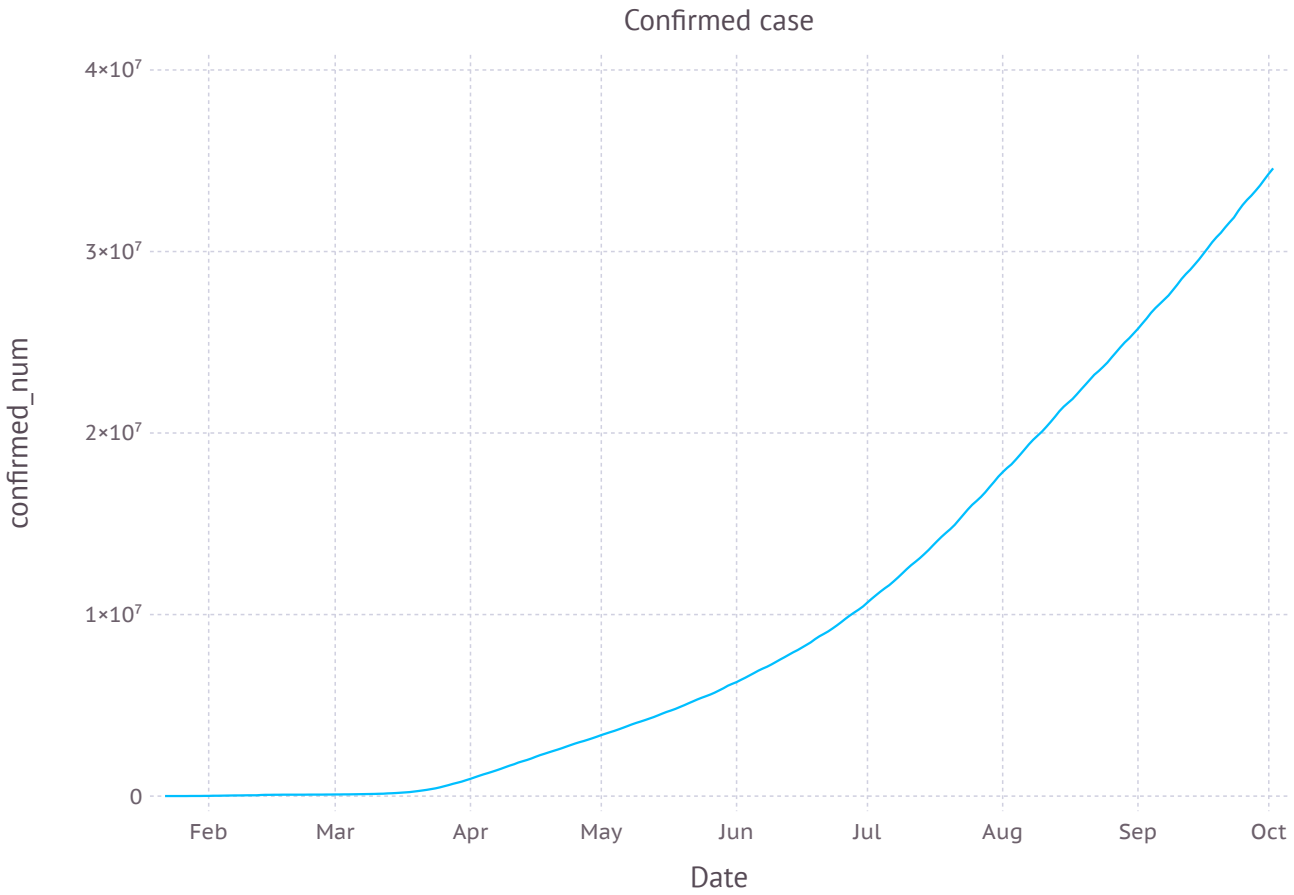
1 rows × 4 columns

	Date	confirmed_num	deaths_num	recovered_num
	Date…	Int64	Int64	Int64
1	2020-10-02	34582601	1027825	24029499

The ratio of confirmed cases to the world population is **0.004433666794871795**.

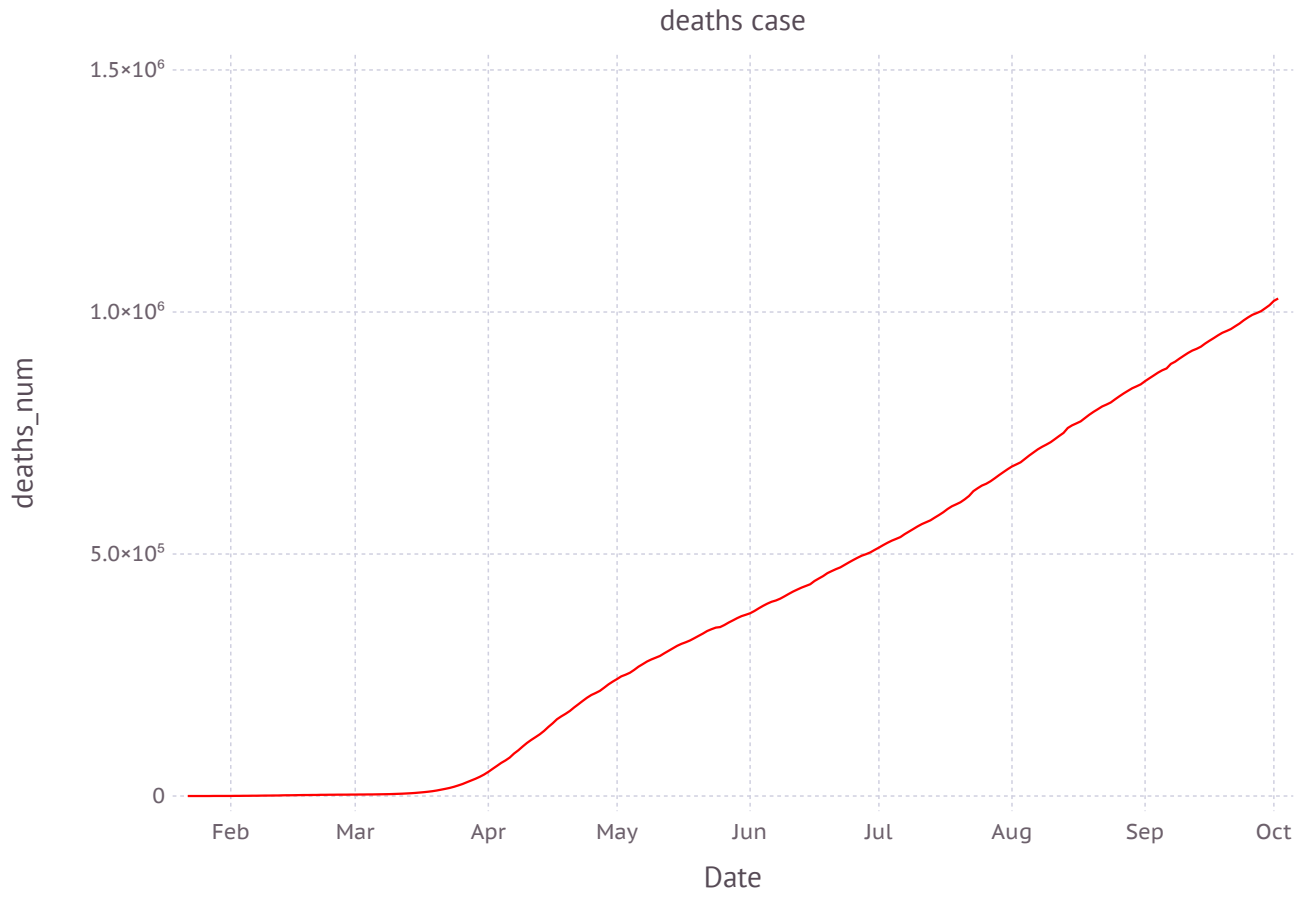
```
set_default_plot_size(18cm ,13cm)
```

```
confirmed =
```



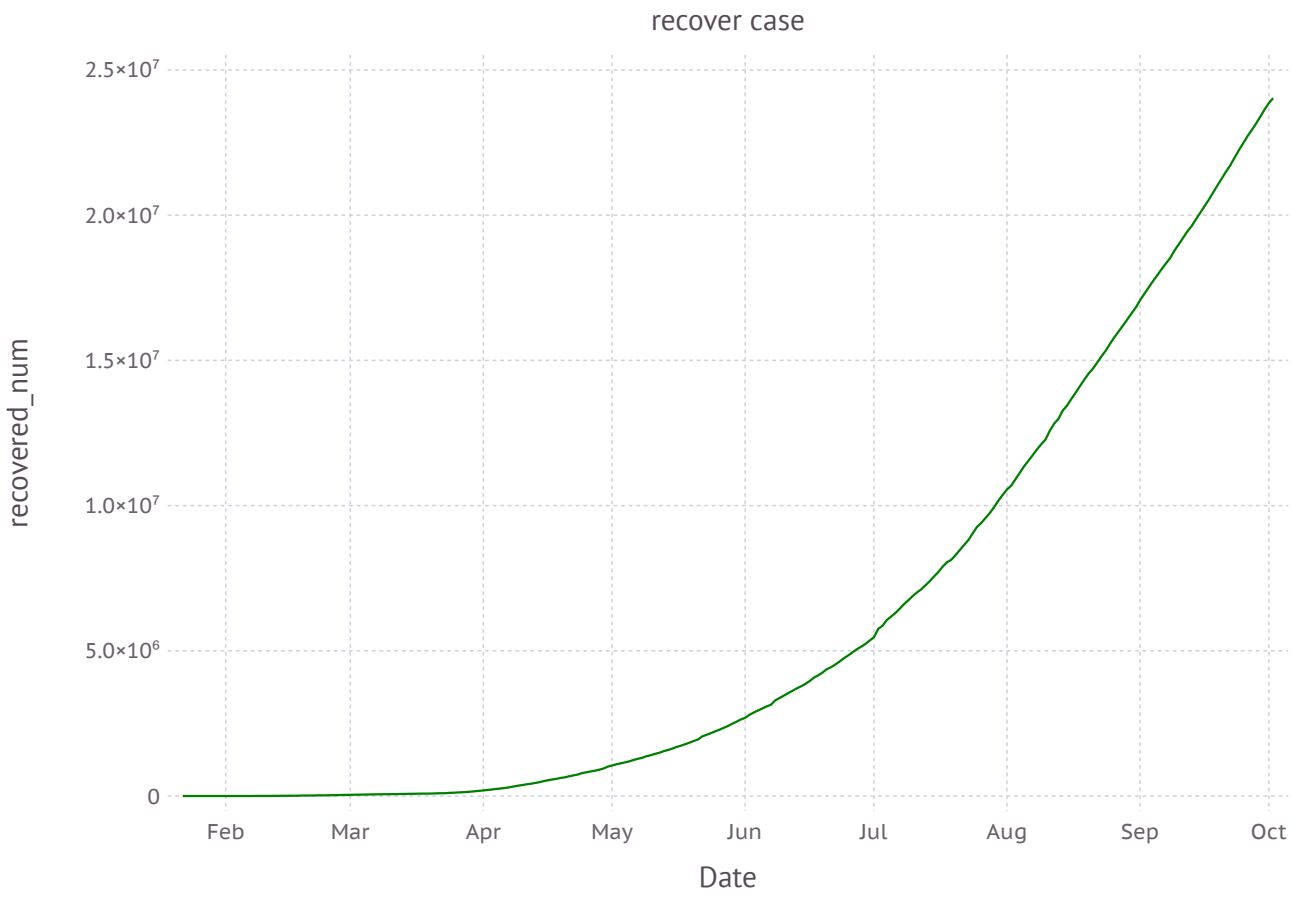

```
confirmed = plot(Global_num, x=:Date, y=:confirmed_num,Geom.line,
Guide.title("Confirmed case"))
```

```
death =
```




```
death = plot(Global_num, x=:Date, y=:deaths_num,Geom.line, Guide.title("deaths case"), Theme(default_color=colorant"red"))
```

```
recover =
```




```
recover = plot(Global_num, x=:Date, y=:recovered_num,Geom.line, Guide.title("recover case"), Theme(default_color=colorant"green"))
```

As we can see, Both confirmed cases and recovered cases have increased exponentially. the deaths cases have increased linearly.

Prediction

Global Trend:

It is useful to understand the global trend of an increase in the number of cases over time. There is always a pattern in any data, but the concern is how strongly data follows a pattern. COVID-19 spreads exponentially.

I think of some ways to estimate the curve:

- Numerical Analysis:✗
- linear regression: ✓
- neural network: ✓

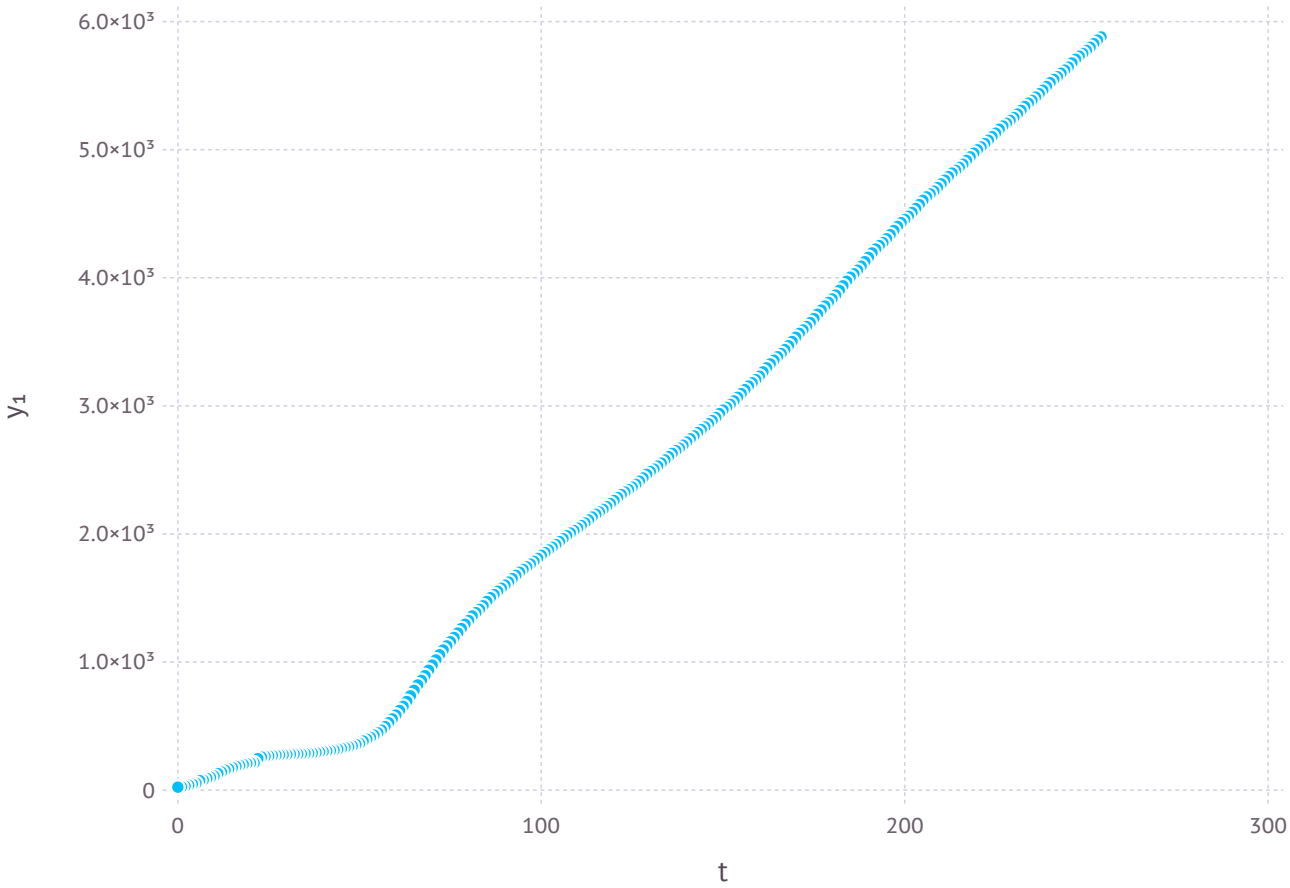
If we choose Numerical Analysis, and use the Interpolation and the Lagrange Polynomial to fiting curve, only can use very small data. because If the data is too large, the curve relation fitted by

numerical analysis has many higher order terms,rather than machine learning. So to use most of the data set, we choose the linear regression and neural network to fitting curve.

Linear regression:

So, we first focus on confirmed case:

if we do a transmission: $y_1 = \sqrt{y}$, set t = 0,1,..., then draw a scatter plot.The following：




```
begin
    y = Global_num.confirmed_num
    y₁ = sqrt.(y)
    t = collect(0:n-1)
    new_dt = DataFrame(t=t, y=y, y₁=y₁)
    plot(new_dt, x=:t, y=:y₁, Geom.point)
end
```

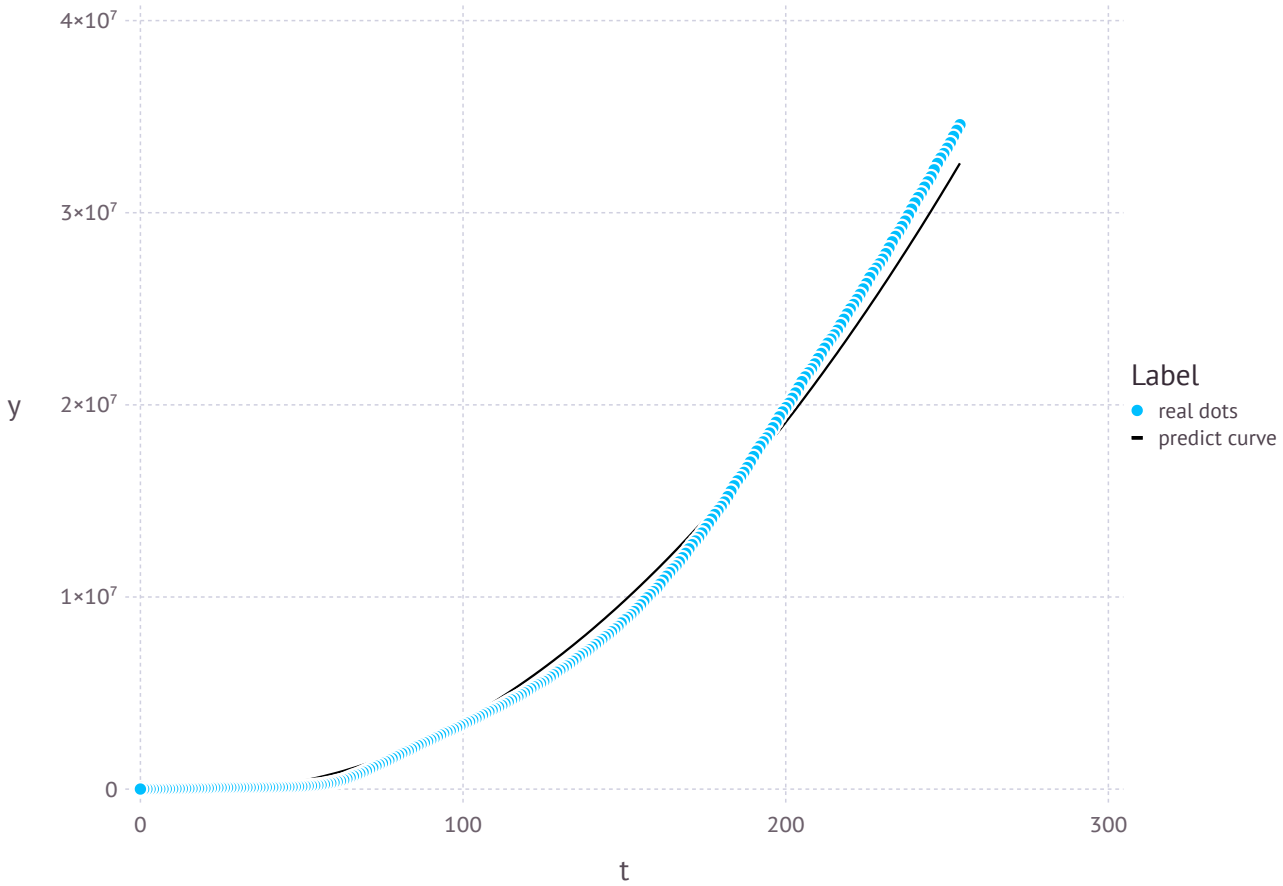
$$\sqrt{y} = Xw + b$$

```
Float64[3.41397e5, 3.13064e5, 2.85959e5, 260080.0, 2.35429e5, 2.12004e5, 1.89807e5
```

```
begin
    X = [t ones(n)]
    β = inv(X'*X)*X'*y₁
    ŷ = (X*β).^2
    new_dt.pre_y = ŷ
end
```

so the β_1 is 24.77057725959722, β_0 equal -584.2921209950298

predict function : $y = (24.7706t - 584.2921)^2$




```
plot(layer(new_dt, x=:t, y=:y, Geom.point),layer(new_dt, x=:t, y=:pre_y, Geom.line,Theme(default_color="black")),Guide.manual_color_key("Label", ["real dots","predict curve"], ["deepskyblue","black"],shape=[Shape.circle, Shape.hline]))
```

Other ways: ***Neural Network***

`import Flux` to do meachine learning.

show all code here:

```
using Flux, Statistics
using Flux.Data: DataLoader
using Flux: throttle
using Parameters: @with_kw
using DelimitedFiles
using IterTools: ncycle
using Dates
using DataFrames
using CSV
using Plots
```

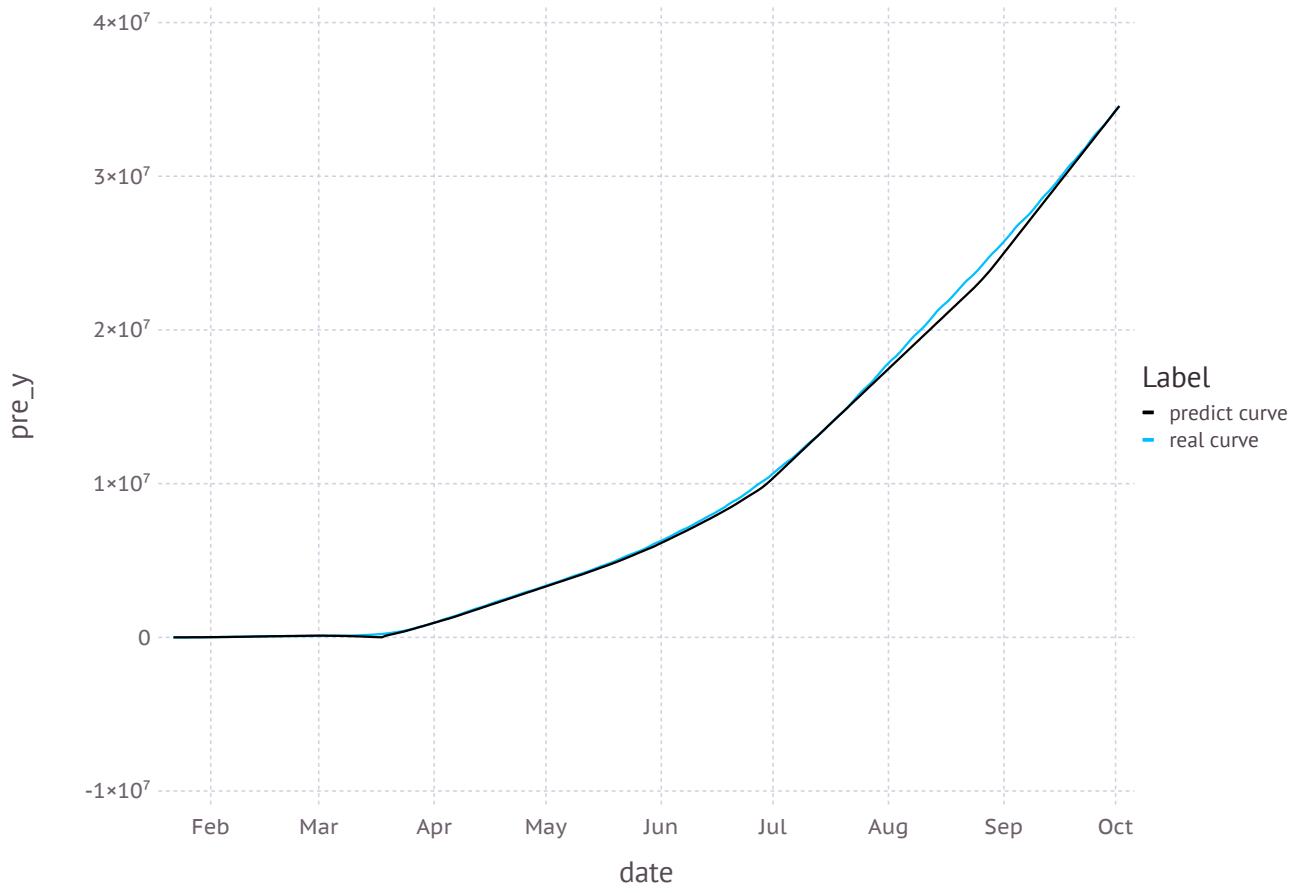
```
@with_kw mutable struct Args
    η::Float64 = 0.001
    batchsize::Int = 1
    epochs::Int = 1000
end
cd(@__DIR__)
pwd()
#read data:
dt, Header = readdlm("time_series_covid19_comfirmed_global.csv", ',', header=true)
dt_deaths, Header_deaths = readdlm("time_series_covid19_deaths_global.csv", ',', he
ader=true)
dt_recover,Header_recov = readdlm("time_series_covid19_recovered_global.csv",',',he
ader=true)


#create a new datafram to store datum
comfirmed_num = []
deaths_num = []
recovered_num = []
for i in 5:259
    push!(comfirmed_num, sum(dt[:, i]))
    push!(deaths_num, sum(dt_deaths[:, i]))
    push!(recovered_num, sum(dt_recover[:,i]))
end
dates = Date(2020, 1, 22):Day(1):Date(2020, 10, 2)
Global_num = DataFrame(Date=dates,
    comfirmed_num=comfirmed_num,
    deaths_num=deaths_num,
    recovered_num=recovered_num)

#read train_data
n, = size(comfirmed_num)
x = 0:n-1
y = Global_num.comfirmed_num
args = Args()
train_data = DataLoader((Array(x), Float64.(y)), batchsize=args.batchsize)

# define leaky relu
Lelu(x, α=100) = (x ≥ 0 ? x : x/α)

#define Model
m = Chain(
    Dense(1, 40, Lelu),
    Dense(40, 40, Lelu),
    Dense(40, 40, Lelu),
    Dense(40, 1, Lelu)
)

#define loss function
loss(x, y) = Flux.mse(m(x), y)
```

```
#define parameters
ps = Flux.params(m)

#define Opt
opt = ADAM(args.η)

#train model
Flux.train!(loss, ps, ncycle(train_data, args.epochs), opt)

#visualize
flux_y = []
for i in Array(x)
    push!(flux_y, Array(m([i]))[1])
end
flux_y = Float64.(flux_y)
plot(dates, [flux_y Float64.(y)],
    label=["predict" "real"], xlabel="date", ylabel="Numbers", size=(900, 600))
```

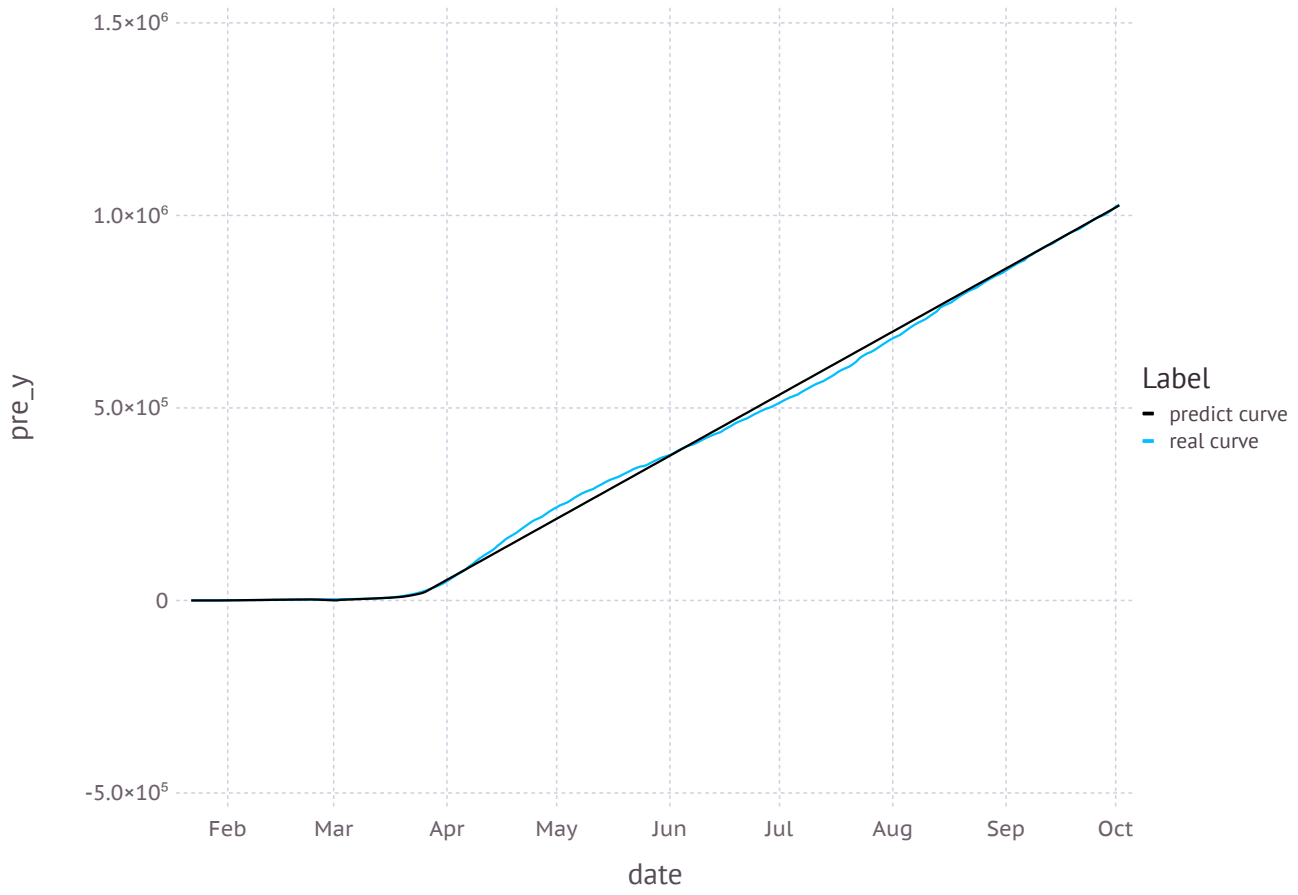
i write a function to include above code, named predict_curve.

```
predict_curve (generic function with 4 methods)
```

```
function predict_curve(x, y, η::Float64=0.001, epochs::Int=1000, batchsize::Int=1)
    dates = Date(2020, 1, 22):Day(1):Date(2020, 10, 2)
    train_data = Flux.Data.DataLoader((x, y), batchsize=batchsize)
    Lelu(x, α=100) = (x ≥ 0 ? x : x/α)
    m = Chain(
    Dense(1, 40, Lelu),
    Dense(40, 40, Lelu),
    Dense(40, 40, Lelu),
    Dense(40, 1, Lelu))
    loss(x, y) = Flux.mse(m(x), y)
    ps = Flux.params(m)
    opt = ADAM(η)
    Flux.train!(loss, ps, ncycle(train_data, epochs), opt)
    flux_y = []
    for i in Array(x)
        push!(flux_y, Array(m([i]))[1])
    end
    new_dt_pre = DataFrame(date=dates, real_y=Float64.(y), pre_y=Float64.(flux_y))
    plot(layer(new_dt_pre, x=:date, y=:pre_y,
Geom.line,Theme(default_color="black")),layer(new_dt_pre, x=:date, y=:real_y,
Geom.line),Guide.manual_color_key("Label", ["predict curve","real curve"],
["black","deepskyblue"],shape=[Shape.hline, Shape.hline]))

end
```




```
predict_curve(t, Float64.(Global_num.confirmed_num))
```



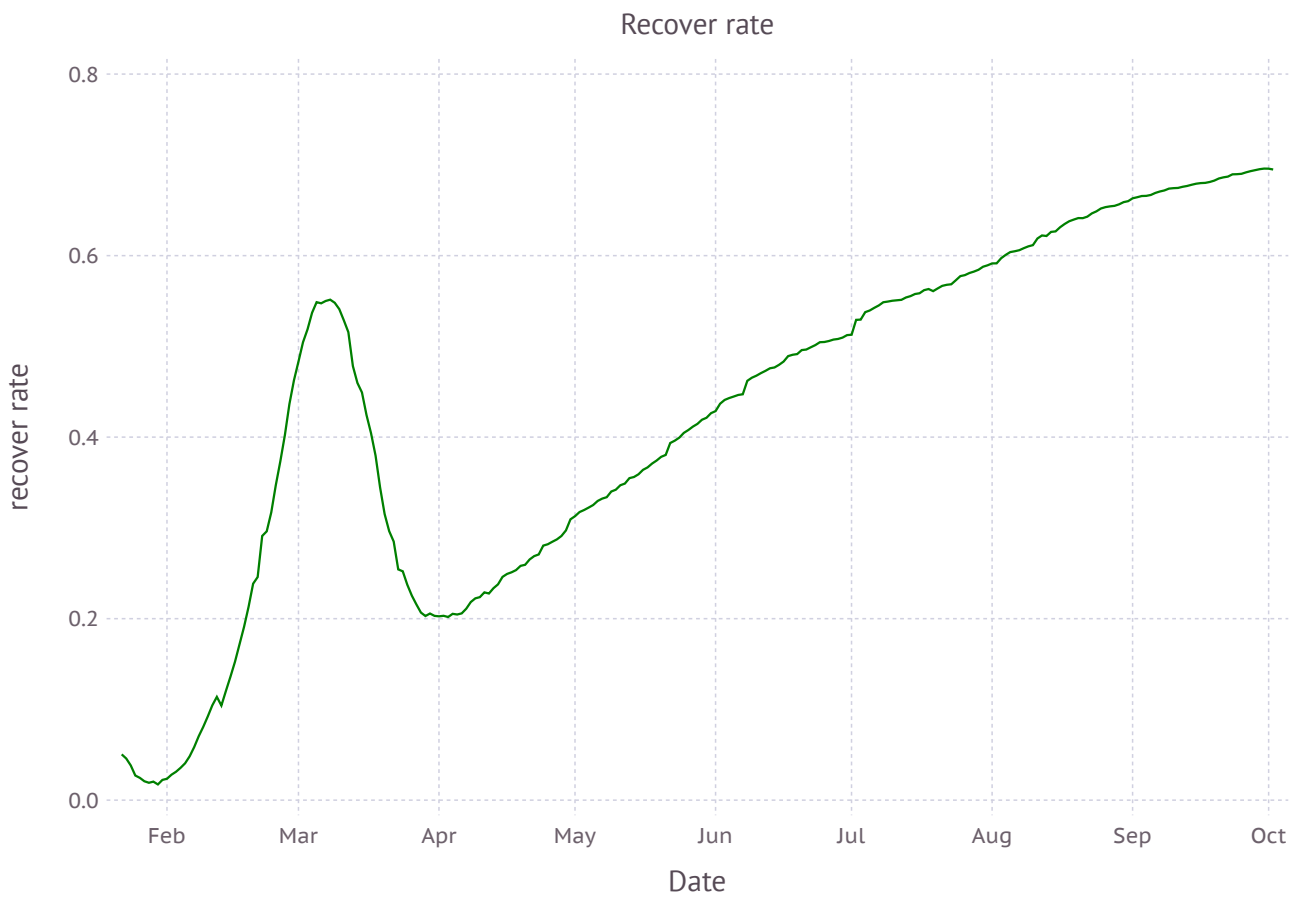
```
predict_curve(t, Float64.(Global_num.deaths_num))
```

I choose 2 hidden layer to build a neural network. each layer have 40 nuits. it will get a higher performence in fitting curve. but the model more complicated than linear regression.

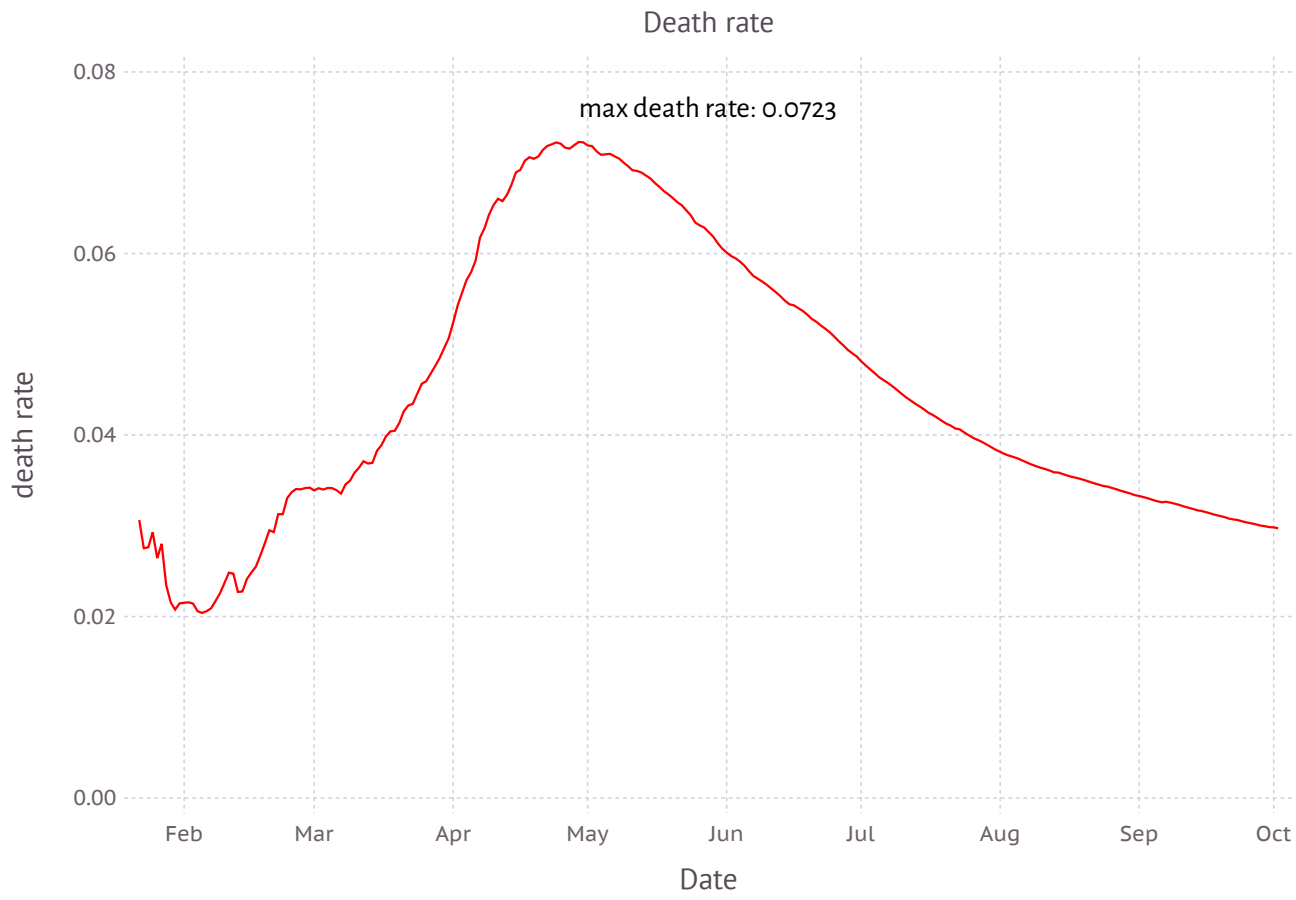
If there are no other special circumstances, the model can be used to make predictions.

> All models are wrong, but some are useful.

In the short term, I think you can use machine learning models for prediction, but linear regression is better in terms of the big trends.




```
plot(x=Global_num.Date,y = Global_num.recovered_num ./
Global_num.confirmed_num,Geom.line, Theme(default_color="green"),
Guide.xlabel("Date"), Guide.ylabel("recover rate"), Guide.title("Recover rate"))
```




```
plot(x=Global_num.Date,y = Global_num.deaths_num ./
Global_num.confirmed_num,Geom.line, Theme(default_color="red"),
Guide.xlabel("Date"), Guide.ylabel("death rate"), Guide.title("Death rate"),
Guide.annotation(compose(context(), Compose.text(Date(2020,4,29), 0.075, "max death
rate: 0.0723"))))
```

The *max death rate* is 0.0723 on April,29, 2020.now we want to see the Daily increase cases nums by using the Globa_num dataframe to calculate two new columns, names `confirmedin` and `recoverin` and `deathin`.

```
Int64[0, 1, 8, 16, 14, 26, 49, 2, 38, 42, 46, 103, 64, 66, 72, 70, 85, 87,
```

```
begin
    n, = size(confirmed_num)
    confirmedin = [0]
    deathin = [0]
    recoveredin = [0]
    for i in 2:n
        push!(confirmedin, confirmed_num[i]-confirmed_num[i-1])
        push!(deathin, deaths_num[i] - deaths_num[i-1])
        push!(recoveredin, recovered_num[i]-recovered_num[i-1])
    end
    Global_num.confirmedin = confirmedin
    Global_num.recoveredin = recoveredin
    Global_num.deathin = deathin
end
```

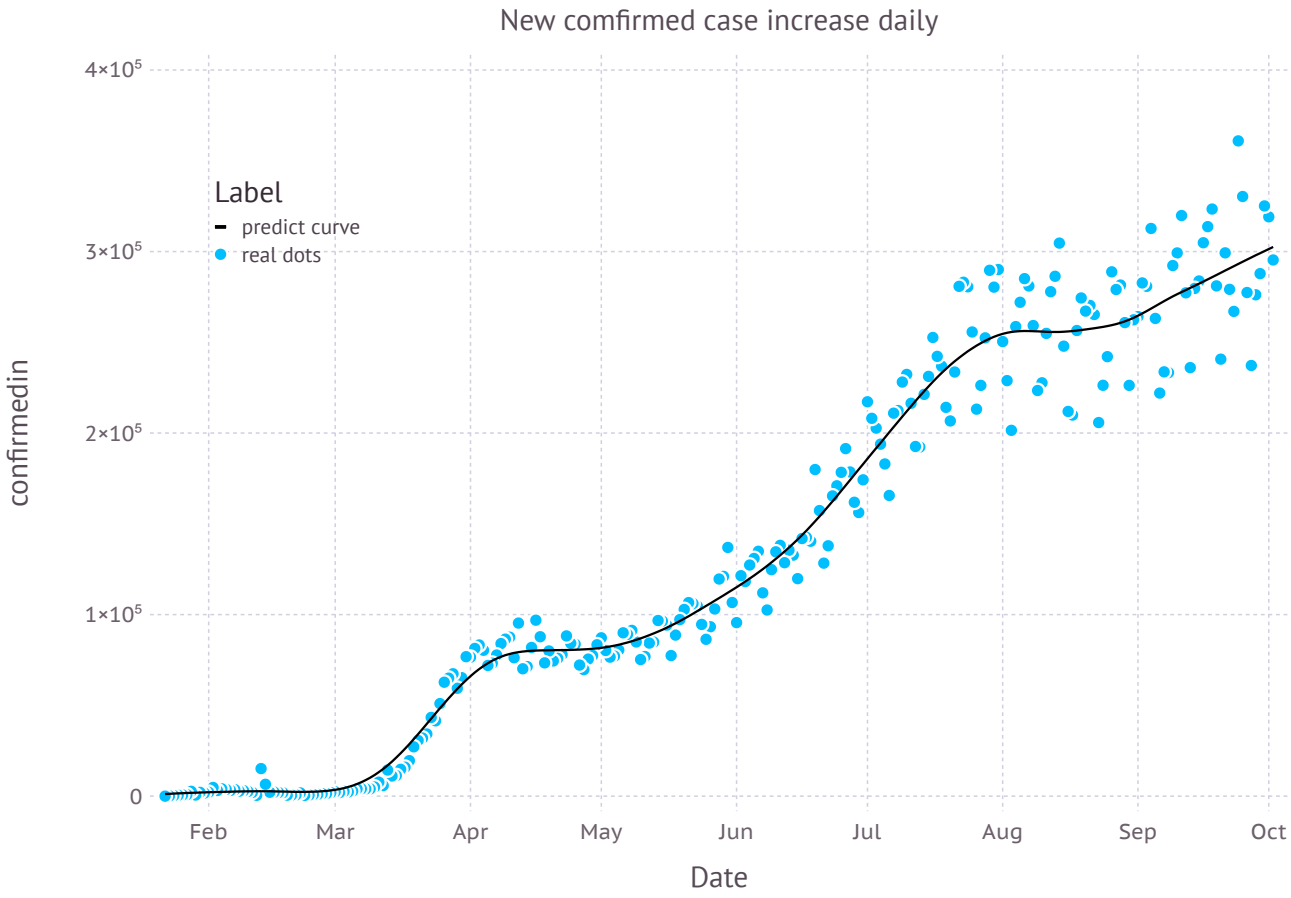
we plot the date and Daily increase numbers of confirmed, recovered and death. and use `loess` method to plot the fit curve on the plot.

As we can get the ***Max increase numbers*** here:

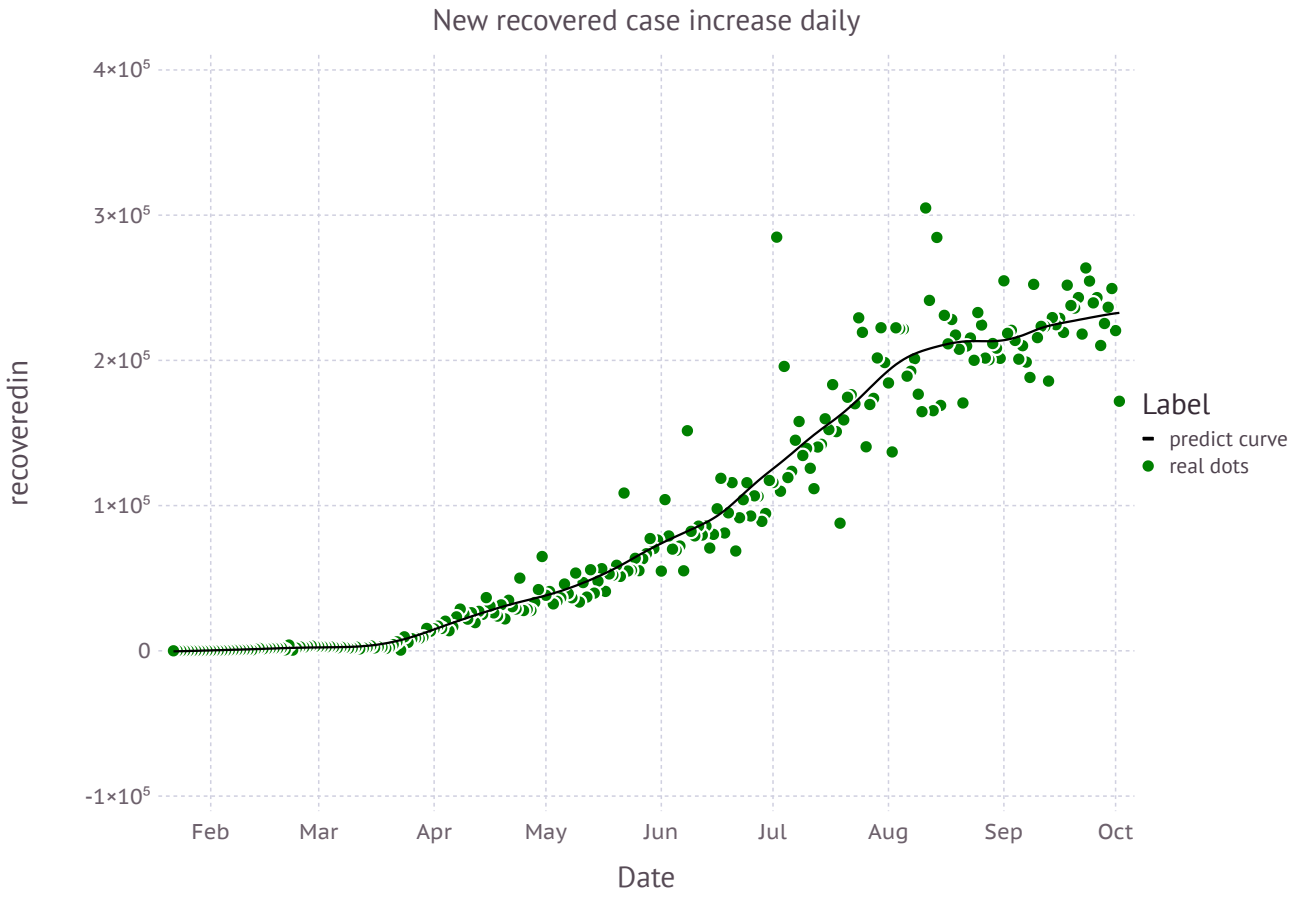
Confirmed: 360956

Recovered: 304953

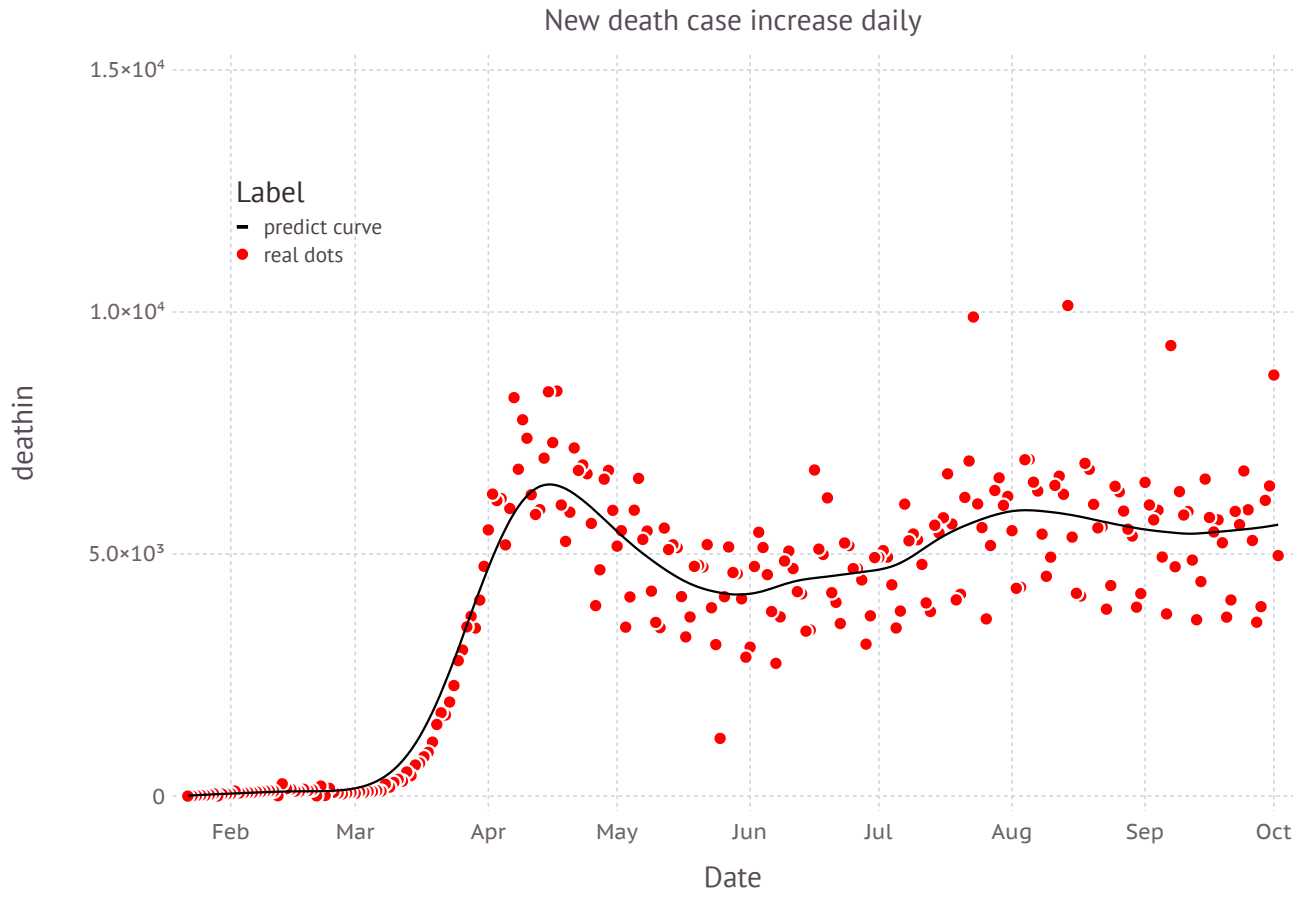
Death: 10134




```
plot(Global_num, x=:Date, y=:confirmedin,
layer(Geom.smooth(method=:loess,smoothing=0.2),
Theme(default_color="black")),Geom.point, Guide.title("New comfirmed case increase
daily"),Guide.manual_color_key("Label", ["predict curve","real dots"],
["black","deepskyblue"],shape=[Shape.hline, Shape.circle],pos=[0.05w,-0.28h]))
```




```
plot(Global_num, x=:Date,
y=:recoveredin,layer(Geom.smooth(method=:loess,smoothing=0.2),
Theme(default_color="black")), Geom.point,
Theme(default_color="green"),Guide.title("New recovered case increase
daily"),Guide.manual_color_key("Label", ["predict curve","real dots"],
["black","green"],shape=[Shape.hline, Shape.circle]))
```




```
plot(Global_num, x=:Date, y=:deathin,
layer(Geom.smooth(method=:loess,smoothing=0.2),
Theme(default_color="black")),Geom.point,Theme(default_color="red"),Guide.title("New death case increase daily"),Guide.manual_color_key("Label", ["predict curve","real dots"], ["black","red"],shape=[Shape.hline, Shape.circle],pos=[0.05w,-0.28h]))
```

if we look at the trend of the fitting curve, I would argue that the increase in the number of cases has not slowed down.

```
md"***"
```

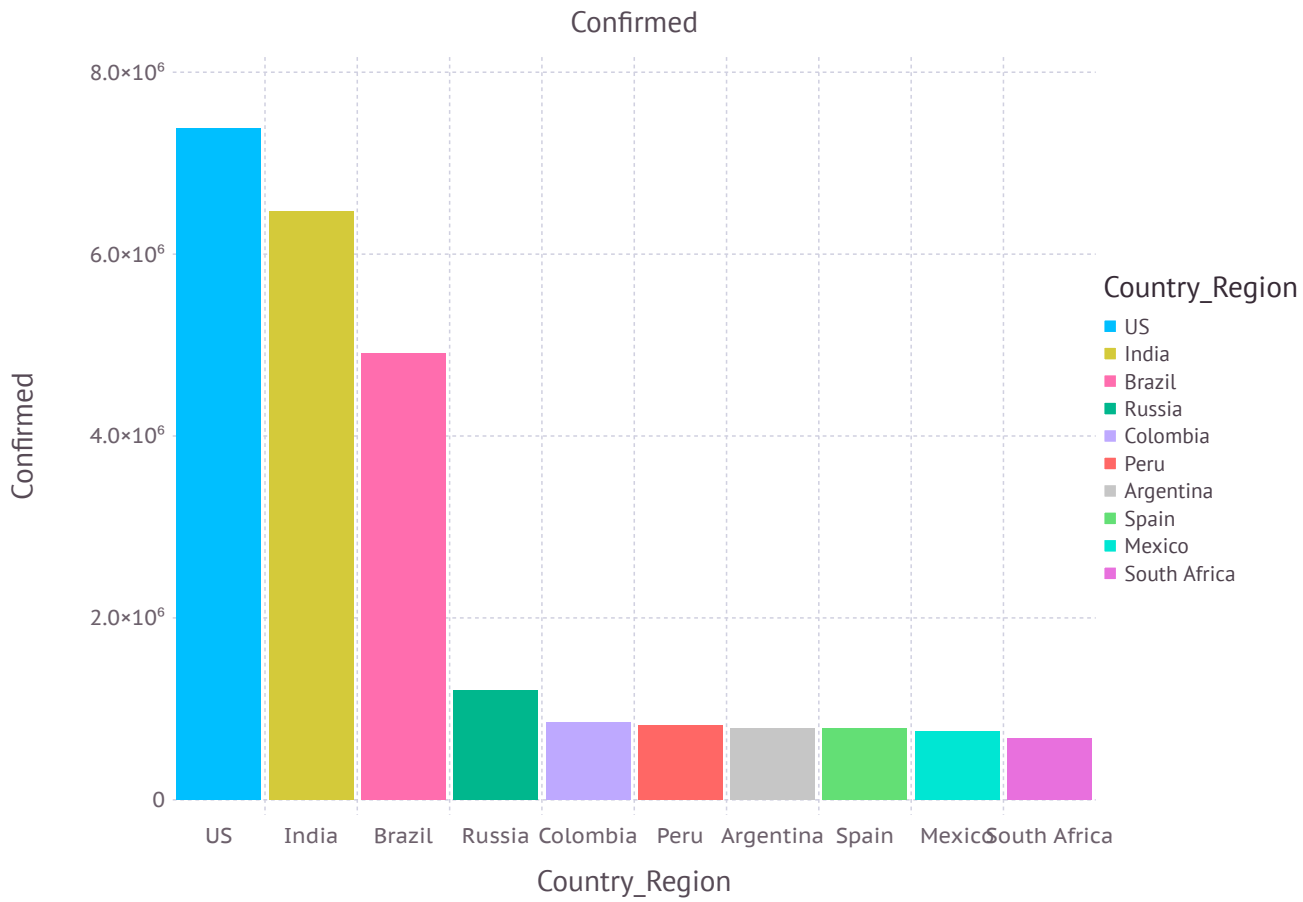
top_case =

10 rows × 14 columns (omitted printing of 9 columns)

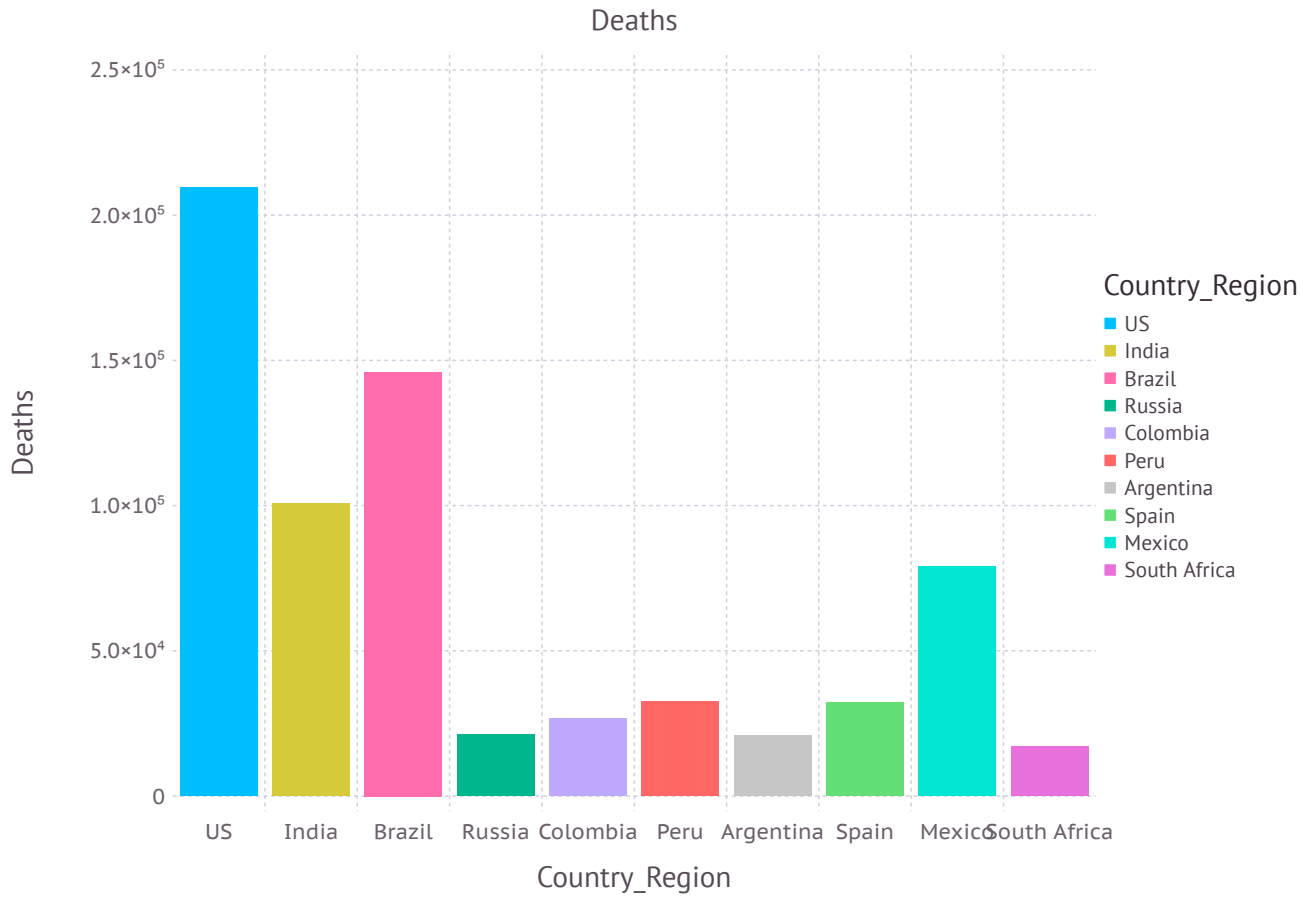
	Country_Region String	Last_Update String	Lat Float64?	Long_ Float64?	Confirmed Float64
1	US	2020-10-04 04:23:44	40.0	-100.0	7.38219e6
2	India	2020-10-04 04:23:44	20.5937	78.9629	6.47354e6
3	Brazil	2020-10-04 04:23:44	-14.235	-51.9253	4.90683e6
4	Russia	2020-10-04 04:23:44	61.524	105.319	1.19866e6
5	Colombia	2020-10-04 04:23:44	4.5709	-74.2973	848147.0
6	Peru	2020-10-04 04:23:44	-9.19	-75.0152	821564.0
7	Argentina	2020-10-04 04:23:44	-38.4161	-63.6167	790818.0

	Country_Region String	Last_Update String	Lat Float64?	Long_ Float64?	Confirmed Float64
8	Spain	2020-10-04 04:23:44	40.4637	-3.74922	789932.0
9	Mexico	2020-10-04 04:23:44	23.6345	-102.553	757953.0
10	South Africa	2020-10-04 04:23:44	-30.5595	22.9375	679716.0

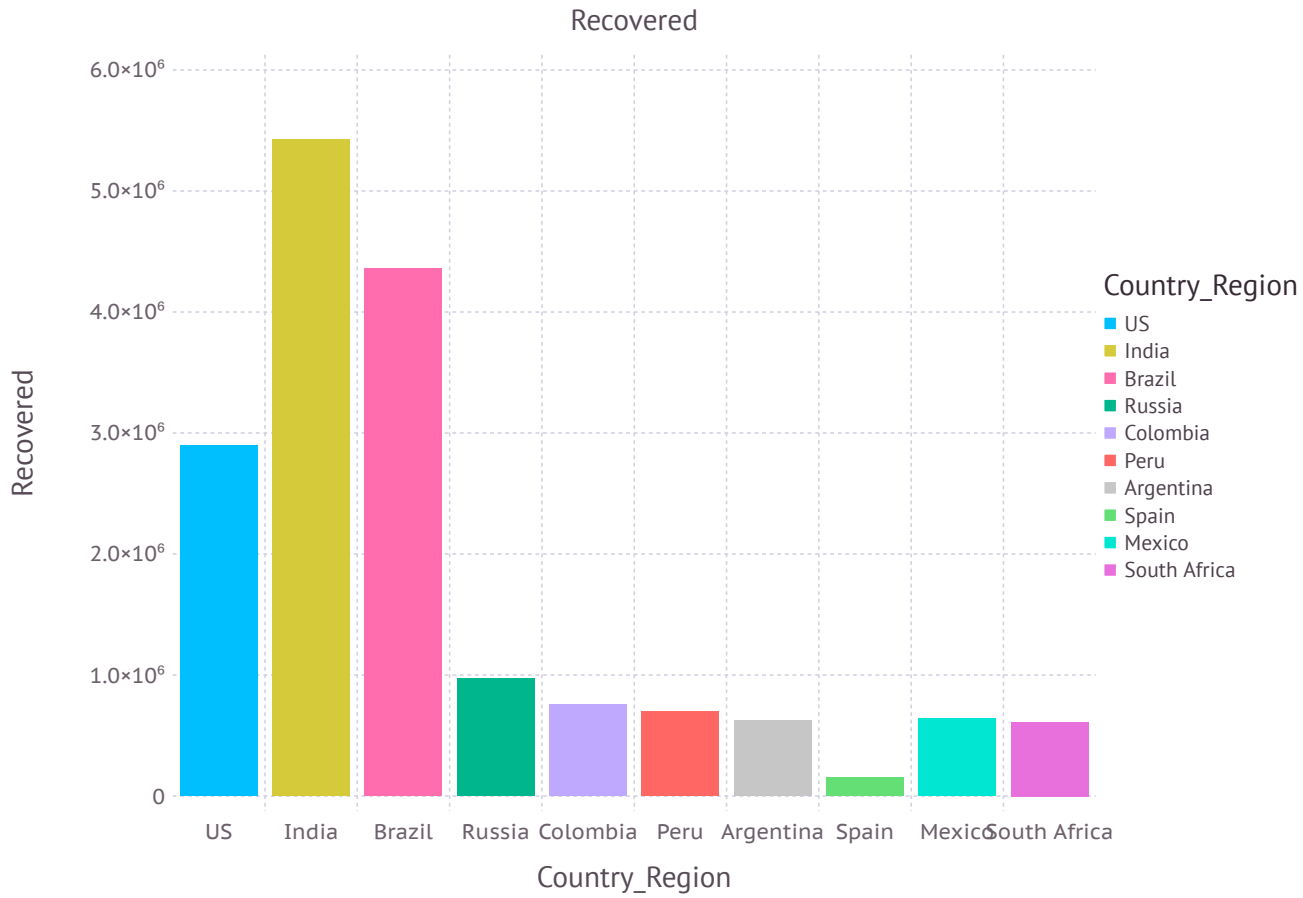
```
top_case = sort(CSV.read("cases_country.csv"), :Confirmed, rev=true)[1:10,:]
```




```
plot(top_case, x=:Country_Region, y=:Confirmed, Geom.bar,
color=:Country_Region,Theme(bar_spacing=1mm), Guide.title("Confirmed"))
```




```
plot(top_case, x=:Country_Region, y=:Deaths, Geom.bar,
color=:Country_Region,Theme(bar_spacing=2mm),Guide.title("Deaths"))
```




```
plot(top_case, x=:Country_Region, y=:Recovered, Geom.bar,
color=:Country_Region,Theme(bar_spacing=2mm), Guide.title("Recovered"))
```

time_case_country =

266 rows × 259 columns (omitted printing of 253 columns)

	Province/State String?	Country/Region String	Lat Float64	Long Float64	1/22/20 Int64	1/23/20 Int64
1	*missing*	Afghanistan	33.9391	67.71	0	0
2	*missing*	Albania	41.1533	20.1683	0	0
3	*missing*	Algeria	28.0339	1.6596	0	0
4	*missing*	Andorra	42.5063	1.5218	0	0
5	*missing*	Angola	-11.2027	17.8739	0	0
6	*missing*	Antigua and Barbuda	17.0608	-61.7964	0	0
7	*missing*	Argentina	-38.4161	-63.6167	0	0
8	*missing*	Armenia	40.0691	45.0382	0	0
9	Australian Capital Territory	Australia	-35.4735	149.012	0	0
10	New South Wales	Australia	-33.8688	151.209	0	0
11	Northern Territory	Australia	-12.4634	130.846	0	0
12	Queensland	Australia	-27.4698	153.025	0	0
13	South Australia	Australia	-34.9285	138.601	0	0

	Province/State String?	Country/Region String	Lat Float64	Long Float64	1/22/20 Int64	1/23/20 Int64
14	Tasmania	Australia	-42.8821	147.327	0	0
15	Victoria	Australia	-37.8136	144.963	0	0
16	Western Australia	Australia	-31.9505	115.861	0	0
17	*missing*	Austria	47.5162	14.5501	0	0
18	*missing*	Azerbaijan	40.1431	47.5769	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮

```
time_case_country = CSV.read("time_series_covid19_confirmed_global.csv")
```

266 rows × 256 columns (omitted printing of 246 columns)

	1/22/20 Int64	1/23/20 Int64	1/24/20 Int64	1/25/20 Int64	1/26/20 Int64	1/27/20 Int64	1/28/20 Int64	1/29/20 Int64	1/30/20 Int64	1/31/2 Int6
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	3	4	4	4	4	4
11	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	1	3	2
13	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	1	1	1	1	2	3
16	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

```
begin
    time_case_country.Country_Region = time_case_country[:, 2]
    t_case_country = time_case_country[:,5:end]
end
```

Q =

67,830 rows × 3 columns

	Country_Region String	variable Cat…	value Int64
1	Afghanistan	1/22/20	0
2	Albania	1/22/20	0
3	Algeria	1/22/20	0
4	Andorra	1/22/20	0
5	Angola	1/22/20	0
6	Antigua and Barbuda	1/22/20	0
7	Argentina	1/22/20	0
8	Armenia	1/22/20	0
9	Australia	1/22/20	0
10	Australia	1/22/20	0
11	Australia	1/22/20	0
12	Australia	1/22/20	0
13	Australia	1/22/20	0
14	Australia	1/22/20	0
15	Australia	1/22/20	0
16	Australia	1/22/20	0
17	Austria	1/22/20	0
18	Azerbaijan	1/22/20	0
⋮	⋮	⋮	⋮

```
Q = stack(t_case_country, Not(:Country_Region))
```

case_time_series_country_top =

2,550 rows × 3 columns

	Country_Region String	variable Cat…	value Int64
1	Argentina	1/22/20	0
2	Brazil	1/22/20	0
3	Colombia	1/22/20	0
4	India	1/22/20	0
5	Mexico	1/22/20	0
6	Peru	1/22/20	0
7	Russia	1/22/20	0
8	South Africa	1/22/20	0
9	Spain	1/22/20	0
10	US	1/22/20	1
11	Argentina	1/23/20	0
12	Brazil	1/23/20	0

	Country_Region String	variable Cat…	value Int64
13	Colombia	1/23/20	0
14	India	1/23/20	0
15	Mexico	1/23/20	0
16	Peru	1/23/20	0
17	Russia	1/23/20	0
18	South Africa	1/23/20	0
⋮	⋮	⋮	⋮

```
case_time_series_country_top = Q |> @filter(_.Country_Region in top_case.Country_Region) |> DataFrame
```

```
String["1/22/20", "1/22/20", "1/22/20", "1/22/20", "1/22/20", "1/22/20", "1/22/20"
```

```
case_time_series_country_top.Date = String.(case_time_series_country_top[:,2])
```

2,550 rows × 4 columns

	Country_Region String	variable Cat…	value Int64	Date String
1	Argentina	1/22/20	0	1/22/20
2	Brazil	1/22/20	0	1/22/20
3	Colombia	1/22/20	0	1/22/20
4	India	1/22/20	0	1/22/20
5	Mexico	1/22/20	0	1/22/20
6	Peru	1/22/20	0	1/22/20
7	Russia	1/22/20	0	1/22/20
8	South Africa	1/22/20	0	1/22/20
9	Spain	1/22/20	0	1/22/20
10	US	1/22/20	1	1/22/20
11	Argentina	1/23/20	0	1/23/20
12	Brazil	1/23/20	0	1/23/20
13	Colombia	1/23/20	0	1/23/20
14	India	1/23/20	0	1/23/20
15	Mexico	1/23/20	0	1/23/20
16	Peru	1/23/20	0	1/23/20
17	Russia	1/23/20	0	1/23/20
18	South Africa	1/23/20	0	1/23/20
⋮	⋮	⋮	⋮	⋮

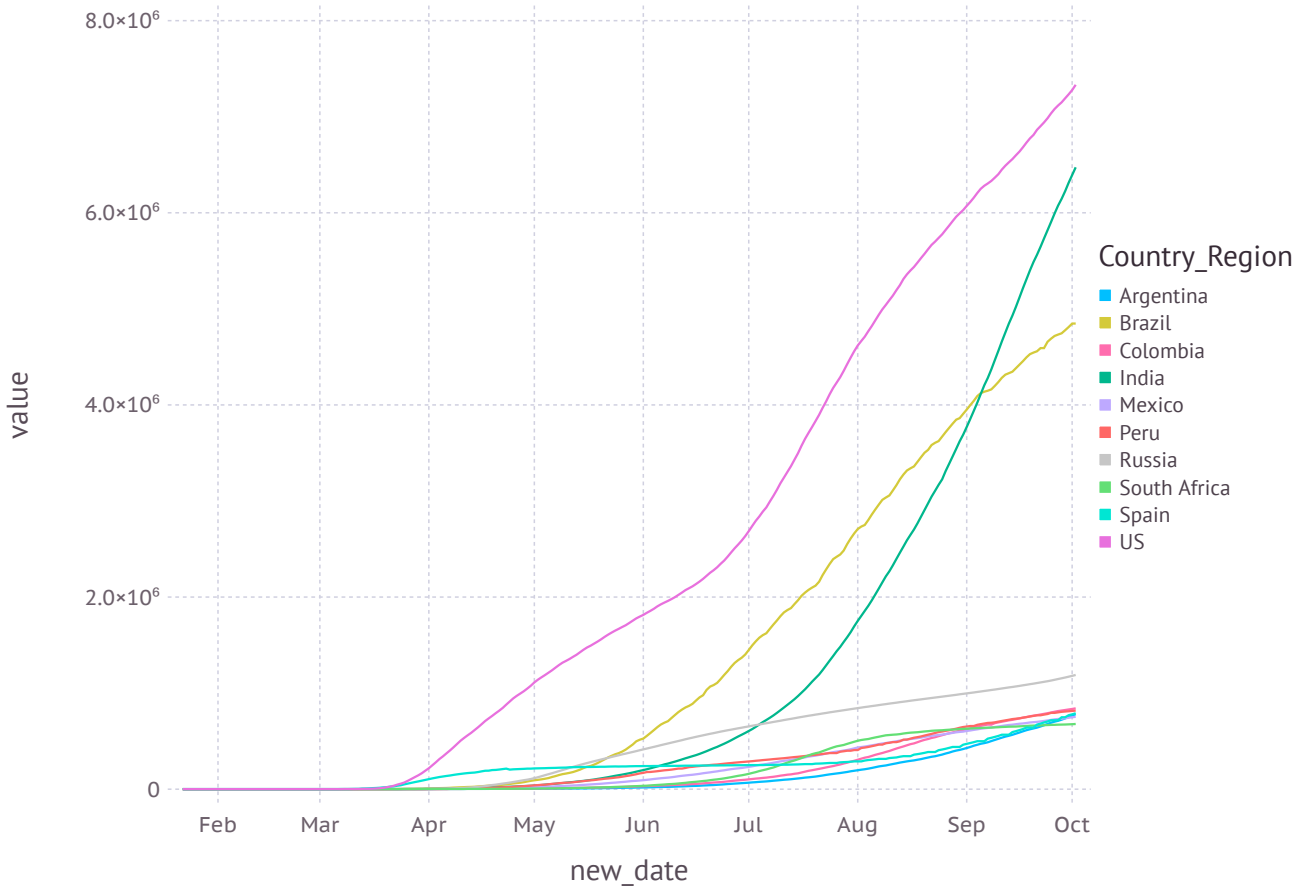
```
case_time_series_country_top
```

```
change_date (generic function with 1 method)
```

```
function change_date(x)
    mon = parse(Int64, String(split(string(x, "20"),"/")[1]))
    day = parse(Int64, String(split(string(x, "20"),"/")[2]))
    year = parse(Int64, String(split(string(x, "20"),"/")[3]))
    return Date(year,mon, day)
end

```

```
Date[2020-01-22, 2020-01-22, 2020-01-22, 2020-01-22, 2020-01-22, 2020-01-22, 2020-
```

```
case_time_series_country_top.new_date = change_date.
(case_time_series_country_top.Date)
```




```
plot(case_time_series_country_top, x=:new_date,
y=:value,color=:Country_Region,Geom.line)
```

If we focus on **Global Democracy Index**(data from **wiki**), we can get a table. then someting interesting here. in the top10 confirmed case countries, i show here:

top_case_democracy =
10 rows × 3 columns

	Country String	Confirmed Float64	Democracy_index Float64
1	US	7.38219e6	7.96
2	India	6.47354e6	6.9

	Country String	Confirmed Float64	Democracy_index Float64
3	Brazil	4.90683e6	6.86
4	Russia	1.19866e6	3.11
5	Colombia	848147.0	7.13
6	Peru	821564.0	6.6
7	Argentina	790818.0	7.02
8	Spain	789932.0	8.29
9	Mexico	757953.0	6.09
10	South Africa	679716.0	7.24

```
democracy =  Float64[7.96, 6.9, 6.86, 3.11, 7.13, 6.6, 7.02, 8.29, 6.09, 7.24]
```

The top10 confirmed case except Russia, The democracy index is high. China democracy index is 2.26, but The number of confirmed cases is low and the mortality rate is low， even though China has the world's largest population。 😂