ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет экономических наук

Образовательная программа «Экономика»

КУРСОВАЯ РАБОТА

«Прогнозирование макроэкономических рядов с использованием алгоритма SSA»

Выполнила:

Студентка группы БЭК161
Волкова Анастасия Эдуардовна

Научный руководитель:

старший преподаватель
департамента прикладной экономики
Демешев Борис Борисович

Москва 2018

1 Введение

Одним из сильных и перспективных методов анализа и прогнозирования временных рядов является метод SSA (Singular Spectrum Analysis), в России известный как «Гусеница». Кроме того, существуют работы об адаптации идей данного алгоритма для анализа временных рядов с пропущенными наблюдениями[1]. Несмотря на то, что базовые идеи алгоритма впервые появились в работах Дэвида Брумхеда (David Broomhed,[2]), сам метод SSA для одномерных рядов наиболее полно был описан в монографии Н.Э.Голяндиной и др. ([3]). Среди других исследований этих же ученых также можно найти работы, посвященные анализу многомерных временных рядов с использованием SSA [4] и выбору параметров для настройки работы алгоритма [5]. Кроме того, существует пособие Н.Голяндиной и А.Коробейникова по использованию данного алгоритма с помощью пакета RSSA в RStudio [6].

Одна из отличительных черт алгоритма – возможность проводить исследование структуры ряда (выявление трендовой, гармонических и шумовых составляющих) без предположения о его модели. Также, алгоритм неплохо работает на зашумленных рядах и рядах с выбросами. Зашумленные ряды метод гусеницы сглаживает и получается достаточно хороший прогноз. Кроме того, SSA достаточно быстро работает.

Данная статья может помочь:

- Изучить алгоритм SSA и наглядно представить его устройство на уровне, понятном новичкам в анализе данных (составить так называемый Guide для новичков).
- Проанализировать применение алгоритма SSA в прогнозировании макроэкономических рядов, выявить его сильные стороны.
- Подобрать оптимальное значение экзогенных параметров модели, которые подойдут для большинства рядов.
- А также планируется сравнить качество работы алгортима SSA с более простым алгоритмом, например: "Показатели завтра будут такими же как вчера чтобы в дальнейшем можно было сравнить SSA с другими сложными алгоритмами.

2 Базовый алгоритм SSA

Цель алгоритма SSA (Singular Spectrum Analysis) - разбить ряд на аддитивные составляющие, чтобы были различимы тренд, циклическая компонента и шум. Проще говоря, преобразовать исходный ряд с помощью математических инструментов так, чтобы было понятно какие из измерений случайны, а какие могут быть использованы для построения прогноза.
Алгоритм состоит из 4 основных этапов:

- Построение траекторной матрицы

- Сингулярное разложение
- Группировка
- Диагональное усреднение

Введем обозначения:

$$(f_i)_{i=1}^N - \text{исходный временной ряд}$$

$$N - \text{длина ряда}$$

$$f_i - \text{i-ая компонента ряда}$$

2.1 Шаг 1. Построение траекторной матрицы

Алгоритм начинается с шага преобразования исходных данных (временного ряда) в траекторную матрицу (даннная матрица имеет такое название, так как вдоль диагоналей, перпендикулярных главной, стоят одинаковые элементы).

Сначала выбираем длину ряда (или длину гусеницы). Длина гусеницы должна быть меньше длины исходного ряда. Данный параметр задается или подбирается методом проб и ошибок самим исследователем, так как единого правила выбора длины гусеницы не существует.

Чтобы построить траекторную матрицу длины L (пока для простоты будем считать, что нужный параметр L нам заранее известен), нужно в первой строке оставить $N - L + 1$ элементов исходного ряда, а далее, для построения каждой последующей строки, сдвигаем предыдущую на один элемент влево, добавляя по одному новому элементу в конец.

Можно провести аналогию с очередью в магазине: каждый первый человек из очереди уходит, а вновь пришедший встает в конец, и так L раз.
Получаем траекторную матрицу X размерности $L \times K$, где $K = N - L + 1$

$$X = \begin{pmatrix} f_1 & f_2 & \dots & f_{N-L+1} \\ f_2 & f_3 & \dots & f_{N-L+2} \\ \vdots & \vdots & \ddots & \vdots \\ f_L & f_{L+1} & \dots & f_N \end{pmatrix}$$

Пример 1.

Есть исходный ряд $f = (1, 2, 3, 4, 5, 6, 7, 8)$ из 8 элементов $N = 8$ при длине гусеницы $L = 4$.

Построим траекторную матрицу X:

$$X = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

На всех диагоналях, перпендикулярных главной, находятся одинаковые элементы. Теперь матрица построена - значит можно переходить ко второму шагу.

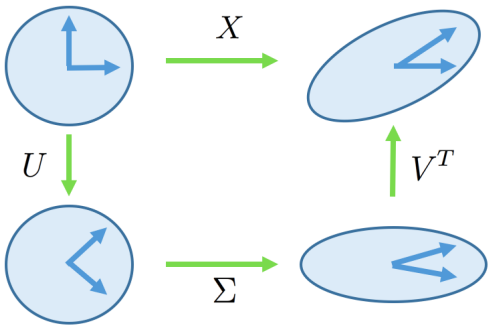
2.2 Шаг 2. Сингулярное разложение (SVD - разложение)

Так как мы хотим выделить тренд, шум и другие компоненты, нужно сначала разложить исходную матрицу на более простые. Поэтому цель данного шага - представить ранее полученную матрицу X в виде суммы элементарных матриц.

Это можно сделать с помощью собственных чисел матриц XX^T и X^TX. Идея SVD-метода заключается в том, что существуют ортогональные матрицы U и V и матрица Σ (по наполнению она напоминает диагональную матрицу, но необязательно является квадратной), через комбинацию которых можно выразить матрицу X.

$$X = U \times \Sigma \times V^T$$

Матрица X задает некоторые линейные преобразования (например, повороты или растяжения). Наша задача - представить данное сложное линейное преобразование в виде последовательных более простых. Так как матрицы U и V ортогональные, они задают некий поворот или отражение, то есть сначала мы как-то поворачиваем исходную матрицу, затем с помощью матрицы Σ ее растягиваем, а затем снова поворачиваем.



Для ясности разоберем данный алгоритм на простом примере, но сначала небольшое идейное отступление.

Небольшое отступление: „Почему именно собственные значения и векторы? “

На самом деле в этом и заключается основная идея алгоритма.

Собственные векторы — это направления, в которых матрица лишь растягивает или сжимает векторы, не меняя при этом их направления. Мы хотим максимально сохранить важные данные, которые содержатся в нашей матрице, а собственные векторы дают наиболее характерные направления движения и изменения ее векторов. Наиболее важными для нас будут векторы, имеющие наибольшие значения собственных векторов, остальные векторы с малыми значениями можно будет не учитывать, тем самым мы упростим себе вычислительную задачу.

Напоминание о том, как считаются собственные числа и векторы:

$$Ax = \lambda x, x \neq 0$$

x — собственный вектор

$\lambda -$ собственное значение

Далее разберем SVD - разложение **на примере**. Сначала рассмотрим простой пример SVD - разложения для произвольной матрицы, а затем продолжим преобразовывать нашу матрицу из примера 1 в рамках алгоритма SSA.

Пример 2.

Обычно данный этап не выполняется вручную, так как с этим прекрасно справляется программа, но мы рассмотрим последовательность действий, чтобы понимать ее смысл. Так как в данном случае нам интересен сам процесс преобразования, для простоты вычислений возьмем произвольную матрицу 4×2.

$$X = \begin{pmatrix} 1 & 2 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Нужно найти собственные значения и векторы матриц $W = XX^T$ и $S = X^TX$ Сначала с помощью умножения получим матрицы W и S

$$W = \begin{pmatrix} 1 & 2 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 0 & 0 \\ 2 & 3 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 5 & 7 & 0 & 0 \\ 7 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$S = \begin{pmatrix} 2 & 5 \\ 5 & 13 \end{pmatrix}$$

Теперь нужно найти собственные значения и векторы для матриц W и S

$$\begin{pmatrix} 5-\lambda & 7 & 0 & 0 \\ 7 & 10-\lambda & 0 & 0 \\ 0 & 0 & -\lambda & 0 \\ 0 & 0 & 0 & -\lambda \end{pmatrix} \times U = (W - \lambda I)U = 0$$

$$\begin{pmatrix} 2-\lambda & 5 \\ 5 & 13-\lambda \end{pmatrix} \times V = (S - \lambda I)V = 0$$

Решив характеристические уравнения, найдем собственные значения:

$$W : \lambda_1 = 0.0669 \quad \lambda_2 = 14.933 \quad \lambda_3 = 0 \quad \lambda_4 = 0$$

$$S : \lambda_1 = 0.0669 \quad \lambda_2 = 14.933$$

Теперь можем записать матрицы U, V и Σ, подставив собственные значения в характеристическое уравнение.

Стоит напомнить:

$$U, V - \text{ортогональные матрицы,}$$

$$\text{то есть } UU^T = I, \text{ где } I - \text{ единичная матрица}$$

$$\Sigma \text{ состоит из квадратных корней собственных чисел матриц } U \text{ и } V$$

Теперь мы можем представить матрицу X как комбинацию матриц U, V, Σ.

$$X = U_{4\times4} \times \Sigma_{4\times2} \times V^T_{2\times2}$$

$$\begin{pmatrix} 1 & 2 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} -0.576 & -0.817 & 0 & 0 \\ -0.817 & 0.576 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 3.864 & 0 \\ 0 & 0.259 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \times \begin{pmatrix} -0.360 & -0.933 \\ -0.933 & 0.361 \end{pmatrix}$$

Перед тем, как продолжить рассмотрение примера преобразования нашей матрицы из примера 1 в рамках алгоритма SSA, запишем алгоритм в общем виде:

Пусть $W = XX^T$, $S = X^TX$, тогда существуют ортогональные матрицы U и V и матрица Σ

$$X = U\Sigma V^T$$

Тогда $u_1, u_2, \ldots, u_L$ - собственные векторы матрицы W, а $v_1, v_2, \ldots, v_L$ - соответственно матрицы S, $\Sigma = \text{diag}(\sqrt{\lambda_1} \ldots \sqrt{\lambda_L})$

$$W = XX^T = (U\Sigma V)(U\Sigma V)^T = U\Sigma V V^T \Sigma^T U^T = U\Sigma^2 U^T$$
$$S = X^TX = (U\Sigma V)^T(U\Sigma V) = V^T\Sigma^T U^T U\Sigma V = V^T\Sigma^2 V$$

Собственные значения матриц S и W и располагаются в убывающем порядке.

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_L \geq 0$$

Тогда, используя векторы u_i , λ_i и матрицу X, можно выразить v_i

$$v_i = \frac{X^T u_i}{\sqrt{\lambda_i}}$$

Комбинация из показателей $(u_i, \sqrt{\lambda_i}, v_i)$ является SVD разложением матрицы X_i
Таким образом матрица X раскладывается в сумму элементарных матриц

$$X = \sum X_i = \sum_{i=1}^{L} \sqrt{\lambda_i} v_i^T u_i$$

Теперь реализуем SVD-разложение для матрицы X из примера 1.

$$X = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \\ 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

Сначала посчитаем матрицы $W = XX^T$ и $S = X^TX$:

$$W = \begin{pmatrix} 55 & 70 & 85 & 100 \\ 70 & 90 & 110 & 130 \\ 85 & 110 & 135 & 160 \\ 100 & 130 & 160 & 190 \end{pmatrix} \quad S = \begin{pmatrix} 30 & 40 & 50 & 60 & 70 \\ 40 & 54 & 68 & 82 & 96 \\ 50 & 68 & 86 & 104 & 122 \\ 60 & 82 & 104 & 126 & 148 \\ 70 & 96 & 122 & 148 & 174 \end{pmatrix}$$

Вычислим собственные значения $\sqrt{\lambda_i}$ и собственные векторы для матриц W и S и запишем их в матрицы U и V соответственно. Далее запишем матрицу X как произведение:

$$X = U \times \Sigma \times V^T$$

$$X = \begin{pmatrix} -0.271 & -0.725 & 0.627 & -0.063 & 0.055 \\ -0.352 & -0.419 & -0.694 & -0.451 & 0.121 \\ -0.432 & -0.114 & -0.244 & 0.852 & 0.122 \\ -0.513 & 0.192 & 0.063 & -0.098 & -0.829 \\ -0.593 & 0.498 & 0.248 & -0.239 & 0.530 \end{pmatrix} \times \begin{pmatrix} 26.8 \\ 1.86 \\ 1.203 \times 10^{-15} \\ 2.237 \times 10^{-16} \\ 5.819 \times 10^{-17} \end{pmatrix}$$

$$\times \begin{pmatrix} -0.271 & -0.352 & -0.432 & -0.513 & -0.593 \\ -0.725 & 0.419 & 0.114 & -0.192 & 0.498 \\ 0.322 & -0.651 & 0.024 & 0.613 & -0.309 \\ 0.541 & -0.389 & -0.292 & -0.408 & 0.550 \\ -0.059 & -0.352 & 0.845 & -0.395 & -0.037 \end{pmatrix}$$

Перемножив все векторы матриц U, V и $\sqrt{\lambda_i}$ по отдельности, получим матрицы X_i.

Например, чтобы получить матрицу X_1 нам нужно взять выделенные векторы и перемножить их в соответствии с формулой $X_i = \sqrt{\lambda_i} v_i^T u_i$.

$$X = \begin{pmatrix} -0.271 & -0.725 & 0.627 & -0.063 & 0.055 \\ -0.352 & -0.419 & -0.694 & -0.451 & 0.121 \\ -0.432 & -0.114 & -0.244 & 0.852 & 0.122 \\ -0.513 & 0.192 & 0.063 & -0.098 & -0.829 \\ -0.593 & 0.498 & 0.248 & -0.239 & 0.530 \end{pmatrix} \times \begin{pmatrix} 26.8 \\ 1.86 \\ 1.203 \times 10^{-15} \\ 2.237 \times 10^{-16} \\ 5.819 \times 10^{-17} \end{pmatrix}$$

$$\times \begin{pmatrix} -0.271 & -0.352 & -0.432 & -0.513 & -0.593 \\ -0.725 & 0.419 & 0.114 & -0.192 & 0.498 \\ 0.322 & -0.651 & 0.024 & 0.613 & -0.309 \\ 0.541 & -0.389 & -0.292 & -0.408 & 0.550 \\ -0.059 & -0.352 & 0.845 & -0.395 & -0.037 \end{pmatrix}$$

Тогда мы получим матрицу X_1.

$$X_1 = \begin{pmatrix} 1.821 & 2.353 & 2.885 & 3.417 \\ 2.481 & 3.207 & 3.932 & 4.658 \\ 3.142 & 4.061 & 4.980 & 5.899 \\ 3.803 & 4.915 & 6.027 & 7.139 \\ 4.465 & 5.769 & 7.075 & 8.380 \end{pmatrix}$$

Таким образом, мы разложили матрицу X в сумму элементарных матриц X_i.

2.3 Шаг 3. Группировка матриц

Для того, чтобы отделить тренд от остальных компонент, все матрицы X_i нужно разбить на несколько групп. К трендовой группе относят те составляющие, собственные векторы которых медленно меняются. Группировка заключается в группировке элементарных матриц X_i и суммировании матриц внутри каждой группы.

$$X = X_1 + X_2 + \ldots + X_L$$

где $X_i = U_i U_i^T X$ и есть m непересекающихся подмножеств I_1,$I_2, \ldots, I_m$

$$X = \sum_{i=1}^{m} Y_i,$$

где $Y_i = \sum_{k \in I_i} V_k$ - результирующая матрица каждого подмножества (сумма всех V_k, разделенных по группам, например, тренд, шум, периодика).

Значимость каждой элементарной матрицы V_k соответствует собственному числу λ_k. К тренду относятся векторы U_i, которые меняются относительно медленно, то есть компоненты с меньшими номерами вносят наибольший вклад, незначительные компоненты имеют большие номера.

Пример 3.

Обратимся к посчитанным в предыдущем примере собственным числам. Посчитаем вклад каждого собственного числа в значение их общей суммы в процентном выражении.

Тогда:

$\lambda_{1,\%} = 93.7\%$

$\lambda_{2,\%} = 6\%$

$\lambda_{3,\%} = 2.588 \times 10^{-17}\%$

$\lambda_{4,\%} = 3.989 \times 10^{-18}\%$

Заметим, что первое собственные число является наиболее значимым, значит его можно определить как тренд. Второе имеет гораздо меньший вес, а остальные практически не вносят вклад, так как их значения крайне малы. Для примера можем сгруппировать 1 и 2 матрицы и взять их в качестве тренда. Очевидно, что в силу простоты примера, выбирая данные компоненты, мы получим практически неизменный исходный ряд, так как соответствующие собственные числа дают в сумме почти 100%.

Итак, нам нужна сумма матриц X_1 и X_2.

$$\begin{pmatrix} 1.821 & 2.353 & 2.885 & 3.417 \\ 2.481 & 3.207 & 3.932 & 4.658 \\ 3.142 & 4.061 & 4.980 & 5.899 \\ 3.803 & 4.915 & 6.027 & 7.139 \\ 4.465 & 5.769 & 7.075 & 8.380 \end{pmatrix} + \begin{pmatrix} -0.820 & -0.352 & 0.115 & 0.582 \\ -0.481 & -0.207 & 0.067 & 0.342 \\ -0.142 & -0.061 & 0.019 & 0.101 \\ 0.196 & 0.084 & -0.027 & -0.139 \\ 0.535 & 0.230 & -0.075 & -0.380 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

Как и предполагалось, мы получили исходнью матрицу. Ничего страшного, доведем пример до конца, тем более, по факту мы все-таки отбросили часть информации, которая нам не необходима.

Теперь мы определили главные компоненты нашего ряда, но пока они записаны в виде матриц. Чтобы сделать прогноз, нужно преобразовать имеющиеся у нас данные обратно в ряды. Для этого переходим к финальному шагу алгоритма.

2.4 Шаг 4. Диагональное усреднение

Чтобы явно выделить тренд и использовать его для прогнозирования, нужно произвести процедуру, обратную разложению и свернуть данный тренд во временной ряд с помощью диагонального усреднения. На данном этапе каждая из матриц Y_i, полученных в предыдущем пункте будет преобразована в новый ряд длины N.

Идея похожа на алгоритм шага 1, только в обратном порядке. Для начала лучше представить, что мы имеем на данном этапе. После группировки образовалось несколько траекторных матриц, которые нам нужно свернуть в ряды (такие как тренд, шум и так далее). Воспользуемся диагональным усреднением по побочным диагоналям, то есть разделим сумму элементов побочной диагонали на их количество.

Предположим, что L^* в данной матрице - это $\min(L,K)$, а K^* соответственно $\max(L,K)$. Тогда нижний индекс при f обозначает в какой по счету элемент ряда, будет усредняться диагональ.

$$Y_i = \begin{pmatrix} f_1 & f_2 & \dots & \dots & f_{L^*} & \dots & f_{K^*} \\ f_2 & f_3 & \dots & f_{L^*} & \dots & f_{K^*} & f_{K^*+1} \\ \vdots & \vdots & f_{L^*} & \vdots & f_{K^*} & \vdots & \vdots \\ \vdots & f_{L^*} & \vdots & f_{K^*} & \vdots & \vdots & \vdots \\ f_{L^*} & \dots & f_{K^*} & \dots & \dots & \dots & f_N \end{pmatrix}$$

От f_1 до f_L^* каждый k-ый элемент встречается ровно k раз.
От f_L^* до f_K^* все элементы встречаются L^* раз, так как в матрице L^* строк.

От f_K^* до f_N чем больше номер элемента, тем меньше раз он встречается, а именно $N - k + 1$ раз.

Пример 4.

Например, если мы получили матрицу:

$$Y_i = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 1 \\ 2 & 3 & 4 & 5 & 6 \end{pmatrix}$$

Тогда ее усреднением будет новый ряд f:

$$f = (1, \frac{6+2}{2}, \frac{3+7+2}{3}, \frac{4+8+3}{3}, \frac{5+9+4}{3}, \frac{1+5}{2}, 6)$$

$$f = (1, 4, 4, 5, 6, 3, 6)$$

Пример 5.

Теперь вернемся к нашей разложенной матрице X из предыдущего примера и повторим те же действия для суммы матриц X_1 и X_2.

$$X_1 + X_2 = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

Усреднением получим новый реконструированный ряд f:

$$f = (1, 2, 3, 4, 5, 6, 7, 8)$$

Как уже отмечалось, выбранные компоненты содержали в себе почти всю информацию об исходной матрице, поэтому неудивительно, что в итоге мы идеально восстановили исследуемый ряд.

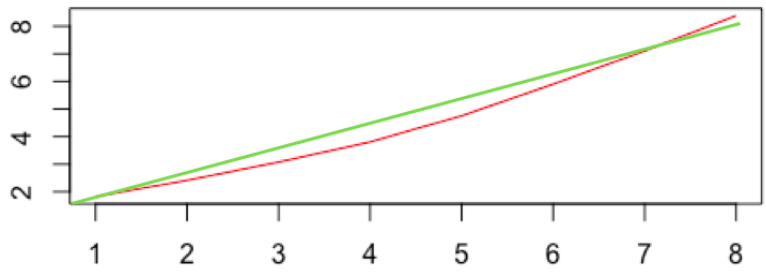
Смысл алгоритма - отбросить ненужную информацию, чтобы максимально сохранялся только тренд. Давайте посмотрим как изменится наш ответ, если мы возьмем только первый собственный вектор.

$$X_1 = \begin{pmatrix} 1.821 & 2.353 & 2.885 & 3.417 \\ 2.481 & 3.207 & 3.932 & 4.658 \\ 3.142 & 4.061 & 4.980 & 5.899 \\ 3.803 & 4.915 & 6.027 & 7.139 \\ 4.465 & 5.769 & 7.075 & 8.380 \end{pmatrix}$$

В результате усреднения получаем:

$$f = (1.821, 2.417, 3.078, 3.803, 4.755, 5.898, 7.107, 8.380)$$

Сравним полученный ряд с исходным:



В целом, заметно, что мы идем близко к тренду, но все-таки есть некоторые неточности, так как часть информации мы не учитывали при реконструкции. Наш исходный ряд был крайне простой, в нем не было сезонных колебаний и шума, поэтому в учебных целях, мы намеренно отбрасывали часть тренда. Для данного ряда даже первоклассник, обнаружив закономерность, сможет сделать прогноз. Однако алгоритм рассчитан на ряды с более сложной структурой, где помимо тренда будут присутствовать и другие компоненты.

В общем виде можно представить алгоритм так:

$$f_k^i = \begin{cases} \frac{1}{k}\sum_{i=1}^{k} \tilde{X}_{k,k-i+1}, 1 \le k < L; \\ \frac{1}{L}\sum_{i=1}^{L} \tilde{X}_{k,k-i+1}, L \le k \le K; \\ \frac{1}{N-k+1}\sum_{i=1}^{N-k+1} \tilde{X}_{i+k-K,K-i+1}, K < k \le N \end{cases}$$

Система возникает из-за того, что все элементы диагоналей матрицы встречаются в ней разное количество раз, и нужно понять, на какое число нужно делить сумму показателей для их усреднения.

Таким образом, после группировки и усреднения мы получаем новые временные ряды с явно выделенным трендом, цикличностью и другими составляющими.

2.5 Прогнозирование

После того как мы произвели декомпозицию и реконструкцию ряда, можно приступить к прогнозированию показателей на следующие периоды.
Для этого сначала необходимо ввести несколько новых обозначений:
Пусть $\upsilon^2 = \pi_1^2 + \pi_2^2 + \ldots + \pi_L^2$, где π_i - это последняя компонента собственного вектора $U_i (i = 1, \ldots, L)$.
Известно, что $\upsilon^2 < 1$
Определим вектор $A = (\alpha_1, \ldots, \alpha_{L-1})$:

$$A = \frac{1}{1 - \upsilon_2} \sum_{i=1}^{L} \pi_i U_i$$

Может быть доказано, что последнияя компонента x_L любого вектора $X = (x_1, \ldots, x_L)^T$ это линейная комбинация первых компонент $(x_1, \ldots, x_{L-1})$

$$x_L = \alpha_1 x_{L-1} + \ldots + \alpha_{L-1} x_1$$

Переходим к самой процедуре прогнозирования. Определим временной ряд $X_{N+h} = (x_1, \ldots, x_{N+h})$ по формуле:

$$x_i = \begin{cases} \tilde{x_i}, \text{ для } i = 1, \ldots, N \\ \sum_{j=1}^{L-1} \alpha_j x_{i-j} \text{ для } i = N+1, \ldots, N+h \end{cases}$$

То есть от 1 до N это данные, которые мы имеем, а после N, все то, что не знаем, но хотим спрогнозировать.

3 Использование алгоритма SSA на практике

3.1 Знакомство с RSSA-package.

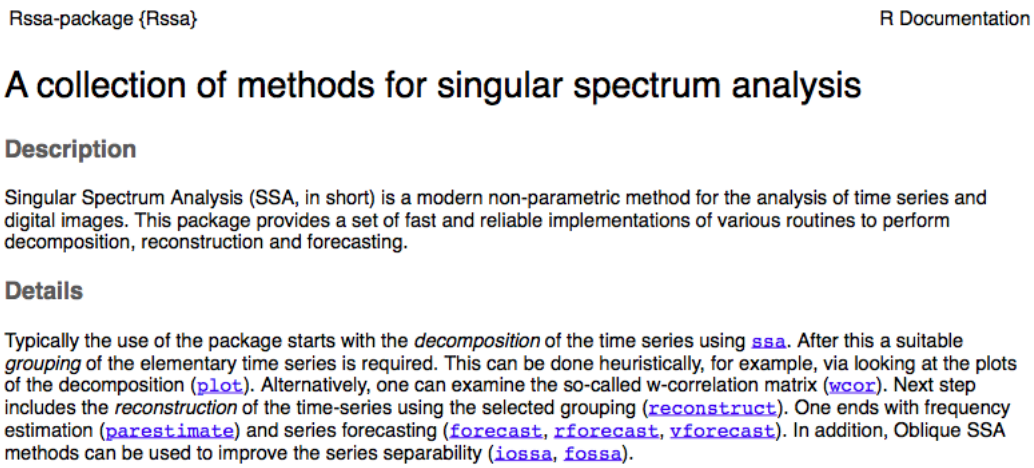
Теперь перейдем к практической части. Попробуем реализовать алгоритм SSA и посмотрим, как он работает на реальных данных. Для этого будем использовать свободную программную среду R, поскольку там уже существует нужная библиотека RSSA-package, с помощью которой мы будем вызывать те или иные команды алгоритма.

Чтобы использовать данный пакет, его нужно подключить из Repository Cran (в инструментах).

Далее, посмотрим, как устроена данная библитека:

```
library("Rssa")
help("Rssa")
```

Появляется документация, с которой можно кратко ознакомиться.



Rssa-package {Rssa} R Documentation

A collection of methods for singular spectrum analysis

Description

Singular Spectrum Analysis (SSA, in short) is a modern non-parametric method for the analysis of time series and digital images. This package provides a set of fast and reliable implementations of various routines to perform decomposition, reconstruction and forecasting.

Details

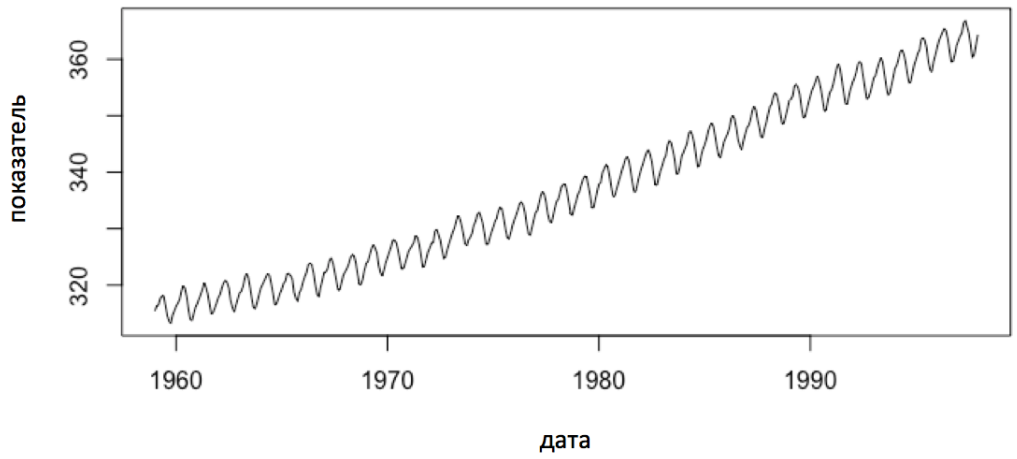
Typically the use of the package starts with the *decomposition* of the time series using ssa. After this a suitable *grouping* of the elementary time series is required. This can be done heuristically, for example, via looking at the plots of the decomposition (plot). Alternatively, one can examine the so-called w-correlation matrix (wcor). Next step includes the *reconstruction* of the time-series using the selected grouping (reconstruct). One ends with frequency estimation (parestimate) and series forecasting (forecast, rforecast, vforecast). In addition, Oblique SSA methods can be used to improve the series separability (iossa, fossa).

Ближе к концу документации приводится пример использования библиотеки на стандартном временном ряду co2, с форматом которого так же можно ознакомиться с помощью help.

3.2 Анализ простого ряда.

Возьмем встроенный временной ряд co2, содержащий 468 ежемесячных наблюдений уровня концентрации углекислого газа близ вулкана Мауна Лоа с 1959 по 1997 год. Изобразим данный ряд на графике:

```
plot(co2)
```



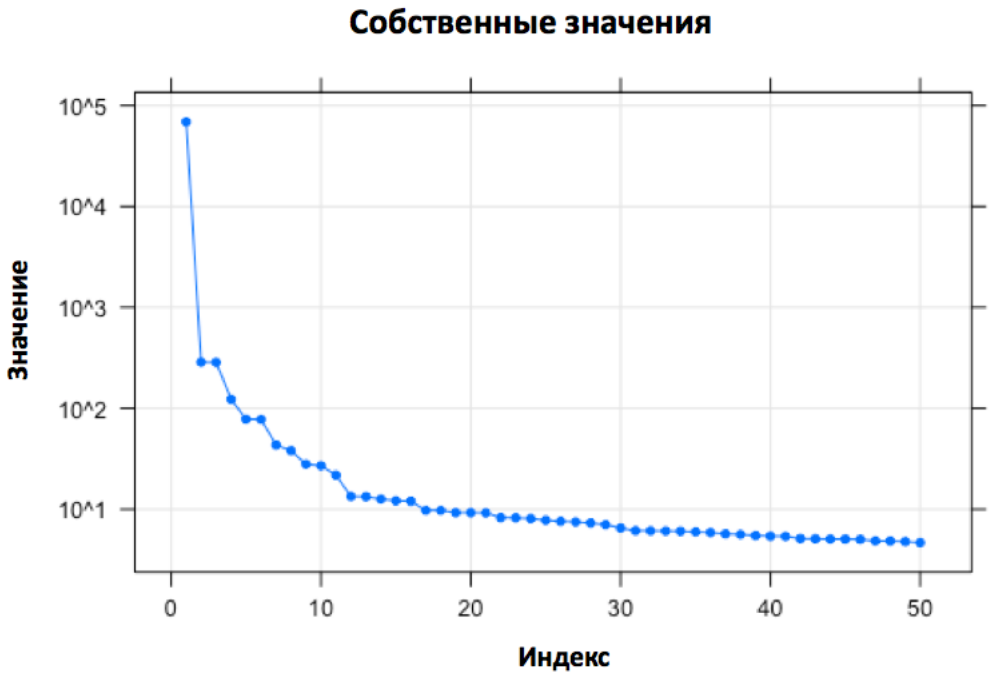
Шаг 1 - декомпозиция. На данном этапе применим к нашему ряду co2 встроенную функцию ssa, которая выполняет построение траекторной матрицы и SVD-разложение и выдает нам результат (его можно посмотреть в summary). Значение длины гусеницы L для начала берем произвольное, но не больше, чем половина длины исходного ряда, например 120.

```
s <- ssa(co2, L = 120)
summary(s)
```

```
Call:
ssa(x = co2, L = 120)

Series length: 468,     Window length: 120,     SVD method: eigen
Special triples:  0

Computed:
Eigenvalues: 50,        Eigenvectors: 50,       Factor vectors: 0

Precached: 0 elementary series (0 MiB)
```

Теперь можно посмотреть собственные значения и векторы, которые посчитал алгоритм.

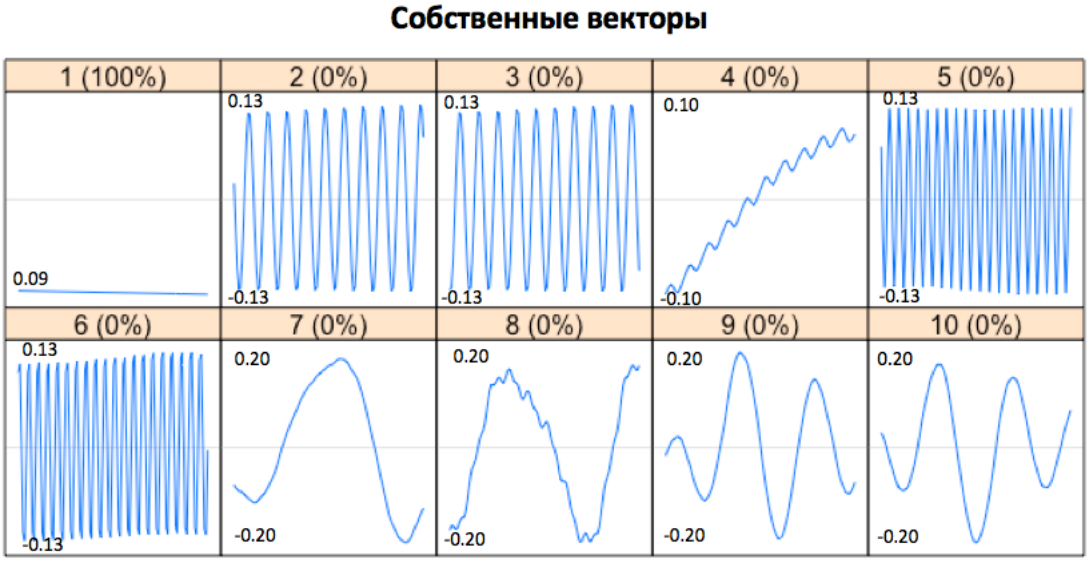
```
plot(s)
```



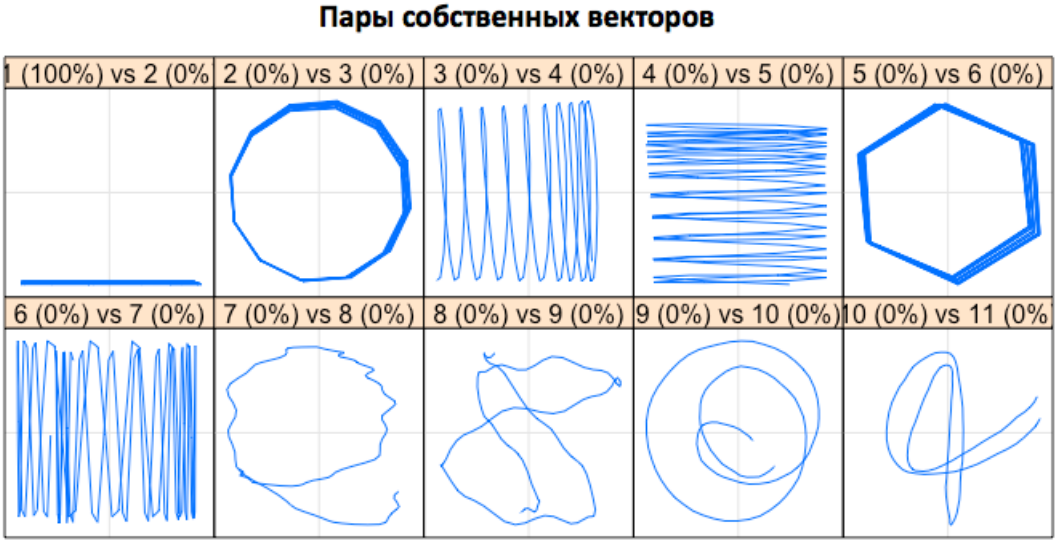
Первых десяти будет вполне достаточно, так как на графике собственных значений видно, что каждый последующий вектор играет все меньшую роль в формировании ряда.

Посмотрим на графики выбранных нами векторов:

```
plot(s, type = "vectors",)
```



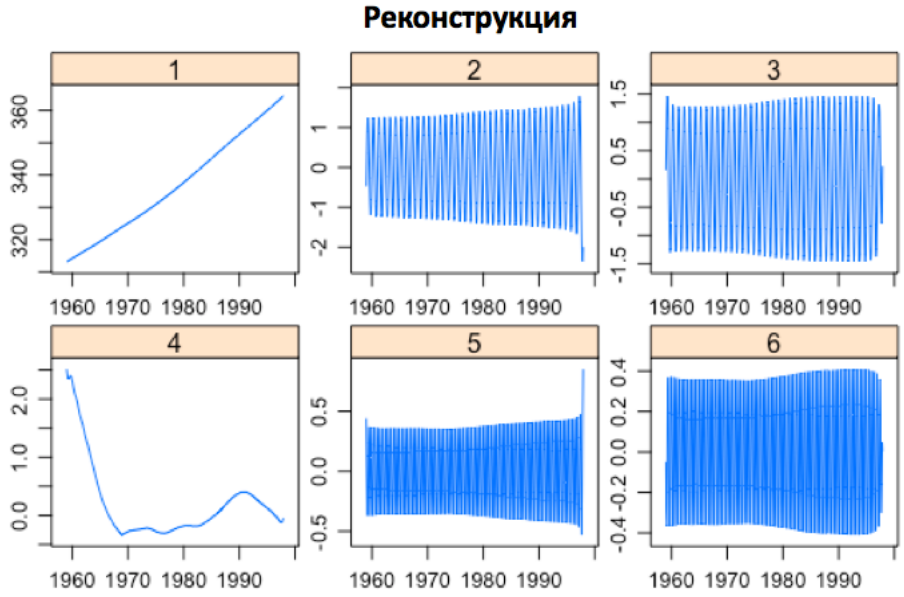
На первый взгляд можно заметить, что вектора 2 и 3, 5 и 6 имеют попарно достаточно похожую структуру. Чтобы сгруппировать векторы по парам для дальнейшего исследования, построим их парные графики.



Действительно, четкие фигуры с явным периодом присутствуют у пар векторов 2 и 3, 5 и 6, поэтому остановим свой выбор на первых 6 векторах.

Реконструируем ряд отдельно по каждому выбранному собственному вектору, чтобы было видно, как нам их потом группировать.

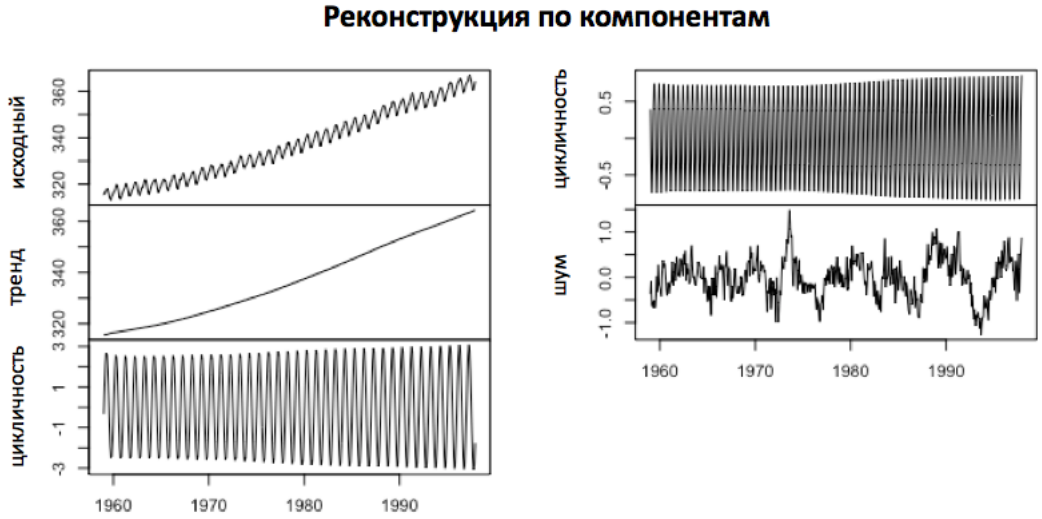
```
plot(s, type = "series", groups = as.list(1:6))
```



Теперь необходимо сгруппировать выбранные векторы: в качестве тренда возьмем 1 и 4 вектора, в качестве сезонности и шума (2 и 3), (5 и 6), так как видно, что они имеют похожую структуру.

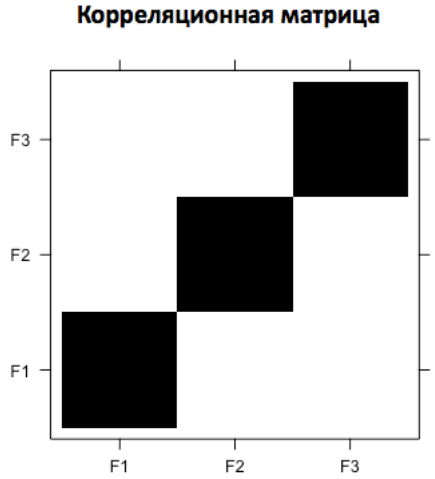
Перейдем к шагу реконструкции ряда в соответствии с новыми группами векторов.

```
recon <- reconstruct(s, groups = list(c(1,4), c(2,3), c(5,6)))
res <- residuals(recon)
plot(recon)
```



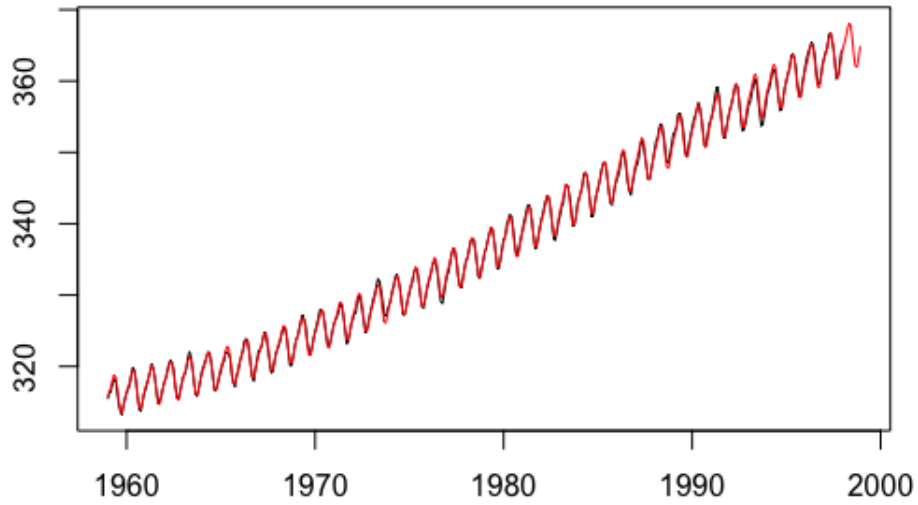
Построим корреляционную матрицу пар векторов, чтобы убедиться, что они не коррелируют между собой.

```
plot(wcor(s, groups = list(c(1,4), c(2,3), c(5,6))))
```



Построим прогноз на 12 месяцев вперед, используя векторный способ прогнозирования.

```
vector <- vforecast(s, groups = list(1:6), len = 12, only.new = FALSE)
plot(cbind(co2, vector), plot.type="single", col=c("black","red"))
```



Таким образом, мы выполнили нашу задачу, а именно: разложили исходный ряд, выбрали компоненты, по которым произвели реконстукцию и сделали годовой прогноз.

3.3 Настройка параметров для более сложных рядов.

Если ряд имеет сложную структуру, то сначала его рекомендуется сгладить, выбрав маленькую длину окна, взять главную компоненту и построить по ней тренд. Далее, если удастся обнаружить периодику, то стоит выбрать длину кратную периоду (период можно разглядеть на графиках пар векторов, выделив Т-угольники - синусоиды периода Т).

Теперь попробуем реализовать алгоритм SSA на реальных данных. Возьмем датасет M4, который используется в соревнованиях по прогнозированию. Для начала посмотрим описание данного датасета. Чтобы использовать данный пакет, его нужно подключить из Repository Cran (в инструментах). Посмотрим, как устроена данная библитека

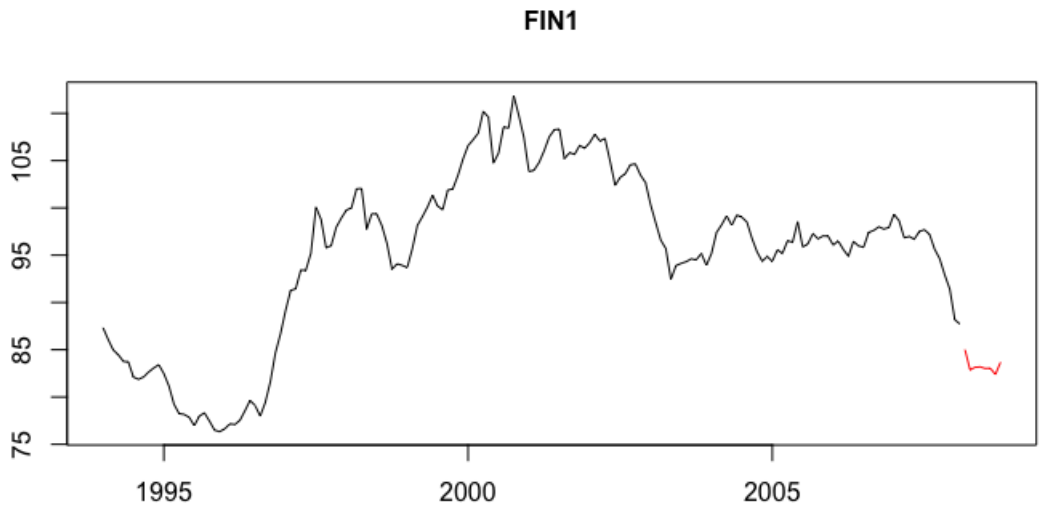
```
library(M4comp)
data <- M4
help(M4)
print(M4)
```

	Period					
Type	YEARLY	BIANNUALLY	QUARTERLY	MONTHLY	WEEKLY	DAILY
BUSINESS-INDUSTRY	6	0	28	980	486	0
CLIMATE	0	0	0	0	0	1000
DEMOGRAPHICS	999	0	0	1	0	0
ECONOMICS	735	13	374	792	71	15
FINANCE	0	0	0	477	776	747
INTERNET-TELECOM	500	0	0	0	500	0
INVENTORY	0	0	0	1500	0	0

В данном датасете содержится большое количество рядов с различной частотой сбора данных.

Попробуем проанализировать какой-нибудь ряд, подобрав нужные параметры для алгоритма вручную, пользуясь теми правилами, которые были упомянуты выше. Например, возьмем и изобразим первый ряд датасета M4, который называется FIN1.

```
summary(M4[[1]])
plot(M4[[1]])
```



Запишем в переменную data1 большую часть ряда, оставив остальную часть для оценки качества будущего прогноза.

```
data1 = ts(data[[1]][["past"]])
```

Сгладим ряд, так как он имеет сложную структуру. Сглаживание ряда происходит при выборе небольшой длины гусеницы. Например, возьмем длину $L = 12$.

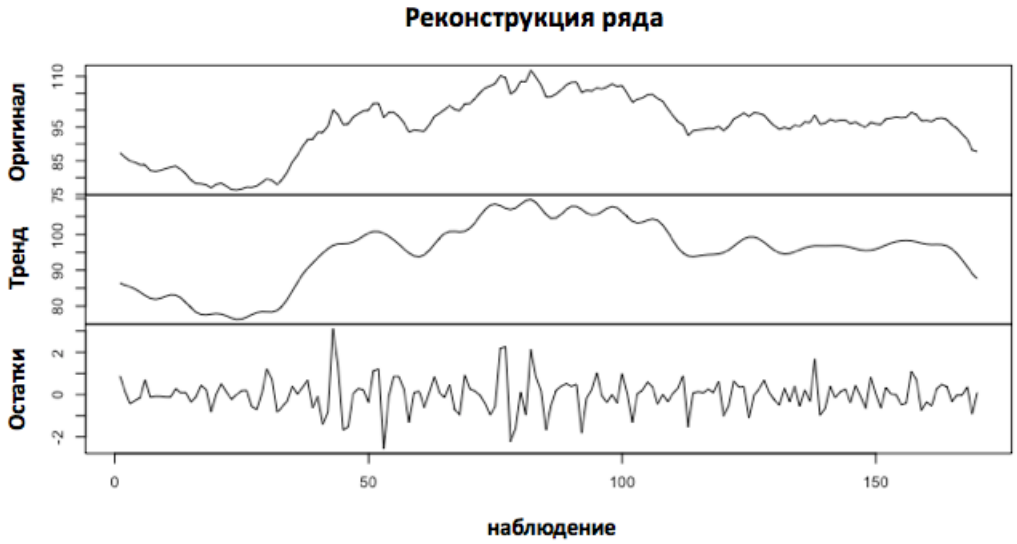
```
s1 <- ssa(data1, L = 12)
```

Произведем реконструкцию по первому вектору, запишем результат в качестве тренда и изобразим сглаженный тренд поверх исходного ряда.

```
res1 <- reconstruct(s1, groups = list(1))
trend <- res1$F1
plot(res1, add.residuals = FALSE, plot.type = "single",
col = c("black", "red"), lwd = c(1, 2))
```

Посмотрим, что осталось от ряда после выделения тренда. Запишем остатки в переменную res.trend.

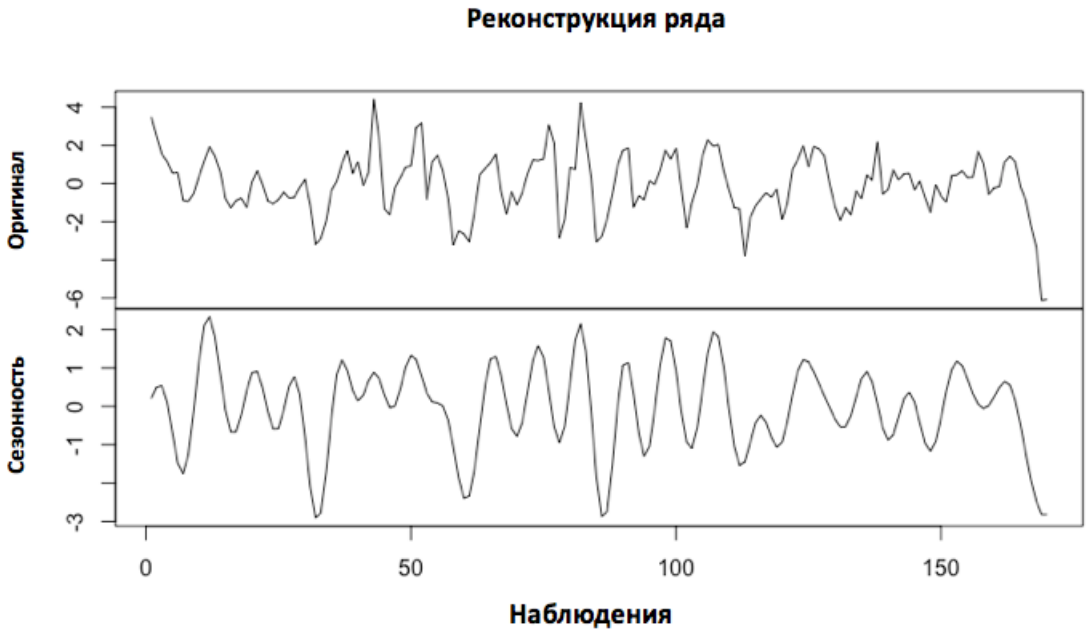
```
res.trend <- residuals(res1)
```



Теперь займемся поиском сезонной компоненты. Для этого берем все, что осталось после выделения тренда и производим алгоритм для этих данных, выбрав длину ряда побольше, чем в первый раз. Например, выберем $L = N/2 = 85$

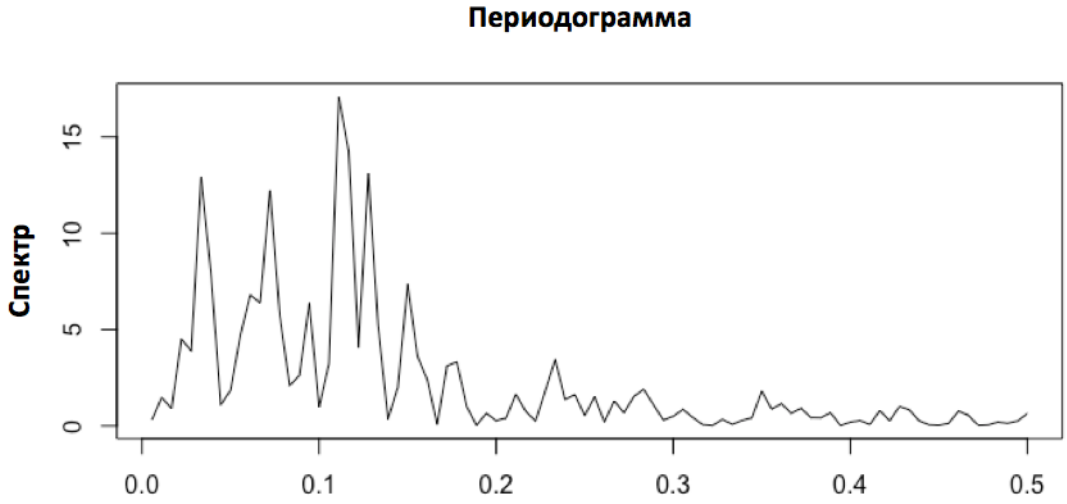
```
res.trend <- residuals(res1)
s2 <- ssa(res.trend, L = 85)
```

```
res2 <- reconstruct(s2, groups=list(1:9))
seasonality <- res2$F1
resid <- residuals(res2)
plot(res2, add.residuals = FALSE)
```



Найдем периодику сезонности. Она понадобится нам в дальнейшем для выбора оптимальных параметров для данного ряда.

```
spec.pgram(res.trend, detrend = FALSE, log = "no")
```



Из графика можно заметить, что присутствует сезонность с периодами примерно 9, 4.5, и т.д. Таким образом выберем длину, кратную 9, например 36.

Теперь посмотрим на пары векторов, чтобы определить кол-во векторов для прогнозирования.

Несмотря на то, что фигуры нечёткие, можно примерно различить 9-угольник в комбинации 8-го и 9-го векторов. Таким образом, выберем вектора с 1 по 9. Теперь реализуем алгоритм SSA для длины ряда $L = 18$ (тоже число кратное 9), сделаем прогноз на 6 измерений и проверим его качество с помощью метрики MAPE (Mean Absolute Percentage Error).

```
s2 <- ssa(data1, L = 18)
forecast <- vforecast(s2, groups = list(1:9), len = 6)
error <- accuracy(forecast, M4[[1]]$future[1:6])
error
```

Получаем значение MAPE $= 0.899\%$ Это достатчно хороший результат, значит значения подобраны хорошо. Попробуем рассмотреть ближайшие значения параметров.

При векторах 1-9: $L = 17$ MAPE $= 1.84$, $L = 19$ MAPE $= 1.27$

При векторах 1-10: $L = 17$ MAPE $= 1.12$, $L = 19$ MAPE $= 0.98$

Таким образом, мы еще раз убеждаемся, что изначально мы подобрали параметры качественно.

4 Подбор универсальных параметров

Чтобы проверить есть ли некая зависимость точности прогнозирования от длины окна и количества выбранных компонентов в общем случае, проанализируем изменения качества алгоритма при разных параметрах для нескольких рядов. Ранее мы тщательно подходили к подбору параметров для каждого ряда, учитывая его особенности. Поэтому будет разумно предположить, что в общем случае будет достаточно сложно подобрать универсальные параметры для моделей.

Для исследования возьмем первые 10 рядов из уже использовавшегося датасета M4. Предварительно ознакомившись со структурой рядов (например, визуализировав их), можно заметить, что они совершенно разные.

Сначала попробуем протестировать несколько вариантов длин гусеницы при фиксированном количестве векторов для всех рядов. Способ измерения ошибки остается тот же: MAPE. Например, для $L = 120$ код выглядит так:

```
data <- M4
all_results <- list()
MAPE <- list()

for (series_no in 1:10) {
  y <- M4[[series_no]]
  y_past <- y$past
  res_ssa_1 <- ssa(y_past, L = 120)
  forecast <- vforecast(res_ssa_1 , groups = list(1:10), len = 6)
  error <- accuracy(forecast, M4[[series_no]]$future[1:6])
  all_results[[series_no]] <- error
  MAPE[[series_no]] <- all_results[[series_no]][5]
}
```

Результаты расчетов средней абсолютной ошибки в процентном выражении можно представить в виде таблице, где есть все точные значения.

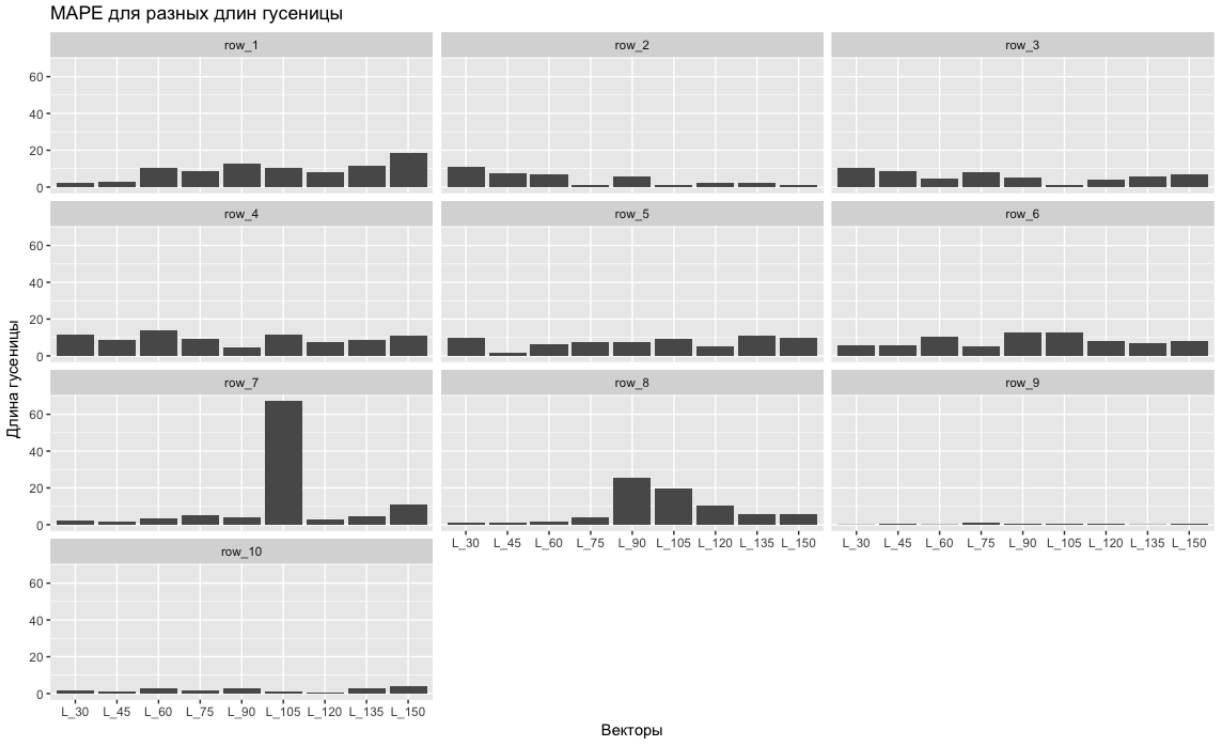
Таблица 1: MAPE для различных длин гусеницы

	length	row_1	row_2	row_3	row_4	row_5	row_6	row_7	row_8	row_9
1	L_30	2.41	10.80	10.22	11.46	9.63	6.00	2.54	0.84	0.16
2	L_45	3.08	7.42	8.54	8.75	1.94	6.01	1.74	0.98	0.28
3	L_60	10.14	6.68	4.64	13.77	6.41	10.53	3.30	1.71	0.22
4	L_75	8.39	1.09	8.28	9.41	7.62	5.46	5.11	3.94	0.89
5	L_90	12.65	5.50	5.10	4.57	7.63	12.72	3.81	25.33	0.36
6	L_105	10.58	0.98	1.04	11.78	9.01	12.96	67.40	19.98	0.43
7	L_120	8.15	2.03	3.75	7.73	5.16	8.29	2.64	10.31	0.27
8	L_135	11.39	2.46	5.54	8.69	10.72	6.87	4.39	5.69	0.19
9	L_150	18.57	1.32	7.09	10.92	9.65	7.83	11.11	5.61	0.57

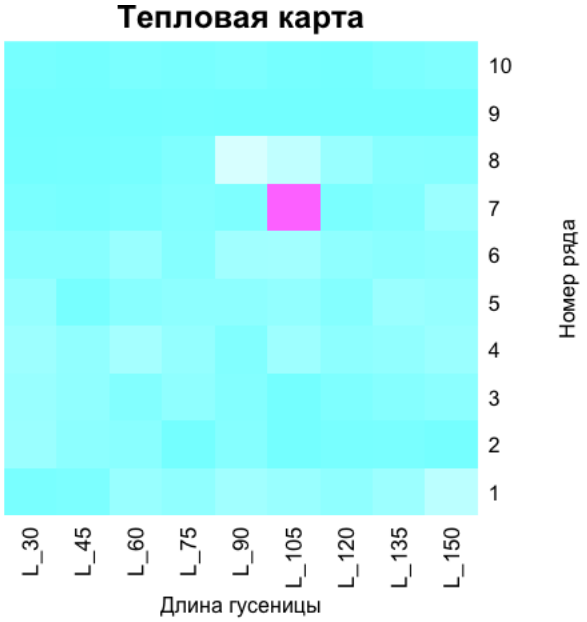
Однако, чтобы сделать какие-то выводы, визуализируем результаты.

Попробуем рассмотреть зависимость качества прогноза от длины ряда с помощью диаграмм. На графиках для каждого ряда по оси x представлены значения

исследуемых параметров, а по оси y - значение ошибки в процентном выражении. Чем выше столбик, тем хуже прогноз.

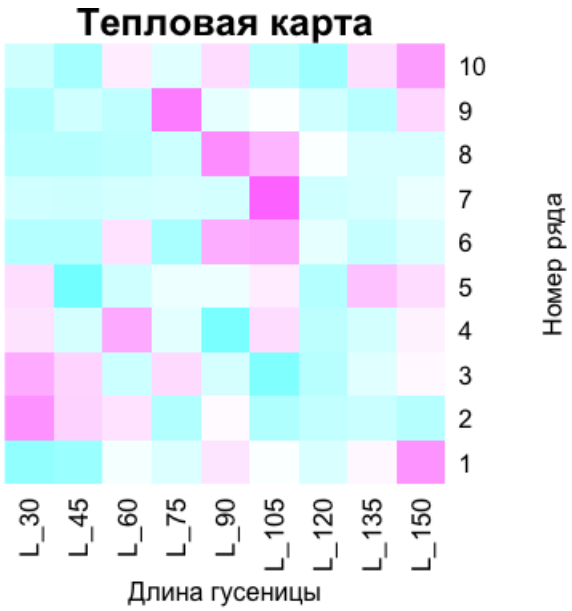


Теперь представим эти же данные на тепловых картах. На данном графике голубой цвет символизирует меньшую ошибку, а фиолетовый соответственно большую.



Еще по результатам в таблице, можно заметить, что какие-то ряды прогнозируются лучше, а какие-то хуже. То есть из-за эффекта масштаба, трудно заметить,

какая именно длина гусеницы дает меньшую ошибку для каждого рассматриваемого ряда. Поскольку для каждого ряда ошибка лежит в разумных пределах, попробуем для каждого ряда вычесть из спрогнозированного значения среднее по ряду и поделить на стандартное отклонение. Возможно, тогда мы увидим некоторую зависимость.

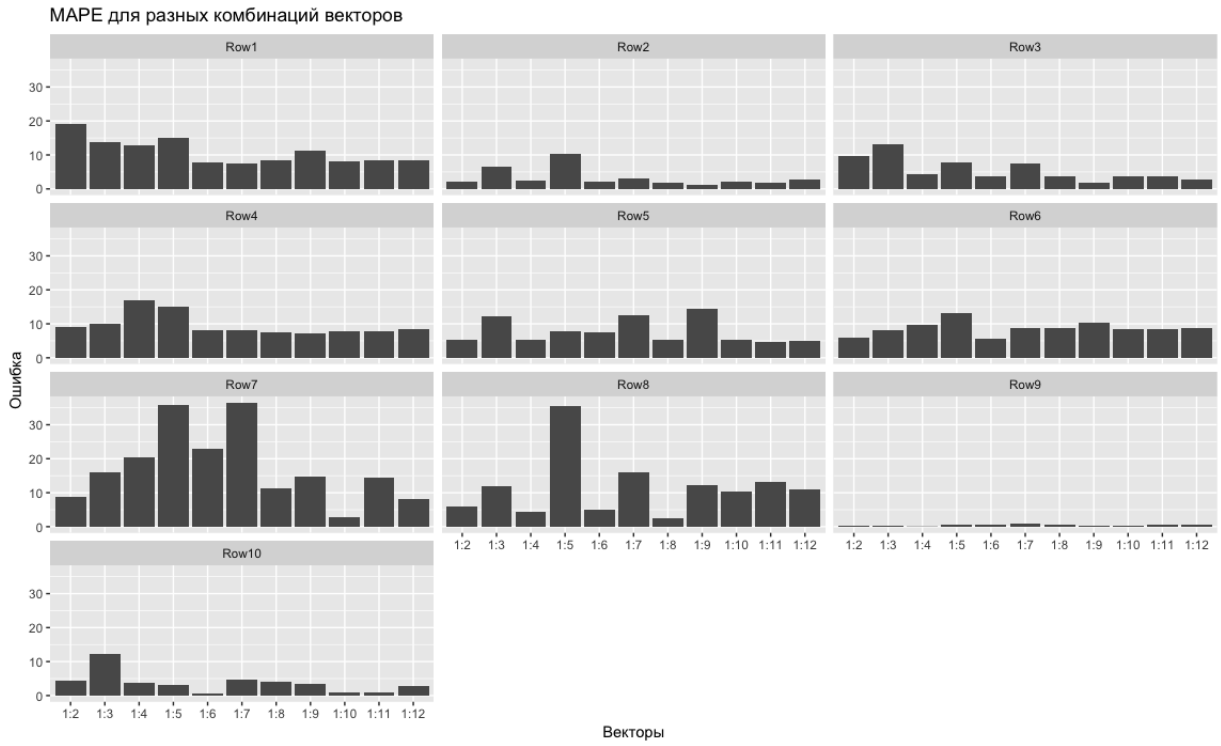


Нельзя сделать однозначные выводы какое значение лучше. Что хорошо подходит одним рядам, совершенно не подходит другим. Хотя можно заметить, что $L = 120$ в среднем дает неплохой прогноз для всех рядов (качество не самое лучшее, но для всех 10 рядов достаточно однородно).

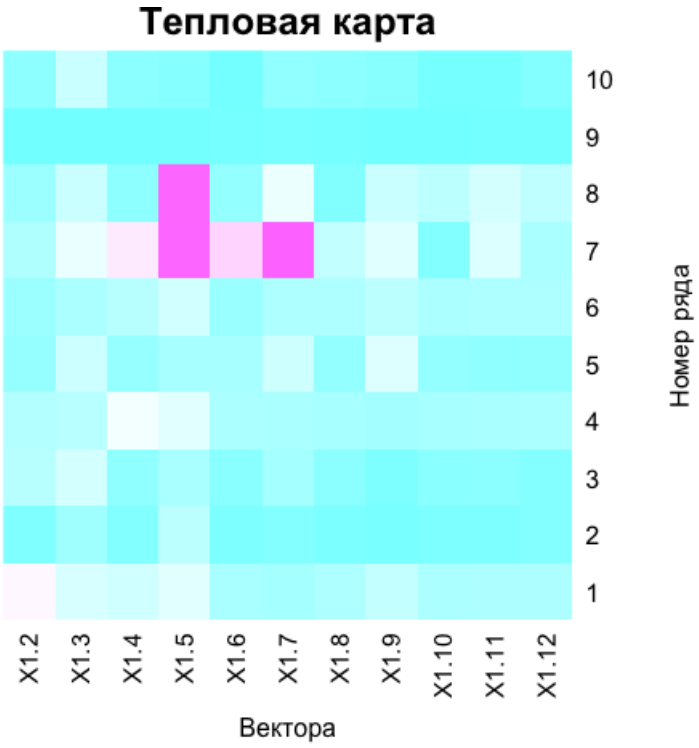
Теперь протестируем разное количество векторов, используемых для прогнозирования. Длину окна возьмем $L = 120$. Количество векторов обозначено как $1 : N$ (то есть берем первые N векторов).

Таблица 2: MAPE для различных комбинаций векторов

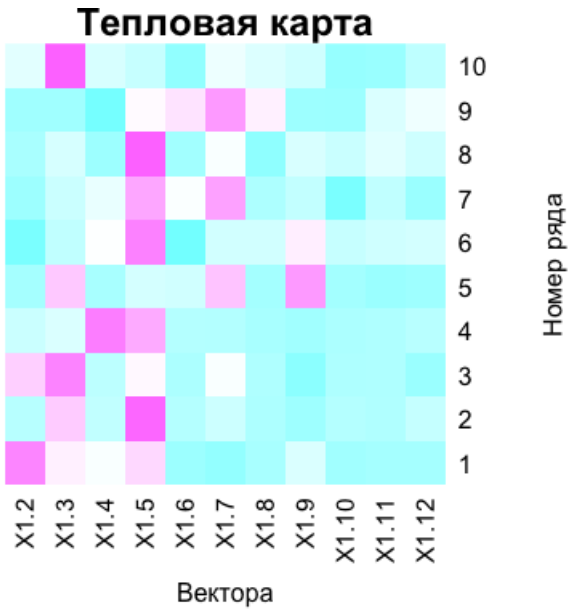
	vectors	Row1	Row2	Row3	Row4	Row5	Row6	Row7	Row8	Row9	Row10
1	1:2	19.11	2.13	9.69	9.18	5.26	6.00	8.68	6.01	0.29	4.29
2	1:3	13.83	6.57	13.12	9.86	12.27	8.13	15.94	11.95	0.29	12.11
3	1:4	12.87	2.54	4.43	16.92	5.41	9.77	20.50	4.37	0.11	3.80
4	1:5	14.96	10.20	7.87	14.93	7.69	13.03	35.68	35.57	0.66	3.05
5	1:6	7.85	2.04	3.67	8.11	7.55	5.68	22.90	5.08	0.73	0.52
6	1:7	7.34	2.94	7.35	8.05	12.47	8.62	36.52	16.13	0.99	4.73
7	1:8	8.50	1.77	3.81	7.51	5.22	8.61	11.18	2.41	0.69	4.04
8	1:9	11.25	1.14	1.89	7.18	14.47	10.23	14.69	12.19	0.28	3.44
9	1:10	8.15	2.03	3.75	7.73	5.16	8.29	2.64	10.31	0.27	0.79
10	1:11	8.39	1.90	3.83	7.88	4.64	8.55	14.30	13.25	0.51	0.90
11	1:12	8.42	2.70	2.73	8.34	4.89	8.63	8.00	10.84	0.59	2.67



Теперь представим эти же данные на тепловых картах. На данном графике голубой цвет символизирует меньшую ошибку, а фиолетовый соответственно большую.



Теперь пронормируем значения по рядам для лучшей визуализации.



Если делать вывод по такой же логике как с длиной окна, то в данном случае более менее подходят вектора с 1 по 10.

Так как ранее мы вручную подбирали параметры для ряда M4[[1]], можно сравнить качество прогнозов. Совсем не удивительно, что параметры, подобранные вручную, дают наилучший прогноз. Ранее мы брали длину ряда 36 и получали MAPE = 0.899. Здесь же, при $L = 30$: MAPE = 2.414, а при $L = 45$: MAPE = 3.085, то есть нет прямой зависимости, поэтому подбор универсальных параметров весьма ненадежен.

Подводя итог, можно сделать вывод, что для алгоритма SSA лучше настраивать параметры вручную. Данный алгоритм дает неплохой прогноз, но для этого необходим тщательный подбор параметров модели, исходя из особенностей исследуемого ряда.

Список литературы

[1] Осипов Е.В. Голяндина, Н.Э. Метод "Гусеницаssa для анализа временных рядов с пропусками. *Мат. модели. Теория и приложения*, 2005.

[2] David S Broomhead and Gregory P King. Extracting qualitative dynamics from experimental data. *Physica D: Nonlinear Phenomena*, 1986.

[3] Н. Э. Голяндина. Метод «Гусеница»-ssa: прогноз временных рядов. *Издательский Центр «Академия»*, 2004.

[4] Голяндина Н.Э Степанов, Д.В. Варианты метода «Гусеница»-ssa для прогноза многомерных временных рядов. In *Труды IV Международной конференции «Идентификация систем и задачи управления» SICPRO'05. Москва, 25-28 января*, page 1831, 2005.

[5] Голяндина Н. Александров, Ф. Выбор параметров при автоматическом выделении трендовых и периодических составляющих временного ряда в рамках подхода «Гусеница»-SSA. In *Труды IV Международной конференции"Идентификация систем и задачи управления"SICPRO*, 2005.

[6] Golyandina N. Korobeynikov, A. Basic singular spectrum analysis and forecasting with r. *Computational Statistics & Data Analysis*, 71(934–954), 2014.