NAnal - Example

Author: Mike Kuklinski
Date: 5/24/16
Language: R
Computer: Windows 10, 64-bit
Description: Walk through functions using an example data frame with NA values

I begin by sourcing the NAnal functions

```
suppressMessages(source('NAnal.R'))
```

Source file contains `df_test`, an example data frame

```
df_test
```

```
##     a  b  c  d  e f  g  h i  j  k  l m n  o  p ans
## 1   1  1  1 NA  1 1  1 NA 1  1  1  1 2 1 NA  7   1
## 2   2  2  2 NA  2 7  2  2 2  2  3  2 3 2  2  7   2
## 3   3 NA  3  3  3 2  3  3 3  3  3 NA 4 3 NA  6   3
## 4  NA NA NA  4 NA 9 NA NA 4 NA NA  5 4 4 NA  1   4
## 5  NA NA NA  5 NA 1 NA NA 5 NA NA NA 5 5 NA  8   5
## 6   6  6  6  6  6 1  6 NA 6  6  6  6 6 6 NA NA   6
## 7   7  7  7  7  7 8  7 NA 7  7  7  7 7 7 NA NA   7
## 8   1  1  1  1  1 2  1 NA 1  1  1  1 1 1 NA  6   1
## 9  NA  1  1  1  1 5  1  1 1  1 NA  1 1 1  1  4   1
## 10 NA  1  1  1  1 9  1  1 1  1  1  1 1 1  1  5   1
```

```
# Percent NA
orig_na_percent <- round(sum(is.na(df_test))/(nrow(df_test)*ncol(df_test)), 2)
orig_na_percent
```

```
## [1] 0.22
```

NAnal.var2ans

`NAnal.var2ans` will return a table showing the correlation, percent NA, and pvalue associated with each predictor variable.

```
# Using suppresswarnings() because df_test has 'essentially perfect fit' values compared to answer vari
df_var2ans <- suppressWarnings(NAnal.var2ans(df_test, 'ans'))
df_var2ans
```

```
##   variable index cor_to_ans abs_cor_to_ans num_NA pct_NA NAlessCor  pvalue
## a        a     1       1.00           1.00      4    0.4      TRUE 0.00000
## b        b     2       1.00           1.00      3    0.3      TRUE 0.00000
## c        c     3       1.00           1.00      2    0.2      TRUE 0.00000
## d        d     4       1.00           1.00      2    0.2      TRUE 0.00000
## e        e     5       1.00           1.00      2    0.2      TRUE 0.00000
## f        f     6       0.01           0.01      0    0.0      TRUE 0.98467
## g        g     7       1.00           1.00      2    0.2      TRUE 0.00000
## h        h     8       1.00           1.00      6    0.6      TRUE 0.00000
## i        i     9       1.00           1.00      0    0.0      TRUE 0.00000
## j        j    10       1.00           1.00      2    0.2      TRUE 0.00000
## k        k    11       0.99           0.99      3    0.3      TRUE 0.00003
## l        l    12       0.99           0.99      2    0.2      TRUE 0.00000
## m        m    13       0.98           0.98      0    0.0      TRUE 0.00000
## n        n    14       1.00           1.00      0    0.0      TRUE 0.00000
## o        o    15       1.00           1.00      7    0.7      TRUE 0.00000
## p        p    16      -0.04           0.04      2    0.2     FALSE 0.92322
```

NAnal.reduce_comb

`NAnal.reduce_comb` returns a list of every possible index combination by which the original data frame could be subset by complete cases.

```
df_reduce_combs <- NAnal.reduce_comb(df_test)
# Number of combinations
length(df_reduce_combs$lists)
```

```
## [1] 59
```

```
head(df_reduce_combs$lists,10)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 2
##
## [[3]]
## [1] 3
##
## [[4]]
## [1] 4
##
## [[5]]
## [1] 8
##
## [[6]]
## [1] 11
##
## [[7]]
## [1] 12
##
## [[8]]
## [1] 15
##
## [[9]]
## [1] 16
##
## [[10]]
## [1] 1 2
```

NAnal.score

`NAnal.score` runs through the list of index combinations provided by `NAnal.reduce_comb` and provides the following information for each resulting subset data frame:

Table Output names:
- AdjVarIdx - Data frame indices being subset by
- ADJVarNames - Data frame variable names associated with the indices
- NumberObs - Number of remaining observations
- PercentCC - Percent of complete cases
- PercentNA - Percent NA remaining
- Wt_Vote - Weight Score of subset Data frame

I set `min_vote_count` equal to 6. This just means that when scoring the combinations, I'd like to give more weight to resulting data frames with 6 or more observations.

```
df_scores <- NAnal.score(df_test, df_reduce_combs$lists, 6)
head(df_scores, 30)
```

```
##    AdjVarIdx ADJVarNames NumberObs PercentCC PercentNA   Wt_Vote
## 8         15           o         3         0      0.08 0.8800364
## 22       2 8         b h         3         0      0.08 0.8800364
## 35      3 15         c o         3         0      0.08 0.8800364
## 30    2 8 11       b h k         2         0      0.06 0.8797468
## 45   3 11 15       c k o         2         0      0.06 0.8797468
## 57     11 15         k o         2         0      0.06 0.8797468
## 24      2 16         b p         5         0      0.11 0.8751100
## 5          8           h         4         0      0.10 0.8743857
## 31   2 11 16       b k p         4         0      0.10 0.8743857
## 33       3 8         c h         4         0      0.10 0.8743857
## 27    2 4 16       b d p         3         0      0.10 0.8735005
## 44    3 8 11       c h k         3         0      0.10 0.8735005
## 56      8 11         h k         3         0      0.10 0.8735005
## 25     2 4 8       b d h         2         0      0.09 0.8723939
## 29 2 4 11 16     b d k p         2         0      0.09 0.8723939
## 39    3 4 15       c d o         2         0      0.09 0.8723939
## 50      4 15         d o         2         0      0.09 0.8723939
## 13      1 15         a o         1         0      0.06 0.8709712
## 16     1 2 8       a b h         1         0      0.06 0.8709712
## 28  2 4 8 11     b d h k         1         0      0.06 0.8709712
## 42 3 4 11 15     c d k o         1         0      0.06 0.8709712
## 53   4 11 15       d k o         1         0      0.06 0.8709712
## 36      3 16         c p         6         0      0.12 0.8708116
## 46   3 11 16       c k p         5         0      0.12 0.8697624
## 58     11 16         k p         5         0      0.12 0.8697624
## 40    3 4 16       c d p         4         0      0.12 0.8685034
## 2          2           b         7         0      0.13 0.8671745
## 17    1 2 16       a b p         3         0      0.12 0.8669645
## 37     3 4 8       c d h         3         0      0.12 0.8669645
## 43 3 4 11 16     c d k p         3         0      0.12 0.8669645
```

NAnal.score - continued

Looking over the scores, I can see that by subsetting the original data frame by complete cases of variable `o`, I'll get 3 rows/observations but reduce the NA percent to 8% respectively.

```
# Get list of index combinations
reduce_idx_list <- df_scores$AdjVarIdx
# Identify the first combination (associated with 'o')
idx_num <- grep('^o$',df_scores$ADJVarNames)
# Get indices as a list
reduce_idx <- lapply(str_split(as.character(reduce_idx_list[idx_num]), pattern = ' '), as.integer)
reduce_idx
```

```
## [[1]]
## [1] 15
```

```
# Subset original data frame by complete cases of indices
new_df <- df_test[complete.cases(df_test[,reduce_idx[[1]]]),]
new_df
```

```
##     a b c  d e f g h i j  k l m n o p ans
## 2   2 2 2 NA 2 7 2 2 2 2  3 2 3 2 2 7   2
## 9  NA 1 1  1 1 5 1 1 1 1 NA 1 1 1 1 4   1
## 10 NA 1 1  1 1 9 1 1 1 1  1 1 1 1 1 5   1
```

Similarly, I see that if I subset by complete cases of `c p`, I get 6 rows/observations and 12% NA.

```
# Identify the first combination (associated with 'c')
idx_num <- grep('^c p$',df_scores$ADJVarNames)
# Get indices as a list
reduce_idx <- lapply(str_split(as.character(reduce_idx_list[idx_num]), pattern = ' '), as.integer)
reduce_idx
```

```
## [[1]]
## [1]  3 16
```

```
# Subset original data frame by complete cases of indices
new_df <- df_test[complete.cases(df_test[,reduce_idx[[1]]]),]
new_df
```

```
##     a  b c  d e f g  h i j  k  l m n  o p ans
## 1   1  1 1 NA 1 1 1 NA 1 1  1  1 2 1 NA 7   1
## 2   2  2 2 NA 2 7 2  2 2 2  3  2 3 2  2 7   2
## 3   3 NA 3  3 3 2 3  3 3 3  3 NA 4 3 NA 6   3
## 8   1  1 1  1 1 2 1 NA 1 1  1  1 1 1 NA 6   1
## 9  NA  1 1  1 1 5 1  1 1 1 NA  1 1 1  1 4   1
## 10 NA  1 1  1 1 9 1  1 1 1  1  1 1 1  1 5   1
```

Once you're satisfied with the ratio between the number of observations and percent NA, I would recommend using `na.roughfix()` or `rfImpute()` from the `randomforest` package to fill in the remaining NA values.