Mapping the Results of the 2010 Australian Federal Election in R

Simon Jackman
Stanford University

September 9, 2010
## List of Figures

1. Outline of the Division of Brisbane. ................................................................. 8
2. Brisbane, with Labor 2PP vote percentage by polling place. ......................... 19
3. Melbourne, Green first preferences by polling place. ..................................... 21
4. Grayndler, Green first preferences by polling place. ..................................... 22
5. Batman, Green first preferences by polling place. ......................................... 23
6. Fowler, informality by polling place. ............................................................. 24
7. Kingston, Swing in ALP 2PP by polling place. ............................................. 25
8. Bennelong, Swing in ALP 2PP by polling place. ........................................... 26
9. Melbourne Metropolitan Area, Labor 2PP vote percentage by polling place. .... 28
10. Melbourne Metropolitan Area, Green primary vote share by polling place. .... 29
11. Sydney Metropolitan Area, Labor 2PP vote percentage by polling place. ........ 30
12. Brisbane Metropolitan Area, Green primary vote share by polling place. ...... 31
13. Brisbane Metropolitan Area, Labor 2PP swing by polling place. ................ 32
Our goal is to produce some maps of the House of Representatives election results using the mapping functionality in R.

## 1 Election results

We begin by downloading election results from the AEC website. First, we get the first preference by polling place data. These live in files specific to each state. The AEC gives us CSV formatted data, with a one line informational header that we ignore:

```r
stateAbbs <- c("NSW","VIC","QLD","WA","SA","TAS","ACT","NT")
res <- NULL
for(j in stateAbbs){
  thisFile <- paste(stem,j,".csv",sep="")
  con <- url(thisFile,open="r")
  cat(paste("downloading",thisFile,"\n"))
  tmpRes <- read.table(file=con,
    header=TRUE,
    stringsAsFactors=FALSE,
    quote="\",
    skip=1,
    sep="","
  )
  close(con)
  res <- rbind(res,tmpRes)
}
```

The resulting data frame is rather large:

```r
summary(res)
```

<table>
<thead>
<tr>
<th>StateAb</th>
<th>DivisionID</th>
<th>DivisionNm</th>
<th>PollingPlaceID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length:60663</td>
<td>Min. :101.0</td>
<td>Length:60663</td>
<td>Min. :1</td>
</tr>
<tr>
<td>Class:character</td>
<td>1st Qu.:138.0</td>
<td>Class:character</td>
<td>1st Qu.:2625</td>
</tr>
<tr>
<td>Mode:character</td>
<td>Median:182.0</td>
<td>Mode:character</td>
<td>Median:5921</td>
</tr>
<tr>
<td>Mean:186.9</td>
<td>3rd Qu.:228.0</td>
<td>Mean:12661</td>
<td></td>
</tr>
<tr>
<td>Max.:316.0</td>
<td>3rd Qu.:12031</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PollingPlace</th>
<th>CandidateID</th>
<th>Surname</th>
<th>GivenNm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length:60663</td>
<td>Min. :999</td>
<td>Length:60663</td>
<td>Length:60663</td>
</tr>
<tr>
<td>Class:character</td>
<td>1st Qu.:20824</td>
<td>Class:character</td>
<td>Class:character</td>
</tr>
<tr>
<td>Mode:character</td>
<td>Median:21162</td>
<td>Mode:character</td>
<td>Mode:character</td>
</tr>
<tr>
<td>Mean:18213</td>
<td>3rd Qu.:21570</td>
<td>Mean:58818</td>
<td></td>
</tr>
<tr>
<td>Max.:22036</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BallotPosition</th>
<th>Elected</th>
<th>HistoricElected</th>
<th>PartyAb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. :1.0</td>
<td>Length:60663</td>
<td>Length:60663</td>
<td>Length:60663</td>
</tr>
<tr>
<td>1st Qu.:2.0</td>
<td>Class:character</td>
<td>Class:character</td>
<td>Class:character</td>
</tr>
<tr>
<td>Median:4.0</td>
<td>Mode:character</td>
<td>Mode:character</td>
<td>Mode:character</td>
</tr>
<tr>
<td>Mean:150.5</td>
<td>3rd Qu.:6.0</td>
<td>Mean:12661</td>
<td></td>
</tr>
<tr>
<td>Max.:999.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PartyNm</th>
<th>OrdinaryVotes</th>
<th>Swing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length:60663</td>
<td>Min. :0.0</td>
<td>Min. :-262.0500</td>
</tr>
<tr>
<td>Class:character</td>
<td>1st Qu.:10.0</td>
<td>1st Qu.: -0.5600</td>
</tr>
<tr>
<td>Mode:character</td>
<td>Median:44.0</td>
<td>Median:0.7600</td>
</tr>
<tr>
<td>Mean:182.6</td>
<td>Mean:0.7447</td>
<td></td>
</tr>
<tr>
<td>3rd Qu.:189.0</td>
<td>3rd Qu.:2.6800</td>
<td></td>
</tr>
<tr>
<td>Max.:6473.0</td>
<td>Max.:82.6800</td>
<td></td>
</tr>
</tbody>
</table>
We first compute an object that converts the polling place level vote counts to percentages of the formal vote, computes the informal vote percentage, and the total number of votes cast at a particular polling place. We use the `by` function, breaking the `res` object into subsets by each unique level of the `PollingPlaceID` variable (we convert the latter to a factor, which the `by` function is expecting). For each polling place we then extract the formal votes (votes with ballot position no equal to 999). We sum these vote to get the totalFormal vote; the informal vote share is the number of informal votes divided by the sum of all votes, times 100. We convert the ordinary vote tallies to vote shares by dividing by the totalFormal vote; the `PartyAb` variable gives us a way of associating vote shares with parties, and we tag the vote shares with these labels as names.

```r
voteShares <- by(res,
  factor(res$PollingPlaceID,levels=unique(res$PollingPlaceID)),
  function(x){
    formal <- x$BallotPosition!=999
    totalFormal <- sum(x$OrdinaryVotes[formal],na.rm=TRUE)
    informal <- x$OrdinaryVotes[!formal]/sum(x$OrdinaryVotes,na.rm=TRUE)*100
    props <- NULL
    if(totalFormal>0){
      props <- x$OrdinaryVotes[formal]/totalFormal*100
      if(length(props)>0)
        names(props) <- c("Total","Informal",x$PartyAb[formal])
      }
    return(props)
  }, simplify=FALSE)
```

`voteShares` is a list with names given by the unique polling place identifiers:

```r
names(voteShares)[1:3]
```

```
[1] "31" "67" "56500"
```

Now we write a simple extraction function that can pull all the vote shares for a given party, given a division or set of divisions. We first look at the party abbreviations that appear in the file:

```r
table(res$PartyAb)
```

```
AFN  ALP  ASXP  BAP  CA  CAL  CDP  CEC  CLP  DEM  DLP  FFP  GRN  IND  LDP
8951  235  8951  291 149 161  46  2680  859  98 1504 355 6374 8951  5941  1338
LNQ  LP  NAFD  NCP  NP  ON  SAL  SEP  SPA  TCS
1718  6242  311  113 1552 1182  611  484  1110  456
```

The party labels we are likely to be interested in include ALP and GRN, with the Coalition candidates appearing under the labels LP, NP, CLP (Country Liberal Party) and LNQ (the Liberal-National party in Queensland). We use these party groupings when we extract primary vote shares at the polling-place level for a given division:

```r
getPrimaryVotes <- function(dvision,party){
  if(party=="Coalition")
    party <- c("LP","NP","CLP","LNQ")
  ok <- res$DivisionNm %in% division
  thePollingPlaces <- unique(res$PollingPlaceID[ok])
}
```
+ thePollingPlacesNames <- res$PollingPlace[match(thePollingPlaces,
    res$PollingPlaceID)]
+ theResults <- voteShares[match(thePollingPlaces,names(voteShares))]
+ theDivisionNames <- res$DivisionNm[match(thePollingPlaces,res$PollingPlaceID)]
+ 
+ ## dump bad polling places -- no vote totals
+ bad <- unlist(lapply(theResults,is.null))
+ 
+ ## return totalFormal and target party results in a data frame
+ tmp <- data.frame(total=unlist(lapply(theResults[!bad],
    function(x)x["Total"])),
    x=unlist(lapply(theResults[!bad],
    function(x)sum(x[match(party,names(x))],na.rm=TRUE)) ),
    division=theDivisionNames[!bad],
    PollingPlace=thePollingPlacesNames[!bad],
    PollingPlaceID=thePollingPlaces[!bad])
+ rownames(tmp) <- NULL
+ return(tmp)
+ }

This function returns the requested data in the variable called x. We give this function a test drive:

> getPrimaryVotes("Brisbane","ALP")

<table>
<thead>
<tr>
<th>total</th>
<th>x</th>
<th>division</th>
<th>PollingPlace</th>
<th>PollingPlaceID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1771</td>
<td>36.98475</td>
<td>Brisbane</td>
<td>Alderley</td>
<td>5167</td>
</tr>
<tr>
<td>2551</td>
<td>15.13132</td>
<td>Brisbane</td>
<td>Ascot</td>
<td>6017</td>
</tr>
<tr>
<td>2802</td>
<td>33.08351</td>
<td>Brisbane</td>
<td>Ashgrove</td>
<td>5168</td>
</tr>
<tr>
<td>1263</td>
<td>30.32462</td>
<td>Brisbane</td>
<td>Ashgrove West (Brisbane)</td>
<td>55316</td>
</tr>
<tr>
<td>839</td>
<td>34.80334</td>
<td>Brisbane</td>
<td>Ballymore</td>
<td>5170</td>
</tr>
<tr>
<td>2152</td>
<td>28.48513</td>
<td>Brisbane</td>
<td>Brisbane City (Brisbane)</td>
<td>5566</td>
</tr>
<tr>
<td>2004</td>
<td>29.09182</td>
<td>Brisbane</td>
<td>Brisbane City BRISBANE PPVC</td>
<td>32606</td>
</tr>
<tr>
<td>1849</td>
<td>24.06706</td>
<td>Brisbane</td>
<td>Clayfield</td>
<td>6029</td>
</tr>
<tr>
<td>2350</td>
<td>24.29787</td>
<td>Brisbane</td>
<td>Clayfield West</td>
<td>6030</td>
</tr>
<tr>
<td>734</td>
<td>28.20163</td>
<td>Brisbane</td>
<td>Divisional Office (PREPOLL)</td>
<td>30147</td>
</tr>
<tr>
<td>1118</td>
<td>34.25760</td>
<td>Brisbane</td>
<td>Dorrington</td>
<td>5171</td>
</tr>
<tr>
<td>1139</td>
<td>35.29412</td>
<td>Brisbane</td>
<td>Enoggera</td>
<td>5174</td>
</tr>
<tr>
<td>1359</td>
<td>33.25975</td>
<td>Brisbane</td>
<td>Fortitude Valley</td>
<td>5576</td>
</tr>
<tr>
<td>948</td>
<td>39.76793</td>
<td>Brisbane</td>
<td>Gaythorne</td>
<td>5177</td>
</tr>
<tr>
<td>1531</td>
<td>36.12018</td>
<td>Brisbane</td>
<td>Gordon Park</td>
<td>38318</td>
</tr>
<tr>
<td>2143</td>
<td>23.93840</td>
<td>Brisbane</td>
<td>Hamilton</td>
<td>6034</td>
</tr>
<tr>
<td>1467</td>
<td>23.44922</td>
<td>Brisbane</td>
<td>Hendra</td>
<td>6035</td>
</tr>
<tr>
<td>852</td>
<td>31.57277</td>
<td>Brisbane</td>
<td>Hendra East</td>
<td>6036</td>
</tr>
<tr>
<td>423</td>
<td>36.40662</td>
<td>Brisbane</td>
<td>Herston</td>
<td>5195</td>
</tr>
<tr>
<td>1330</td>
<td>36.09023</td>
<td>Brisbane</td>
<td>Ithaca Creek</td>
<td>5182</td>
</tr>
<tr>
<td>1399</td>
<td>28.28428</td>
<td>Brisbane</td>
<td>Ithaca South</td>
<td>5183</td>
</tr>
<tr>
<td>1352</td>
<td>32.32249</td>
<td>Brisbane</td>
<td>Kalinga (Brisbane)</td>
<td>6037</td>
</tr>
<tr>
<td>1385</td>
<td>29.24188</td>
<td>Brisbane</td>
<td>Kelvin Grove</td>
<td>5186</td>
</tr>
<tr>
<td>1606</td>
<td>35.61644</td>
<td>Brisbane</td>
<td>Lutwyche</td>
<td>31645</td>
</tr>
<tr>
<td>2157</td>
<td>31.24710</td>
<td>Brisbane</td>
<td>Merthyr</td>
<td>5580</td>
</tr>
<tr>
<td>1599</td>
<td>28.33021</td>
<td>Brisbane</td>
<td>Milton (Brisbane)</td>
<td>5581</td>
</tr>
<tr>
<td>2963</td>
<td>30.27337</td>
<td>Brisbane</td>
<td>New Farm</td>
<td>5584</td>
</tr>
<tr>
<td>1912</td>
<td>26.09833</td>
<td>Brisbane</td>
<td>New Farm South</td>
<td>33080</td>
</tr>
<tr>
<td>2179</td>
<td>32.67554</td>
<td>Brisbane</td>
<td>Newmarket</td>
<td>5191</td>
</tr>
<tr>
<td>880</td>
<td>34.77273</td>
<td>Brisbane</td>
<td>Newmarket South</td>
<td>5192</td>
</tr>
<tr>
<td>1444</td>
<td>30.33241</td>
<td>Brisbane</td>
<td>Paddington</td>
<td>5194</td>
</tr>
<tr>
<td>1074</td>
<td>25.97765</td>
<td>Brisbane</td>
<td>Petrie Terrace</td>
<td>5588</td>
</tr>
<tr>
<td>1161</td>
<td>30.57709</td>
<td>Brisbane</td>
<td>Red Hill</td>
<td>5190</td>
</tr>
<tr>
<td>152</td>
<td>36.84211</td>
<td>Brisbane</td>
<td>Special Hospital Team 1</td>
<td>32614</td>
</tr>
<tr>
<td>71</td>
<td>39.43662</td>
<td>Brisbane</td>
<td>Special Hospital Team 2</td>
<td>32615</td>
</tr>
<tr>
<td>1442</td>
<td>28.71012</td>
<td>Brisbane</td>
<td>Spring Hill</td>
<td>5591</td>
</tr>
<tr>
<td>1095</td>
<td>37.07763</td>
<td>Brisbane</td>
<td>Stafford (Brisbane)</td>
<td>31499</td>
</tr>
<tr>
<td>387</td>
<td>36.95090</td>
<td>Brisbane</td>
<td>Stafford West (Brisbane)</td>
<td>31502</td>
</tr>
<tr>
<td>622</td>
<td>40.83601</td>
<td>Brisbane</td>
<td>Whinstanes</td>
<td>6054</td>
</tr>
<tr>
<td>3754</td>
<td>30.79382</td>
<td>Brisbane</td>
<td>Wilston</td>
<td>5203</td>
</tr>
</tbody>
</table>
> getPrimaryVotes("Brisbane","GRN")

<table>
<thead>
<tr>
<th>total</th>
<th>x</th>
<th>division</th>
<th>PollingPlace</th>
<th>PollingPlaceID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1771</td>
<td>20.553360</td>
<td>Brisbane</td>
<td>Alderley</td>
<td>5167</td>
</tr>
<tr>
<td>2 2551</td>
<td>8.898471</td>
<td>Brisbane</td>
<td>Ascot</td>
<td>6017</td>
</tr>
<tr>
<td>3 2802</td>
<td>21.984927</td>
<td>Brisbane</td>
<td>Ashgrove</td>
<td>5168</td>
</tr>
<tr>
<td>4 1263</td>
<td>18.052257</td>
<td>Brisbane</td>
<td>Ashgrove West (Brisbane)</td>
<td>53316</td>
</tr>
<tr>
<td>5 839</td>
<td>21.573302</td>
<td>Brisbane</td>
<td>Ballymore</td>
<td>5170</td>
</tr>
<tr>
<td>6 2152</td>
<td>25.697026</td>
<td>Brisbane</td>
<td>Brisbane City (Brisbane)</td>
<td>5566</td>
</tr>
<tr>
<td>7 2004</td>
<td>23.902196</td>
<td>Brisbane</td>
<td>Brisbane City BRISBANE PPVC</td>
<td>32606</td>
</tr>
<tr>
<td>8 1849</td>
<td>13.737155</td>
<td>Brisbane</td>
<td>Clayfield</td>
<td>6029</td>
</tr>
<tr>
<td>9 2350</td>
<td>18.042553</td>
<td>Brisbane</td>
<td>Clayfield West</td>
<td>6030</td>
</tr>
<tr>
<td>10 734</td>
<td>17.293731</td>
<td>Brisbane</td>
<td>Divisional Office (PREPOL)</td>
<td>30147</td>
</tr>
</tbody>
</table>

Two-party preferred results by polling place are in one CSV file on the AEC site.

The resulting object contains a variable called `Australian.Labor.Party.Percentage` which we will use in our plotting; no additional computation is required. Accordingly, we can write a very simple “extraction” given division names:
It is important to remember that 2PP tallies are not available in every district; the AEC describes as “non-classic” those divisions in which the contest is not an ALP/Coalition contest. There are a relatively large number of these divisions this time. In these divisions the Australian.Labor.Party.Percentage is set to zero at all polling places, at least until the AEC finalizes the “two-candidate” result and then turns to the “2PP-for-scrutiny” count. For instance, in Grayndler:

```r
> summary(getTPP("Grayndler",type="tpp"))
```

2 Location of Polling Places

The AEC have done a wonderful service for the geeks of the world by providing us with a CSV file with lat/longs of the polling places:
3 Electoral Division Shape Files

The AEC also gives us free access to the shape files for Australia's 150 Commonwealth electoral divisions (seats or “electorates” in the House of Representatives). These are in a zip file, which we download separately, and then read into R with the maptools package:

```r
library(maptools)
gpclibPermit()
shp <- readShapeSpatial(fn="shp/COM_ELB_region", verbose=TRUE)
summary(shp)
```

We write a simple function that extracts the lat/long data from the shape files (boundaries of divisions and islands etc contained in divisions), so we can use them to augment some plotting functions later on:

```r
linesDivision <- function(division){
  require(sp)
  divisionLocal <- division
  divisionLocal[division=="McEwen"] <- "Mcewen"
  divisionLocal[division=="McMillan"] <- "Mcmillan"
  divisionLocal[division=="McPherson"] <- "Mcpherson"
  d <- match(divisionLocal,shp$ELECT_DIV)
  out <- NULL
  if(!is.na(d)){
    require(ggplot2)
    require(maptools)
    gpclibPermit()
    out <- fortify(shp[d,])
  }
  return(out)
}
```

We give this function a test spin:

```r
summary(linesDivision("Brisbane"))
```

Note: polygon geometry computations in maptools depend on the package gpclib, which has a restricted licence. It is disabled by default; to enable gpclib, type gpclibPermit()
Checking rgeos availability as gpclib substitute:
FALSE

<table>
<thead>
<tr>
<th></th>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>3rd Qu.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td>153.0</td>
<td>153.0</td>
<td>153.0</td>
<td>153.1</td>
<td>153.1</td>
</tr>
<tr>
<td>lat</td>
<td>-27.48</td>
<td>-27.44</td>
<td>-27.43</td>
<td>-27.41</td>
<td>-27.40</td>
</tr>
</tbody>
</table>

As a test, we plot the electorate of Brisbane, with the following code block producing Figure 1. We also extract the labpt attribute of the division data, and add it to the plot.

R Code

```
> labPoint <- function(division,cex=1,col=gray(.65)){
>   d <- match(division,shp$ELECT_DIV)
>   out <- NULL
>   if(!is.na(d)){
>     tmp <- shp[shp$ELECT_DIV==division,]
>     out <- tmp@polygons[[1]]@Polygons[[1]]@labpt
>     text(out[1],out[2],
>           textUpperSpace(division),
>           cex=cex,
>           col=col,
>           xpd=NA)
>   }
>   return(invisible(NULL))
>
> textUpperSpace <- function(string){
>   toupper(paste(unlist(strsplit(x=string,split="")),collapse=" "))
> }
> plot(lat~long,
>      data=linesDivision("Brisbane"),
>      type="l",
>      axes=FALSE,
>      xlab="",
>      ylab="",
>      asp=1)
> labPoint("Brisbane")
```

4 Google Maps

We will overlay the electoral data on static, terrain maps from Google Maps. We use the functions in the RGoogleMaps package; e.g., the GetMap.bbox and PlotOnStaticMap functions. These functions will produce 640 pixel by 640 pixel JPGs. We convert these to grayscale JPGs, so as to highlight the vote share data which we will overlay with colored plotting symbols.

A key task is to merge the polling-place level voting data with the geo-location data in the ppData object. We write the plotting function so as to be able to handle multiple divisions, and either primary results for a specific party or two party results, or swing in the 2PP results.

We now define a lengthy plotting function, that takes a vector of division names as an argument. We also pass two other arguments, the name of a party and the type of voting results we want plotted. A substantial part of the function is the legend-making; we could easily farm this out to a supporting function, keeping this “main” plotting function a little less spaghetti-like.
Figure 1: Outline of the Division of Brisbane. The smooth, curved portion of the southern boundary of the division is the Brisbane River.
> makeGoogleMap <- function(division,
+                      party=c("ALP","Coalition","GRN","Total","Informal"),
+                      type="primary",
+                      s=3,
+                      debug=FALSE){
+  require(maptools)
+  require(RgoogleMaps)
+  require(ReadImages)
+  ## file name for jpg
+  fName <- makeFileName(division,party,type)
+  fName <- paste("jpg/",fName,".jpg",sep="")
+  ## parse divisions
+  divisionOriginal <- division
+  division <- parseDivisionNames(division)
+  if(length(division)==0)
+    stop("no valid divisions")
+  ## subset polling place data for the selected divisions
+  places <- ppData[ppData$DivisionNm %in% division,]
+  ## screen odd places/results
+  places <- screenPollingPlaces(places,division,debug)
+  ## get corresponding voting data
+  if(type=="primary"){
+    voteData <- getPrimaryVotes(division,party)
+  }
+  if(type=="tpp" | type=="swing"){
+    voteData <- getTPP(division,type)
+    if(all(voteData$x==0))
+      stop("no 2PP data available")
+  }
+  ## merge lat-longs of polling places
+  places <- merge(x=voteData,
+                   y=places,
+                   all.x=TRUE,
+                   all.y=FALSE,
+                   by="PollingPlaceID")
+  if(length(grep("DivisionNm",names(places)))>1){
+    places$DivisionNm <- places$DivisionNm.x
+    places$DivisionNm.x <- NULL
+    places$DivisionNm.y <- NULL
+  }
+  ## clean out any remaining missing data
+  places <- places[!is.na(places$x) & !is.nan(places$x),]
+  places <- places[!is.na(places$Latitude) & !is.na(places$Longitude),]
+  ## error checking, save to external file
+  save("places",
+        file="places.rda")
+  if(debug)
+    print(places[order(places$x),
+               c("x","PollingPlaceNm","Latitude","Longitude")])
+  ## colors for points
+  cols <- colorMaker2(places$x,
+                      party=party,
+                      type=type,
+                      alpha=200)
+  ## size of points
+  scaleFactor <- sqrt(s*3/max(places$total))
+ cexPoints <- scaleFactor * sqrt(places$total)
+ 
+ ## bounding box
+ bb <- qbbox(lat=range(places$Latitude,na.rm=TRUE),
+ lon=range(places$Longitude,na.rm=TRUE))
+ 
+ ## centering
+ m <- c(median(places$Latitude,na.rm=TRUE),
+ median(places$Longitude,na.rm=TRUE))
+ 
+ ## initial map
+ MyMap <- GetMap.bbox(bb$lonR,
+ bb$latR,
+ center=m,
+ NEWMAP=TRUE,
+ maptype="terrain",
+ destfile="tmpmap.jpg",
+ RETURNIMAGE=TRUE,
+ GRAYSCALE=FALSE,
+ verbose=0)
+ 
+ ## manually convert to gray
+ MyMap <- grayConvert(MyMap)
+ 
+ ## jpeg creation
+ cat(paste("creating file",fName,"\n"))
+ jpeg(file=fName,
+ width=640,
+ height=640,
+ quality=100)
+ 
+ ## division lines
+ n <- length(division)
+ for(i in 1:n){
+ l <- linesDivision(division[i])
+ if(i==1){
+ tmp <- PlotOnStaticMap(MyMap=MyMap,
+ NEWMAP=FALSE,
+ destfile="tmpmap.jpg",
+ lat=l[1,"lat"],
+ lon=l[1,"long"],
+ FUN=points,
+ col=rgb(r=1,g=165/255,b=0,alpha=.5),
+ cex=.1,
+ verbose=FALSE,
+ GRAYSCALE=TRUE,
+ add=FALSE)
+ } } 
+ 
+ ## overlay some polling place stuff
+ cex <- 1.5
+ alpha <- .75
+ lwd <- 7
+ if(n>2){
+ cex <- .75
+ alpha <- .9
+ lwd <- 3
+ }
+ 
+ if(is.factor(l$piece))
+ l$piece <- match(l$piece,levels(l$piece))
+ J <- max(l$piece)
+ if(J>0){
+ for(j in 1:J){
+ ok <- l$piece==j
+ PlotOnStaticMap(MyMap=MyMap,
+ NEWMAP=FALSE,
+ destfile="mymap.jpg",
+ lat=l[ok,"lat"],
+ lat=l[ok,"long"],
+ FUN=points,
+ col=rgb(r=1,g=165/255,b=0,alpha=.5),
+ cex=.1,
+ verbose=FALSE,
+ GRAYSCALE=TRUE,
+ add=FALSE)
+ } } 
+ 
+ # calculating number of points
+ nPoints <- length(places$Place)
## points for polling places
for(i in 1:n)
  ok <- places$DivisionNm==division[i]
  PlotOnStaticMap(MyMap=MyMap,
                  NEWMAP=FALSE,
                  destfile="mymap.jpg",
                  lat=places[ok,"Latitude"],
                  lon=places[ok,"Longitude"],
                  FUN=points,
                  col="black",
                  bg=cols[ok],
                  lwd=.33,
                  pch=21,
                  cex=cexPoints[ok],
                  add=TRUE,
                  verbose=FALSE)
}

## legend
legendFunc(MyMap,division,type,party,places)

## title bar, bottom of graph
titleBar(MyMap,divisionOriginal,type,party)
dev.off()
We write a utility function that returns the limits (in lat/long co-ords) of the current map, extracting the `BBOX` component of the map:

```r
getLims <- function(MyMap) {
  lims <- matrix(c(MyMap$BBOX$ll[, "lon"],
                  MyMap$BBOX$ur[, "lon"],
                  MyMap$BBOX$ll[, "lat"],
                  MyMap$BBOX$ur[, "lat"],
                  2, 2, byrow=TRUE)
  return(lims)
}
```

We create a function that generates a caption bar at the bottom of the graph:

```r
titleBar <- function(MyMap, division, type, party) {
  ## put a polygon on the bottom 4 percent of the graph
  lims <- getLims(MyMap)
  ul <- c(x=lims[1,1],
          y=-.94*lims[2,1] + .06*lims[2,2])
  lr <- c(x=lims[1,2],
          y=lims[2,1])
  }
Definitions of some supporting functions. We gather up some collections of division names so we can call them with a sensible tag (e.g., “Sydney Metro” etc).

R Code

```r
> parseDivisionNames <- function(divisions) {
+ reservedNames <- c("all",
+ "Sydney Harbourside",
+ "Sydney Metro",
+ "Brisbane Metro",
+ "Melbourne Metro",
+ "Adelaide Metro",
+ "Perth Metro",
+ "Tasmania",
+ "South East Queensland",
+ "Sydney Basin")
+ }
+ ```
A function for cleaning out small polling places that lie a long way from the remainder of the data for a given polling place, pre-polls, postals, "special hospital votes" and some other error-screening.

```r
screenPollingPlaces <- function(places, division, debug) {
  tooEast <- places$Longitude > (153 + 36/60 + 54.47/3600)
  tooWest <- places$Longitude < (113 + 9/60)

  division <- unique(res$DivisionNm),
  "Sydney Harbourside" = c("Wentworth", "Sydney", "Grayndler", "Reid",
    "Bennelong", "North Sydney", "Warringah"),
  "Sydney Metro" = c("Wentworth", "Sydney", "Grayndler", "Reid",
    "Bennelong", "North Sydney", "Warringah",
    "Kingsford Smith", "Barton",
    "Watson", "Blaxland", "Parramatta",
    "Fowler", "Werriwa", "Chifley",
    "Greenway", "Mitchell", "Bradfield",
    "MacKellar", "Lindsay", "McMahon",
    "Werriwa", "Hughes", "Cook"),
  "Sydney Basin" = c("Wentworth", "Sydney", "Grayndler", "Reid",
    "Bennelong", "North Sydney", "Warringah",
    "Kingsford Smith", "Barton",
    "Watson", "Blaxland", "Parramatta",
    "Fowler", "Werriwa", "Chifley",
    "Greenway", "Mitchell", "Bradfield",
    "MacKellar", "Lindsay", "McMahon",
    "Werriwa", "Hughes", "Cook",
    "Macarthur", "Cunningham", "Thorsby",
    "Macquarie", "Berejiklian", "Robertson",
    "Dobell", "Charlton", "Shortland",
    "Newcastle"),
  "Melbourne Metro" = c("Gorton", "Maribyrnong",
    "Wills", "Melbourne", "Wills", "Batman",
    "Scullin", "JagaJaga", "Menzies", "Kooyong",
    "Deakin", "Higgins", "Melbourne Ports",
    "Goldstein", "Hotham", "Chisholm", "Bruce",
    "Aston", "Isaacs", "Holt", "Dunkley",
    "Gellibrand"),
  "Brisbane Metro" = c("Lilley", "Brisbane",
    "Bonner", "Bowman", "Rankin", "Moreton",
    "Dickson",
    "Griffith", "Oxley", "Ryan", "Petrie"),
  "South East Queensland" = c("Lilley", "Brisbane",
    "Bonner", "Bowman", "Rankin", "Moreton",
    "Griffith", "Oxley", "Ryan", "Petrie",
    "Dickson", "Longman", "Fisher", "Fairfax",
    "Blair", "Wright", "Forde", "Fadden",
    "Moncrieff", "McPherson", "Groom"),
  "Adelaide Metro" = c("Port Adelaide", "Makin",
    "Sturt", "Adelaide", "Hindmarsh", "Boothby",
    "Kingston"),
  "Perth Metro" = c("Fremantle", "Tangney",
    "Swan", "Hasluck", "Curtin", "Perth",
    "Stirling", "Cowen", "Moore"),
  "Tasmania" = c("Bass", "Braddon", "Franklin",
    "Lyons", "Denison")
  }
  divisionLocal <- division
  ok <- divisionLocal %in% unique(ppData$DivisionNm)
  return(divisionLocal[ok])
}
```

R Code

```r
> screenPollingPlaces <- function(places, division, debug){
+ tooEast <- places$Longitude>(153 + 36/60 + 54.47/3600)
+ tooWest <- places$Longitude<(113 + 9/60)
+ }
```
badPlaces <- tooEast | tooWest

tmpData <- places[!badPlaces,]

## clobber missing division
tmpData <- tmpData[!is.na(tmpData$DivisionNm),]

## dump Jervis Bay from Fraser
if("Fraser" %in% division)
  tmpData <- tmpData[!tmpData$DivisionNm=="Fraser" & tmpData$Longitude>149.3993],]

## dump Moreton Island from Bonner
if("Bonner" %in% division)
  tmpData <- tmpData[tmpData$PollingPlaceNm!="Tangalooma",]

## dump Moreton Bay Islands from Bowman
if("Bowman" %in% division)
  tmpData <- tmpData[!(tmpData$DivisionNm=="Bowman" & tmpData$Longitude>153.3060),]

## Bass Strait
if("Bass" %in% division)
  tmpData <- tmpData[!(tmpData$DivisionNm=="Bass" & tmpData$Latitude > -40.9),]

## screen out capital cities
dNames <- unique(tmpData$DivisionNm)
j <- length(dNames)
keep1 <- rep(TRUE, dim(tmpData)[1])

## loop over divisions
for(i in 1:j){
  if(debug)
    cat(paste("This division is",dNames[i],"\n"))

  ## kill capital city votes
  ## except if we are in divisions called "Brisbane", "Sydney",
  ## "Melbourne", "Adelaide", "Perth" etc

  theCapital <- capitalCity(dNames[i])
  if(dNames[i]!=theCapital){
    badLoc <- paste(theCapital,
                   " (",dNames[i],")",
                   sep="")
    if(debug):
      cat(paste("searching for",badLoc,"\n"))
    cat("table on exact match:\n")
    print(table(tmpData$PollingPlaceNm==badLoc))
    cat("grep:\n")
    print(grep(badLoc,tmpData$PollingPlaceNm))
    }

  keep1[tmpData$PollingPlaceNm==badLoc] <- FALSE

  ## kill "City" votes
  badLoc <- paste(capitalCity(division[i]),
                   "City")
  if(debug)
    print(badLoc)
  keep1[grep(badLoc,tmpData$PollingPlaceNm)] <- FALSE
}

if(debug)
  cat(paste("found",sum(!keep1),"bad locations\n"))
## clobber PREPOLL
keep3 <- rep(TRUE, dim(tmpData)[1])
keep3[expand.grid("PREPOLL", tmpData$PollingPlaceNm)] <- FALSE
keep3[expand.grid("PPVC", tmpData$PollingPlaceNm)] <- FALSE
if(debug)
  cat(paste("found", sum(!keep3), "PREPOLL/PPVC locations\n"))

## clobber "Special Hospital"
keep4 <- rep(TRUE, dim(tmpData)[1])
keep4[expand.grid("Special Hospital", tmpData$PollingPlaceNm)] <- FALSE
if(debug)
  cat(paste("found", sum(!keep4), "special hospital locations\n"))

## missing geo data
keep5 <- !is.na(tmpData$Latitude) & !is.na(tmpData$Longitude)
if(debug)
  cat(paste("found", sum(!keep5), "location with no geodata\n"))

return(tmpData[keep1 & keep3 & keep4 & keep5,])

#### state capital look up function
stateCapital <- function(state){
  switch(state,
    "NSW"="Sydney",
    "VIC"="Melbourne",
    "QLD"="Brisbane",
    "SA"="Adelaide",
    "WA"="Perth",
    "TAS"="Hobart",
    "NT"="Darwin",
    "ACT"="Canberra")
}

stateLookUp <- function(division){
  res$StateAb[match(division, res$DivisionNm)]
}

capitalCity <- function(division){
  stateCapital(stateLookUp(division))
}

### makeFileName
makeFileName <- function(division, party, type){
  reservedNames <- c("all",
                    "Sydney Harbourside",
                    "Sydney Metro",
                    "Brisbane Metro",
                    "Melbourne Metro",
                    "Adelaide Metro",
                    "Perth Metro",
                    "Tasmania",
                    "South East Queensland",
                    "Sydney Basin")
  if(division %in% reservedNames){
    fName <- reservedNames[reservedNames==division]
    fName <- gsub(fName, pattern=" ", replacement="")
  } else{
    fName <- division
  }
  if(type=="primary")
    fName <- paste(fName, party, type, sep="_")
  if(type=="tpp" | type=="swing")
}
4.1 Colors

Some functions for managing colors. The `grayConvert` function converts the Google Maps jpg to grayscale. The `colorMaker2` function generates some reasonable sensible colors for overlaying vote shares on the maps, making use of the `RColorBrewer` package.

```R
> grayConvert <- function(obj){
+ myTile <- obj$myTile
+ tmp <- myTile
+ myTile <- myTile[, , 1]
+ myTile <- 0.3*tmp[, , 1] + 0.59*tmp[, , 2] + 0.11*tmp[, , 3]
+ attr(myTile, "class") <- c("imagematrix", "array")
+ attr(myTile, "type") <- "grey"
+ obj$myTile <- myTile
+ return(obj)
}

> colorMaker2 <- function(x,party,type,alpha){
+ require(RColorBrewer)
+ 
+ ## two-party preferred, swing, colors
+ if(type=="swing"){
+ z <- c(rev(brewer.pal(n=9,"Blues")),
+ brewer.pal(n=9,"Reds"))
+ z <- z[-c(1,length(z))]
+ maxDiff <- max(abs(x),na.rm=TRUE)
+ zBreaks <- seq(-maxDiff,maxDiff,length=length(z)+1)
+ z <- z[as.numeric(cut(x,breaks=zBreaks,include.lowest=TRUE))]
+ }
+ 
+ ## two-party preferred, colors
+ if(type=="tpp"){
+ z <- c(rev(brewer.pal(n=9,"Blues")),
+ brewer.pal(n=9,"Reds"))
+ z <- z[-c(1,length(z))]
+ maxDiff <- max(abs(50-x),na.rm=TRUE)
+ zBreaks <- seq(50-maxDiff,50+maxDiff,
+ length=length(z)+1)
+ z <- z[as.numeric(cut(x,breaks=zBreaks,include.lowest=TRUE))]
+ }
+ 
+ ## primary votes
+ if(type=="primary"){
+ if(party=="ALP"){
+ z <- brewer.pal(n=9,name="Reds")
+ z <- z[as.numeric(cut(x,
+ breaks=seq(min(x,na.rm=TRUE),
+ max(x,na.rm=TRUE),
+ length=10),
+ include.lowest=TRUE))]
+ }
+ 
+ if(party=="Coalition"){
+ z <- brewer.pal(n=9,name="Blues")
+ z <- z[as.numeric(cut(x,
+ breaks=seq(min(x,na.rm=TRUE),
+ max(x,na.rm=TRUE),
+ length=10),
+ include.lowest=TRUE))]
+ }
```

5 Using the functions

We give the `makeGoogleMap` function a test-spin, looking at the ALP’s 2PP vote shares at polling places in the division of Brisbane. The line of code below produces Figure 2.

```R
> makeGoogleMap(division="Brisbane",type="tpp",s=7)
```

```
creating file jpg/Brisbane_tpp.jpg
```

We also take a look at the functionality we’ve created for displaying primary vote shares. This time we take the seat of Melbourne, looking at Green primary vote shares by polling place, with the resulting map appearing in Figure 3.

```R
> makeGoogleMap(division="Melbourne",type="primary",party="GRN",s=7)
```

```
creating file jpg/Melbourne_GRN_primary.jpg
```

We verify that these data are in fact correct, by examining the raw polling place data for the Abbotsford booth in the division of Melbourne, as we read it from the AEC site.

```R
> tmpData <- res[res$DivisionNm=="Melbourne",
+  c("PollingPlace",
+  "PollingPlaceID",
+  "PartyAb",
+  "BallotPosition",
+  "OrdinaryVotes")]
> tmpData <- tmpData[tmpData$PollingPlace=="Abbotsford",]
> g <- tmpData$OrdinaryVotes[tmpData$PartyAb=="GRN"]
> total <- sum(tmpData$OrdinaryVotes[tmpData$BallotPosition!=999])
> g
```

18
Figure 2: Brisbane, with Labor 2PP vote percentage by polling place.
We also look at the `voteShares` object:

```r
> voteShares["4610"]
```

<table>
<thead>
<tr>
<th>Total</th>
<th>Informal</th>
<th>FFP</th>
<th>GRN</th>
<th>ASXP</th>
<th>DEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1616.00000000</td>
<td>4.0380048</td>
<td>0.9900990</td>
<td>42.8836634</td>
<td>1.3613861</td>
<td>0.7425743</td>
</tr>
<tr>
<td>SPA</td>
<td>0.3712871</td>
<td>35.5816832</td>
<td>18.0693069</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We compare this output with that from the `getPrimaryVotes` extraction function:

```r
> getPrimaryVotes(division="Melbourne",party="GRN")[1,]
```

<table>
<thead>
<tr>
<th>total</th>
<th>x</th>
<th>division</th>
<th>PollingPlace</th>
<th>PollingPlaceID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1616</td>
<td>42.88366</td>
<td>Melbourne</td>
<td>Abbotsford</td>
<td>4610</td>
</tr>
</tbody>
</table>

Satisfied that the data are lining up ok, we look at a few more primary vote maps, with the following code generating Figures 4 to 8.

```r
> makeGoogleMap(division="Grayndler",type="primary",party="GRN",s=7)
creating file jpg/Grayndler_GRN_primary.jpg
```

```r
> makeGoogleMap(division="Batman",type="primary",party="GRN",s=7)
creating file jpg/Batman_GRN_primary.jpg
```

```r
> makeGoogleMap(division="Fowler",type="primary",party="Informal",s=7)
creating file jpg/Fowler_Informal_primary.jpg
```

```r
> makeGoogleMap(division="Kingston",type="swing",s=7)
creating file jpg/Kingston_swing.jpg
```

```r
> makeGoogleMap(division="Bennelong",type="swing",s=7)
creating file jpg/Bennelong_swing.jpg
```
Figure 3: Melbourne, Green first preferences by polling place.
Figure 4: Grayndler, Green first preferences by polling place.
Figure 5: Batman, Green first preferences by polling place.
Figure 6: Fowler, informality by polling place.
Figure 7: Kingston, Swing in ALP 2PP by polling place.
Figure 8: Bennelong, Swing in ALP 2PP by polling place.
5.1 Collections of Divisions

We look at some of our “aggregated” graphs too. For instance, Figure 9 shows ALP 2PP results by polling place throughout the Melbourne area. 2PP data is not available in the division of Melbourne, but a reasonably stark geographic partitioning of Labor and Coalition support is quite apparent. When generating a plot with a large number of polling places we set the scale parameter $s$ to a relatively small value; here we try $s=2$.

```
R Code
> makeGoogleMap(division="Melbourne Metro", +   type="tpp", +   s=2)
creating file jpg/MelbourneMetro_tpp.jpg
```

```
R Code
> makeGoogleMap(division="Melbourne Metro", +   type="primary", +   party="GRN", +   s=2)
creating file jpg/MelbourneMetro_GRN_primary.jpg
```

```
R Code
> makeGoogleMap(division="Sydney Metro", +   type="tpp", +   s=2)
creating file jpg/SydneyMetro_tpp.jpg
```

```
R Code
> makeGoogleMap(division="Brisbane Metro", +   type="primary",party="GRN",s=2)
creating file jpg/BrisbaneMetro_GRN_primary.jpg
```

```
R Code
> makeGoogleMap(division="Brisbane Metro", +   type="swing",s=2)
creating file jpg/BrisbaneMetro_swing.jpg
```

Green primary vote share in the Brisbane metro area:

Swing in ALP 2PP vote shares in the Brisbane metro area:
Figure 9: Melbourne Metropolitan Area, Labor 2PP vote percentage by polling place.
Figure 10: Melbourne Metropolitan Area, Green primary vote share by polling place.
Figure 11: Sydney Metropolitan Area, Labor 2PP vote percentage by polling place.
Figure 12: Brisbane Metropolitan Area, Green primary vote share by polling place.
Figure 13: Brisbane Metropolitan Area, Labor 2PP swing by polling place.
We also create some of the other “aggregations” that we pre-defined in the `parseDivisionNames` function. We also create maps showing the distribution of Green support and rates of informal balloting.

```r
> for(division in theNames){
  + makeGoogleMap(division=division,type="tpp",s=2)
  + makeGoogleMap(division=division,type="primary",party="GRN",s=2)
  + makeGoogleMap(division=division,type="primary",party="Informal",s=2)
  + }
```

We now make all the division-level TPP jpgs, excluding the seats that are still considered “non-classic” and without a ALP/Coalition 2PP count.

```r
> theDivisions <- unique(ppData$DivisionNm)
  + skip=1,
  + sep="","",
  + stringsAsFactors=FALSE,
  + quote="\\"
  + header=TRUE)
> for(division in theDivisions){
  + if (!(division %in% nonClassic$DivisionNm)){
  + makeGoogleMap(division=division,type="tpp",s=7)
  + }
  + }
```