Axes in the first U.S. senate, 1789-1791

J.A.Hartigan Yale University
September 2013

1 About the First Federal Congress (1789-1791)

from http://earlyrepublic.press.jhu.edu/about/aboutFFC.html

"All legislative Powers herein granted shall be vested in a Congress of the United States, which shall consist of a Senate and House of Representatives."

(United States Constitution, Article I, Section 1)

The critically important role of the First Federal Congress (FFC), 1789-1791, as the legislative body which began to implement and interpret the new Constitution of the United States is well recognized. The conception of the federal government occurred at the Federal Convention which drafted the U.S. Constitution in Philadelphia, but it was not until the First Federal Congress began to make decisions and pass enabling legislation that life was breathed into that government. The responsibility for the success or failure of the American Revolution rested with the members of this most important and productive Congress in U.S. History.

The significance of this Congress cannot be overstated. It acted as a virtual second sitting of the Federal Convention, addressing issues that the Convention found too potentially divisive to handle, such as the jurisdiction and structure of the federal judiciary, the funding of the federal and state Revolutionary War debts, and the location of the federal capital. The First Congress fleshed out the structure of the federal government outlined in the Constitution and provided stability for the new nation. Despite the difficult and divisive issues facing them, the members overcame their political and regional differences and left to the future a sturdy foundation on which a great nation could be built.

"In no nation, by no Legislature, was ever so much done in so short a period for the establishment of Government, Order, ... & general tranquility"

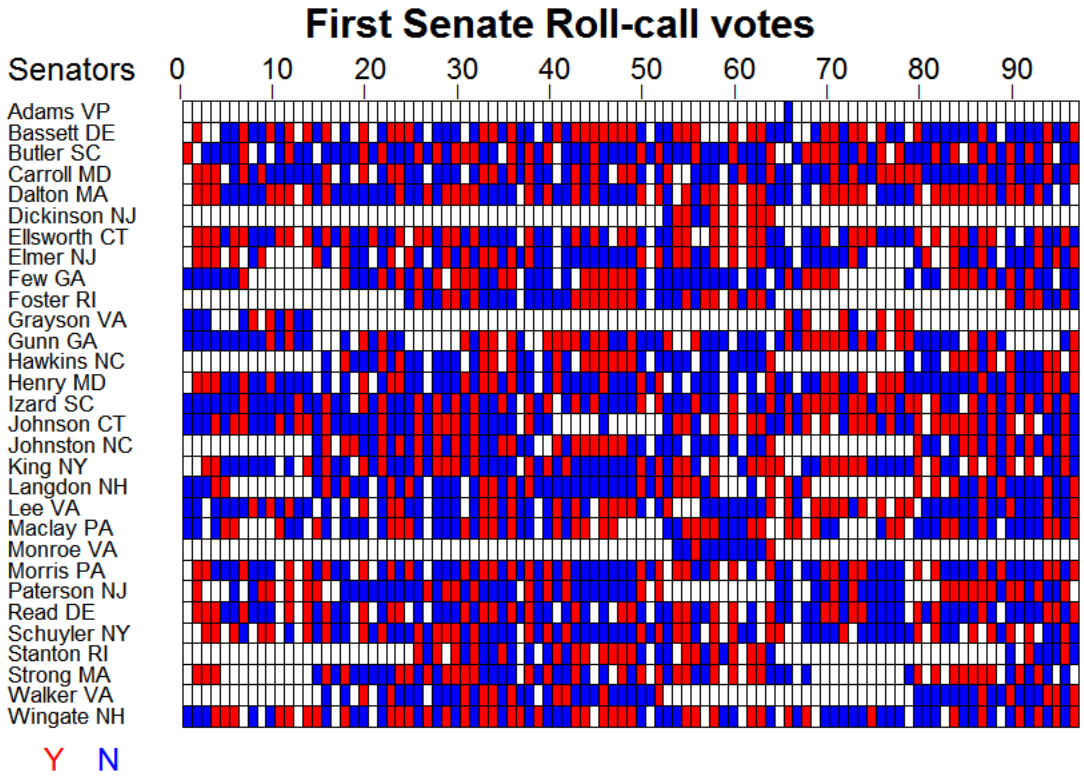
2 Identifying issues First Senate 1789-1791

We will identify the principal issues that were decided in the first Congress(4 March 1789- 3 March 1791), using the Yes and No votes recorded in The Senate Journal. Votes were recorded when required by one-fifth of the Senators present. There would have been votes on many of these issues in the House of Representatives. At this time, there were no formal parties, although there was regional alliances on some issues.

These votes are available on on-line in the Senate Journal. See the Data Preparation Section for the procedures used to obtain all the roll call votes in the first senate**.**

3 Display of votes

```
votes<-read.table("data/FirstSenateVotes",
                  header=T,as.is=T)
tiff("pictures/FirstSenate.tif", w=900, h=600)

# See the function section for Chplot
Chplot(votes,ch=c(".","Y","N"),
    col=c("white","red","blue"), ylab="Senators")
title(" First Senate Roll-call votes", cex.main=2.5)

dev.off()
```



We see Adams, the Vice President, only voted once, to break a tie. Three of the senators were absent most of the time, partly because their elections were delayed. And there are some signs of clustering in the votes 40-50, where a number of senators vote the same way six or seven votes in a row.

4 Replace missing values by modes :

Make matrix numerical:

```
nvotes <- matrix(NA,dim(votes)[1],dim(votes)[2])
nvotes[votes == "Y"] <- 1
nvotes[votes == "N"] <- 0
nvotes <- data.frame(nvotes)
rownames(nvotes) <- rownames(votes)
```

Drop the senators whose votes are mostly missing:

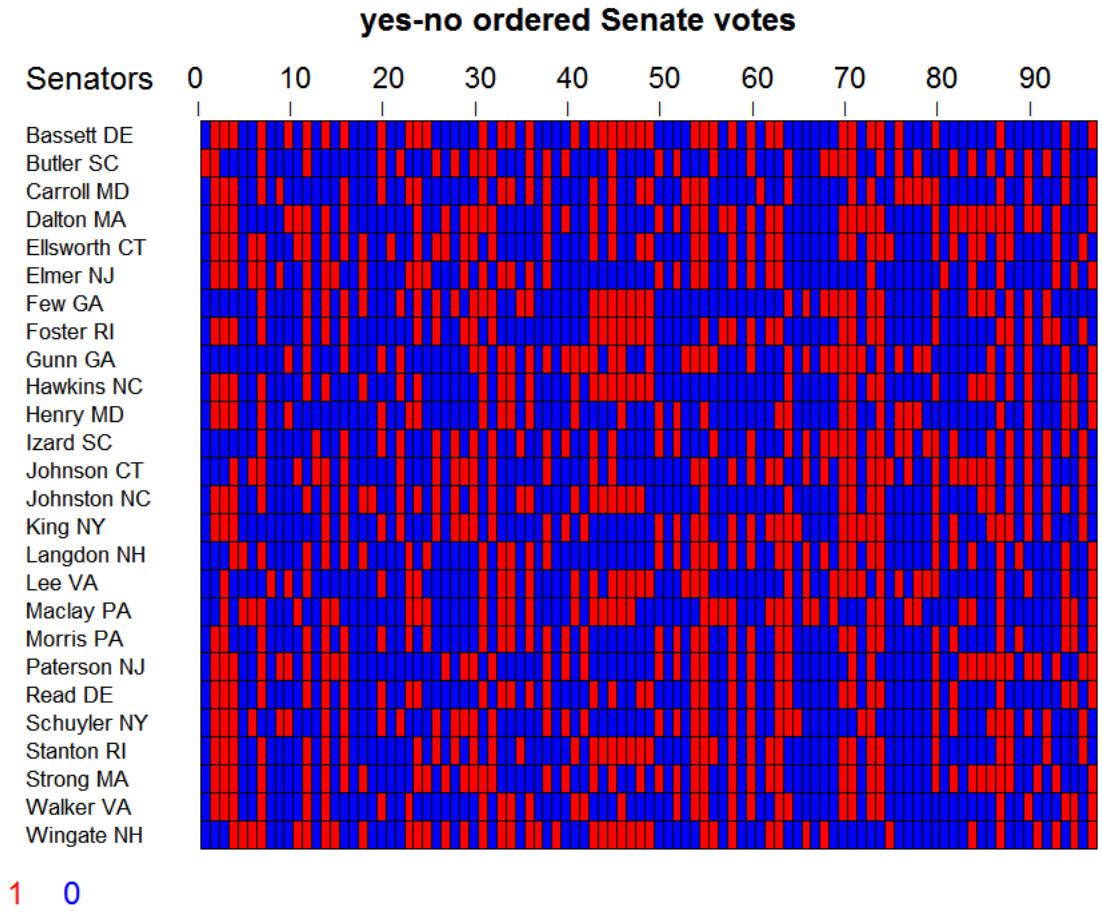
```
drop <-c("Adams VP","Dickinson NJ","Grayson VA",
       "Monroe VA")
votes <- votes[!(rownames(votes) %in% drop), ]
nvotes <- nvotes[!(rownames(nvotes) %in% drop), ]
```

Replace missing values by modal values:
(The modal value is 0 just when the mean is less than or equal to 0.5)

```
for( col in 1:dim(nvotes)[2]){
one <- mean(nvotes[, col], na.rm=T) > 0.5
nas <- is.na(nvotes[, col])
nvotes[nas &  one, col] <- 1
nvotes[nas & !one, col] <- 0
}
```

Order the Senators by their mean vote:

```
zmean <- function(x) mean(x[x != 0])
ryn <- apply( nvotes, 1, zmean)

tiff("pictures/FirstOrderedSenate.tif", w=900,
h=700)
Chplot(nvotes[order(ryn),], ch=c("1","0"),
     col=c("red","blue"), ylab="Senators")
title(" yes-no ordered Senate votes", cex.main=2)

dev.off()
```



After ordering by the mean number of Yes's, we see quite a number of rectangular blocks of the same color, indicating small blocks of senators and bills. See for example votes 43:49 about the seat of Government.

5 Specify Bill classifications

We need to know something about the roll call votes to interpret the clusters.

Look at codebooks for dates and descriptions:

```
des <- scan("data/Bills", w="", sep="\n")
head(des)
```

```
[1] "JULY 17, 1789  TO ESTABLISH THE JUDICIAL COURTS OF THE
UNITED STATES.                          "
[2] "JULY 18, 1789  FOR ESTABLISHING THE DEPARTMENT OF FOREIGN
AFFAIRS     "
[3] "JULY 18, 1789 TO CONCUR IN ESTABLISHING THE DEPARTMENT OF
FOREIGN AFFAIRS.                                    "
[4] "AUG. 4, 1789 TO ESTABLISH THE DEPARTMENT OF WAR       "
[5] "AUG. 4, 1789 TO PROVIDE FOR GOVERNMENT OF THE TERRITORY
NORTHWEST OF THE OHIO RIVER      "
[6] "AUG. 18, 1789 TO PROVIDE FOR THE EXPENSES OF NEGOTIATIONS
WITH THE INDIAN TRIBES"
```

```
# government seat bills, debt bills, establishing govt
bills
g <- c(17:21, 31:35, 39:56, 86, 87, 93:94)
d <- 57:80
e <- c(1:7, 11:13, 15:16, 88:89)
gd <-rep("Mis",97)
gd[g] <-"Gov"
gd[d] <- "Dbt"
gd[e] <- "Est"
```

6 Kmeans clusters of senators (3 clusters)

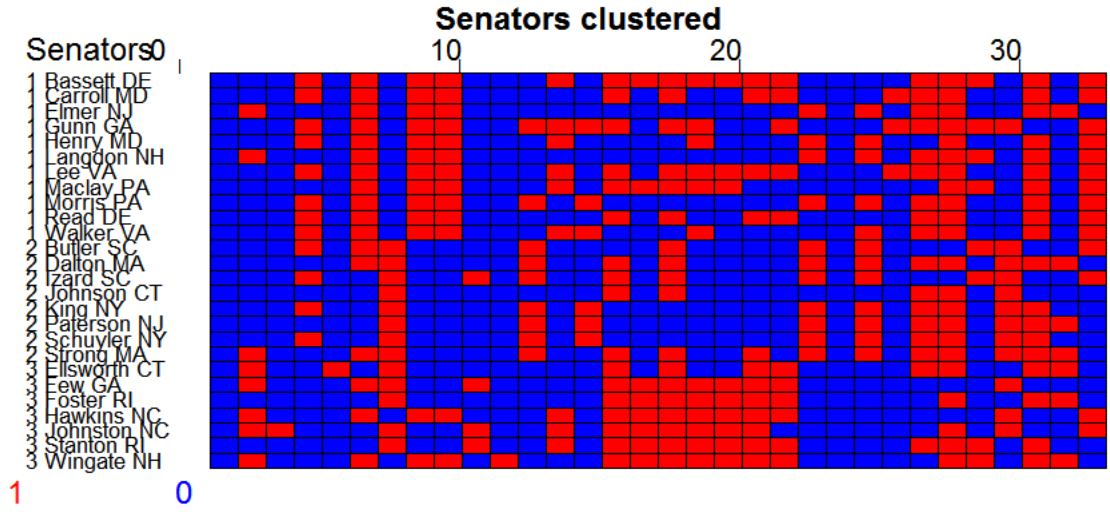
```
cluster <- Stablekm(nvotes, centers=3)
```

```
Total within sums of squares: 311.55
Cluster  1 : Bassett DE Carroll MD Elmer NJ Hawkins NC Henry MD Langdon NH Lee VA Maclay PA Morris PA Read DE Walker VA Wingate NH
Cluster  2 : Dalton MA Ellsworth CT Foster RI Johnson CT King NY Paterson NJ Schuyler NY Stanton RI Strong MA
Cluster  3 : Butler SC Few GA Gunn GA Izard SC Johnston NC
```

```
cord <- order(cluster)
orv <- nvotes[cord,]
rownames(orv) <- paste(sort(cluster), rownames(orv))
```

Plot Senators' votes in order of k means clusters:

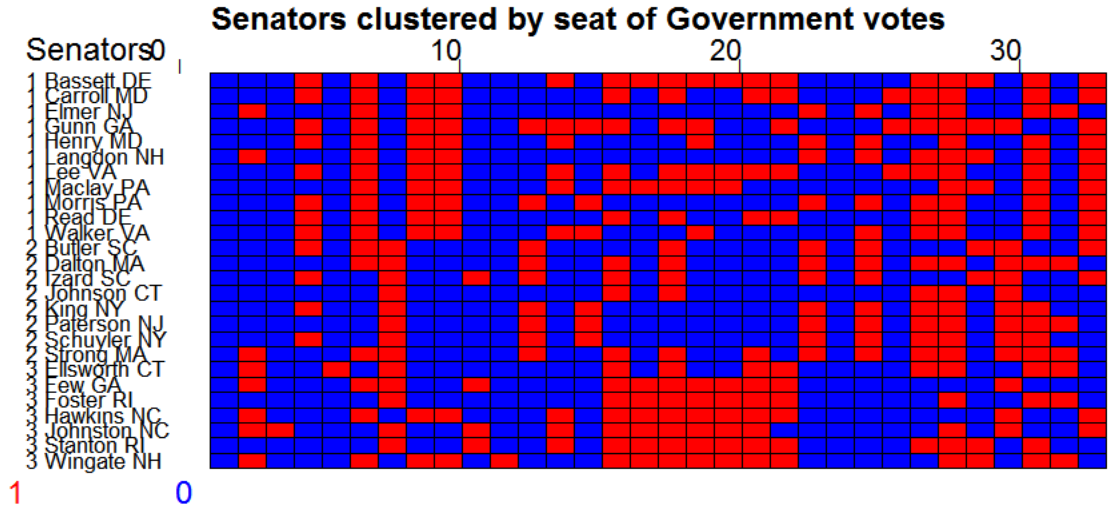
```
tiff("pictures/ClusteredSenators.tif", w=900, h=400)
Chplot(orvotes, ch=c("1","0"), col=c("red","blue"),
ylab="Senators")
title(" Senators clustered", cex.main=2)
dev.off()
```



The three clusters consist of MidAtlantic States, Northern States, and Southern States. Each of the clusters has a distinct pattern of voting on the seat of government bills, `17:21, 31:35, 39:56, 86, 87, 93:94`

Redo the clusters on the seat of government bills :

```
gnvotes <- nvotes[, g]
cluster <- Stablekm(gnvotes, centers=3)

Total within sums of squares: 95.24837662
Cluster  1 : Bassett DE Carroll MD Elmer NJ Gunn GA Henry MD
Langdon NH Lee VA Maclay PA Morris PA Read DE Walker VA
Cluster  2 : Butler SC Dalton MA Izard SC Johnson CT King NY
Paterson NJ Schuyler NY Strong MA
Cluster  3 : Ellsworth CT Few GA Foster RI Hawkins NC Johnston
NC Stanton RI Wingate NH

cord <- order(cluster);orvotes <- gnvotes[cord,]
rownames(orvotes) <- paste(sort(cluster),
                           rownames(orvotes))
tiff("pictures/ClusteredSenatorsGov.tif",w=900,
h=400)
Chplot(orvotes, ch=c("1","0"), col=c("red","blue"),
ylab="Senators")
title(" Senators clustered by seat of Government
votes", cex.main=2)
dev.off()
```



We see sharp clusters for the seat of government issues. The most numerous cluster consists of MidAtlantic Senators. The remaining two clusters are mixed Yankee and Southern.

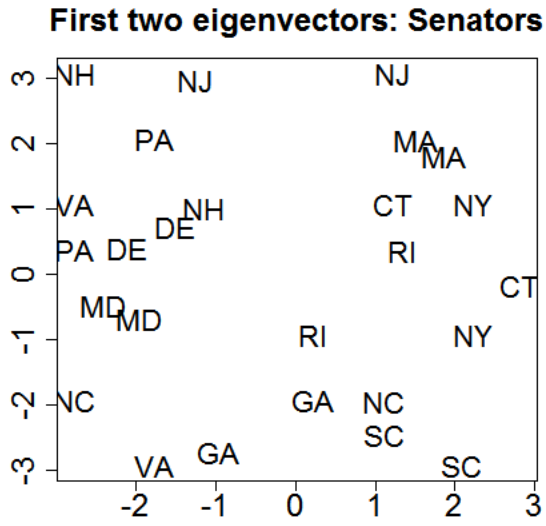
7 Principal Components

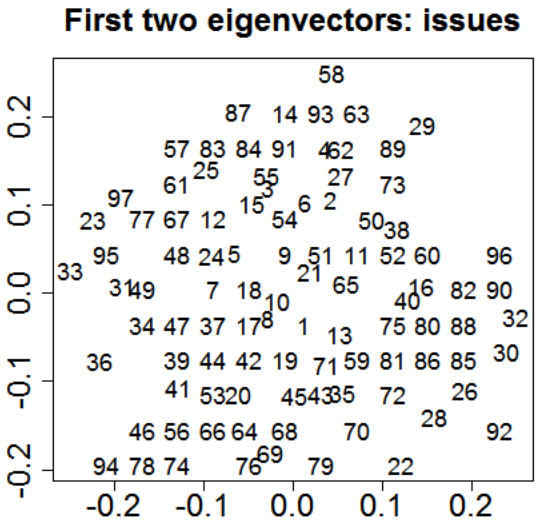
Pick state names off senators for plotting:

```
rstates<-unlist(
strsplit(rownames(votes),split=" ") )
rstates <- rstates[seq(2, length(rstates), 2)]
pv <- prcomp(nvotes)
```

Plot first two eigenvectors for senators and votes:

```
tiff("pictures/PrincComponentsStates.tif", w=900,
h=450)
par(mfrow=c(1,2))
xy <- Separate(pv$x[, 1], pv$x[, 2], 1)

plot(xy, pch="",cex.axis=2,xlab="",ylab="",
main="First two eigenvectors: Senators", cex.main=2)

text(xy, rstates, cex=1.8)
xy<-Separate(pv$rotation[,1],pv$rotation[,2],0.04)

plot(xy, pch="", cex.axis=2, xlab="", ylab="",
  main="First two eigenvectors: issues",
cex.main=2 )

text(xy[, 1], xy[, 2], 1:dim(votes)[2], cex=1.5)

dev.off()
```





The points in both pictures have been separated to reduce overlap. We see in the Senator plot the same clustering as appeared in k-means with a Southern cluster, a Yankee cluster, and a MidAtlantic cluster plus Virginia and New Hampshire. The votes having large first eigenvector absolute values on the issues plot, in the 30's, 80's, and 90's , corresponding to government debt bills, on which the thrifty Yankees and the spendthrifty MidAtlantic States disagree.

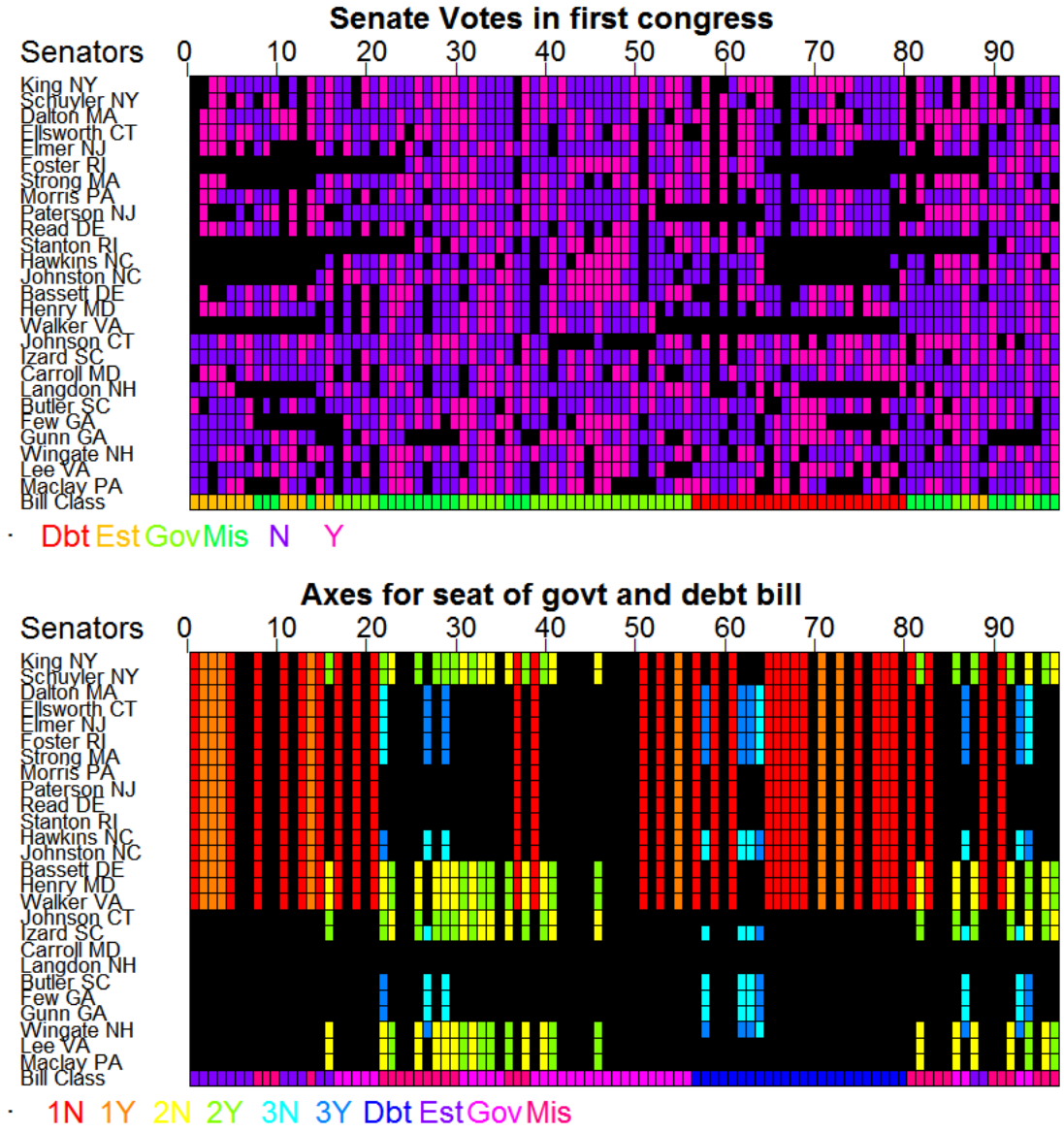
8 Axes in the first Senate

Each axis consists of two opposing clusters of Senators, together with the issues on which they disagree. The computation finds the closest approximation to the data in a sum of rank 1 matrices corresponding to the axes, each an outer product of vectors taking the values (-1, 0, 1). For the senators the non-zero values define the two opposing clusters. For the issues, the non-zero values define a negative or affirmative vote on the issue.
The fit is constrained so that each member of an axis agrees in his votes with a threshold of .75 of the votes on each issue in the axis.
Substitute 0.5 for missing votes to avoid overestimating agreement in the missing votes:

```
mnvotes <- nvotes
mnvotes[is.na(votes)] <- 0.5
u <-Axes(2*mnvotes-1, iter=10, naxes=3,
threshold=.75)
```

The bill classifications were previously stored in gd.

```
tiff("pictures/govdebt.tif", w=900, h=890)
par(mfrow=c(2,1))

# plot original data for comparison with model
rvoteg <- rbind(votes[order(u$rsort),],gd)
rownames(rvoteg) <-
c(rownames(votes)[order(u$rsort)], "Bill Class")

Chplot(rvoteg, ylab="Senators")
title("Senate Votes in first congress", cex.main=2)

# plot fitted model showing axes
fitg <- rbind(u$fit[order(u$rsort), ], gd)
rownames(fitg) <- c(rownames(votes)[order(u$rsort)],
"Bill Class")
Chplot(fitg, ylab="Senators")
title("Axes for seat of govt and debt bill",
cex.main=2)
dev.off()
```



Axis 1: Wide range of bills

Yeas: Northern States

Nays: None

Axis 2: Seat of Government and Establishing Depts

Ayes: MidAtlantic and VA

Nays: NY CT SC

Axis 3: Wide ranging bills

Ayes: South

Nays: Yankees

9 Conclusions

There is a rough division into Yankee, MidAtlantic, and South in the Senators. The Yankees combine with the South, for deciding the seat of Government; it went temporarily to Philadelphia. There is a combination of Yankees and MidAtlantic Senators, with no unified opposition from the South, for deciding the Government Debt and Establishment issues.

10 Data Preparation

10.1 Collect journal issues containing votes between MARCH 4, 1789 and MARCH 3, 1791

The following URL lists the dates when the Senate met between 1789 and 1791, and lists the journal issues containing the report of senate activities on those dates.

```
session<-
"http://memory.loc.gov/cgi-bin/query/r?ammem/hlaw:@
field%28DOCID+@lit%28sj001T000%29%29: "
```

Paste this link into the web address line to view the web page. Right click on the web page and then select page source to see the html code that controls the web page; further click on the url's in the page code to get their page codes, which contain the actual voting data.

```
ss <- scan(session, w="", sep = "\n")
ss[15:17]

[1] "<a
href=\"/cgi-bin/query/r?ammem/hlaw:@field(DOCID+@lit(sj0013
))\">JOURNAL OF THE FIRST SESSION OF THE SENATE OF THE UNITED
STATES,</a><br>"
[2] "<a
href=\"/cgi-bin/query/r?ammem/hlaw:@field(DOCID+@lit(sj0014
))\">WEDNESDAY, MARCH 4, 1789.</a><br>"
[3] "<a
href=\"/cgi-bin/query/r?ammem/hlaw:@field(DOCID+@lit(sj0015
))\">WEDNESDAY, MARCH 11, 1789.</a><br>"
```

For example, line 2 identifies an URL whose page code contains data for March 4, 1789.

The following regular expression identifies a date such as MARCH 4, 1789, (capitals, comma or space, number, comma, space, number):

```
expdate <- "[A-Z]+[ |,]+[0-9]+, [0-9]+"
```

Find lines in the journal corresponding to first congress dates:

```
ssuse <- ss[grepl(expdate, ss)]
wlastdate <- max( grep("MARCH 3, 1791", ssuse) )
ssuse <- ssuse[1:wlastdate]
head(ssuse, 3)
```

```
[1] "<a
href=\"/cgi-bin/query/r?ammem/hlaw:@field(DOCID+@lit(sj0014
))\">WEDNESDAY, MARCH 4, 1789.</a><br>"
[2] "<a
href=\"/cgi-bin/query/r?ammem/hlaw:@field(DOCID+@lit(sj0015
))\">WEDNESDAY, MARCH 11, 1789.</a><br>"
[3] "<a
href=\"/cgi-bin/query/r?ammem/hlaw:@field(DOCID+@lit(sj0016
))\">THURSDAY, MARCH 12, 1789.</a><br>"
```

```
dates <- regmatches(ssuse, regexpr(expdate, ssuse))
head(dates, 3)
```

```
[1] "MARCH 4, 1789"  "MARCH 11, 1789" "MARCH 12, 1789"
```

```
tail(dates, 3)
```

```
[1] "MARCH 2, 1791" "MARCH 3, 1791" "MARCH 3, 1791"
```

This is the correct range of dates for the first congress.

Identify the codes for the journal issues containing the senate reports for each date:

```
codes <-regmatches( ssuse,
regexpr("sj00[0-9][0-9]+", ssuse) )
cat("codes:", head(codes), "")
```

```
codes: sj0014 sj0015 sj0016 sj0017 sj0018 sj0019
```

10.2 Run through all the journal issues corresponding to meeting dates, and combine all the reports for those dates, 1789-1791:

```
if (FALSE){
base <-
"http://memory.loc.gov/cgi-bin/query/r?ammem/hlaw:@
field(DOCID+@lit("
journal <- ""
for (code in codes){
file <- paste( base, code, ")):", sep="")
s <- scan(file, w="", sep="\n", quiet=T)
# add in the dates before each new section for later
journal <- c(journal, dates[codes==code], s)
}
cat(journal, file="data/journal", sep="\n")
}
```

The previous step is time consuming, and requires repeated accessing of many web files, so we bypassed it to read in the data from the collected journals saved in a previous run.

```
journal <- scan("data/journal", w="", sep="\n")
expdate <- "[A-Z]+[ |,]+[0-9]+, [0-9]+"
head(journal, 5)
```

```
[1] NA
[2] "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01
Transitional//EN\">"
[3] "<HTML>"
[4] "\'5ct<HEAD>"
[5] "\'5ct\'5ct<TITLE><a
href=\"/ammem/amlaw/lwsj.html\">Senate Journal</a>
--WEDNESDAY, MARCH 4, 1789.</TITLE>"
```

10.3 Identify voting paragraphs.

General rule: whatever can go wrong, will go wrong. You must be deeply skeptical that you have identified the voting lines correctly, and must constantly print out checks that you have done what you intended. You will be surprised.

Change "the Vice President" to Adams, who votes only to break ties:

```
journal <- gsub("the Vice President", "Adams",
journal)
```

Identify different forms of Yeas and Nays:

```
Yeas <- grep("Yeas\.|Yeas-", journal)
Nays <- grep("Nays\.|Nay\.|Nays-|Nay,", journal)
```

A typical vote:

```
journal[Yeas[1]+ (0:1)]
```

```
[1] "<p>Yeas.--Messrs. Bassett, Carroll, Dalton, Ellsworth,
Elmer, Few, Gunn, Henry, Johnson, Izard, Morris, Paterson,
Read, and Strong.</p>"
[2] "<p>Nays.--Messrs. Butler, Grayson, Langdon, Lee, Maclay,
and Wingate.</p>"
```

```
cat("Yeas: ", Yeas, "")
```

```
Yeas:  3012 3044 3060 3585 3599 4121 4323 4465 4470 4481 4629
4686 4729 4786 4973 5353 5508 5519 5529 5573 5647 6371 9125 9195
9301 9309 9423 9429 9433 9487 9821 9833 9839 9845 10097 10382
10536 10593 10610 10614 10623 10629 10633 10637 10643 10684
10688 10695 10699 10703 10707 10771 10777 10781 10831 10837
11257 11274 11282 11286 11359 11443 11451 11455 11459 11465
11469 11473 11554 11603 11835 11840 11847 11855 11862 11866
11870 11875 11881 13339 13895 14056 14060 14552 14948 15257
15315 15345 15409 15545 15596 15730 15744 15750 15762 16231
16393
```

```
cat("Nays: ", Nays, "")
```

```
Nays:  3013 3045 3061 3586 3600 4122 4324 4466 4471 4484 4630
4687 4730 4787 4974 5354 5509 5520 5530 5574 5648 6372 9126 9196
9304 9310 9426 9430 9434 9490 9822 9834 9840 9846 10100 10383
10537 10594 10611 10615 10624 10630 10634 10638 10644 10685
10689 10696 10700 10704 10708 10772 10778 10782 10832 10838
11258 11275 11283 11287 11360 11444 11452 11456 11460 11466
11470 11474 11555 11604 11836 11841 11848 11858 11863 11867
11871 11876 11882 13340 13898 14057 14061 14553 14949 15258
15316 15346 15410 15546 15597 15731 15747 15751 15763 16232
16394
```

We check that the Nays lines are the Yeas lines plus 1.

10.4 Combine dates and voting records

Locate the dates appearing in the collection:

```
dates <- grep(expdate,substr(journal,1,20))
head(dates)
```

```
integer(0)
```

```
ly <- length(Yeas)
firstdate <- rep(0,ly)
datevote <- rep(0, 3*ly)
```

Find date just before vote, combine dates and votes:

```
for( i in 1:ly) firstdate[i] <- max(dates[
dates<Yeas[i] ])
index <- seq(1, 3*ly, 3)
datevote[index] <- firstdate
datevote[index+1] <- Yeas
datevote[index+2] <- Nays
jd <- journal[datevote]
head(jd)
```

```
[1] NA
[2] "<p>Yeas.--Messrs. Bassett, Carroll, Dalton, Ellsworth,
Elmer, Few, Gunn, Henry, Johnson, Izard, Morris, Paterson,
Read, and Strong.</p>"
[3] "<p>Nays.--Messrs. Butler, Grayson, Langdon, Lee, Maclay,
and Wingate.</p>"
[4] NA
[5] "<p>Yeas.--Messrs. Few, Grayson, Gunn, Johnson, Izard,
Langdon, Lee, Maclay, and Wingate.</p>"
[6] "<p>Nays.--Messrs. Bassett, Carroll, Dalton, Elmer, Henry,
Morris, Paterson, Read, Strong, and Adams.</p>"
```

Keep separate records of date indices and votes:

```
jdates <-grepl(expdate,substr(jd,1,20))
jvotes <-!jdates
jdv <- jd[!jdates]
```

10.5 Correct character anomalies in voting lists

Unify separation characters in vote list to be blank:

```
jdv <- gsub(";|\.|,| and ", " ", jdv)
jdv <- gsub("Mr", "Messrs", jdv)
```

Pick out voter list on each line:

```
gents <- regexpr("Messrs[A-Za-z ]+", jdv)
jdm <- regmatches(jdv, gents)
```

Eliminate Messrs:

```
jdm <- gsub("Messrs", " ", jdm)
head(jdm, 3)
```

```
[1] "   Bassett  Carroll  Dalton  Ellsworth  Elmer  Few
Gunn  Henry  Johnson  Izard  Morris  Paterson  Read  Strong
"
[2] "   Butler  Grayson  Langdon  Lee  Maclay  Wingate "
[3] "   Few  Grayson  Gunn  Johnson  Izard  Langdon  Lee
Maclay  Wingate "
```

Unify spellings:

```
jdm <- gsub("sch","Sch",jdm)
jdm <- gsub("Basserr|Basset |Basseth|Basett",
"Bassett", jdm)
jdm <- gsub("Long|Lung","Lang", jdm)
jdm <- gsub("EI","El", jdm)
jdm <- gsub("Stau","Stan", jdm)
jdm <- gsub("Henrys","Henry",jdm)
jdm <- gsub("Fosters","Foster",jdm)
jdm <- gsub("Dickison","Dickinson",jdm)
jdm <- gsub("O","o",jdm)
jdm <- gsub("Bead|Reed|Head|Leard", "Read", jdm)
jdm <- gsub("Poster", "Foster", jdm)
jdm <- gsub("Buffer", "Butler", jdm)
jdm <- gsub("Herris", "Morris", jdm)
```

Collect all the names, separate by single blank:

```
collect <- NULL
for( i in 1:length(jdm)) collect <- c(collect, jdm[i])
collect <- gsub("  ", " ",collect)
collect <- unique( unlist( strsplit(collect, " ") ) )
head(collect)
```

```
[1] ""          "Bassett"   "Carroll"   "Dalton"
[5] "Ellsworth" "Elmer"
```

Check names against michigan code books:

```
so1<-substr(scan("data/da9822.s01.dat", w="",
sep="\n"), 26, 34)
sort( unique(so1) )
```

```
[1] "BASSETT  " "BUTLER   " "CARROLL  " "DALTON   "
 [5] "DICKINSON" "ELLSWORTH" "ELMER    " "FEW      "
 [9] "FOSTER   " "GRAYSON  " "GUNN     " "HAWKINS  "
[13] "HENRY    " "IZARD    " "JOHNSON  " "JOHNSTON "
[17] "KING     " "LANGDON  " "LEE      " "MACLAY   "
[21] "MONROE   " "MORRIS   " "PATERSON " "READ     "
[25] "SCHUYLER " "STANTON  " "STRONG   " "WALKER   "
[29] "WINGATE  "
```

```
collect <- sort(collect[collect != ""])
print(collect)
```

```
[1] "Adams"     "Bassett"   "Butler"    "Carroll"
 [5] "Dalton"    "Dickinson" "Ellsworth" "Elmer"
 [9] "Few"       "Foster"    "Grayson"   "Gunn"
[13] "Hawkins"   "Henry"     "Izard"     "Johnson"
[17] "Johnston"  "King"      "Langdon"   "Lee"
[21] "Maclay"    "Monroe"    "Morris"    "Paterson"
[25] "Read"      "Schuyler"  "Stanton"   "Strong"
[29] "Walker"    "Wingate"
```

10.6 Insert states from codebook

The codebook information matches state with senator:

```
cs <- substr(scan("data/da9822.s01.dat", w="",
sep="\n"),10,34)
css <- substr(cs, 17, 26)
print(css)
```

```
[1] "JOHNSON  " "ELLSWORTH" "BASSETT  " "READ     "
 [5] "FEW      " "GUNN     " "CARROLL  " "HENRY    "
 [9] "DALTON   " "STRONG   " "LANGDON  " "WINGATE  "
[13] "ELMER    " "PATERSON " "DICKINSON" "SCHUYLER "
[17] "KING     " "HAWKINS  " "JOHNSTON " "MACLAY   "
[21] "MORRIS   " "STANTON  " "FOSTER   " "BUTLER   "
[25] "IZARD    " "LEE      " "GRAYSON  " "WALKER   "
[29] "MONROE   "
```

Pick off state names in the order of the senators:

```
substr(cs[order(css)], 4, 10)
```

```
[1] "DELAWAR" "SOUTH C" "MARYLAN" "MASSACH" "NEW JER"
 [6] "CONNECT" "NEW JER" "GEORGIA" "RHODE I" "VIRGINI"
[11] "GEORGIA" "NORTH C" "MARYLAN" "SOUTH C" "CONNECT"
[16] "NORTH C" "NEW YOR" "NEW HAM" "VIRGINI" "PENNSYL"
[21] "VIRGINI" "PENNSYL" "NEW JER" "DELAWAR" "NEW YOR"
[26] "RHODE I" "MASSACH" "VIRGINI" "NEW HAM"
```

Enter the State abbreviations in the right order by hand:

```
states <- "VP DE SC MD MA NJ CT NJ GA RI VA GA NC MD
SC CT NC NY NH VA PA VA PA NJ DE NY RI MA VA NH"
states <-unlist(strsplit(states," "))
print(states)
```

```
[1] "VP" "DE" "SC" "MD" "MA" "NJ" "CT" "NJ" "GA" "RI" "VA"
[12] "GA" "NC" "MD" "SC" "CT" "NC" "NY" "NH" "VA" "PA" "VA"
[23] "PA" "NJ" "DE" "NY" "RI" "MA" "VA" "NH"
```

This is an easier way to enter the states data than laboriously concatenating all the quoted states.

```
length(states)
```

```
[1] 30
```

```
length(collect)
```

```
[1] 30
```

```
senators<- paste(collect,states,sep=" ")
```

10.7 Make data frame with Yea and Nay votes for each issue and and Senator

```
jd[jvotes] <- jdm
jd <- gsub("  "," ",jd)
head(jd, 3)
```

```
[1] "  Bassett Carroll Dalton Ellsworth Elmer Few Gunn Henry
Johnson Izard Morris Paterson Read Strong "
[2] "  Butler Grayson Langdon Lee Maclay Wingate "
[3] "  Few Grayson Gunn Johnson Izard Langdon Lee Maclay Wingate
"
```

```
tail(jd, 3)
```

```
[1] "  Butler Dalton Ellsworth Few Foster Johnson Johnston
Izard King Paterson Schuyler Stanton Strong "
[2] "  Bassett Carroll Elmer Gunn Hawkins Henry Langdon Lee
Maclay Morris Read Walker Wingate "
[3] "  Butler Ellsworth Few Foster Johnson Johnston Izard King
Schuyler Stanton "
```

```
votes <- data.frame(matrix(".", nrow=
length(collect),
            ncol=length(jd)/3), stringsAsFactors=F )
```

Convert jd into Y and N votes:

```
for ( cols in 1:dim(votes)[2]){
for ( rows in 1:dim(votes)[1] ){
Yeasnames<-unlist(
strsplit(jd[seq(2,length(jd),3)][cols], split=" "))
if (collect[rows] %in% Yeasnames) votes[rows, cols] <-
"Y"
Naysnames<-unlist(
strsplit(jd[seq(3,length(jd),3)][cols],split=" ") )
if ( collect[rows] %in% Naysnames) votes[rows, cols]
<- "N"
}
```

```
}

rownames(votes) <- senators
votedates <- jd[jdates]
write.table(votes,"data/FirstSenateVotes")
votes[1:5, 1:10]
```

```
          X1 X2 X3 X4 X5 X6 X7 X8 X9 X10
Adams VP   .  .  .  .  .  .  .  .  .   .
Bassett DE .  Y  .  .  N  N  Y  N  N   Y
Butler SC  Y  .  N  N  N  N  Y  .  N   .
Carroll MD .  Y  Y  Y  .  N  Y  N  Y   N
Dalton MA  .  Y  Y  Y  N  N  N  N  N   Y
```

11 Functions

```
Stablekm <- function(data, iter=100, centers=2){
# does stable kmeans by repeating calculation,
reordering cluster allocation by size; Find best set
in iter tries, mainly to make sure you get the same
clusters every time.

best <- 10^20
for (it in 1:iter){
km <- kmeans(data, centers=centers)
if (km$tot.withinss < best){
  cluster <- km$cluster
  best <- km$tot.withinss
}
}
cat("Total within sums of squares:", best, "\n")

#Order the clusters by cluster size:
size <- table(cluster)
ordcluster <- rank(-size)[cluster]
names(ordcluster) <- names(cluster)

for(clus in 1:centers)
cat("Cluster ", clus, ":",
names(ordcluster)[ordcluster==clus], "\n")

return(ordcluster)
}
```

```
Separate <- function(x,y,d){
# Separate each pair of x,y by at least d in x or y
direction
#working grid

if(d==0) return(list(x=x,y=y))
nrow <- round( (max(y) - min(y))/d ) + 1
ncol <- round( (max(x) - min(x))/d ) + 1
if(nrow*ncol == 0) return(" x or y have 0 range")
if(length(x) != length(y)) return(" x and y of
different length")

wg <- matrix(T, nrow, ncol)

# assign x and y one at a time
xx <- x
yy <- y
for(i in 1:length(x)){
ix <- round((x[i] - min(x))/d) + 1
iy <- round((y[i] - min(y))/d) + 1

# dont search if a place is empty
good <- wg[iy, ix]
if(good) wg[iy,ix] <- F
if(good) next

# find closest empty spot
for( k in 1: max(nrow, ncol)){
if(k == max(nrow, ncol)) return(" can't separate by d")
  iys <- (1:nrow)[(abs((1:nrow)-iy) == k)]
  ixs <- (1:ncol)[(abs((1:ncol)-ix) == k)]

# alternate choices so shifts are unbiased overall
  if(ix %/% 2 == 0) ixs <- ixs[length(ixs):1]
  if(iy %/% 2 == 0) iys <- iys[length(iys):1]

  if(sum(wg[iys, ixs]) > 0) break
```

```
}
# find the best x direction to move point
for(ix in ixs)
  if(sum(wg[iys, ix])>0) break

iy <- min(which( wg[iys, ix] ))
iy <- iys[iy]

# move point to new position, make wg matrix false there
wg[iy, ix] <- F
xx[i] <- min(x) + (ix-1)*d
yy[i] <- min(y) + (iy-1)*d
}

return(cbind(xx,yy))
}
```

```
Chplot <- function(data, ch="", col="", ylab=""){
# plots a matrix of character data
# use ch for list of characters, col for corresponding
colors

par(mar = c(1,1,2,1))

# make up colors if not specified
dv <- as.vector(as.matrix(data))
udv <- sort(unique(dv))
if (length(ch) == 1)
col <- c(1,rainbow(length(udv)+1))[-c(6, 7)]
if(length(ch)==1) ch <- udv
nrows <- dim(data)[1]
ncols <- dim(data)[2]

# plot framework
plot(1:ncols, (1:ncols) * nrows/ncols, pch="",
 xlab="", ylab = "", axes=F,
 ylim =c(0 ,nrows+3), xlim=c(-ncols/5, ncols))

# insert rectangles in x,y positions for each character
for ( chs in ch){
dvc <- which( dv == chs & !is.na(dv) ) - 1
x <- dvc %/% nrows + 1
y <- nrows - dvc %% nrows
index <- which(chs ==ch)
rect(x - 0.5, y - 0.5, x + 0.5, y + 0.5,
     col=col[index], lwd=0)

# legend for colors
```

```
text(-ncols/5+6*(index-1), -1 , chs,
     col=col[index], cex=2, xpd=T)
}

# various titles and labels
text( seq(0, ncols, 10)-2, rep(nrows+2, ncols %/% 10),
  seq(0, ncols, 10), pos=4, cex=1.8)
text( seq(0, ncols, 10)-1, rep(nrows+1, ncols %/% 10),
  rep("|", ncols %/% 10), pos=4)
text(rep(-ncols/5, nrows), nrows:1, rownames(data),
  pos=4, cex=1.4)
text(-ncols/5, nrows+2, ylab, pos=4, cex=2)

# restore margins
par(mar = c(5,4,4,2))

invisible()
}
```

```
Axes<-function(data,naxes=2,iter=30,threshold=.80){
# constructs Axes for votes, with Y=1 N=-1 missing =
0   Each Axis is a rank 1 matrix taking values 1, -1,
and 0. The Axes might overlap. The model M is the sum
of Axes, and the fit minimizes sum (M- D)^2) subject
to the constraint that  a senator is included in Axis
if his agreement with Axis exceeds threshold. The
procedure is similar to a singular value decomposition
in which the eigenvectors are constrained to take
values -1, 0 or 1 and the eigenvalues are 1.

thr <- threshold

# truncate function
truncate <- function(data, thr) {
data[data > thr] <- 1
data[data < -thr] <- -1
data[abs(data) <= thr] <- 0
return(data)
}

data <- as.matrix(data)

# begin with an svd decomposition
bestvalue <- sum(data^2)
svdd <- svd(data, nu=naxes, nv=naxes)

if(naxes > 1)
 rb <- truncate(svdd$u %*%
diag(sqrt(svdd$d)[1:naxes]), thr)
if(naxes ==1)
   rb <- truncate(svdd$u * sqrt(svdd$d), thr)
for(it in 1:iter){
 if (det( t(rb) %*% rb ) == 0)
   return(" data of rank less than Axes")
cb<-t ( truncate(solve(t(rb)%*%rb) %*% t(rb) %*%
               data,thr ) )

rb<-truncate(data %*% cb %*% solve(t(cb) %*% cb),thr)
```

```
value <-(sum( (data- rb %*% t(cb))^2 ) )
if(value >= bestvalue) break
bestvalue <- value
}

# sort so that yes's in earliest Axes come first
rsort <- -rb %*% 10^( -(1:naxes) )
csort <- -cb %*% 10^(-(1:naxes) )

# show fitted Axes in characters
fit <-  matrix(".",dim(data)[1], dim(data)[2])

for( b in naxes:1){
nfit <- rb[, b] %*% t(cb[, b])
fit[nfit ==  1] <- paste(b, "Y", sep="")
fit[nfit == -1] <- paste(b, "N", sep="")
}

fit <- data.frame(fit,stringsAsFactors = FALSE)
rownames(fit) <- rownames(data)
return( list(rb=rb, cb=cb, value=bestvalue,
        csort=csort, rsort=rsort, fit=fit) )

}
```