2021 年兰州大学数学建模竞赛论文

新冠的预测与控制

陈雨节　　大气科学学院
刘 璐　　数学与统计学院
陈永青　数学与统计学院

新冠的预测与控制

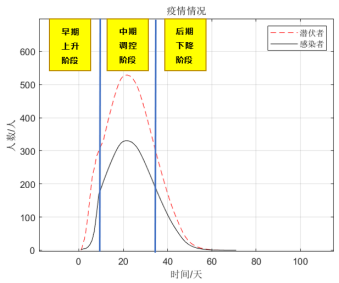
摘要

前两年爆发的新冠疫情给世界人民带来了极大的灾难，控制新冠疫情是所有人民共同的目标。本文运用了聚类分析划分了城市疫情的严重程度，运用主成分分析确定了影响疫情的要素，并且建立了 $SEIR$ 和 $ARMA$ 两种模型进行疫情的预测。最后给出了疫情防控的建议。

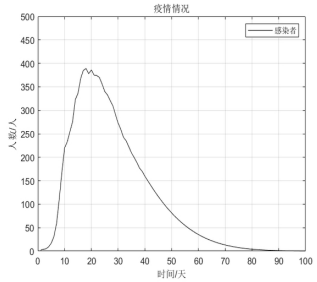
对于问题一，首先，我们计算出每个城市的确诊率 (确诊率 $=\frac{A}{B}$, 其中 A=29 天确诊总人数，B= 城市人口数) 来判断城市疫情的严重程度，接着用 $min-maxnormalization($ 离差标准化) 给每个确诊率赋分，最后用 $k-means$ 聚类方法分为五类，第一类表示比较轻微，第二类表示轻微，第三类表示比较严重，第四类表示严重，第五类表示非常严重，然后我们将第一类以及第二类划归为轻微，第三，第四，第五类划归为严重。我们用 0 代表轻微，用 1 代表严重, 确定了每个城市的疫情严重程度。

对于问题二，我们首先对所给数据进行数据预处理，包括缺失值的的填补，异常值的筛选，数据的统计描述以及数据的正态性检验等，并绘制了六个指标的概率密度函数图。接下来我们使用主成分分析法确定了影响疫情发展情况的影响因素为 $prop.elder, Population, density, pergdp, Meantem2020$ 这五个指标。

对于问题三，我们除了题目中所给的数据外，我们从网上查找补充了温州市 3 月 31 号之前的新增确诊病例，新增出院人数，新增死亡人数。然后我们分别从病毒传播角度和统计学角度建立了 $SEIR$ 和 $ARMA$ 两种预测模型。两个预测模型预测结果基本一致，说明预测模型较为合理，准确度高。两个模型的预测结果如下：



(a) $SEIR$ 预测结果



(b) $ARMA$ 模型的疫情情况预测

关键字： $k-means$ 聚类分析　主成分分析　$SEIR$ 模型　$ARMA$ 模型

一、 问题重述

1.1 问题背景

前两年的新冠病毒在世界范围内造成了大面积的疫情，引起了社会的极大恐慌，导致对疫情的预测与控制成为十分重要的研究课题。

1.2 问题的提出

在给出了世界一些城市在 2020 年 2 月份的疫情情况以及其他的一些数据后，本文主要研究以下三个问题：

1. 对各城市的疫情情况进行分类，标注出哪些城市疫情严重，哪些城市疫情轻微。
2. 探讨疫情发展情况与哪些城市数据有关系。
3. 根据数据建立 $Wenzhou$(温州) 的疫情模型，并给出疫情的预测与控制建议。

二、 模型假设

- 假设题目已给数据真实可靠
- 假设新增确诊病例均为本土病例

三、 符号说明

表 **1** 符号说明

符号	含义
A	29 天确诊总人数
B	城市人口数
0	轻微
1	严重
$\widehat{F_i}$	指标 i 的均值

表 **2**　符号说明

符号	含义
λ_i	特征值
ν_i	特征向量
r_{ij}	指标 i 和指标 j 之间的相关系数
Y_i	第 i 个主成分
V_1	指标 $prop.elder$
V_2	指标 $Population$
V_3	指标 $density$
V_4	指标 $Latitude$
V_5	指标 $perhdp$
V_6	指标 $Meanrem2020$
S	易感人群
I	患病者
D	死亡患者
E	潜伏者
R	康复者
h	感染力
S_i	指标 i 的标准差
α	接触率
β	感染率
γ	治愈率

表 **3**　符号说明

符号	含义
k	病死率
i	转病率
r_{ij}	指标 i 和指标 j 之间的相关系数
α	潜伏者转阳率
β	传染概率
r	指标平均每个病人每天接触的人数
r_2	潜伏者平均每天接触的人数
β_2	传染概率 (接触潜伏者)
β_3	潜伏者的转阴率
W_t	第 t 天的康复率
x_t	第 t 天的新增出院人数
y_t	第 t 天的已有患者数
ε_t	模拟康复率的真实值与 $ARMA$ 模型预测值之差
p	AR 部分的阶数
q	MA 模型部分的参数
$\widetilde{F}_{ij}$	F_{ij} 进行标准化后的值
F_{ij}	第 j 个样本的第 i 个指标值
S_i	指标 i 的标准差

四、 问题分析

问题总体分析流程图：

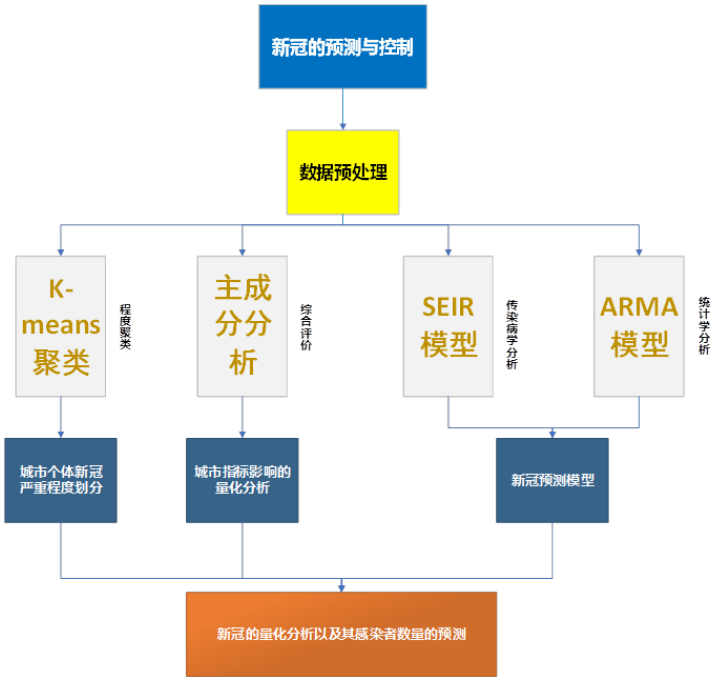


图 **2**　问题总体分析流程图

4.1 问题一分析

对于问题一，首先，我们计算出每个城市的确诊率 (确诊率 $= \frac{A}{B}$, 其中 A=29 天确诊总人数，B= 城市人口数) 来判断城市疫情的严重程度，接着用 $min-maxnormalization($ 离差标准化) 给每个确诊率赋分，最后用 $k-means$ 聚类方法分为五类，第一类表示比较轻微，第二类表示轻微，第三类表示比较严重，第四类表示严重，第五类表示非常严重，然后我们将第一类以及第二类划归为轻微，第三，第四，第五类划归为严重。我们用 0 代表轻微，用 1 代表严重, 确定了每个城市的疫情严重程度。

4.2 问题二分析

对于问题二，我们首先对所给数据进行数据预处理，包括缺失值的的填补，异常值的筛选，数据的统计描述以及数据的正态性检验等，并绘制了六个指标的概率密度函数图。接下来我们使用主成分分析法确定了影响疫情发展情况的影响因素为 $prop.elder, Population, density, pergdp, Meantem2020$ 这五个指标。

4.3 问题三分析

对于问题三，我们除了题目中所给的数据外，我们从网上查找补充了温州市 3 月 31 号之前的新增确诊病例，新增出院人数，新增死亡人数。然后我们分别从病毒传播角度和统计学角度建立了 $SEIR$ 和 $ARMA$ 两种预测模型。两个预测模型预测结果基本一致，说明预测模型较为合理，准确度高。

五、 模型的建立与求解

5.1 数据预处理

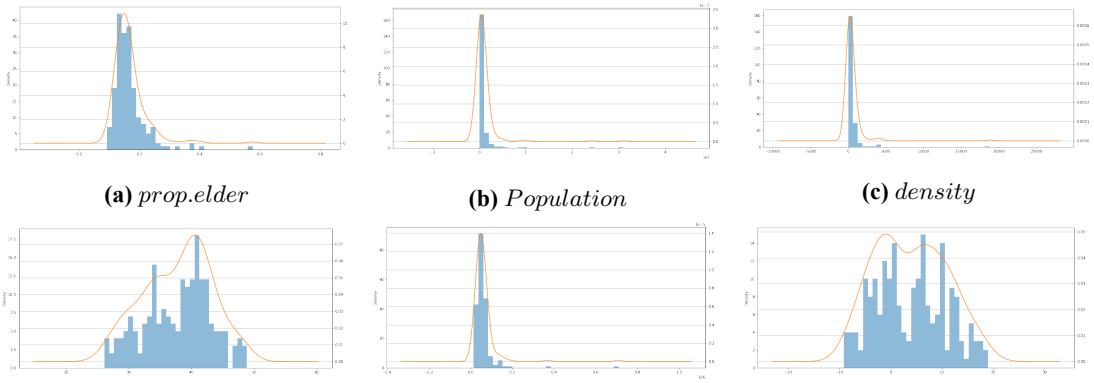
对于题目所给数据，我们进行了初步的数据预处理，包括缺失值的填补，异常值的筛选，数据的统计描述以及数据的正态检验等。

六个指标数据的统计描述表如下：

表 **4**　数据描述统计表

	prop.elder	**Population**	**density**	**Latitude**	**pergdp**	**Meantem2020**
count	200	200	200	200	200	200
mean	0.1665309	985412.36	500.7107	37.813981	9801.147	3.8817245
std	0.0545816	2949009.214	1443.671	5.334009	7282.922	6.8224410
min	0.0924004	8419	0.96613	26.118713	4412.858	-9.1666666
25%	0.1355609	154983.75	62.35112	33.95431075	38967.978	-1.3194444
50%	0.1564775	392077.5	186.2441	38.8266315	50303.981	4.1944444
75%	0.1788839	829162.75	523.0537	41.652126	66722.639	9.4861111
max	0.5758733	31017900	18720.9	48.842653	709722.82	18.944444

经检验，该数据集无异常值，但是有缺失值，通过网上查找资料 [1]，我们补上了 $Virginia-Fairfax$ 的 $pergdp$(人均 gdp)：709722.82。并且数据不符合正态分布 (ktest 检验 p 值均小于 0.05)，其概率密度函数图如下：



(a) *prop.elder* **(b)** *Population* **(c)** *density*

(d) *Latitude* **(e)** *pergdp* **(f)** *Meantem*2020

5.2 城市疫情严重程度的判定

首先，我们计算出每个城市的确诊率来判断城市疫情的严重程度，接着用 $min-maxnormalization$(离差标准化) 给每个确诊率赋分，最后用 $k-means$ 聚类方法分为五类，第一类表示比较轻微，第二类表示轻微，第三类表示比较严重，第四类表示严重，第五类表示非常严重，然后我们将第一类以及第二类划归为轻微，第三，第四，第五类划归为严重。我们用 0 代表轻微，用 1 代表严重。其中 30 个城市的判断结果如下:

5.3 基于主成分分析确定疫情发展情况的影响因素

为了探究疫情发展情况的影响因素，本文选择采用主成分分析对六个指标进行分析，具体步骤如下：

第一步：为了消除量纲的影响，对六个指标进行标准化处理。

$$\widetilde{F}_{ij}=\frac{F_{ij}-\widehat{F}_i}{S_i}$$

其中 $\widehat{F}_i$ 和 S_i 分别为指标 i 的均值和标准差。

第二步：计算标准化数据的相关系数矩阵。

记第 i 个指标和第 j 个指标之间的相关系数为 r_{ij},

$$r_{ij}=\frac{\sum_{k=1}^{6}\widetilde{F}_{ik}\widetilde{F}_{jk}}{6-1}$$

则相关矩阵为 $R=(r_{ij})_{6\times 6}$, 其中 $r_{ii}=1, r_{ij}=r_{ji}$。

第三步：计算相关系数矩阵的特征值和特征向量。

将计算得到的矩阵的特征值记为 $\lambda_i(i=1,2,...,6)$, 对应的特征向量记为 $\nu_i(i=1,2,...,6)$, 其中 $\nu_i=(u_{1i},u_{2i},u_{3i},u_{4i},u_{5i},u_{6i})^T$, 这 6 个特征向量组成 6 个新的指标：

表 **5** 判断结果

城市	确诊率	赋分	严重程度
$Wenzhou$	5.42×10^{-5}	0.195250723534233	0
$Shanghai$	1.37×10^{-5}	0	0
$Chongqing$	1.79×10^{-5}	0.020049006029124	0
$Arizona-Maricopa$	1.23×10^{-4}	0.530039876966016	0
$Arizona-Pima$	1.47×10^{-4}	0.644362742266113	0
$Massachusetts-Norfolk$	3.45×10^{-3}	16.5974986296562	1
$Arkansas-Jefferson$	3.81×10^{-4}	1.77599281806472	0
$California-Alameda$	1.61×10^{-4}	0.715527994196268	0
$California-ContraCosta$	1.52×10^{-4}	0.668006517264771	0
$California-Fresno$	4.32×10^{-5}	0.142394851369714	0
$California-Humboldt$	1.32×10^{-4}	0.570732551756564	0
$California-Shasta$	3.40×10^{-3}	16.3455618553139	1
$California-Madera$	2.11×10^{-4}	0.953846592303555	0
$California-Orange$	9.51×10^{-5}	0.392846967565178	0
$California-Placer$	1.35×10^{-4}	0.586610315895895	0
$California-Riverside$	1.35×10^{-4}	0.584333382452345	0
$California-Sacramento$	9.51×10^{-5}	0.392549776082824	0
$California-Sonoma$	4.58×10^{-3}	22.0423605521108	1
$California-SanDiego$	1.06×10^{-4}	0.447346382507415	0
$California-SanFrancisco$	2.28×10^{-4}	1.03172420412192	0

$$\begin{cases} Y_1 = u_{11}\widetilde{F}_1 + u_{21}\widetilde{F}_2 + ... + u_{61}\widetilde{F}_6 \\ Y_2 = u_{12}\widetilde{F}_1 + u_{22}\widetilde{F}_2 + ... + u_{62}\widetilde{F}_6 \\ ... \\ Y_6 = u_{16}\widetilde{F}_1 + u_{26}\widetilde{F}_2 + ... + u_{66}\widetilde{F}_6 \end{cases}$$

其中，Y_i 为第 i 个主成分，$i = 1, 2, ..., 6$。

第四步：确定主成分并确定疫情发展情况的影响因素。根据以上步骤，本文利用 $Python$ 求得各指标的相关系数表以及各指标的特征值和对应的特征向量矩阵：

表 **6** 六个指标的相关系数表

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1	-0.10	-0.11	-0.13	-0.17	0.19
x_2	-0.10	1	0.15	-0.19	0.12	0.13
x_3	-0.11	0.15	1	0.03	0.47	-0.06
x_4	-0.13	-0.19	0.03	1	0.08	-0.81
x_5	-0.17	0.12	0.47	0.08	1	-0.08
x_6	0.19	0.13	-0.06	-0.81	-0.08	1

表 **7** 六个指标的特征值，贡献率和累计贡献率

	x_1	x_2	x_3	x_4	x_5	x_6
特征值	1.97139	1.59941	0.96147	0.78940	0.52410	0.18438
贡献率	0.32692	0.26523	0.15944	0.13091	0.08691	0.03508
累计贡献率	0.32692	0.59215	0.75159	0.8825	0.96941	0.99999

为了书写的方便,在下文中,我们也用 $V_1, V_2, V_3, V_4, V_5, V_6$ 分别来代表指标 $prop.elder$, $Population, density, Latitude, pergdp, Meantem2020$。

从表 4 可以看出，前五个主成分的累计贡献率已高达 96.941%, 因此我们考虑只取前五个主成分，由特征向量矩阵可以写出前五个主成分：

第一主成分：

$$Y_1 = -0.26881x_1 - 0.13212x_2 + 0.19352x_3 + 0.63418x_4 + 0.24070x_5 - 0.64243x_6$$

表示在 V_3, V_4, V_5 上有正载荷，在 V_1, V_2, V_6 上有负载荷，且在 V_6 上有较高程度的载荷。

第二主成分：

$$Y_2 = -0.25386x_1 + 0.43183x_2 + 0.58186x_3 - 0.22571x_4 + 0.57079x_5 + 0.18373x_6$$

表示在 V_2, V_3, V_5, V_6 上有正载荷，在 V_1, V_4 上有负载荷，且在 V_3 与 V_5 上有较高程度的载荷。

第三主成分：

$$Y_3 = 0.69702x_1 - 0.52182x_2 + 0.37836x_3 + 0.02159x_4 + 0.30646x_5 + 0.24070x_6$$

表示在 V_1, V_3, V_4, V_5, V_6 上有正载荷，在 V_2 上有负载荷，且在 V_1 上有较高程度的载荷。

第四主成分：

$$Y_4 = 0.60008x_1 + 0.71797x_2 + 0.03459x_3 + 0.21885x_4 - 0.15131x_5 - 0.22898x_6$$

表示在 V_1, V_2, V_3, V_4 上有正载荷，在 V_5, V_6 上有负载荷，且在 V_2 上有较高程度的载荷。

第五主成分：

$$Y_5 = 0.12030x_1 + 0.07859x_2 - 0.69199x_3 + 0.03876x_4 + 0.70585x_5 + 0.02777x_6$$

表示在 V_1, V_2, V_4, V_5, V_6 上有正载荷，在 V_3 上有负载荷，且在 V_5 上有较高程度的载荷。

综合以上分析，指标 V_1, V_2, V_3, V_5, V_6 在五个主成分中均占比较高，则可判定 $prop.elder$, $Population, density, pergdp, Meantem2020$ 这五个指标对于疫情发展情况均有较大影响。

5.4 温州市预测模型的建立与求解

除了题目中所给的数据外，我们从网上查找补充 [6] 了温州市 3 月 31 日之前的新增确诊病例，新增出院人数，新增死亡人数。

5.4.1 修正 $SEIR$ 预测模型建立

首先我们从病毒传播模型的角度对问题展开分析。新冠病毒具有二次感染性 [5]，即治愈的患者也有可能再次复阳。由于 $COVID-19$ 还具有潜伏期，因此我们这里选用考虑潜伏期的传染病模型 $SEIR$。该模型基本流程图如下：

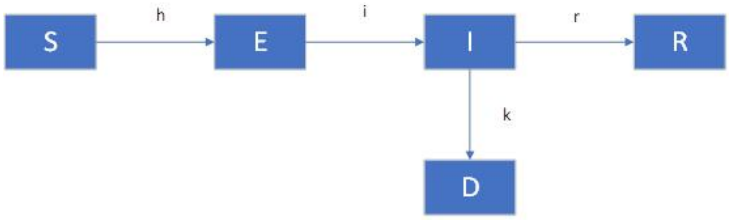


图 **4**　$SEIR$ 基本流程图

其中 S 为易感人群，I 为患病者，D 为死亡患者，E 为潜伏者，R 为康复者；h 为感染力，表示每位患者平均每日可感染人数，$h=\alpha\beta$, 其中 α 为接触率，即平均每个人接触到患者的概率，β 为感染效率，即被感染的概率；i 为转病率，r 为治愈率，k 为病死率。

建立微分方程组：

易感者可表示为：

$$dS/dt=-h(E+S) \tag{1}$$

潜伏者可表示为：

$$dE/dt=h(E+I)-iE \tag{2}$$

患病者可表示为：

$$dI/dt=iE-rl-kI \tag{3}$$

康复者可表示为：

$$dR/dt=rI \tag{4}$$

病死患者可表示为：

$$dD/dt=kI \tag{5}$$

但是这种情况只适用于现阶段不加以任何应对措施，并忽略人口出生，死亡等影响，所以会导致患病者相对减少。因此对该模型进行修正，修正结果如下：

易感者可表示为：

$$S_{t+1}=S_t-r\beta I_t-r_2\beta_2E_t+\beta_3E_{t-14} \tag{6}$$

潜伏者可表示为：

$$E_{t+1}=E_t+r\beta I_t+r_2\beta_2E_t-\beta_3E_{t-14} \tag{7}$$

患病者可表示为：

$$I_{t+1}=I_t+\alpha E_t-(\gamma+k)I_t \tag{8}$$

康复者可表示为：

$$R_{t+1}=R_t+\gamma I_t \tag{9}$$

病死患者可表示为：

$$Dt+1=D_t+kI_t \tag{10}$$

其中，α 为潜伏者转阳率，β 为传染概率 (接触者，患者),r 为平均每个病人每天接触的人数，γ 为患者康复概率,β_2 为传染概率 (接触潜伏者),β_3 为潜伏者的转阴率，r_2 为潜伏者平均每天接触的人数，k 为患者死亡率。

5.4.2 修正 $SEIR$ 预测模型求解与分析

温州市 $SEIR$ 疫情情况预测结果如图：

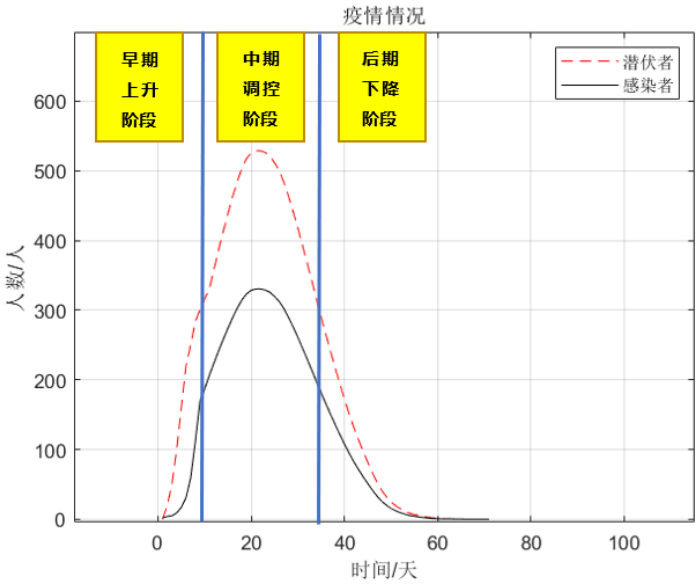


图 **5**　$SEIR$ 预测结果

可以看出，该曲线共分为三个阶段，第一个阶段是早期上升阶段：属于病毒爆发期，若不加以控制，潜伏者和感染者人数还将持续上升，如图 6 所示；第二阶段为中期调控阶段，经过封城隔离，佩戴口罩等措施，患病人数虽仍有上升，但是后期得到了控制；第三阶段为后期下降阶段，该阶段由于前面的措施不断进行，患者人数下降明显，最后患病人数趋于 0。

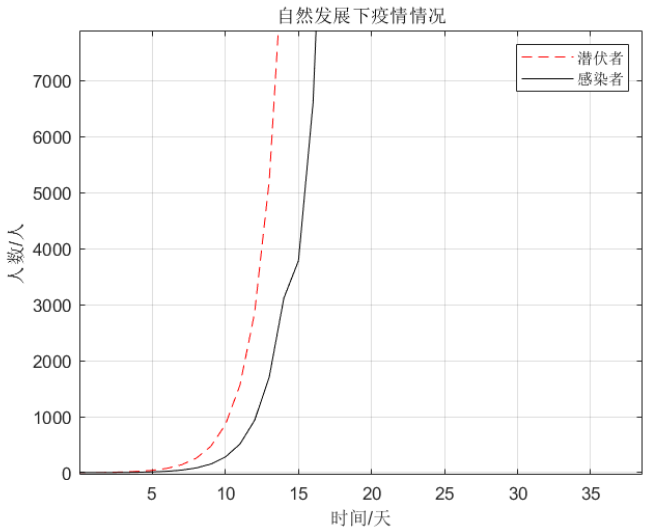


图 **6**　自然发展下疫情情况

5.5 $ARMA$ 预测模型建立与求解

模型的定义 [4]：

ARMA 模型的形式为:

$$W_t = \sum_{i=1}^{p} \alpha_i W_{t-i} + \varepsilon_i + \sum_{i=1}^{q} \beta_i \varepsilon_{t-i}$$

基于时间序列分析的线性自回归移动平均 $ARMA$ 模型预测方法认为康复率与过去的康复率存在某种联系。它是一种结合了自回归模型 AR 和移动平均模型 MA 的混合模型，能够解决需要高阶 AR 或者 MA 模型才能充分描述数据动态结构的问题，基本思想是将两模型相结合使模型阶数保持很小，优化原有模型计算量过大的问题。

$ARMA$ 模型中我们选用指标 $W_{t+1} = \frac{x_{t+1}}{y_t}$ 做迭代来进行预测, 其中 W_t 代表第 t 天的康复率，x_t 代表第 t 天的新增出院人数，y_t 代表第 t 天的已有患者数。

上面我们用 $SEIR$ 模型从病毒传播模型角度展开了分析，接着我们用 $ARMA$ 模型从统计层面进行预测与分析。

第一步：导入实验数据

指标 W_t28 天的变化如图：

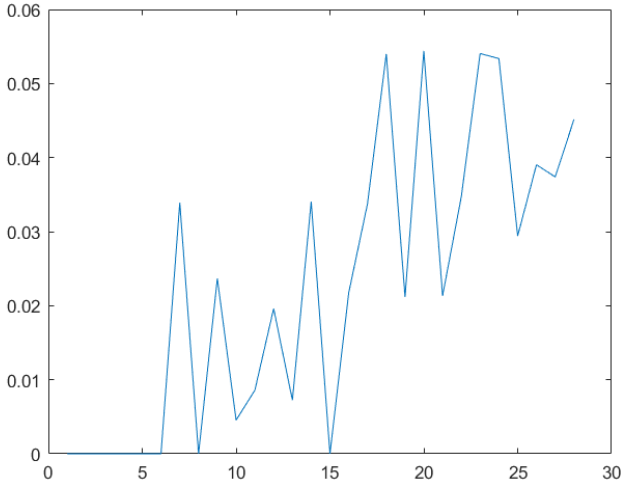


图 **7**　**28** 天指标变化情况

第二步：平稳性检验

使用 $ARMA$ 模型要求时间序列必须是平稳的，我们用 ADF 和 $KPSS$ 进行检验。

得到 $y\_h\_adf = 0$,$y\_h\_kpss = 0$, 没有通过检验。

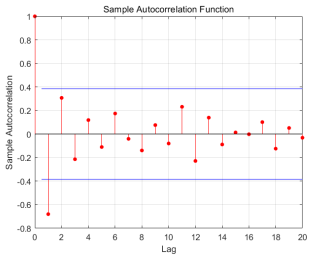
此时对 Y 取差分，再进行检验：

得到 $yd_1\_h\_adf = 0$,$yd_1\_h\_kpss = 1$，通过检验。

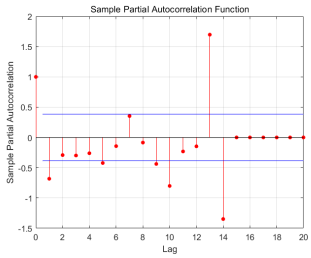
第三步：确定 $ARMA$ 模型阶数

我们用 ACF(自相关函数) 和 $PACF$(偏自相关函数) 法来确定模型阶数:

做出的结果如下两图：



(a) 自相关函数 ACF



(b) 偏自相关函数 $PACF$

对于我们关注的 $PACF(p,q)$，通俗的说，$PACF$ 最后一个在蓝线外 (即阈值外) 的 Lag 值就是 p 值；ACF 最后一个在蓝线外 (即阈值外) 的 Lag 值就是 q 值。按照这个标准，$p = 14$,$q = 1$。

第四步：残差要求：

为了确保确定的阶数合适，还需要进行残差检验。残差即原始的信号减弱模型拟合出的信号后的残余信号。如果残差是随机正态分布的、不自相关的，这说明残差是一段白噪声信号，也就说明有用的信号已经都被提取到 $ARMA$ 模型中了。

通过 $DurbinWatson$(杜斌-瓦特森) 函数对相关性进行检验：

运算结果为 2.11847，这个值越接近 2 越说明残差不存在一阶相关性。

上述检验可以证明，残差接近正态分布，且相互独立，可以认为 $ARMA$ 建模符合要求。

第五步：预测

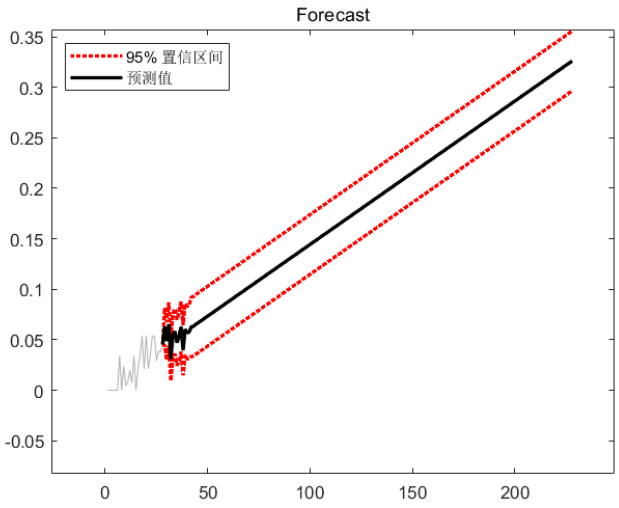


图 **9**　指标 W_t 的预测

上图中灰线为用来训练的 28 个数据，黑线为未来值的预测，红线为 95% 置信区间上下限，也就是说未来真实值有 95% 的概率落在这个范围内。可以看到使用 $ARMA$ 方法进行长期预测的结果是趋势性的。

第六步：$ARMA$ 模型的疫情情况预测：

由赛题数据可知，新增确诊人数在第 29 天后可以忽略不计，因此我们选择公式

$$I_{t+1} = I_t - I_t W_{t+1}$$

做迭代来预测第 29 天之后的疫情情况。

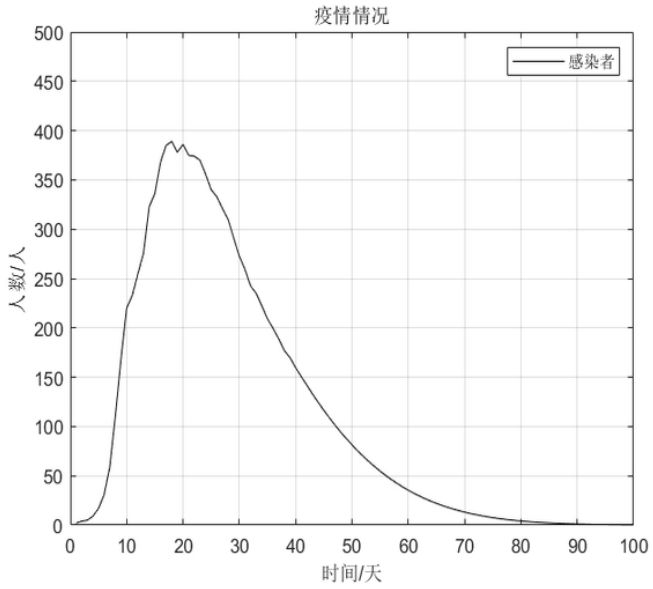


图 **10** $ARMA$ 模型的疫情情况预测

可以看出，$ARMA$ 模型预测结果跟 $SEIR$ 模型预测结果较为一致，跟真实数据也比较相符。

5.5.1 疫情防控建议

温州市 1 月平均气温为 10.4 摄氏度，新冠病毒在低温下易于长期存活 [2]。温州市的气温比较适合新冠病毒存活，因此应该加强管控措施。如对无症状感染者进行排查，禁止非必要的聚集性活动，严格执行戴口罩等措施。在温州市疫情爆发阶段，如果不采取任何措施，每个病人每天接触的人数 r，潜伏者平均每天接触的人数 r_2 处于一个高的水平，短时间内，感染人数会呈现指数级上升。因此管控措施十分必要，管控措施不限于：1. 隔离确诊者 2. 密切关注和观察可能的潜伏者 3. 戴口罩 4. 避免聚集性活动。

六、 模型总结

6.1 模型优点

- 模型假设合理，考虑全面，与真实情况较为符合
- 除了题目所给的数据外，建模时网上查找了更多的数据使模型更具真实性
- 运用 $SEIR$ 和 $ARMA$ 两种预测模型进行疫情预测，结果更为可靠
- 引入潜伏者的 $SEIR$ 模型相对于传统的 $SI, SIR, SEIR$ 和 SIS 模型有明显优越性，尤其是在通过分段拟合等措施改进模型后，整体拟合效果得到大幅提升。可以给 $COVID-19$ 等类似传染病疫情防控提供较大的参考和指导价值 [3]。

- $ARMA$ 模型不依赖外界因素对康复率的干扰信息，不需要建立精确的疾病传播模型，仅仅依靠康复率的历史数据就可以完成单步或多步预测，有效预报时间较长。$ARMA$ 模型综合了自回归模型 AR 和移动平均模型 MA 的优点，保存了较小的阶数，有效的减小了计算量 [4]。

6.2 模型缺点

- ARMA 模型中没有考虑后期重大医学进展可能导致康复率的突变
- SEIR 模型没有考虑每个人的潜伏期可能是不同的，而将所有人的潜伏期设为 14 天

参考文献

[1] U.S. Data and Statistics | USAGov,https://www.usa.gov/statistics,2021 年 6 月 5 日

[2] 温度对新冠病毒稳定性的影响 [J]. 中国比较医学杂志,2021,31(03):103.

[3] 林俊锋. 基于引入隐形传播者的 SEIR 模型的 COVID-19 疫情分析和预测 [J]. 电子科技大学学报,2020,49(03):375-382.

[4] 张凯临, 李玉超. 基于 ARMA 模型船摇预报的船用稳定平台 PID 控制算法研究 [J]. 中国海洋大学学报 (自然科学版),2021,51(07):115-121.

[5] 张莹, 王丽丽, 张家伟. 新冠二次感染病例带来哪些防疫启示 [J]. 健康中国观察,2020(09):44-45.

[6] 浙江省卫生健康委员会,https://wsjkw.zj.gov.cn/,2021 年 6 月 5 日.

附录 A　问题一所有判断结果

表 **8**　判断结果

城市	确诊率	赋分	严重程度
$Wenzhou$	5.42×10^{-5}	0.195250723534233	0
$Shanghai$	1.37×10^{-5}	0	0
$Chongqing$	1.79×10^{-5}	0.020049006029124	0
$Arizona-Maricopa$	1.23×10^{-4}	0.530039876966016	0
$Arizona-Pima$	1.47×10^{-4}	0.644362742266113	0
$Arizona-Pinal$	1.47×10^{-4}	0.644362742266113	0
$Arkansas-Jefferson$	3.81×10^{-4}	1.77599281806472	0
$California-Alameda$	1.61×10^{-4}	0.715527994196268	0
$California-ContraCosta$	1.52×10^{-4}	0.668006517264771	0
$California-Fresno$	4.32×10^{-5}	0.142394851369714	0
$California-Humboldt$	1.32×10^{-4}	0.570732551756564	0
$California-LosAngeles$	1.31×10^{-4}	0.570732551756564	0
$California-Madera$	2.11×10^{-4}	0.953846592303555	0
$California-Orange$	9.51×10^{-5}	0.392846967565178	0
$California-Placer$	1.35×10^{-4}	0.586610315895895	0
$California-Riverside$	1.35×10^{-4}	0.584333382452345	0
$California-Sacramento$	9.51×10^{-5}	0.392549776082824	0
$California-SanBenito$	4.58×10^{-4}	1.03172420412192	0
$California-SanDiego$	1.06×10^{-4}	0.447346382507415	0
$California-SanFrancisco$	2.28×10^{-4}	1.03172420412192	0

表 **9** 判断结果

城市	确诊率	赋分	严重程度
$California-SanMateo$	3.60×10^{-4}	1.67096996268283	0
$California-SantaClara$	3.33×10^{-4}	1.54284986196598	0
$California-SantaCruz$	1.49×10^{-4}	0.655218656813299	0
$California-Shasta$	2.22×10^{-5}	0.0409202750355879	0
$California-Solano$	7.61×10^{-5}	0.301120427156331	0
$California-Sonoma$	1.16×10^{-4}	0.493618741326706	0
$California-Ventura$	1.28×10^{-4}	0.551902986499722	0
$California-Yolo$	7.26×10^{-5}	0.284052251491252	0
$Colorado-Arapahoe$	3.70×10^{-4}	1.71983497114676	0
$Colorado-Denver$	5.69×10^{-4}	2.68204495817569	0
$Colorado-Douglas$	2.98×10^{-4}	1.36988648767497	0
$Colorado-Eagle$	3.40×10^{-3}	16.3455618553139	1
$Colorado-ElPaso$	3.00×10^{-4}	1.38055247394111	0
$Colorado-Gunnison$	4.58×10^{-3}	22.0423605521108	1
$Colorado-Jefferson$	3.67×10^{-4}	1.705435089028	0
$Colorado-Larimer$	2.40×10^{-4}	1.09031640220141	0
$Connecticut-Fairfield$	1.32×10^{-3}	6.30021932662992	0
$Connecticut-Litchfield$	4.80×10^{-4}	2.2521438066629	0
$Florida-Alachua$	2.67×10^{-4}	1.22094215873969	0
$Florida-Broward$	5.19×10^{-4}	2.43685446744	0

表 **10**　判断结果

城市	确诊率	赋分	严重程度
$Florida-Charlotte$	8.65×10^{-5}	0.351114085871699	0
$Florida-Clay$	1.62×10^{-4}	0.715486650496574	0
$Florida-Collier$	2.93×10^{-4}	1.34914139861021	0
$Florida-Duval$	1.49×10^{-4}	0.654973976683056	0
$Florida-Hillsborough$	1.57×10^{-4}	0.689449935254001	0
$Florida-Nassau$	9.32×10^{-5}	0.383537450370527	0
$Florida-Okaloosa$	1.54×10^{-4}	0.678833463652783	0
$Florida-Pasco$	6.49×10^{-5}	0.246727762355786	0
$Florida-Pinellas$	1.19×10^{-4}	0.507744398987832	0
$Florida-SantaRosa$	1.15×10^{-4}	0.471904564472485	0
$Florida-Sarasota$	1.43×10^{-4}	0.623632934037343	0
$Florida-Seminole$	1.73×10^{-4}	0.769330201225611	0
$Florida-Volusia$	1.10×10^{-4}	0.462575187313334	0
$Georgia-Bartow$	1.12×10^{-3}	5.33123954565039	0
$Georgia-Charlton$	7.71×10^{-5}	0.305868801025836	0
$Georgia-Cherokee$	2.36×10^{-4}	1.07311826718378	0
$Georgia-Cobb$	3.01×10^{-4}	1.38761002201063	0
$Georgia-DeKalb$	3.60×10^{-4}	1.67527422007596	0
$Georgia-Floyd$	2.86×10^{-4}	1.313691898878	0
$Georgia-Fulton$	4.05×10^{-4}	1.88702407710457	0

表 **11**　判断结果

城市	确诊率	赋分	严重程度
$Georgia-Gordon$	2.42×10^{-4}	1.10504699776703	0
$Georgia-Gwinnett$	1.56×10^{-4}	0.687994664659705	0
$Georgia-Lee$	1.44×10^{-4}	6.90639200465824	0
$Georgia-Lowndes$	1.72×10^{-4}	0.763533318119738	0
$Illinois-Cook$	1.47×10^{-4}	0.644362742266113	0
$Illinois-Kane$	6.65×10^{-4}	3.14322507254602	0
$Illinois-McHenry$	1.87×10^{-4}	0.837146642021196	0
$Indiana-Adams$	1.69×10^{-4}	0.747033279662473	0
$Indiana-Boone$	2.81×10^{-5}	0.0691268849448519	0
$Indiana-Hendricks$	1.94×10^{-4}	0.870171553812666	0
$Indiana-Howard$	2.87×10^{-4}	1.32084650708082	0
$Indiana-Johnson$	1.46×10^{-3}	0.636855512558961	0
$Indiana-Marion$	4.54×10^{-4}	2.12716073911053	0
$Indiana-Nobel$	7.08×10^{-4}	3.35125990236502	0
$Iowa-Johnson$	4.21×10^{-5}	0.13677093189618	0
$Iowa-Pottawattamie$	4.30×10^{-4}	2.0077117076983	0
$Kansas-Johnson$	3.21×10^{-5}	0.0884939531761216	0
$Kentueky-Jeffersen$	1.81×10^{-4}	0.805998466673646	0
$Louisiana-Caddo$	9.02×10^{-4}	4.28481174872151	0
$Louisiana-Jefferson$	1.75×10^{-3}	8.39558893613612	0

表 **12** 判断结果

城市	确诊率	赋分	严重程度
$Louisiana-Orleans$	3.45×10^{-3}	16.5974986296562	1
$Maryland-Baltimore-city$	1.96×10^{-4}	0.877496287244345	0
$Maryland-Harford$	9.06×10^{-5}	0.3708031623984	0
$Maryland-Montgomery$	2.86×10^{-4}	1.31388626673301	0
$Maryland-PrinceGeorge's$	2.72×10^{-4}	1.24471222568865	0
$Massachusetts-Berkshire$	3.54×10^{-4}	1.64056350912062	0
$Massachusetts-Essex$	1.63×10^{-4}	0.722158110391281	0
$Massachusetts-Middlesex$	1.50×10^{-4}	0.657325385358196	0
$Massachusetts-Norfolk$	1.87×10^{-4}	0.835154324041034	0
$Massachusetts-Worcester$	5.14×10^{-4}	2.41306852103547	0
$Michigan-Oakland$	1.24×10^{-3}	5.93384091084475	0
$Michigan-Wayne$	4.41×10^{-4}	2.06327845929466	0
$Minnesota-Anoka$	9.68×10^{-4}	4.60413548949735	0
$Minnesota-Ramsey$	2.08×10^{-4}	0.93544659470109	0
$Missouri-St.Louis$	1.69×10^{-4}	0.748893157757913	0
$Nebraska-Douglas$	3.56×10^{-4}	1.65025363232244	0
$Nebraska-Knox$	7.88×10^{-4}	3.73590789003006	0
$Nevada-Clark$	5.14×10^{-4}	2.41441892151016	0
$Nevada-Washoe$	3.37×10^{-4}	1.56217704292635	0
$NewHampshire-Grafton$	9.40×10^{-5}	0.38735356373896	0

表 **13** 判断结果

城市	确诊率	赋分	严重程度
$NewHampshire-Rockingham$	3.23×10^{-4}	1.49474536359582	0
$NewJersey-Bergen$	2.23×10^{-2}	11.1096971664415	0
$NewJersey-Burlington$	3.19×10^{-4}	1.47247454403097	0
$NewJersey-Hudson$	1.44×10^{-3}	6.88708688635921	0
$NewJersey-Monmouth$	1.40×10^{-3}	6.69147080067248	0
$NewJersey-Passaic$	1.65×10^{-3}	7.90242537282203	0
$NewJersey-Union$	1.16×10^{-3}	7.68268906965183	0
$NewMexico-Bernalillo$	1.49×10^{-4}	0.651925998186806	0
$NewMexico-SantaFe$	2.27×10^{-4}	1.02726785149077	0
$NewMexico-Socorro$	1.12×10^{-4}	0.510494520099528	0
$NewYork-Nassau$	4.75×10^{-3}	22.8337906368178	1
$NewYork-NewYork$	2.07×10^{-2}	100	1
$NewYork-Rockland$	6.78×10^{-3}	32.6684031141235	1
$NewYork-Saratoga$	4.69×10^{-4}	2.19839830075107	0
$NewYork-Suffolk$	3.39×10^{-3}	16.3020350582649	1
$NewYork-Ulster$	1.04×10^{-3}	4.96009966322204	0
$NewYork-Westchester$	8.80×10^{-3}	42.4260923769903	1
$NorthCarolina-Chatham$	1.78×10^{-4}	0.791554223656297	0
$NorthCarolina-Forsyth$	9.23×10^{-5}	0.379284600781249	0
$NorthCarolina-Johnston$	4.93×10^{-4}	0.171825894663712	0

表 **14** 判断结果

城市	确诊率	赋分	严重程度
$NorthCarolina-Mecklenburg$	3.07×10^{-4}	1.41615806681018	0
$NorthCarolina-Onslow$	2.02×10^{-5}	0.0313501413527092	0
$NorthCarolina-Wake$	1.34×10^{-4}	0.578799237176675	0
$Ohio-Cuyahoga$	3.54×10^{-4}	1.64097603122018	0
$Ohio-Stark$	7.80×10^{-5}	0.310374114748494	0
$Oklahoma-Tulsa$	9.41×10^{-5}	0.387776012687061	0
$Oregon-Deschutes$	1.20×10^{-4}	0.511865960018395	0
$Oregon-Douglas$	3.63×10^{-4}	0.108745948816399	0
$Oregon-Jackson$	8.65×10^{-5}	0.351344362314303	0
$Oregon-Klamath$	5.91×10^{-5}	0.219053082360364	0
$Oregon-Marion$	3.14×10^{-5}	0.45034341290671	0
$Oregon-Multnomah$	1.12×10^{-4}	0.474660965239395	0
$Oregon-Polk$	1.53×10^{-4}	0.669820405692587	0
$Oregon-Umatilla$	5.16×10^{-4}	0.182744063205256	0
$Oregon-Washington$	2.58×10^{-4}	1.1772456422548	0
$Pennsylvania-Bucks$	3.96×10^{-4}	1.84675144723243	0
$Pennsylvania-Delaware$	4.89×10^{-5}	2.29240839895973	0
$Pennsylvania-Monroe$	7.96×10^{-3}	3.77757253612911	0
$Pennsylvania-Montgomery$	5.89×10^{-4}	2.77616594606617	0
$Pennsylvania-Philadelphia$	5.62×10^{-4}	2.64526112842927	0

表 **15** 判断结果

城市	确诊率	赋分	严重程度
$Pennsylvania-Wayne$	1.37×10^{-4}	0.592572873118169	0
$RhodeIsland-Providence$	2.45×10^{-4}	1.11736987050297	0
$SouthCarolina-Charleston$	2.88×10^{-4}	1.3248771636805	0
$SouthCarolina-Kershaw$	1.27×10^{-3}	6.04101694397202	0
$SouthCarolina-Lancaster$	1.36×10^{-4}	0.591514966741759	0
$SouthCarolina-Spartanburg$	7.96×10^{-5}	0.318095778048218	0
$SouthDakota-Beadle$	1.06×10^{-3}	5.04559720889902	0
$SouthDakota-CharlesMix$	1.07×10^{-4}	0.450547188619204	0
$SouthDakota-Davison$	1.01×10^{-4}	0.421452786738768	0
$SouthDakota-Minnehaha$	1.3×10^{-4}	0.559274755870524	0
$SouthDakota-Pennington$	3.58×10^{-5}	0.106480377657788	0
$Tennessee-Davidson$	5.69×10^{-4}	2.6793445291044	0
$Tennessee-Shelby$	3.87×10^{-4}	1.80078445305341	0
$Tennessee-Sullivan$	5.07×10^{-5}	0.178580219772951	0
$Tennessee-Williamson$	4.36×10^{-4}	2.03729900615095	0
$Texas-Collin$	1.33×10^{-4}	0.577118031069104	0
$Texas-Dallas$	1.85×10^{-4}	0.826598131318664	0
$Texas-FortBend$	1.51×10^{-4}	0.662682885950094	0
$Texas-Gregg$	3.23×10^{-5}	0.0897499123505302	0
$Texas-Harris$	1.12×10^{-4}	0.473996163108719	0

表 **16**　判断结果

城市	确诊率	赋分	严重程度
$Texas-Montgomery$	1.10×10^{-4}	0.464581056847152	0
$Texas-Tarrant$	6.67×10^{-5}	0.255461712489964	0
$Utah-Davis$	1.96×10^{-4}	0.880548020480865	0
$Utah-Weber$	1.44×10^{-4}	0.630280344796118	0
$Vermont-Bennington$	4.49×10^{-4}	2.10097488854259	0
$Virginia-Arlington$	3.54×10^{-4}	1.64056350912062	0
$Virginia-Fairfax$	3.81×10^{-4}	1.77599281806472	0
$Virginia-Loudoun$	1.61×10^{-4}	0.715527994196268	0
$Washington-Clark$	1.52×10^{-4}	0.668006517264771	0
$Washington-Grant$	4.32×10^{-5}	0.142394851369714	0
$Washington-Island$	1.32×10^{-4}	0.570732551756564	0
$Washington-Jefferson$	3.40×10^{-3}	16.3455618553139	1
$Washington-King$	2.11×10^{-4}	0.953846592303555	0
$Washington-Kitsap$	9.51×10^{-5}	0.392846967565178	0
$Washington-Kittitas$	1.35×10^{-4}	0.586610315895895	0
$Washington-Pierce$	1.35×10^{-4}	0.584333382452345	0
$Washington-Skagit$	9.51×10^{-5}	0.392549776082824	0
$Washington-Whatcom$	4.58×10^{-3}	22.0423605521108	1
$Wisconsin-Dane$	1.06×10^{-4}	0.447346382507415	0
$Wisconsin-Pierce$	2.28×10^{-4}	1.03172420412192	0

表 **17** 判断结果

城市	确诊率	赋分	严重程度
$California-Imperial$	1.37×10^{-4}	0.597289347076996	0
$California-Marin$	3.58×10^{-4}	1.66227971137814	0
$California-SanLuisObispo$	2.50×10^{-4}	1.14024978989903	0
$Colorado-Summit$	4.19×10^{-4}	1.95721066218127	0
$Florida-Escambia$	1.17×10^{-4}	0.499642452397125	0
$Florida-Lake$	1.23×10^{-4}	0.529384505903963	0
$Florida-Lee$	2.01×10^{-4}	0.905866719465248	0
$Florida-Manatee$	9.62×10^{-5}	0.398173664122441	0
$Florida-Sumter$	3.50×10^{-4}	1.62053761188619	0
$Georgia-Coweta$	1.58×10^{-4}	0.694723859230293	0
$Georgia-Fayette$	2.38×10^{-4}	1.08223748444798	0
$Georgia-Polk$	2.35×10^{-4}	1.07011789064852	0
$Kentucky-Harrison$	5.86×10^{-4}	2.76096044089949	0
$Louisiana-Ascension$	1.15×10^{-4}	5.31477473027687	0
$Massachusetts-Suffolk$	1.16×10^{-4}	5.55377234076366	0
$NewJersey-Camden$	3.21×10^{-4}	1.48513690263028	0
$NorthCarolina-Cabarrus$	1.51×10^{-4}	0.664473010989613	0
$NorthCarolina-Durham$	3.57×10^{-4}	1.65556240486047	0
$Virginia-Henrico$	1.21×10^{-4}	0.520022634246052	0
$Washington-Snohomish$	1.318×10^{-4}	6.25912432836043	0

附录 B　代码源程序

2.1　预处理

```
preprocessing.py
import numpy as np
import pandas as pd

f=pd.read_excel(r".\Data.xlsx")
cc=f.values
ctname=1
day=1
data=cc
ctMap=[[1,cc[0,1]]]
tempc = cc[0, 1]
Increase=[cc[0,9]]
for i in range(5800):

if cc[i,1]==tempc:
data[i,1]=ctname
data[i,8]=day
day=day+1
if i!=0:
inc=cc[i,9]-cc[i-1,9]
Increase.append(inc)
else:
tempc=cc[i,1]
ctname=ctname+1
day=1
ctMap.append([ctname,tempc])
data[i, 1] = ctname
data[i, 8] = day
day = day + 1
Increase.append(cc[i,9])
inc_np=np.array(Increase)
inc_np=inc_np.reshape(-1,1)
data2=np.hstack((data,inc_np))
ctMap_np=np.array(ctMap)
```

2.2　$k-means$

```
  Kmeans.py
  import pandas as pd
  import numpy as np
  import sklearn.preprocessing as pre
```

```
from sklearn.cluster import KMeans
data=pd.read_csv(r".\data2.csv",delimiter=",",header=None)
data=data.values
score=[]

for i in range(200):
soc=data[(i+1)*29-1,9]/data[(i+1)*29-1,3]
score.append(soc)
ratio=np.array(score)
score=pre.minmax_scale(ratio)*100
```

2.3 主成分分析

```
PCA.py
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
import numpy as np
from scipy.stats import kstest
data=pd.read_csv(r ".\PcaPre.csv",delimiter=",",header=None)
cc=data.describe()
data=data.values
hh=[]
pp=[]
for i in range(7):
[h,p]=kstest(data[:,i],"norm")
hh.append(h)
pp.append(p)
data2=data[:,0:6]
data2 = scale(data2) # 标准化
# data2 = np.corrcoef(np.transpose(data)) #计算协方差阵或相关系数阵
pca = PCA(n_components=6) #调整主成分个数
pca.fit(data2)
aa=pca.explained_variance_ # 输出特征根
bb=pca.explained_variance_ratio_ # 输出解释方差比
dd=np.sum(bb)
cc=pca.components_ # 输出主成分
PcaData=data2.dot(np.transpose(cc))
PcaData=np.hstack((PcaData,data[:,6].reshape(200,1)))
```

2.4 $SEIR$ 模型

```
SEIR.m
```

```
clear;clc;
%参数设置
N=9300000;
I=2;%传染者
R=0;%康复者
D=0;%死亡患者数量
E=0;%潜伏者
S=N-I;% 易感染者
T=0:200;
r=2;%接触病患的人数
a=0.325;%潜伏者患病概率
B=0.8;%感染概率
y=0.143;%康复概率
B2=0.03;%接触潜伏者
B3=0.0859;%转阴率
r2=10;%潜伏者每天接触的人数
k=0.025373;%死亡率

for idx =1:14
S(idx+1)=S(idx)-r*B*I(idx)-r2*B2*E(idx);%易感人数迭代
E(idx+1)=E(idx)+r*B*I(idx)+r2*B2*E(idx);%潜伏者人数迭代
I(idx+1)=I(idx)+a*E(idx)-(y+k)*I(idx);%患病人数迭代
R(idx+1)=R(idx)+y*I(idx);%康复人数迭代
D(idx+1)=D(idx)+k*I(idx);%死亡患者人数迭代
end
r=5;%接触病患的人数
a=0.125;%潜伏者患病概率
B=0.8;%感染概率
y=0.143;%康复概率
B2=0.03;%接触潜伏者
B3=0.859;%转阴率
r2=20;%潜伏者每天接触的 人数
k=0.025373;%死亡率

for idx =15:T(-1)
S(idx+1)=S(idx)-r*B*I(idx)-r2*B2*E(idx)+B3*E(idx-14);%易感人数迭代
E(idx+1)=E(idx)+r*B*I(idx)+r2*B2*E(idx)-B3*E(idx-14);%潜伏者人数迭代
I(idx+1)=I(idx)+a*E(idx)-(y+k)*I(idx);%患病人数迭代
R(idx+1)=R(idx)+y*I(idx);%康复人数迭代
D(idx+1)=D(idx)+k*I(idx);%死亡患者人数迭代
end

plot(T,E,'--r',T,I,'-k');
grid on;
xlabel('时间/天');
ylabel('人数/人');
axis([0 100 0 500]);
```

```
%legend('潜伏者','感染者','康复者','死亡者');
legend('潜伏者','感染者');
% set(gca,'ytick',[0:50:500]);
title('自然发展下疫情情况');
```

2.5 $ARMA$ 模型

```
ARMA.m
%导入数据
%plot(Y);
y_h_adf = adftest(Y);
y_h_kpss = kpsstest(Y);
Yd1 = diff(Y);
yd1_h_adf = adftest(Yd1);
yd1_h_kpss = kpsstest(Yd1);
data=Y;
figure
autocorr(Yd1)
figure
parcorr(Yd1)
AR_Order=1;
MA_Order=14;
Mdl = arima(AR_Order, 1, MA_Order); %第二个变量值为1，即是一阶差分
EstMdl = estimate(Mdl,data);
[res,~,logL] = infer(EstMdl,data); %res即残差

% Durbin-Watson 统计是计量经济学分析中最常用的自相关度量
diffRes0 = diff(res);
SSE0 = res'*res;
DW0 = (diffRes0'*diffRes0)/SSE0 ;% Durbin-Watson
    statistic，该值接近2，则可以认为序列不存在一阶相关性。
step = 200; %预测步数为300
[forData,YMSE] = forecast(EstMdl,step,'Y0',data);
lower = forData - 1.96*sqrt(YMSE); %95置信区间下限
upper = forData + 1.96*sqrt(YMSE); %95置信区间上限

figure()
plot(data,'Color',[.7,.7,.7]);
hold on
h1 = plot(length(data):length(data)+step,[data(end);lower],'r:','LineWidth',2);
plot(length(data):length(data)+step,[data(end);upper],'r:','LineWidth',2)
h2 = plot(length(data):length(data)+step,[data(end);forData],'k','LineWidth',2);
legend([h1 h2],'95% 置信区间','预测值',...
'Location','NorthWest')
title('Forecast')
```

```
hold off
```