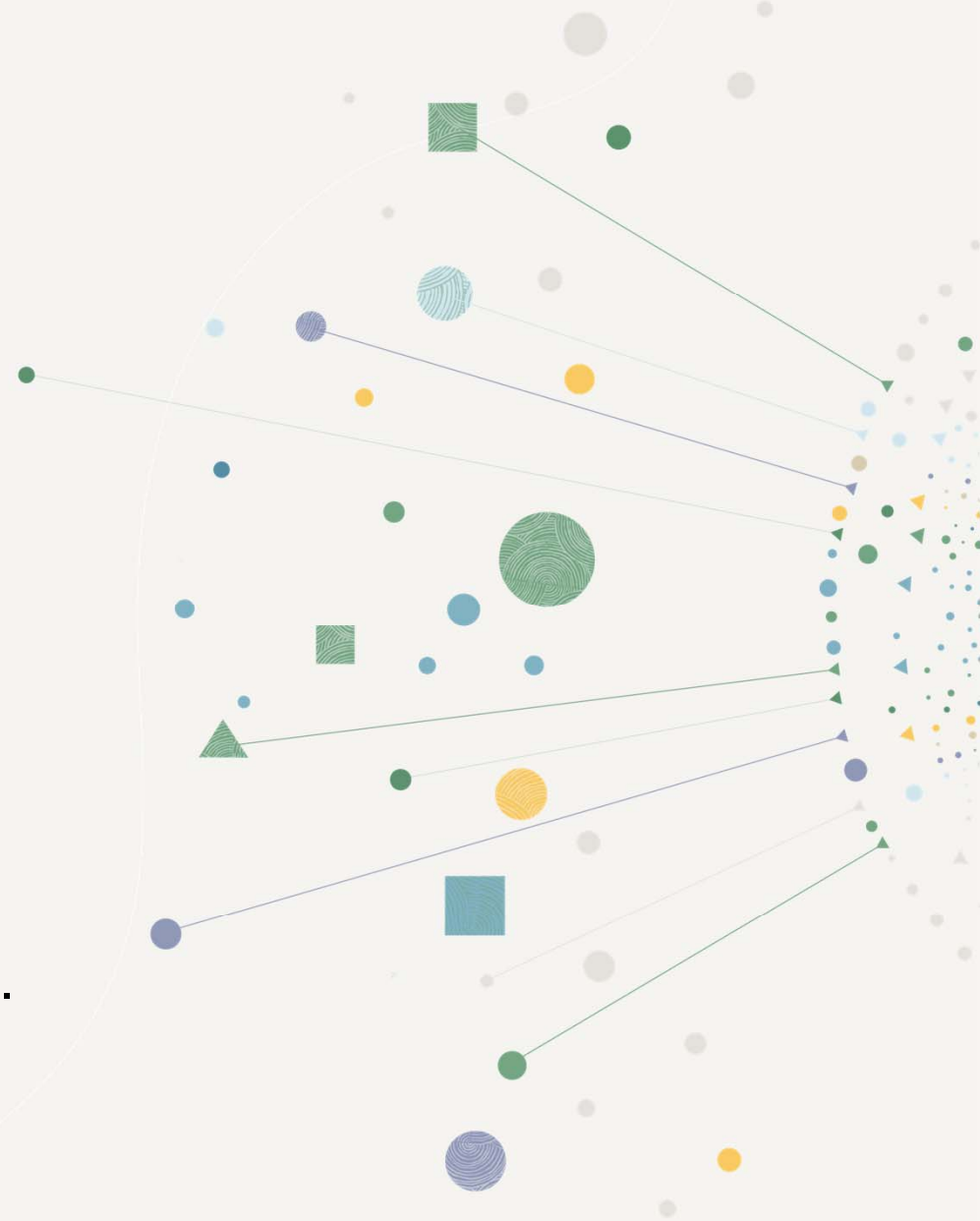
Основы анализа больших данных

Лекция 2. Язык программирования и программная среда R. Ключевые особенности. Классы данных. Векторы в R.



Введение

В настоящее время анализ больших наборов данных и использование параллельной обработки являются активными направлениями исследований в области аналитики данных.

Язык статистического программирования R, несмотря на внутреннюю реализацию в основном однопоточными операциями, может успешно использоваться вместе с различными подходами к параллельным вычислениям в сложных задачах анализа больших данных.

Что такое R

R - язык программирования для статистической обработки данных и работы с графикой, и в тоже время это свободная программная среда для статистических расчетов с открытым исходным кодом, развиваемая в рамках проекта GNU.

R применяется везде, где нужна работа с данными.

R без особых проблем может использоваться и там, где сейчас принято использовать коммерческие программы анализа – MatLab, Maple, SAS/Stat, SPSS, Statistica, MiniTab, Wolfram Mathematica и др.

Основная вычислительная мощь R лучше всего проявляется при статистическом анализе: от вычисления средних величин до вейвлет-преобразований временных рядов.

География использования R очень широка и разнообразна.

История R

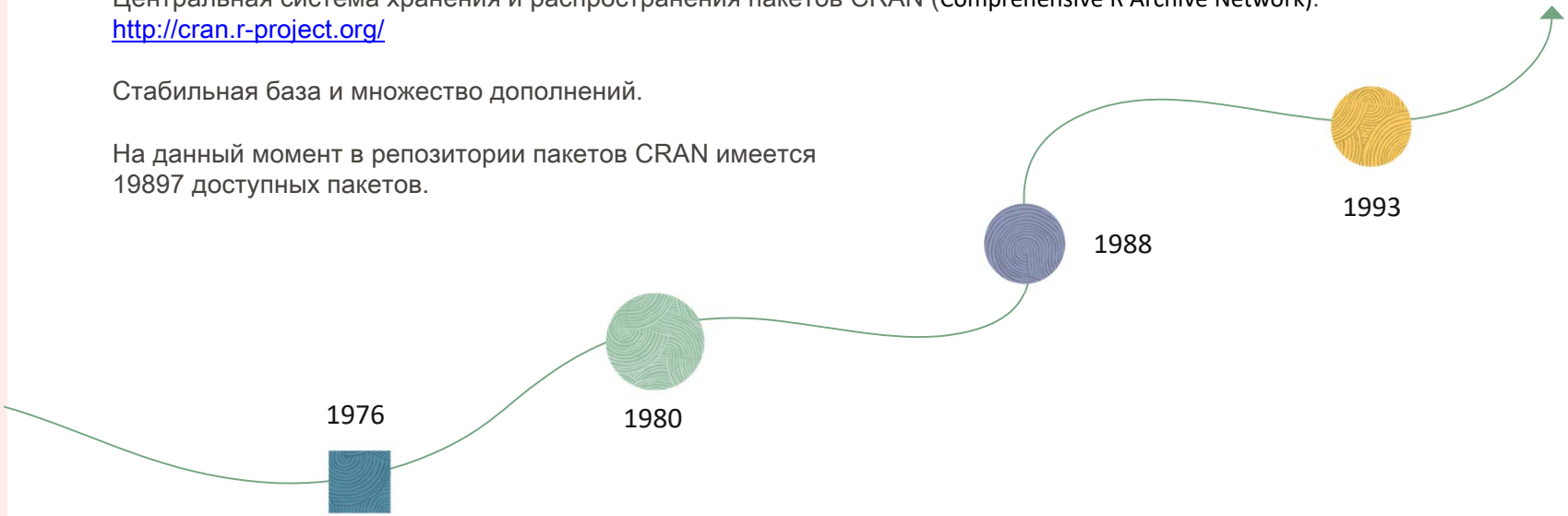
R возник как свободный аналог среды S-PLUS, которая в свою очередь является коммерческой реализацией языка расчётов S (Bell Labs, 1976 г.). Начиная с 3й версии (1988 г.), коммерческая реализация S называется S-PLUS (Insightful, затем TIBCO Software).

В августе 1993 г. двое новозеландских учёных Robert Gentleman и Ross Ihaka анонсировали свою новую разработку, которую они назвали R.

Центральная система хранения и распространения пакетов CRAN (Comprehensive R Archive Network): http://cran.r-project.org/

Стабильная база и множество дополнений.

На данный момент в репозитории пакетов CRAN имеется 19897 доступных пакетов.



Особенности R

Преимущества:
- Гибкость, расширяемость (скрипты, пакеты - install.packages() + library())
- Свободный код
- Кроссплатформенность
- Базовая комплектация R занимает относительно мало места на жестком диске
- Работа с большими массивами данных
- Хорошие графические возможности

Недостатки:
- Интерфейс командной строки
- Нет коммерческой поддержки
- Относительная медлительность

Еще 2 особенности:
1) R результат любой операции с числами трактует как вектор единичной длины. Скаляров в R, вообще говоря, просто нет;
2) Элементы векторов нумеруются с единицы, а не с нуля, как во многих языках программирования.

Параллелизм в R

Ключевой технологией для параллельного выполнения кода на языке R является стандарт OpenMP (https://www.openmp.org/).

Обзор инструментария R для организации параллельных вычислений

- Пакет snow (Simple Network of Workstations)
- Пакет snowfall (Easier cluster computing)
- Пакет snowFT (Fault Tolerant Simple Network of Workstations)
- Пакет parallelLogger
- Пакет future (Unified Parallel and Distributed Processing in R for Everyone)
- Расширения пакета future

Классы (структуры) данных в R

- Векторы (vector)
- Матрицы (matrix)
- Многомерные массивы (array)
- Факторы (factor)
- Списки (list)
- Таблицы данных (data.frame)

Типы данных:
- numeric – числа (делится на integer и double или real)
- complex – объекты комплексного типа
- logical – логические объекты
- character – символьные объекты
- raw – объекты потокового типа

raw < logical < integer < real < complex < character

Векторы в R

- Вектор – самая базовая структура данных в R
- На векторах основаны более сложные структуры (matrix, list, factor, data.frame)
- Векторизация – одна из ключевых концепций R

$$f: \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix}$$

- Полезные функции для работы с векторами: length(), max() и min(), mean(), sum(), prod(), sort(), cumsum()

- Вектор – индексированный набор данных одного типа

Способы задания векторов в R

- Через функцию vector() (или альтернативы – numeric(), logical(), complex(), character())

```
> Vec <- vector(length=2)
> Vec[1] <- 3
> Vec[2] <- 4
Vec
[1] 3 4
```

- Через функцию конкатенации c():

```
> Vec <- c(3, 4)
```

- Через функцию scan():

```
> Vec <- scan()
1: 3 4
3:
Read 2 items
```

- Создание числовых последовательностей – оператор : (с шагом 1), функция seq() (с произвольным шагом)
- Повторение векторов: функция rep()
- Создание именованных векторов с помощью names()
- Через функции, генерирующие (псевдо)случайные числа – например, runif()