CS249 — Basic Data Science — Final Exam
Due: 11:55 pm Sunday June 12, 2016
D. Stott Parker, Thuy Vu

- IMPORTANT: There are four equally-weighted questions.
- IMPORTANT: In the .zip file for the final is a directory called `my_answers/`. For grading, please put all files you produce (writing, programs, and output) into the appropriate subdirectory. Then zip it and upload `my_answers.zip` to CCLE.
- IMPORTANT: Post questions to Piazza if you have questions. Note: questions do not have ot be Public, you can post Private Questions to us. Do not talk with anyone about this exam. DO YOUR OWN WORK.

1. Grant Applications

The [APM] textbook used in this course studies a Grant Application dataset, particularly in chapters 12 and 14. This dataset is available at Kaggle (`kaggle.com`) in their `Predict Grant Applications` contest.

For this problem, we ask you to:

- get the grant application datasets, available from Kaggle: `unimelb_train.csv` and `unimelb_test.csv`.
- develop a script or notebook that yields predictions for the test set.
  NOTE: the classification for grant applications is named `Grant.Status`, in the second column; 1 means successful, 0 unsuccessful.
- submit your predictions to Kaggle for the test set. (This requires you to sign up as a Kaggle user.)
- obtain the best score you can from Kaggle (with your predictions).
- put the following files in your `my_answers/grants/` directory:
  - the script or notebook you generated (`grants/script....`)
  - your file of predictions for the test set (`grants/predictions.csv`)
  - a summary text file (`grants/README.txt`) giving your Kaggle id, Kaggle score, and Kaggle rank

The [APM] book offers many insights about this dataset, but instead of using the raw data, it uses digests that are produced by an R script called `CreateGrantData.R` (in the `AppliedPredictiveModeling package`). If you use the methods in the book, you can benefit from running this script and executing `save.image(file="grantData.RData")`. This stores all computed results in a saved state that can be quickly reloaded with `load("grantData.RData")`. For example, this reloading command appears in four scripts for chapters in the book: `12_Discriminant_Analysis.R`, `13_Non-Linear_Class.R`, `14_Class_Trees.R`, `18_Importance.R`.

2. Attractiveness

The file `attractiveness.tsv` consists of data accumulated from a (now defunct) site called `facestat.com`. At this site, people uploaded photos with data about themselves, and also made snarky comments about photos of other people. (The idea was reminiscent of the 'Hot-or-Not' program of Mark Z. that was immortalized in the movie *The Social Network*.) The dataset is a distillation of information provided by users, and has many missing values. This dataset has 6 columns, titled `age`, `weight`, `attractive`, `intelligence`, `trustworthy`, and `male`; the final column is a boolean value that is TRUE when the person is male.

The data is interesting; for example Figure 1 shows how attractiveness varies with age. This set of plots suggests that men and women differ on 'attractiveness'.

In this problem we want to discriminate between men and women with the data provided.
Given values for age, weight, attractiveness, etc., can we predict gender?

(a) Using the training data `attractiveness_train.csv`, develop a model predicting gender.
(There are many missing values, so devise a strategy for dealing with them.)

(b) Using the file `attractiveness_test.csv` as input, generate predictions `attractiveness_predictions.csv` of gender values. (The value should be TRUE if the gender is male, and FALSE otherwise.)

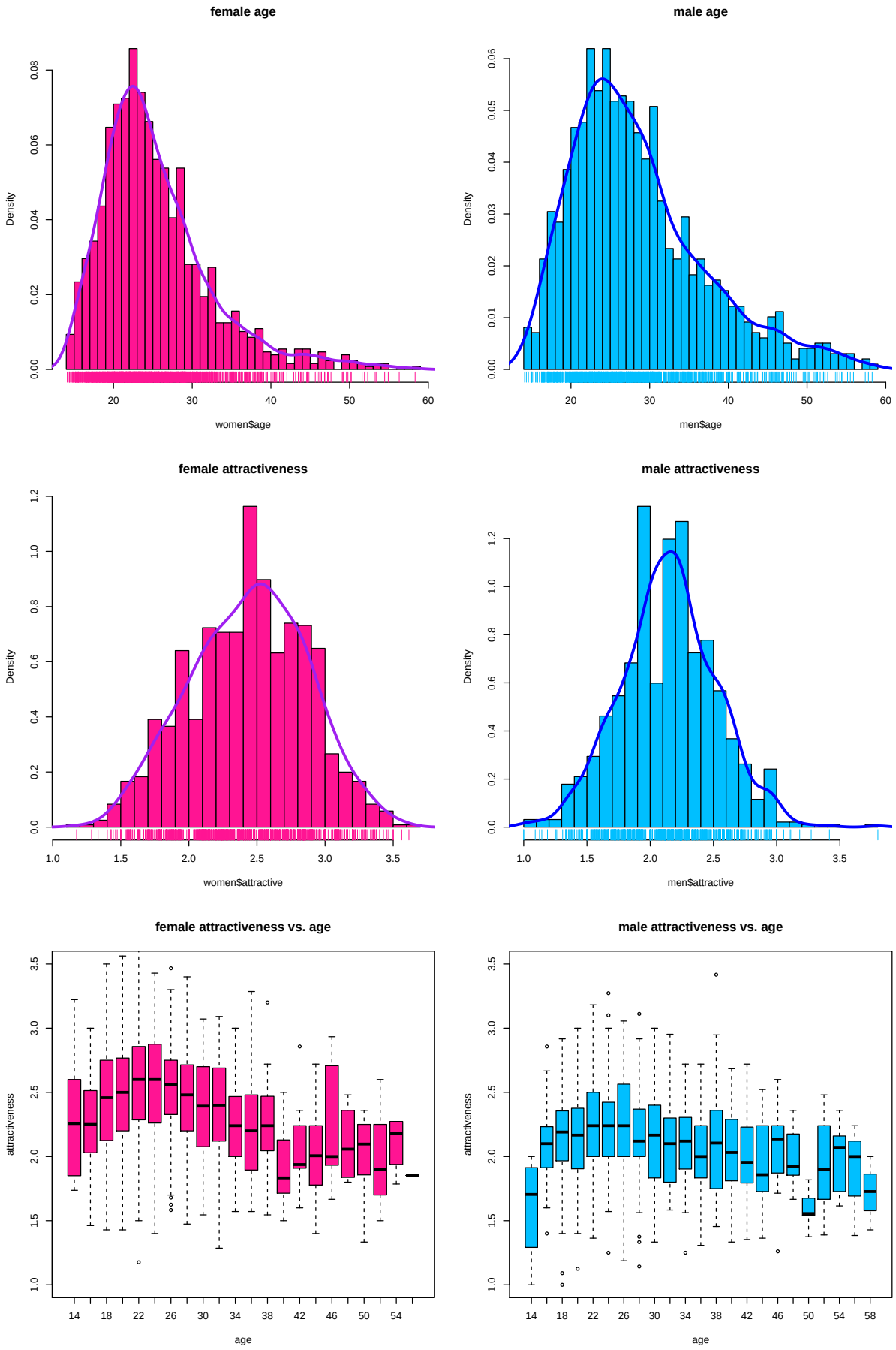

Figure 1: Age vs. Attractiveness in the facestat data

(c) Implement **L_2-regularized Logistic Regression** using a popular method known as IRLS (Iteratively-Reweighted Least Squares). IRLS is described in Murphy's text (Machine Learning), in Algorithm 8.2 in section 8.3.4; as an example, you are provided a very similar simple implementation in R, called `IRLS_logistic_regression.R`. Develop an implementation that supports L_2 regularization, which is discussed in section 8.3.6. (You can implement this in any language you prefer.)

Note: Murphy points out that L_2-regularized IRLS is very much like Ridge Regression (L_2-regularized Least Squares), described in section 7.5.1.

(d) The regularized Logistic Regression method you just developed cannot handle missing values. The attractiveness dataset has *many* missing values. Either find a way to use your method to derive the best cross-validated gender classification accuracy you can get for this dataset, or explain why the method cannot be meaningfully used on the dataset.

3. Unicorns

In class we studied information about data science startups, using Crunchbase data about tech companies and investors. In this problem we study *unicorns* — young companies with a valuation exceeding $1B. A list of about 160 unicorns is available at `fortune.com/unicorns` and a similar database is included as the file `unicorns.tsv`.

A list of 50 'future unicorns' — `predictions by CBinsights.com of companies that could become unicorns` — is included as the file `future_unicorns.tsv`.

We also are using `a December 2015 export of the Crunchbase Data`, including two files `companies.tsv` and `investments.tsv`. These files include information on unicorns and 66,000 other companies; the information was exported in December 2015 and is almost current.

> In this problem we are going to construct 'social networks' linking investors and companies (and, in particular, unicorns). We want to determine whether the social network has two properities: (1) a 'rich-get-richer' structure, in which the node degree distribution follows a power law; (2) a 'small-world' structure, in which the graph can be large and locally-well-connected, but the diameter is small (distances between companies are never large). (Good references are the review by Chakrabarti and Faloutsos, and Chapters 18 and 20 from the Easley-Kleinberg book.)

Figure 2 shows a subset of the unicorn–investor network, illustrating how an interactive graph of the data looks. (You can use any network analysis package. For Python, use of `networkx` is common, but `igraph` is also popular.)

(NOTE: in lecture I went over Jupyter notebooks showing related use of `networkx` — one reading Crunchbase, and one doing graph mining on Data Science Startups. These are included in the `additional_material/` subdirectory.)

You are asked to construct three undirected, unweighted graphs in which the nodes represent tech companies — and two companies are connected if they have an investor in common.

- Construct a company–company graph in which the companies are *unicorns*. The `unicorns` and `investors_in_unicorns` data provide the data you need; there are about 160 unicorns, so this graph is relatively small.
- Construct a company–company graph in which the companies are *unicorns or future unicorns*. The datasets `unicorn_table`, `investors_in_unicorns`, `future_unicorn_table`, and `investors_in_future_unicorns` have the necessary information. There are about 200 such companies, so again this graph is small.
- Construct a company–company graph using all of the Crunchbase data. The `companies` and `investments` tables provide the data you need; there are over 66,000 companies, so this graph is much larger.

Then, for each of these three graphs:

(a) determine the number of connected components.

(b) compute the degree distribution (sequence of node degree values), and print the nodes with the top 10 values. Determine if the distribution follows a power law. Chapter 18 from the Easley-Kleinberg book is a good reference.

(c) compute eigenvector centrality of the nodes, and print the nodes with the top 10 values.

(d) determine the diameter of the largest component (the greatest distance between two companies).

(e) compute the clustering coefficient $C^{\triangle}$ (= (3 times the number of triangles) / (the number of paths of length 2)), and use it to determine whether the network is a small-world network. (Use this coefficient, although there are other clustering coefficients that people use to define a 'small world' network — see e.g. the review by Chakrabarti and Faloutsos, and `[nextworkx]`.)

Finally, for the second graph (`unicorn` + `future_unicorn` network), determine whether there is any future unicorn that is disconnected from all current unicorns.

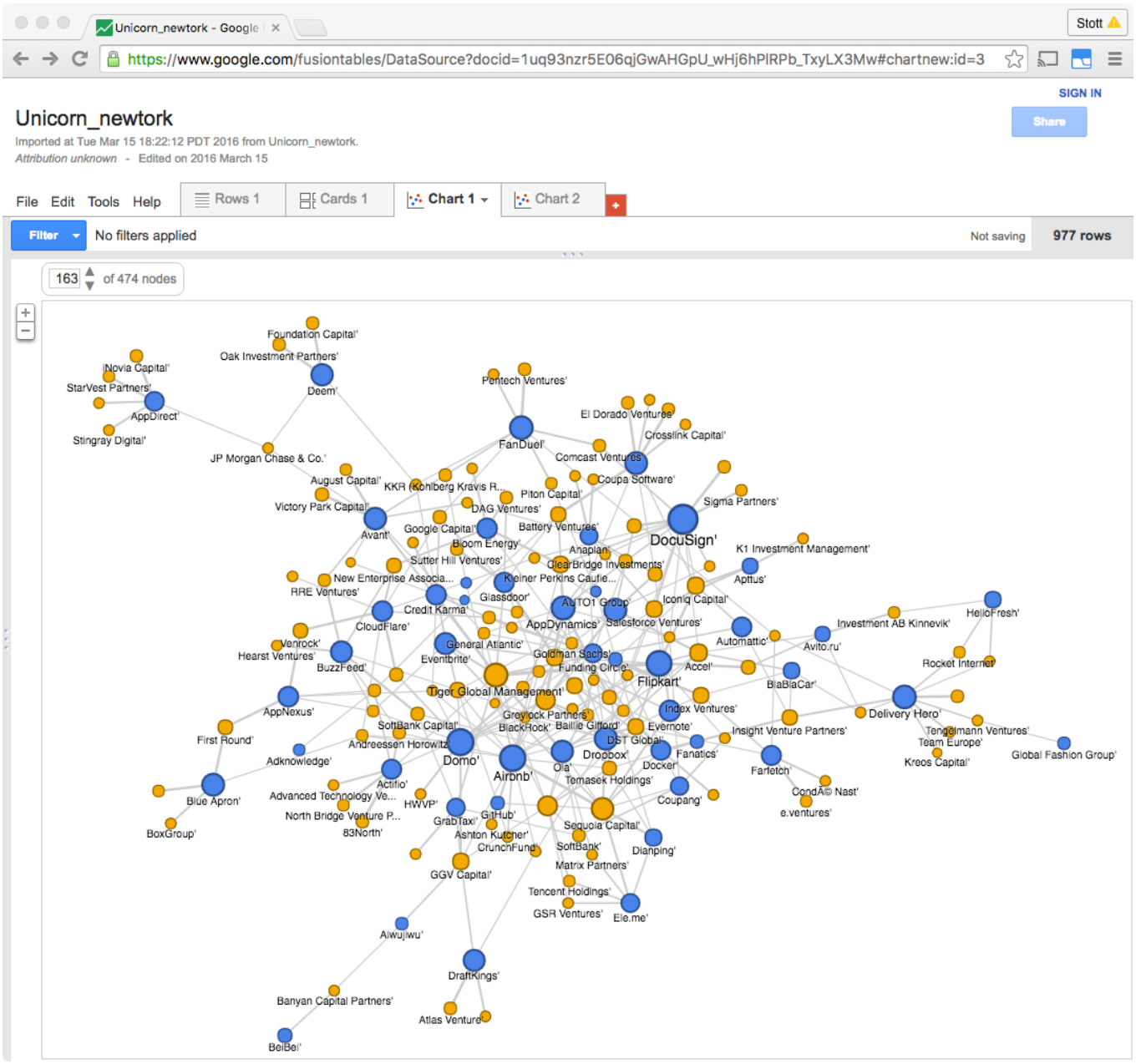

Figure 2: Image of an interactive Google Fusion Table of a network linking Unicorns and investors — produced by Ruth Johnson, an undergraduate CS major, for CS170A in Winter 2016.

5 points Extra credit: *CB Insights* claims to have a machine learning algorithm for identifying unicorns. Based on their description of the 'Cruncher' technology behind it, try to identify specific methods they use to predict future unicorns. In case it matters, several on their list became Unicorns recently: Avant, Decolar.com, Ele.me, Gusto, HelloFresh, Okta, and Thumbtack.

4. Polarization of Voting in the U.S. Senate

Roll call vote is a standard method of voting in the U.S. Senate. The history of roll call votes is maintained at voteview.com. A recent analysis of this data (by Drew Conway, attached) uses MDS (Multi-Dimensional Scaling) to display it as in Figure 3. This visualization suggests how *polarized* voting in the Senate has become, and offers some perspective on changes in 'bipartisanship' in United States government.

MDS a technique for embedding data points of high dimension (in this case, the large number of votes in each Congress) in a much lower-dimensional space (in this case, 2D plots) — in a way that preserves distances as closely as possible. Using MDS, the roll call votes for the Senate in the 102nd through 113th Congresses (each Congress covers a two-year period) show some change over time, shown in in the figure.

How can we measure polarization? This is the question we want you to answer here.

Collaboration between Republicans and Democrats has declined, to the point that voting patterns in the two parties appear largely decoupled. Republican senators often vote as a block (i.e., approximately all vote the same way on each roll call), independent of what Democrat senators do. Intuitively, this is what is meant by polarization.

Do the following:

(a) produce a *biclustering* visualization of the voting history of each Congress (102nd through 113th).

(b) define (in English) a measure of *polarization* by staring at this visualization (or other analysis you do) — a single numeric value — and explain why it is a good measure of polarization.

For this assignment, polarization can be represented by any mathematical model that measures clustering or similarity of intra-party votes, and dissimilarity of inter-party votes.

(c) calculate your measure of polarization for each Congress.

(d) plot the timeline showing the value of your polarization measure for each Congress.

To produce a bicustering display, for example, in R you might modify the script `chapter09.R` by Conway to use a biclustering or heatmap package. In scikit-learn, the function `SpectralCoclustering()` can be used with a method like scikit-learn.org/stable/_downloads/plot_spectal_coclustering.py to obtain a display of the biclustered data — with a final `plt.savefig()`.

'Is there a single right answer for the measure of polarization?' Obviously not. The question only asks you to define polarization mathematically. As an example, the minimum distance between all pairs of voting records involving senators of different parties is an almost-reasonable measure. Generate a good measure of polarization.

In summary, complete the following:

- produce a biclustering display, and upload the resulting biclustering image file to CCLE.
- develop a script or notebook with your definition of *polarization*.
- put the following files in your `my_answers/voting/` directory:
  - biclustering script or notebook (`voting/biclustering.*`), including PDF output
  - polarization script or notebook (`voting/polarization.*`), including PDF output
  - a summary text file (`voting/README.txt`).

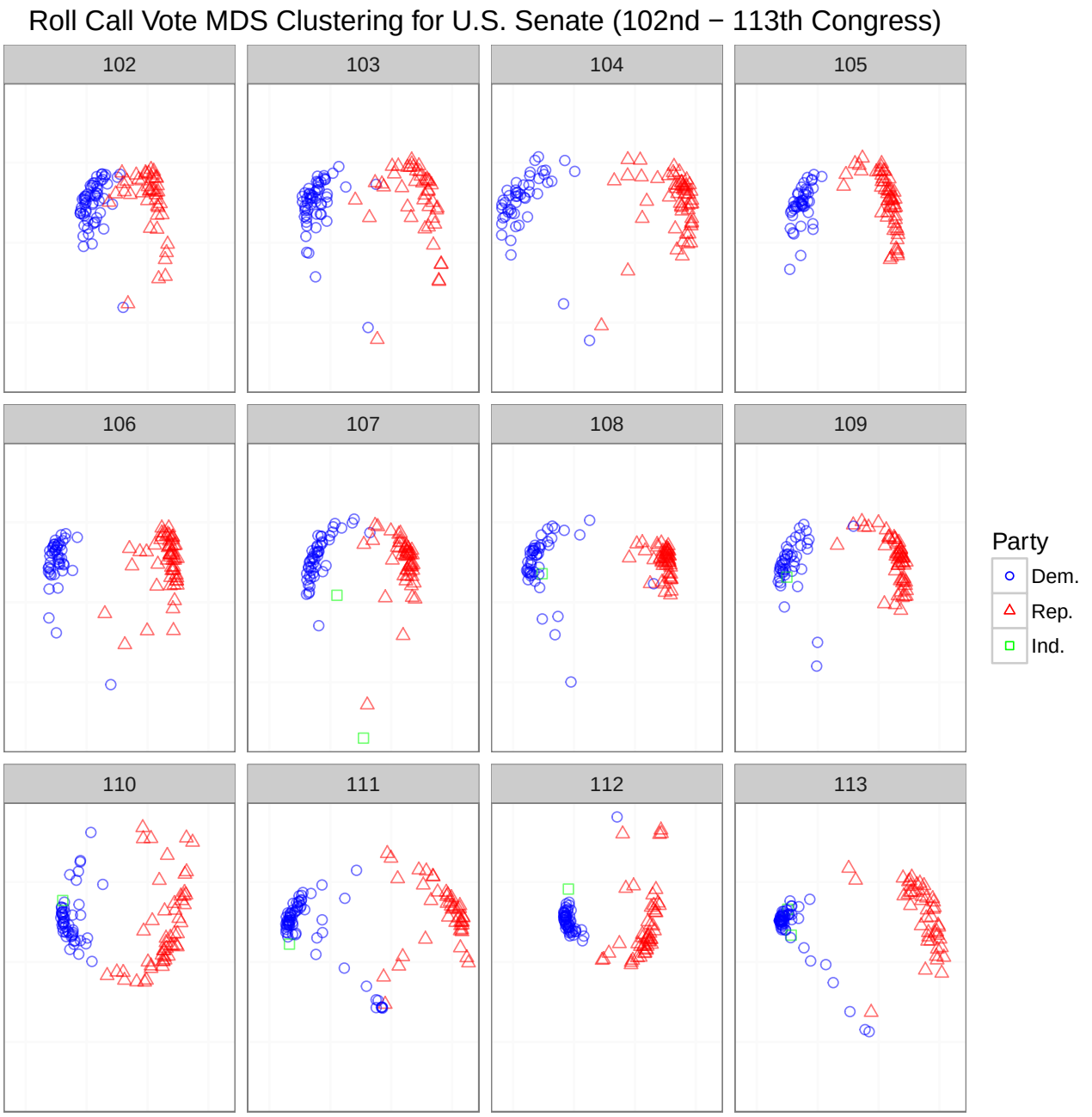

Figure 3: Change in voting patterns in the U.S. Senate over time, from the 102nd to 113th Congress — a time period spanning from January 1991 through December 2015. The clustering of voters by party and the changes in separation between parties suggest 'polarization'.