# Estimating Ideal Points: the 110th U.S. Senate

Simon Jackman

June 24, 2009

## 1 Data

We begin by reading the roll call data for the 110th U.S. Senate, the current U.S. Senate as of the time of writing. These data are available on Jeffrey Lewis' website at UCLA (Lewis scrapes the data from the Senate's own site), and can be read with the `readKH` function in the `pscl` package:

R Code

```
> require(pscl)
> s110 <- readKH(file="http://adric.sscnet.ucla.edu/rollcall/static/S110.ord",
+                dtl=NULL)
```

Lewis also provides a CSV file with vote-specific descriptive information, which we will attach to the `rollcall` object:

R Code

```
> voteInfo <- read.csv(file="http://adric.sscnet.ucla.edu/rollcall/static/S110desc.csv")
> s110$vote.data <- voteInfo
> dimnames(s110$vote.data)[[2]]
```

R output

```
[1] "date"        "session"     "number"      "bill"        "question"
[6] "result"      "description" "yeatotal"    "naytotal"
```

### 1.1 Lop-Sided Roll Calls

We can inspect the number of lop-sided rollcalls by extracting the appropriate component of a call to `summary`:

R Code
```
> lopSided <- summary(s110,verbose=TRUE)$lopSided
```

There are 69 unaninmous roll calls in the `rollcall` object, but as many as 140 roll calls are decided by margins of 5 or fewer Senators, and by default will be discarded from the analysis. This feature can be controlled by the user, via the `dropList` argument; we typically lose discrimination among extremist legislators when deleting lop-sided roll calls, since its extremists that usually form the minorities on these types of roll calls. Later we will investigate retaining all but unanimous roll calls in the analysis.

## 1.2 Absenteeism

There is considerable variation in rates of absenteeism in these data, which we can extract by looking at the `missing%` column of the `legisTab` matrix produced by passing the `rollcall` object to the generic `summary` function:

R Code
```
> z <- summary(s110,verbose=TRUE)$legisTab[,"missing%"]
```

We summarize the absenteeism rates with a dotplot of the 33 highest rates of absenteeism in Figure 1. Unsurprisingly, the presidential candidates McCain, Obama, and Clinton have high rates of absenteeism. Tim Johnson (D, SD) has been in ill health for much of the 110th Senate, some 311 votes.

R Code
```
> z <- sort(z,decreasing=TRUE)
> pdf(file="s110-absenteeDotPlot.pdf",
+     width=11,height=17)
> trellis.par.set("axis.text"=list(cex=1.5),
+                 "par.xlab.text"=list(cex=1.5))
> dotplot(z[33:1],
+         cex=2,
+         scales=list(alternating=3),
+         xlab="Missing (%)")
> dev.off()
```
R output
```
null device
          1
```

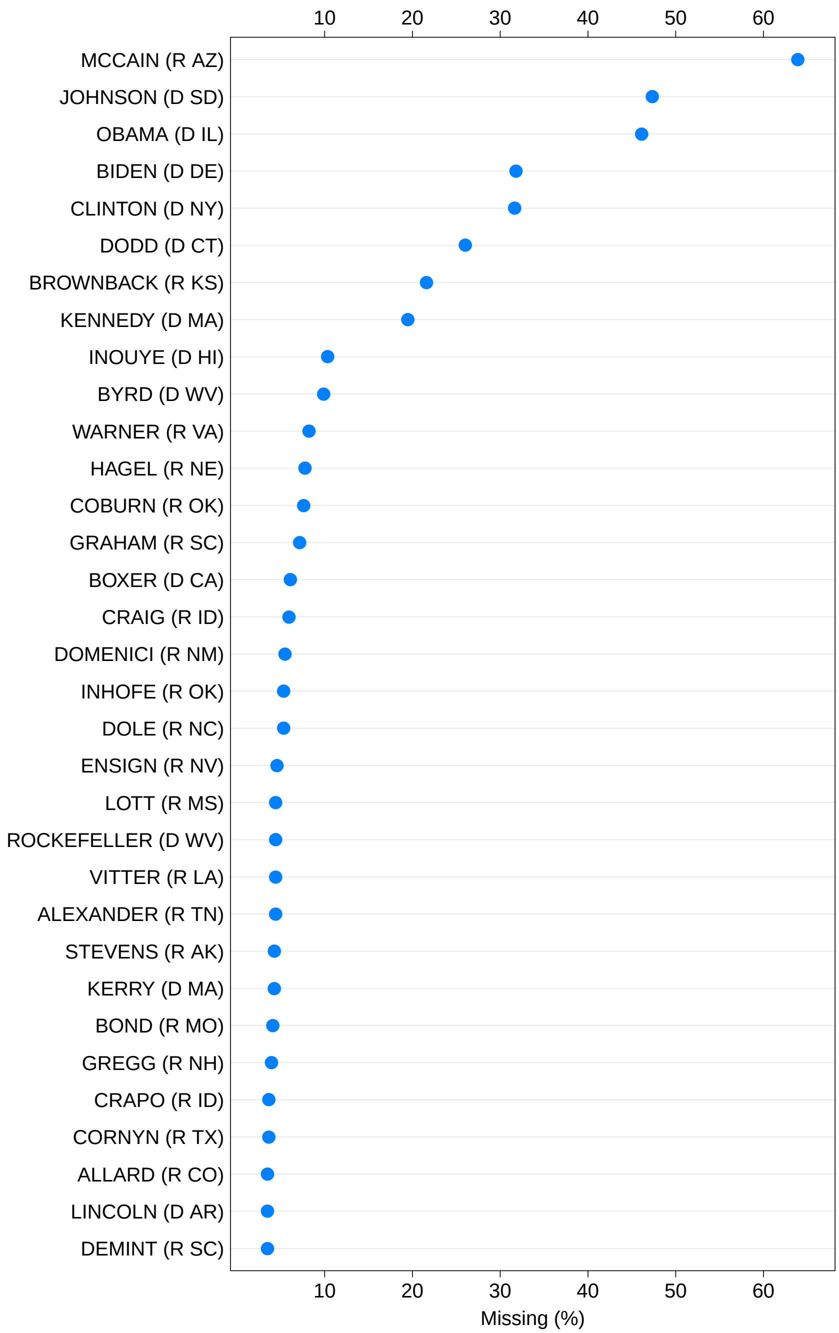



Figure 1: Rates of Missing Roll Call Votes, 110th U.S. Senate

## 1.3 Completeness

It is also useful to ask if the roll call matrix forms a complete graph among the legislators. This will usuall always be true in a modern legislature, at least over a reasonably short time frame. The difficulty here is that one subset of the legislators do not share at least one (and preferably) more roll calls with other legislators, then the two sets of legislators can not be validly scaled. The situation here is analagous to a similar problem in standardized testing: if one group of students take exam "A", and another set take exam "B", then in what sense can we compare the performance of the students? We require overlap in the groups of students, or in the test items comprising the two tests.

We extract the roll-call matrix from the `rollcall` object, converting the Poole-Rosenthal vote codes to 0 for "Nay" votes and "1" for "Yea" votes, and explictly keeping all but unanimous roll calls. Code actual votes as a "1" and missing votes of all kinds as "0". Now form a $n$-by-$n$ adjacency matrix, **S**, with $S_{ij} = 1$ if there exists a roll call where legislators $i$ and $j$ voted, and $S_{ij} = 0$ if legislators $i$ and $j$ do not share any roll call votes; I accomplish this in the code block below by manipulating the output of `dist` function in `R`.

R Code

```
> y <- convertCodes(dropRollCall(s110,dropList=list(lop=0)))
> y[!is.na(y)] <- 1
> y[is.na(y)] <- 0
> n <- dim(y)[1]
> s <- matrix(NA,n,n)
> dimnames(s) <- list(dimnames(s110$votes)[[1]],
+                     dimnames(s110$votes)[[1]])
> for(i in 1:n){
+   for(j in 1:n){
+     s[i,j] <- any(y[i,]==1 & y[j,]==1)
+   }
+ }
```

The matrix **S** can also be thought of as an adjacency matrix, representing pairwise distances between legislators in a social network (a graph). If the legislators can be jointly scaled, then it is possible to form a path of finite length from any one legislator to any other. On the other hand, if legislator $i$ and $j$ are disconnected, then they can't be jointly scaled, and the network path between them will be of indeterminite length. Thus, a test for whether a complete joint scaling is possible is to see if there exists a path from legislator $i$ to legislator $j$, $\forall\, i \neq j$.

The following code block implements this test, using a network distance function `dst` created by Peter Hoff of the University of Washington.

R Code

```
> dst <- function(Y,infd=FALSE){
+   ##calculates distances between nodes based on path length
+   ##returns d=g if nodes are not connected
+
+   g <- dim(Y)[1]
+   Dst <- Yr <- Y
+   Dst <- Y*(Y==1) + g*(Y==0)
+   for(r in 2:(g-1)) {
+     Yr <- Yr%*%Y
+     Dst <- Dst+(r-g)*( Yr>0 & Dst==g )
+   }
+   if(infd){
+     for(i in 1:g){
+       for(j in 1:g) {
+         if( Dst[i,j]==g ){
+           Dst[i,j] <- Inf
+         }
+       }
+     }
+   }
+   diag(Dst) <- 0
+
+   Dst
+ }
> d <- dst(s)
> table(d,exclude=NULL)
```

R output

```
d
    0     1     2  <NA>
  102 10294     8     0
```

R Code

```
> suspect <- apply(d,1,function(x)any(x>1))
> suspect <- (1:n)[suspect]
> dimnames(s110$votes)[[1]][suspect]
```

R output

```
[1] "LOTT (R MS)"     "JOHNSON (D SD)"  "THOMAS (R WY)"   "WICKER (R MS-1)"
[5] "BARASSO (R WY)"
```

R Code

```
> s[suspect,suspect]
```

R output

```
                LOTT (R MS) JOHNSON (D SD) THOMAS (R WY) WICKER (R MS-1)
LOTT (R MS)            TRUE           TRUE          TRUE           FALSE
JOHNSON (D SD)         TRUE           TRUE         FALSE            TRUE
THOMAS (R WY)          TRUE          FALSE          TRUE           FALSE
WICKER (R MS-1)       FALSE           TRUE         FALSE            TRUE
BARASSO (R WY)         TRUE           TRUE         FALSE            TRUE
                BARASSO (R WY)
LOTT (R MS)               TRUE
JOHNSON (D SD)            TRUE
THOMAS (R WY)            FALSE
WICKER (R MS-1)           TRUE
BARASSO (R WY)            TRUE
```

In this case we see that there are a small number of legislators with non-overlapping voting records. But because so many other senators serve continuously throughout the 110th Senate, we can easily form a network over the entire set of legislators. In short, a joint scaling is feasible in this case.

## 2 Simple Methods Based on Agreement Scores

We again extract the roll-call matrix from the `rollcall` object, converting the Poole-Rosenthal vote codes to 0 for "Nay" votes and "1" for "Yea" votes, and again keeping all but unanimous roll calls:

R Code
```
> y <- convertCodes(dropRollCall(s110,dropList=list(lop=0)))
> dim(y)
```

We convert the roll call matrix to a $n$-by-$n$ matrix of *agreement scores*, **A**, where $A_{ij}$ is the proportion of times that legislator $i$ votes the same way as legislator $j$. Where two legislators have non-overlapping voting histories, I arbitrarily set the agreement score to 0.

R Code
```
> agreement <- function(x){
+   n <- dim(x)[1]
+   k <- dim(x)[2]
+   a <- matrix(NA,n,n)
+   for(i in 1:n){
+     for(j in 1:n){
+       a[i,j] <- sum(x[i,]==x[j,],na.rm=TRUE)/sum(!is.na(x[i,]) & !is.na(x[j,]))
+     }
+   }
+   a[is.nan(a)] <- 0
+   a
+ }
> a <- agreement(y)
```

Where two legislators have non-overlapping voting histories, I arbitrarily set them to have an agreement score of zero.

We convert the agreement scores into distances by subtracting them from 1, and squaring the result:

R Code
```
> d <- (1-a)^2
```

When then double-center the resulting matrix **D**, subtracting out mean and column sums, adding in the matrix mean, and dividing by minus 2:

R Code

```
> doubleCenter <- function(x){
+   n <- dim(x)[1]
+   k <- dim(x)[2]
+   rowMeans <- matrix(apply(x,1,mean,na.rm=TRUE),n,k,byrow=TRUE)
+   colMeans <- matrix(apply(x,2,mean,na.rm=TRUE),n,k,byrow=FALSE)
+   matrixMean <- matrix(mean(x,na.rm=TRUE),n,k)
+   (x - rowMeans - colMeans + matrixMean)/-2
+ }
> d <- doubleCenter(d)
```

We now extract the first eigen-vector of the double-centered **D** matrix. In fact, Poole (2005) proves that if voting is perfect in one dimension, then the first eigenvector of the double-centered matrix of squared agreement scores recovers the ideal points of the legislators up to arbitrary rank preserving transformation.

R Code

```
> e <- eigen(d)
> lambda <- e$values
> x <- e$vector[,1] * sqrt(lambda[1])
> names(x) <- dimnames(s110$votes)[[1]]
> sort(x)[1:20]
```

R output

```
      OBAMA (D IL)       BIDEN (D DE)     CLINTON (D NY)        DODD (D CT)
        -0.3910405         -0.3625589         -0.3591632         -0.3392774
SANDERS (Indep VT)  LAUTENBERG (D NJ)       BOXER (D CA)    MENENDEZ (D NJ)
        -0.3369048         -0.3357700         -0.3357030         -0.3349416
 WHITEHOUSE (D RI)     KENNEDY (D MA)       BROWN (D OH)        LEAHY (D VT)
        -0.3287828         -0.3283529         -0.3260019         -0.3246544
     DURBIN (D IL)      HARKIN (D IA)        REED (D RI)     SCHUMER (D NY)
        -0.3235736         -0.3227137         -0.3224092         -0.3204099
   FEINGOLD (D WI)       KERRY (D MA)    STABENOW (D MI)      MURRAY (D WA)
        -0.3193665         -0.3155626         -0.3147576         -0.3136100
```

R Code

```
> sort(x,decreasing=TRUE)[1:20]
```

R output

```
   DEMINT (R SC)    COBURN (R OK)    INHOFE (R OK)    ALLARD (R CO)
       0.5138812        0.4837451        0.4519997        0.4310796
      KYL (R AZ)    ENSIGN (R NV)      BURR (R NC)      ENZI (R WY)
       0.4301306        0.4214651        0.4140246        0.4119679
   MCCAIN (R AZ)   BUNNING (R KY)    VITTER (R LA)    CORNYN (R TX)
       0.4103463        0.3978523        0.3946331        0.3875766
  BARASSO (R WY)  SESSIONS (R AL)    GRAHAM (R SC) CHAMBLISS (R GA)
       0.3822370        0.3670989        0.3524873        0.3511578
   THOMAS (R WY)     GREGG (R NH) MCCONNELL (R KY)   ISAKSON (R GA)
       0.3505522        0.3472778        0.3425885        0.3404445
```

# 3 Heckman-Snyder via Principal Components

Now form a cross-legislator correlation matrix, using pairwise handling of missing data. We then set any missing pairwise correlations to zero.

R Code
```
> r <- cor(t(y),use="pairwise")
> dim(r)
```
R output
```
[1] 102 102
```
R Code
```
> table(is.na(r))
```
R output
```
FALSE  TRUE
10396     8
```
R Code
```
> r[is.na(r)] <- 0
```

The eigen-structure of the correlation matrix `r` reveals much about the structure of the roll call matrix. First we look at the eigen-values, $\lambda_j, j = 1, \ldots, n$, with $\lambda_1 > \lambda_2 > \ldots > \lambda_n$. In roll call data it is common to see a very large first eigenvalue, corresponding to a large, first dimension underlying the roll call votes (e.g., the left-right ideological continuum, perhaps reinforced by party-line voting). We plot the first 20 eigenvalues of `r` in Figure 2.

R Code
```
> e <- eigen(r)
> lambda <- e$values
> n <- dim(e)[1]
> plot(lambda[1:20],
+      type="b",
+      las=1,
+      ylab="Eigenvalues")
> abline(h=1)
```

Observe that the first eigenvalue of `r` is 49.97, the 2nd eigenvalue is 20.31, and that there are only 9 eigenvalues greater than 1.

The first *d* eigenvectors of `r` are usually good (if rough) estimates of legislative ideal points in *d* dimensions. Here we plot the first two eigenvectors in Figure 3; red points indicate Republicans, blue points indicate Democrats.

R Code
```
> v <- e$vectors[,1:2]
> n <- dim(v)[1]
> col <- rep("black",n)
> col[s110$legis.data$party=="D"] <- "blue"
> col[s110$legis.data$party=="R"] <- "red"
> plot(v[,1],
+      v[,2],
+      col=col,
```

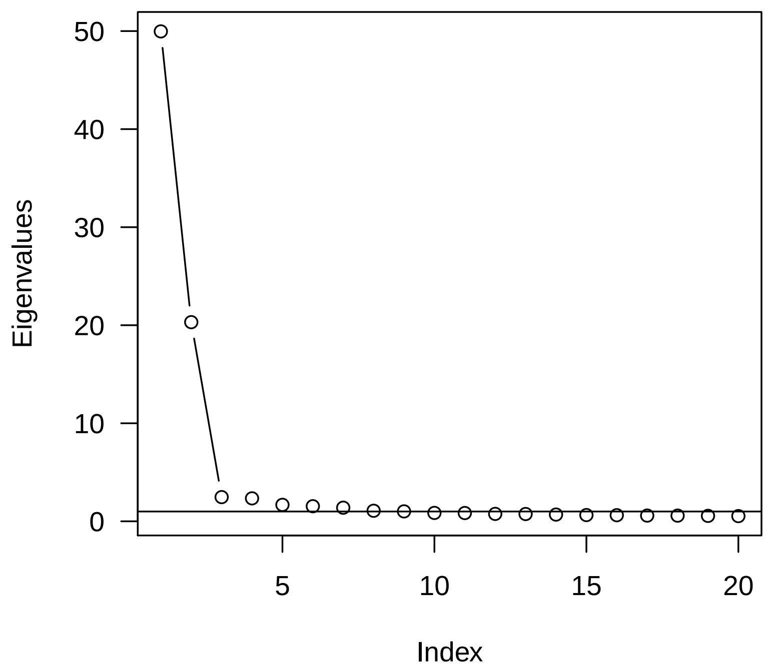



Figure 2: Eigenvalues of Correlation Matrix

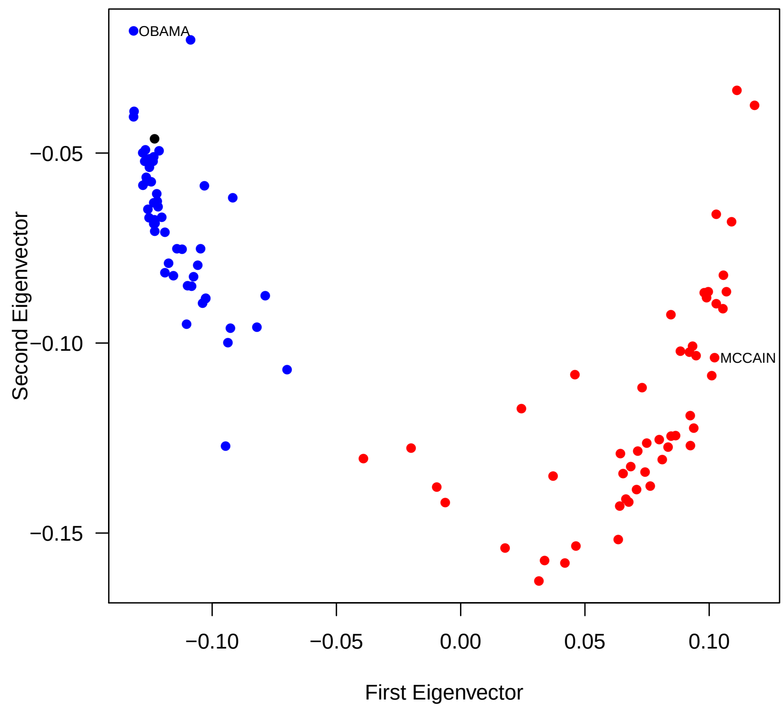



Figure 3: Eigenvectors as Ideal Point Estimates

```
+       las=1,
+       pch=16,
+       xlab="First Eigenvector",
+       ylab="Second Eigenvector")
> obama <- grep("OBAMA",dimnames(s110$votes)[[1]])
> text(v[obama,1],
+       v[obama,2],
+       "OBAMA",
+       adj=-.1,
+       cex=.65)
> mcCain <- grep("MCCAIN",dimnames(s110$votes)[[1]])
> text(v[mcCain,1],
+       v[mcCain,2],
+       "MCCAIN",
+       adj=-.1,
+       cex=.65)
```

# 4 W-NOMINATE

For details on the Gaussian utility, logistic model underlying NOMINATE, see Poole (2005), or Poole and Rosenthal (1997). A helpful vignette appears in Poole et al. (2007).

Here I obtain ideal point estimates by calling `wnominate` with the 110th Senate data used above. The default is to fit a 2-dimensional model. Goodness of fit statistics can be generated by passing the fitted model object to a `summary` method:

R Code
```
> require(wnominate)
```

R output
```
## W-NOMINATE Ideal Point Package
## Copyright 2006 - 2009
## Keith Poole, Jeffrey Lewis, James Lo, and Royce Carroll
## Support provided by the U.S. National Science Foundation
## NSF Grant SES-0611974
```

R Code
```
> w1 <- wnominate(s110,
+                  dims=1,
+                  polarity=c("MCCONNELL (R KY)"))
```

R output
```
Preparing to run W-NOMINATE...

        Checking data...

                All members meet minimum vote requirements.

                Votes dropped:
                ... 107 of 657 total votes dropped.

        Running W-NOMINATE...

                Getting bill parameters...
                Getting legislator coordinates...
                Starting estimation of Beta...
                Getting bill parameters...
                Getting legislator coordinates...
                Starting estimation of Beta...
                Getting bill parameters...
                Getting legislator coordinates...


W-NOMINATE estimation completed successfully.
W-NOMINATE took 8.583 seconds to execute.
```

R Code
```
> summary(w1)
```

R output
```
SUMMARY OF W-NOMINATE OBJECT
----------------------------

```

```
Number of Legislators:          102 (0 legislators deleted)
Number of Votes:          550 (107 votes deleted)
Number of Dimensions:          1
Predicted Yeas:                  27946 of 30339 (92.1%) predictions correct
Predicted Nays:                  19034 of 21837 (87.2%) predictions correct
Correct Classifiction:          90.04%
APRE:                          0.701
GMP:                          0.79


The first 10 legislator estimates are:
                coord1D se1D
BYRD (D WV)      -0.740    0
INOUYE (D HI)    -0.655    0
KENNEDY (D MA)   -0.861    0
STEVENS (R AK)    0.106    0
COCHRAN (R MS)    0.306    0
LOTT (R MS)       0.469    0
BIDEN (D DE)     -0.892    0
DOMENICI (R NM)   0.173    0
BAUCUS (D MT)    -0.589    0
DODD (D CT)      -0.903    0
```

R Code

```
> w2 <- wnominate(s110,
+                 dims=2,
+                 polarity=c("MCCONNELL (R KY)",
+                   "BYRD (D WV)"))
```

R output

```
Preparing to run W-NOMINATE...

        Checking data...

                All members meet minimum vote requirements.

                Votes dropped:
                ... 107 of 657 total votes dropped.

        Running W-NOMINATE...

                Getting bill parameters...
                Getting legislator coordinates...
                Starting estimation of Beta...
                Getting bill parameters...
                Getting legislator coordinates...
                Starting estimation of Beta...
                Getting bill parameters...
                Getting legislator coordinates...
                Getting bill parameters...
                Getting legislator coordinates...
                Estimating weights...
                Getting bill parameters...
                Getting legislator coordinates...
                Estimating weights...
                Getting bill parameters...
                Getting legislator coordinates...
```

```
W-NOMINATE estimation completed successfully.
W-NOMINATE took 13.28 seconds to execute.
```

R Code

```
> summary(w2)
```

R output

```
SUMMARY OF W-NOMINATE OBJECT
----------------------------

Number of Legislators:          102 (0 legislators deleted)
Number of Votes:          550 (107 votes deleted)
Number of Dimensions:          2
Predicted Yeas:                 28137 of 30339 (92.7%) predictions correct
Predicted Nays:                 19346 of 21837 (88.6%) predictions correct
Correct Classifiction:          90.05% 91.01%
APRE:                          0.701 0.73
GMP:                          0.79 0.808


The first 10 legislator estimates are:
                coord1D coord2D
BYRD (D WV)      -0.772   0.636
INOUYE (D HI)    -0.657  -0.238
KENNEDY (D MA)   -0.835  -0.550
STEVENS (R AK)    0.122  -0.477
COCHRAN (R MS)    0.323  -0.325
LOTT (R MS)       0.483  -0.412
BIDEN (D DE)     -0.893  -0.259
DOMENICI (R NM)   0.189  -0.688
BAUCUS (D MT)    -0.590   0.770
DODD (D CT)      -0.898  -0.431
```

R Code

```
> plot(w1,las=1)
```

R Code

```
> plot(w2,las=1)
```

R output

```
NULL
```

The `wnominate` package provides plot methods for objects of class `nomObject`; see Figures 4 and 5.

Let's compare the output of W-NOMINATE with some of the more simple things we tried via eigendecompositions:

R Code

```
> comparisonData <- data.frame(wnom=w1$legislators$coord1D,
+                              agreementScore=x)
> plot(wnom~agreementScore,
+      data=comparisonData,
+      xlab="Agreement Score Estimate",
+      ylab="W-NOMINATE Estimate",
+      col=col)
```

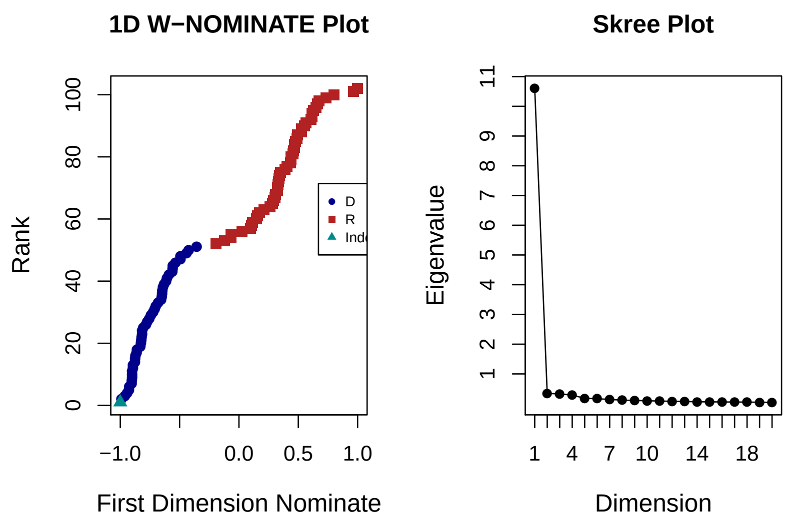
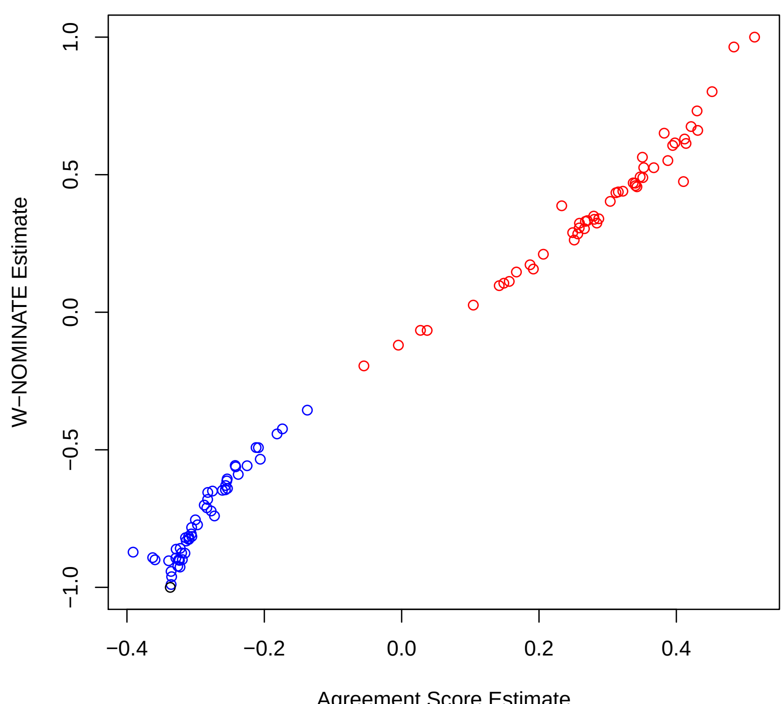



Figure 4: Default plot method, 1 dimensional WNOMINATE fit to 110th U.S. Senate



```
R output
1 NULL
```

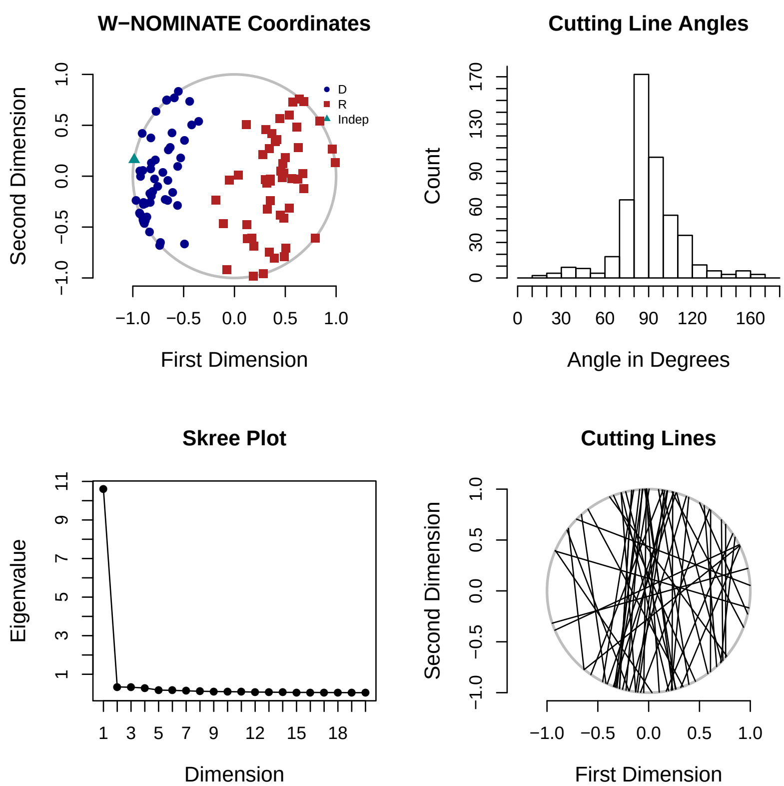



Figure 5: Default plot method, 2 dimensional WNOMINATE fit to 110th U.S. Senate

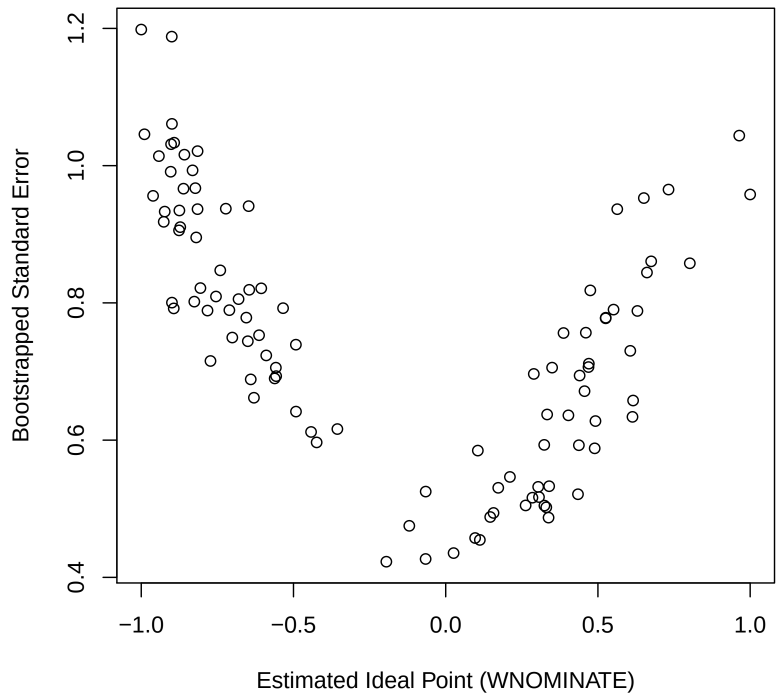
## 4.1 Standard Errors via Parametric bootstrapping

We use the bootstrapping options in the `wnominate` function to generate approximate standard errors for the legislators' ideal points:

R Code
```
> w1s <- wnominate(s110,dims=1,
+                  polarity="MCCONNELL (R KY)",
+                  trials=30)
```
R output
```
Preparing to run W-NOMINATE...

        Checking data...

                All members meet minimum vote requirements.

                Votes dropped:
                ... 107 of 657 total votes dropped.

        Running W-NOMINATE...

                Getting bill parameters...
                Getting legislator coordinates...
                Starting estimation of Beta...
                Getting bill parameters...
                Getting legislator coordinates...
                Starting estimation of Beta...
                Getting bill parameters...
                Getting legislator coordinates...
                Starting bootstrap iterations...
............................                Computing standard errors...
W-NOMINATE estimation completed successfully.
W-NOMINATE took 310.377 seconds to execute.
```
R Code
```
> results <- w1s$legislators[,c("coord1D","se1D")]
```

Note that some of these standard errors are quite large. We also have the curious fact that legislators placed close to the boundary of the `NOMINATE` parameter space pick up quite large standard errors.

R Code
```
> plot(results[,1],results[,2],
+       xlab="Estimated Ideal Point (WNOMINATE)",
+       ylab="Bootstrapped Standard Error")
```




# 5 Quadratic normal model, via Bayesian simulation (Clinton, Jackman and Rivers)

We fit the quadratic/normal, two-parameter IRT model used by Clinton, Jackman and Rivers (2004), via a Markoc chain Monte Carlo algorithm. We use the default settings in `ideal`:

R Code

```
> id1 <- ideal(s110,
+              verbose=TRUE,
+              normalize=TRUE,
+              mda=FALSE)
```

R output

```
ideal: analysis of roll call data via Markov chain Monte Carlo methods.

Subsetting rollcall object s110 using dropList
Using the following codes to represent roll call votes:
```

```
Yea:                 1 2 3
Nay:                 4 5 6
Abstentions:         7 8 9
Not In Legislature:        0

Ideal Point Estimation

Number of Legislators                102
Number of Items                        588

checking for any user-supplied priors...
setting prior means for ideal points to all zeros
setting prior precisions for ideal points to all 1
setting prior means for item parameters to all zeros
setting prior precisions for item parameters to all 0.01

checking start values...
will use eigen-decomposition method to get start values for ideal points...done
running 588 vote-specific probit GLMs
 for start values for item/bill parameters
 conditional on start values for ideal points...done
using the following start values for ideal points (summary follows):
       V1
 Min.   :-0.9305
 1st Qu.:-0.8505
 Median :-0.2450
 Mean   :-0.0545
 3rd Qu.: 0.7739
 Max.   : 0.8688
using the following start values for item parameters (summary follows):
       V1                  V2
 Min.   :-20.0000   Min.   :-20.0000
 1st Qu.: -2.9193   1st Qu.: -2.1763
 Median : -0.9734   Median : -0.4322
 Mean   : -0.7021   Mean   : -0.8354
 3rd Qu.:  2.0992   3rd Qu.:  0.4224
 Max.   : 20.0000   Max.   : 20.0000

Starting MCMC Iterations...

Current Iteration: 500 (5% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 1000 (10% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 1500 (15% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 2000 (20% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 2500 (25% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 3000 (30% of 10000 iterations requested)
MDA sigma= 1.000
```

```

Current Iteration: 3500 (35% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 4000 (40% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 4500 (45% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 5000 (50% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 5500 (55% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 6000 (60% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 6500 (65% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 7000 (70% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 7500 (75% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 8000 (80% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 8500 (85% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 9000 (90% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 9500 (95% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 10000 (100% of 10000 iterations requested)
MDA sigma= 1.000

MCMC sampling done, computing posterior means for ideal points...
done
```

R Code

```
> summary(id1)
```

R output

```
Markov chain Monte Carlo Analysis of Roll Call Data
       (2-parameter item-response modeling)
Source:                 http://adric.sscnet.ucla.edu/rollcall/static/S110.ord
ideal was called as follows:
ideal(object = s110, mda = FALSE, normalize = TRUE, verbose = TRUE,
    d = 1, codes = list(yea = c(1, 2, 3), nay = c(4, 5, 6), notInLegis = 0,
        missing = c(7, 8, 9)), dropList = list(codes = "notInLegis",
        lop = 0), maxiter = 10000, thin = 100, burnin = 5000,
    impute = FALSE, store.item = FALSE)
```

```
Number of Legislators:         102
Number of Votes:         588
Number of Dimensions:         1
Number of Iterations:         10000
        Thinned By:         100
        Burn-in:         5000

Ideal Points (Posterior Means), by Party
                     Mean   2.5%  97.5%
D: Dimension 1     -0.908 -1.472 -0.302
Indep: Dimension 1 -1.392 -1.392 -1.392
R: Dimension 1      0.918  0.150  1.857

Ideal Points, Dimension 1(sorted by posterior means):
                     Mean Std.Dev.   2.5%  97.5%
MENENDEZ (D NJ)    -1.474    0.101 -1.685 -1.317
LAUTENBERG (D NJ)  -1.474    0.105 -1.637 -1.304
BOXER (D CA)       -1.466    0.116 -1.712 -1.276
SANDERS (Indep VT) -1.392    0.103 -1.615 -1.227
HARKIN (D IA)      -1.386    0.109 -1.631 -1.207
DURBIN (D IL)      -1.323    0.085 -1.466 -1.186
BROWN (D OH)       -1.320    0.091 -1.515 -1.151
CLINTON (D NY)     -1.280    0.105 -1.460 -1.114
BIDEN (D DE)       -1.227    0.078 -1.350 -1.099
SCHUMER (D NY)     -1.222    0.077 -1.350 -1.082
KERRY (D MA)       -1.220    0.084 -1.360 -1.080
DODD (D CT)        -1.181    0.075 -1.320 -1.039
LEAHY (D VT)       -1.168    0.063 -1.300 -1.069
CARDIN (D MD)      -1.154    0.069 -1.314 -1.037
MURRAY (D WA)      -1.120    0.066 -1.247 -1.016
KENNEDY (D MA)     -1.113    0.064 -1.239 -1.019
WHITEHOUSE (D RI)  -1.104    0.062 -1.228 -0.992
WYDEN (D OR)       -1.100    0.060 -1.192 -0.961
CANTWELL (D WA)    -1.096    0.063 -1.232 -0.983
OBAMA (D IL)       -1.088    0.072 -1.229 -0.970
AKAKA (D HI)       -1.081    0.062 -1.195 -0.957
REED (D RI)        -1.068    0.066 -1.188 -0.957
STABENOW (D MI)    -1.054    0.067 -1.172 -0.914
BINGAMAN (D NM)    -1.008    0.053 -1.111 -0.906
LEVIN (D MI)       -0.973    0.047 -1.053 -0.861
KLOBUCHAR (D MN)   -0.963    0.055 -1.045 -0.865
FEINGOLD (D WI)    -0.915    0.055 -1.036 -0.823
KOHL (D WI)        -0.907    0.049 -0.984 -0.801
FEINSTEIN (D CA)   -0.842    0.050 -0.933 -0.747
CASEY (D PA)       -0.803    0.043 -0.885 -0.725
BYRD (D WV)        -0.799    0.054 -0.902 -0.711
MIKULSKI (D MD)    -0.783    0.046 -0.885 -0.701
REID (D NV)        -0.733    0.043 -0.803 -0.644
INOUYE (D HI)      -0.729    0.044 -0.805 -0.646
ROCKEFELLER (D WV) -0.725    0.045 -0.808 -0.629
NELSON (D FL)      -0.723    0.047 -0.807 -0.646
TESTER (D MT)      -0.715    0.053 -0.801 -0.635
DORGAN (D ND)      -0.701    0.048 -0.803 -0.628
SALAZAR (D CO)     -0.657    0.049 -0.737 -0.585
WEBB (D VA)        -0.655    0.039 -0.730 -0.583
CONRAD (D ND)      -0.640    0.044 -0.731 -0.567
```

| | | | | | |
|---|---|---|---|---|---|
| 67 | BAUCUS (D MT) | -0.627 | 0.043 | -0.709 | -0.556 |
| 68 | LINCOLN (D AR) | -0.568 | 0.040 | -0.640 | -0.487 |
| 69 | CARPER (D DE) | -0.546 | 0.035 | -0.611 | -0.491 |
| 70 | JOHNSON (D SD) | -0.506 | 0.049 | -0.597 | -0.412 |
| 71 | MCCASKILL (D MO) | -0.474 | 0.042 | -0.557 | -0.400 |
| 72 | PRYOR (D AR) | -0.442 | 0.038 | -0.504 | -0.370 |
| 73 | LIEBERMAN (D CT) | -0.397 | 0.040 | -0.485 | -0.338 |
| 74 | LANDRIEU (D LA) | -0.339 | 0.036 | -0.406 | -0.272 |
| 75 | BAYH (D IN) | -0.291 | 0.037 | -0.356 | -0.215 |
| 76 | NELSON (D NE) | -0.222 | 0.034 | -0.291 | -0.152 |
| 77 | SNOWE (R ME) | 0.016 | 0.026 | -0.032 | 0.055 |
| 78 | COLLINS (R ME) | 0.133 | 0.034 | 0.079 | 0.196 |
| 79 | SPECTER (R PA) | 0.201 | 0.027 | 0.153 | 0.251 |
| 80 | SMITH (R OR) | 0.206 | 0.024 | 0.168 | 0.251 |
| 81 | COLEMAN (R MN) | 0.323 | 0.035 | 0.268 | 0.380 |
| 82 | VOINOVICH (R OH) | 0.387 | 0.037 | 0.330 | 0.447 |
| 83 | STEVENS (R AK) | 0.446 | 0.032 | 0.386 | 0.503 |
| 84 | MURKOWSKI (R AK) | 0.462 | 0.031 | 0.399 | 0.520 |
| 85 | LUGAR (R IN) | 0.511 | 0.031 | 0.449 | 0.569 |
| 86 | WARNER (R VA) | 0.525 | 0.032 | 0.474 | 0.592 |
| 87 | DOMENICI (R NM) | 0.559 | 0.037 | 0.494 | 0.626 |
| 88 | HAGEL (R NE) | 0.564 | 0.034 | 0.506 | 0.622 |
| 89 | ROBERTS (R KS) | 0.692 | 0.042 | 0.611 | 0.766 |
| 90 | GRASSLEY (R IA) | 0.707 | 0.037 | 0.636 | 0.764 |
| 91 | HATCH (R UT) | 0.716 | 0.033 | 0.656 | 0.772 |
| 92 | ALEXANDER (R TN) | 0.743 | 0.029 | 0.683 | 0.793 |
| 93 | SUNUNU (R NH) | 0.754 | 0.034 | 0.703 | 0.829 |
| 94 | COCHRAN (R MS) | 0.765 | 0.039 | 0.692 | 0.845 |
| 95 | BOND (R MO) | 0.771 | 0.033 | 0.717 | 0.829 |
| 96 | DOLE (R NC) | 0.775 | 0.044 | 0.701 | 0.866 |
| 97 | MARTINEZ (R FL) | 0.786 | 0.033 | 0.721 | 0.839 |
| 98 | BENNETT (R UT) | 0.804 | 0.036 | 0.737 | 0.868 |
| 99 | HUTCHISON (R TX) | 0.808 | 0.040 | 0.738 | 0.880 |
| 100 | CORKER (R TN) | 0.815 | 0.038 | 0.745 | 0.896 |
| 101 | SHELBY (R AL) | 0.884 | 0.042 | 0.821 | 0.968 |
| 102 | WICKER (R MS-1) | 0.914 | 0.069 | 0.796 | 1.033 |
| 103 | BROWNBACK (R KS) | 0.956 | 0.046 | 0.862 | 1.055 |
| 104 | THUNE (R SD) | 0.975 | 0.045 | 0.877 | 1.035 |
| 105 | CRAIG (R ID) | 0.993 | 0.045 | 0.915 | 1.072 |
| 106 | MCCAIN (R AZ) | 1.016 | 0.083 | 0.876 | 1.163 |
| 107 | MCCONNELL (R KY) | 1.034 | 0.040 | 0.956 | 1.093 |
| 108 | LOTT (R MS) | 1.043 | 0.062 | 0.921 | 1.140 |
| 109 | ISAKSON (R GA) | 1.045 | 0.048 | 0.942 | 1.117 |
| 110 | GREGG (R NH) | 1.058 | 0.048 | 0.973 | 1.141 |
| 111 | CRAPO (R ID) | 1.073 | 0.040 | 0.994 | 1.138 |
| 112 | CHAMBLISS (R GA) | 1.078 | 0.047 | 0.996 | 1.161 |
| 113 | GRAHAM (R SC) | 1.125 | 0.051 | 1.058 | 1.222 |
| 114 | SESSIONS (R AL) | 1.127 | 0.046 | 1.019 | 1.204 |
| 115 | THOMAS (R WY) | 1.173 | 0.112 | 0.988 | 1.370 |
| 116 | CORNYN (R TX) | 1.217 | 0.043 | 1.142 | 1.298 |
| 117 | BUNNING (R KY) | 1.301 | 0.051 | 1.216 | 1.409 |
| 118 | BURR (R NC) | 1.308 | 0.050 | 1.226 | 1.397 |
| 119 | VITTER (R LA) | 1.334 | 0.056 | 1.223 | 1.436 |
| 120 | ENZI (R WY) | 1.348 | 0.052 | 1.261 | 1.442 |
| 121 | BARASSO (R WY) | 1.375 | 0.048 | 1.284 | 1.458 |
| 122 | ALLARD (R CO) | 1.396 | 0.056 | 1.311 | 1.526 |
| 123 | ENSIGN (R NV) | 1.426 | 0.056 | 1.326 | 1.523 |

```
KYL (R AZ)          1.482    0.052  1.401  1.581
INHOFE (R OK)       1.673    0.076  1.549  1.792
COBURN (R OK)       1.919    0.076  1.802  2.099
DEMINT (R SC)       2.052    0.085  1.865  2.198
```

R Code

```
> comparisonData$cjr <- id1$xbar
```

R Code

```
> id2 <- ideal(s110,d=2,
+               mda=FALSE,
+               verbose=TRUE)
```

R output

```
ideal: analysis of roll call data via Markov chain Monte Carlo methods.

normalize option is only meaningful when d=1
Subsetting rollcall object s110 using dropList
Using the following codes to represent roll call votes:
Yea:                 1 2 3
Nay:                 4 5 6
Abstentions:         7 8 9
Not In Legislature:         0

Ideal Point Estimation

Number of Legislators                 102
Number of Items                         588

checking for any user-supplied priors...
setting prior means for ideal points to all zeros
setting prior precisions for ideal points to all 1
setting prior means for item parameters to all zeros
setting prior precisions for item parameters to all 0.01

checking start values...
will use eigen-decomposition method to get start values for ideal points...done
running 588 vote-specific probit GLMs
 for start values for item/bill parameters
 conditional on start values for ideal points...done
using the following start values for ideal points (summary follows):
       V1                 V2
 Min.   :-0.9305   Min.   :-0.54271
 1st Qu.:-0.8505   1st Qu.:-0.12527
 Median :-0.2450   Median : 0.04935
 Mean   :-0.0545   Mean   :-0.01242
 3rd Qu.: 0.7739   3rd Qu.: 0.14044
 Max.   : 0.8688   Max.   : 0.29942
using the following start values for item parameters (summary follows):
       V1                  V2                  V3
 Min.   :-20.0000   Min.   :-20.0000   Min.   :-20.0000
 1st Qu.: -2.4470   1st Qu.: -3.2487   1st Qu.: -2.0984
 Median : -0.9526   Median : -0.9578   Median : -0.4115
 Mean   : -0.3328   Mean   : -1.3599   Mean   : -0.9837
 3rd Qu.:  2.0381   3rd Qu.:  1.0267   3rd Qu.:  0.4825
 Max.   : 20.0000   Max.   : 20.0000   Max.   : 20.0000

```

```
Starting MCMC Iterations...

Current Iteration: 500 (5% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 1000 (10% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 1500 (15% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 2000 (20% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 2500 (25% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 3000 (30% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 3500 (35% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 4000 (40% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 4500 (45% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 5000 (50% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 5500 (55% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 6000 (60% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 6500 (65% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 7000 (70% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 7500 (75% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 8000 (80% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 8500 (85% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 9000 (90% of 10000 iterations requested)
MDA sigma= 1.000

Current Iteration: 9500 (95% of 10000 iterations requested)
```

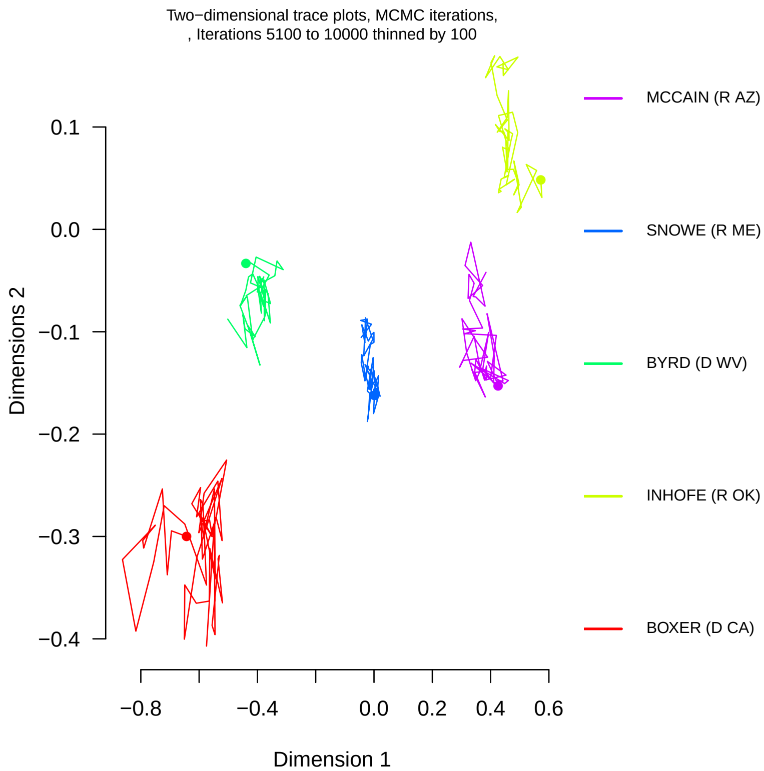
```
MDA sigma= 1.000

Current Iteration: 10000 (100% of 10000 iterations requested)
MDA sigma= 1.000

MCMC sampling done, computing posterior means for ideal points...
done
```

R Code

```
> tracex(id2,d=1:2,
+         legis=c("BOXER","INHOFE","BYRD","SNOWE","MCCAIN"),
+         showAll=TRUE)
```

R output

```
matching BOXER with BOXER (D CA)
matching INHOFE with INHOFE (R OK)
matching BYRD with BYRD (D WV)
matching SNOWE with SNOWE (R ME)
matching MCCAIN with MCCAIN (R AZ)
```




R Code

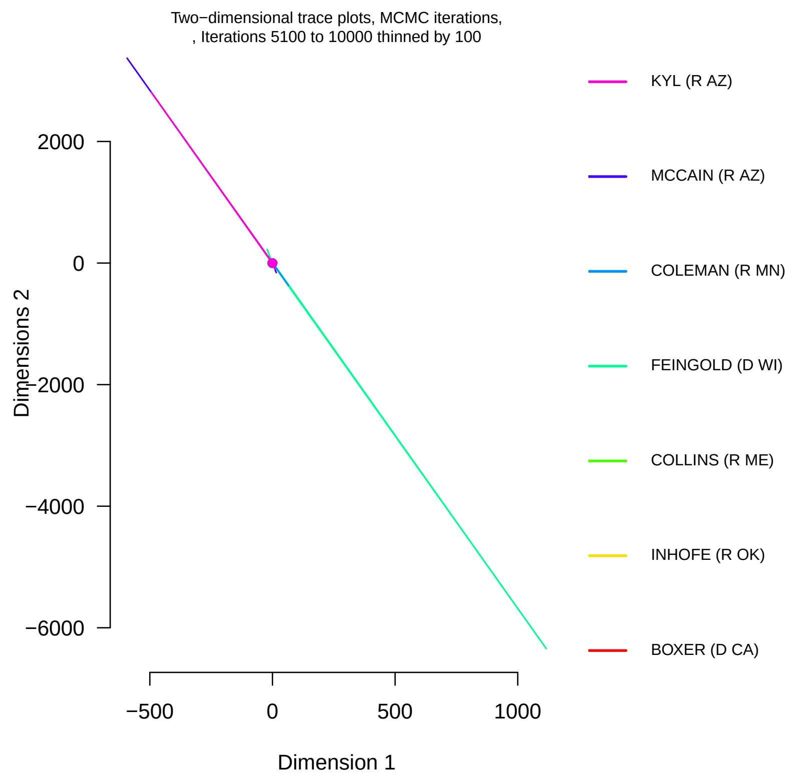
```
> id2pp <- postProcess(id2,
+                      constraints=list(BOXER=c(-1,0),
+                        INHOFE=c(1,0),
+                        SNOWE=c(0,1)))
```

R Code

```
> tracex(id2pp,d=1:2,
+         legis=c("BOXER","INHOFE","COLLINS","FEINGOLD","COLEMAN",
+           "MCCAIN","KYL"),
+         showAll=TRUE)
```

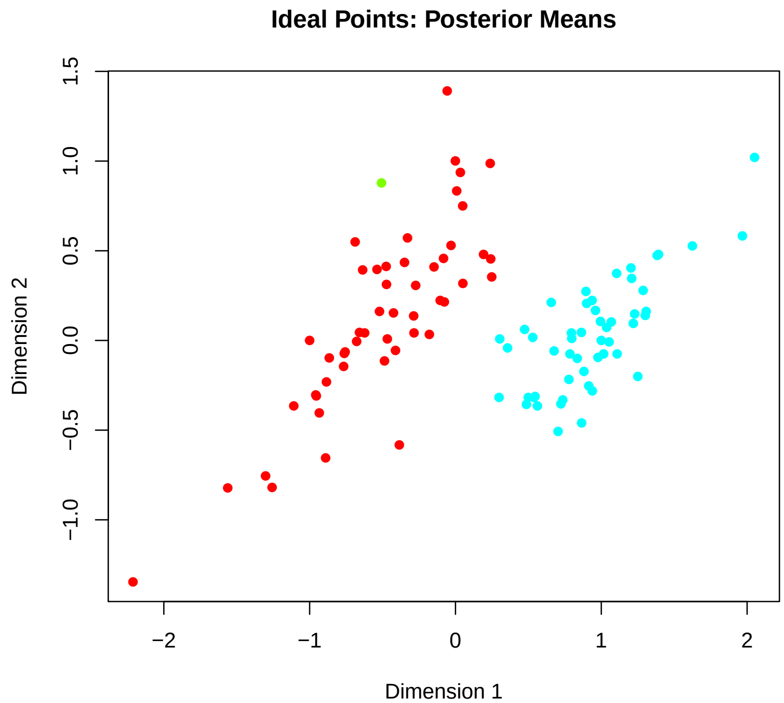
R output

```
matching BOXER with BOXER (D CA)
matching INHOFE with INHOFE (R OK)
matching COLLINS with COLLINS (R ME)
matching FEINGOLD with FEINGOLD (D WI)
matching COLEMAN with COLEMAN (R MN)
matching MCCAIN with MCCAIN (R AZ)
matching KYL with KYL (R AZ)
```




R Code

```
> id2weird <- postProcess(id2,
+                         constraints=list(MCCONNELL=c(1,0),
+                           BYRD=c(0,1),
+                           BOXER=c(-1,0)))
> plot(id2weird)
```




R Code

```
> pdf(file="CJRidealPoints.pdf",
+      height=13,width=8.5)
> plot(id1,
+       showAllNames=TRUE)
```

R output

```
Looking up legislator names and party affiliations
in rollcall object s110
```

R Code

```
> dev.off()
```

R output

```
quartz
     2
```

# 6 Quadratic logistic model, via JAGS

We use the following JAGS program:

JAGS code

```
model{
        ## loop over legislators
```

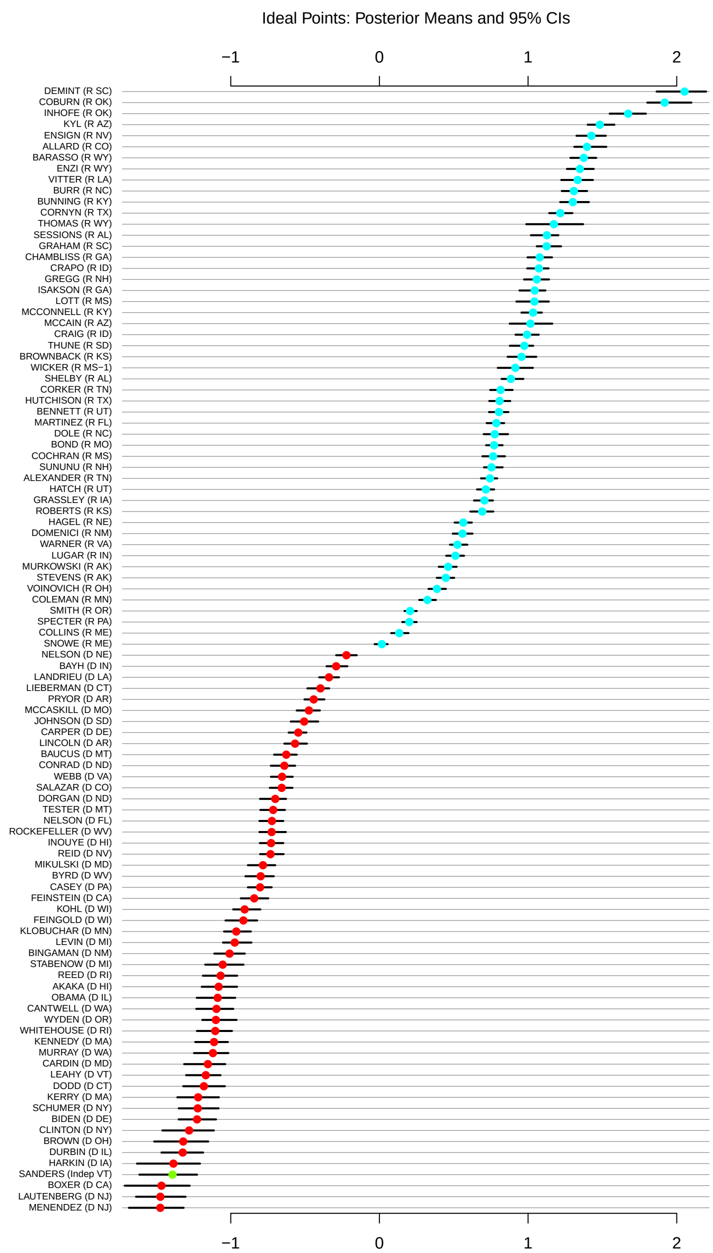



Figure 6: Estimated Ideal Points (Posterior Means) and marginal 95% HPDs, quadratic normal model (CJR).

```
        for(i in 1:n){
              ## loop over roll calls
              for(j in 1:m){
                          probit(p[i,j]) <- x[i]*beta[j] - alpha[j]
                    y[i,j] ~ dbern(p[i,j])
                    }
        }

        ## priors
        for(i in 1:n){
                x[i] ~ dnorm(0,1)
        }

        for(j in 1:m){
              beta[j] ~ dnorm(0,.01)
              alpha[j] ~ dnorm(0,.01)
        }
}
```

R Code

```
> require(rjags)
```

R output

```
loading JAGS module
   basemod
   bugs
```

R Code

```
> y <- convertCodes(dropRollCall(s110,dropList=list(lop=0)))
> n <- dim(y)[1]
> m <- dim(y)[2]
> forJags <- list(y=y,
+                 n=n,
+                 m=m)
> xinits <- rep(0,n)
> xinits[s110$legis.data$party=="R"] <- 1
> xinits[s110$legis.data$party=="D"] <- -1
> inits <- list(x=xinits,
+               beta=rep(0,m),
+               alpha=rep(0,m))
> inits <- list(inits)
> jagsModel <- jags.model(file="probit.bug",
+                         data=forJags,
+                         inits=inits)
```

R output

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 241187
```

R Code

```
> probitModel <- coda.samples(jagsModel,
+                            variable.names="x",
+                            n.iter=1e3)
> summary(probitModel)
```

R output

```
Iterations = 1001:2000
Thinning interval = 1
Number of chains = 1
```

```
Sample size per chain = 1000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

            Mean      SD  Naive SE Time-series SE
x[1]   -0.28095 0.02265 0.0007162       0.002243
x[2]   -0.26473 0.02336 0.0007388       0.002348
x[3]   -0.39111 0.03294 0.0010417       0.003078
x[4]    0.11127 0.02197 0.0006949       0.002478
x[5]    0.20831 0.02527 0.0007993       0.002807
x[6]    0.29951 0.02931 0.0009268             NA
x[7]   -0.43026 0.03941 0.0012461       0.003460
x[8]    0.14334 0.02278 0.0007205       0.002383
x[9]   -0.22835 0.01941 0.0006139       0.001926
x[10]  -0.41052 0.03537 0.0011185       0.003192
x[11]   0.19023 0.02351 0.0007436             NA
x[12]  -0.48425 0.03862 0.0012212       0.003111
x[13]  -0.41062 0.03337 0.0010552       0.003204
x[14]  -0.37497 0.02731 0.0008637       0.002382
x[15]  -0.28067 0.02310 0.0007305       0.002284
x[16]   0.19498 0.02409 0.0007616             NA
x[17]   0.12824 0.02269 0.0007174             NA
x[18]  -0.25944 0.02180 0.0006894       0.002204
x[19]  -0.02466 0.01778 0.0005622       0.001942
x[20]  -0.34770 0.02547 0.0008056       0.002386
x[21]   0.13368 0.02218 0.0007013             NA
x[22]   0.27996 0.02785 0.0008807       0.358062
x[23]  -0.25574 0.02108 0.0006667       0.002120
x[24]   0.29777 0.02699 0.0008536       0.002981
x[25]   0.18614 0.02428 0.0007678             NA
x[26]  -0.43106 0.03453 0.0010918       0.003121
x[27]  -0.38559 0.02828 0.0008943       0.002659
x[28]   0.03260 0.01921 0.0006074       0.002104
x[29]  -0.36117 0.02609 0.0008249       0.002451
x[30]  -0.50475 0.04533 0.0014333       0.003760
x[31]  -0.42384 0.03483 0.0011015       0.003230
x[32]   0.29379 0.02824 0.0008931       0.003194
x[33]  -0.26067 0.02283 0.0007218       0.002278
x[34]  -0.50875 0.04459 0.0014100       0.003535
x[35]  -0.20691 0.02000 0.0006325       0.001872
x[36]  -0.46634 0.04075 0.0012887       0.003328
x[37]   0.29177 0.03570 0.0011290       0.003379
x[38]  -0.25989 0.02047 0.0006472       0.002235
x[39]   0.38227 0.03308 0.0010459       0.003601
x[40]  -0.39932 0.03213 0.0010159       0.002808
x[41]   0.50141 0.04047 0.0012797       0.004289
x[42]  -0.19304 0.02286 0.0007229       0.002014
x[43]   0.43676 0.03514 0.0011113       0.003907
x[44]   0.21555 0.02463 0.0007789             NA
x[45]  -0.23356 0.02187 0.0006917       0.002170
x[46]   0.32952 0.04385 0.0013868       0.003705
x[47]  -0.32699 0.02688 0.0008502       0.002792
x[48]  -0.15477 0.01877 0.0005935       0.001864
x[49]   0.40915 0.03279 0.0010369       0.003471
x[50]  -0.38100 0.02707 0.0008560       0.002366
x[51]  -0.48218 0.04402 0.0013921       0.004045
```

```
x[52]   -0.21149 0.02008 0.0006350       0.001989
x[53]    0.30591 0.02769 0.0008756             NA
x[54]   -0.50657 0.04402 0.0013920       0.003642
x[55]   -0.46247 0.04096 0.0012952       0.003323
x[56]    0.30801 0.02777 0.0008782             NA
x[57]    0.26910 0.02778 0.0008785       0.002928
x[58]    0.25331 0.03195 0.0010103       0.002958
x[59]    0.41970 0.03480 0.0011004       0.003747
x[60]    0.37864 0.03308 0.0010461       0.003736
x[61]    0.57748 0.04441 0.0014042       0.005023
x[62]    0.32324 0.02909 0.0009198             NA
x[63]   -0.36724 0.02790 0.0008822       0.002545
x[64]    0.20904 0.02482 0.0007849             NA
x[65]    0.27582 0.02819 0.0008916             NA
x[66]    0.29709 0.02856 0.0009031       0.003177
x[67]    0.38486 0.03261 0.0010312       0.003524
x[68]    0.63280 0.05162 0.0016325       0.005602
x[69]   -0.38084 0.03060 0.0009676       0.003121
x[70]   -0.10062 0.01841 0.0005821       0.002013
x[71]   -0.44363 0.04518 0.0014286       0.003905
x[72]    0.11520 0.02131 0.0006739             NA
x[73]   -0.17355 0.01956 0.0006186       0.001974
x[74]    0.07068 0.01981 0.0006263       0.002243
x[75]    0.21392 0.02357 0.0007454       0.002556
x[76]    0.20176 0.02442 0.0007723             NA
x[77]    0.34928 0.02902 0.0009178       0.003212
x[78]   -0.23968 0.02067 0.0006535       0.002035
x[79]    0.21529 0.02450 0.0007747             NA
x[80]   -0.39220 0.03831 0.0012113       0.003246
x[81]   -0.34145 0.02688 0.0008500       0.002608
x[82]   -0.17929 0.01840 0.0005820       0.001821
x[83]   -0.25700 0.02063 0.0006525       0.002011
x[84]   -0.28720 0.02311 0.0007309       0.002193
x[85]   -0.39606 0.03027 0.0009572       0.002846
x[86]    0.22454 0.02523 0.0007978             NA
x[87]   -0.23925 0.02095 0.0006626       0.001902
x[88]    0.40070 0.03259 0.0010306       0.003524
x[89]   -0.30158 0.02460 0.0007780       0.002441
x[90]    0.22588 0.02416 0.0007639             NA
x[91]    0.22232 0.02523 0.0007980             NA
x[92]   -0.40139 0.03064 0.0009689       0.002679
x[93]   -0.32253 0.02532 0.0008006       0.002668
x[94]    0.32031 0.02921 0.0009238       0.003394
x[95]   -0.13922 0.01809 0.0005720       0.001947
x[96]    0.00887 0.01842 0.0005826       0.002070
x[97]    0.14876 0.02258 0.0007141             NA
x[98]    0.03582 0.01945 0.0006149             NA
x[99]    0.39067 0.03098 0.0009798       0.003326
x[100]  -0.12115 0.01797 0.0005682       0.001920
x[101]   0.09180 0.02144 0.0006780             NA
x[102]   0.25044 0.02555 0.0008079             NA

2. Quantiles for each variable:

              2.5%       25%       50%      75%     97.5%
x[1]    -0.3269554 -0.297316 -0.281766 -0.26356 -0.240940
x[2]    -0.3088036 -0.281371 -0.265322 -0.24675 -0.222399
```

```
x[3]   -0.4565632 -0.413353 -0.389952 -0.36869 -0.332624
x[4]    0.0727888  0.093464  0.111023  0.12861  0.150997
x[5]    0.1663932  0.188479  0.206693  0.22719  0.256671
x[6]    0.2475048  0.277234  0.297582  0.32126  0.355292
x[7]   -0.5160564 -0.452653 -0.426811 -0.40265 -0.364965
x[8]    0.1047815  0.124896  0.142979  0.16208  0.184175
x[9]   -0.2664531 -0.241794 -0.227995 -0.21327 -0.193667
x[10]  -0.4766981 -0.434652 -0.408718 -0.38486 -0.344222
x[11]   0.1459322  0.172140  0.190591  0.20861  0.231750
x[12]  -0.5662577 -0.508079 -0.481081 -0.45737 -0.418009
x[13]  -0.4715493 -0.434381 -0.410615 -0.38668 -0.350521
x[14]  -0.4270008 -0.394298 -0.375059 -0.35516 -0.323417
x[15]  -0.3236235 -0.296718 -0.280515 -0.26341 -0.238732
x[16]   0.1516637  0.176226  0.194723  0.21319  0.241650
x[17]   0.0881306  0.109834  0.127690  0.14696  0.166887
x[18]  -0.3013489 -0.274825 -0.259463 -0.24398 -0.216943
x[19]  -0.0580513 -0.038487 -0.024504 -0.01020  0.006309
x[20]  -0.3947890 -0.365740 -0.347677 -0.32996 -0.298851
x[21]   0.0960774  0.115061  0.132817  0.15205  0.173250
x[22]   0.2323612  0.258790  0.278376  0.30026  0.336722
x[23]  -0.2977299 -0.269603 -0.255037 -0.24084 -0.217241
x[24]   0.2503818  0.276854  0.296157  0.31795  0.349110
x[25]   0.1422502  0.167798  0.184840  0.20455  0.232523
x[26]  -0.4993107 -0.453833 -0.429574 -0.40630 -0.367500
x[27]  -0.4444377 -0.404196 -0.384834 -0.36511 -0.335433
x[28]   0.0002978  0.016146  0.032663  0.04918  0.064834
x[29]  -0.4107458 -0.380058 -0.362113 -0.34232 -0.310391
x[30]  -0.6063331 -0.532315 -0.501476 -0.47351 -0.425930
x[31]  -0.4976478 -0.447463 -0.421169 -0.40040 -0.360800
x[32]   0.2448392  0.272595  0.290368  0.31474  0.347437
x[33]  -0.3051652 -0.276753 -0.260916 -0.24364 -0.217212
x[34]  -0.5996374 -0.537015 -0.505559 -0.47692 -0.429342
x[35]  -0.2462507 -0.222108 -0.205554 -0.19184 -0.171344
x[36]  -0.5504782 -0.493104 -0.461840 -0.43762 -0.400808
x[37]   0.2267032  0.267194  0.290246  0.31712  0.362421
x[38]  -0.2987124 -0.274292 -0.259426 -0.24433 -0.222099
x[39]   0.3218716  0.358651  0.379915  0.40690  0.448126
x[40]  -0.4624382 -0.420855 -0.398855 -0.37696 -0.338126
x[41]   0.4262856  0.470993  0.498154  0.52862  0.582070
x[42]  -0.2374833 -0.209072 -0.193071 -0.17720 -0.149766
x[43]   0.3735102  0.408809  0.435320  0.46357  0.501237
x[44]   0.1731292  0.196754  0.214285  0.23314  0.262645
x[45]  -0.2747153 -0.249396 -0.233265 -0.21813 -0.193180
x[46]   0.2489218  0.298526  0.328473  0.35893  0.418382
x[47]  -0.3817779 -0.346161 -0.326173 -0.30697 -0.280426
x[48]  -0.1893692 -0.168965 -0.156168 -0.14040 -0.118497
x[49]   0.3486191  0.385368  0.407807  0.43195  0.471761
x[50]  -0.4304030 -0.400060 -0.382079 -0.36207 -0.326996
x[51]  -0.5704634 -0.509937 -0.482155 -0.45089 -0.401454
x[52]  -0.2495004 -0.225494 -0.211878 -0.19687 -0.175701
x[53]   0.2584088  0.285385  0.303890  0.32572  0.360872
x[54]  -0.6052537 -0.533115 -0.504662 -0.47599 -0.427790
x[55]  -0.5494753 -0.487124 -0.459198 -0.43306 -0.391208
x[56]   0.2577032  0.287415  0.306408  0.32785  0.361266
x[57]   0.2222066  0.248273  0.268001  0.29016  0.322979
x[58]   0.1910510  0.230929  0.252127  0.27563  0.318768
x[59]   0.3555928  0.393723  0.417298  0.44595  0.487260
```

```
x[60]    0.3159063  0.355332  0.377852  0.40287  0.440031
x[61]    0.4971975  0.545960  0.574924  0.60856  0.658455
x[62]    0.2694016  0.302103  0.321673  0.34476  0.379575
x[63]   -0.4230213 -0.386260 -0.366053 -0.34804 -0.315259
x[64]    0.1678766  0.188369  0.207812  0.22851  0.255736
x[65]    0.2270732  0.254510  0.272341  0.29713  0.334114
x[66]    0.2476465  0.275464  0.295857  0.31673  0.354523
x[67]    0.3283951  0.360646  0.383263  0.40749  0.448753
x[68]    0.5305639  0.595380  0.633684  0.67076  0.728693
x[69]   -0.4449550 -0.400356 -0.379795 -0.35799 -0.329084
x[70]   -0.1351236 -0.114354 -0.100157 -0.08620 -0.067962
x[71]   -0.5332813 -0.473771 -0.441623 -0.41098 -0.362055
x[72]    0.0773166  0.097952  0.115346  0.13379  0.150714
x[73]   -0.2127465 -0.187559 -0.173166 -0.15880 -0.137864
x[74]    0.0357180  0.054511  0.070073  0.08689  0.105472
x[75]    0.1715053  0.195623  0.211990  0.23124  0.257732
x[76]    0.1564269  0.183365  0.201654  0.22003  0.246631
x[77]    0.2997129  0.325502  0.346578  0.37104  0.404957
x[78]   -0.2828111 -0.253374 -0.239260 -0.22555 -0.199935
x[79]    0.1726569  0.195945  0.214601  0.23429  0.261456
x[80]   -0.4753719 -0.417755 -0.389500 -0.36604 -0.324400
x[81]   -0.3945387 -0.360330 -0.339349 -0.32272 -0.293754
x[82]   -0.2137400 -0.192340 -0.178870 -0.16645 -0.144679
x[83]   -0.2971594 -0.272272 -0.257426 -0.24229 -0.215344
x[84]   -0.3293774 -0.303427 -0.286984 -0.27107 -0.241819
x[85]   -0.4616623 -0.414766 -0.394043 -0.37621 -0.338599
x[86]    0.1807475  0.204639  0.223383  0.24428  0.270935
x[87]   -0.2821732 -0.254736 -0.238307 -0.22437 -0.200232
x[88]    0.3420190  0.375967  0.400162  0.42428  0.464612
x[89]   -0.3469191 -0.320434 -0.302778 -0.28322 -0.252992
x[90]    0.1818257  0.208350  0.225211  0.24331  0.272333
x[91]    0.1788438  0.203312  0.219841  0.24247  0.271227
x[92]   -0.4654018 -0.422113 -0.399485 -0.38049 -0.346203
x[93]   -0.3754045 -0.339965 -0.322115 -0.30357 -0.274804
x[94]    0.2686103  0.297563  0.318479  0.34277  0.375317
x[95]   -0.1747248 -0.152749 -0.139175 -0.12504 -0.107159
x[96]   -0.0228292 -0.006799  0.008177  0.02400  0.039769
x[97]    0.1093281  0.130607  0.149217  0.16684  0.187999
x[98]   -0.0002293  0.019712  0.035715  0.05216  0.068440
x[99]    0.3356843  0.368416  0.387315  0.41385  0.454635
x[100] -0.1550237 -0.134105 -0.120949 -0.10649 -0.088279
x[101]  0.0568285  0.072074  0.091602  0.11037  0.130351
x[102]  0.2072956  0.230500  0.248444  0.27004  0.301324
```

R Code

```
> comparisonData$jags <- apply(probitModel[[1]],2,mean)
> summary(comparisonData)
```

R output

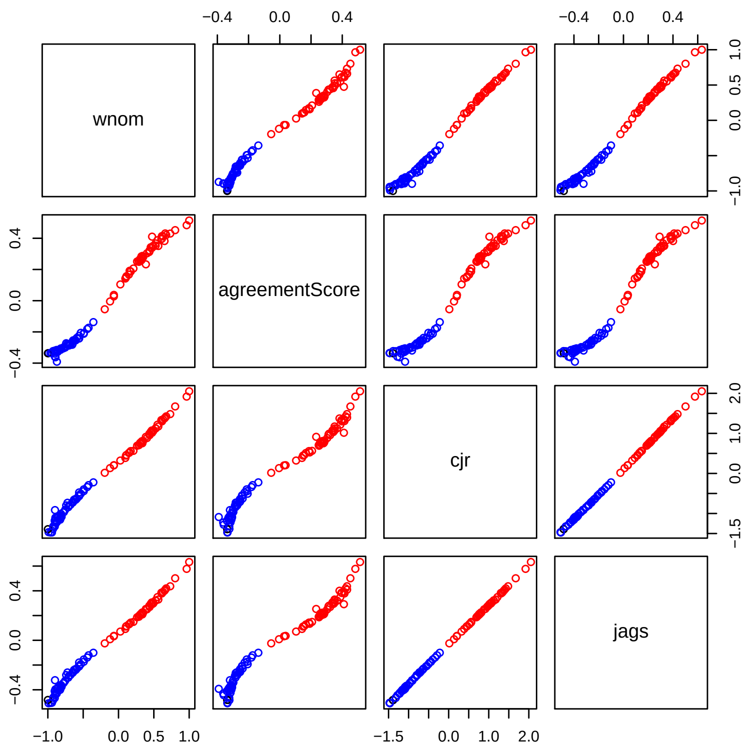
```
      wnom          agreementScore               cjr.V1
 Min.   :-1.0000   Min.   :-3.910e-01   Min.   :-1.47412175636e+00
 1st Qu.:-0.7799   1st Qu.:-3.042e-01   1st Qu.:-9.50918325398e-01
 Median :-0.2754   Median :-9.615e-02   Median :-1.02741310837e-01
 Mean   :-0.1781   Mean   : 1.823e-18   Mean   :-1.23726600860e-18
 3rd Qu.: 0.3990   3rd Qu.: 2.995e-01   3rd Qu.: 9.06801341107e-01
 Max.   : 1.0000   Max.   : 5.139e-01   Max.   : 2.05246994240e+00
      jags
 Min.   :-0.50875
 1st Qu.:-0.33784
```

```
Median :-0.06264
Mean   :-0.03432
3rd Qu.: 0.25259
Max.   : 0.63280
```

Now compare the different estimates that we have:

R Code

```
> pairs(comparisonData,col=col)
```




# References


Clinton, Joshua D., Simon Jackman and Douglas Rivers. 2004. "The Statistical Analysis of Roll Call Data." *American Political Science Review* 98:355--370.

Poole, Keith, Jeffrey Lewis, James Lo and Royce Carroll. 2007. *wnominate: WNOMINATE Roll Call Analysis Software.* R package version 0.93.

Poole, Keith T. 2005. *Spatial Models of Parliamentary Voting*. New York: Cambridge University Press.

Poole, Keith T. and Howard Rosenthal. 1997. *Congress: A Political-Economic History of Roll Call Voting*. New York: Oxford University Press.