GSoC 2016 Proposal: Abhinav Agarwal

Project Info

Project title: Speed and visualization Improvements to R package 'BayesBD'
Short title: Improvements to BayesBD
Idea Page: Link

Student Bio

I am Abhinav Agarwal. I love to code and I love R. I have been using R for my projects and assignments in my university Machine Learning and Data Mining courses. Last year I applied to GSoC for the first time under Scilab but was not selected. I was working on the Scidoe toolbox. This project idea interests me because I think BayesBD has great scope for development and application. There are not many boundary detection packages in R and if taken in the right direction BayesBD can prove to be even more useful to the R community than it is now. I am qualified because of my experience with R and C. I have used both C and C++ for the past 6 years or so ever since I started coding. I would go on to say C is my primary goto all purpose language. I have also used QT for building various gui projects in C++. Some projects I have recently worked on are:

1. Research project under Dr. K. Hari Babu, CSIS Department, BITS Pilani 'Congestion Control Techniques in a SDN aware TCP module'
2. Research project under Dr. A. S. Mandal, Head, Perception Engineering Lab, Central Electronics Engineering Research Institute (CEERI), Pilani 'Facial Expression Recognition Systems using Saliency maps'
3. Various machine learning projects including a full fledged active learning email filter, a speaker recognition system, document classifiers, text mining tools all written in R.
4. I built a compiler from scratch for a toy language, automating the calculation of first and follow sets and parse tree generation.

I have also done various MOOCS with highest grades pertaining to Data Science and related concepts. Some are Machine Learning by Stanford University, Intro to NLP by University of Michigan, the first three Data Science specialization courses by John Hopkins University which are 'The Data Scientist's Toolbox', 'R Programming' and 'Getting and Cleaning Data, and some of the Data Camp courses. I am proficient with git and use it frequently. I have hosted some code on my github profile. I will hopefully continue contributing after the GSoC time period because I truly believe in Open Source ideologies and it would give me great joy to become an active and significant member of the community.

Contact Information

Name: Abhinav Agarwal
Postal Address: H.No. 394, Sector - 40, Urban Estate, Gurgaon, Haryana 122003, India
Telephone: +91-8239820850
Email: aagarwal.gtr@gmail.com (Primary), f2013124@pilani.bits-pilani.ac.in
Other: Skype: aagarwal.gtr abhinav.io Google+

Student Affiliation

Institution: Birla Institute of Technology & Science (BITS), Pilani, India
Program: Bachelor of Engineering (with Honors), Computer Science
Stage of Completion: Third year, expected to graduate in May 2017

Schedule Conflicts

During most of the community bonding period (April 22 - May 22) I will have regular classes at college. During most of the coding period (May 23 - Aug 23) I will not have any engagements at all. Mid July onwards I will most probably taking a course but I do not expect it to be a hindrance since it won't demand much of my time. In conclusion, no perceivable conflicts.

Mentors

Name: Meng Li
Email: meng.li@stat.duke.edu
Contact History: I have been in touch with Meng Li through email after he put up the idea page. I have had conversations with him regarding my attempt at the qualification task, my programming background and the idea I had in my mind and asked for his approval for writing this proposal.

Names: Vijay Barve
Emails: vijaybarve@ku.edu
Contact History: I haven’t had a chance to contact him as his details were declared very recently (25 March).

Coding Plan & Methods

Speed & Efficiency Improvements:

Detecting boundary of an image based on noisy observations is a fundamental problem of image processing and image segmentation.It has wide applications such as tumor detection in medical fields, development assessment in economics, climate changes in environment studies, etc. While there are many matured statistical methods in this field, the visualization of them is far from satisfactory thus prevents a larger scientific community to actually use them. The R package `BayesBD` is a package to implement nonparametric Bayesian boundary detection. The goal of the project is to improve the speed of estimation procedure and provide visualization.
In the referenced paper[1], authors proposed a nonparametric Bayesian method tailored to detect the boundary, which is viewed as a closed smooth (d − 1)- dimensional manifold without boundary.
The proposed method achieves all the following four goals (i)–(iv) when estimating the boundary.

1. Guaranteed geometric restrictions on the boundary such as closedness and smoothness.
2. Convergence at the (nearly) minimax rate , adaptively to the smoothness of the boundary.
3. Possibility and convenience of joint inference
4. Computationally efficient algorithm.

In the proposed Bayesian approach, an efficient Markov chain Monte Carlo (MCMC) sampling is designed based on the analytical eigen decomposition of the squared exponential periodic (SEP) kernel , for various noise distributions.
Authors also developed a very efficient MCMC method for sampling posterior distribution based on a SEP Gaussian process prior using the explicit eigen structure of the SEP Gaussian process obtained. The algorithm is generic and hence can be used for posterior computation in other curve estimation problems on the circle such as directional data analysis while using the SEP Gaussian process prior.

The current implementation of BayesBD function is in R language. To improve the performance of this function, I will implement it in C++. A simple MCMC (Markov Chain Monte Carlo) algorithm for constructing a Gibbs sampler for the bivariate distribution in C shows a speedup of 53 times over R-code. Following is the compared code described in the blog[3]:

R:

```
gibbs=function(N,thin)
{
    mat=matrix(0,ncol=3,nrow=N)
    mat[,1]=1:N
    x=0
    y=0
    for (i in 1:N) {
        for (j in 1:thin) {
            x=rgamma(1,3,y*y+4)
            y=rnorm(1,1/(x+1),1/sqrt(2*x+2))
        }
        mat[i,2:3]=c(x,y)
    }
    mat=data.frame(mat)
    names(mat)=c("Iter","x","y")
    mat
}

writegibbs=function(N=50000,thin=1000)
{
    mat=gibbs(N,thin)
    write.table(mat,"data.tab",row.names=FALSE)
}

writegibbs()
```

C:

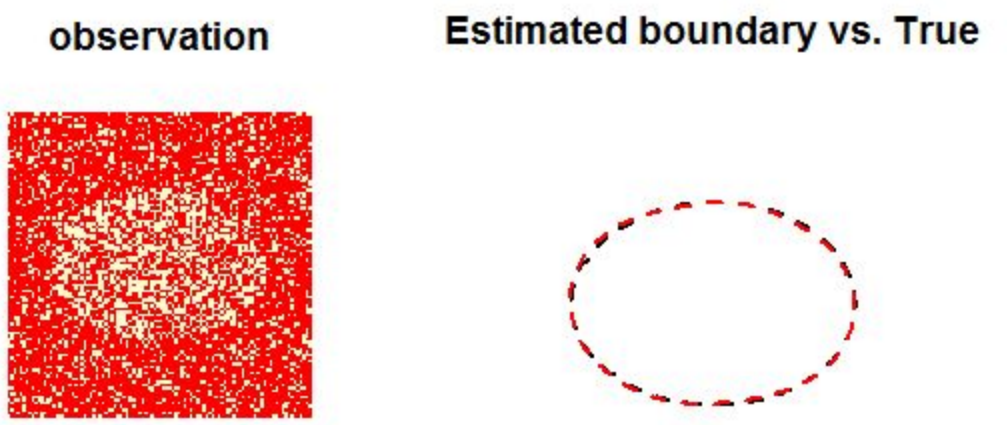
```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_randist.h>

void main()
{
  int N=50000;
  int thin=1000;
  int i,j;
  gsl_rng *r = gsl_rng_alloc(gsl_rng_mt19937);
  double x=0;
  double y=0;
  printf("Iter x y\n");
  for (i=0;i<N;i++) {
    for (j=0;j<thin;j++) {
      x=gsl_ran_gamma(r,3.0,1.0/(y*y+4));
      y=1.0/(x+1)+gsl_ran_gaussian(r,1.0/sqrt(2*x+2));
    }
    printf("%d %f %f\n",i,x,y);
  }
}
```

I will use Rcpp to embed C++ code in R language. The best possible way to combine R and C++, would be to use R for the parts that don't involve serious computation, taking advantage of R's rapid coding, ease of use, input/output capabilities and graphics, and using C++ for the parts involving heavy computations. I am confident about this strategy as the same strategy is used by various other R packages. MCMCpack package also uses a conventional R interface, uses R for error checking, compiled C and C++ for model fitting (which is extremely fast), and the coda package for analysis of the posterior sample.

There are various libraries in C++ for implementation of MCMC algorithms such as MCMC++, I will write unit tests for the above function. These unit tests would be first written against the function BayesBD.binary. These functional tests will ensure my enhanced code in C++ is functionally correct. I will also add performance tests for this function. We will do testing for original BayesBD.binary and enhanced function. Performance enhancement would be measured by running multiple tests on same environment. We would take average of multiple runs of same test to remove variables like load average on environment.

VIsualization Improvements:

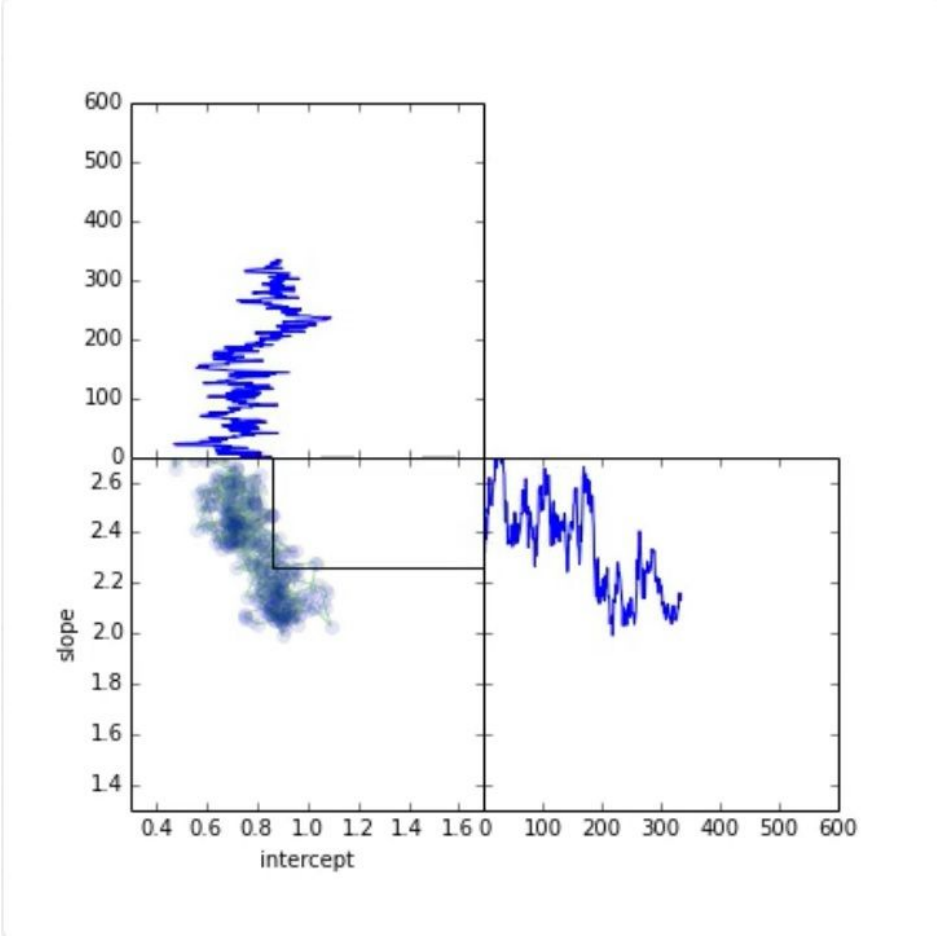
Visualization of an advanced statistical method, like ours is very important for appeal of the package to the general novice users. The current visualisation is hardcoded for an ellipse, a circle and so on. For a general user, say a doctor to use this package he must be able to input an external image, see how the algorithm is working on it during iterations and how the final result is being produced. The current visual representation of the result is show in this image:



I propose to have comprehensive visualization function that will do what I have described and animate boundary detection iterations on the plot as the algorithm is running. There are various visualization techniques for MCMC procedure.

[4] describes an example like I would want to create for the animation part. I will add an input output interface to this.

Here is a screenshot from this animation video (within a different context):



The R code will call the C functions for each iteration and extract returned data after each iteration. This data will then be animated on the plot. This visualisation code will therefore be written in R. There is a similar implementation in the asbio[7] package. I will consult it for design and layout.

Earlier I was considering building a QT GUI tool for the package that would include the same this visualization and be a full front end for interactive usage. But I understand that speeding up the algorithm is the first priority since that is primarily what will constitute ease of usage. So, I abandoned that idea.

Timeline

- Community Bonding: April 22 - May 22: Clear doubts with mentors, if any. Make sure we are on the same page. I plan on doing extensive reading of the relevant literature including the paper [1] and package vignette.

May 23 – June 20: First Half

- Week 1: May 23 – June 29: Write unit tests for BayesBD.binary image function. This will help in understanding all the corner cases of the algorithm.
- Week 2: May 30 – June 5: Implement the BayesBD function in Rcpp. Ensure that unit tests written in previous week are passing. This ensures that code is functionally correct.
- Weeks 3 & 4: June 6 – June 19: Performance testing of above function in various scenarios. Discussion with mentor on performance improvement gained, and looking at ways to further improve if required. Write documentation. Prepare Midterm report.

Week 5: June 20 – June 27: Midterm Evaluations Week. Submit/post midterm report. Discuss progress so far. Re-evaluate schedule. Deal with backlogs, if any.

Midterm Deliverables: Significant performance improvement in BayesBD function by implementing it in C++. Adding unit tests to ensure alternate implementation does not break any functionality.

July 3 – August 17: Second Half

- Weeks 6 & 7: June 27 – July 10: Implement functions for visualization after submitting a mockup for mentor approval.
- Weeks 8 & 9: July 11 – July 24: Write docs & tests for visualization, test everything, verify results, solve possible issues.
- Weeks 10 & 11: July 25 – August 7: Buffer weeks.
- Week 12: August 8 – August 14: Pencils Down. Prepare Final Report.

- Week 13: August 15 – August 24: Final Evaluations Week. Submit final report. Review project.

Final Deliverables: Midterm deliverables + functions for visualization, all documentation and tests, final report.

I always hope to finish ahead of projected schedule and overwork the first few weeks. Since this project involves deciding coding scheduling since if performance gain is not satisfactory I will have to rethink approach, I have left last two weeks as buffer. They also serve as backup if any unforeseen circumstances force delays.

Management of Coding Project

Testing & Documentation:

I plan to post weekly updates with current status and on my blog at abhinav.io. Any delayed posts will indicate a problem but I will always keep my mentor updated. I will use GitHub to host my repo. I plan to commit my work everyday. No commit for more than three working days without an explanation on my blog and to the mentor will indicate a problem. BayesBD uses the devtools workflow and I will follow that as well. For writing tests for the ported code I have two options: use RUnit conventionally for the Rcpp code but in the process expose the C++ code to R, or pick one of the great many C++ unit testing frameworks like the Boost Test library, or Google C++ Testing library. I choose to do the former since there is we have no constraint of exposing the C++ code to R and I assume it will be more efficient that way. And, because there will still be non C++ code in the package which will have to be written tests for. Having a single testing convention is better than having separate ones for the R code and the C++ code for sake of clarity.

Test

Medium: write a C/C++ function for a loop and call this function in R.
There are three ways this can be done: the .C function interface, the .Call function interface and the Rcpp package. I chose to use Rcpp because the first two options need the user to compile the C code using R CMD SHLIB adding the need for manual intervention which is not preferred.

```
#install package Rcpp if not found
if(!require(Rcpp)) {
  install.packages("Rcpp")
  library(Rcpp)
}

#simple C++ function which sums natural numbers till and including n
cppFunction('int func1(int n) {
  int i, sum = 0;
  for(i = 1; i <= n; i++)
    sum += i;
  return sum;
}')

#call to the C++ function
func1(10)

#alternatively use this function to source the code from the cpp file
#which contains the definition of func2() multiplying natural numbers
till and including n
sourceCpp("func2.cpp")
func2(10)
```

Func2.cpp

```
#include <Rcpp.h>

using namespace Rcpp;
// [[Rcpp::export]]
int func2(int n) {
  int i, sum = 1;
  for(i = 1; i <= n; i++)
    sum *= i;
  return sum;
}
```

References

[1] Li, M. and Ghosal, S.(2015). Bayesian Detection of Image Boundaries

[2] Li, M. (2015), BayesBD, R package for Bayesian boundary detection in images using Gaussian process priors.

[3] Wilkinson, D. Gibbs sampler in various languages (revisited)

[4] Wiecki, T. Animating MCMC with PyMC3 and Matplotlib

[5] Flaxman, A. MCMC in Python, PyMC Step Methods and their pitfalls

[6] Martin, A. and Quinn, K. MCMCpack Developer Documentation and Coding Specification

[7] Animation of MCMC walks in bivariate normal space: anm.mc.bvn {asbio}