Example: Mosaic in Action

north carolina (simple + county)

Matt Mohn

2025-12-16

```
## Load necessary packages
library(sf)
library(tidyverse)
library(igraph)
library(sf)
library(readr)
library(ggplot2)
library(dplyr)
library(shadowtext)
library(scales)


## Load mosaic files
source("timing_functions.R")
source("scoring_functions.R")
source("tree_functions.R")
source("partition_functions.R")
source("recom_functions.R")
source("graph_functions.R")
source("output_functions.R")
source("chain_runner.R")
source("cleanup_globals.R")
source("graphics.R")
```

Example 1: Simple Annealing

Let's setup and run the chain.

We'll use a generalized North Carolina VTD shapefile (generalization simplifies the shapefile - this doesn't affect the annealing in any way but makes plotting our results faster). It's saved here as `North_Carolina_Simplified.shp`

By default, Mosaic will run for 1000 iterations with a `pdev_tolerance` (acceptable population deviation) of +/-5%, and applies a cut edge weight of 1 (`weight_cut_edges` = 1). It will produce 5 districts by default, but we want North Carolina to have the 14 congressional districts it has in real life, so we set `num_districts` to 14 here.

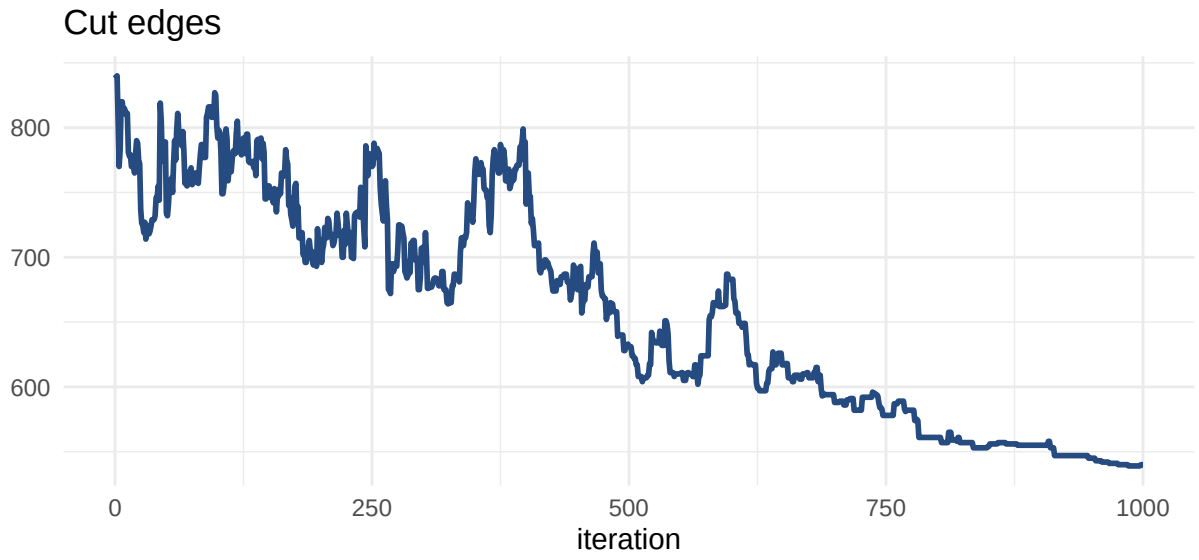
```
SHAPEFILE <- "shapefiles/North_Carolina_Simplified.shp"

results <- run_chain(
  shapefile_path = SHAPEFILE,
  num_districts = 14,
  seed=123456
)
```

```
## Loading shapefile: shapefiles/North_Carolina_Simplified.shp
## Loading cached graph from: cache/cache_North_Carolina_Simplified.rds
## Loaded 2666 precincts
## Total population: 10,439,388
## Ideal per district: 745,671
## Population tolerance: 5.0%
##
## County data found: 100 unique counties
##
## Creating initial partition...
## GUIDED cooling: T0=167.6 to T=1.0 at iteration 900 (90%) at rate=0.994326
## Optimization enabled: Initial score=838.0, Temp=167.6
##
## Running 1000 ReCom steps...
##
## ITER      SCORE     TEMP      CUTS   CNTY   BUNK     MM DIFF   EF GAP    D     R     C      % ACC
## 100       792       94.9      792    --     -        --        --        --    --    --     92.0
## 200       704       53.7      704    --     -        --        --        --    --    --     92.0
## 300       708       30.4      708    --     -        --        --        --    --    --     90.0
## 400       741       17.2      741    --     -        --        --        --    --    --     84.0
##
## ITER      SCORE     TEMP      CUTS   CNTY   BUNK     MM DIFF   EF GAP    D     R     C      % ACC
## 500       631       9.7       631    --     -        --        --        --    --    --     66.0
## 600       683       5.5       683    --     -        --        --        --    --    --     46.0
## 700       588       3.1       588    --     -        --        --        --    --    --     36.0
## 800       561       1.8       561    --     -        --        --        --    --    --     22.0
## 900       555       1.0       555    --     -        --        --        --    --    --     10.0
##
## ITER      SCORE     TEMP      CUTS   CNTY   BUNK     MM DIFF   EF GAP    D     R     C      % ACC
## 1000      540       0.6       540    --     -        --        --        --    --    --     16.0
##
## Completed in 13.3s (13.3ms/step, 75.4 iter/s)
## Optimization stats: 563 accepts, 437 rejects (56.3% accept rate)
## Score: 838.0 to 540.0 (D=-298.0)
##
## ========================================
## FINAL ANALYSIS
```

```
## ========================================
## ========================================
##
##
## ========================================
## RESULTS SAVED
## ========================================
## Final assignment: output/final_assignment_12182025_0033.csv
## Metrics: output/metrics_12182025_0033.csv
## Snapshots: output/snapshot_assignments_12182025_0033.csv
## ========================================
```

The results of the annealing are encouraging: the program initialized with 792 cuts (which isn't bad itself) and then fell to 540 over the course of the run.

The score perfectly matches the number of cuts because that's the only weight that we used.

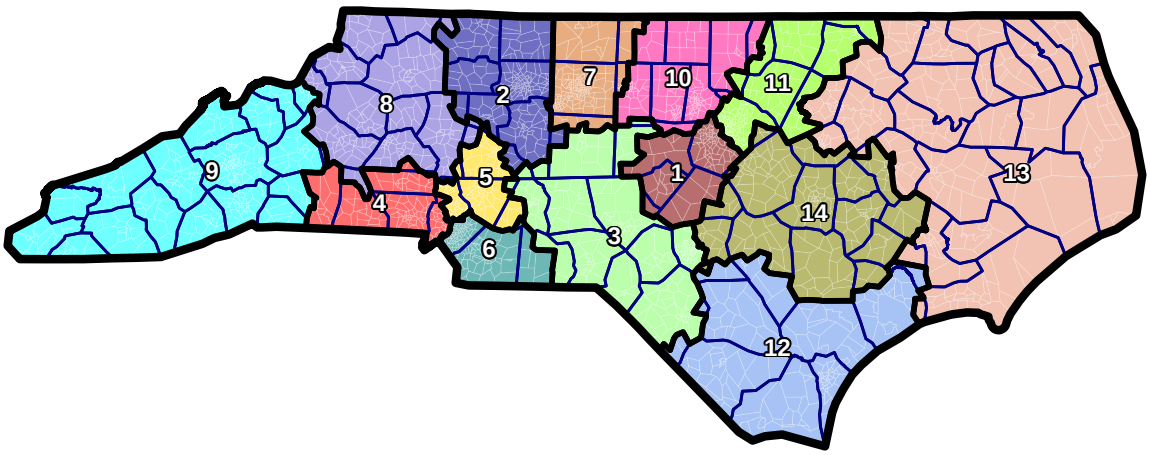


Knowing the results is only a part of the fun. Let's see what the map actually looks like.

Mosaic includes the `mosaic_plot` file, which can create nice-looking maps on the fly.

```
# By default, mosaic_plot will call from the most recently
# saved final_assignments file.

p <- mosaic_plot(shapefile_path = SHAPEFILE,
                 type="simple",
                 title="North Carolina: Example 1",
                 subtitle="Congressional districts, no weights",
                 border_outline = TRUE,
                 district_outline = TRUE,
                 county_outline = TRUE,
                 number_labels = TRUE,
```

```
                  precinct_outline = TRUE)
p
```



It looks great! All of the districts are fairly compact and there aren't many weird shapes. One issue jumps out though - we're splitting counties all over the place (look at Charlotte or Raleigh!) That makes sense- we didn't tell Mosaic to try in the first place. For that, let's move to example 2.

Example 2: Preserving Counties

To better preserve counties, we take the same `run_chain()` call from earlier- but make two critical changes.

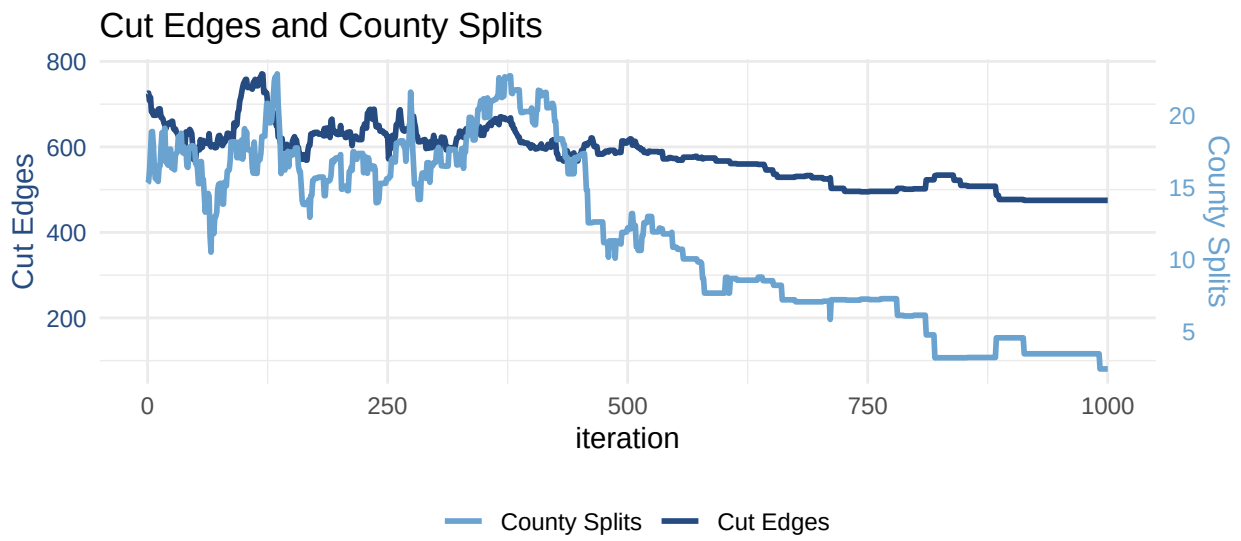
- First, we apply `county_bias`. The recombination algorithm usually randomly selects connections in the spanning graph to cut. However, we can induce it to draw cuts *between* counties, instead of *within* them, by increasing the sampling probability of edges that are also county boundaries. By setting `county_bias` to 10, we make it 10 times as likely that a county boundary is selected compared to a typical connection.
- Second, although `county_bias` will naturally lead to maps with fewer county splits, we also want the algorithm to try and value low-splitting as a general practice. We set `weight_county_splits` to 15. There are no units involved, but by default this means that 1 county split will be worth as much as 15 additional cut edges in annealing.

```
SHAPEFILE <- "shapefiles/North_Carolina_Simplified.shp"

results <- run_chain(
  shapefile_path = SHAPEFILE,
  num_districts = 14,
  seed=12345,
  county_bias=10,
  weight_county_splits=15,
)
```

```
## Loading shapefile: shapefiles/North_Carolina_Simplified.shp
## Loading cached graph from: cache/cache_North_Carolina_Simplified.rds
## Loaded 2666 precincts
## Total population: 10,439,388
## Ideal per district: 745,671
## Population tolerance: 5.0%
##
## County data found: 100 unique counties
##
## Creating initial partition...
## GUIDED cooling: T0=191.0 to T=1.0 at iteration 900 (90%) at rate=0.994181
## Optimization enabled: Initial score=954.9, Temp=191.0
## County-aware mode: bias factor = 10.0
##
## Running 1000 ReCom steps...
##
## ITER      SCORE     TEMP      CUTS    CNTY    BUNK      MM DIFF    EF GAP     D      R      C      % ACC
## 100       1021      106.5     749     18.10   -         --         --         --     --     --     96.0
## 200       891       59.4      634     17.14   -         --         --         --     --     --     80.0
## 300       865       33.2      614     16.76   -         --         --         --     --     --     86.0
## 400       913       18.5      605     20.55   -         --         --         --     --     --     76.0
##
## ITER      SCORE     TEMP      CUTS    CNTY    BUNK      MM DIFF    EF GAP     D      R      C      % ACC
## 500       797       10.3      614     12.21   -         --         --         --     --     --     50.0
## 600       682       5.8       567     7.67    -         --         --         --     --     --     28.0
## 700       634       3.2       528     7.07    -         --         --         --     --     --     30.0
## 800       594       1.8       502     6.13    -         --         --         --     --     --     18.0
## 900       545       1.0       477     4.56    -         --         --         --     --     --     14.0
##
## ITER      SCORE     TEMP      CUTS    CNTY    BUNK      MM DIFF    EF GAP     D      R      C      % ACC
## 1000      511       0.6       475     2.40    -         --         --         --     --     --     20.0
##
## Completed in 14.4s (14.4ms/step, 69.7 iter/s)
```

```
## Optimization stats: 516 accepts, 484 rejects (51.6% accept rate)
## Score: 954.9 to 511.0 (D=-443.9)
##
## ========================================
## FINAL ANALYSIS
## ========================================
##
## County Splits Score: 2.40
## ========================================
##
##
## ========================================
## RESULTS SAVED
## ========================================
## Final assignment: output/final_assignment_12182025_0034.csv
## Metrics: output/metrics_12182025_0034.csv
## Snapshots: output/snapshot_assignments_12182025_0034.csv
## ========================================
```

The initial score (954) reflects the number of cut edges (725) plus 10 times the number of county splits (15.3). After 1000 iterations, the score has fallen to 511, with 475 cut edges and just 2.40 county splits.



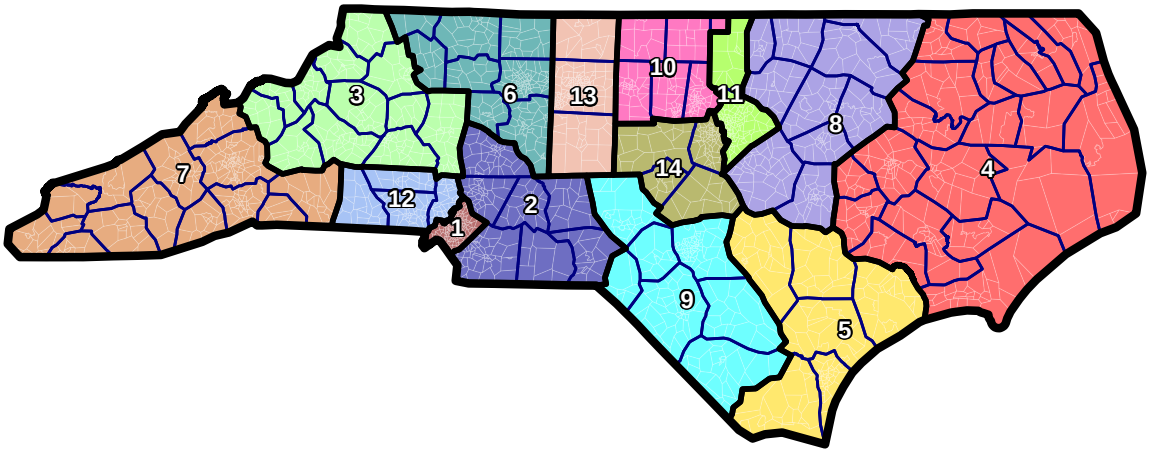
For due diligence's sake, let's plot the map just to be sure.

```
p <- mosaic_plot(shapefile_path = SHAPEFILE,
                 type="simple",
                 title="North Carolina: Example 2",
                 subtitle="Congressional districts with county weights",
                 border_outline = TRUE,
                 district_outline = TRUE,
                 county_outline = TRUE,
                 number_labels = TRUE,
```

```
                  precinct_outline = TRUE)
p
```

North Carolina: Example 2

Congressional districts with county weights



Looks good!

There are different ways of measuring county splits. Four counties are split here (Granville, Mecklenburg, Wake, and Wilkes) but because Meck and Wake are larger than a district each, we subtract them from the total, taking us down to 2.

The remaining 0.4 comes from a measure of fragmentation- this penalizes splits that are especially egregious.

Now that we know how to draw goegraphically nice-looking maps, Example 3 will explore using partisan data.

Example 3: Republican Gerrymander

For Example 3, let's attempt to draw a Republican gerrymander- which, not coincidentally, is what the North Carolina General Assembly has been doing for the past two decades anyway. Learn from the best, right?

Let's use the following settings:

- Decrease `county_bias` to 2. First off, we care less about county splits this time around. Second, when we're looking to pull off partisan objectives, we don't want to constrain the set of possible combinations available to us.
- Decrease `weight_county_splits` to 5- again, nice to have, but not something mission-critical.
- Increase `num_steps` to 3000 up from the default 1000, to give us more room to maneuver.
- Set `target_efficiency_gap` to 0.5, meaning we aim for Republican votes to be 40% more efficient than Democratic ones. We weight this a 100 using `weight_efficiency_gap`.
- Set `target_mean_median` to 0.15, meaning we want the median district in North Carolina to be 15% more Republican by share of the vote than the state as a whole. If the state votes for a Democrat, we still want a majority to be won by Republicans. We weight this a 50 using `weight_mean_median`.
- Finally, set `target_dem_seats` to 2. This is pretty self-explanatory! We could aim for 0, but that would be very difficult and we don't want the score to be too meaningless. We weight this even further, to 250, using `weight_dem_seats`.

```
SHAPEFILE <- "shapefiles/North_Carolina_Simplified.shp"

results <- run_chain(
  shapefile_path = SHAPEFILE,
  num_districts = 14,
  seed=123456,
  county_bias=2,
  weight_county_splits=5,
  target_efficiency_gap=0.5,
  weight_efficiency_gap=100,
  target_mean_median=0.15,
  weight_mean_median=50,
  num_steps=3000,
  target_dem_seats=2,
  weight_dem_seats=250
)
```

```
## Loading shapefile: shapefiles/North_Carolina_Simplified.shp
## Loading cached graph from: cache/cache_North_Carolina_Simplified.rds
## Loaded 2666 precincts
## Total population: 10,439,388
## Ideal per district: 745,671
## Population tolerance: 5.0%
##
## County data found: 100 unique counties
##
## Creating initial partition...
## GUIDED cooling: T0=978.9 to T=1.0 at iteration 2700 (90%) at rate=0.997453
## Optimization enabled: Initial score=4894.7, Temp=978.9
## County-aware mode: bias factor = 2.0
##
## Running 3000 ReCom steps...
##
## ITER      SCORE     TEMP     CUTS   CNTY   BUNK     MM DIFF   EG ADJ    D     R     C     % ACC
## 100       4830      758.6    778    57.52  -        +1.9      +12.9     5.0   9.0   3.3   88.0
## 200       2982      587.8    724    53.18  -        +2.8      +20.9     4.1   9.9   2.2   90.0
## 300       4344      455.5    783    59.56  -        +2.2      +13.7     4.7   9.3   3.1   80.0
```

```
## 400       4527      352.9     775     64.84   -         +2.0      +14.6     4.9     9.1     3.7     78.0
##
## ITER      SCORE     TEMP      CUTS    CNTY    BUNK      MM DIFF   EG ADJ    D       R       C       % ACC
## 500       3797      273.5     766     61.73   -         +2.1      +16.9     4.5     9.5     2.9     86.0
## 600       3139      211.9     719     58.95   -         +3.3      +20.0     4.1     9.9     2.3     84.0
## 700       2982      164.2     632     52.99   -         +3.6      +19.9     4.1     9.9     2.2     88.0
## 800       3006      127.2     736     52.22   -         +3.4      +19.8     4.0     10.0    2.1     82.0
## 900       3015      98.6      801     57.14   -         +3.1      +21.1     4.0     10.0    2.1     70.0
##
## ITER      SCORE     TEMP      CUTS    CNTY    BUNK      MM DIFF   EG ADJ    D       R       C       % ACC
## 1000      2713      76.4      737     47.47   -         +3.1      +22.0     3.9     10.1    1.8     68.0
## 1100      2864      59.2      681     56.51   -         +2.5      +21.2     4.0     10.0    2.0     70.0
## 1200      2811      45.9      684     52.59   -         +3.1      +22.4     4.0     10.0    2.1     66.0
## 1300      2653      35.5      716     48.09   -         +4.0      +22.6     3.9     10.1    1.8     48.0
## 1400      2519      27.5      682     50.08   -         +3.9      +23.1     3.8     10.2    1.6     54.0
##
## ITER      SCORE     TEMP      CUTS    CNTY    BUNK      MM DIFF   EG ADJ    D       R       C       % ACC
## 1500      2411      21.3      675     46.13   -         +3.8      +24.1     3.8     10.2    1.5     50.0
## 1600      2388      16.5      656     46.48   -         +3.6      +23.8     3.7     10.3    1.5     48.0
## 1700      2450      12.8      639     41.99   -         +3.2      +23.3     3.8     10.2    1.6     44.0
## 1800      2174      9.9       674     38.77   -         +5.1      +24.5     3.6     10.4    1.1     38.0
## 1900      2227      7.7       708     42.09   -         +4.7      +24.3     3.5     10.5    1.1     24.0
##
## ITER      SCORE     TEMP      CUTS    CNTY    BUNK      MM DIFF   EG ADJ    D       R       C       % ACC
## 2000      2168      6.0       655     42.56   -         +4.4      +24.5     3.5     10.5    1.1     22.0
## 2100      2156      4.6       694     42.60   -         +4.6      +24.8     3.5     10.5    1.0     14.0
## 2200      2110      3.6       656     40.70   -         +4.7      +25.1     3.5     10.5    1.0     12.0
## 2300      2105      2.8       658     40.62   -         +4.7      +24.9     3.5     10.5    1.0     16.0
## 2400      2069      2.1       636     35.20   -         +5.0      +24.7     3.5     10.5    1.0     8.0
##
## ITER      SCORE     TEMP      CUTS    CNTY    BUNK      MM DIFF   EG ADJ    D       R       C       % ACC
## 2500      2062      1.7       623     34.61   -         +4.7      +24.7     3.5     10.5    1.0     10.0
## 2600      2055      1.3       623     33.36   -         +4.8      +24.7     3.5     10.5    1.0     6.0
## 2700      2043      1.0       620     33.57   -         +5.0      +24.7     3.5     10.5    1.0     8.0
## 2800      2043      0.8       627     33.22   -         +5.0      +24.8     3.5     10.5    1.0     2.0
## 2900      2045      0.6       631     33.23   -         +5.0      +24.8     3.5     10.5    1.0     6.0
##
## ITER      SCORE     TEMP      CUTS    CNTY    BUNK      MM DIFF   EG ADJ    D       R       C       % ACC
## 3000      2045      0.5       631     33.23   -         +5.0      +24.8     3.5     10.5    1.0     8.0
##
## Completed in 49.0s (16.3ms/step, 61.3 iter/s)
## Optimization stats: 1361 accepts, 1639 rejects (45.4% accept rate)
## Score: 4894.7 to 2044.6 (D=-2850.2)
##
## ========================================
## FINAL ANALYSIS
## ========================================
## District          DEM        REP     DEM%     REP%   Margin   P(DEM) Winner
## ------------------------------------------------------------------------------
## D1              165791     224162    42.5%    57.5%    15.0%     3.6% REP
## D2              279827     161550    63.4%    36.6%    26.8%    99.7% DEM
## D3              160533     198429    44.7%    55.3%    10.6%     9.0% REP
## D4              175933     228023    43.6%    56.4%    12.9%     5.6% REP
## D5              171118     224416    43.3%    56.7%    13.5%     4.9% REP
## D6              168377     222809    43.0%    57.0%    13.9%     4.5% REP
## D7              170273     227545    42.8%    57.2%    14.4%     4.1% REP
## D8              160673     241431    40.0%    60.0%    20.1%     1.2% REP
## D9              190138     236741    44.5%    55.5%    10.9%     8.3% REP
## D10             167491     213946    43.9%    56.1%    12.2%     6.4% REP
```

```
## D11            323387     114775    73.8%    26.2%    47.6%   100.0% DEM
## D12            250579      87110    74.2%    25.8%    48.4%   100.0% DEM
## D13            172067     243616    41.4%    58.6%    17.2%     2.2% REP
## D14            157422     272388    36.6%    63.4%    26.7%     0.3% REP
## ------------------------------------------------------------------------------
## TOTAL         2713609    2896941    48.4%    51.6%
##
## Mean Dem Share:   48.4%
## Median Dem Share: 43.4%
## MM: R+5.0% (tgt R+15.0%) | MM Score: 1
## EG: R+24.8% (tgt R+50.0%) | EG Score: 6
## Expected Seats: D 3.5 | R 10.5 (tgt D=2.0) | Seats Score: 2
##
## County Splits Score: 33.23
## ========================================
##
##
## ========================================
## RESULTS SAVED
## ========================================
## Final assignment: output/final_assignment_12182025_0035.csv
## Metrics: output/metrics_12182025_0035.csv
## Snapshots: output/snapshot_assignments_12182025_0035.csv
## ========================================
```

The results are encouraging. `MM DIFF` is R+5.0 (not R+15, our target, but pretty good for 3000 runs), and `EG ADJ` (efficiency gap adjusted over a few different swing scenarios) hit R+25.

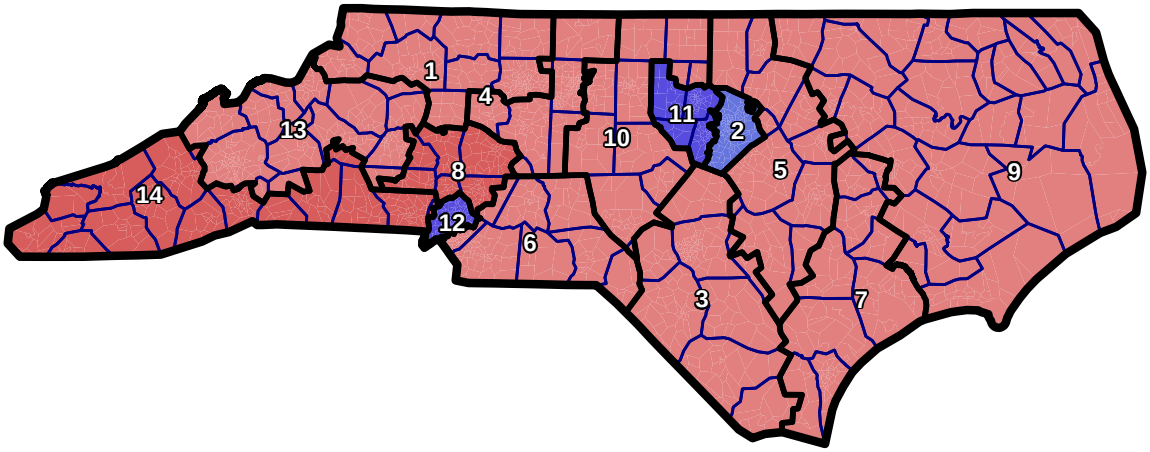
The model expects Republicans to win 10.5 seats over the long-run average, with Democrats winning only 3.5 - but the seat table makes it pretty clear that in most years this is a 11-3 map.

Once again, we graph using `mosaic_plot` - this time, however, we can set `type` to 'partisan' and look at the districts' final partisanship. This only works if we ran the annealing using partisan data- otherwise the annealing will not store this information.

```
p <- mosaic_plot(shapefile_path = SHAPEFILE,
                 type="partisan",
                 title="North Carolina: Example 3",
                 subtitle="Republican congressional gerrymander",
                 border_outline = TRUE,
                 district_outline = TRUE,
                 county_outline = TRUE,
                 number_labels = TRUE,
                 precinct_outline = FALSE)
p
```

North Carolina: Example 3

Republican congressional gerrymander



Anyone familiar with North Carolina gerrymandering history will immediately recognize this kind of map.

- The Triad is cracked; Greensboro and Winston-Salem are put into separate districts. District 1 (the Winston-to-TN district) looks like Virginia Foxx's NC-5 from 2001-2021. Greensboro itself is split perfectly in half.
- Charlotte, Raleigh, Chapel Hill, and Durham are all packed into deep blue districts.
- The Black Belt is split. Goldsboro, Kinston, and Rocky Mount all end up in three different districts.
- Two 'claws' form around the Triangle (Districts 5 and 10) instead of a district forming along the VA border.
- Fayetteville's district reaches south (towards South Carolina). Any other direction would be more purple.
- The purple-ish suburbs around Charlotte end up in three different districts.

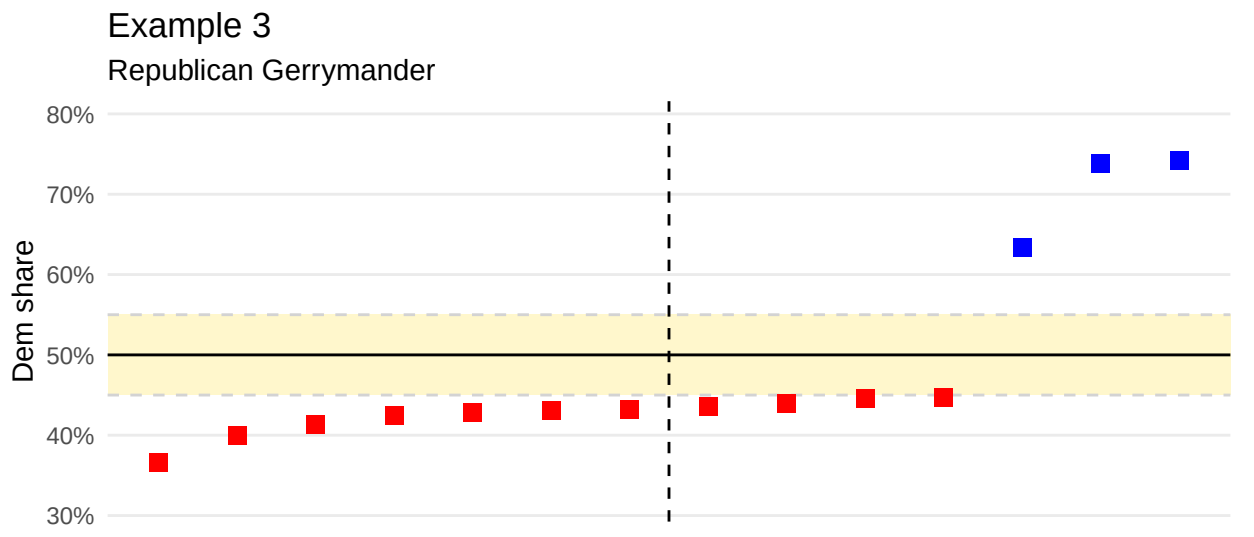
All of these choices (major and minor) contribute to this kind of partisan lean, but the algorithm figured out unknowingly what took Republican mapmakers a while to master.

Let's check out the partisan skew using `mosaic_partisan_plot`

```
p <- mosaic_partisan_plot(title="Example 3",
                          subtitle="Republican Gerrymander")
```

```
## Loading data...
## Using most recent metrics CSV: output/metrics_12182025_0035.csv
## Loading metrics for partisan plot...
## Rendering partisan graph plot...
```

```
p
```



Gerrymandering in North Carolina is a bipartisan tradition, however, so for the reverse take, check out Example 4.

Example 4: Democratic Gerrymander

For the Democratic attempt, we will basically reverse the settings from Example 3. The changes are:

- Set `target_efficiency_gap` to -0.5, meaning we aim for **Democratic** votes to be 50% more efficient than Republican ones.
- Set `target_mean_median` to -0.1, meaning we want the median district in North Carolina to be 15% **less** Republican by share of the vote than the state as a whole.
- Finally, set `target_dem_seats` to 10.

For concision, let's suppress the console output summaries with `verbose_console` = FALSE.

```
SHAPEFILE <- "shapefiles/North_Carolina_Simplified.shp"

results <- run_chain(
  shapefile_path = SHAPEFILE,
  num_districts = 14,
  seed=123456,
  county_bias=2,
  weight_county_splits=5,
  target_efficiency_gap=-0.3,
  weight_efficiency_gap=100,
  target_mean_median=-0.1,
  weight_mean_median=50,
  num_steps=3000,
  target_dem_seats=11,
  weight_dem_seats=250,
  verbose_console=FALSE

)
```

```
## Loading shapefile: shapefiles/North_Carolina_Simplified.shp
## Loading cached graph from: cache/cache_North_Carolina_Simplified.rds
## Loaded 2666 precincts
## Total population: 10,439,388
## Ideal per district: 745,671
## Population tolerance: 5.0%
##
## County data found: 100 unique counties
##
## Creating initial partition...
## GUIDED cooling: T0=2356.7 to T=1.0 at iteration 2700 (90%) at rate=0.997128
## Optimization enabled: Initial score=11783.5, Temp=2356.7
## County-aware mode: bias factor = 2.0
##
## Running 3000 ReCom steps...
## Iter: 100/3000
## Iter: 200/3000
## Iter: 300/3000
## Iter: 400/3000
## Iter: 500/3000
## Iter: 600/3000
## Iter: 700/3000
## Iter: 800/3000
## Iter: 900/3000
## Iter: 1000/3000
## Iter: 1100/3000
## Iter: 1200/3000
## Iter: 1300/3000
## Iter: 1400/3000
```

```
## Iter: 1500/3000
## Iter: 1600/3000
## Iter: 1700/3000
## Iter: 1800/3000
## Iter: 1900/3000
## Iter: 2000/3000
## Iter: 2100/3000
## Iter: 2200/3000
## Iter: 2300/3000
## Iter: 2400/3000
## Iter: 2500/3000
## Iter: 2600/3000
## Iter: 2700/3000
## Iter: 2800/3000
## Iter: 2900/3000
## Iter: 3000/3000
##
##
## Completed in 49.2s (16.4ms/step, 61.0 iter/s)
## Optimization stats: 1318 accepts, 1682 rejects (43.9% accept rate)
## Score: 11783.5 to 3454.8 (D=-8328.6)
##
## ======================================
## FINAL ANALYSIS
## ======================================
## District          DEM        REP     DEM%     REP%   Margin   P(DEM) Winner
## ------------------------------------------------------------------------------
## D1              131802     290996    31.2%    68.8%    37.7%     0.0% REP
## D2              109458     281488    28.0%    72.0%    44.0%     0.0% REP
## D3              236940     176329    57.3%    42.7%    14.7%    96.2% DEM
## D4              193716     223180    46.5%    53.5%     7.1%    17.5% REP
## D5              219294     166322    56.9%    43.1%    13.7%    95.3% DEM
## D6              236493     172669    57.8%    42.2%    15.6%    96.9% DEM
## D7              187239     267543    41.2%    58.8%    17.7%     2.0% REP
## D8              244807     155924    61.1%    38.9%    22.2%    99.2% DEM
## D9              200467     148919    57.4%    42.6%    14.8%    96.2% DEM
## D10             210725     150713    58.3%    41.7%    16.6%    97.5% DEM
## D11             127459     292161    30.4%    69.6%    39.3%     0.0% REP
## D12             145212     256104    36.2%    63.8%    27.6%     0.2% REP
## D13             234138     160019    59.4%    40.6%    18.8%    98.4% DEM
## D14             235859     154574    60.4%    39.6%    20.8%    99.0% DEM
## ------------------------------------------------------------------------------
## TOTAL          2713609    2896941    48.4%    51.6%
##
## Mean Dem Share:   48.7%
## Median Dem Share: 57.1%
## MM: D+8.4% (tgt D+10.0%) | MM Score: 0
## EG: D+10.7% (tgt D+30.0%) | EG Score: 4
## Expected Seats: D 8.0 | R 6.0 (tgt D=11.0) | Seats Score: 9
##
## County Splits Score: 33.59
## ======================================
##
##
## ======================================
## RESULTS SAVED
## ======================================
## Final assignment: output/final_assignment_12182025_0036.csv
## Metrics: output/metrics_12182025_0036.csv
```

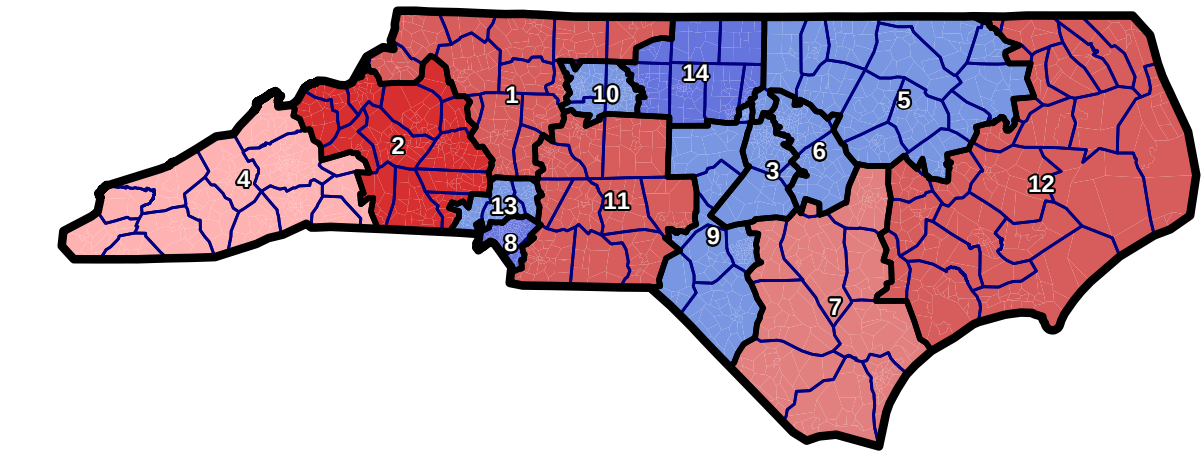
```
## Snapshots: output/snapshot_assignments_12182025_0036.csv
## ========================================
```

The results look great, but in in the opposite direction from earlier. `MM DIFF` is D+8.4, meaning the median district gave Kamala Harris 57.1% of the vote, while the average across the state was just 48.7%.

`EG ADJ` is D+10.7, and the model expects Democrats to win 8.0 seats and Republicans 6.0.

Again, we graph using `mosaic_plot`.

```
p <- mosaic_plot(shapefile_path = SHAPEFILE,
                 type="partisan",
                 title="North Carolina: Example 4",
                 subtitle="Democratic congressional gerrymander",
                 border_outline = TRUE,
                 district_outline = TRUE,
                 county_outline = TRUE,
                 number_labels = TRUE,
                 precinct_outline = FALSE)
p
```



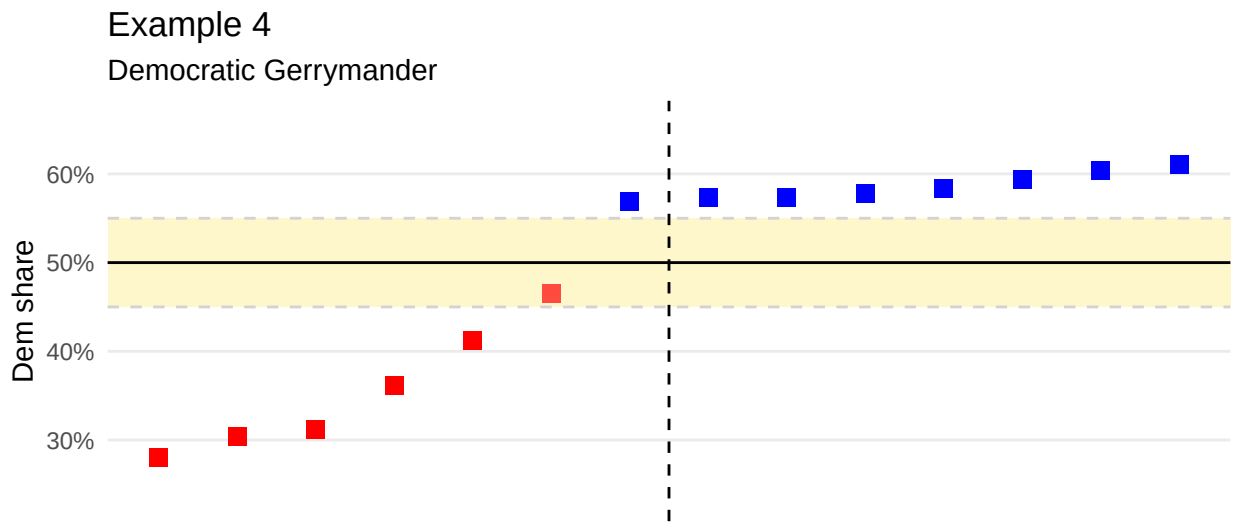
Here, many of the opposite decisions from the Republican map have been taken.

- Charlotte is cracked in two- not packed. The resulting districts (12 and 6) both vote solidly blue.
- Similarly, the Triangle is cracked- not packed- and its blue votes are distributed into 5 different districts.

- The Piedmont Triad is mostly united in District 10 (with some of Greensboro's excess blue elsewhere). Fayetteville's district aims north to purple Raleigh suburbs, not east.

On the Tennessee border, Mosaic sees a chance at a competitive district, and while it can't draw a blue one, it follows the ghost of Heath Shuler to come up with one (District 4) that Trump only won by 7 points.

```
## Loading data...
## Using most recent metrics CSV: output/metrics_12182025_0036.csv
## Loading metrics for partisan plot...
## Rendering partisan graph plot...
```



Example 5: A Fair State Senate Map

Let's deploy the partisan tools for a good cause- drawing a fair/competitive North Carolina State Senate map.

- Switch `target_mean_median` to 0 and `target_efficiency_gap` to 0. This is the same as favoring neither party.
- Introduce `weight_competitiveness` to 15. This metric tries to increase the share of districts that are competitive, where a 100% competitive district is defined as having a 50/50 chance of going to either party.
- Bolster our county tools, like `county_bias` and `weight_county_splits` - since we're after noble causes now.

```
SHAPEFILE <- "shapefiles/North_Carolina_Simplified.shp"

results <- run_chain(
  shapefile_path = SHAPEFILE,
  num_districts = 50,
  seed=123456,
  county_bias=5,
  weight_county_splits=10,
  target_mean_median=0,
  weight_mean_median=150,
  target_efficiency_gap=0,
  weight_efficiency_gap=100,
  weight_competitiveness=5,
  num_steps=5000,
  verbose_console = FALSE

)
```

```
## Loading shapefile: shapefiles/North_Carolina_Simplified.shp
## Loading cached graph from: cache/cache_North_Carolina_Simplified.rds
## Loaded 2666 precincts
## Total population: 10,439,388
## Ideal per district: 208,788
## Population tolerance: 5.0%
##
## County data found: 100 unique counties
##
## Creating initial partition...
## GUIDED cooling: T0=519.8 to T=1.0 at iteration 4500 (90%) at rate=0.998611
## Optimization enabled: Initial score=2599.0, Temp=519.8
## County-aware mode: bias factor = 5.0
##
## Running 5000 ReCom steps...
## Iter: 100/5000
## Iter: 200/5000
## Iter: 300/5000
## Iter: 400/5000
## Iter: 500/5000
## Iter: 600/5000
## Iter: 700/5000
## Iter: 800/5000
## Iter: 900/5000
## Iter: 1000/5000
## Iter: 1100/5000
## Iter: 1200/5000
```

```
## Iter: 1300/5000
## Iter: 1400/5000
## Iter: 1500/5000
## Iter: 1600/5000
## Iter: 1700/5000
## Iter: 1800/5000
## Iter: 1900/5000
## Iter: 2000/5000
## Iter: 2100/5000
## Iter: 2200/5000
## Iter: 2300/5000
## Iter: 2400/5000
## Iter: 2500/5000
## Iter: 2600/5000
## Iter: 2700/5000
## Iter: 2800/5000
## Iter: 2900/5000
## Iter: 3000/5000
## Iter: 3100/5000
## Iter: 3200/5000
## Iter: 3300/5000
## Iter: 3400/5000
## Iter: 3500/5000
## Iter: 3600/5000
## Iter: 3700/5000
## Iter: 3800/5000
## Iter: 3900/5000
## Iter: 4000/5000
## Iter: 4100/5000
## Iter: 4200/5000
## Iter: 4300/5000
## Iter: 4400/5000
## Iter: 4500/5000
## Iter: 4600/5000
## Iter: 4700/5000
## Iter: 4800/5000
## Iter: 4900/5000
## Iter: 5000/5000
##
##
## Completed in 116.3s (23.3ms/step, 43.0 iter/s)
## Optimization stats: 3522 accepts, 1478 rejects (70.4% accept rate)
## Score: 2599.0 to 1896.3 (D=-702.7)
##
## ========================================
## FINAL ANALYSIS
## ========================================
## District          DEM        REP     DEM%     REP%   Margin   P(DEM) Winner
## ------------------------------------------------------------------------------
## D1              56550      47027    54.6%    45.4%     9.2%    88.3% DEM
## D2              53863      93152    36.6%    63.4%    26.7%     0.3% REP
## D3              43195      75551    36.4%    63.6%    27.2%     0.3% REP
## D4              39169      74003    34.6%    65.4%    30.8%     0.1% REP
## D5              52431      43802    54.5%    45.5%     9.0%    87.8% DEM
## D6              53660      53195    50.2%    49.8%     0.4%    52.4% DEM
## D7              61162      60527    50.3%    49.7%     0.5%    52.9% DEM
## D8              66218      60327    52.3%    47.7%     4.7%    73.6% DEM
## D9              51658      61485    45.7%    54.3%     8.7%    12.9% REP
## D10             48476      72286    40.1%    59.9%    19.7%     1.3% REP
```

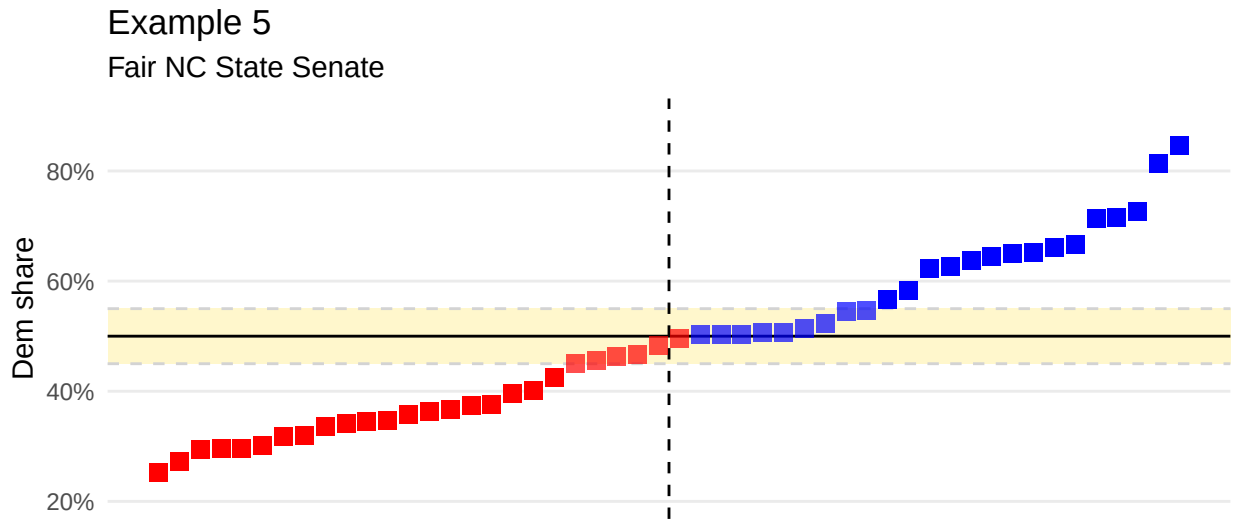
```
## D11              73798      52967    58.2%    41.8%    16.4%    97.4% DEM
## D12              68050      38586    63.8%    36.2%    27.6%    99.8% DEM
## D13              57438      56645    50.3%    49.7%     0.7%    53.8% DEM
## D14              56269      53020    51.5%    48.5%     3.0%    65.8% DEM
## D15              75511      40220    65.2%    34.8%    30.5%    99.9% DEM
## D16              32604      78236    29.4%    70.6%    41.2%     0.0% REP
## D17              75725      41641    64.5%    35.5%    29.0%    99.8% DEM
## D18              35217      83591    29.6%    70.4%    40.7%     0.0% REP
## D19              93253      16854    84.7%    15.3%    69.4%   100.0% DEM
## D20              45663      52867    46.3%    53.7%     7.3%    16.7% REP
## D21              59593      68040    46.7%    53.3%     6.6%    18.9% REP
## D22              71095      28310    71.5%    28.5%    43.0%   100.0% DEM
## D23              76768      45691    62.7%    37.3%    25.4%    99.6% DEM
## D24              44111      83913    34.5%    65.5%    31.1%     0.1% REP
## D25              62079      30942    66.7%    33.3%    33.5%    99.9% DEM
## D26              41429      69566    37.3%    62.7%    25.3%     0.4% REP
## D27              32750      78065    29.6%    70.4%    40.9%     0.0% REP
## D28              62418      24942    71.4%    28.6%    42.9%   100.0% DEM
## D29              34411      73341    31.9%    68.1%    36.1%     0.0% REP
## D30              25684      54960    31.8%    68.2%    36.3%     0.0% REP
## D31              42241      41203    50.6%    49.4%     1.2%    56.8% DEM
## D32              75689      17378    81.3%    18.7%    62.7%   100.0% DEM
## D33              72150      55207    56.7%    43.3%    13.3%    94.9% DEM
## D34              34782      92695    27.3%    72.7%    45.4%     0.0% REP
## D35              30280      89453    25.3%    74.7%    49.4%     0.0% REP
## D36              59561      60780    49.5%    50.5%     1.0%    44.5% REP
## D37              71258      36456    66.2%    33.8%    32.3%    99.9% DEM
## D38              82511      44235    65.1%    34.9%    30.2%    99.9% DEM
## D39              76100      28712    72.6%    27.4%    45.2%   100.0% DEM
## D40              40844      67933    37.5%    62.5%    24.9%     0.4% REP
## D41              39401      78008    33.6%    66.4%    32.9%     0.1% REP
## D42              65865      64316    50.6%    49.4%     1.2%    56.5% DEM
## D43              42831      52120    45.1%    54.9%     9.8%    10.4% REP
## D44              32455      75241    30.1%    69.9%    39.7%     0.0% REP
## D45              39199      70148    35.8%    64.2%    28.3%     0.2% REP
## D46              74098      44908    62.3%    37.7%    24.5%    99.5% DEM
## D47              41580      80313    34.1%    65.9%    31.8%     0.1% REP
## D48              46911      71808    39.5%    60.5%    21.0%     1.0% REP
## D49              60227      64314    48.4%    51.6%     3.3%    32.7% REP
## D50              35378      47961    42.5%    57.5%    15.1%     3.5% REP
## ------------------------------------------------------------------------------
## TOTAL          2713609    2896941    48.4%    51.6%
##
## Mean Dem Share:   48.7%
## Median Dem Share: 48.9%
## MM: D+0.2% (tgt R+0.0%) | MM Score: 0
## EG: R+2.1% (tgt R+0.0%) | EG Score: 0
## Expected Seats: D 22.2 | R 27.8
##
## County Splits Score: 27.48
## ========================================
##
##
## ========================================
## RESULTS SAVED
## ========================================
## Final assignment: output/final_assignment_12182025_0038.csv
## Metrics: output/metrics_12182025_0038.csv
## Snapshots: output/snapshot_assignments_12182025_0038.csv
```

```
## ======================================
```

The results are compelling.

- The mean Democratic vote share (using Kamala Harris' 2024 result) is 48.7%, and the median is 48.9%. Trump wins a majority of districts, but if Harris had won, she likely would have too.
- The efficiency gap is R+2.1%, well within our tolerance.
- 9.3 competitive seats are forecast- almost 20% of the total map- creating a wide battleground.

```
## Loading data...
## Using most recent metrics CSV: output/metrics_12182025_0038.csv
## Loading metrics for partisan plot...
## Rendering partisan graph plot...
```

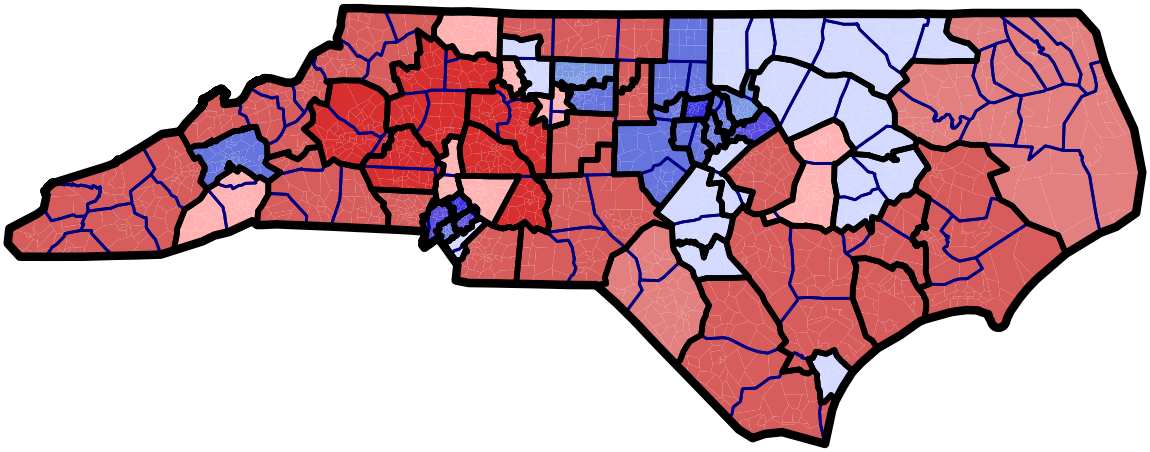


... and visualize! Note- in reality, this map would not pass the extremely strict splitting laws that North Carolina has in place for the NCGA. But it's fun to imagine.

```
p <- mosaic_plot(shapefile_path = SHAPEFILE,
                 type="partisan",
                 title="North Carolina: Example 5",
                 subtitle="Balanced and competitive NC Senate map",
                 border_outline = TRUE,
                 district_outline = TRUE,
                 county_outline = TRUE,
                 number_labels = FALSE,
                 precinct_outline = FALSE)
p
```

North Carolina: Example 5

Balanced and competitive NC Senate map



Example 6: Dealing with Incumbents

Let's imagine that tomorrow a court orders that North Carolina draw a new, fair congressional map. *However*, the court orders that the remedial map try and keep incumbents apart.

Mosaic can help here. The program can handle both pro-bunking and anti-bunking lists. Pro-bunking lists aim to keep precincts together (like communities of interest). Anti-bunking lists aim to keep precincts apart (like incumbent homes).

We establish the incumbent list here. These are the precinct IDs (GEOIDs) for each North Carolina congressperson (roughly - we don't really know where they live). The first two values are the weight and the exponent (here, we go with 1000 and 1 respectively).

```
NC_incumbent_list <- c(
  1000,              # Weight: heavy penalty for splitting
  1,                 # Exponent: linear penalty
  "3707900BEAR", #Davis
  "37183001-27", #Ross
  "3714701504A", #Murphy
  "3713500000H", #Foushee
  "37011000002", #Foxx
  "37059000013", #McDowell
  "37129000W25", #Rouzer
  "37179000014", #Harris
  "37125000SSP", #Hudson
  "37035000013", #Harrigan
  "370890000FR", #Edwards
  "37119000011", #Adams
  "37183001-36", #Knott
  "3704500KM-N"  #Moore
)
```

We pass the bunking list by adding a call to `create_bunking_lists()` to `run_chain()`. Multiple sub-lists can be added to either `anti_bunking` or `pro_bunking`.

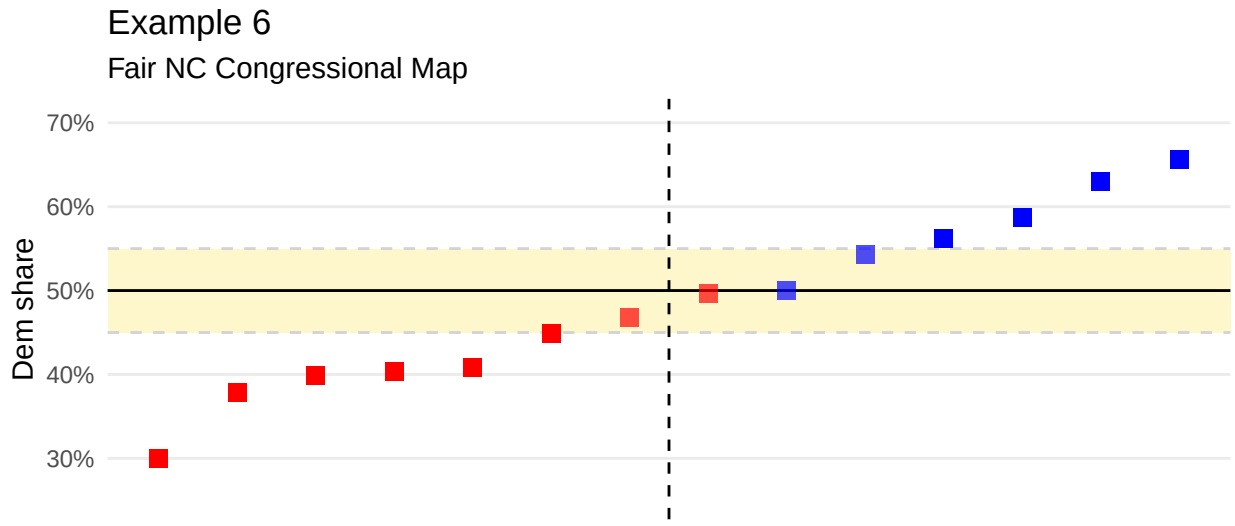
```
## Loading shapefile: shapefiles/North_Carolina_Simplified.shp
## Loading cached graph from: cache/cache_North_Carolina_Simplified.rds
## Loaded 2666 precincts
## Total population: 10,439,388
## Ideal per district: 745,671
## Population tolerance: 5.0%
##
## County data found: 100 unique counties
##
## Creating initial partition...
## Anti-Bunking List 1: weight=1000.0, exponent=1.0, 14 precincts
## GUIDED cooling: T0=409.4 to T=1.0 at iteration 2250 (90%) at rate=0.997330
## Optimization enabled: Initial score=2047.0, Temp=409.4
## County-aware mode: bias factor = 3.0
##
## Running 2500 ReCom steps...
## Iter: 100/2500
## Iter: 200/2500
## Iter: 300/2500
## Iter: 400/2500
```

```
## Iter: 500/2500
## Iter: 600/2500
## Iter: 700/2500
## Iter: 800/2500
## Iter: 900/2500
## Iter: 1000/2500
## Iter: 1100/2500
## Iter: 1200/2500
## Iter: 1300/2500
## Iter: 1400/2500
## Iter: 1500/2500
## Iter: 1600/2500
## Iter: 1700/2500
## Iter: 1800/2500
## Iter: 1900/2500
## Iter: 2000/2500
## Iter: 2100/2500
## Iter: 2200/2500
## Iter: 2300/2500
## Iter: 2400/2500
## Iter: 2500/2500
##
##
## Completed in 46.5s (18.6ms/step, 53.7 iter/s)
## Optimization stats: 1104 accepts, 1396 rejects (44.2% accept rate)
## Score: 2047.0 to 828.9 (D=-1218.2)
##
## ======================================
## FINAL ANALYSIS
## ======================================
## District          DEM        REP     DEM%     REP%   Margin Winner
## ------------------------------------------------------------------------
## D1             149158     169646    46.8%    53.2%     6.4% REP
## D2             248386     130408    65.6%    34.4%    31.1% DEM
## D3             203585     206033    49.7%    50.3%     0.6% REP
## D4             175137     264429    39.8%    60.2%    20.3% REP
## D5             255717     179687    58.7%    41.3%    17.5% DEM
## D6             147139     241142    37.9%    62.1%    24.2% REP
## D7             260045     152935    63.0%    37.0%    25.9% DEM
## D8             222738     173636    56.2%    43.8%    12.4% DEM
## D9             123609     288234    30.0%    70.0%    40.0% REP
## D10            200942     246447    44.9%    55.1%    10.2% REP
## D11            202911     202496    50.1%    49.9%     0.1% DEM
## D12            162791     240380    40.4%    59.6%    19.2% REP
## D13            200248     168350    54.3%    45.7%     8.7% DEM
## D14            161203     233118    40.9%    59.1%    18.2% REP
## ------------------------------------------------------------------------
## TOTAL         2713609    2896941    48.4%    51.6%
##
## Mean Dem Share:   48.4%
## Median Dem Share: 48.2%
## MM: R+0.2% (tgt R+0.0%) | MM Score: 0
## EG: R+3.0% (tgt R+0.0%) | EG Score: 0
##
## County Splits Score: 9.40
##
## ------------------------------------------------------------------------
## Anti-Bunking Metrics:
##   List 1: 14 precincts in 13 districts | metric=0.148 | score=148.4 (w=1000 e=1.0)
```

```
## ========================================
##
##
## ========================================
## RESULTS SAVED
## ========================================
## Final assignment: output/final_assignment_12182025_0039.csv
## Metrics: output/metrics_12182025_0039.csv
## Snapshots: output/snapshot_assignments_12182025_0039.csv
## ========================================
```

Partisanship looks good. No major mean/median skew, and a healthy number of competitive districts.

```
## Loading data...
## Using most recent metrics CSV: output/metrics_12182025_0039.csv
## Loading metrics for partisan plot...
## Rendering partisan graph plot...
```



We can also pass bunking lists to `mosaic_plot()` - which makes it easy to visually check out the results.

```
p <- mosaic_plot("shapefiles/North_Carolina_Simplified.shp",
                 bunking_lists = create_bunking_lists(
                   anti_bunking  = list(NC_incumbent_list)
                 ),
                 title = "Example 6: Fair NC Congressional Map Without Double-Bunking",
                 subtitle = "Yellow diamonds: incumbent locations",
                 district_outline = TRUE,
                 number_labels = TRUE,
                 county_outline = TRUE,
                 border_outline = TRUE)
p
```

Example 6: Fair NC Congressional Map Without Double−Bunking

Yellow diamonds: incumbent locations