purr::map() demo for Code review

PennCHOP microbiome program, Scott Garrett Daniel

July 23, 2019

Contents

Packages and setting up data

Based on: https://aosmith.rbind.io/2019/07/22/automate-model-fitting-with-loops/

```
#devtools::install_github("thomasp85/patchwork")
library(purrr) # v. 0.3.2
library(ggplot2) # v. 3.2.0
library(patchwork) # v. 0.0.1, github only, see above ^
library(broom) # v. 0.5.2

dat = structure(list(group = structure(c(1L, 1L, 1L, 1L, 1L, 1L, 1L,
1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 1L, 2L, 2L, 2L, 2L, 2L, 2L, 2L, 2L,
2L, 2L, 2L, 2L, 2L, 2L), .Label = c("a", "b"), class = "factor"),
    resp = c(10.48, 9.87, 11.1, 8.56, 11.15, 9.53, 8.99, 10.06,
    11.02, 10.57, 11.85, 10.11, 9.25, 11.66, 10.72, 8.34, 10.58,
    10.47, 9.46, 11.13, 8.35, 9.69, 9.82, 11.47, 9.13, 11.53,
    11.05, 11.03, 10.84, 10.22), slp = c(38.27, 46.33, 44.29,
    35.57, 34.78, 47.81, 50.45, 46.31, 47.82, 42.07, 31.75, 65.65,
    47.42, 41.51, 38.69, 47.84, 46.22, 50.66, 50.69, 44.09, 47.3,
    52.53, 53.63, 53.38, 27.34, 51.83, 56.63, 32.99, 77.5, 38.24
    ), grad = c(0.3, 0.66, 0.57, 0.23, 0.31, 0.48, 0.5, 0.49,
    2.41, 0.6, 0.27, 0.89, 2.43, 1.02, 2.17, 1.38, 0.17, 0.47,
    1.1, 3.28, 6.14, 3.8, 4.35, 0.85, 1.13, 1.11, 2.93, 1.13,
    4.52, 0.13)), class = "data.frame", row.names = c(NA, -30L) )
head(dat)
```

```
##   group  resp   slp grad
## 1     a 10.48 38.27 0.30
## 2     a  9.87 46.33 0.66
## 3     a 11.10 44.29 0.57
## 4     a  8.56 35.57 0.23
## 5     a 11.15 34.78 0.31
## 6     a  9.53 47.81 0.48
```

Function for model generation

```
ttest_fun = function(response) {
  form = paste(response, "~ group")
  lm(as.formula(form), data = dat)
}
ttest_fun(response = "resp")
```

```
##
## Call:
```

```
## lm(formula = as.formula(form), data = dat)
##
## Coefficients:
## (Intercept)       groupb
##     10.3280      -0.1207
```

Set names for vars we want to model

```
vars = names(dat)[2:4]
vars
```

```
## [1] "resp" "slp"  "grad"
```

```
vars = set_names(vars)
vars
```

```
##   resp    slp   grad
## "resp"  "slp" "grad"
```

Map the model to each var

```
models = vars %>%
     map(ttest_fun)
models
```

```
## $resp
##
## Call:
## lm(formula = as.formula(form), data = dat)
##
## Coefficients:
## (Intercept)       groupb
##     10.3280      -0.1207
##
##
## $slp
##
## Call:
## lm(formula = as.formula(form), data = dat)
##
## Coefficients:
## (Intercept)       groupb
##       43.91         4.81
```

```
##
##
## $grad
##
## Call:
## lm(formula = as.formula(form), data = dat)
##
## Coefficients:
## (Intercept)       groupb
##      0.8887       1.2773
```

```
#same thing rather than set_names in above code block
# vars %>%
#      set_names() %>%
#      map(ttest_fun)
```

Create a ggplot graph function

```
resid_plots = function(model, modelname) {
     output = augment(model) #extract residuals and fitted values via broom::augment()

     res.v.fit = ggplot(output, aes(x = .fitted, y = .resid) ) +
          geom_point() +
          theme_bw(base_size = 16)

     res.box = ggplot(output, aes(x = "", y = .resid) ) +
          geom_boxplot() +
          theme_bw(base_size = 16) +
          labs(x = NULL)

     res.v.fit + res.box +
          plot_annotation(title = paste("Residuals plots for", modelname) )
# plot_annotation is part of patchwork and glues together the individual plots
}
```

Loop through models and graph each one

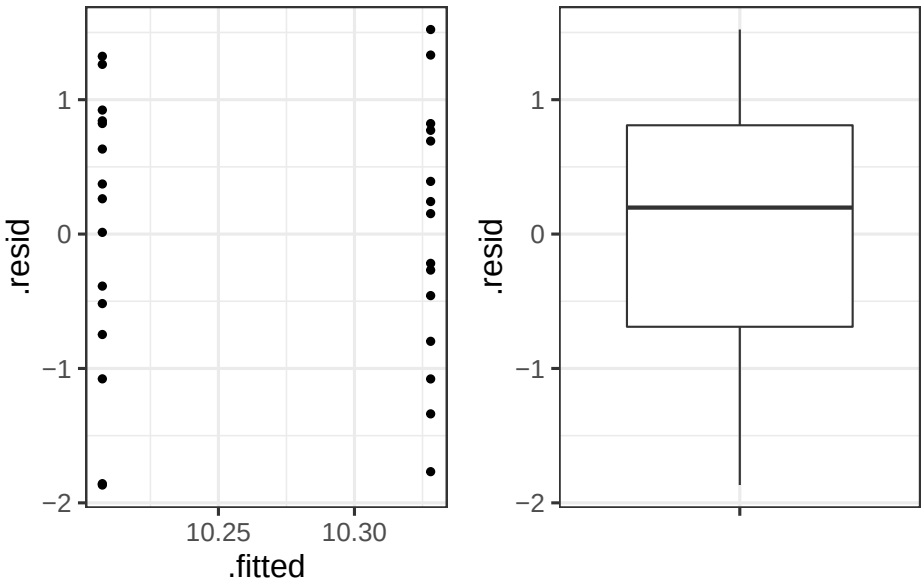
```
#for one model:
#resid_plots(model = models[[1]], modelname = names(models)[1])

#loop through all models
```

```
residplots = imap(models, resid_plots)
residplots
```

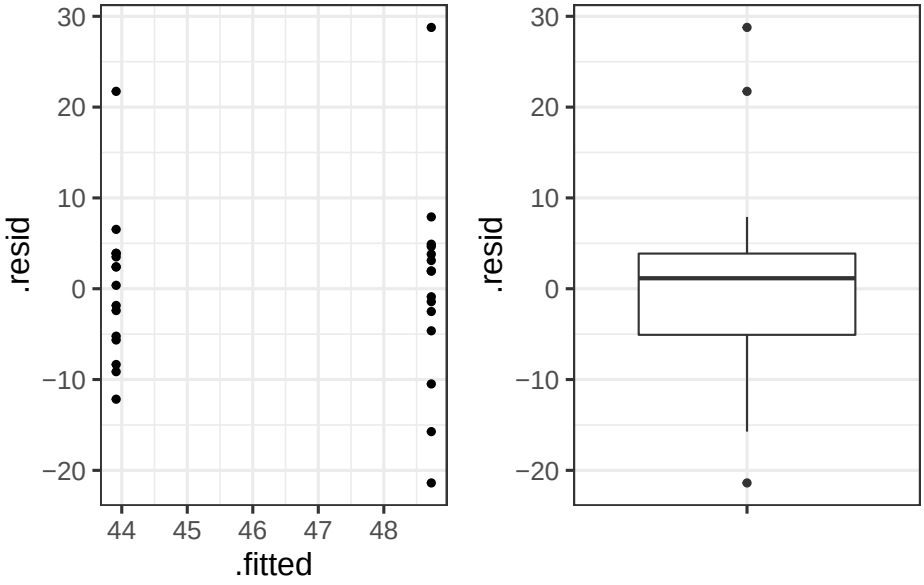
```
## $resp
```

Residuals plots for resp



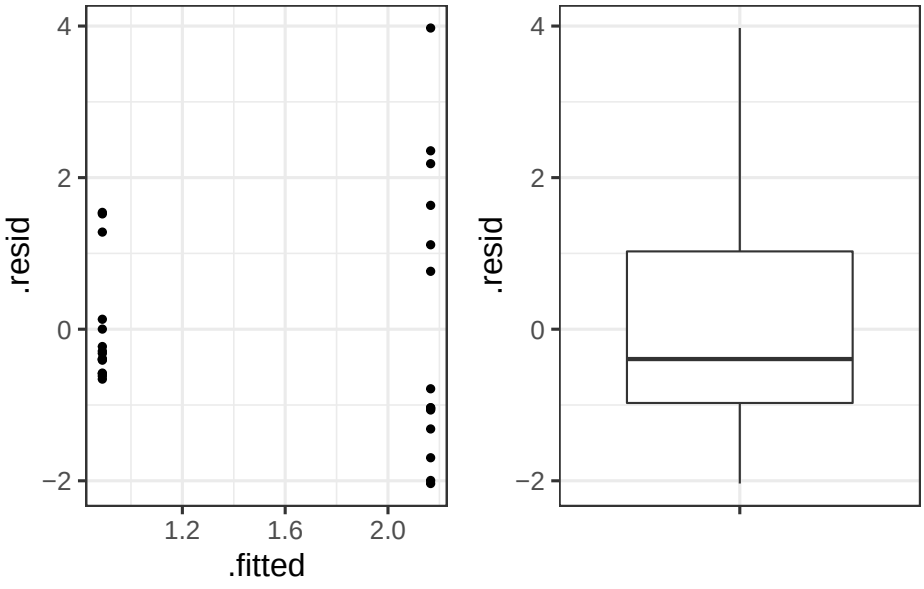

```
##
## $slp
```

Residuals plots for slp




```
##
## $grad
```

Residuals plots for grad



Updating one of the models or subsetting

```
gradmod = ttest_fun("log(grad)")

models$log_grad = gradmod
models$grad = NULL
models
```

```
## $resp
##
## Call:
## lm(formula = as.formula(form), data = dat)
##
## Coefficients:
## (Intercept)       groupb
##     10.3280      -0.1207
##
##
## $slp
##
## Call:
## lm(formula = as.formula(form), data = dat)
##
## Coefficients:
## (Intercept)       groupb
##       43.91         4.81
##
##
## $log_grad
##
## Call:
## lm(formula = as.formula(form), data = dat)
##
## Coefficients:
## (Intercept)       groupb
##     -0.4225       0.7177
```

```
models[!names(models) %in% "slp"]
```

```
## $resp
##
## Call:
## lm(formula = as.formula(form), data = dat)
##
## Coefficients:
## (Intercept)       groupb
```

```
##     10.3280      -0.1207
##
##
## $log_grad
##
## Call:
## lm(formula = as.formula(form), data = dat)
##
## Coefficients:
## (Intercept)       groupb
##     -0.4225       0.7177
```

Extracting statistics from the models

```
res_anova = map_dfr(models, tidy, conf.int = TRUE, .id = "variable")

#map_dfr and map_dfc return data frames by row-binding and col-binding respectively
res_anova
```

```
## # A tibble: 6 x 8
##   variable term    estimate std.error statistic  p.value conf.low conf.high
##   <chr>    <chr>      <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
## 1 resp     (Inter~   10.3       0.260    39.7   3.60e-26   9.80     10.9
## 2 resp     groupb    -0.121     0.368    -0.328 7.45e- 1  -0.874     0.632
## 3 slp      (Inter~   43.9       2.56     17.2   2.18e-16  38.7      49.2
## 4 slp      groupb     4.81      3.62      1.33  1.95e- 1  -2.61     12.2
## 5 log_grad (Inter~   -0.423     0.255    -1.66  1.09e- 1  -0.945     0.0997
## 6 log_grad groupb     0.718     0.361     1.99  5.64e- 2  -0.0208    1.46
```

Example from one of the projects I'm working on

Sample data

Important response and predictor variables

```
#in 16S bacterial data, we care about how the diversity changes from one condition to the ne
responses = c("shannon", "richness", "faith_pd")
predictors = c("Location", "study_day")
```

Modify ttest function

```
ttest_funs = function(response, predictor) {

  form = paste0(response, " ~ ", predictor)
  lm(as.formula(form), data = s)

}
```

Run it!

```
predictors = set_names(predictors,predictors)
responses = set_names(responses,responses)

for (x in predictors) {
  responses %>%
    map(~ ttest_funs(.,x))
}

#map2(responses, predictors, ~ ttest_funs(.x,.y))

#ttest_fun(..1, ..2)
```