

R Generate Variables Conditional on Other Variables with elseif, Approximate all.equals(), na_if() and is.na(), etc (dplyr)

Fan Wang

2020-04-22

Contents

Generate Variables Conditional On Others 1

Generate Variables Conditional On Others

Go to the [RMD](#), [R](#), [PDF](#), or [HTML](#) version of this file. Go back to [fan's REconTools](#) Package, [R Code Examples](#) Repository ([bookdown site](#)), or [Intro Stats with R](#) Repository ([bookdown site](#)).

case_when Basic Example Given several other variables, and generate a new variable when these variables satisfy conditions. Note that case_when are ifelse type statements. So below

1. group one is below 16 MPG
2. when do $qsec \geq 20$ second line that is elseif, only those that are ≥ 16 are considered here
3. then think about two dimensional mpg and qsec grid, the lower-right area, give another category to manual cars in that group

```
# Get mtcars
df_mtcars <- mtcars

# case_when with mtcars
df_mtcars <- df_mtcars %>%
  mutate(mpg_qsec_am_grp =
    case_when(mpg < 16 ~ "< 16 MPG",
              qsec >= 20 ~ "> 16 MPG & qsec >= 20",
              am == 1 ~ "> 16 MPG & asec < 20 & manual",
              TRUE ~ "Others"))

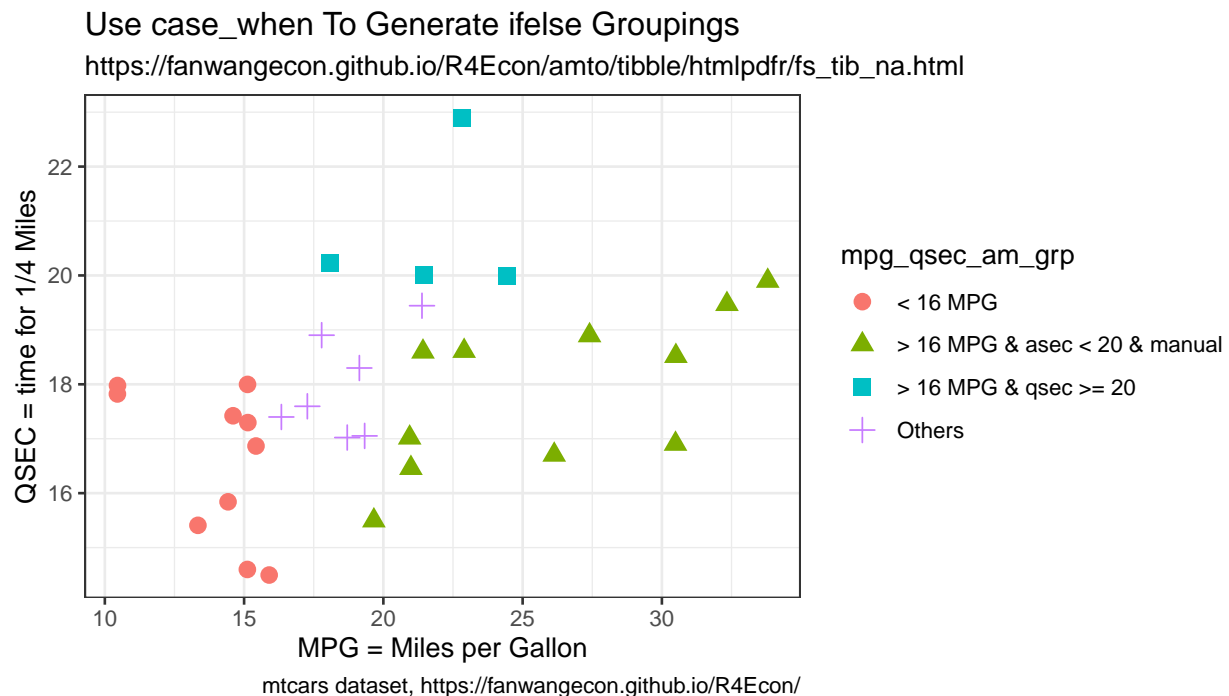
# # For dataframe
# df.reg <-df.reg %>% na_if(-Inf) %>% na_if(Inf)
# # For a specific variable in dataframe
# df.reg.use %>% mutate(!!(var.input) := na_if(!sym(var.input), 0))
#
# # Setting to NA
# df.reg.use <- df.reg.guat %>% filter(!sym(var.mth) != 0)
# df.reg.use.log <- df.reg.use
# df.reg.use.log[which(is.nan(df.reg.use$prot.imputed.log)),] = NA
# df.reg.use.log[which(df.reg.use$prot.imputed.log==Inf),] = NA
# df.reg.use.log[which(df.reg.use$prot.imputed.log==Inf),] = NA
# df.reg.use.log <- df.reg.use.log %>% drop_na(prot.imputed.log)
# # df.reg.use.log$prot.imputed.log
```

Now we generate scatter plot based on the combined factors

```
# Labeling
st_title <- paste0('Use case_when To Generate ifelse Groupings')
st_subtitle <- paste0('https://fanwangecon.github.io/',
                      'R4Econ/amto/tibble/htmlpdf/fs_tib_na.html')
st_caption <- paste0('mtcars dataset, ',
                     'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'MPG = Miles per Gallon'
st_y_label <- 'QSEC = time for 1/4 Miles'

# Graphing
plt_mtcars_casewhen_scatter <-
  ggplot(df_mtcars,
    aes(x=mpg, y=qsec,
        colour=mpg_qsec_am_grp,
        shape=mpg_qsec_am_grp)) +
  geom_jitter(size=3, width = 0.15) +
  labs(title = st_title, subtitle = st_subtitle,
       x = st_x_label, y = st_y_label, caption = st_caption) +
  theme_bw()

# show
print(plt_mtcars_casewhen_scatter)
```



Generate NA values if Variables have Certain Value In the example below, in one line:

1. generate a random standard normal vector
2. two set na methods:
 - if the value of the standard normal is negative, set value to -999, otherwise MPG, replace the value -999 with NA

- `case_when` only with type specific NA values
 - [Assigning NA yields error in case_when](#)
 - note we need to conform NA to type
3. generate new categorical variable based on NA condition using `is.na` with both string and numeric NAs jointly considered.
- fake NA string to be printed on chart

```
# Get mtcars
df_mtcars <- mtcars

# Make some values of mpg randomly NA
# the NA has to conform to the type of the remaining values for the new variable
# NA_real_, NA_character_, NA_integer_, NA_complex_
set.seed(2341)
df_mtcars <- df_mtcars %>%
  mutate(mpg_wth_NA1 = na_if(
    case_when(
      rnorm(n(),mean=0,sd=1) < 0 ~ -999,
      TRUE ~ mpg),
    -999)) %>%
  mutate(mpg_wth_NA2 = case_when(
    rnorm(n(),mean=0,sd=1) < 0 ~ NA_real_,
    TRUE ~ mpg)) %>%
  mutate(mpg_wth_NA3 = case_when(
    rnorm(n(),mean=0,sd=1) < 0 ~ NA_character_,
    TRUE ~ "shock > 0 string"))

# Generate New Variables based on if mpg_wth_NA is NA or not
# same variable as above, but now first a category based on if NA
# And we generate a fake string "NA" variable, this is not NA
# the String NA allows for it to be printed on figure
df_mtcars <- df_mtcars %>%
  mutate(group_with_na =
    case_when(is.na(mpg_wth_NA2) & is.na(mpg_wth_NA3) ~
      "Rand String and Rand Numeric both NA",
    mpg < 16 ~ "< 16 MPG",
    qsec >= 20 ~ "> 16 MPG & qsec >= 20",
    am == 1 ~ "> 16 MPG & asec < 20 & manual",
    TRUE ~ "Fake String NA"))

# show
kable(head(df_mtcars %>% select(starts_with('mpg')),13)) %>%
  kable_styling_fc()

## Setting to NA
# df.reg.use <- df.reg.guat %>% filter(!sym(var.mth) != 0)
# df.reg.use.log <- df.reg.use
# df.reg.use.log[which(is.nan(df.reg.use$prot.imputed.log)),] = NA
# df.reg.use.log[which(df.reg.use$prot.imputed.log==Inf),] = NA
# df.reg.use.log[which(df.reg.use$prot.imputed.log==Inf),] = NA
# df.reg.use.log <- df.reg.use.log %>% drop_na(prot.imputed.log)
# df.reg.use.log$prot.imputed.log
```

Now we generate scatter plot based on the combined factors, but now with the NA category

mpg	mpg_wth_NA1	mpg_wth_NA2	mpg_wth_NA3
21.0	NA	NA	shock > 0 string
21.0	21.0	21.0	NA
22.8	NA	NA	NA
21.4	NA	21.4	NA
18.7	NA	18.7	NA
18.1	18.1	NA	shock > 0 string
14.3	14.3	NA	shock > 0 string
24.4	NA	24.4	NA
22.8	22.8	22.8	NA
19.2	19.2	NA	NA
17.8	NA	NA	NA
16.4	16.4	16.4	NA
17.3	NA	NA	shock > 0 string

```

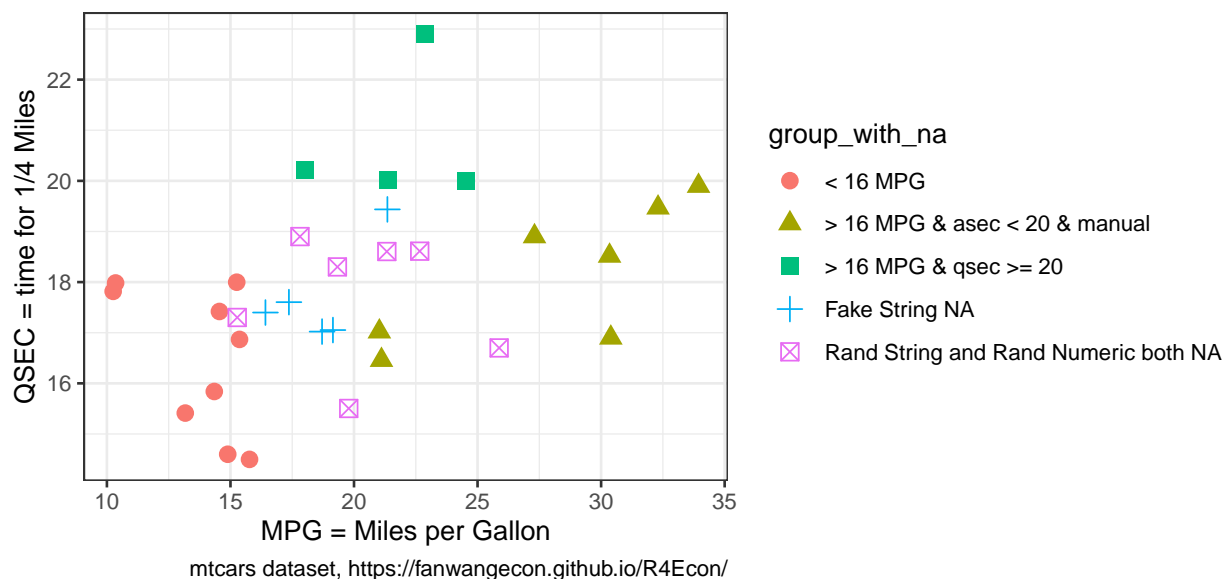
# Labeling
st_title <- paste0('Use na_if and is.na to Generate and Distinguish NA Values\n',
  'NA_real_, NA_character_, NA_integer_, NA_complex_')
st_subtitle <- paste0('https://fanwangecon.github.io/',
  'R4Econ/amto/tibble/htmlpdf/fs_tib_na.html')
st_caption <- paste0('mtcars dataset, ',
  'https://fanwangecon.github.io/R4Econ/')
st_x_label <- 'MPG = Miles per Gallon'
st_y_label <- 'QSEC = time for 1/4 Miles'

# Graphing
plt_mtcars_ifisna_scatter <-
  ggplot(df_mtcars,
    aes(x=mpg, y=qsec,
      colour=group_with_na,
      shape=group_with_na)) +
  geom_jitter(size=3, width = 0.15) +
  labs(title = st_title, subtitle = st_subtitle,
    x = st_x_label, y = st_y_label, caption = st_caption) +
  theme_bw()

# show
print(plt_mtcars_ifisna_scatter)

```

Use `na_if` and `is.na` to Generate and Distinguish NA Values
`NA_real_`, `NA_character_`, `NA_integer_`, `NA_complex_`
https://fanwangecon.github.io/R4Econ/amto/tibble/htmlpdf/fs_tib_na.html



Approximate Values Comparison

- r values almost the same
- `all.equal`

From numeric approximation, often values are very close, and should be set to equal. Use `isTRUE(all.equal)`. In the example below, we randomly generates four arrays. Two of the arrays have slightly higher variance, two arrays have slightly lower variance. They sd are to be 10 times below or 10 times above the tolerance comparison level. The values are not the same in any of the columns, but by allowing for almost true given some tolerance level, in the low standard deviation case, the values differences are within tolerance, so they are equal.

This is an essential issue when dealing with optimization results.

```
# Set tolerance
tol_lvl = 1.5e-3
sd_lower_than_tol = tol_lvl/10
sd_higher_than_tol = tol_lvl*10

# larger SD
set.seed(123)
mt_runif_standard <- matrix(rnorm(10,mean=0,sd=sd_higher_than_tol), nrow=5, ncol=2)

# small SD
set.seed(123)
mt_rnorm_small_sd <- matrix(rnorm(10,mean=0,sd=sd_lower_than_tol), nrow=5, ncol=2)

# Generates Random Matirx
tb_rnorm_runif <- as_tibble(cbind(mt_rnorm_small_sd, mt_runif_standard))

# Are Variables the same, not for strict comparison
tb_rnorm_runif_approxi_same <- tb_rnorm_runif %>%
```

```

mutate(V1_V2_ALMOST_SAME =
  case_when(isTRUE(all.equal(V1, V2, tolerance=tol_lvl)) ~
    paste0('TOL=',sd_lower_than_tol,', SAME ALMOST'),
    TRUE ~
    paste0('TOL=',sd_lower_than_tol,', NOT SAME ALMOST')) %>%
mutate(V3_V4_ALMOST_SAME =
  case_when(isTRUE(all.equal(V3, V4, tolerance=tol_lvl)) ~
    paste0('TOL=',sd_higher_than_tol,', SAME ALMOST'),
    TRUE ~
    paste0('TOL=',sd_higher_than_tol,', NOT SAME ALMOST'))

# Print
kable(tb_rnorm_runif_approxi_same) %>% kable_styling_fc_wide()

```

V1	V2	V3	V4	V1_V2_ALMOST_SAME	V3_V4_ALMOST_SAME
-0.0000841	0.0002573	-0.0084071	0.0257260	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST
-0.0000345	0.0000691	-0.0034527	0.0069137	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST
0.0002338	-0.0001898	0.0233806	-0.0189759	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST
0.0000106	-0.0001030	0.0010576	-0.0103028	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST
0.0000194	-0.0000668	0.0019393	-0.0066849	TOL=0.00015, SAME ALMOST	TOL=0.015, NOT SAME ALMOST