MTGJSON.Rmd

Ted Tinker and Lilian Lu

November 21, 2017

Abstract

Magic: The Gathering is a collectible-card-game set in a fantasy multiverse. Since its introduction in 1993 more than 17,000 MTG cards have been designed and published in more than seventy sets. Information on these cards is available in a JSON file from mtgjson.com, but this format is inconvenient in Rstudio. In this project we parce the JSON file into a dataframe so we may use Rstudio to answer the following questions:

How does a card's color-identity (either colorless or any combination of the colors white(W), blue(U), black(B), red(R), and green(G)) influence the rest of its statistics? Based on that, can a card's statistics be used to algorithmically predict its color-identity? How has the relation between color-identity and statistics changed over Magic's lifetime? And, how do these questions relate to the design philosophy of Magic's creators? (Data mining techniques used, key results, conclusions.)

Introduction

Our primary goal is to consistently algorithmically sort Magic: The Gathering cards by color-identity based on the rest of their statistics. This will be difficult because Magic: The Gathering cards come in a variety of types and levels of complexity, and color-identity is the product of subjective game-design considerations by the designers. Expert humans are able to assign color-identities to cards with ease; Mark Rosewater, Magic: The Gathering's head designer, arbitrates the color-philosophies with intricate precision. In "Mechanical Color Pie 2017" he associates every keyword in the game's standard lexicon with each color at one of four different levels: primary, secondary, tertiary, or none of the above. Given the detailed knowledge required to accurately enforce the color-philosophies, an algorithm for assigning color-identity based on card-data may be useful for designing the game, or understanding the game's design.

This project is inspired by the RoboRosewater twitter account. This twitter account posts the output of a neural network trained on Magic: The Gathering cards. Some of RoboRosewater's cards are humorously unreadable; others are syntactically correct and theoretically playable. However, even RoboRosewater's best cards have mismatched color-identities as the program cannot grasp the subtle intricacies of "flavor," or relation to in-game lore and philosophies. This project endeavors to improve color-identity identification.

The dataset of all Magic: The Gathering cards was downloaded from mtgjson.com in the form of a JSON file. JSON is a natural choice for Magic: The Gathering cards because the cards have inconsistant variables present based on their card-type. Some variables have high consistency (for example, each card has a name), but some variables may not be present on all card types (for example, only creature cards have power and toughness), and some variables are lists whose lengths vary from card to card (for example, the list of all sets in which the card was printed). The data was most likely scraped from Gatherer, Magic's offical online card index. Therefore the data is most likely accurate, up-to-date, and complete, but in a format which Rstudio cannot easily handle.

(Clear description of the problem and techniques we will use. Briefly describe results, good and bad, and conclusions.)

Data and Methods

```
# install.packages("tidyverse")       # Installation only necessary on the first run
# install.packages("rjson")
# install.packages("gtools")
# install.packages("stringr")
# install.packages("ggplot2")

library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------------------------------- tidyv
## v ggplot2 2.2.1     v purrr   0.2.4
## v tibble  1.3.4     v dplyr   0.7.4
## v tidyr   0.7.2     v stringr 1.2.0
## v readr   1.1.1     v forcats 0.2.0
## -- Conflicts -------------------------------------------------------------------------- tidyverse_c
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(rjson)      # To read JSON
library(gtools)
library(stringr)    # To interact with strings
library(ggplot2)    # Nice plots

setwd("/Users/Theodore/Desktop/R_Studio_Stuff/data")  # Change this to your own working directory
MTG.json <- fromJSON(file="AllCards.json",method='C')
    # JSON file from https://mtgjson.com (not great for R)

rows=17761    # When testing, decrease this so you don't use the whole data-set every time

MTG.df <- data.frame(matrix(ncol=21,nrow=rows))   # Start with 21 variables; others are generated.
```

Here is an example of a typical card in JSON format:

```
MTG.json[["Serra Angel"]]
```

```
## $layout
## [1] "normal"
##
## $name
## [1] "Serra Angel"
##
## $manaCost
## [1] "{3}{W}{W}"
##
## $cmc
## [1] 5
##
## $colors
## [1] "White"
##
## $type
## [1] "Creature <U+0097> Angel"
##
```

```
## $types
## [1] "Creature"
##
## $subtypes
## [1] "Angel"
##
## $text
## [1] "Flying (This creature can't be blocked except by creatures with flying or reach.)\nVigilance (A
##
## $power
## [1] "4"
##
## $toughness
## [1] "4"
##
## $imageName
## [1] "serra angel"
##
## $printings
##  [1] "LEA"     "LEB"     "2ED"     "CED"     "CEI"     "3ED"     "4ED"
##  [8] "ATH"     "pWOS"    "7ED"     "8ED"     "9ED"     "10E"     "DDC"
## [15] "M10"     "M11"     "ME4"     "CMD"     "M12"     "M13"     "M14"
## [22] "M15"     "DD3_DVD" "ORI"     "V15"     "W16"     "EMA"     "W17"
## [29] "CMA"     "IMA"
##
## $legalities
## $legalities[[1]]
## $legalities[[1]]$format
## [1] "Commander"
##
## $legalities[[1]]$legality
## [1] "Legal"
##
##
## $legalities[[2]]
## $legalities[[2]]$format
## [1] "Legacy"
##
## $legalities[[2]]$legality
## [1] "Legal"
##
##
## $legalities[[3]]
## $legalities[[3]]$format
## [1] "Modern"
##
## $legalities[[3]]$legality
## [1] "Legal"
##
##
## $legalities[[4]]
## $legalities[[4]]$format
## [1] "Standard"
##
```

```
## $legalities[[4]]$legality
## [1] "Legal"
##
##
## $legalities[[5]]
## $legalities[[5]]$format
## [1] "Vintage"
##
## $legalities[[5]]$legality
## [1] "Legal"
##
##
##
## $colorIdentity
## [1] "W"
```

Clearly this must be parsed to yield useful information in Rstudio. The following chunk should probably be run in a separate script for stability reasons.

```
# The function "isItThere" is vital to parsing JSON.
# Nested lists and inconsistant variable types require it
# for producing consistant dataframe row-lengths.
# The argument "json" is a JSON object.
# "x" is the position in the JSON list to open;
# each will be a Magic: The Gathering card.
# "string" is the name of the variable to check for in the JSON element x.
# "otherwise" is what the program should return
# if the variable doesn't exist; sometimes "", sometimes NA.
# "after" is the nested list index in case we need to check
# elements of a nested list which may not exist.
isItThere <- function(json,x,string,otherwise,after="") {
  if(after!="") {if(is.null(json[[x]][[string]][after])) {return(otherwise)} else
  {return(json[[x]][[string]][after])}} else
    if(is.null(json[[x]][[string]])) {return(otherwise)} else {return(json[[x]][[string]])}}

# Color-identity is expressed as up to five of the characters WUBR and G. Sort them to make factors.
sapply(1:rows, function(x) {colorIdentity <- sort(c(isItThere(MTG.json,x,"colorIdentity",NA,1),
                                                    isItThere(MTG.json,x,"colorIdentity",NA,2),
                                                    isItThere(MTG.json,x,"colorIdentity",NA,3),
                                                    isItThere(MTG.json,x,"colorIdentity",NA,4),
                                                    isItThere(MTG.json,x,"colorIdentity",NA,5)))
colorIdentity <- colorIdentity[!is.na(colorIdentity)]

# Supertype, Type, and Subtype are collections of zero to three character strings
supertype <- c(isItThere(MTG.json,x,"supertypes",NA,1),
               isItThere(MTG.json,x,"supertypes",NA,2))
supertype <-supertype[!is.na(supertype)]

type <- c(isItThere(MTG.json,x,"types",NA,1),
          isItThere(MTG.json,x,"types",NA,2))
type <-type[!is.na(type)]

subtype <- c(isItThere(MTG.json,x,"subtypes",NA,1),
             isItThere(MTG.json,x,"subtypes",NA,2),
             isItThere(MTG.json,x,"subtypes",NA,3))
```

```
subtype <-subtype[!is.na(subtype)]

# The list of printings is a collection of character-string abbreviations of set-names.
# We will decompose it into printing dates later.
printList <- c(unlist(lapply(1:length(MTG.json[[x]][["printings"]]),function (y)
                             {MTG.json[[x]][["printings"]][[y]]})))

MTG.df[x,] <<- c(MTG.json[[x]][["name"]],     # We describe all variables as we name them below
                 isItThere(MTG.json,x,"manaCost",0),
                 paste(colorIdentity,collapse=""),
                 MTG.json[[x]][["cmc"]],
                 str_count(isItThere(MTG.json,x,"manaCost",""),"W"),
                 str_count(isItThere(MTG.json,x,"manaCost",""),"U"),
                 str_count(isItThere(MTG.json,x,"manaCost",""),"B"),
                 str_count(isItThere(MTG.json,x,"manaCost",""),"R"),
                 str_count(isItThere(MTG.json,x,"manaCost",""),"G"),
                 paste(supertype,collapse=" "),
                 paste(type,collapse=" "),
                 paste(subtype,collapse=" "),
                 isItThere(MTG.json,x,"text",NA),
                 nchar(isItThere(MTG.json,x,"text","")),
                 isItThere(MTG.json,x,"power",NA),
                 isItThere(MTG.json,x,"toughness",NA),
                 isItThere(MTG.json,x,"loyalty",NA),
                 length(MTG.json[[x]][["rulings"]]),
                 length(MTG.json[[x]][["legalities"]]),
                 length(MTG.json[[x]][["printings"]]),
                 paste(printList, collapse=" "))})
```

Now we add column names and types:

```
MTG.df <- setNames(MTG.df,c("Name",                 # Character; MTG card name
                            "ManaCost",             # Character; Mana Cost in typical online format
                            "ColorIdentity",        # Factor; Color identity
                            "ConvertedManaCost",    # Integer; number of mana required to cast card
                            "White",                # Integer; number of white mana required
                            "Blue",                 # Integer; number of blue mana required
                            "Black",                # Integer; number of black mana required
                            "Red",                  # Integer; number of red mana required
                            "Green",                # Integer; number of green mana required
                            "Supertype",            # Character; large categories of cards
                            "Type",                 # Character; main card types
                            "Subtype",              # Character; supplimentary types
                            "Text",                 # Character; Rules text, not flavor text
                            "TextLength",           # Integer; number of characters in Text
                            "Power",                # Often an integer, sometimes a character
                            "Toughness",            # Often an integer, sometimes a character
                            "Loyalty",              # Integer; for Planeswalkers
                            "Rulings",              # Number of rulings
                            "Legalities",           # Number of legalities
                            "Printings",            # Number of printings
                            "PrintingsList"))       # Character; all set abbrvs, to make year-lists

MTG.df <- transmute(MTG.df, Name=as.character(Name),
```

```
                    ManaCost=as.character(ManaCost),
                    ColorIdentity=as.factor(ColorIdentity),
                    ConvertedManaCost=as.integer(ConvertedManaCost),
                    White=as.integer(White),
                    Blue=as.integer(Blue),
                    Black=as.integer(Black),
                    Red=as.integer(Red),
                    Green=as.integer(Green),
                    Supertype=as.character(Supertype),
                    Type=as.character(Type),
                    Subtype=as.character(Subtype),
                    Text=as.character(Text),
                    TextLength=as.integer(TextLength),
                    Power=as.character(Power),          # Mostly integers, some odd characters
                    Toughness=as.character(Toughness), # Mostly integers, some odd characters
                    Loyalty=as.integer(Loyalty),
                    Rulings=as.integer(Rulings),
                    Legalities=as.integer(Legalities),
                    Printings=as.integer(Printings),
                    PrintingsList=as.character(PrintingsList))
```

The variable PrintingsList is not a convenient way to check when cards were published. The following mutation is an example of a series of functions we applied to the data to turn card-set abbreviations into printing years:

```
MTG.df <- mutate(MTG.df, y1993 = as.numeric(str_count(MTG.df$PrintingsList,"LEA") >= 1|
                                            str_count(MTG.df$PrintingsList,"LEB") >= 1|
                                            str_count(MTG.df$PrintingsList,"2ED") >= 1|
                                            str_count(MTG.df$PrintingsList,"ARN") >= 1))
# The sets Alpha, Beta, Second Edition, and Arabian Nights were printed in 1993.
# We performed a similar mutation for every year up to 2018.
```

The Text variable, as it is, is not helpful for algorithmically predicting color-identity. We count words relevant to each color using Mark Rosewater's "Mechanical Color Pie 2017" as a guide. Below, find the summation of words related to the color white as an example:

```
MTG.df <- mutate(MTG.df, WhiteWords = 0+str_count(MTG.df$Text,"exile") +
                                      str_count(MTG.df$Text,"prevent") +
                                      str_count(MTG.df$Text,"defender") +
                                      str_count(MTG.df$Text,"double strike") +
                                      str_count(MTG.df$Text,"first strike") +
                                      str_count(MTG.df$Text,"flying") +
                                      str_count(MTG.df$Text,"indestructible") +
                                      str_count(MTG.df$Text,"lifelink") +
                                      str_count(MTG.df$Text,"protection") +
                                      str_count(MTG.df$Text,"token") +
                                      str_count(MTG.df$Text,"vigilance") +
                                      str_count(MTG.df$Text,"Exile") +
                                      str_count(MTG.df$Text,"Prevent") +
                                      str_count(MTG.df$Text,"Defender") +
                                      str_count(MTG.df$Text,"Double strike") +
                                      str_count(MTG.df$Text,"First strike") +
                                      str_count(MTG.df$Text,"Flying") +
                                      str_count(MTG.df$Text,"Indestructible") +
                                      str_count(MTG.df$Text,"Lifelink") +
```

```
                                        str_count(MTG.df$Text,"Protection") +
                                        str_count(MTG.df$Text,"Token") +
                                        str_count(MTG.df$Text,"Vigilance"))
MTG.df$WhiteWords[is.na(MTG.df$WhiteWords)] <- 0
# We performed a similar process with every other color
```

Some of these words are featured in more than one color. Nevertheless, they may prove invaluable in sorting. Observe the following:

```
table(MTG.df[MTG.df$WhiteWords>=4,]$ColorIdentity)
```

```
## < table of extent 0 >
```

```
table(MTG.df[MTG.df$BlueWords>=4,]$ColorIdentity)
```

```
## < table of extent 0 >
```

```
table(MTG.df[MTG.df$BlackWords>=4,]$ColorIdentity)
```

```
## < table of extent 0 >
```

```
table(MTG.df[MTG.df$RedWords>=4,]$ColorIdentity)
```

```
## < table of extent 0 >
```

```
table(MTG.df[MTG.df$GreenWords>=4,]$ColorIdentity)
```

```
## < table of extent 0 >
```

One might find slight correlation between the presence of a color's symbol in the color-identity and the presence of related key-words.

Now that preprocessing is complete, we may begin using visualizations to choose appropriate methods.

(Describe what actions were done, discuss our observations and conclusions. Describe each step of the data-mining technique used. Compare at least two models. Describe the model-validation technique we used to choose the best.)

Results

Conclusions

References

Schults, Robert. "Provides Magic: the Gathering Card Data in JSON Format." *MTGJSON*, 13 Nov. 2017, mtgjson.com/.

Rosewater, Mark. "Mechanical Color Pie 2017." *MAGIC : THE GATHERING*, Wizards of the Coast, 5 June 2017, magic.wizards.com/en/articles/archive/making-magic/mechanical-color-pie-2017-2017-06-05.

"RoboRosewater (@RoboRosewater)." *Twitter*, 3 July 2015, twitter.com/RoboRosewater?ref_src=twsrc%5Egoogle%7Ctwcamp

"Gatherer." *Magic : The Gathering*, Wizards of the Coast, gatherer.wizards.com/Pages/Default.aspx.