Semiparametric Regression of Big Data in R

Nathaniel E. Helwig

Department of Statistics
University of Illinois at Urbana-Champaign

CSE Big Data Workshop: May 29, 2014

Outline of Talk

1) Introduction to R
- Downloading R
- Basic calculations
- Using R functions
- Object classes in R

2) High Performance Computing
- Limitations of R
- Optimized BLAS
- Parallel computing
- Big data issues

3) Flights Example
- Reading data into R
- Parametric analysis
- Nonparametric analysis
- Semiparametric analysis

4) Miscellaneous
- El Niño example
- EEG example
- Twitter example
- Future work

R = Free and Open-Source Statistics

R is a free and open-source software environment for statistics.

- Created by Ross Ihaka and Robert Gentleman (at the University of Auckland, New Zealand)
- Based on S language created by John Chambers (at Bell Labs)
- Currently managed by The R Project for Statistical Computing http://www.r-project.org/

You can freely download R for various operating systems:

- Mac
- Windows
- Linux

RStudio IDE

RStudio IDE is a free and open-source integrated development environment (IDE) for R.

- Basic R interface is a bit rough (particularly on Windows)
- RStudio offers a nice environment through which you can use R
- Freely available at http://www.rstudio.com/

You can freely download RStudio IDE for various operating systems:

- Mac
- Windows
- Linux

Storing and Manipulating Values in R

Define objects `x` and `y` with values of 3 and 2, respectively:

```
> x=3
> y=2
```

Some calculations with the defined objects `x` and `y`:

```
> x+y
[1] 5
```

```
> x*y
[1] 6
```

```
> x^y
[1] 9
```

```
> x-y
[1] 1
```

```
> x/y
[1] 1.5
```

```
> x==y
[1] FALSE
```

Warning: R is case sensitve, so `x` and `X` are not the same object.

Some Basic R Functions

Define objects `x` and `y`:

```
> x=c(1,3,4,6,8)
> y=c(2,3,5,7,9)
```

Calculate the correlation:

```
> cor(x,y)
[1] 0.988765
```

Calculate the covariance:

```
> cov(x,y)
[1] 7.65
```

Combine as columns

```
> cbind(x,y)
     x y
[1,] 1 2
[2,] 3 3
[3,] 4 5
[4,] 6 7
[5,] 8 9
```

Combine as rows

```
> rbind(x,y)
  [,1] [,2] [,3] [,4] [,5]
x    1    3    4    6    8
y    2    3    5    7    9
```

Object-Oriented Style Programming

R is an object-oriented language, where an "object" is a general term.

Any R object $\mathfrak{x}$ has an associated "class", which indicates the type of object that $\mathfrak{x}$ represents.

- S3 classes: simple naming convention used by most R packages
- S4 classes: more advanced (true) object-oriented class system

Overall idea of object-oriented style programming:

- Some R functions are only defined for a particular class of input $\mathfrak{x}$
- Other R functions perform different operations depending on the class of the input object $\mathfrak{x}$.

Some Basic R Classes

`numeric` class:

```
> x=c(1,3,-2)
> x
[1]  1  3 -2
> class(x)
[1] "numeric"
```

`character` class:

```
> x=c("a","a","b")
> x
[1] "a" "a" "b"
> class(x)
[1] "character"
```

`integer` class:

```
> x=c(1L,3L,-2L)
> x
[1]  1  3 -2
> class(x)
[1] "integer"
```

`factor` class:

```
> x=factor(c("a","a","b"))
> x
[1] a a b
Levels: a b
> class(x)
[1] "factor"
```

Some More R Classes

`matrix` class:

```
> x=c(1,3,-2)
> y=c(2,0,7)
> z=cbind(x,y)
> z
      x y
[1,]  1 2
[2,]  3 0
[3,] -2 7
> class(z)
[1] "matrix"
```

`data.frame` class:

```
> x=c(1,3,-2)
> y=c("a","a","b")
> z=data.frame(x,y)
> z
   x y
1  1 a
2  3 a
3 -2 b
> class(z)
[1] "data.frame"
```

Two Major Limitations of R

R is great for statistics and data visualization, but. . .

1. R is NOT optimized for parallel computing
   - Default build uses single-threaded BLAS
   - Default build has no parallel computing ability

2. R reads all data into virtual memory
   - Default build can NOT read data from file on demand
   - Need to buy more RAM??

Linking R to Faster BLAS

R's standard BLAS library is very stable, but single-threaded.

Can link R to optimized BLAS. Popular choices include:

- OpenBLAS: http://www.openblas.net/
- ATLAS: http://math-atlas.sourceforge.net/
- MKL: https://software.intel.com/en-us/intel-mkl

For your particular OS and BLAS combination, instructions to link to R can (typically) be found on someone's blog. . . Google it

- OpenBLAS is simple to link to R on Unbuntu (I've heard)

Linking R to Faster BLAS in Mac OS

Mac OS comes with `vecLib` BLAS, which are simple to link to R.

- Supported by Apple's Accelerate Framework

In R 2.15 and before, input the following into Terminal:

```
cd /Library/Frameworks/R.framework/Resources/lib
ln -sf libRblas.vecLib.dylib libRblas.dylib
```

In R 3.0 and above, `libRblas.vecLib.dylib` is not included with the R download, but you can still link to `vecLib` by replacing `libRblas.vecLib.dylib` with

```
/System/Library/Frameworks/Accelerate.framework/Frameworks/vecLib.framework/Versions/Current/libBLAS.dylib
```

Linking R to Faster BLAS in Mac OS (example)

Example on MacBook Pro (2.53 GHz Intel Core 2 Duo, 4GB RAM):

```
# with R's standard BLAS
> set.seed(123)
> x=matrix(rnorm(10^6),10^4,100)
> system.time({svd(x)})
   user  system elapsed
  0.798   0.008   0.827


# with Apple's vecLib BLAS
> set.seed(123)
> x=matrix(rnorm(10^6),10^4,100)
> system.time({svd(x)})
   user  system elapsed
  0.574   0.013   0.589
```

Parallel/Multicore Computing Packages

R has many add-on packages for parallel computing:

- `multicore`: basic multicore/parallel processing
- `Rmpi`: interface (wrapper) to message-passing interface
- `snow`: Simple Network Of Workstations
- `snowfall`: interface (wrapper) to `snow`
- Note that there are several others too (see link below)

http://cran.r-project.org/web/views/HighPerformanceComputing.html

Parallel Package Comparisons

Helpful table from Schmidberger et al. (2009):

	Learnability	Efficiency	Memorability	Errors	Satisfaction
RmpiR	+	−−	++	+	+
rpvm	−−	−−	+	−	−−
nws	+	+	++	+	+
snow	+	++	++	+	++
snowFT	+	++	++	+	++
snowfall	++	++	++	+	++
papply	+	+	++	+	0
biopara	−−	−−	0	0	−−
taskPR	+	−	++	0	−

Table 3: Assessment of the usability of the R packages for computer clusters (ranging from ++ to −−).

++ is good (well-developed and stable)
−− is bad (under-developed and unstable).

R $\neq$ Big Data Software

R is NOT a big data software

- R holds all objects in virtual memory
- R has implicit memory limits
- Maximum array dimension: $2^{31} - 1 \approx 2 \times 10^9$

Address space memory limits (for all objects) are system-specific

- Windows
  - 32-bit: 2Gb OS-imposed limit
  - 64-bit: 8Tb OS-imposed limit
- Unix
  - 32-bit: 4Gb OS-imposed limit
  - 64-bit: "essentially infinite" (e.g., 128Tb for Linux on x86_64 cpus)

Some Notes on `bigmemory` Package

The `bigmemory` package features the `big.matrix` object class.

- Points to data structure in C++
- `big.matrix` objects are *call by reference*, so need special analysis functions (typical R objects are call by value)
- `biganalytics` and `bigtabulate` packages provide analysis functions for `big.matrix` objects

Uses standard C++ data structures:

Type	Bytes	Approx. Range
`double`	8	$1.7\text{E} \pm 308$
`integer`	4	$\pm 2,147,483,647$
`short`	2	$\pm 32,767$
`char`	1	± 127

Airline On-Time Performance

From Statistical Computing and Statistical Graphics 2009 Data Expo, *American Statistical Association*
http://stat-computing.org/dataexpo/2009/

Full data set contains flight arrival/departure details for all US commercial flights from October 1987 to April 2008.

- Have 29 variables about arrival/departure
- Have about 120 million records (flights)
- Data size: 12GB

We will focus on data from 2003–2008 (approximately 4GB).

Big Data Problems and Solutions

Problem:
With 4GB of RAM, how can I even get the 4GB of data into R??

Solution 1:
Preprocess the data outside of R, and input the (smaller) subset of data you want to analyze in R.

Solution 2:
You could use the `big.matrix` function from `bigmemory` package (can read data from memory or file, so you can exceed RAM)

Simple BASH Preprocessing (via awk)

Shell script that finds all flights with non-missing data and 1–120 minute delays (and a valid departure time):

```
#!/bin/bash

# flights.sh
#
#
# Created by Nathaniel E. Helwig on 5/20/14.
# Copyright 2014 University of Illinois. All rights reserved.


cd "/Users/Nate/Desktop/CSE_2014/Rcode/data/"

pr *.csv | cut -d "," -f1,2,5,15,16 | \
  awk -F, '(!/Year/) && (!/NA/) && ($4 > 0) && ($4 <= 120) && ($5 > 0) && ($5 <= 120) \
          { if(length($3)==3 && substr($3,2,2)<=59) { \
               print $1","$2","substr($3,1,1)","$4","$5 \
             } else if(length($3)==4 && substr($3,1,2)<=24 && substr($3,3,2)<=59) { \
               print $1","$2","substr($3,1,2)","$4","$5 \
             } \
          }' > flights.csv

wc -l flights.csv
```

Saves `Year`, `Month`, `DepHour`, `DepDelay`, and `ArrDelay` in file `flights.csv`; then prints number of lines in `flights.csv`

Reading Flights Data into R (first time)

First we need to open R and load the big packages:

```
> library(bigmemory)       # ver 4.4.6
Loading required package: bigmemory.sri
Loading required package: BH

bigmemory >= 4.0 is a major revision since 3.1.2; please see packages
biganalytics and and bigtabulate and http://www.bigmemory.org for more information.

> library(biganalytics)    # ver 1.1.1
> library(biglm)           # ver 0.9-1
Loading required package: DBI
> library(bigsplines)      # ver 1.0-1
```

First time you read-in data use `read.big.matrix` function:

```
> mypath="/Users/Nate/Desktop/CSE_2014/Rcode/data/"
> flights<-read.big.matrix(filename=paste(mypath,"flights.csv",sep=""),
                           col.names=c("Year","Month","DepHour","DepDelay","ArrDelay"),
                           type="short",backingfile="flights.bin",backingpath=mypath,
                           descriptorfile="flights.desc")
```

and create a `backingfile` and `descriptorfile` for later read-ins.

Reading Flights Data into R (another time)

Look at data we just read-in:

```
> class(flights)
[1] "big.matrix"
attr(,"package")
[1] "bigmemory"
> flights[1:4,]
     Year Month DepHour DepDelay ArrDelay
[1,] 2003     1      17       23       29
[2,] 2003     1      10       52       18
[3,] 2003     1      17        4        3
[4,] 2003     1      18       64       82
```

When you reread-in the data, use `attach.big.matrix` function:

```
> library(bigmemory)
> mypath="/Users/Nate/Desktop/CSE_2014/Rcode/data/"
> flights<-attach.big.matrix("flights.desc",backingfile="flights.bin",backingpath=mypath)
> flights[1:4,]
     Year Month DepHour DepDelay ArrDelay
[1,] 2003     1      17       23       29
[2,] 2003     1      10       52       18
[3,] 2003     1      17        4        3
[4,] 2003     1      18       64       82
```

Some Descriptive Statistics

```
# print data dimensions and column names
> dim(flights)
[1] 11157390        5
> colnames(flights)
[1] "Year"     "Month"    "DepHour"  "DepDelay" "ArrDelay"


# look at variable ranges
> apply(flights,2,range)
     Year Month DepHour DepDelay ArrDelay
[1,] 2003     1       1        1        1
[2,] 2008    12      24      120      120


# look at variable means
> apply(flights,2,mean)
       Year       Month     DepHour    DepDelay    ArrDelay
2005.690344    6.518229   14.691607   28.984666   27.382003


# look at correlation between DepDelay and ArrDelay
> cor(flights[,4],flights[,5])
[1] 0.8635188
```

Simple Linear Regression (SLR) Model

The simple linear regression model has the form

$$y_i = \mu + \beta x_i + \epsilon_i \qquad \text{or} \qquad \mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

for $i \in \{1, \ldots, n\}$ where

- $\mathbf{y} = (y_1, \ldots, y_n)'$ with $y_i \in \mathbb{R}$
- $\mu \in \mathbb{R}$ is the regression intercept
- $\beta \in \mathbb{R}$ is the regression slope and $\boldsymbol{\beta} = (\mu, \beta)'$
- $x_i \in \mathbb{R}$ is the predictor for the i-th observation
- $\mathbf{X} = [\mathbf{1}_n, \mathbf{x}]$ is the design matrix with $\mathbf{x} = (x_1, \ldots, x_n)'$
- $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)'$ with $\epsilon_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$

Ordinary Least-Squares solution: $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \sim \mathrm{N}(\boldsymbol{\beta}, \sigma^2(\mathbf{X}'\mathbf{X})^{-1})$

SLR of Full Flights Data

Predict `ArrDelay` from `DepDelay`:

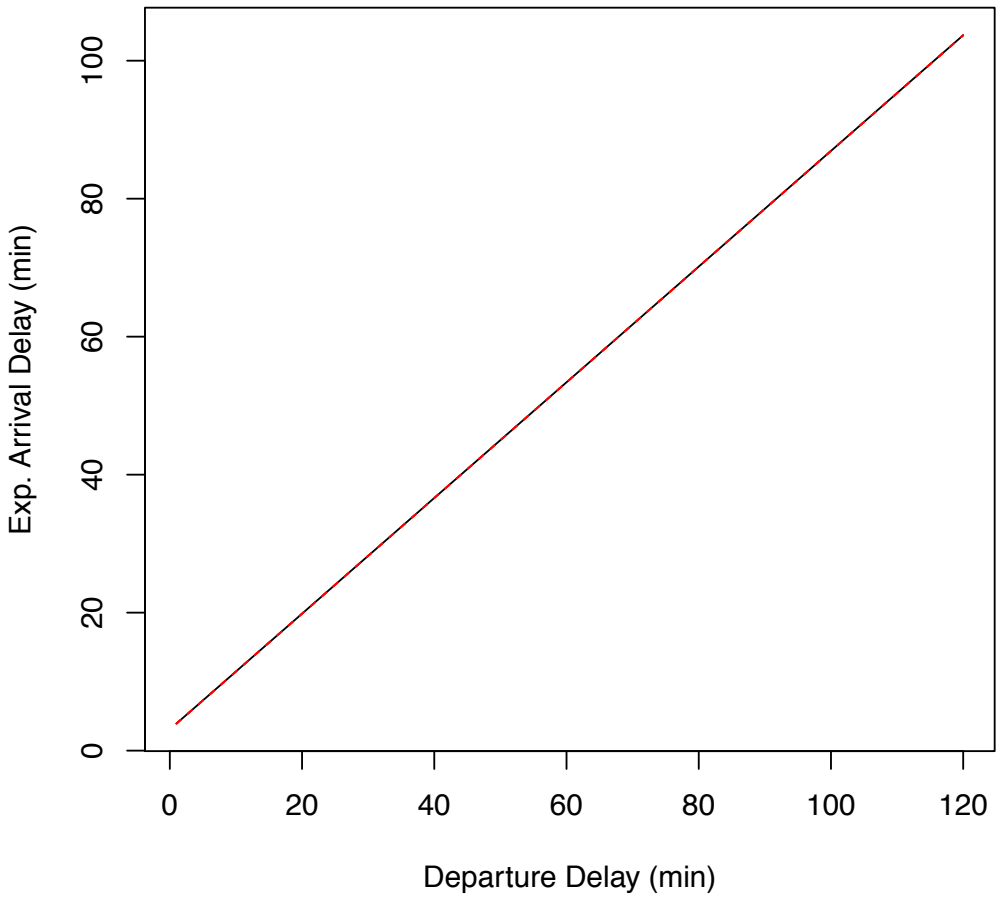
```
# fit big linear regression model (using big.matrix interface)
> linmod=biglm.big.matrix(ArrDelay~DepDelay,data=flights)
> linsum=summary(linmod)
> linsum
Large data regression model: biglm(formula = formula, data = data, ...)
Sample size =  11157390
              Coef    (95%    CI)     SE p
(Intercept) 3.0798 3.0683 3.0913 0.0058 0
DepDelay    0.8384 0.8382 0.8387 0.0001 0
> linsum$rsq
[1] 0.7456648
```

$$\widehat{\texttt{ArrDelay}} = 3.0798 + 0.8384(\texttt{DepDelay})$$

$$R^2 = 0.7456648$$

Plot Full SLR Results

Plot regression line with 95% pointwise confidence interval:



R Code for Full SLR Plot

```
# create prediction function
> newdata=data.frame(DepDelay=seq(1,120,length=200),ArrDelay=rep(0,200))
> linpred=predict(linmod,newdata,se.fit=TRUE,make.function=TRUE)
> yhat=linpred[[1]](newdata$DepDelay)
> yhatse=sqrt(diag(linpred[[2]](newdata$DepDelay)))
```

```
# plot regression line with pointwise confidence intervals
> x11(width=6,height=6)
> plot(newdata$DepDelay,yhat,type="l",
+      xlab="Departure Delay (min)",
+      ylab="Exp. Arrival Delay (min)")
> lines(newdata$DepDelay,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$DepDelay,yhat-2*yhatse,lty=2,col="red")
```

SLR of Random Subset of Flights Data

Predict `ArrDelay` from `DepDelay` using 10^6 observations:

```
 # get subset of data
> set.seed(123)
> subidx=sample.int(nrow(flights),10^6)
> flightsub=as.data.frame(flights[subidx,])


 # fit big linear regression model (using biglm)
> linmods=biglm(ArrDelay~DepDelay,data=flightsub)
> linsums=summary(linmods)


 # compare solutions
> linsum
Large data regression model: biglm(formula = formula, data = data, ...)
Sample size =  11157390
             Coef    (95%    CI)     SE p
(Intercept) 3.0798 3.0683 3.0913 0.0058 0
DepDelay    0.8384 0.8382 0.8387 0.0001 0

> linsums
Large data regression model: biglm(ArrDelay ~ DepDelay, data = flightsub)
Sample size =  1000000
             Coef    (95%    CI)     SE p
(Intercept) 3.0852 3.0468 3.1236 0.0192 0
DepDelay    0.8378 0.8369 0.8388 0.0005 0
```

Multiple Linear Regression (MLR) Model

The multiple linear regression model has the form

$$y_i = \mu + \sum_{j=1}^{p} \beta_j x_{ij} + \epsilon_i \qquad \text{or} \qquad \mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

for $i \in \{1, \ldots, n\}$ where

- $\mathbf{y} = (y_1, \ldots, y_n)'$ with $y_i \in \mathbb{R}$
- $\mu \in \mathbb{R}$ is the regression intercept
- $\beta_j \in \mathbb{R}$ is the j-th predictor's slope and $\boldsymbol{\beta} = (\mu, \beta_1, \ldots, \beta_p)'$
- $x_{ij} \in \mathbb{R}$ is the j-th predictor for the i-th observation
- $\mathbf{X} = [\mathbf{1}_n, \mathbf{x}_1, \ldots, \mathbf{x}_p]$ is the design matrix with $\mathbf{x}_j = (x_{1j}, \ldots, x_{nj})'$
- $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)'$ with $\epsilon_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$

Ordinary Least-Squares solution: $\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \sim \mathrm{N}(\boldsymbol{\beta}, \sigma^2(\mathbf{X}'\mathbf{X})^{-1})$

MLR of Full Flights Data

Predict `ArrDelay` from `Year`, `Month`, `DepHour`, and `DepDelay`:

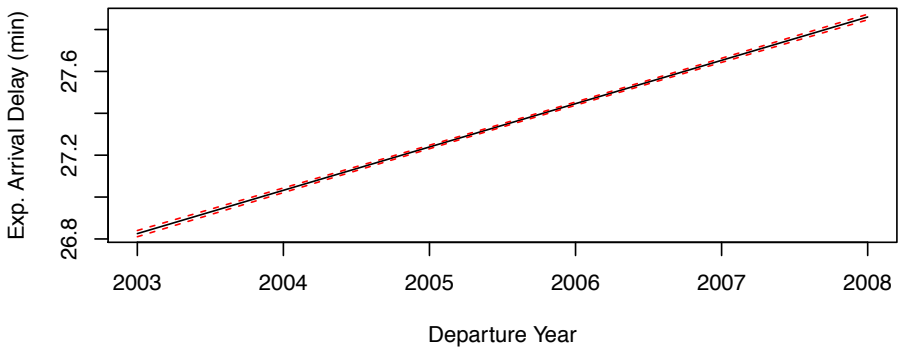
```
# fit big linear regression model (using big.matrix interface)
> mlrmod=biglm.big.matrix(ArrDelay~Year+Month+DepHour+DepDelay,data=flights)
> mlrsum=summary(mlrmod)
> mlrsum
Large data regression model: biglm(formula = formula, data = data, ...)
Sample size =  11157390
                Coef      (95%       CI)     SE p
(Intercept) -417.8778 -427.3628 -408.3928 4.7425 0
Year           0.2069    0.2022    0.2116 0.0024 0
Month          0.0172    0.0150    0.0194 0.0011 0
DepHour        0.4260    0.4242    0.4278 0.0009 0
DepDelay       0.8254    0.8251    0.8257 0.0001 0
> mlrsum$rsq
[1] 0.750886
```

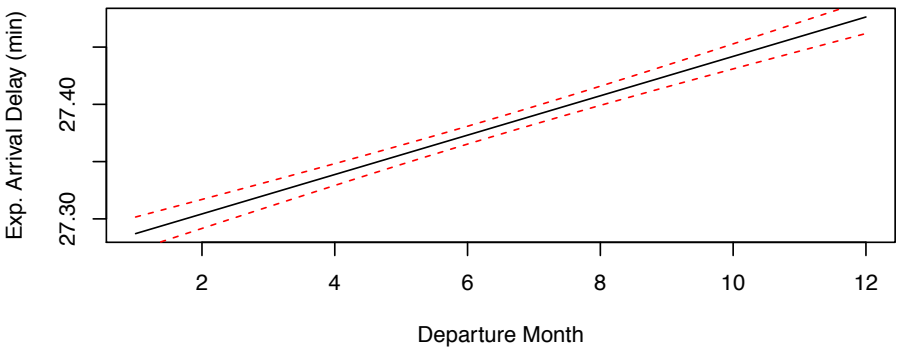
$$\widehat{\texttt{ArrDelay}} = -417.8778 + 0.2069(\texttt{Year}) + 0.0172(\texttt{Month}) + 0.4260(\texttt{DepHour}) + 0.8254(\texttt{DepDelay})$$

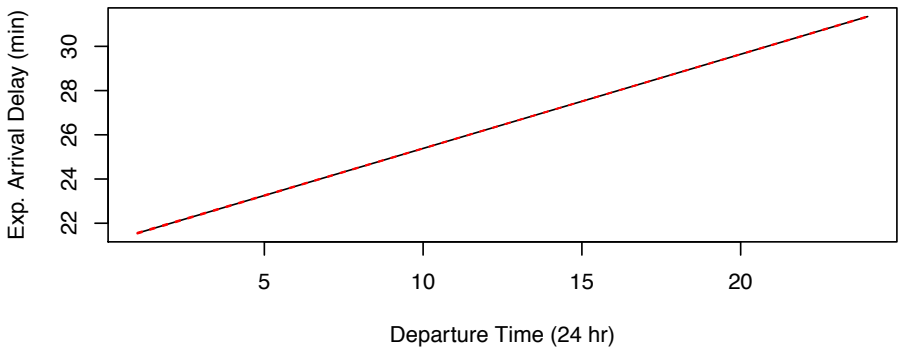
$R^2 = 0.750886$

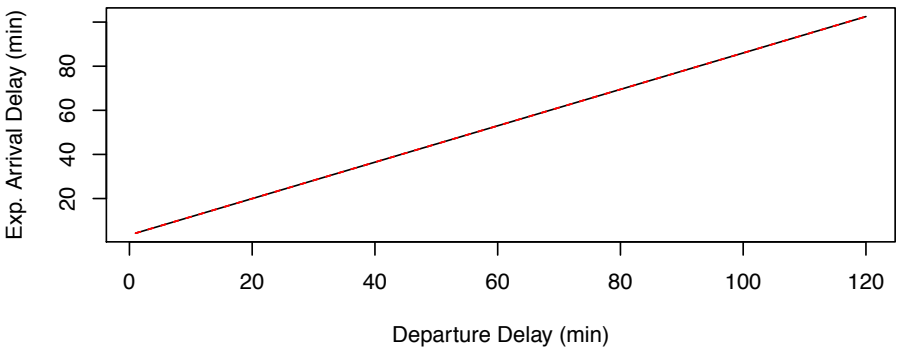
Plot Full MLR Results

Plot regression lines (with 95% CIs) at average covariate values:









R Code for Full MLR Plot

```
# create new data and prediction function
> newdata=data.frame(Year=seq(2003,2008,length=200),Month=seq(1,12,length=200),
+                    DepHour=seq(1,24,length=200),DepDelay=seq(1,120,length=200),ArrDelay=0)
> mlrpred=predict(mlrmod,newdata,se.fit=TRUE,make.function=TRUE)
> mfs=apply(flights,2,mean)      # get variable means
# plot line and 95% pointwise CI for Year
> yhat=mlrpred[[1]](cbind(newdata[,1],mfs[2],mfs[3],mfs[4]))
> yhatse=sqrt(diag(mlrpred[[2]](cbind(newdata[,1],mfs[2],mfs[3],mfs[4]))))
> plot(newdata$Year,yhat,type="l",xlab="Departure Year",ylab="Exp. Arrival Delay (min)")
> lines(newdata$Year,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$Year,yhat-2*yhatse,lty=2,col="red")
# plot line and 95% pointwise CI for Month
> yhat=mlrpred[[1]](cbind(mfs[1],newdata[,2],mfs[3],mfs[4]))
> yhatse=sqrt(diag(mlrpred[[2]](cbind(mfs[1],newdata[,2],mfs[3],mfs[4]))))
> plot(newdata$Month,yhat,type="l",xlab="Departure Month",ylab="Exp. Arrival Delay (min)")
> lines(newdata$Month,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$Month,yhat-2*yhatse,lty=2,col="red")
# plot line and 95% pointwise CI for DepHour
> yhat=mlrpred[[1]](cbind(mfs[1],mfs[2],newdata[,3],mfs[4]))
> yhatse=sqrt(diag(mlrpred[[2]](cbind(mfs[1],mfs[2],newdata[,3],mfs[4]))))
> plot(newdata$DepHour,yhat,type="l",xlab="Departure Time (24 hr)",ylab="Exp. Arrival Delay (min)")
> lines(newdata$DepHour,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$DepHour,yhat-2*yhatse,lty=2,col="red")
# plot line and 95% pointwise CI for DepDelay
> yhat=mlrpred[[1]](cbind(mfs[1],mfs[2],mfs[3],newdata[,4]))
> yhatse=sqrt(diag(mlrpred[[2]](cbind(mfs[1],mfs[2],mfs[3],newdata[,4]))))
> plot(newdata$DepDelay,yhat,type="l",xlab="Departure Delay (min)",ylab="Exp. Arrival Delay (min)")
> lines(newdata$DepDelay,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$DepDelay,yhat-2*yhatse,lty=2,col="red")
```

MLR of Random Subset of Flights Data

Predict `ArrDelay` from other variables using 10^6 observations:

```
 # fit big linear regression model (using biglm)
> mlrmods=biglm(ArrDelay~Year+Month+DepHour+DepDelay,data=flightsub)
> mlrsums=summary(mlrmods)


 # compare solutions
> mlrsum
Large data regression model: biglm(formula = formula, data = data, ...)
Sample size =  11157390
                 Coef      (95%       CI)     SE p
(Intercept) -417.8778 -427.3628 -408.3928 4.7425 0
Year           0.2069    0.2022    0.2116 0.0024 0
Month          0.0172    0.0150    0.0194 0.0011 0
DepHour        0.4260    0.4242    0.4278 0.0009 0
DepDelay       0.8254    0.8251    0.8257 0.0001 0

> mlrsums
Large data regression model: biglm(ArrDelay ~ Year + Month + DepHour + DepDelay, data = flightsub)
Sample size =  1000000
                 Coef      (95%       CI)      SE p
(Intercept) -400.7051 -432.3175 -369.0928 15.8062 0
Year           0.1983    0.1826    0.2141  0.0079 0
Month          0.0211    0.0137    0.0285  0.0037 0
DepHour        0.4245    0.4186    0.4305  0.0030 0
DepDelay       0.8249    0.8239    0.8259  0.0005 0
```

Nonparametric Regression (NPR) Model

The nonparametric regression model has the form

$$y_i = \eta(\mathbf{x}_i) + \epsilon_i$$

where

- $y_i \in \mathbb{R}$ is the real-valued response for the i-th observation
- η is an unknown smooth function
- $\mathbf{x}_i \in \mathcal{X}$ is the predictor vector for the i-th observation
- $\epsilon_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$ is Gaussian measurement error

Smoothing Spline Approach to NPR

Smoothing spline approach estimates η in tensor product reproducing kernel Hilbert space (see Gu, 2013; Helwig, 2013; Wahba, 1990).

Find the η that minimizes the penalized least-squares functional

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \eta(\mathbf{x}_i))^2 + \lambda J(\eta)$$

where

- $\lambda \geq 0$ is global smoothing parameter
- J is nonnegative penalty functional quantifying roughness of η

Smoothing Spline Estimation

Optimal η can be approximated as

$$\eta_{\boldsymbol{\lambda}}(\mathbf{x}) = \sum_{v=1}^{u} d_v \phi_v(\mathbf{x}) + \sum_{t=1}^{q} c_t \rho_c(\mathbf{x}, \check{\mathbf{x}}_t)$$

where

- $\{\phi_v\}_{v=1}^{u}$ are basis functions spanning null space
- ρ_c denotes the reproducing kernel of contrast space (if $\mathbf{x}$ is multidimensional, θ_k parameters are buried in ρ_c)
- $\{\check{\mathbf{x}}_t\}_{t=1}^{q} \subset \{\mathbf{x}_i\}_{i=1}^{n}$ are the selected spline knots
- $\mathbf{d} = \{d_v\}_{u\times 1}$ and $\mathbf{c} = \{c_t\}_{q\times 1}$ are unknown coefficients
- $\boldsymbol{\lambda} = (\lambda, \theta_1, \ldots, \theta_k)$ are unknown smoothing parameters

Given fixed $\boldsymbol{\lambda}$, there is a closed form solution for optimal coefficients (see Kim & Gu, 2004; Helwig, 2013; Helwig & Ma, in press).

- Select $\boldsymbol{\lambda}$ by minimizing GCV score (Craven & Wahba, 1979)

Some Notes on `bigsplines` Package

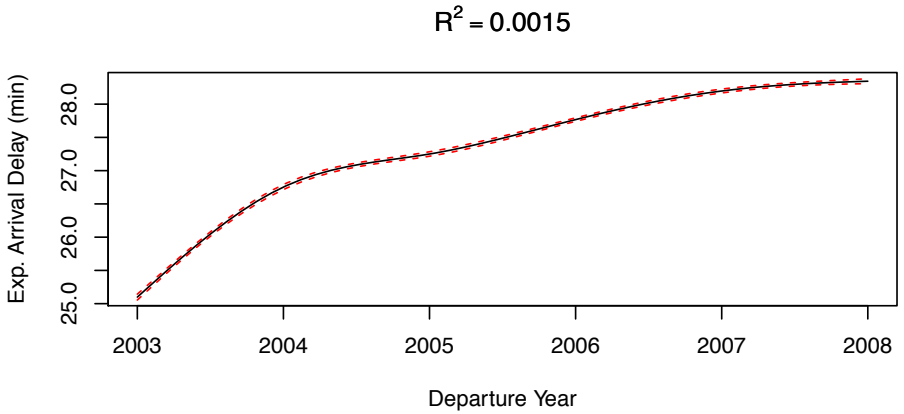
Currently, four primary functions in `bigsplines` package:

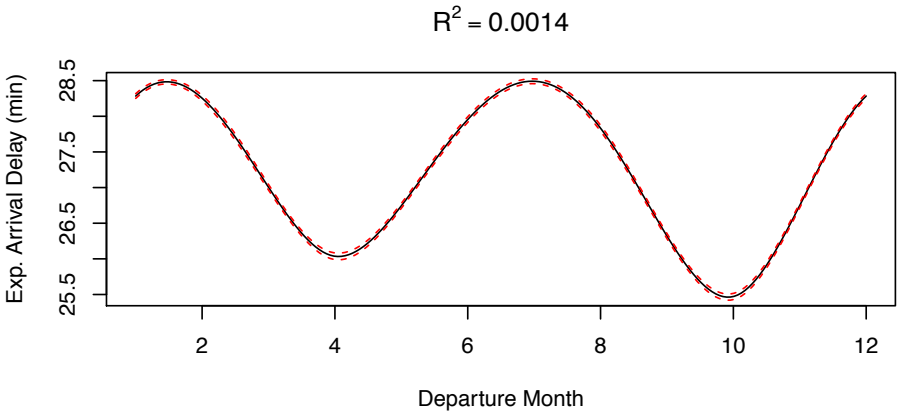
- `bigspline`: Unidimensional cubic smoothing splines
  - Unconstrained or periodic
- `bigtps`: Cubic thin-plate splines
  - Supports 1-, 2-, or 3-dimensional predictors
- `bigssa`: Smoothing Spline Anova (tensor product splines)
  - Supports cubic, cubic periodic, thin-plate, and nominal splines
- `bigssp`: Smoothing Splines with Parametric effects
  - More general than `bigssa`; supports parametric predictors too

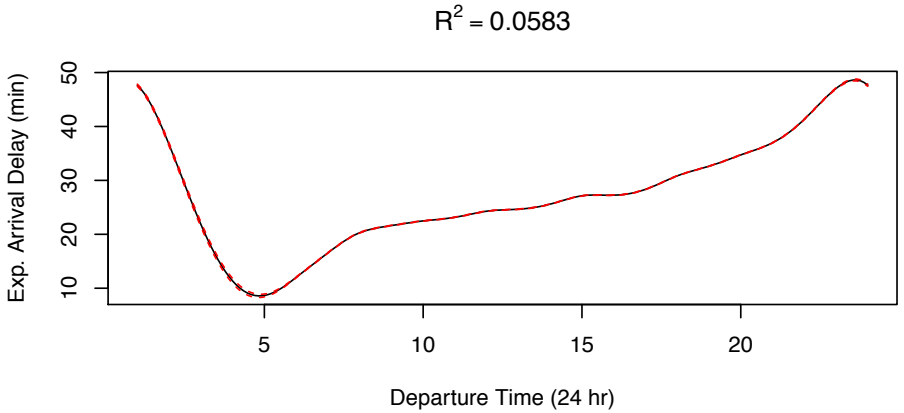
NPR of Full Flights Data

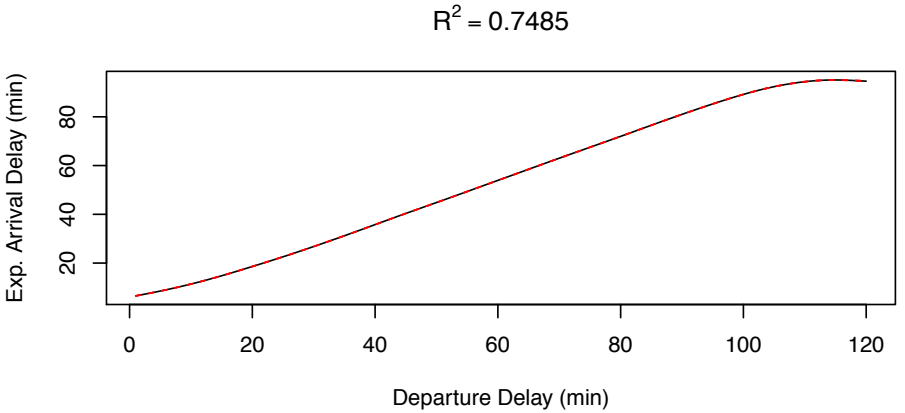
Four separate models using `bigspline` function:

$y_i = \eta(x_i) + \epsilon_i$



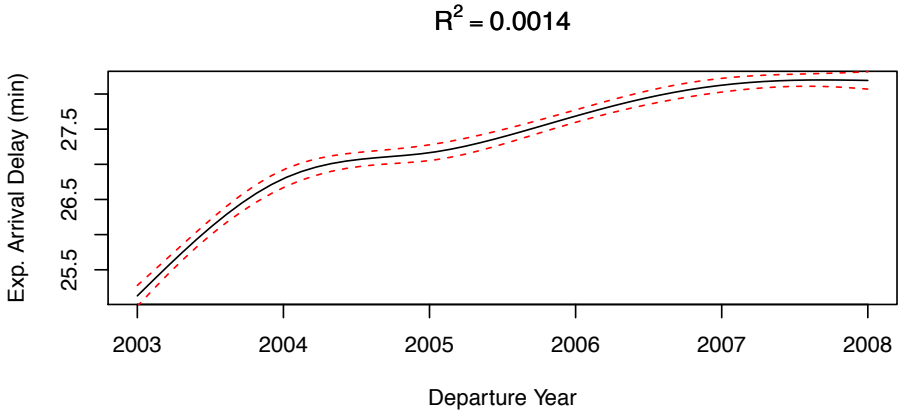


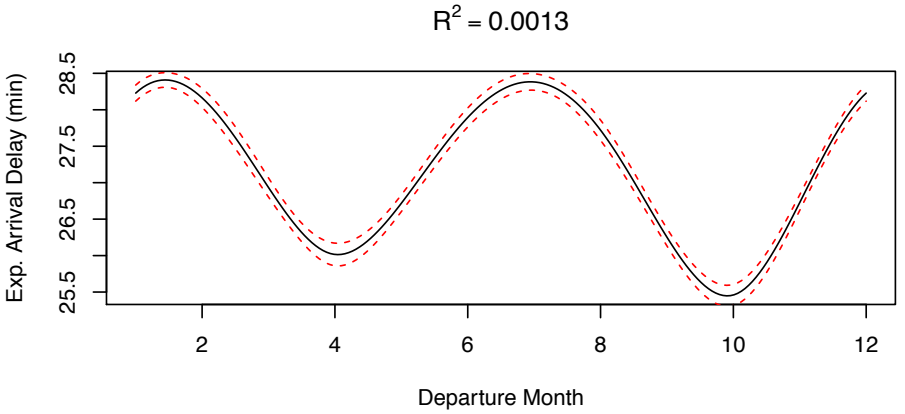


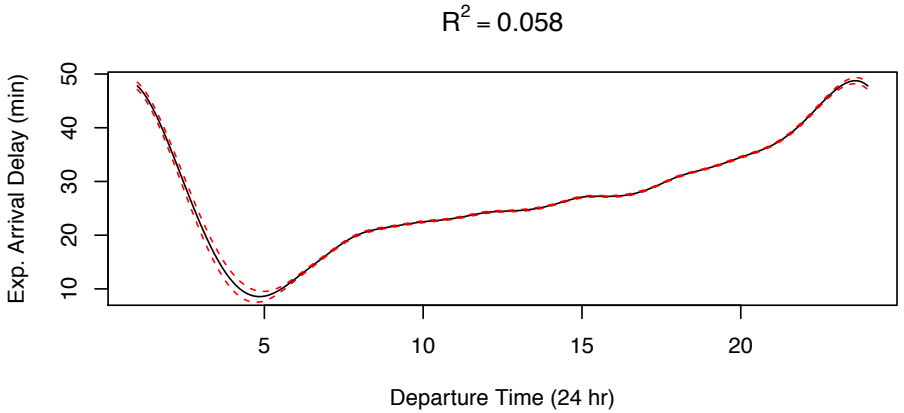


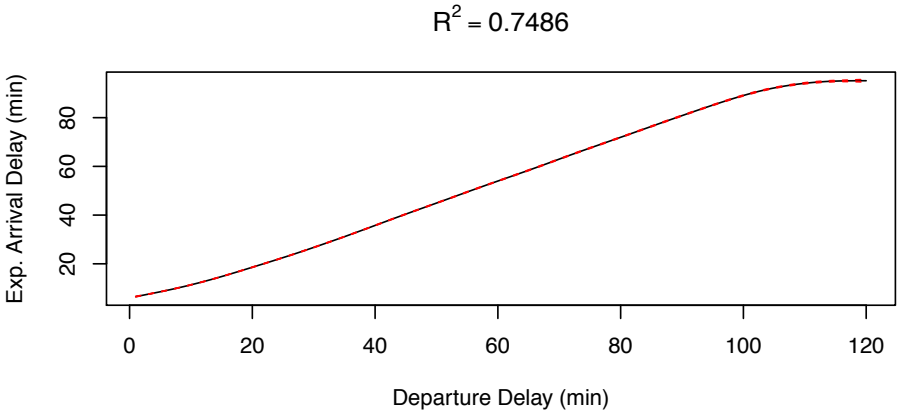
NPR of Random Subset of Flights Data

Same four models using random subset of 10^6 observations:









R Code for Full NPR Plots

```
# Year vs. ArrDelay using cubic spline with 4 knots
> smod=bigspline(flights[,1],flights[,5],nknots=4)
> newdata=data.frame(Year=seq(2003,2008,length=50))
> spred=predict(smod,newdata,se.fit=TRUE)
> yhat=spred[[1]]
> yhatse=spred[[2]]
> plot(newdata$Year,yhat,type="l",xlab="Departure Year",
+      ylab="Exp. Arrival Delay (min)",main=bquote(R^2==.(round(smod$info[2],4))))
> lines(newdata$Year,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$Year,yhat-2*yhatse,lty=2,col="red")


# Month vs. ArrDelay using periodic cubic spline with 6 knots
> smod=bigspline(flights[,2],flights[,5],type="per",nknots=6)
> newdata=data.frame(Month=seq(1,12,length=200))
> spred=predict(smod,newdata,se.fit=TRUE)
> yhat=spred[[1]]
> yhatse=spred[[2]]
> plot(newdata$Month,yhat,type="l",xlab="Departure Month",
+      ylab="Exp. Arrival Delay (min)",main=bquote(R^2==.(round(smod$info[2],4))))
> lines(newdata$Month,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$Month,yhat-2*yhatse,lty=2,col="red")
```

R Code for Full NPR Plots (continued)

```
# DepHour vs. ArrDelay using periodic cubic spline with 12 knots
> smod=bigspline(flights[,3],flights[,5],type="per",nknots=12)
> newdata=data.frame(DepHour=seq(1,24,length=200))
> spred=predict(smod,newdata,se.fit=TRUE)
> yhat=spred[[1]]
> yhatse=spred[[2]]
> plot(newdata$DepHour,yhat,type="l",xlab="Departure Time (24 hr)",
+      ylab="Exp. Arrival Delay (min)",main=bquote(R^2==.(round(smod$info[2],4))))
> lines(newdata$DepHour,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$DepHour,yhat-2*yhatse,lty=2,col="red")


# DepDelay vs. ArrDelay using cubic spline with 10 knots
> smod=bigspline(flights[,4],flights[,5],type="cub",nknots=10)
> newdata=data.frame(DepDelay=seq(1,120,length=200))
> spred=predict(smod,newdata,se.fit=TRUE)
> yhat=spred[[1]]
> yhatse=spred[[2]]
> plot(newdata$DepDelay,yhat,type="l",xlab="Departure Delay (min)",
+      ylab="Exp. Arrival Delay (min)",main=bquote(R^2==.(round(smod$info[2],4))))
> lines(newdata$DepDelay,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$DepDelay,yhat-2*yhatse,lty=2,col="red")
```

Semiparametric Regression (SPR) Model

The semiparametric regression model has the form

$$y_i = \mu + \sum_{j=1}^{p} \beta_j x_{ij} + \eta(\mathbf{z}_i) + \epsilon_i$$

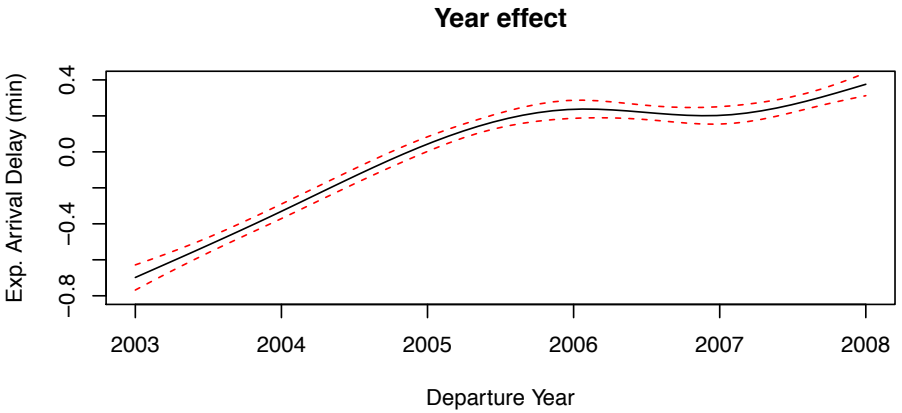
for $i \in \{1, \ldots, n\}$ where

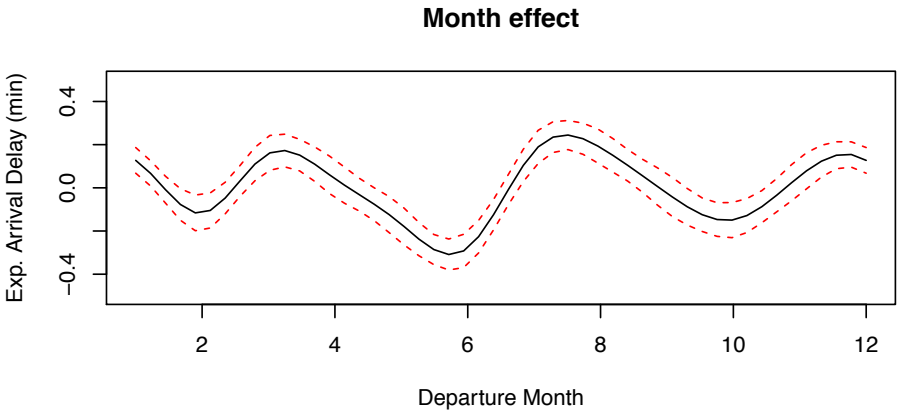
- $y_i \in \mathbb{R}$ is the real-valued response for the i-th observation
- $\mu \in \mathbb{R}$ is the regression intercept
- $\beta_j \in \mathbb{R}$ is the j-th predictor's regression slope
- x_{ij} is j-th parametric predictor for i-th observation
- η is an unknown smooth function
- $\mathbf{z}_i \in \mathcal{Z}$ is nonparametric predictor vector for i-th observation
- $\epsilon_i \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma^2)$ is Gaussian measurement error

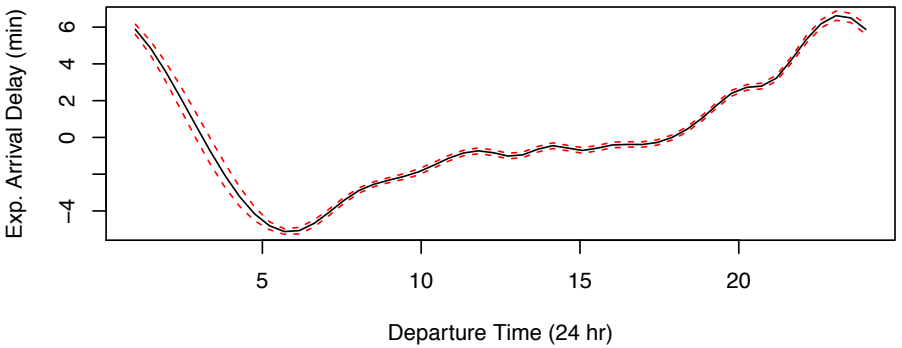
SPR of Random Subset of Flights Data

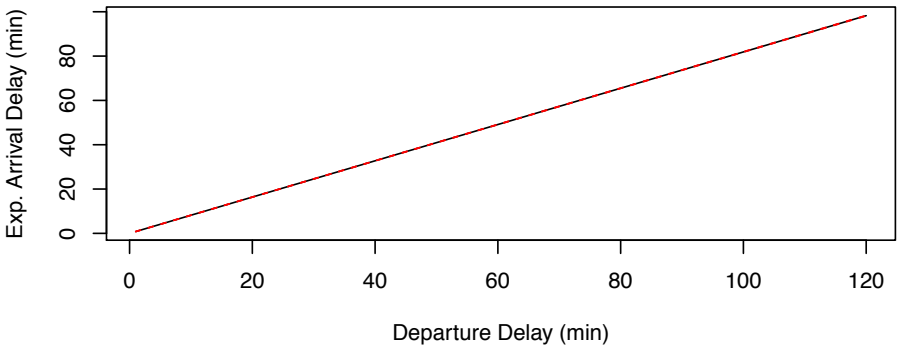
SPR via `bigssp` using random subset of 10^6 observations:

$$y_i = \mu + \eta_1(\texttt{Year}_i) + \eta_2(\texttt{Month}_i) + \eta_3(\texttt{DepHour}_i) + \beta\texttt{DepDelay}_i + \epsilon_i$$









R Code for Fitting SPR Model

Cubic spline for `Year`, periodic cubic splines for `Month` and `DepHour`, and parametric effect for `DepDelay`.

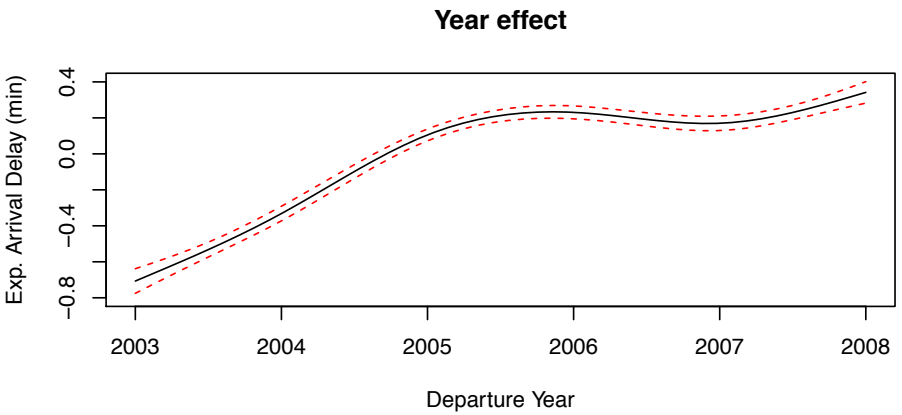
```
# fit semiparametric model
> smod=bigssp(ArrDelay~Year+Month+DepHour+DepDelay,data=flightsub,
+             type=list(Year="cub",Month="per",DepHour="per",DepDelay="prm"),nknots=30,
+             rparm=list(Year=0.01,Month=0.01,DepHour=0.01,DepDelay=5),skip.iter=FALSE)
> smod

Predictor Types:
 Year Month DepHour DepDelay
  cub   per     per      prm

Fit Statistics:
         gcv          rsq          aic          bic
1.675821e+02 7.471554e-01 7.959350e+06 7.959704e+06

Algorithm Converged:
TRUE
```

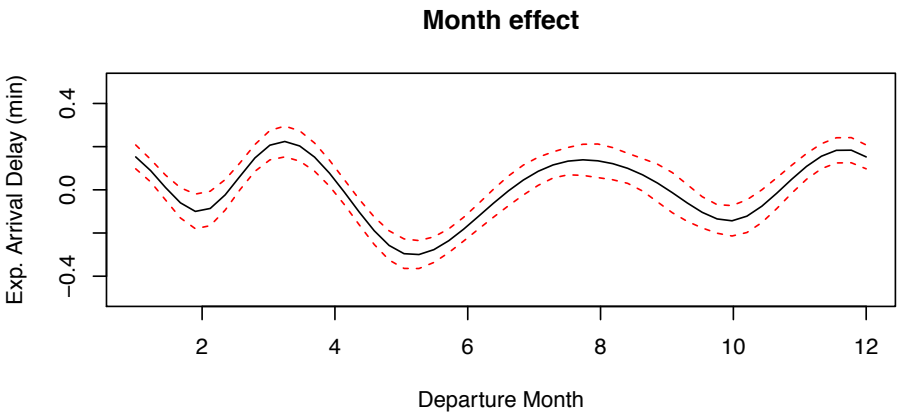
See Helwig (2013) and Helwig and Ma (in prep) for discussion of rounding parameters.

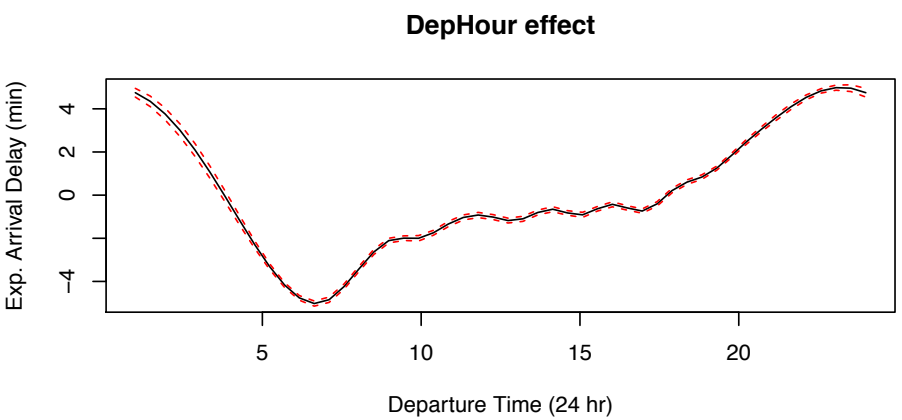
NPR of Random Subset of Flights Data

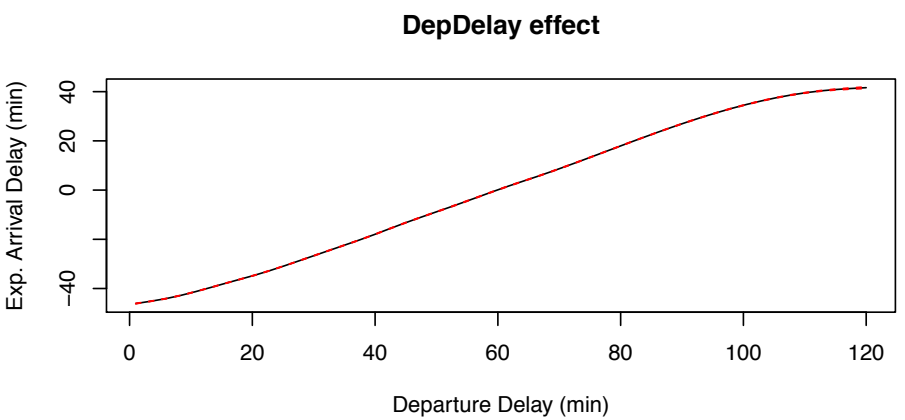
NPR via `bigssp` using random subset of 10^6 observations:

$$y_i = \mu + \eta_1(\texttt{Year}_i) + \eta_2(\texttt{Month}_i) + \eta_3(\texttt{DepHour}_i) + \eta_4(\texttt{DepDelay}_i) + \epsilon_i$$









R Code for Fitting NPR Model

Cubic spline for `Year`, periodic cubic splines for `Month` and `DepHour`, and cubic spline for `DepDelay`.

```
# fit nonparametric model
> smod=bigssp(ArrDelay~Year+Month+DepHour+DepDelay,data=flightsub,
+             type=list(Year="cub",Month="per",DepHour="per",DepDelay="cub"),nknots=30,
+             rparm=list(Year=0.02,Month=0.01,DepHour=0.01,DepDelay=0.02),skip.iter=FALSE)
> smod

Predictor Types:
 Year Month DepHour DepDelay
  cub   per     per      cub

Fit Statistics:
         gcv          rsq          aic          bic
1.625505e+02 7.547485e-01 7.928866e+06 7.929255e+06

Algorithm Converged:
TRUE
```

R Code for SPR/NPR Plots

```
# plot line and 95% Bayesian CI for Year
> newdata=data.frame(Year=seq(2003,2008,length=50))
> spred=predict(smod,newdata,se.fit=TRUE,include="Year")
> yhat=spred[[1]]
> yhatse=spred[[2]]
> plot(newdata$Year,yhat,type="l",ylim=c(-.8,.4),xlab="Departure Year",
+      ylab="Exp. Arrival Delay (min)",main="Year effect")
> lines(newdata$Year,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$Year,yhat-2*yhatse,lty=2,col="red")
```

```
# plot line and 95% Bayesian CI for Month
> newdata=data.frame(Month=seq(1,12,length=50))
> spred=predict(smod,newdata,se.fit=TRUE,include="Month")
> yhat=spred[[1]]
> yhatse=spred[[2]]
> plot(newdata$Month,yhat,type="l",ylim=c(-.5,.5),xlab="Departure Month",
+      ylab="Exp. Arrival Delay (min)",main="Month effect")
> lines(newdata$Month,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$Month,yhat-2*yhatse,lty=2,col="red")
```

R Code for SPR/NPR Plots (continued)

```
# plot line and 95% Bayesian CI for DepHour
> newdata=data.frame(DepHour=seq(1,24,length=50))
> spred=predict(smod,newdata,se.fit=TRUE,include="DepHour")
> yhat=spred[[1]]
> yhatse=spred[[2]]
> plot(newdata$DepHour,yhat,type="l",xlab="Departure Time (24 hr)",
+      ylab="Exp. Arrival Delay (min)",main="DepHour effect")
> lines(newdata$DepHour,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$DepHour,yhat-2*yhatse,lty=2,col="red")


# plot line and 95% Bayesian CI for DepDelay
> newdata=data.frame(DepDelay=seq(1,120,length=50))
> spred=predict(smod,newdata,se.fit=TRUE,include="DepDelay")
> yhat=spred[[1]]
> yhatse=spred[[2]]
> plot(newdata$DepDelay,yhat,type="l",xlab="Departure Delay (min)",
+      ylab="Exp. Arrival Delay (min)",main="DepDelay effect")
> lines(newdata$DepDelay,yhat+2*yhatse,lty=2,col="red")
> lines(newdata$DepDelay,yhat-2*yhatse,lty=2,col="red")
```

Things to Come...

Methodological plans for the `bigsplines` package include:

- `bigssg`: Smoothing Spline Generalized regression (extension of `bigssp` for non-Gaussian data)
- `bigssd`: Smoothing Spline Density estimation
- Significance testing for non/parametric effects
- Support for random effects

Computational plans for the `bigsplines` package include:

- Further algorithm improvements (can be 10x faster)
- Support for parallel computation
- Support for `big.matrix` objects

Contact Information

Summer contact:

- Nathaniel E. Helwig, *Visiting Assistant Professor*
- Department of Statistics
  University of Illinois at Urbana-Champaign
- Email: nhelwig2@illinois.edu

Future contact:

- Nathaniel E. Helwig, *Assistant Professor*
- School of Statistics and Department of Psychology
  University of Minnesota (Twin Cities)
- Email: helwig@umn.edu

References

Craven, P. and G. Wahba (1979). Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik 31*, 377–403.

Gu, C. (2013). *Smoothing Spline ANOVA Models* (Second ed.). New York: Springer-Verlag.

Helwig, N. E. (2013, May). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Ph.D. thesis, University of Illinois at Urbana-Champaign.

Helwig, N. E. (in prep.). Nonparametric analysis of group differences in multivariate time series data: Application to electroencephalography data analysis.

Helwig, N. E. and P. Ma (in prep.). Nonparametric Gaussian regression for ultra large samples: Scalable computation via rounding parameters.

Helwig, N. E. and P. Ma (in press). Fast and stable multiple smoothing parameter selection in smoothing spline analysis of variance models with large samples. *Journal of Computational and Graphical Statistics*.

Helwig, N. E., P. Ma, and S. Wang (in prep.). Nonparametric analysis of spatiotemporal trends in social media data.

Kim, Y.-J. and C. Gu (2004). Smoothing spline gaussian regression: More scalable computation via efficient approximation. *Journal of the Royal Statistical Society, Series B 66*, 337–356.

Schmidberger, M., M. Morgan, D. Eddelbuettel, H. Yu, L. Tierney, and U. Mansmann (2009). State of the art in parallel computing with R. *Journal of Statistical Software 31*, 1–27.

Wahba, G. (1990). *Spline models for observational data*. Philadelphia: Society for Industrial and Applied Mathematics.