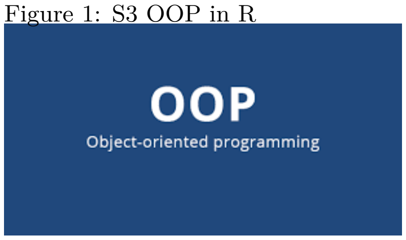
Writing a simple R package in S3.

Ronald (Ryy) Glenn Thomas

2024-01-23

Table of contents

Figure 1: S3 OOP in R



1 Introduction

If you, like me, feel its time to expand your R programming armamentarium to include S3 methods. This blog may help. Where to start?

In this post we'll walk through an example of a simple "table 1" function using S3 methods.

We'll start with the 'raw' data from a sample of the Penguins data set and return a dataframe with summary measures.

Let's begin by reading the relevant chapters in Advanced R (add reference).

Also useful other references:

Introduction to Scientific Programming and Simulation using R. Jomes. Maillardet, Robinson.

[1608.07161] A Simple Guide to S3 Methods https://arxiv.org/abs/1608.07161

Why your S3 method isn't working | R-bloggers

Dealing with S3 methods in R with a simple example | R-bloggers

Video on S3 Classes in R by Dr Andrew Robinson | R-bloggers

Unexported S3 Methods and R Packages | R-bloggers

Simple Guide to S3 Methods | R-bloggers

The S3 OOP system | R-bloggers

S3 methods allow coders to write functions that perform differently for different classes of objects.

In our project we want to build a function that creates a row in the 'Table 1' table for each factor in the formula regardless of the mode of the factor.

7/1/23 Now reading Nick Tierney R journal paper.

```
source("~/shr/zz.tools.R")
library(pacman)

p_load(tidyverse, dplyr, gapminder, thematic, palmerpenguins, tidyverse, knitr, lubridate, reac
knitr::opts_chunk$set(collapse = T)
set.seed(101)
dat <- palmerpenguins::penguins %>%
  fil(!is.na(sex))
dat1  <- slice_sample(dat, n=10) |>
sel(species, island, bill_length_mm)
```

```
#' Table one summaries
#'
#' Summarizes baseline trial results by treatment
#' @param data dataframe
#' @param form formula y ~ x1 + x2
#' @param ... extra parameters passed through to speciality functions
#' @return a dataframe
#' @examples
#' table1(dat2, form = arm ~ sex + age, annot = FALSE)
```

```r
#' @export
table1 <- function(form, data, ...) {
  UseMethod("table1")
}

row_name <- function(x, nm, ...) {
  UseMethod("row_name")
}

row_name.character <- function(x, nm, ...) {
  # browser()
  u <- unique(x)
  categs <- paste("    ", u[!is.na(u)])
  return(c(nm, categs))
}

row_name.factor <- row_name.character

row_name.logical <- row_name.character

row_name.numeric <- function(x, nm, ...) {
  return(nm)
}

row_summary <- function(x) {
  UseMethod("row_summary")
}

row_summary.character <- function(x) {
  # browser()
  df <- data.frame(x = x, y = dep) |> na.omit()
  t1 <- df |>
    tabyl(x, y) |>
    adorn_percentages("col") |>
    adorn_pct_formatting(digits = 0) |>
    adorn_ns(position = "front") |>
    select(-x)
  # browser()
  t1 <- as_tibble(t1)
  t2 <- table(df$x, df$y) |> as.data.frame.matrix()
  rbind("", t1)
}

row_summary.factor <- row_summary.character
row_summary.logical <- row_summary.character

row_summary.numeric <- function(x) {
  sp <- split(x, dep)
  nms <- names(sp)
  mm <- sp |>
    map_vec(mean, na.rm = TRUE) |>
    round(2) |>
    as.character() |>
    matrix(1)
  ss <- sp |>
    map_vec(sd, na.rm = TRUE) |>
    round(2) |>
    paste0("(", x = _, ")") |>
    matrix(1)
  bb <- paste(unlist(mm), unlist(ss)) |> matrix(nrow = nrow(mm))
  colnames(bb) <- nms
  bb <- bb |> as_tibble()
  bb
}

row_pv <- function(x) {
  UseMethod("row_pv")
}

row_pv.character <- function(x) {
  tab <- data.frame(x = x, y = dep) |>
    na.omit() |>
    tabyl(x, y)
  if (!(nrow(tab) >= 2 & ncol(tab) >= 2)) {
    return(NA)
  }
  browser()
  pv <- stats::fisher.test(tab, simulate.p.value = TRUE)$p.value |>
    round(4)
  return(c(pv, rep("", nrow(tab))))
```

```r
}

row_pv.factor <- row_pv.character
row_pv.logical <- row_pv.character

row_pv.numeric <- function(x) {
  df <- data.frame(x = x, y = dep)
  tab <- table(x, dep)
  pv <- ifelse((nrow(tab) >= 2 & ncol(tab) >= 2),
    stats::fisher.test(tab, simulate.p.value = TRUE)$p.value, NA
  ) |>
    round(4)
  return(c(pv, rep("", nrow(tab))))
}

row_pv.factor <- row_pv.character
row_pv.logical <- row_pv.character

row_pv.numeric <- function(x) {
  df <- data.frame(x = x, y = dep)
  pv <- tidy(anova(lm(x ~ y, data = df)))$p.value[1] |>
    round(4)
  return(pv)
}

#' @export
#' @describeIn table1 interprets formula and yields publication tables
table1.formula <- function(form, data, ...) {
  vars <- all.vars(form)
  dep <<- data[[vars[1]]]
  indep <- data[vars[-1]]
  # browser()
  left <- indep |>
    imap(row_name, ...) |>
    unlist() |>
    enframe(name = NULL) |>
    setNames("variable")
  right <- indep |>
    map(row_pv) |>
    unlist() |>
    enframe(name = NULL) |>
    setNames("p-value")
  mid <- indep |>
    map_dfr(row_summary) |>
    identity()
  mid <- bind_rows(mid)
  browser()
  bind_cols(left, mid, right)
}



p_load(palmerpenguins, dplyr)
p1 <- sample_n(penguins, 300) |>
  dplyr::select(species, sex, body_mass_g)
# p1 <- pp |> dplyr::mutate(sex1 = sex)
t <- table1(species ~ sex + body_mass_g, data = p1)
```

1.1 Notes

2023-08-03 17:37:04

1. can't handle logical variables yet
2. categorical values should be indented
3. add option to change continuous summary to median IQR
4. review atable, furniture, and tableone for features.
5. maybe a "style" option for NEJM, JAMA, lancet

2 Bug 1

```


library(palmerpenguins)
# table1(species ~ sex + body_mass_g, data = penguins) */

# Error in `map2()`:
```

```
#   In index: 1.
#   With name: sex.
# Caused by error in `UseMethod()`:
# ! no applicable method for 'row_name' applied to an object of class "factor"
# Run `rlang::last_trace()` to see where the error occurred.
```

solved. needed to add “addNA” to the p-value.factor function.

```
zz.table1.c = function(df, form, pv=TRUE, totl=TRUE, grps=TRUE) {
if (!require("pacman")) install.packages("pacman", repo="cran.rstudio.com")
pacman::p_load(janitor )
prep =  function(df, form) {
dfr = df %>%
    ungroup %>%
    sel(all.vars(form[[3]]))
df_list = dfr %>%  split( df_grp) %>%
list_merge(., "Total" = dfr) %>%
purrr::transpose()
}
process1 = function(x){
pv_chr = data.frame(x[["Total"]],df_grp) %>%
pvalue_chr
ll = x[[length(x)]]%>%  as.factor %>% levels
ll_indent = paste("\\hspace{5mm}   ",ll)
sum_chr = x %>%
    lapply(function(x) factor(x, levels=ll)) %>%
    map(categ)   %>% as_tibble  %>%
    cbind(variable=ll_indent, ., 'p-value'=NA) %>%
    mut(variable=as.character(variable)) %>%
    rbind(NA, .)
sum_chr[1,ncol(sum_chr)]=pv_chr
# browser()
return(sum_chr)
}
process2 = function(x){
pv_num = data.frame(x[["Total"]],df_grp) %>%
   pvalue_num
sum_num = x %>%
   map_chr(contin)  %>%
   bind_rows  %>%
   cbind(variable=NA,., 'p-value'=pv_num)    %>%
   mut(variable=as.character(variable))
return(sum_num)
}
contin= function(x) {
s1 = zz.sum.min(x)
paste0(s1['Mean'],"$\\pm$", s1['SD'], " ({\\scriptsize $",s1['N'],"$})") }
categ = function(x) {
prps = table(x) %>% prop.table  %>%  round(2)*100
cnts_prps =  table(x)%>%
paste0(.," ({\\scriptsize $",prps,"$})")
}
pvalue_num = function(df) {
tidy(anova(lm(as.formula(paste(names(df),  \
collapse="~")), data = df)))$p.value[1]
}
pvalue_chr = function(df) {
tab =  table(df[,1], df[,2])
ifelse((nrow(tab) >=2 & ncol(tab) >=2),
       stats::fisher.test(tab,simulate.p.value=T)$p.value, NA)
}
fieldclass =sapply(df, class)%>%  enframe %>%
slice(match(all.vars(form[[3]]),name))
groupclass =sapply(df, class)%>%  enframe %>%
slice(match(all.vars(form[[2]]),name))
df_grp<- df %>%  pull(groupclass$name)
df2 = prep(df, form)
out = df2 %>%
  map_if(fieldclass$value=="numeric" | fieldclass$value=="integer", \
  function(x){process2(x)}) %>%
       map_if(fieldclass$value=="character", function(x){process1(x)})   %>%
imap(function(x,y) {
         y2 = ifelse(fieldclass$value[fieldclass$name == y]=="character",
             paste(y, "-- {\\scriptsize no. (\\%)}"), y)
     x[1,1]=y2
#  browser()
     x
       }) %>%
   bind_rows()
```

```
on= names(out)
nn = tabyl(df_grp)%>%
    adorn_totals() %>%
    pull(n)
names(out) = paste(rep("{\\bf",length(on)),on, \
c("",paste0("\\scriptsize(n=",nn,")"),""),rep("}",length(on)))
if (!grps) out = out %>%  sel(contains("variable"),\
contains("Total"),contains("p-value"))
if (!pv) {
    out = out %>%  sel(-contains("p-value"))}
if (!totl) out = out %>%  sel(-contains("Total"))
return(out)
}
```