Collaborative Development in R

A Case Study with the sparsebn package

Bryon Aragam
July 2017

Carnegie Mellon University

@itsrainingdata
@itsrainingdata
http://www.cs.cmu.edu/~naragam
The sparsebn package

An R package for learning large-scale graphical models and complex networks from data.

- **Fast**: Uses Rcpp.
- **Scalable**: Built for large networks.
- **Consistent**: Sound statistical foundation when $p \gg n$. 

![Diagram of a complex network]
**Current status**

**Features**
- 10-100x faster than existing methods: **5000 nodes in ~82s**
- Automatic handling of mixed observational/experimental data
- Structure learning and parameter estimation
- Model selection
- Visualization
- Compatibility with existing R packages

**In the pipeline**
- Even **faster algorithms** for even **bigger graphs**
- More models: Count data, GLMs, etc.
- Blacklists and whitelists
- Cytoscape
- Network comparison and evaluation

Want the latest? [https://github.com/itsrainingdata/sparsebn/tree/dev/](https://github.com/itsrainingdata/sparsebn/tree/dev/)
library("sparsebn")
data("pathfinder")
dat <- sparsebnData(pathfinder$data,
                     type = "continuous")
dags <- estimate.dag(dat)
plotDAG(dags)

5 lines to DAG goodness
Nonconvex, nonsmooth, combinatorial optimization problem \(\Rightarrow\) black-box solvers don't work.

**Customized optimization algorithms** for

- Different types of data
- Different statistical models
- Different objective functions
- Different regularizers
Nonconvex, nonsmooth, combinatorial optimization problem $\Rightarrow$ **black-box solvers don’t work.**

**Customized optimization algorithms** for

- Different types of data
- Different statistical models
- Different objective functions
- Different regularizers

Unlike existing, stable methods, these algorithms were **untested**, **unstable**, and **frequently changing**.

- Rapid prototyping
- Fast release cycles
- Easy to add new methods
Challenges: Humans

Your dev team

- PI (team leader) Suzy
- Senior graduate student (senior dev) Jane
- Junior graduate student (junior dev) Bob
- + other students (devs)
Challenges: Humans

Your dev team

- PI (team leader) Suzy
- Senior graduate student (senior dev) Jane
- Junior graduate student (junior dev) Bob
- + other students (devs)

The setting

- Suzy has an idea for a fancy new R package (app) that uses some pretty cutting edge ideas
Challenges: Humans

*Your dev team*

- PI (team leader) Suzy
- Senior graduate student (senior dev) Jane
- Junior graduate student (junior dev) Bob
- + other students (devs)

*The setting*

- Suzy has an idea for a fancy new R package (app) that uses some pretty cutting edge ideas
- Jane has spent several months developing this fancy new package
Challenges: Humans

Your dev team

- PI (team leader) Suzy
- Senior graduate student (senior dev) Jane
- Junior graduate student (junior dev) Bob
- + other students (devs)

The setting

- Suzy has an idea for a fancy new R package (app) that uses some pretty cutting edge ideas
- Jane has spent several months developing this fancy new package
- Bob has just come on board, and would like to extend Jane’s algorithm
Challenges: Humans

Your dev team

- PI (team leader) Suzy
- Senior graduate student (senior dev) Jane
- Junior graduate student (junior dev) Bob
- + other students (devs)

The setting

- Suzy has an idea for a fancy new R package (app) that uses some pretty cutting edge ideas
- Jane has spent several months developing this fancy new package
- Bob has just come on board, and would like to extend Jane’s algorithm
- Jane leaves
Challenges: Humans

Your dev team

- PI (team leader) Suzy
- Senior graduate student (senior dev) Jane
- Junior graduate student (junior dev) Bob
- + other students (devs)

The setting

- Suzy has an idea for a fancy new R package (app) that uses some pretty cutting edge ideas
- Jane has spent several months developing this fancy new package
- Bob has just come on board, and would like to extend Jane’s algorithm
- Jane leaves
- Bob cries
Two solutions

Each programmer develops their own, independent package

- Pros: Maximum flexibility, development is embarrassingly parallelizable
- Cons: Lengthy dev cycles, duplicated code
- Bob has to re-write / re-factor most of Jane’s code
Two solutions

Each programmer develops their own, independent package

- Pros: Maximum flexibility, development is embarrassingly parallelizable
- Cons: Lengthy dev cycles, duplicated code
- Bob has to re-write / re-factor most of Jane’s code

The whole group develops a single, monolithic package

- Pros: Minimal duplication, efficient code
- Cons: Less flexibility, costly maintenance and upgrades, development is coupled
- Bob has to re-write / re-factor most of Jane’s code
Is there a just-right Goldilocks solution?
sparsebnUtils defines the **interface** that ccdrAlgorithm / discretecdAlgorithm must adopt (i.e. I/O):

\[
\text{Input} \rightarrow \textbf{black box} \rightarrow \text{Output}
\]

sparsebn simply loads these dependencies.
Your R package

- **Base package** is your workhorse
  - Common classes, subroutines, utilities, and glue code
  - Defines I/O for algorithm packages: **Think carefully about this**
  - Collaboratively maintained
- **Algorithm / model package(s)**
  - Implements different algorithms
  - Independently maintained
- **Wrapper package**
  - **Doesn’t do anything**: Just loads dependencies
  - All functionality should live in dependencies
  - Exception: Wrapper methods, datasets
Devs don’t want you to know this one cool trick

**Advantages**

- **Independent, asynchronous development**
  - Bob no longer relies on Jane

- CRAN submission takes longer
- Multiple RStudio windows...?
Devs don’t want you to know this one cool trick

Advantages

- **Independent, asynchronous development**
  - Bob no longer relies on Jane
- **Rapid prototyping**
  - Filter out good/bad ideas quickly with minimal overhead
Devs don’t want you to know this one cool trick

Advantages

- **Independent, asynchronous development**
  - Bob no longer relies on Jane
- **Rapid prototyping**
  - Filter out good/bad ideas quickly with minimal overhead
- **Flexibility**
  - Works well with devs of different skill levels, seniority
  - Model packages mature at own pace
Advantages

- **Independent, asynchronous development**
  - Bob no longer relies on Jane
- **Rapid prototyping**
  - Filter out good/bad ideas quickly with minimal overhead
- **Flexibility**
  - Works well with devs of different skill levels, seniority
  - Model packages mature at own pace

Disadvantages

- **CRAN submission takes longer**
- Multiple RStudio windows...?
Thanks!

Thank you for your time!

Collaborators (in alphabetical order)

- Jiaying Gu, UCLA
- Dacheng Zhang, UCLA
- Qing Zhou, UCLA
- **You?** (submit a pull request! [https://github.com/itsrainingdata/](https://github.com/itsrainingdata/))

References


