

# Python for Practice

## NPRG067

<http://d3s.mff.cuni.cz>



CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

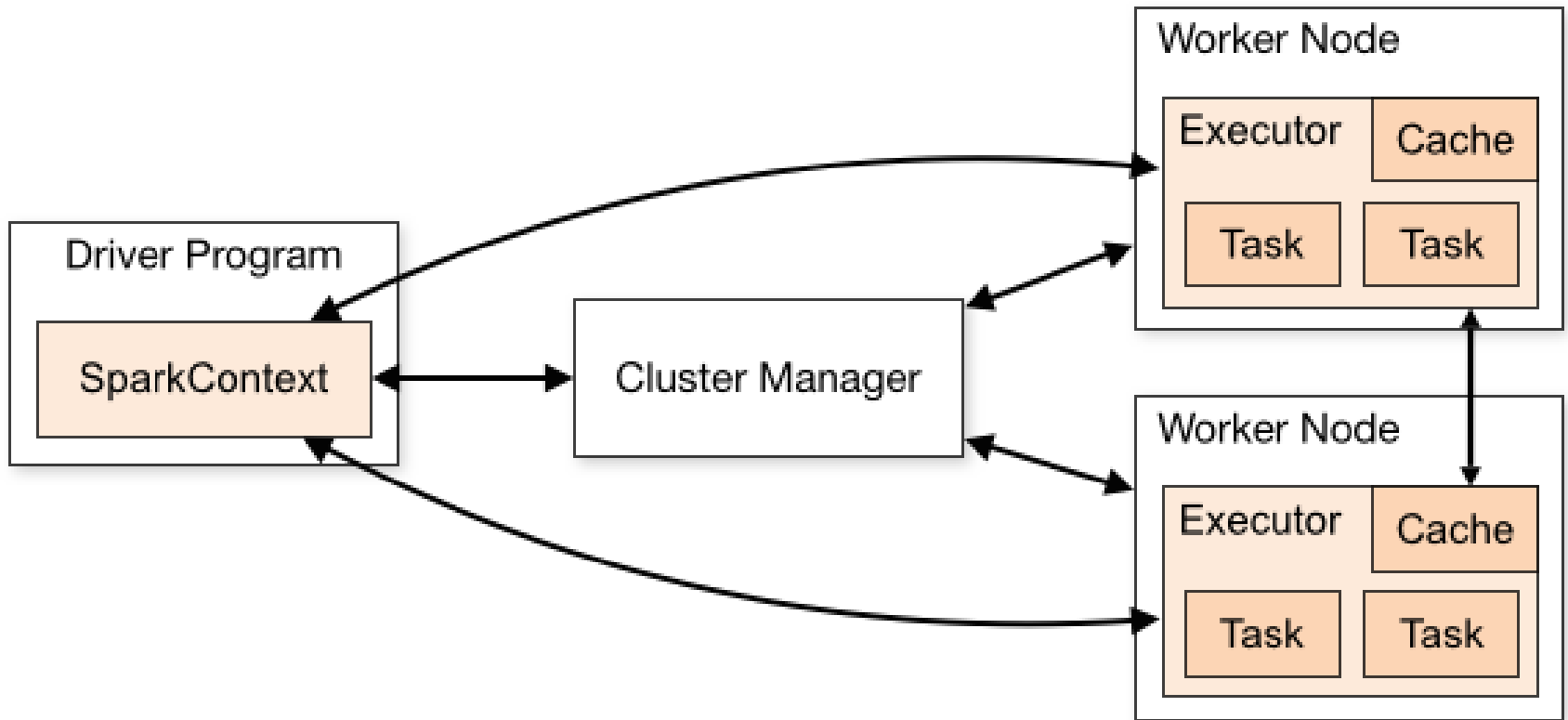
*Tomas Bures*  
*Petr Hnetynka*  
*Ladislav Peška*

{bures, hnetynka}@d3s.mff.cuni.cz  
peska@ksi.mff.cuni.cz

# Spark

- Open-source cluster computing framework
- Part of Apache's family of big-data tools
- Runs as standalone or on top of Hadoop Yarn and Apache Mesos
- Can access data stored in HDFS, Cassandra, Hbase, ...
- Main API in Scala
- Binding to other languages Java, **Python**, R

# Spark Architecture



# Types of API

- RDD (Resilient distributed dataset)
  - Collections of serializable black-box objects
  - More low-level operations
- DataFrames
  - Tables of rows with schema
  - Queries mimic SQL
  - Query optimization before execution
  - Two main variants
    - SQL-like statements (strings)
    - Query builder API
- Seamless interoperability between RDDs and DataFrames
- Note that there is no DataSet API (as in Scala)

# First Example

- Exercise: e01
- Uses the DataFrames API

# Different APIs

- Exercise: e03
- “Show max 10 reviewers which have the salary above average. Sort them by ascending age.”
- Examples:
  - DataFrame API
  - SQL API
  - RDD API

# DataFrame API

- Query building functions
  - Projection: select, selectExpr, drop, withColumn, withColumnRenamed, agg, replace
  - Selection: filter, where
  - Deduplication of rows: distinct, dropDuplicates
  - Handling null values: dropna, fillna
  - Joins: join, crossJoin
  - Set operations: intersect, subtract
  - Grouping: groupBy
  - Operations for online analytical processing (OLAP): cube, rollup, crosstab
  - Ordering, limiting: limit, orderBy / sort, sortWithinPartitions
  - Statistics over the whole dataframe: approxQuantile, cov, corr, ...
- Actions
  - coalesce, repartition
  - collect, take, first, head
  - toLocalIterator
  - count
  - foreach, foreachPartition
  - write.xxx

# DataFrame API

- RDD
  - rdd, toJSON
- Exploratory analysis
  - describe
  - sample, sampleBy
  - randomSplit
- Debugging
  - show, printSchema
  - explain
- Miscellaneous functions
  - cache, persist

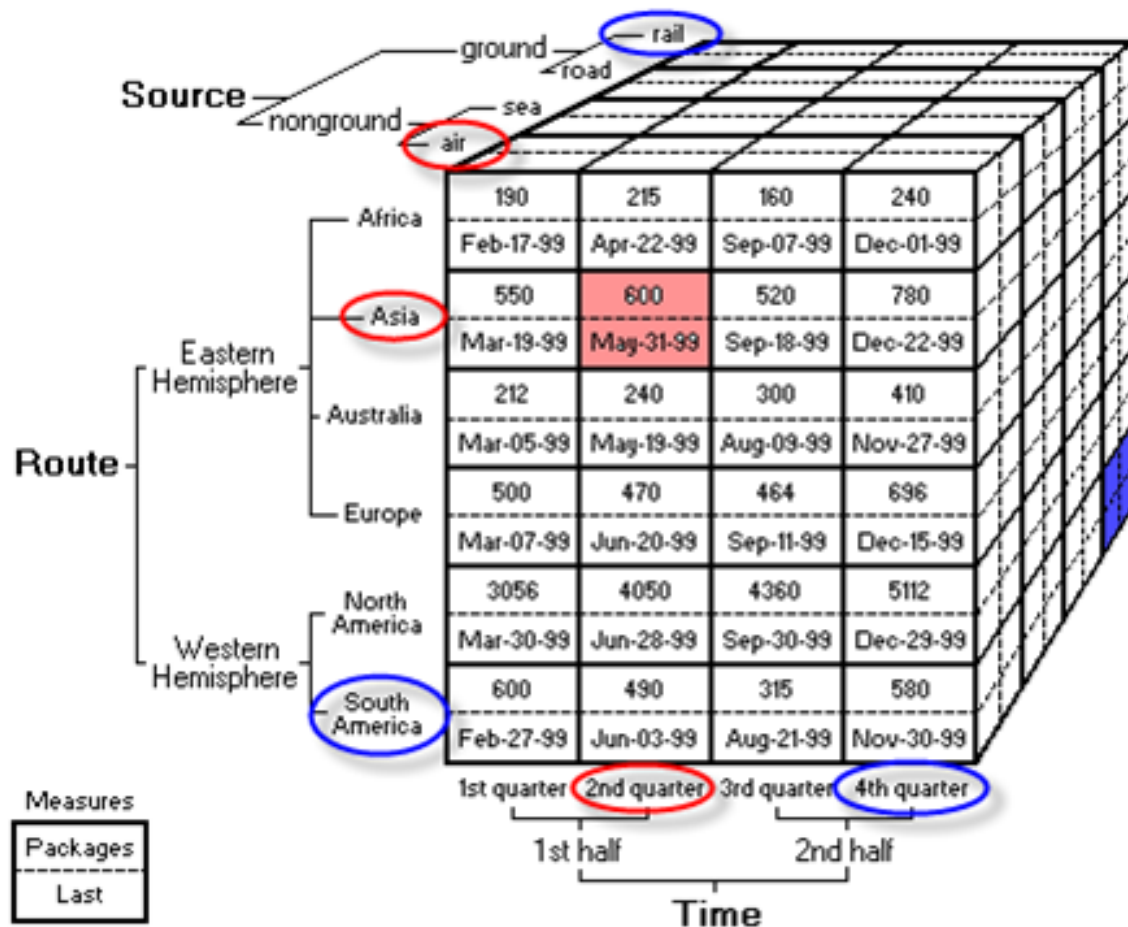


# Aggregations

- Exercise: e05
- “Compute the sum of review lengths by every reviewer and make summary statistics of the lengths along with breakdown by gender and age”

# Cube operator

- Used in online analytical processing
- Returns aggregated data for different combinations of aggregations
- Runs faster than doing the aggregations one-by-one



# Column Mapping

- A number of functions pre-defined for operations over columns:
  - <http://spark.apache.org/docs/2.2.0/api/python/pyspark.sql.html#module-pyspark.sql.functions>
- User defined mapping
  - Via transformation to RDD and then using RDD.map
  - Or via user-defined functions (UDF)
- Exercise: e09

# Windows

- Per-row functions in the context of a window
  - Aka **partition by**
- Window functions:
  - row\_number, rank, dense\_rank, lag, lead, ntile, percent\_rank, rank, cume\_dist
- Exercise: e11
- Exercise: e13

# Determining Trends

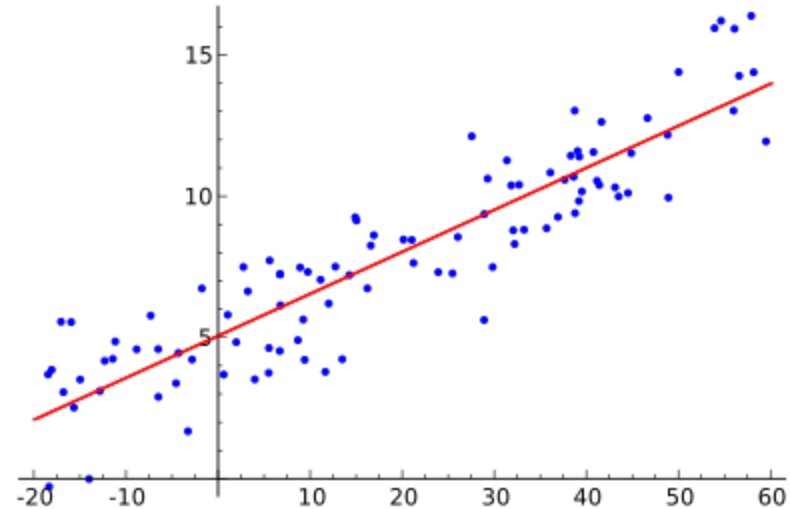
- “Is there a trend in time for the length of reviews per reviewer? E.g. does anyone tend to write longer and longer reviews?”
- Exercise: e16
- Math on the next two slides...

# Simple Linear Regression

- Simple Linear regression

$$y = \alpha + \beta x$$

$$y_i = \alpha + \beta x_i + \varepsilon_i$$



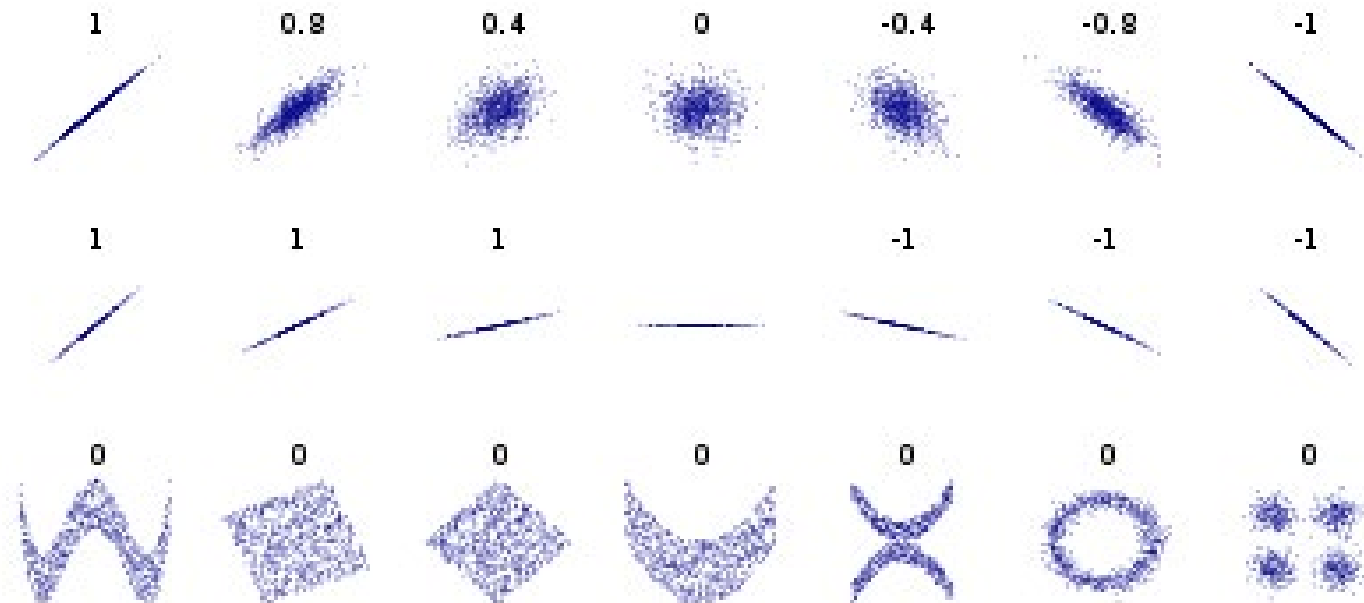
$$\hat{\alpha} = \bar{y} - \hat{\beta} \bar{x},$$

$$\hat{\beta} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

# Quality of Fit

- Pearson's correlation coefficient
  - determines how dependent data are

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$



# Statistical Testing

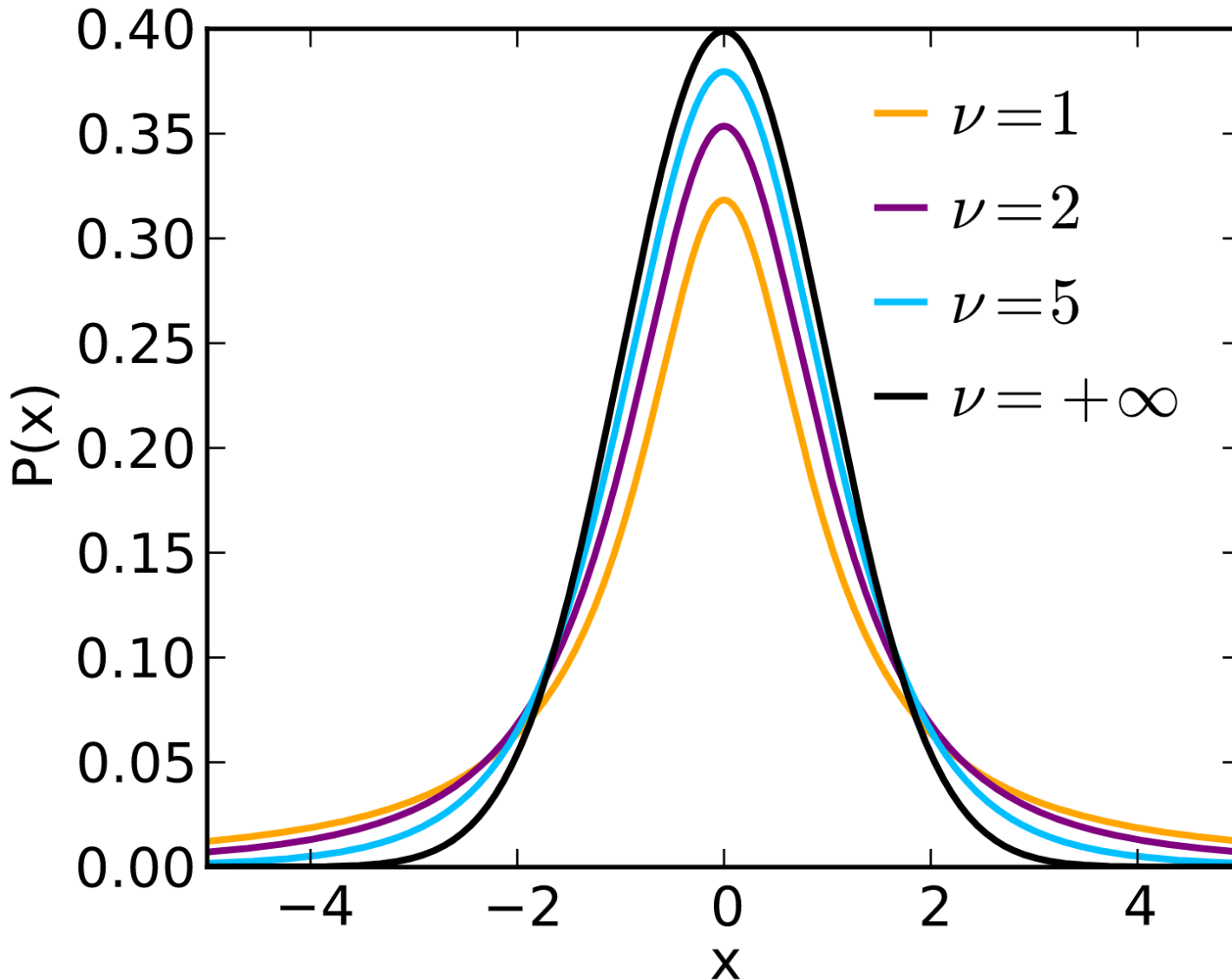
- “Check whether there is a statistical difference between the length of reviews written by men and women.”

- Welch’s t-test with  $\nu$  degrees of freedom

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}}$$
$$\nu \approx \frac{\left(\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}\right)^2}{\frac{s_1^4}{N_1^2 \nu_1} + \frac{s_2^4}{N_2^2 \nu_2}}$$



# t-distribution



# Structured columns

- Columns may contain structured data or lists
- They can be indexed with the usual Pythonic way
  
- `df.select(x.y.z)`
- `df.select(x[0][1])`
  
- Exercise: e19