A Monte Carlo Investigation of Locally Weighted Regression

Aaron Swoboda and Sam Carruthers

September 28, 2012

This document writes up the results of the recent run of `uberScript.R`. It contains the following code:

```r
# set our simulation parameters
Replications = 100
sample.size = c(50, 100, 200, 500, 1000)
error.sd = c(2, 4, 6)
B1.spatial.var = c(0, .1, .2, .3)
B2.spatial.var = c(0, .1, .2, .3)

# now march through the different parameter combinations running the simulations

for( i in 1:meta.sim.num) {
  start = Sys.time()
  simRepOut = simulationReplicator(Replications, sim.parameters[i, ], MC = TRUE)
  simOut = simRepReorganizer(simRepOut)

  R2Output[as.character(sim.parameters[i, "sample.size"]),
           as.character(sim.parameters[i, "error.sd"]),
           as.character(sim.parameters[i, "B1.spatial.var"]),
           as.character(sim.parameters[i, "B2.spatial.var"]), , ] = simOut[[1]]

  MetricOutput[as.character(sim.parameters[i, "sample.size"]),
               as.character(sim.parameters[i, "error.sd"]),
               as.character(sim.parameters[i, "B1.spatial.var"]),
               as.character(sim.parameters[i, "B2.spatial.var"]), , , ] = simOut[[2]]
  end = Sys.time()

  print(paste("For loop", i,"of", meta.sim.num))
  print(round(difftime(end, start, units = "m"), 2))
  save(R2Output, MetricOutput, file = "SpecificationSims/uberScriptOutput.RData")
}
```

I'm not going to run that code here (it took almost a month to run on the R Server), but let's load up the results and start to look at them. Or at least come up with some questions to ask of the data and a plan for the future.

```
load("../Data/uberScriptOutput20120919.RData")
dimnames(MetricOutput)

## $ss
## [1] "50"   "100"  "200"  "500"  "1000"
##
## $error.sd
## [1] "2" "4" "6"
##
## $B1sv
## [1] "0"   "0.1" "0.2" "0.3"
##
## $B2sv
## [1] "0"   "0.1" "0.2" "0.3"
##
## $simNum
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"
##  [12] "12"  "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"
##  [23] "23"  "24"  "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"
##  [34] "34"  "35"  "36"  "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"
##  [45] "45"  "46"  "47"  "48"  "49"  "50"  "51"  "52"  "53"  "54"  "55"
##  [56] "56"  "57"  "58"  "59"  "60"  "61"  "62"  "63"  "64"  "65"  "66"
##  [67] "67"  "68"  "69"  "70"  "71"  "72"  "73"  "74"  "75"  "76"  "77"
##  [78] "78"  "79"  "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"  "88"
##  [89] "89"  "90"  "91"  "92"  "93"  "94"  "95"  "96"  "97"  "98"  "99"
## [100] "100"
##
## $optimized
##  [1] "AICc"     "corB0"    "corB1"    "corB2"    "CV"       "GCV"
##  [7] "R2"       "RMSE.B0"  "RMSE.B1"  "RMSE.B2"  "SCV"      "ttest%B0"
## [13] "ttest%B1" "ttest%B2"
##
## $metric
##  [1] "bandwidths" "B0.cor"     "B1.cor"     "B2.cor"     "B0.RMSE"
##  [6] "B1.RMSE"    "B2.RMSE"    "B0.t.perc"  "B1.t.perc"  "B2.t.perc"
## [11] "GCV"        "SCV"        "CV"         "AICc"       "R2"
##

dimnames(R2Output)

## $ss
## [1] "50"   "100"  "200"  "500"  "1000"
##
## $error.sd
## [1] "2" "4" "6"
##
## $B1sv
## [1] "0"   "0.1" "0.2" "0.3"
##
## $B2sv
## [1] "0"   "0.1" "0.2" "0.3"
##
## $simNum
##   [1] "1"   "2"   "3"   "4"   "5"   "6"   "7"   "8"   "9"   "10"  "11"
##  [12] "12"  "13"  "14"  "15"  "16"  "17"  "18"  "19"  "20"  "21"  "22"
##  [23] "23"  "24"  "25"  "26"  "27"  "28"  "29"  "30"  "31"  "32"  "33"
##  [34] "34"  "35"  "36"  "37"  "38"  "39"  "40"  "41"  "42"  "43"  "44"
##  [45] "45"  "46"  "47"  "48"  "49"  "50"  "51"  "52"  "53"  "54"  "55"
##  [56] "56"  "57"  "58"  "59"  "60"  "61"  "62"  "63"  "64"  "65"  "66"
##  [67] "67"  "68"  "69"  "70"  "71"  "72"  "73"  "74"  "75"  "76"  "77"
##  [78] "78"  "79"  "80"  "81"  "82"  "83"  "84"  "85"  "86"  "87"  "88"
##  [89] "89"  "90"  "91"  "92"  "93"  "94"  "95"  "96"  "97"  "98"  "99"
## [100] "100"
##
## $R2
## [1] "OLS" "LWR"
##
```

So, we ran some simulations, varying the sample size of the data set, the standard deviation of the error term in the model and the degree of spatial variation in the model coefficients.

Each simulation was conducted as follows:

1. Grab the simulation parameters.
2. Generate the data according to the model and parameters.
3. Choose a number of observations to include in the Locally Weighted Regression.
4. Run Locally Weighted Regression on the data using the chosen bandwidth for each observation within the dataset.
5. Calculate a number of model metrics for each bandwidth
6. Repeat previous two steps for a number of bandwidths, ranging from only 5 data points to a model approaching a global Ordinary Least Squares model (in our case, we still had declining weights based on distance, but all observations received positive weight in the regresssion).
7. Collect data on each metric when each metric is optimized. For instance, when we choose the bandwidth associated with the lowest GCV score, what are the other metric values (β RMSEs, etc.)

We kept track of the following model performance metrics, the pseudo R^2 of the model results, the correlation between the $\hat{\beta}$ and the true β, the percent of the observations for which we can reject the null hypothesis that $\hat{\beta} = \beta$, cross validation scores (leave one out, generalized, and standardized according to Paez), lastly the AIC score.

0.1 Data Generation Process

The Data Generation Process is achieved using the `DataGen` function, the code for which is given below.

```
source("../SimFunctions.R")
DataGen

## function (sample.size, error.sd, B1.spatial.var, B2.spatial.var)
## {
##     n = sample.size
##     east = runif(sample.size) * 10
##     north = runif(sample.size) * 10
##     indep.var1 = runif(sample.size) * 10
##     indep.var2 = runif(sample.size) * 10
##     trueB0 = 0
##     trueB1 = B1.spatial.var * north + 1 - 5 * B1.spatial.var
##     trueB2 = B2.spatial.var * east + 1 - 5 * B2.spatial.var
##     error = rnorm(sample.size, 0, error.sd)
##     dep.var = trueB0 + indep.var1 * trueB1 + indep.var2 * trueB2 +
##         error
##     output = data.frame(dep.var, north, east, indep.var1, indep.var2,
##         trueB0, trueB1, trueB2, error)
##     output
## }
```

The dependent variable is produced as follows:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + error \tag{1}$$

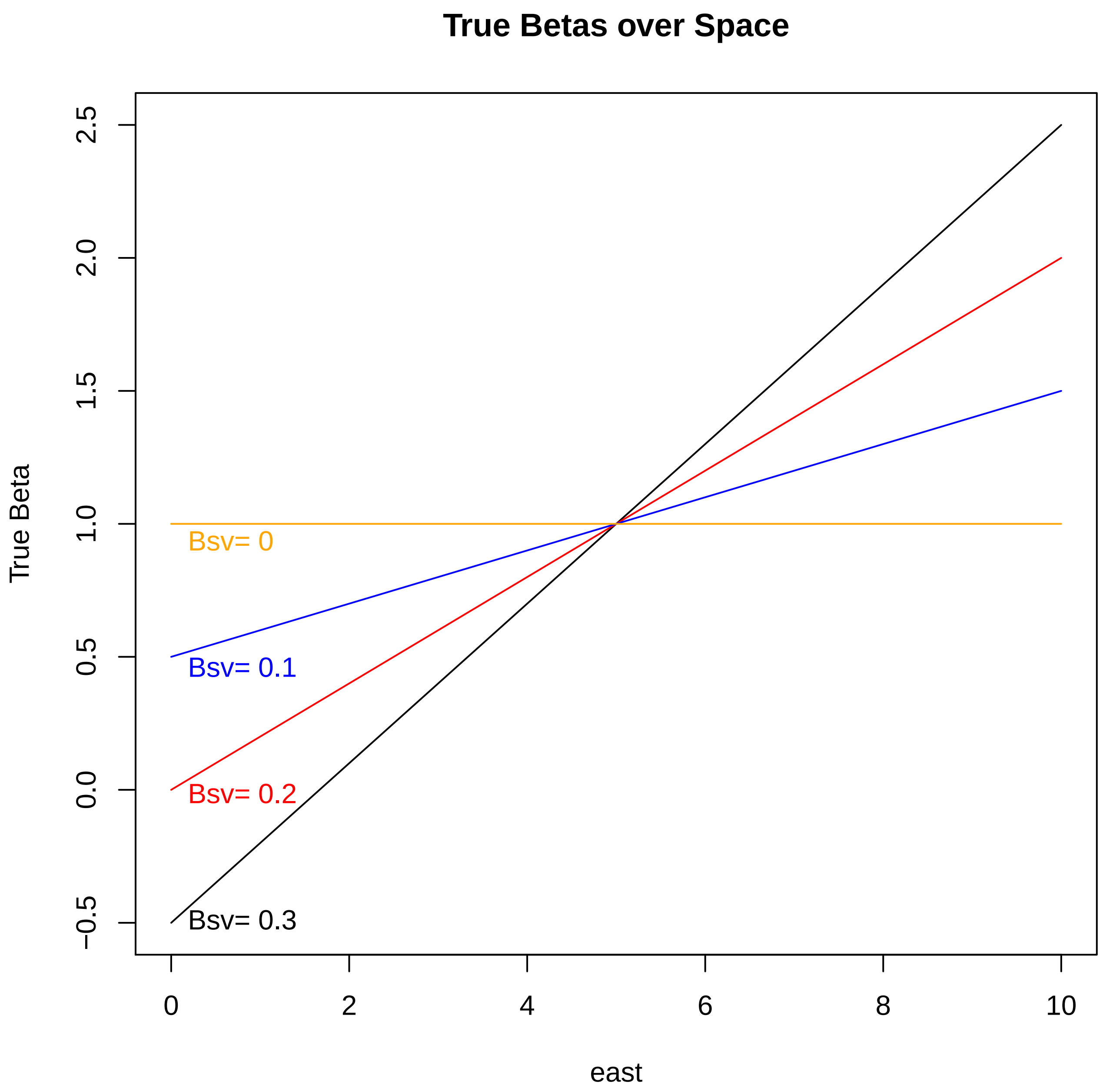
where $\beta_i = f(location)$ for i in $\{1, 2\}$ and $error \sim n(0, \sigma^2)$. The function for β_i is:

$$\beta_2 = 1 + Bsv * east - 5 * Bsv \tag{2}$$

Because our data are located on a cartesian plane $(east, north)$ where $0 < east, north < 10$, and $Bsv = \{0, 0.1, 0.2, 0.3\}$, the βs can be visualized as:

```
east = north = 0:10
BetaFunc = function(x, Bsv) {
    1 + Bsv * x - 5 * Bsv
}

plot(east, BetaFunc(east, 0.3), type = "l", xlab = "east", ylab = "True Beta",
    main = "True Betas over Space")
lines(east, BetaFunc(east, 0.2), col = "red")
lines(east, BetaFunc(east, 0.1), col = "blue")
lines(east, BetaFunc(east, 0), col = "orange")
text(rep(0, 4), seq(0.925, -0.5, length = 4), paste("Bsv=", (0:3)/10),
    pos = 4, col = c("orange", "blue", "red", "black"))
```


True Betas over Space
True Beta
east
Bsv= 0
Bsv= 0.1
Bsv= 0.2
Bsv= 0.3
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
0
2
4
6
8
10