赛区评阅编号（由赛区组委会填写）：

2021 高教社杯全国大学生数学建模竞赛

承　诺　书

我们仔细阅读了《全国大学生数学建模竞赛章程》和《全国大学生数学建模竞赛参赛规则》（以下简称“竞赛章程和参赛规则”，可从 http://www.mcm.edu.cn 下载）。

我们完全清楚，在竞赛开始后参赛队员不能以任何方式，包括电话、电子邮件、“贴吧”、QQ 群、微信群等，与队外的任何人（包括指导教师）交流、讨论与赛题有关的问题；无论主动参与讨论还是被动接收讨论信息都是严重违反竞赛纪律的行为。

我们完全清楚，在竞赛中必须合法合规地使用文献资料和软件工具，不能有任何侵犯知识产权的行为。否则我们将失去评奖资格，并可能受到严肃处理。

我们以中国大学生名誉和诚信郑重承诺，严格遵守竞赛章程和参赛规则，以保证竞赛的公正、公平性。如有违反竞赛章程和参赛规则的行为，我们将受到严肃处理。

我们授权全国大学生数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

我们参赛选择的题号（从 A/B/C/D/E 中选择一项填写）：A

我们的报名参赛队号（12 位数字全国统一编号）：4321

参赛学校（完整的学校全称，不含院系名）：天津大学

参赛队员 (打印并签名) ：1. 谢远峰

2. 元祥渊

3. 陈柯亘

指导教师或指导教师组负责人 (打印并签名)：王宏

（指导教师签名意味着对参赛队的行为和论文的真实性负责）

日期: 2021 年 09 月 15 日

（请勿改动此页内容和格式。此承诺书打印签名后作为纸质论文的封面，注意电子版论文中不得出现此页。以上内容请仔细核对，如填写错误，论文可能被取消评奖资格。）

赛区评阅编号：
（由赛区填写）_____

全国评阅编号：
（全国组委会填写）_____

2021 高教社杯全国大学生数学建模竞赛

编 号 专 用 页

赛区评阅记录（可供赛区评阅时使用）：

评阅人						
备注						

送全国评阅统一编号:
（赛区组委会填写）

（请勿改动此页内容和格式。此编号专用页仅供赛区和全国评阅使用，参赛队打印后装订到纸质论文的第二页上。注意电子版论文中不得出现此页。）

全国大学生数学建模竞赛

摘要

目的：本篇文章基于传染病 SEIRD 模型研究美国阿肯色州 COVID-19 发展趋势，使用 python 工具建立传染病模型，仿真揭示美国阿肯色州从 2020 年 3 月 6 日到 2021 年 3 月 6 日的疫情情况。

方法：研究选取 covid_tracking.csv 数据集，通过 python 分析数据，使用微分方程传播模型求解。最终得出仿真模拟预测结果，与实际值进行比较。

结论：我们针对美国阿肯色州的病毒传播情况进行建模并对比观察。我们认为严格的疫情防控和控制移动范围、控制聚集等活动对 COVID-19 的传染起到了良性作用，为了减小感染病毒可能性，减少大型集会、减少跨区域移动对防控病毒的传播起到有效作用。就疫苗接种而言，各州从有效率角度考虑，需尽可能多选择辉瑞和 Moderna 疫苗，并减少聚集。

关键字： COVID-19　Python　SEIRD 模型

目录

一、 数学模型的建立和分析

1.1 假设与符号说明

对于该数学问题我们有以下几点假设：

- 疫情统计数据是可靠的
- 病人处于潜伏期时无传染性，潜伏者均会转变为感染者，感染后康复不会再被感染
- 选定地区迁入和迁出人口大致相等，不考虑出生和自然死亡，即总人口基本不变
- 人群均匀混合。任何感染者以概率 α 接触其它易感者，概率用总体均值替代
- 病毒处于稳态，未发生变异，疾病的传播率固定
- 假设人群中只能通过感染新冠病毒进行患病
- 死亡人群不再具有感染性，从感染人群中剔除
- S: 易感人群；E: 传染人群；I: 感染人群；D: 死亡人群；R: 康复人群
- β: 传染率；σ: 发病率；μ: 死亡率；γ: 康复率

1.2 分析问题

2020 年威胁人类健康的新型冠状病毒传染率高、无特定易感人群、有一定致死率，已经在全球范围传播。美国政府在初期不作为，导致了国内人民遭受了大面积的疫情袭击。我们选取了阿肯色州（Arkansas）作为目标，对该地区的疫情情况进行建模和分析。考虑到新冠肺炎具有明显的潜伏期以及高于一般传染病的致死率，基于基本的 SIR 模型，调整采用更适用于恶性传染病的 SEIRD 模型；并在完成模型构建的基础上，与已有的疫情统计数据相结合，对阿肯色州地区的疫情作出分析。

1.3 基础模型

1.4 建立数学模型

新冠肺炎没有特定的易感人群，但是可以确定的是免疫力低下的人群和接触人员更多的人群更容易患病，因此我们将这一部分人设置为 S，即该模型中的易感人群；使用 E 表示被病毒传染的人群，使用 I 来表示感染人群，潜伏人群 E 经过一定的潜伏期后均会变成感染人群 S。D 表示死亡人群，R 表示康复人群。β 是传染率，反映病毒传播的快慢；σ 为发病率，表示潜伏者转变成感染者的概率，它的倒数是潜伏期天数；μ 为死亡率，表示的是感染者因新冠肺炎死亡的概率；γ 为康复率，表示感染者恢复健康的概率。

1. $\frac{dS}{dt} = -\frac{\beta SI}{N}$: 易感者随时间变化
2. $\frac{dE}{dt} = \frac{\beta SI}{N} - \sigma E$: 传染者（潜伏期）E 随时间 t 变化
3. $\frac{dI}{dt} = \sigma E - \gamma I - \mu I$: 感染者随时间 t 的变化
4. 感染者随时间发展转变为两部分，死亡者 D/康复者 R
5. $\frac{dR}{dt} = \gamma I$: 康复者随时间 t 的变化
6. $\frac{dD}{dt} = \mu I$: 死亡者随时间 t 的变化
7. N=S+E+I+R+D: 模型涉及的人口总和

其中 β 参数与系数 R0 成正比。R0 通常由传染病学家分析疫情特定情况得出，表示的是疾病的传染性，R0 决定了一个感染者可以将疾病传染给多少个人。若 R0<1，则传染有希望停止；若 R0=1，则传染是较稳定的或者局限在一个地方；若 R0>1，则疾病在没有外界的干预下将会扩散。

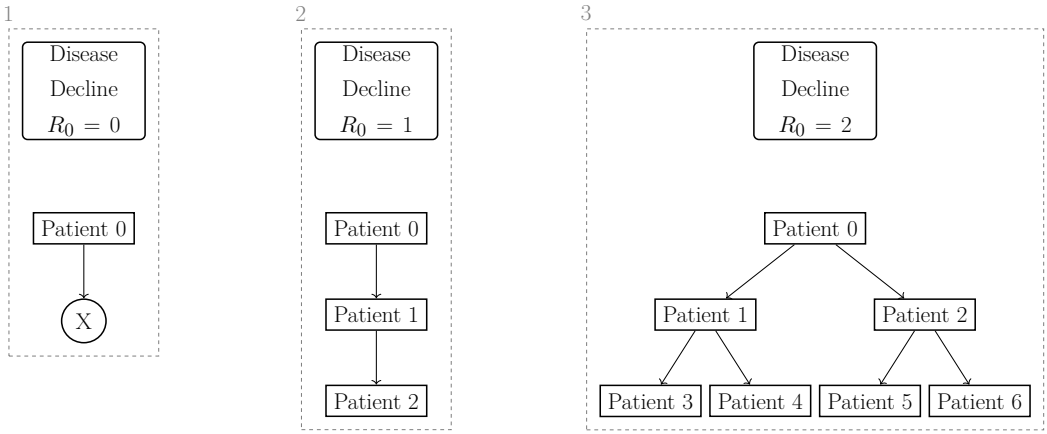


图 **1**　传染模型搭建

其余参数 sigma(σ)，mu(μ)，gamma(γ) 均可以通过临床数据得出。

在各个国家和地区的参数都不相同，需要根据当地情况调整。如：政府发布戴口罩、限制出行的政策可以降低易感者 S_0，封锁城市减少人员流动会降低传染率 b，疫苗的产生和治疗方法的成熟会显著提升治愈率 g。

在特定的假设条件下，我们完成了 SEIRD 模型的构建：

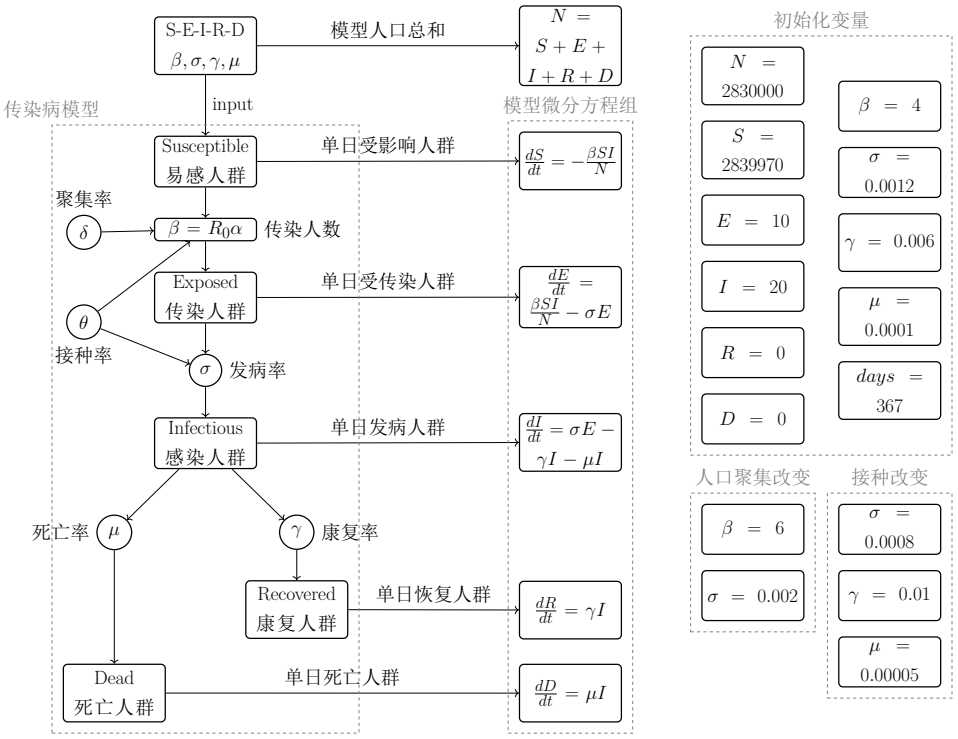


图 **2**　**SEIRD** 模型流程图及超参参考值

1.5 数据集的获取与筛选

需要获取的目标数据集是阿肯色州的官方疫情统计数据，Microsoft Azure 提供了 COVID-19 Data Lake，包含了诸多地区的新冠肺炎相关数据集。我们在获取的数据集发现该数据集涵盖了一些对于本次建模工作不重要的信息，因此需要进行针对性筛选。对下载好的 csv 文件进行清洗，最终得到了 2020 年 3 月 6 日到 2021 年 3 月 6 日阿肯色州的疫情统计数据，并且获取了拟合模型所需要的参数。

1.5.1 数据集的选取

为建立 SEIRD 模型，在允许范围拥有数据类型较为符合要求的数据集。通过比对各官方网站上不同来源的数据。最终选定的数据集属性表如下

属性	名称	属性	名称	属性	名称
state	州	n_icu_currently	ICU 离开人数	date_checked	核对日期
positive	阳性	on_ventilator_currently	使用呼吸机数	death	死亡人数
negative	阴性	on_ventilator_cumulative	总计使用呼吸机数	hospitalized	住院人数
pending	待检测	recovered	康复人数	total_test_results	总计检测人数
hospitalized_currently	当前住院人数	data_quality_grade	数据可信度	pos_neg	检测阳阴性比例
hospitalized_cumulated	总计住院人数	last_update_et	最近更细日期	fips	联邦信息管理
death_increase	单日新增死亡人数	hospitalized_increase	单日新增住院人数	negative_increase	单日新增阴性人数
positive_increase	检测阳性人数	total_test_results_increase	单日新增检测数	fips_code	管理代号

表 **1**　数据集属性浏览表

1.5.2 数据集的清洗

A 相似值处理

很多特征值之间具有相关性，这些特征并不需要全部关注。比如 death，death_increase，在该模型的处理过程中只需要关注 death 值。

B 无关值处理

有一些变量在该模型中并没有影响效用。比如 positive、negative、negative_increase、positive_increase，这些本身并不需要被关注，因此可以排除在外。

C 密切值处理

我们是针对一个特定地区（例如本队选取阿肯色州进行分析），因此在 state 类别的数据我们只需要筛选标注 AR 的数据行。

1.5.3 人工检查

人工提取的过程较为繁琐，主观性较强，但是人工提取可以保证模型数据选取的准确性，在对庞大数据进行降维的过程，人工往往是比较有效的方法。通过进行缺失值统计，利用零值进行填充，利用绘图函数进行历史数据的绘制

1.6 调整模型参数

经过上述步骤，获得的最终分析数据属性如下

1. date: 时间记录
2. positive: 单日病毒检测阳性人群数量
3. recovered: 单日康复人群数量
4. death: 单日死亡人群数量

根据疫情的发展趋势，调整比对得出初始数据

1.7 代码解释

```
# 微分方程模型求解
def ode_model(z, t, beta, sigma, gamma, mu):
    S, E, I, R, D = z           # z 为数组装载了 5 个元素
    N = S + E + I + R + D       # 总人数
    dS/dt = -beta * S * I / N
    dE/dt = beta * S * I / N - sigma * E
    dI/dt = sigma * E - gamma * I - mu * I
    dR/dt = gamma * I
    dD/dt = mu * I
    return [dS/dt, dE/dt, dI/dt, dR/dt, dD/dt]
```

```
# scipy 使用 odeint 函数求解微分方程
def ode_solver(t, initial_conditions, params):
    initE, initI, initR, initN, initD = initial_conditions
    beta, sigma = params['beta'].value, params['sigma'].value
    gamma, mu = params['gamma'].value, params['mu'].value
    initS = initN - (initE + initI + initR + initD)
    res = odeint(ode_model, [initS, initE, initI, initR,initD], t, args=(beta, sigma, gamma, mu))
    return res
```

```python
# 演示效果动态调参
def main(initE, initI, initR, initD, initN, beta, sigma, gamma, mu, days, param_fitting):
    initial_conditions = [initE, initI, initR, initN, initD]
    params['beta'].value,params['sigma'].value,params['gamma'].value,params['mu'].value = [beta,sigma,gamma,mu]
    tspan = np.arange(0, days, 1)   # 横坐标
    sol = ode_solver(tspan, initial_conditions, params) # 返回二维数组，6 列
    S, E, I, R, D = sol[:, 0], sol[:, 1], sol[:, 2], sol[:, 3], sol[:, 4]   # 前面所有行，逗号后面是所有列

    fig = go.Figure()   # 动态图，绘图
    fig1 = go.Figure()
    if not param_fitting:   # 对应按钮，改变或不改变
        fig.add_trace(go.Scatter(x=tspan, y=S, mode='lines+markers', name='Susceptible'))
        fig.add_trace(go.Scatter(x=tspan, y=E, mode='lines+markers', name='Exposed'))
        fig.add_trace(go.Scatter(x=tspan, y=I, mode='lines+markers', name='Infected'))
        fig.add_trace(go.Scatter(x=tspan, y=R, mode='lines+markers',name='Recovered'))
        fig.add_trace(go.Scatter(x=tspan, y=D, mode='lines+markers',name='Death'))
        fig1.add_trace(go.Scatter(x=tspan, y=I, mode='lines+markers', name='Infected'))
        fig1.add_trace(go.Scatter(x=tspan, y=R, mode='lines+markers',name='Recovered'))
        fig1.add_trace(go.Scatter(x=tspan, y=D, mode='lines+markers',name='Death'))
    if param_fitting:
        fig.add_trace(go.Scatter(x=tspan, y=df_temp2.positive, mode='lines+markers', name='Infections Observed',
    ↪  line=dict(dash='dash')))
        fig.add_trace(go.Scatter(x=tspan, y=df_temp2.recovered, mode='lines+markers', name='Recovered Observed',
    ↪  line=dict(dash='dash')))
        fig.add_trace(go.Scatter(x=tspan, y=df_temp2.death, mode='lines+markers', name='Deaths Observed',
    ↪  line=dict(dash='dash')))

    if days <= 30:  # 对时长进行步长设置
        step = 1
    elif days <= 90:
        step = 7
    else:
        step = 30

    # 横坐标和纵坐标名称图片宽度高度
    fig.update_layout(title='Simulation of SEIRD Model', xaxis_title='Day', yaxis_title='Counts', title_x=0.5,
    ↪  width=900, height=600)
    fig.update_xaxes(tickangle=-90, tickformat = None, tickmode='array', tickvals=np.arange(0, days + 1, step))
    fig1.update_layout(title='SEIRD test',xaxis_title='Day', yaxis_title='Counts', title_x=0.5, width=1000,
    ↪  height=600)
    if not os.path.exists("images"):
        os.mkdir("images")  # 创建图片所在文件夹
        fig.write_image("images/seird_simulation.png")  # 生成图像
        fig.show()   # 展示图像
```

1.8 分析阿肯色州 COVID_19 病毒的传播情况

1.8.1 初步分析拟合

图 **3**　参数调整可视化图像

1.8.2 使用 L-M 算法进行优化

连续历元权重变化公式：$\Delta W_m = -\dfrac{d_m}{(H_m + e^{\lambda} I)}$ 在 python 中使用 lmfit 库中的最小化模块，并运用 Levenberg-Marquardt 算法来最小化非线性最小二乘。对残差编码进行并将其限制在最小值。优化完成后，生成对应模型对比图和优化后的模型参数：

图 **4**　优化参数可视化对比图

```
[[Variables]]
beta:   2.62206596  +/- 0.07065303  (2.69%) (init = 4)
sigma:  0.00119439  +/- 2.0997e-05  (1.76%) (init = 0.0012)
gamma:  0.01052104  +/- 1.6378e-04  (1.56%) (init = 0.006)
mu: 5.4349e-11  +/- 1.8250e-05  (33578621.44%) (init = 0.0001)
[[Correlations]] (unreported correlations are < 0.100)
C(beta, sigma) = -0.953     C(beta, gamma) = -0.550     C(sigma, gamma) = 0.530
C(sigma, mu) = -0.389       C(beta, mu) = 0.213
```

1.8.3 指标计算补充

使用优化参数调用定义后的 ode_solver 计算 MAE（平均绝对误差）和 RMSE （均方根误差）

```
Fitted MAE                                    Fitted RMSE
Infected: 20794.332654685975                  Infected: 27499.729191344883
Recovered: 14536.028287085463                 Recovered: 21159.039391924664
Dead: 1348.6585045293205                      Dead: 2000.0961938063565
```

1.8.4 结论总结

1. 阿肯色州的预测曲线的拐点远早于实际数据，这说明防疫措施并不到位，图像很晚达到拐点说明该地区的确诊人数仍然将快速增长，需要政府发布更有效的政令，民众更遵守疫情防控规则，才能使疫情拐点更早到来。
2. 阿肯色州的治愈人数曲线从 100 天到 150 天明显高于预测曲线，说明医疗物资的供给提高和疫情防治的方法成熟使得治愈人数上升；但 150 天后曲线又相互吻合，推测是病毒变异的出现使得患者更难被治愈。
3. 阿肯色州的死亡人数始终高于预测曲线，说明医疗防治手段还需要进一步整体提升。

二、 人类移动和聚集行为对新冠病毒传播的影响

2.1 阿肯色州人群移动与聚集行为调查分析

新冠病毒流行改变着人们的流动方式，需要流行病学模型来捕捉这些流动性变化对严重急性呼吸系统综合症冠状病毒传播的影响。为了应对新冠肺炎危机，各国都颁布了国内禁流命令，以减少个人之间的联系，并减缓 SARS-新冠肺炎-29 的传播。

将流行病学模型与移动模型相结合，不仅能够准确地适应观察到的病例计数，而且能够进行详细的分析，从而对 covid-19 提供更有效和更公平的政策反应信息。

美国阿肯色州人群移动性在 2020 年 3 月急剧下降。例如，在 3 月的第一周和 2020 年 4 月的第一周之间，阿肯色州的整体 POI 访问量下 54.7%。

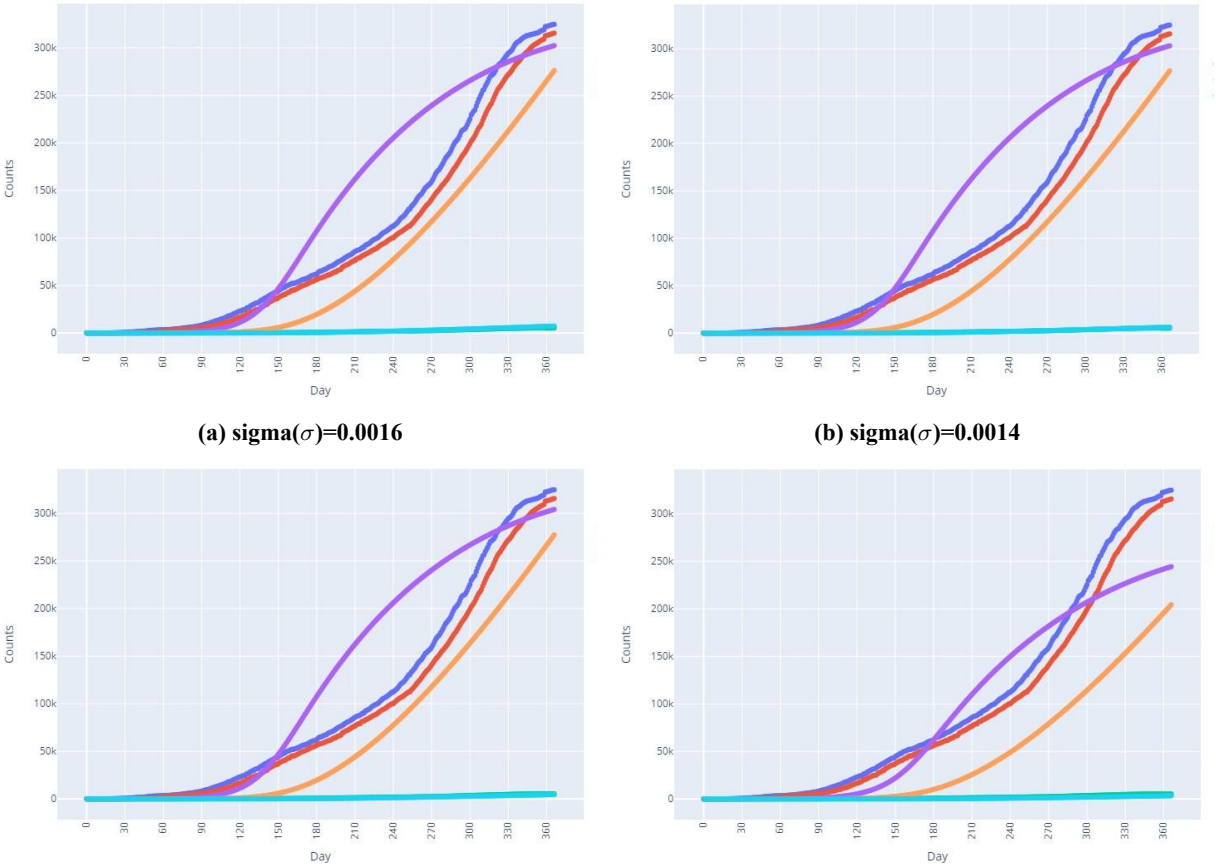
查阅资料得知，低收入人群具有更高的传播率，但是人均访问量的差异并不能完全解释感染差异。以下分析提供用于分析流动性减少或增多、重新开放移动计划和聚集人口差异的实验程序的详细信息。

2.2 分析现有图像推断病毒传播

结论: 发病率随聚集人数增多/移动人群计数增加而增大；传染人数随之上升。采用控制变量进行验证。

初始图像在原有的图像上进行调整，考虑到新冠病毒变异快、爆发猛，即使在封闭区域也会自发增强感染性，因此在原有预测数据的基础上进一步感染人数，以贴合疫情本身特性。

图 5　原始参考图



(a) sigma(σ)=0.0016　　**(b) sigma(σ)=0.0014**

(c) sigma(σ)=0.0010　　**(d) sigma(σ)=0.0008**

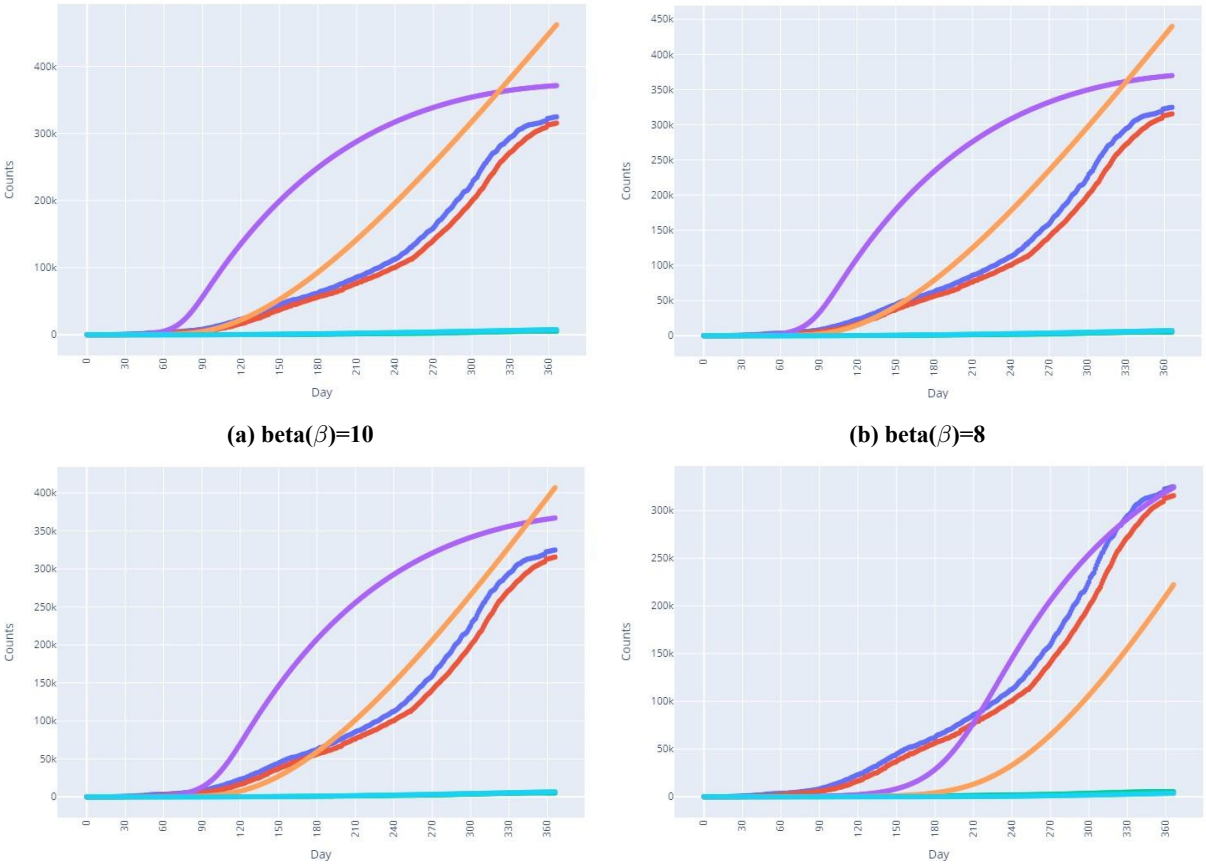
图 **6**　参数对比图

调整数据，进行控制变量分析。保持 beta = 4 不变，首先改变 sigma，上调或下调。

已知原数据中我们控制 sigma 初值为 0.0012，那么在增加到 0.0014 时代表发病率增高，而这是聚集人数增多/移动基数变大导致的。分析图像趋势可知，此时观测到的感染人数开始多于预测的感染人数。随着 sigma 值继续提高到 0.0016，图像继续发　微小改变，感染人数观测值比预测值多了更多。

当 sigma 值减小到 0.0010 时，预测感染与观测感染都变少了。说明随着聚集人数和移动人群减少，感染数变小。随着 sigma 值降到更低的水平，预测感染与观测感染变得更少，并且产生图形上更　的变化。

综上，当仅仅考虑到发病率的时候，我们可以认为聚集程度和移动范围与发病率呈正相关。与此相对应的是，随着聚集和移动的人数提高，发病率变大。



(a) beta(β)=10　　**(b) beta(β)=8**

(c) beta(β)=6　　**(d) beta(β)=2**

图 **7**　参数对比图

调整数据，进行控制变量分析。保持 sigma = 0.0012 不变，改变 beta，上调或下调。

已知原数据中我们控制 beta 初值为 4，那么在增加到 6/8/10 时代表感染人数增加，这是聚集人数增多/移动基数变大导致的。

分析图像趋势可知，beta 为 6 时观测到的感染人数略少于预测的感染人数；而随着 beta 值继续提高到 8，图像继续发生改变，感染人数观测值持续变多，预测值持续变多。beta 提高到 10 时，毫无悬念地，感染人数的观测值和预测值都多于原值情况下观察值和预测值。当 beta 值减小到 2 时，预测感染与观测感染都变少了。说明随着聚集人数和移动人群减少，感染数变小。

综上，当仅仅考虑到感染人数的时候，我们可以认为感染人数受到聚集程度和移动范围影响；也可以认为聚集程度和移动范围与感染人数呈正相关。与此相对应的是，随着聚集和移动的人数提　，感染人数增加。

2.3 结论

我们认为严格的疫情防控和控制移动范围、控制聚集等活动对 COVID-19 的传染起到了良性作用，为了减小感染病毒可能性，减少大型集会、减少跨区域移动对防控病毒的传播起到有效作用。

三、 疫苗存在的情况下对地区疫情进行分析

3.1 分析相关参数改变

以色列是人均接种率最高的国家之一。根据以色列卫生部自然研究报告显示，间隔21天的两剂给药方案可以提供95%的保护率；在第一剂后14-20天和第二剂后7天的疗效分别为46%和92%。在接种人群方面，青少年、服役年轻人和重症患者拥有疫苗优先权。数据显示，与晚期接种疫苗的城市相比，早期接种疫苗的城市的COVID-19病例数和60岁及以上人群的住院人数下降幅度更大、更早；与峰值相比，早期接种疫苗的城市病例减少了88%，重症住院减少了79%，而在接种疫苗较晚的城市，病例减少了78%，重症住院减少了66%。因此疫苗可以显著降低发病率、死亡率，大幅提升治愈率；另一方面，封锁城市、禁止群众集会可以显著抑制人口流动，有利于组织疫苗接种，可以降低病毒的传染率和人群的发病率。

但是疫苗的接种需要群众的意愿支持，这也导致民众对于疫苗的态度会影响疫苗在整个地区发挥的作用，而科学研究显示一种新型疫苗至少需要55%的人口接受才能提供对社会群体较全面的保护。如今社交媒体上流传着关于新冠肺炎病情的虚假信息，例如5G移动网络会随着疫苗进入人体、疫苗会导致人死亡等，会在一定程度上影响地区的接种　作。在英美地区，分别有54.1%和42.5%的受访者表示他们“绝对”接受疫苗以保护自己，有6%和15%表示他们“绝对不会”接受疫苗。另一方面，地区的经济水平和教育水平也会对疫苗接种率造成影响，影响的因素包括疫苗供给和群众对疫苗的认知水平等。

疫苗名称	辉瑞新冠肺炎疫苗	Moderna新冠肺炎疫苗	强生新冠肺炎疫苗
疫苗类型	mRNA疫苗	mRNA疫苗	腺病毒载体疫苗
三期疫苗实验率	95% （16-55岁为95.6%） （56岁以上为93.7%）	94.1% （18-64岁为95.6%） （65岁以上为86.4%）	86% （预防中重度有效率平均为66.1%）
存放温度	−80°C至−60°C （2°C至8°C可存放5日）	−25°C至−15°C （−80°C至−60°C可存放30日） （8°C至25°C可存放12小时）	−20°C以下可存放两年 （2°C至8°C可存放3个月）
使用时间	稀释后常温6小时内用完	开封后6小时内用完	开封后6小时内用完
有效期	6个月以上	6个月以上	6个月以上
说明	无	无	无

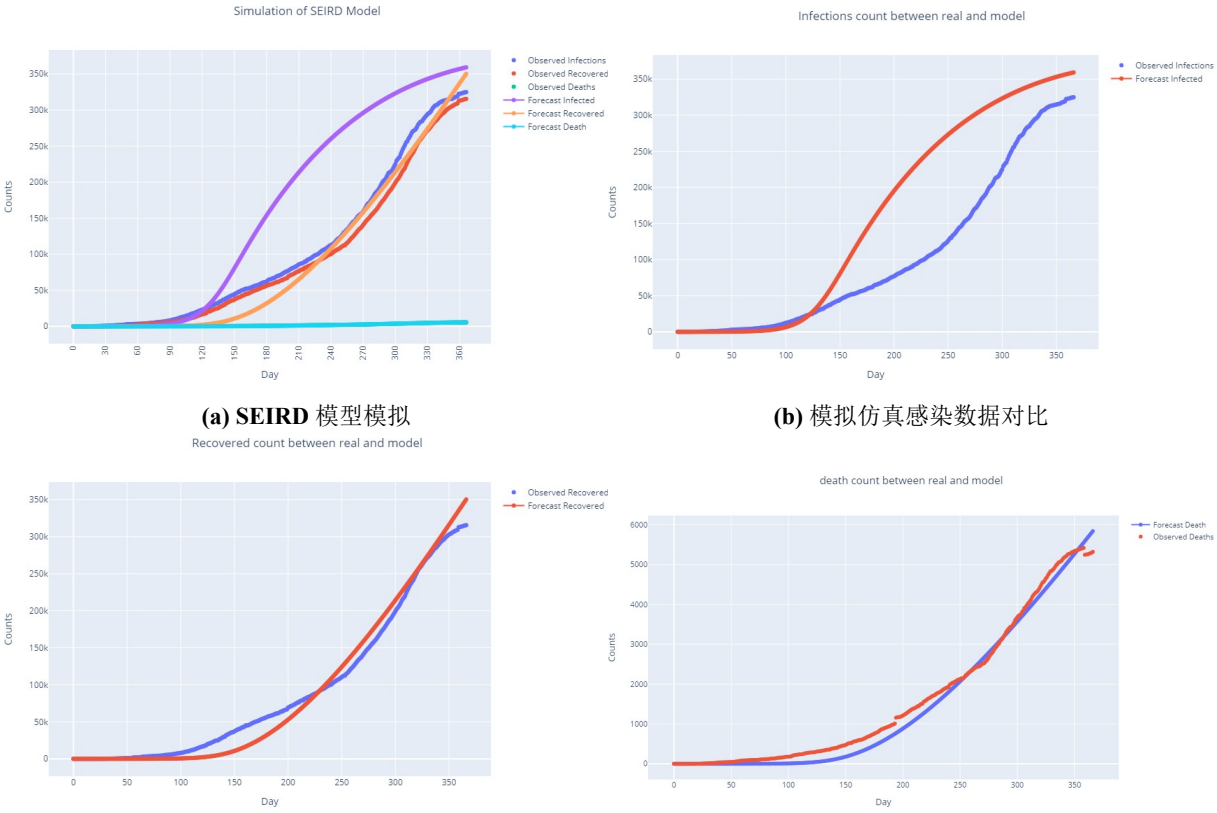
表 **2**　国外主流接种疫苗数据

各州从有效率角度考虑，会尽可能多选择辉瑞和Moderna疫苗，而这两种疫苗的有效率非常接近，因此在调整参数上忽略细微区别。

3.2 分析疫情发展情况

根据 Our World In Data 可查到的阿肯色州疫苗接种率，在 2020 年 1 月份该州正式启动了疫苗接种工作，在 2 月初疫苗接种率达到 10%，在 3 月初达到 20%，4 月初达到 30%。而我们的疫情数据起始日期是 2020 年 3 月 6 日至 2021 年 3 月 6 日，在该范围内对接种和聚集情况进行分析，适当调整参数，观察原数据和预测数据。

在原始数据中，政府在一定程度上限制人员聚集和移动，且可以实现疫苗在人群中的一定程度接种。对比图象如下：



(a) SEIRD 模型模拟

(b) 模拟仿真感染数据对比

(c) 模拟仿真康复数据对比

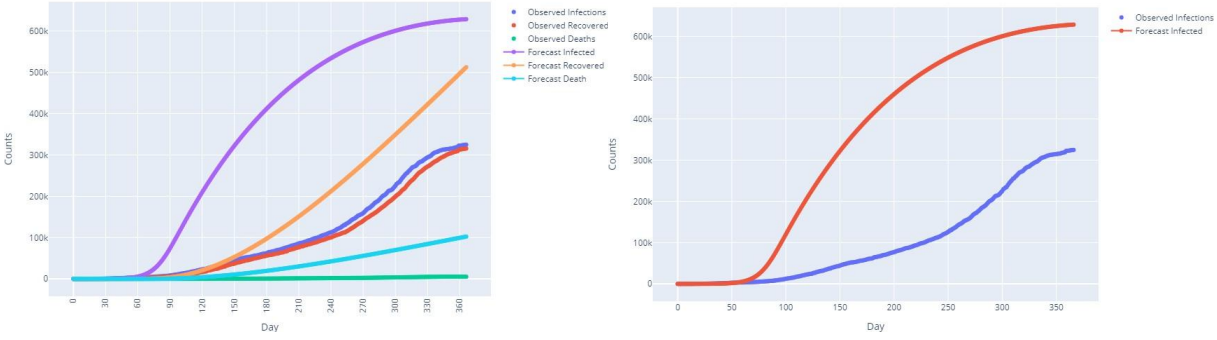
(d) 模拟仿真死亡数据对比

图 **8**　模型实际数据对比图

疫情发展趋势	疫情得到有效控制	疫情未得到控制	疫情的实际发展情况
传染人数 (beta)	1	6	4
发病率 (sigma)	0.0008	0.002	0.0012
康复率 (gamma)	0.001	0.004	0.0034
死亡率 (mu)	0.0005	0.0008	0.0006
群众活动	民众不聚集 自发接种疫苗	民众大范围聚集 极少数人接种疫苗	民众保持日常活动 进一半人接种疫苗
政府管控	严格控制人员聚集活动 疫苗储备充足	几乎不限制人员聚集活动 无疫苗储备	发布公告鼓励居家 少量疫苗储备
说明	除上述参数具有差异，其余模型参数保持不变		

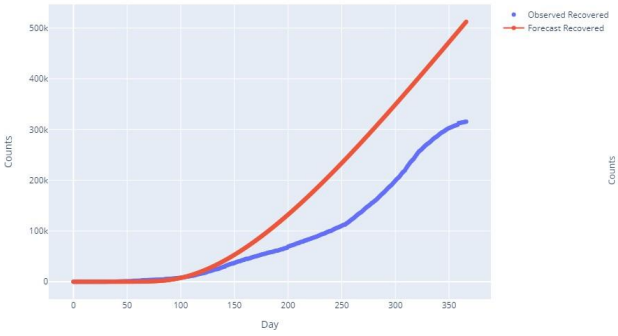
表 **3**　疫情模拟情况参数对比表

假设在聚集且不接种疫苗的情况下，政府　乎不限制人员聚集和移动，且只有极小一部分人接种疫苗。对比图像如下：

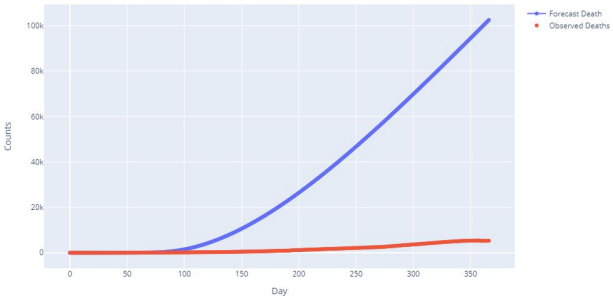


(a) SEIRD 模型模拟

(b) 模拟仿真感染数据对比



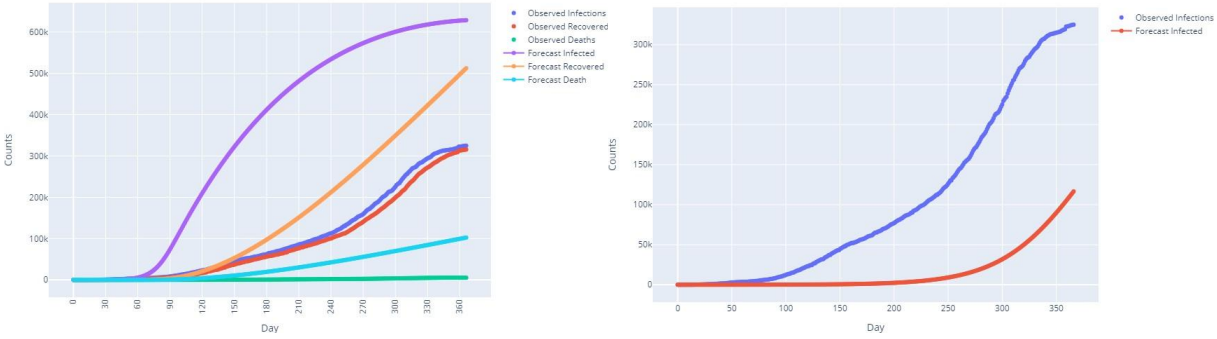
(c) 模拟仿真康复数据对比



(d) 模拟仿真死亡数据对比

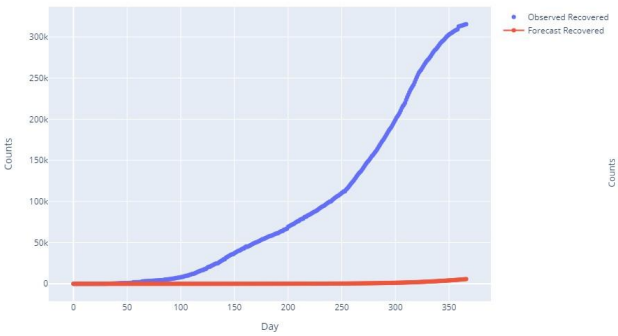
图 **9**　疫情恶化模型实际数据对比图

假设在不聚集且接种疫苗的情况下，政府严格控制人员聚集和移动，群众自发接种疫苗，疫苗储备充足。对比图像如下：

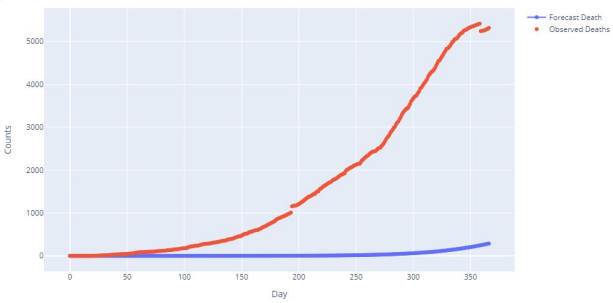


(a) SEIRD 模型模拟

(b) 模拟仿真感染数据对比



(c) 模拟仿真康复数据对比



(d) 模拟仿真死亡数据对比

图 **10**　疫情控制模型实际数据对比图

3.3 结论

观察在不同政策下疫情的发展趋势图，结合已有的在无疫苗情况下的图像，我们很容易得到以下结论：

1. 疫苗对于防控疫情十分重要，体现在降低传染率和升高治愈率上。因此消灭疫情需要有效的疫苗和较高的接种率。
2. 政府需要将疫苗接种和防止人员流动结合起来，既可以降低感染人数，使疫情无法大规模爆发，也可以保护人民的生命财产安全，增强社会稳定。
3. 民众舆论会对政府政策造成一定影响，地区人群对于防疫措施的配合程度会进一步影响疫情发展，因此真实的疫情数据伴随着地区民众意愿有一定波动。

四、 结论

阿肯色州的疫情发展对我国的防疫措施有着一定的参考价值。疫情数据和预测结果均表明，政府需要有强有力的措施来推行政策，且需要民众的积极配合。疫苗研发对于防止疫情爆发有着决定性作用，同时可以最大程度保障人民生命安全。疫情期间需要对群众进行一定的医疗知识宣传，尽可能增加疫苗的接种率，减少“疫苗谬论”，保证绝大多民众有疫苗接种意愿。美国政府在疫情初期应对疫情不利，导致了后来多轮疫情的爆发；现阶段我国的疫情显著向好，应该警惕各种疫情谣言，推进防疫政策，保证人群接种率，防止疫情的再度爆发。

参考文献

[1] Przybylowski Adam,Stelmak Sandra,Suchanek Michal. Mobility Behaviour in View of the Impact of the COVID-19 Pandemic—Public Transport Users in Gdansk Case Study[J]. Sustainability,2021,13(1).

[2] Chang Serina,Pierson Emma,Koh Pang Wei,Gerardin Jaline,Redbird Beth,Grusky David,Leskovec Jure. Mobility network models of COVID-19 explain inequities and inform reopening [J]. Nature,2020.

[3] 李瑞松, 刘洪久. 基于传染病模型应用于中美两国应对 COVID-19 的对比研究[J]. 疾病预防控制通报,2021,36(02):1-5+21.

[4] 邹宇庭, 郑晓练, 缪旭晖, 谭忠. SARS 传播的数学原理及预测与控制[J]. 程数学学报,2003,20(07):29-34.

附录 A　完整可执行程序

```
# python 库导入
import os import sys
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy.integrate import odeint
import plotly.graph_objects as go
import plotly.io as pio
from lmfit import minimize, Parameters, Parameter, report_fit
pio.renderers.default = "notebook"
%matplotlib inline plt.style.use('ggplot')

# 读取数据，绘图的功能模块
from IPython.display import HTML
from ipywidgets.widgets import interact, IntSlider, FloatSlider, Layout, ToggleButton, ToggleButtons
style = {'description_width': '100px'}
slider_layout = Layout(width='99%')

# 图形处理有关
def ode_model(z, t, beta, sigma, gamma, mu):
    """ Reference https://www.idmod.org/docs/hiv/model-seir.html """
    S, E, I, R, D = z
    N = S + E + I + R + D
    dSdt = -beta * S * I / N
    dEdt = beta * S * I / N - sigma * E
    dIdt = sigma * E - gamma * I - mu * I
    dRdt = gamma * I
    dDdt = mu * I
    return [dSdt, dEdt, dIdt, dRdt, dDdt]

# 微分方程传播模型求解
def ode_solver(t, initial_conditions, params):
    initE, initI, initR, initN, initD = initial_conditions
    beta, sigma, gamma, mu = params['beta'].value, params['sigma'].value, params['gamma'].value,
↪   params['mu'].value
    initS = initN - (initE + initI + initR + initD)
    resl = odeint(ode_model, [initS, initE, initI, initR, initD], t, args= (beta, sigma, gamma, mu))
    return resl

# 解常微分方程的函数
df1= pd.read_csv('covid_tracking.csv', sep=',') df_AR = df1.query('state=="AR"') # 阿肯色州
# df_AR.describe() df_temp = df_AR.iloc[::-1]
df_temp.reset_index(drop=True,inplace=True)
df_temp2= df_temp.loc[:, ['date', 'positive','recovered','death']] # df_temp2.tail()
values = {'death': 0,'recovered': 0} df_temp2.fillna(value=values,inplace=True)

# 全部初始化参数
initN = 2830000
initE = 10
initI = 20
initR = 0
initD = 0
sigma = 0.0010
gamma = 0.006
mu = 0.0012
R0 = 2
beta = 4
days = 340
params = Parameters()
params.add('beta', value=beta, min=0, max=10)
params.add('sigma', value=sigma, min=0, max=10)
params.add('gamma', value=gamma, min=0, max=10)
params.add('mu', value=mu, min=0, max=10)

```

```
# 演示效果动态调参
def main(initE, initI, initR, initD, initN, beta, sigma, gamma, mu, days, param_fitting):
    initial_conditions = [initE, initI, initR, initN, initD] params['beta'].value,
    params['sigma'].value,params['gamma'].value, params['mu'].value = [beta, sigma, gamma, mu]
    tspan = np.arange(0, days, 1)
    sol = ode_solver(tspan, initial_conditions, params)
    S, E, I, R, D = sol[:, 0], sol[:, 1], sol[:, 2], sol[:, 3], sol[:, 4]

    # Create traces
    fig = go.Figure()
    fig1 = go.Figure()
    if not param_fitting:
        fig.add_trace(go.Scatter(x=tspan, y=S, mode='lines+markers', name= 'Susceptible'))
        fig.add_trace(go.Scatter(x=tspan, y=E, mode='lines+markers', name= 'Exposed'))
        fig.add_trace(go.Scatter(x=tspan, y=I, mode='lines+markers', name= 'Infected'))
        fig.add_trace(go.Scatter(x=tspan, y=R, mode='lines+markers',name=' Recovered'))
        fig.add_trace(go.Scatter(x=tspan, y=D, mode='lines+markers',name=' Death'))
        fig1.add_trace(go.Scatter(x=tspan, y=I, mode='lines+markers', name='Infected'))
        fig1.add_trace(go.Scatter(x=tspan, y=R, mode='lines+markers',name= 'Recovered'))
        fig1.add_trace(go.Scatter(x=tspan, y=D, mode='lines+markers',name= 'Death'))
    if param_fitting:
        fig.add_trace(go.Scatter(x=tspan, y=df_temp2.positive, mode='lines+markers',name='Infections Observed',
    line = dict(dash= 'dash')))
        fig.add_trace(go.Scatter(x=tspan, y=df_temp2.recovered, mode='lines+markers',name='Recovered Observed',
    line = dict(dash=' dash')))
        fig.add_trace(go.Scatter(x=tspan, y=df_temp2.death, mode='lines+markers',name='Deaths Observed', line =
    dict(dash='das h')))

    if days <= 30:
        step = 1
    elif days <= 90:
        step = 7
    else:
        step = 30

    # Edit the layout
    fig1.update_layout(title='SEIRD test',xaxis_title='Day',yaxis_title='Counts', title_x=0.5,width=1000,
    height=600)
    fig.update_layout(title='Simulation of SEIRD Model',xaxis_title='Day', yaxis_title='Counts',
    title_x=0.5,width=900, height=600)
    fig.update_xaxes(tickangle=-90, tickformat = None, tickmode='array', tickvals=np.arange(0, days + 1, step))
    if not os.path.exists("images"): os.mkdir("images")
        fig.write_image("images/seird_simulation.png")
        fig.show()

# 对应动态调参数图
interact(main,initE=IntSlider(min=0, max=100000, step=1, value=initE, description='initE', style=style,
    layout=slider_layout),
    initI=IntSlider(min=0, max=100000, step=10, value=initI, description='initI', style=style,
    layout=slider_layout),
    initR=IntSlider(min=0, max=100000, step=10, value=initR, description='initR', style=style,
    layout=slider_layout),
    initD=IntSlider(min=0, max=100000, step=10, value=initD, description='initD', style=style,
    layout=slider_layout),
    initN=IntSlider(min=0, max=1380000000, step=10000, value=initN, description='initN', style=style,
    layout=slider_layout),
    beta=FloatSlider(min=0, max=4, step=0.01, value=beta, description='Infection rate', style=style,
    layout=slider_layout),
    sigma=FloatSlider(min=0, max=4, step=0.01, value=sigma, description='Incubation rate', style=style,
    layout=slider_layout),
    gamma=FloatSlider(min=0, max=4, step=0.01, value=gamma, description='Recovery rate', style=style,
    layout=slider_layout),
    mu=FloatSlider(min=0, max=1, step=0.001, value=mu, description='Mortality rate', style=style,
    layout=slider_layout),
    days=IntSlider(min=0, max=600, step=7, value=days, description='Days', style=style, layout=slider_layout),
    param_fitting=ToggleButton(value=False, description='Fitting Mode', disabled=False, button_style='',
    tooltip='Click to show fewer plots', icon='check-circle'))
```

```python
# 误差分析
def error(params, initial_conditions, tspan, data):
    sol = ode_solver(tspan, initial_conditions, params)
    return (sol[:, 2:5] - data).ravel()

# 历史数据拟合
initial_conditions = [initE, initI, initR, initN, initD]
beta = 4
sigma = 0.0012
gamma = 0.006
mu = 0.0001
params['beta'].value = beta
params['sigma'].value = sigma
params['gamma'].value = gamma
params['mu'].value = mu
days = 340
tspan = np.arange(0, days, 1)
data = df_temp2.loc[0:(days-1), ['positive', 'recovered', 'death']].values

print(initial_conditions)
print(params)

# 输出拟合结果
# fit model and find predicted values
result = minimize(error, params, args=(initial_conditions, tspan, data), method='leastsq')
report_fit(result)

# 根据历史数据生参数
final = data + result.residual.reshape(data.shape)
temp_days = 120
fig = go.Figure()
fig.add_trace(go.Scatter(x=tspan, y=data[:, 0], mode='markers', name='Obse rved Infections', line =
↪  dict(dash='dot')))
fig.add_trace(go.Scatter(x=tspan, y=data[:, 1], mode='markers', name='Obse rved Recovered', line =
↪  dict(dash='dot')))
fig.add_trace(go.Scatter(x=tspan, y=data[:, 2], mode='markers', name='Obse rved Deaths', line =
↪  dict(dash='dot')))
fig.add_trace(go.Scatter(x=tspan, y=final[:, 0], mode='lines+markers', name='Fitted Infections'))
fig.add_trace(go.Scatter(x=tspan, y=final[:, 1], mode='lines+markers', name='Fitted Recovered'))
fig.add_trace(go.Scatter(x=tspan, y=final[:, 2], mode='lines+markers', name='Fitted Deaths'))
fig.update_layout(title='SEIRD: Observed vs Fitted',xaxis_title='Day', yaxis_title='Counts',
↪  title_x=0.5,width=1000, height=600)
```