Example 1: Standard Error of a Coefficient

Setting Up

First load the data and R functions you need into the R workspace.

```
load("PlayData_for_Examples.RData") ### load the data for the examples
source("LoadFunctions.R") #### Source the functions required to run the block bootstrap
```

Motivation

Imagine we want to know the standard error of our estimate for β- the coefficient of temperature. First fit the model.

```
set.seed(42)


fit1 = lm (response ~ temperature, data = dat)
summary(fit1)
```

```
##
## Call:
## lm(formula = response ~ temperature, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5766 -0.9687  0.0120  0.9065  5.1925
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.03474    0.06584   0.528    0.598
## temperature  1.14190    0.06975  16.372   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.455 on 498 degrees of freedom
## Multiple R-squared:  0.3499, Adjusted R-squared:  0.3486
## F-statistic:   268 on 1 and 498 DF,  p-value: < 2.2e-16
```

Getting Variogram Practical Range

Fit a variogram on the residuals- notice we can see spatial autocorrelation. Use the variogram practical range as a tuning parameter for the Lahiri method of block selection. Always check the variogram visually. Sometimes you may need to change max.dist, breaks or initial conditions to get a sensible practical range. It is hard to automate this process.

```
ini.vals <- expand.grid(seq(0,10,l=100), seq(0,1,l=100)) #inital values for variofit
variogram_block_length = select_b_length_off_variogram_envelope(x,y,resids = fit1$resid,
max.dist = 0.4, breaks=c(0,0.025,0.05,0.075,0.1,0.15,0.2,0.3,0.4,0.5,0.6,0.7,0.8,1.5),ini=ini.vals)
```

```
## variog: computing omnidirectional variogram
```

```
## variofit: covariance model used is matern
## variofit: weights used: cressie
## variofit: minimisation function used: optim
## variofit: searching for best initial value ... selected values:
##               sigmasq phi    tausq kappa
## initial.value "2.02"  "0.02" "0"   "0.5"
## status        "est"   "est"  "est" "fix"
## loss value: 95.7219934941921
```

```
variogram_block_length
```

```
## [1] 0.20739591 0.06922791
```

Coding StatFunction argument

Define BetaCoeff a function which will be the *StatFunction* argument to *BlockBootApply*. BetaCoeff gets a coefficient for temperature for inference

```
BetaCoeff = function(dat){
  fit1 = lm ("response~temperature", data=dat)
  Beta = fit1$coefficients["temperature"]
  Beta
}
```

Run Block Bootstrap

First the user should create a folder called e.g. LookupTables in their R working directory. As all the examples below use the same site coordinates they all share the same lookup tables. However if a new analysis is being done on different data (with different site co-ordinates), then a new lookup table needs to be created.

```
lookuptables.folderpathname = "LookupTables/"
```

Run Block Bootstrap, using Lahiri method of block size selection, with the variogram to select the tuning parameter. We search over 4 blocks sizes (0- i.e. IID, 0.05, 0.1, 0.2). BlockBootApply uses the size which minimises the Empirical MSE of SE(β). Use 0.2 as tuning block length as per variogram practical range (tuning block length =4).

```
Results = BlockBootApply (x = x ,y = y ,block_Ls = c(0,0.05,0.1, 0.2), Grid_space = 0.01 ,
                          dat = dat ,
                          Stat.function = BetaCoeff,  tuning_block_length = 0.2,
                          NBoot = 500, method.block.length.select = "EmpiricalMSE",
                          type="SE", lookuptables.folderpath=lookuptables.folderpathname)
```

```
## [1] "has foldername, checking for lookuptable"
## [1] "file exists"
## [1] "using lookup"
## [1] "has foldername, checking for lookuptable"
## [1] "file exists"
## [1] "using lookup"
## [1] "has foldername, checking for lookuptable"
## [1] "file exists"
## [1] "using lookup"
## [1] "mutually.exclusive"
## [1] "subregion1"
## [1] "mxy33"
```

```
## [1] "subregion2"
## [1] "mxy33"
## [1] "subregion3"
## [1] "mxy33"
## [1] "subregion4"
## [1] "mxy33"
## [1] "subregion5"
## [1] "mxy33"
## [1] "subregion6"
## [1] "mxy33"
## [1] "subregion7"
## [1] "mxy33"
## [1] "subregion8"
## [1] "mxy33"
## [1] "subregion9"
## [1] "mxy33"
## [1] "subregion1"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion2"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion3"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion4"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion5"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion6"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion7"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion8"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion9"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion1"
## [1] "mxy33"
```

```
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion2"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion3"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion4"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion5"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion6"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion7"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion8"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion9"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion1"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion2"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion3"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion4"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion5"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
```

```
## [1] "subregion6"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion7"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion8"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
## [1] "subregion9"
## [1] "mxy33"
## [1] "file exists"
## [1] "using existing coords..."
```

Check sensitivity to tuning parameter. See that some columns are minimised by block length 0.1, and some columns by 0.05 so tuning block length was imporant here

```
Results$Empirical.MSE
```

```
##                 tuninglength0 tuninglength0.05 tuninglength0.1
## blocklength0     0.0003547597     0.0006171809     0.002031718
## blocklength0.05  0.0008426232     0.0004558952     0.001049705
## blocklength0.1   0.0027512821     0.0015788562     0.001179298
## blocklength0.2   0.0016315219     0.0013584784     0.002096021
##                 tuninglength0.2
## blocklength0        0.004278006
## blocklength0.05     0.002573408
## blocklength0.1      0.001828421
## blocklength0.2      0.003746269
```

The block size parameter chosen is:

```
Results$EmpiricalMSE_block_size
```

```
## tuning = 0.2
##          0.1
```

This is the standard error (estimated via block bootstrap to account for spatial autocorrelation) of β

```
Results$SE.estimate.EmpiricalMSE_block_size
```

```
## [[1]]
## [1] 0.1167922
```