Tidy Tuesday 1/9/19 TV Show Ratings

Regis O'Connor

January 11, 2019

TV Show Ratings vs. The Number of Genres Tagged to the Show

This first foray into R4DS Tidy Tuesday is written in R 3.5.2 Eggshell Igloo. It uses data from ttps://www.economist.com/graphic-detail/2018/11/24/tvs-golden-age-is-real # https://www.imdb.com/ published on GitHub by the TidyTueday tribe.

The data includes ratings and share for several hundred TV shows airing from 1990 to 2018. There were a total of 22 different genres linked to these shows. Each show has up to 3 genres associated with it. It was this genres variable that needed tidying.

The packages used in this code inlcude dplyr, tidyr, stringi, and ggplot2.

The code splits the genres variable for each show into 3 extract variables using regex. Since some of the shows only had a single genre, this meant some shows had NA values after the split. A "none" factor was added to the genre list to take the place of NA and facility the tidy process.

The final plot is in ggplot2.

```
library(dplyr)
library(stringi)
library(tidyr)
library(ggplot2)


# Upload csv file

df1 <- read.csv("IMDb_Economist_tv_ratings.csv")

# Exploratory
# look at the type of variables

sapply(df1, class)

# convert date to date format

df1$date <- as.Date(df1$date)

# Why do titleID and title have a different # of factors?
# Genres has 97 factor levels - lets disaggregate into separate columns with
regex, then gather
```

```r
# The genre values are separated by a comma, so count the commas and add 1 to
get # of genres

df1$genre.count <- stri_count_regex(df1$genres, ",") + 1

summary(df1$genre.count) # max of 2 genre.count / 3 terms per genres



# Create df2 for ease of revision

df2 <- df1


# Split genres variable, remove commas, convert NA's to "none", and turn thes
e into a 23 factors combined
# Remove none factor later.

df2$extract1 <- gsub(",", "", stri_extract(df2$genres, regex="^[:alpha:]*,?")
)
df2$extract2 <- gsub(",", "", stri_extract(df2$genres, regex=",{1}[:alpha:]*,
?"))
df2$extract3 <- gsub(",", "", stri_extract(df2$genres, regex=",{1}[:alpha:]*$
"))


for (i in 1:nrow(df2))
if (is.na(df2$extract1[i])) {
  df2$extract1[i] <- "none"
}

for (i in 1:nrow(df2))
  if (is.na(df2$extract2[i])) {
    df2$extract2[i] <- "none"
  }

for (i in 1:nrow(df2))
  if (is.na(df2$extract3[i])) {
    df2$extract3[i] <- "none"
  }

df2$extract1     <- as.factor(df2$extract1)
df2$extract2     <- as.factor(df2$extract2)
df2$extract3     <- as.factor(df2$extract3)


# Create a list of the genres in 3 steps.
```

```r
# We will use this to create new columns, and then reshape the data
# 1) make a character vector of the levels in  extracts1:extracts3  2) conver
t to factor
# 3) extract the unique levels into genre.list

genre.list <- levels(as.factor(c(levels(df2$extract1), levels(df2$extract2),
levels(df2$extract3))))



# make new df3 for ease of revision
df3 <- df2

# Use genre.list to create 22 new columns for the df. Fill it with a placehol
der value (instead of NA)
df3[,genre.list]  <- "zzz"



# Move these new columns to the front of df3.
# This will make it easy to go through a logic test later.

df3 <- df3 %>%
  select(Action:Western, extract1:extract3, everything())


# Now, start logic testing the match between col names 1:22 vs. extracts1:3
# This looks for matches between genre.list column names and the extracts1:3
# TRUE indicates a match

for (i in 1:length(genre.list))
  for (j in 1:nrow(df3))
    if  (colnames(df3[i])    == as.character(df3$extract1[j])) {
      df3[j,i] <- TRUE
    } else {
     if (colnames(df3[i])    == as.character(df3$extract2[j])) {
        df3[j,i] <- TRUE
     } else {
        if (colnames(df3[i]) == as.character(df3$extract3[j])) {
          df3[j,i] <- TRUE
        } else {
          df3[j,i] <- FALSE
        }
      }
    }
```

```
# Its time to reshape!
# Make a new df to ease of revision
# Gather genres into one column, match them to the extracts split from genres

df4 <- df3

df4 <- df4 %>%
  select(Action:Western, titleId:genre.count) %>%
  gather(genre, truefalse, -titleId, -seasonNumber, -title,-date, -av_rating,
-share,
         -genres, -genre.count) %>% # Be careful!! This is different than gen
res in df1!
  mutate(genre = as.factor(genre))%>%
  filter(truefalse == TRUE) %>%     # trim the meaningless FALSE rows
  filter(genre != "none") %>%       # eliminate that "none" factor
  mutate(genre = factor(genre)) %>% # turn the genre back into a factor varia
ble
  mutate(genre.count = as.factor(genre.count)) #convert the count to a factor
too for legend in ggplot
```

TV Show Ratings by Number of Genres

Created in ggplot2.

```
g <- ggplot(df4, aes(date, av_rating))
p <- g + geom_point(aes(color=genre.count)) +
  labs(title="TV Shows Ratings by Number of Genres, 1990-2018",
       caption ="Source:www.economist.com/graphic-detail/2018/11/24/tvs-golde
n-age-is-real",
       subtitle="Created January 2019") +
  ylab("Average Rating") + xlab("Year") + theme(legend.position="bottom")
print(p)
```