# Memo on second-digit tests done on precinct counts for Democratic Senate primary in South Carolina, 2010

Walter R. Mebane, Jr.

12 June 2010

I computed second-digit Benford's Law (2BL) test statistics using precinct vote count data from the South Carolina 2010 Democratic primary election for U.S. Senate. Michael Miller obtained the data from a South Carolina website and gave them to me. Names and sizes of the data source files are shown below.

The two candidates are Alvin M. Greene and Vic Rawl. The victory by Greene surprised many observers.

Some background on 2BL tests is in Mebane (2010a), currently being revised as Mebane (2010b). The latter paper emphasizes the effect gerrymandering and turnout can have on the digits of vote counts, in a way not discussed in the former paper.

Tests for the second digits of vote counts come in two forms. One uses a Pearson chi-squared statistic and is tied to Benford's Law: $X^2_{2BL} = \sum_{j=0}^{9}(n_j - Nr_j)^2/(Nr_j)$, where $N$ is the number of vote counts of 10 or greater (so there is a second digit), $n_j$ is the number having second digit $j$ and $r_j$ is given by the Benford's Law formula. If the counts whose digits are being tested are statistically independent, then this statistic should be compared to the chi-squared distribution with nine degrees of freedom. In this case a single test statistic must be larger than 16.9 to indicate a statistically significant departure from the Benford's Law expectation.

The second statistic is the mean of the second digits, denoted $\hat{j}$. If the counts' second-digits follow Benford's Law, then the value expected for the second-digit mean is $\bar{j} = \sum_{j=0}^{9} jr_j = 4.187$.

I computed statistics separately for the number of registered voters and for each candidate in three sets: for all ballots together; for non-absentee ballots; and only for absentee ballots. The program that computes the statistics is shown in the file (`tbenf3.R`) included below. Detailed results for all three sets of ballots appear in file `tbenf3.Rout`, also included below. A table of the results for the second and third sets of ballots follows.

| Election Day | $N$ | $X^2_{2BL}$ | $\hat{j}$ | s.e. | 95% CI of $\hat{j}$ | 90% CI of $\hat{j}$ |
|---|---|---|---|---|---|---|
| Greene | 1843 | 11.3 | 4.14 | .067 | (4.01, 4.27) | (4.03, 4.25) |
| Rawl | 1693 | 5.7 | 4.05 | .070 | (3.91, 4.18) | (3.93, 4.16) |
| Registered Voters | 2120 | 37.9 | 3.84 | .061 | (3.72, 3.95) | (3.74, 3.94) |

| Absentee | $N$ | $X^2_{2BL}$ | $\hat{j}$ | s.e. | 95% CI of $\hat{j}$ | 90% CI of $\hat{j}$ |
|---|---|---|---|---|---|---|
| Greene | 45 | 7.76 | 3.6 | .43 | (2.76, 4.44) | (2.90, 4.30) |
| Rawl | 46 | 8.09 | 4.5 | .42 | (3.68, 5.32) | (3.81, 5.19) |

$N$ denotes the number of precincts of each respective type that have a count greater than nine (so there is a second digit), s.e. is the standard error if $\hat{j}$, and CI is confidence interval.

None of the vote counts have a $X^2_{2BL}$ statistic that differs from what one would expect if the second digits of the counts are distributed with the relative frequencies $r_j$ given by the Benford's Law formula. For the counts of Registered Voters, $X^2_{2BL}$ is different, but we have no reason to expect any particular pattern for those numbers.

The value of $\hat{j}$ differs signifcantly from $\bar{j}$, the value expected according to Benford's Law, only for Rawl's election day counts and then only more than trivially if a 90% confidence interval is considered. We observe $\hat{j} < \bar{j}$. For Greene's election day vote counts, $\hat{j}$ does not differ signifcantly from $\bar{j}$. The absentee counts have $\hat{j}$ estimated with a standard error that is too large to be informative.

The value of $\hat{j}$ for Rawl matches the value observed in Mebane (2010a; 2010b) for many losing legislative candidates in U.S. elections during the 1980s and 2000s, and so might not be considered all that unusual. The results in Mebane (2010a; 2010b) are obtained for general elections, however, so whether the patterns observed there hold generally for primary elections is unknown. Mebane (2010a) suggests that the pattern of $\hat{j}$ less than $\bar{j}$ and indeed near 4.0 may be attributed to roll-off. This may be, and in the South Carolina primary many who voted in the Democratic governor's race did not vote in the Democratic Senate race. Hence roll-off cannot be immediately ruled out as a possible explanation. But Mebane (2010b) will state that "gerrymandering" and turnout together better explain the patterns observed for legislative races for $\hat{j}$. "Gerrymandering" here means merely the grouping of individuals into legislative districts, whether or not that is done with intent to skew the results. In the case of the South Carolina primary, the self-sorting of voters into party primary voting groups (Democrat or Republican) might be consider the relevant kind of "gerrymander." Whether the values of $\hat{j}$ observed for Greene and Rawl are to be expected then depends on the pattern of voter turnout. The discussion in Mebane (2010b) won't refer to the South Carolina primary, but hopefully the logic to be described there will be feasible to apply to the South Carolina case.

In the absence of survey data and of a very robust campaign, it is difficult to know what voters may have had in mind when choosing between Greene and Rawl. Some have examined the relation between race and the vote split at the county level, but as far as I know such data have not been analyzed at the level of precincts. Having only county data, the risk of ecological inference fallacy—the pattern of indvidual behavior does not match the behavior of the aggregate—is great. This risk would also exist with precinct data, but it would be less if precincts are racially more homogenous than counties are. Perhaps individual voter history files could be used to tell definitively the race of each person who actually voted. The problem would remain then of trying to determine which voters voted in the Senate race, since not every voter did. But such data may be refined enough to support a version of the kind of simulations described in Mebane (2010b) that is tuned to the conditions of the election in South Carolina.

The value of $\hat{j}$ is also compatible with some kind of election fraud, although in Mebane (2010a; 2010b) and Mebane and Kalinin (2010) the only kind of fraud considered was "coerced" votes. A flaw in voting machines that counted votes differently from the way they were cast might be coercion in the sense of those papers. In these cases the $\hat{j}$ values observed for Greene and Rawl are not what one would expect under coercion according to Mebane (2010a) and Mebane and Kalinin (2010), but they can appear under some of the conditions simulated in Mebane (2010b).

Bottom line: the digit test results are compatible with normal political processes, although more information is needed to form stronger opinions about that; but some kind of fraud cannot be ruled out.

## References

Mebane, Walter R., Jr. 2010a. Election Fraud or Strategic Voting? Paper prepared for presentation at the Annual Meeting of the Midwest Political Science Association, Chicago, IL, April 22–25, 2010.

Mebane, Walter R., Jr. 2010b. Election Fraud or Strategic Voting? Can Second-digit Tests Tell the Difference? Paper prepared for presentation at the Summer Meeting of the Political Methodology Society, University of Iowa, July 22–24, 2010.

Mebane, Walter R., Jr., and Kirill Kalinin. 2010a. Electoral Fraud in Russia: Vote Counts Analysis using Second-digit Mean Tests. Paper prepared for presentation at the Annual Meeting of the Midwest Political Science Association, Chicago, IL, April 22–25, 2010.

Sizes (in bytes), timestamps and names of precinct data files

```
 783 Jun 10 21:13 Abbeville.csv
2887 Jun 10 21:14 Aiken.csv
 568 Jun 10 21:14 Allendale.csv
2746 Jun 10 21:14 Anderson.csv
 717 Jun 10 21:15 Bamberg.csv
 768 Jun 10 21:15 Barnwell.csv
3037 Jun 10 21:16 Beaufort.csv
2063 Jun 10 21:16 Berkeley.csv
 673 Jun 10 21:16 Calhoun.csv
6686 Jun 10 21:17 Charleston.csv
1339 Jun 10 21:17 Cherokee.csv
1041 Jun 10 21:27 Chester.csv
1126 Jun 10 21:28 Chesterfield.csv
1138 Jun 10 21:28 Clarendon.csv
1259 Jun 10 21:28 Colleton.csv
1367 Jun 10 21:28 Darlington.csv
 970 Jun 10 21:29 Dillon.csv
2596 Jun 10 21:29 Dorchester.csv
 704 Jun 10 21:29 Edgefield.csv
1035 Jun 10 21:30 Fairfield.csv
2345 Jun 10 21:30 Florence.csv
1462 Jun 10 21:30 Georgetown.csv
5265 Jun 10 21:32 Greenville.csv
1622 Jun 10 21:32 Greenwood.csv
 895 Jun 10 21:32 Hampton.csv
3993 Jun 10 21:32 Horry.csv
 800 Jun 10 21:33 Jasper.csv
1433 Jun 10 21:33 Kershaw.csv
1244 Jun 10 21:33 Lancaster.csv
```

```
 1365 Jun 10 21:33 Laurens.csv
 1082 Jun 10 21:33 Lee.csv
 3184 Jun 10 21:34 Lexington.csv
  900 Jun 10 21:34 Marion.csv
  830 Jun 10 21:34 Marlboro.csv
  629 Jun 10 21:34 McCormick.csv
 1252 Jun 10 21:35 Newberry.csv
 1226 Jun 10 21:35 Oconee.csv
 2092 Jun 10 21:35 Orangeburg.csv
 1905 Jun 10 21:35 Pickens.csv
 4507 Jun 10 21:36 Richland.csv
  839 Jun 10 21:36 Saluda.csv
 4433 Jun 10 21:36 Spartanburg.csv
 2176 Jun 10 21:36 Sumter.csv
 1089 Jun 10 21:36 Union.csv
 1235 Jun 10 21:36 Williamsburg.csv
 3098 Jun 10 21:37 York.csv
```

file `tbenf3.R`, timestamp `Jun 12 15:33`

```
# test Benford’s Law compliance

asc <- function(x) { as.character(x) }

flist <- system("ls *.csv", intern=TRUE);
dlist <- list();
for (jf in flist) {
  tpipe <- pipe(paste("awk ’NR>2’",jf));
  dlist[[jf]] <- read.csv(tpipe);
  print(names(dlist[[jf]]));
}
nlines <- 0;
for (jf in flist) {
  nlines <- nlines + dim(dlist[[jf]])[1];
}
mat <- matrix(NA, nlines, dim(dlist[[jf]])[2]-1);
precinct <- vector();
ii <- 1;
for (jf in flist) {
  precinct <- c(precinct,asc(dlist[[jf]]$Precinct));
  jlines <- dim(dlist[[jf]])[1];
  mat[ii:(ii+jlines-1),] <- as.matrix(dlist[[jf]][,-1]);
  ii <- ii + jlines;
}
dat <- data.frame(precinct,mat);
```

```
names(dat) <- names(dlist[[1]]);
names(dat)[-(1:2)] <-
  c("Greene E Day","Greene Total Votes","Rawl E Day","Rawl Total Votes","Total");
dim(dat);
names(dat);

runtest <- function(wrk, digit=2) {
  getdigits <- function(num,digit=2) {
    s <- as.character(num);
    idx <- sapply(s, function(x){ nchar(x) >= digit; });
    dout <- ifelse(idx, NA, "");
    if (any(idx)) {
      dout[idx] <- sapply(s[idx], function(x){ substr(x,digit,digit) });
    }
    return(dout);
  }

  # Benford's Law probablity formulas
  # source
  # http://www.mathpages.com/home/kmath302/kmath302.htm
  #
  # PndB:  d, digit value (vector);  n, digit place;  B, base
  # probability that the n-th digit following the first nonzero digit is d,
  #   for numbers in base B
  # d can be a vector;  n and B must be positive scalars
  #
  PndB <- function(d,n,B=10) {
    if (n==0) {
      p <- log(1+1/d)/log(B);
    }
    else {
      seq <- B^(n-1):(B^n-1);
      p <- d*0;
      for (i in 1:length(d)) {
        p[i] <- sum(log(1+1/(d[i]+seq*B)))/log(B);
      }
    }
    return(p);
  }

  least <- ifelse(digit==1, 1, 0);  # initial digit for Benford test
  vec <- least:9;
  set <- rep(0,length(vec));
  names(set) <- as.character(vec);
  ndtable <- names(dtable <- table(d <- getdigits(wrk, digit=digit)));
  m <- mean(d <- as.numeric(d), na.rm=TRUE);
  sd <- sqrt(var(d, na.rm=TRUE))
  set[ndtable[ndtable %in% vec]] <- dtable[ndtable[ndtable %in% vec]];
  sumset <- sum(set);
  dmean <- sum(set * 0:9)/sum(set);
  # Benford's law logarithmic distribution test
  bpred <- sumset*PndB(vec, digit-1);
  chi <- sum(dev <- (set - bpred)^2 / bpred);
  if(is.na(chi)) print( table(getdigits(wrk, digit=digit)) );
  return(list(N=sumset, chi=chi, mean=m, se=sd/sqrt(sumset)));
}

idxT <- asc(dat$Precinct) %in% c("Totals:");
sum(idxT);
idxTA <- asc(dat$Precinct) %in% c("Totals:","Absentee");
sum(idxTA);
idxA <- asc(dat$Precinct) %in% c("Absentee");
sum(idxA);

# all ballots
lapply(dat[!idxT,c(2,4,6)], runtest);
# 95% and 90% confidence intervals
lapply(lapply(dat[!idxT,c(2,4,6)], runtest),
  function(x){ c(x$mean+x$se*qnorm(1-.025)*c(-1,1),
    x$mean+x$se*qnorm(1-.05)*c(-1,1)) });

# all non-absentee ballots
lapply(dat[!idxTA,c(2,4,6)], runtest);
# 95% and 90% confidence intervals
lapply(lapply(dat[!idxTA,c(2,4,6)], runtest),
  function(x){ c(x$mean+x$se*qnorm(1-.025)*c(-1,1),
    x$mean+x$se*qnorm(1-.05)*c(-1,1)) });

# absentee ballots only
lapply(dat[idxA,c(2,4,6)], runtest);
# 95% and 90% confidence intervals
lapply(lapply(dat[idxA,c(2,4,6)], runtest),
  function(x){ c(x$mean+x$se*qnorm(1-.025)*c(-1,1),
    x$mean+x$se*qnorm(1-.05)*c(-1,1)) });
```

file `tbenf3.Rout`, timestamp `Jun 12 15:33`

```
R version 2.11.1 (2010-05-31)
Copyright (C) 2010 The R Foundation for Statistical Computing
```

```
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type ’license()’ or ’licence()’ for distribution details.

R is a collaborative project with many contributors.
Type ’contributors()’ for more information and
’citation()’ on how to cite R or R packages in publications.

Type ’demo()’ for some demos, ’help()’ for on-line help, or
’help.start()’ for an HTML browser interface to help.
Type ’q()’ to quit R.

> # test Benford’s Law compliance
>
> asc <- function(x) { as.character(x) }
>
> flist <- system("ls *.csv", intern=TRUE);
> dlist <- list();
> for (jf in flist) {
+   tpipe <- pipe(paste("awk ’NR>2’",jf));
+   dlist[[jf]] <- read.csv(tpipe);
+   print(names(dlist[[jf]]));
+ }
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
```

```
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
```

```
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
```

```
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
[1] "Precinct"          "Registered.Voters" "Election.Day"
[4] "Total.Votes"       "Election.Day.1"    "Total.Votes.1"
[7] "Total"
> nlines <- 0;
> for (jf in flist) {
+   nlines <- nlines + dim(dlist[[jf]])[1];
+ }
> mat <- matrix(NA, nlines, dim(dlist[[jf]])[2]-1);
> precinct <- vector();
> ii <- 1;
> for (jf in flist) {
+   precinct <- c(precinct,asc(dlist[[jf]]$Precinct));
+   jlines <- dim(dlist[[jf]])[1];
+   mat[ii:(ii+jlines-1),] <- as.matrix(dlist[[jf]][,-1]);
+   ii <- ii + jlines;
+ }
> dat <- data.frame(precinct,mat);
> names(dat) <- names(dlist[[1]]);
> names(dat)[-(1:2)] <-
+   c("Greene E Day","Greene Total Votes","Rawl E Day","Rawl Total Votes","Total");
```

```
> dim(dat);
[1] 2395    7
> names(dat);
[1] "Precinct"           "Registered.Voters"  "Greene E Day"
[4] "Greene Total Votes" "Rawl E Day"         "Rawl Total Votes"
[7] "Total"
>
> runtest <- function(wrk, digit=2) {
+   getdigits <- function(num,digit=2) {
+     s <- as.character(num);
+     idx <- sapply(s, function(x){ nchar(x) >= digit; });
+     dout <- ifelse(idx, NA, "");
+     if (any(idx)) {
+       dout[idx] <- sapply(s[idx], function(x){ substr(x,digit,digit) });
+     }
+     return(dout);
+   }
+
+   # Benford's Law probablity formulas
+   # source
+   # http://www.mathpages.com/home/kmath302/kmath302.htm
+   #
+   # PndB:  d, digit value (vector);  n, digit place;  B, base
+   # probability that the n-th digit following the first nonzero digit is d,
+   #   for numbers in base B
+   # d can be a vector;  n and B must be positive scalars
+   #
+   PndB <- function(d,n,B=10) {
+     if (n==0) {
+       p <- log(1+1/d)/log(B);
+     }
+     else {
+       seq <- B^(n-1):(B^n-1);
+       p <- d*0;
+       for (i in 1:length(d)) {
+         p[i] <- sum(log(1+1/(d[i]+seq*B)))/log(B);
+       }
+     }
+     return(p);
+   }
+
+   least <- ifelse(digit==1, 1, 0);  # initial digit for Benford test
+   vec <- least:9;
+   set <- rep(0,length(vec));
+   names(set) <- as.character(vec);
+   ndtable <- names(dtable <- table(d <- getdigits(wrk, digit=digit)));
+   m <- mean(d <- as.numeric(d), na.rm=TRUE);
+   sd <- sqrt(var(d, na.rm=TRUE))
+   set[ndtable[ndtable %in% vec]] <- dtable[ndtable[ndtable %in% vec]];
+   sumset <- sum(set);
+   dmean <- sum(set * 0:9)/sum(set);
+   # Benford’s law logarithmic distribution test
+   bpred <- sumset*PndB(vec, digit-1);
+   chi <- sum(dev <- (set - bpred)^2 / bpred);
+   if(is.na(chi)) print( table(getdigits(wrk, digit=digit)) );
+   return(list(N=sumset, chi=chi, mean=m, se=sd/sqrt(sumset)));
+ }
>
> idxT <- asc(dat$Precinct) %in% c("Totals:");
> sum(idxT);
[1] 46
> idxTA <- asc(dat$Precinct) %in% c("Totals:","Absentee");
> sum(idxTA);
[1] 92
> idxA <- asc(dat$Precinct) %in% c("Absentee");
> sum(idxA);
[1] 46
>
> # all ballots
> lapply(dat[!idxT,c(2,4,6)], runtest);
$Registered.Voters
$Registered.Voters$N
[1] 2120

$Registered.Voters$chi
[1] 37.92189

$Registered.Voters$mean
[1] 3.835377

$Registered.Voters$se
[1] 0.06089732


$‘Greene Total Votes‘
$‘Greene Total Votes‘$N
[1] 1888

$‘Greene Total Votes‘$chi
[1] 11.19790
```

```
$`Greene Total Votes`$mean
[1] 4.128178

$`Greene Total Votes`$se
[1] 0.06574871


$`Rawl Total Votes`
$`Rawl Total Votes`$N
[1] 1739

$`Rawl Total Votes`$chi
[1] 5.834828

$`Rawl Total Votes`$mean
[1] 4.057504

$`Rawl Total Votes`$se
[1] 0.06878073


> # 95% and 90% confidence intervals
> lapply(lapply(dat[!idxT,c(2,4,6)], runtest),
+   function(x){ c(x$mean+x$se*qnorm(1-.025)*c(-1,1),
+     x$mean+x$se*qnorm(1-.05)*c(-1,1)) });
$Registered.Voters
[1] 3.716021 3.954734 3.735210 3.935545

$`Greene Total Votes`
[1] 3.999313 4.257043 4.020031 4.236325

$`Rawl Total Votes`
[1] 3.922697 4.192312 3.944370 4.170639

>
> # all non-absentee ballots
> lapply(dat[!idxTA,c(2,4,6)], runtest);
$Registered.Voters
$Registered.Voters$N
[1] 2120

$Registered.Voters$chi
[1] 37.92189
```

```
$Registered.Voters$mean
[1] 3.835377

$Registered.Voters$se
[1] 0.06089732


$‘Greene Total Votes‘
$‘Greene Total Votes‘$N
[1] 1843

$‘Greene Total Votes‘$chi
[1] 11.30764

$‘Greene Total Votes‘$mean
[1] 4.141074

$‘Greene Total Votes‘$se
[1] 0.06652799


$‘Rawl Total Votes‘
$‘Rawl Total Votes‘$N
[1] 1693

$‘Rawl Total Votes‘$chi
[1] 5.679481

$‘Rawl Total Votes‘$mean
[1] 4.045481

$‘Rawl Total Votes‘$se
[1] 0.0697181


> # 95% and 90% confidence intervals
> lapply(lapply(dat[!idxTA,c(2,4,6)], runtest),
+   function(x){ c(x$mean+x$se*qnorm(1-.025)*c(-1,1),
+     x$mean+x$se*qnorm(1-.05)*c(-1,1)) });
$Registered.Voters
[1] 3.716021 3.954734 3.735210 3.935545

$‘Greene Total Votes‘
[1] 4.010682 4.271467 4.031646 4.250503
```

```
$‘Rawl Total Votes‘
[1] 3.908836 4.182126 3.930805 4.160157

>
> # absentee ballots only
> lapply(dat[idxA,c(2,4,6)], runtest);


46
$Registered.Voters
$Registered.Voters$N
[1] 0

$Registered.Voters$chi
[1] NaN

$Registered.Voters$mean
[1] NaN

$Registered.Voters$se
[1] NA


$‘Greene Total Votes‘
$‘Greene Total Votes‘$N
[1] 45

$‘Greene Total Votes‘$chi
[1] 7.75986

$‘Greene Total Votes‘$mean
[1] 3.6

$‘Greene Total Votes‘$se
[1] 0.4280564


$‘Rawl Total Votes‘
$‘Rawl Total Votes‘$N
[1] 46

$‘Rawl Total Votes‘$chi
[1] 8.086373

$‘Rawl Total Votes‘$mean
```

```
[1] 4.5

$‘Rawl Total Votes‘$se
[1] 0.4202024


> # 95% and 90% confidence intervals
> lapply(lapply(dat[idxA,c(2,4,6)], runtest),
+   function(x){ c(x$mean+x$se*qnorm(1-.025)*c(-1,1),
+     x$mean+x$se*qnorm(1-.05)*c(-1,1)) });


46
$Registered.Voters
[1] NaN NaN NaN NaN

$‘Greene Total Votes‘
[1] 2.761025 4.438975 2.895910 4.304090

$‘Rawl Total Votes‘
[1] 3.676418 5.323582 3.808829 5.191171

>
> proc.time()
   user  system elapsed
  0.960   0.252   1.184
```