Поповкин Артемий Андреевич
Б9122-02.03.01 СЦТ

Постановка задачи

Требуется:

1. Для модели Мальтуса:
    1. определить численность популяции в будущем времени на основе начальной численности и темпа роста.
2. Для логистической модели:
    1. Исследовать изменение численности популяции с течением времени с учетом начальной численности, темпа роста и емкости среды.

Описание моделей

Модель Мальтуса

Описание

Модель Мальтуса описывает процесс неограниченного роста популяции в условиях избытка ресурсов.
Основное предположение модели - скорость роста популяции пропорциональна ее текущей численности.

Обозначения

- t — время (размерность: годы)

Входные данные

- P_0 — начальная численность популяции. (размерность: тысячи)

Фазовые переменные

- $P(t)$ — численность популяции в момент времени t.

Параметры

- r — темп роста (разница между рождаемостью и смертностью в единицу времени)

Уравнение

$$\frac{dP}{dt} = rP$$

Решением уравнения будет называть функцию, удовлетворяющую следующей задаче Коши:

$$\begin{cases} P' = rP \\ P(0) = P_0 \end{cases}$$

Логистическая модель

Описание

Обозначения

- t — время (размерность: годы)

Входные данные

- P_0 — начальная численность популяции. (размерность: тысячи)

Фазовые переменные

- $P(t)$ — численность популяции в момент времени t.

Параметры

- r — темп роста (разница между рождаемостью и смертностью в единицу времени)
- k — ёмкость среды (максимальная численность популяции, которую может поддерживать среда)

Уравнение

$$\frac{dP}{dt} = rP\left(1 - \frac{P}{k}\right)$$

Решением уравнения будет называть функцию, удовлетворяющую следующей задаче Коши:

$$\begin{cases} P' = rP\left(1 - \frac{P}{k}\right) \\ P(0) = P_0 \end{cases}$$

Аналитическое исследование систем

Модель Мальтуса

Поиск аналитического решения

$\frac{dP}{dt} = rP$
$\frac{dP}{P} = rdt$
$\ln(P) = rt + C \implies P(t) = Ce^{rt}$
Константу C получим из начального условия: $P_0 = P(0) = C$

Таким образом, общее решения задачи имеет вид:

$$P(t) = P_0 \cdot e^{rt}$$

Условие устойчивости

Для начала найдём равновесия модели:
$rP = 0 \implies P = 0$
Следовательно существует лишь тривиальное решение $P_t = 0$

Теперь исследуем полученную равновесную точку на устойчивость:
$(rP)'_x = r$

Получим следующие случаи:

1. $r > 0$
    1. Неподвижная точка P_t **неустойчива**, численность популяции неограниченно растёт
2. $r < 0$
    1. Неподвижная точка P_t **неустойчива**, численность популяции неограниченно растёт
3. $r = 0$
    1. Неподвижная точка P_t **устойчива**, численность популяции не меняется со временем

Логистическая модель

Поиск аналитического решения

$\frac{dP}{dt} = rP\left(1 - \frac{P}{k}\right)$
$\frac{PdP}{P(k-P)} = rdt$
$\frac{dP}{P} + \frac{dP}{k-P} = rdt$

$$\ln\left(\frac{P}{k-P}\right) = rt + C$$
$$\frac{P}{k-P} = Ce^{rt}$$
$$\left(1 + Ce^{rt}P\right) = kCe^{rt}$$
$$P\left(t\right) = \frac{kCe^{rt}}{1 + Ce^{rt}}$$

Константу C получим из начального условия:

$$\frac{kC}{1+C} = P_0 \implies kC = (1+C)P_0 \implies C = \frac{P_0}{k - P_0}$$

Таким образом, общее решения задачи имеет вид:

$$P\left(t\right) = \frac{kP_0 \cdot e^{rt}}{k - P_0 + P_0 e^{rt}}$$

Условие устойчивости

Для начала найдём равновесия модели:

$$\frac{dP}{dt} = rP\left(1 - \frac{P}{k}\right) = 0$$

Получаем два равновесия:

1. $P_t = 0$
2. $P_{nt} = k$

Теперь исследуем полученные точки на устойчивость:

$$\left(rP\left(1 - \frac{P}{k}\right)\right)'_P = r - \frac{rP}{k} - \frac{rP}{k} = r - 2\frac{rP}{k}$$

Подставляя неподвижные точки получим два случая:

1. $r > 0$
    1. Нетривиальное равновесие устойчиво при $\forall P > 0$.
2. $r < 0 \cap P_0 < k$
    1. Тривиальное равновесие устойчивым на всём интервале $(0, k)$.

Численные эксперименты

Для обоих моделей был выбран временной диапазон с 2014 по 2020 гг.

Популяция указана в тысячах.

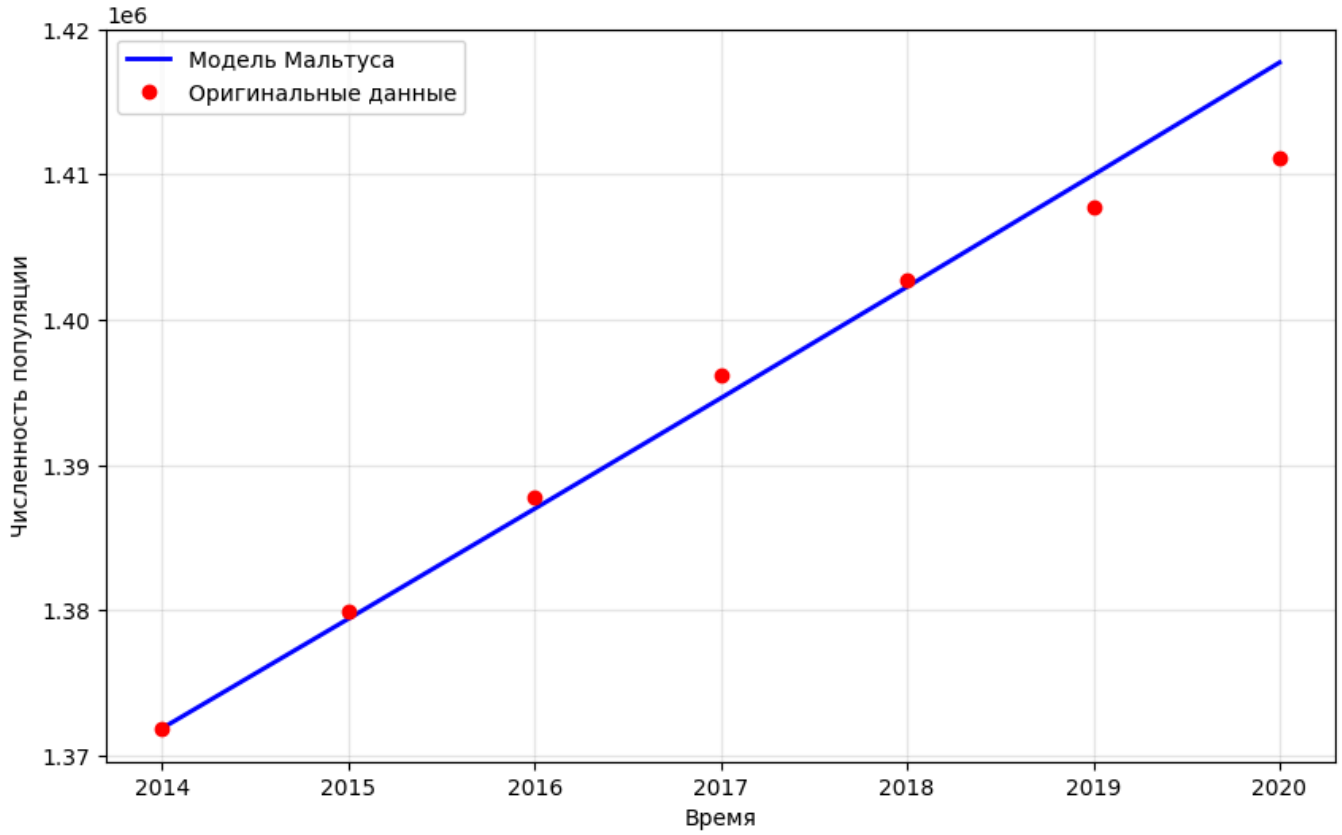
r высчитывал от P_0 до $P\left(t\right)$

Модель Мальтуса

Для Мальтуса была выбрана популяция Китая.
Начальные данные:

year	persons	birth_rate	death_rate	natural_rate	r
2014	1371860	13.83	7.12	6.71	-
2015	1379860	11.99	7.07	4.93	0.22587644624337813
2016	1387790	12.95	7.09	5.86	0.25566042905333936
2017	1396215	12.64	7.06	5.58	0.31242969685478583
2018	1402760	10.86	7.08	3.78	0.3654532792088563
2019	1407745	10.41	7.09	3.32	0.43956235125122173
2020	1411100	8.52	7.07	1.45	0.5830653853249727

$$r = \frac{1}{t}\ln\left(\frac{P\left(t\right)}{P_0}\right)$$
$$r_{mean} = 0.36367459798942564$$

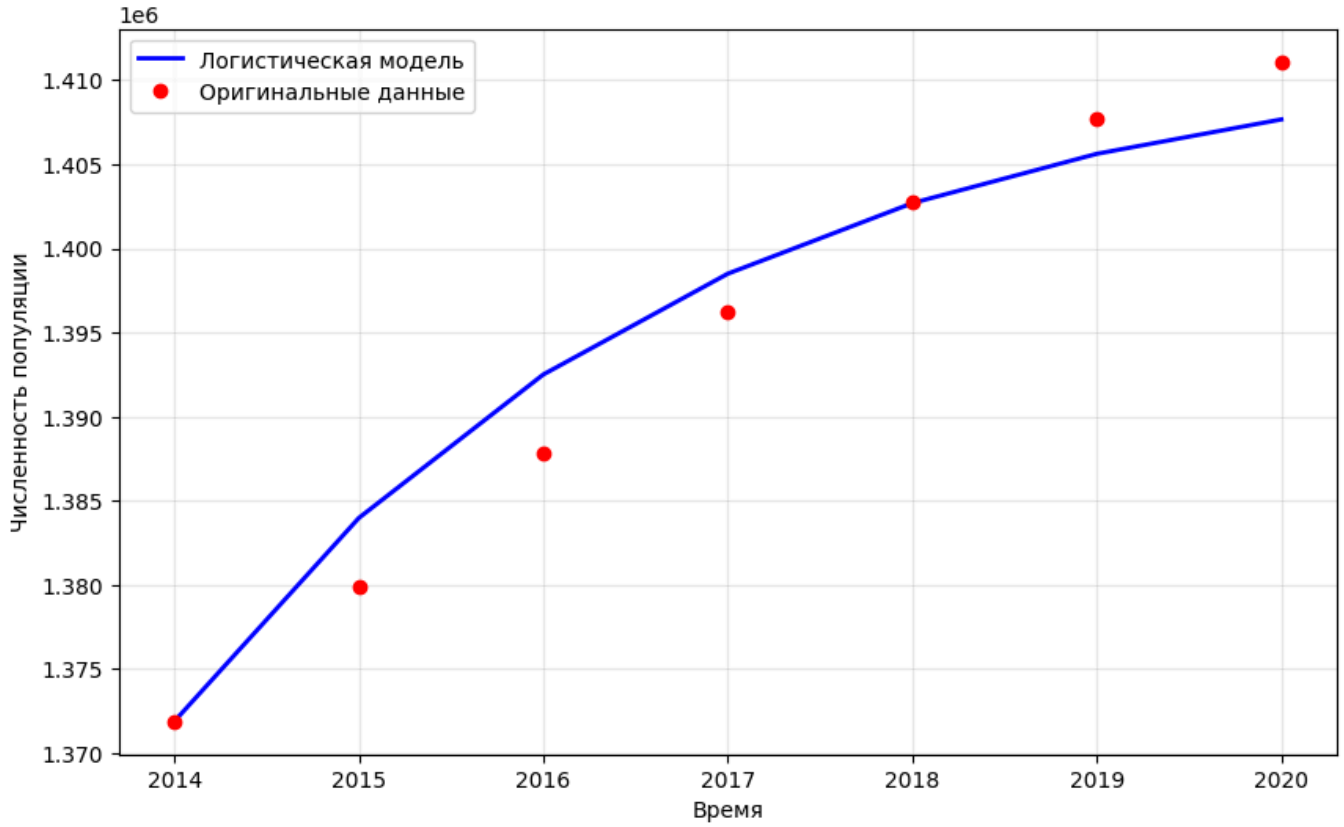


# year	# persons
2014	1371860.0
2015	1379399.8478057499
2016	1386981.1351934783
2017	1394604.089918308
2018	1402268.9409871195
2019	1409975.9186654342
2020	1417725.2544843291

Вывод

Примерно в 2021-м году Китай вышел на своё плато по количеству населения, и уже несколько лет население в нём падает. Но помимо этого результат вполне приемлемый.

Дополнительно



# year	# persons
2014	1371860.0
2015	1383959.451689284
2016	1392496.5078830295
2017	1398493.0822698437
2018	1402691.9243040143
2019	1405625.4977729782
2020	1407671.9153034603

Логистическая же модель для этих данных показывает гораздо более хорошие результаты!

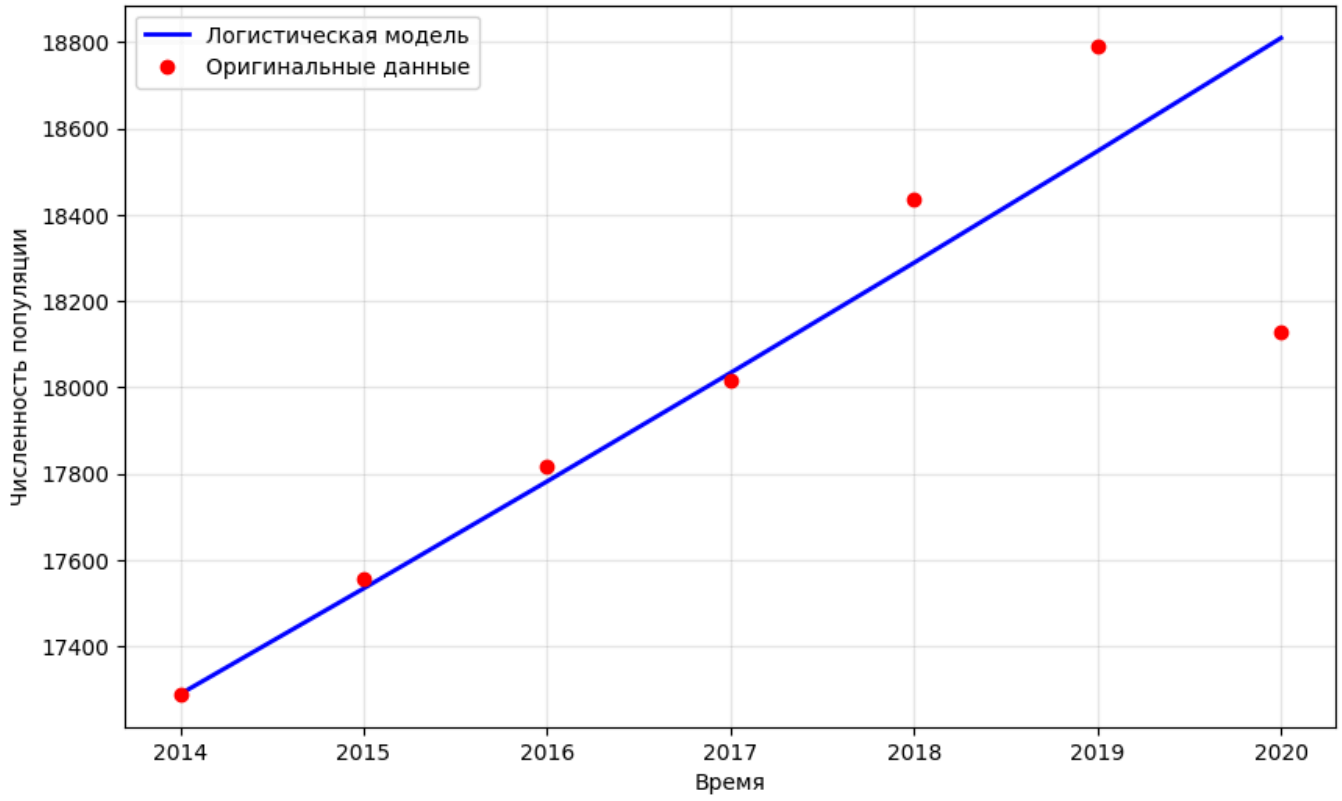
Логистическая модель

Для логистической модели была выбрана популяция Казахстана
В качестве плато предположил, что численность будет увеличиваться до 1 миллиарда (с учётом быстрого роста Казахстана)
Начальные данные:

year	persons	birth_rate	death_rate	natural_rate	r
2014	17289	23.1	7.6	15.5	-
2015	17557	22.7	7.5	16.2	0.015574393036903401

year	persons	birth_rate	death_rate	natural_rate	r
2016	17818	22.52	7.4	15.6	0.015258992628745548
2017	18014	21.7	7.2	14.4	0.013866198044848368
2018	18437	21.8	7.1	14.7	0.01627807247931027
2019	18789	21.8	7.2	14.6	0.01685538472782931
2020	18129	22.8	8.7	14.1	0.008007451027329821

$r_{mean} = 0.014306748657494 45$



# year	# persons
2014	17289.0
2015	17533.759091723263
2016	17781.920520765554
2017	18033.52979552073
2018	18288.63298218224
2019	18547.276710113623
2020	18809.50817722645

В 2020-м году в Казахстане произошёл довольно сильный отток населения, который был позже компенсирован в 2021-м. Так что модель прогнозирует вполне приемлемо.

Заключение

В результате проведенного исследования были построены две фундаментальные модели динамики популяций: модель Мальтуса и логистическая модель.
Было показано, что модель Мальтуса, описывающая неограниченный экспоненциальный рост, хорошо аппроксимирует данные на этапе активного развития популяции, что было подтверждено на примере населения Китая. Однако её прогноз оказывается завышенным в долгосрочной перспективе, так как модель не учитывает ограниченность ресурсов, как было показано на том же Китае, когда он вышел на своё (уже 3-х летнее) плато.
В свою очередь, логистическая модель учитывает ограничения среды, что позволяет строить более реалистичные прогнозы, стремящиеся к определенной ёмкости.

В рамках аналитического исследования были получены общие решения, определены точки равновесия и исследованы на устойчивость. Для логистической модели было проанализировано поведение решения.

Практическим итогом работы стала успешная реализация алгоритмов для подбора параметров моделей и построения прогнозов на основе реальных данных.

Использованные данные

1. https://ru.wikipedia.org/wiki/Население_Китая#Численность_населения_КНР_с_1949_по_2024_год
2. https://ru.wikipedia.org/wiki/Население_Казахстана#Демографические_данные

Листинг кода

```
# Imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Global variables

#Переменные для вычисления (шаг и т. д.)
time_step = 1  # Шаг в годах
additional_prognosis_years = 0


def get_time_array(start: int, end: int, step: float) -> np.ndarray:
    return np.arange(start, end + step, step)
# Funcs
## Math
def malthus_model(t: float, P0: float, r: float) -> float:
    """Вычисляет популяцию по модели Мальтуса.
```

```
    Args:
        t (float): Время.
        P0 (float): Начальная численность.
        r (float): Коэффициент роста.

    Returns:
        float: Численность популяции в момент t.
    """
    return P0 * np.exp(r * t)


def logistic_model(t: float, P0: float, r: float, K: float) -> float:
    """Вычисляет популяцию по логистической модели.

    Args:
        t (float): Время.
        P0 (float): Начальная численность.
        r (float): Коэффициент роста
        K (float): Ёмкость среды.

    Returns:
        float: Численность популяции в момент t.
    """
    return (K * P0 * np.exp(r * t)) / (K - P0 + P0 * np.exp(r * t))


def malthus_calc_r(t: float, P0: float, P: float) -> float:
    """Калькуляция r по Мальтусу

    Args:
        t (float): Разница по времени
        P0 (float): Начальное значение
        P (float): Конечное значение

    Returns:
        float: r
    """
    return 1 / t * np.log(P / P0)


def logistic_calc_r(t: float, P0: float, P: float, K: float) -> float:
    """Калькуляция r по логистической

    Args:
        t (float): Разница по времени
        P0 (float): Начальное значение
```

```
        P (float): Конечное значение

    Returns:
        float: r
    """
    return 1 / t * np.log((np.array(P) * (-(np.array(P0) - K))) /
(np.array(P0) * (K - np.array(P))))
## Graphics
def plot_population(
    t: float,
    P: float,
    label_model: str,
    t_observed: float = None,
    P_observed: float = None,
) -> None:
    """Построение графика популяции

    Args:
        t (float): Время прогноза.
        P (float): Прогноз.
        label_model (str): Наименование модели
        t_observed (float, optional): Время наблюдений. Defaults to None.
        P_observed (float, optional): Наблюдения. Defaults to None.
    """
    plt.figure(figsize=(10, 6))
    plt.plot(t, P, "b-", linewidth=2, label=label_model)
    if t_observed is not None and P_observed is not None:
        plt.plot(
            t_observed, P_observed, "ro", markersize=6, label="Оригинальные
данные"
        )
    plt.xlabel("Время")
    plt.ylabel("Численность популяции")
    plt.legend()
    plt.grid(True, alpha=0.3)
    plt.show()
# China (Malthus)
## Data check

Последняя строка проверочная
df_china = pd.read_csv("../data/china.csv", index_col=0, dtype={"persons":
"int"})
china_person_multiplier = int(1e3)

df_china
r_china_malthus_list = malthus_calc_r(df_china.index[1:] - df_china.index[0],
```

```
[df_china.persons.iloc[0]]* (len(df_china) - 1), df_china.persons[1:].values)
r_china_malthus = np.mean(r_china_malthus_list)
plt.plot(r_china_malthus_list)
plt.show()
calc_idx_china = 0

china_timeline = df_china.index
china_population_timeline = df_china["persons"]

starting_year_china = china_timeline[calc_idx_china]
predict_idx_china = [i for i in range(1, len(df_china))]
interval_len_china = len(df_china) - 1

birth_rate_mean_china = df_china["birth_rate"].mean()
death_rate_mean_china = df_china["death_rate"].mean()
natural_rate_mean_china = df_china["natural_rate"].mean()
P0_china = df_china["persons"].iloc(0)[0]
r_china_malthus

## Calculation
### Malthus
t_china = get_time_array(
    starting_year_china - starting_year_china,
    df_china.index[predict_idx_china[-1]] - starting_year_china +
additional_prognosis_years,
    time_step,
)
P_malthus_china = malthus_model(
    t_china,
    P0_china,
    r_china_malthus,
)
P_malthus_china
df_malthus_china = pd.DataFrame({"year": t_china + starting_year_china,
"persons": P_malthus_china})
df_malthus_china.set_index("year", inplace=True)
df_malthus_china
china_population_timeline
plot_population(t_china + starting_year_china, P_malthus_china, "Модель
Мальтуса", china_timeline, china_population_timeline)
### Logistic
K_china = 1412360
r_china_logistic_list = logistic_calc_r(df_china.index[1:] -
df_china.index[0], [df_china.persons.iloc[0]]* (len(df_china) - 1),
df_china.persons[1:].values, K_china)
r_china_logistic = np.mean(r_china_logistic_list)
```

```
r_china_logistic_list
r_china_logistic
plt.plot(r_china_logistic_list)
plt.show()
t_china = get_time_array(
    starting_year_china - starting_year_china,
    df_china.index[predict_idx_china[-1]] - starting_year_china +
additional_prognosis_years,
    time_step,
)
P_logistic_china = logistic_model(
    t_china,
    P0_china,
    r_china_logistic,
    K_china,
)
P_logistic_china
df_logistic_china = pd.DataFrame(
    {"year": t_china + starting_year_china, "persons": P_logistic_china}
)
df_logistic_china.set_index("year", inplace=True)
df_logistic_china
plot_population(
    t_china + starting_year_china,
    P_logistic_china,
    "Логистическая модель",
    china_timeline,
    china_population_timeline,
)
# Kazahstan
## Data check
df_kaz = pd.read_csv("../data/kazahstan.csv", index_col=0, dtype={"persons":
"int"})
kaz_person_multiplier = int(1e3)

df_kaz
calc_idx_kaz = 0

kaz_timeline = df_kaz.index
kaz_population_timeline = df_kaz["persons"]

starting_year_kaz = kaz_timeline[calc_idx_kaz]
predict_idx_kaz = [i for i in range(1, len(df_kaz))]
interval_len_kaz = len(df_kaz) - 1

birth_rate_mean_kaz = df_kaz["birth_rate"].mean()
```

```
death_rate_mean_kaz = df_kaz["death_rate"].mean()
natural_rate_mean_kaz = df_kaz["natural_rate"].mean()
P0_kaz = df_kaz["persons"].iloc(0)[0]
# r_kaz = natural_rate_mean_kaz / 100
K_china = 1412360
r_kaz_logistic_list = logistic_calc_r(df_kaz.index[1:] - df_kaz.index[0],
[df_kaz.persons.iloc[0]]* (len(df_kaz) - 1), df_kaz.persons[1:].values,
K_china)
r_kaz_logistic = np.mean(r_kaz_logistic_list)
r_kaz_logistic_list
r_kaz_logistic
plt.plot(r_kaz_logistic_list)
plt.show()
## Calculation
### Logistic
t_kaz = get_time_array(
    starting_year_kaz - starting_year_kaz,
    df_kaz.index[predict_idx_kaz[-1]] - starting_year_kaz +
additional_prognosis_years,
    time_step,
)
P_logistic_kaz = logistic_model(
    t_kaz,
    P0_kaz,
    r_kaz_logistic,
    int(1e6),
)
P_logistic_kaz
df_logistic_kaz = pd.DataFrame({"year": t_kaz + starting_year_kaz, "persons":
P_logistic_kaz})
df_logistic_kaz.set_index("year", inplace=True)
df_logistic_kaz
plot_population(
    t_kaz + starting_year_kaz,
    P_logistic_kaz,
    "Логистическая модель",
    kaz_timeline,
    kaz_population_timeline,
)
```