Working with tidymodels

OCRUG meetup

Emil Hvitfeldt

2019-1-29

tidymodels is a "meta-package" for modeling and statistical analysis that share the underlying design philosophy, grammar, and data structures of the tidyverse.

```r
library(tidymodels)
```

```
## ✔ broom     0.5.1        ✔ purrr     0.3.0
## ✔ dials     0.0.2        ✔ recipes   0.1.4
## ✔ dplyr     0.7.8        ✔ rsample   0.0.4
## ✔ ggplot2   3.1.0        ✔ tibble    2.0.1
## ✔ infer     0.4.0        ✔ yardstick 0.0.2
## ✔ parsnip   0.0.1.9000
```

The packages

- broom
- dials
- dplyr
- ggplot2
- infer
- parsnip
- purrr
- recipes
- rsample
- tibble
- yardstick

The packages (tidyverse)

- broom
- dials
- **dplyr**
- **ggplot2**
- infer
- parsnip
- **purrr**
- recipes
- rsample
- **tibble**
- yardstick

The packages (tidyverse)

- broom
- dials
- **dplyr**
- **ggplot2**
- infer
- parsnip
- **purrr**
- recipes
- rsample
- **tibble**
- yardstick

The packages

- broom
- dials
- dplyr
- ggplot2
- infer
- **parsnip**
- purrr
- **recipes**
- **rsample**
- tibble
- **yardstick**

⚠ Disclaimer ⚠

This talk is not designed to give opinions with respect to modeling best practices.

This talk is designed to showcase what packages are available and what they can do.

Consider 32 cars from 1973-74

```
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```r
model_glm <- glm(am ~ disp + drat + qsec, data = mtcars,
                 family = "binomial")
```

```r
model_glm <- glm(am ~ disp + drat + qsec, data = mtcars,
                 family = "binomial")
```

```r
predict(model_glm)
```

```
##           Mazda RX4       Mazda RX4 Wag          Datsun 710
##           105.97448            69.85214            27.10173
##      Hornet 4 Drive   Hornet Sportabout             Valiant
##          -276.59440          -240.20025          -313.42683
##          Duster 360           Merc 240D            Merc 230
##          -158.99087          -123.81721          -284.08255
##            Merc 280           Merc 280C          Merc 450SE
##           -20.37716           -59.07966          -167.78204
##          Merc 450SL         Merc 450SLC  Cadillac Fleetwood
##          -180.68287          -206.48454          -458.77204
## Lincoln Continental   Chrysler Imperial            Fiat 128
##          -427.72554          -357.75792            27.25037
##         Honda Civic      Toyota Corolla       Toyota Corona
##           164.39652            20.76278           -90.84576
##    Dodge Challenger         AMC Javelin          Camaro Z28
##          -211.90064          -189.27739           -74.78345
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2
##          -297.35324            63.64819           184.39936
##        Lotus Europa      Ford Pantera L        Ferrari Dino
##           146.50220            24.28831           162.60217
##       Maserati Bora          Volvo 142E
##            21.69348            33.80864
```

```r
library(glmnet)
model_glmnet <- glmnet(am ~ disp + drat + qsec, data = mtcars,
                       family = "binomial")
```

```r
library(glmnet)
model_glmnet <- glmnet(am ~ disp + drat + qsec, data = mtcars,
                       family = "binomial")
```

```
## Error in glmnet(am ~ disp + drat + qsec, data = mtcars, family = "binomial"): unused argument (data
```

```r
library(glmnet)
model_glmnet <- glmnet(am ~ disp + drat + qsec, data = mtcars,
                       family = "binomial")
```

```
## Error in glmnet(am ~ disp + drat + qsec, data = mtcars, family = "binomial"): unused argument (data
```

```r
model_glmnet <- glmnet(x = as.matrix(mtcars[, c("disp", "drat", "qsec")]),
                       y = mtcars[, "am"],
                       family = "binomial")
```

```r
library(glmnet)
model_glmnet <- glmnet(am ~ disp + drat + qsec, data = mtcars,
                       family = "binomial")
```

```
## Error in glmnet(am ~ disp + drat + qsec, data = mtcars, family = "binomial"): unused argument (data
```

```r
model_glmnet <- glmnet(x = as.matrix(mtcars[, c("disp", "drat", "qsec")]),
                       y = mtcars[, "am"],
                       family = "binomial")
```

```r
model_glm <- glm(x = as.matrix(mtcars[, c("disp", "drat", "qsec")]),
                 y = mtcars[, "am"],
                 family = "binomial")
```

```r
library(glmnet)
model_glmnet <- glmnet(am ~ disp + drat + qsec, data = mtcars,
                       family = "binomial")
```

```
## Error in glmnet(am ~ disp + drat + qsec, data = mtcars, family = "binomial"): unused argument (data
```

```r
model_glmnet <- glmnet(x = as.matrix(mtcars[, c("disp", "drat", "qsec")]),
                       y = mtcars[, "am"],
                       family = "binomial")
```

```r
model_glm <- glm(x = as.matrix(mtcars[, c("disp", "drat", "qsec")]),
                 y = mtcars[, "am"],
                 family = "binomial")
```

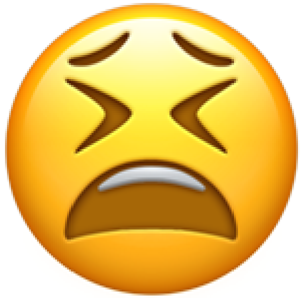
```
## Error in environment(formula): argument "formula" is missing, with no default
```

User-facing problems in modeling in R

- Data must be a matrix (except when it needs to be a data.frame)
- Must use formula or x/y (or both)
- Inconsistent naming of arguments (ntree in randomForest, num.trees in ranger)
- na.omit explicitly or silently
- May or may not accept factors

User-facing problems in modeling in R

- Data must be a matrix (except when it needs to be a data.frame)
- Must use formula or x/y (or both)
- Inconsistent naming of arguments (ntree in randomForest, num.trees in ranger)
- na.omit explicitly or silently
- May or may not accept factors



Syntax for Computing Predicted Class Probabilities

Function	Package	Code
lda	MASS	predict(obj)
glm	stats	predict(obj, type = "response")
gbm	gbm	predict(obj, type = "response", n.trees)
mda	mda	predict(obj, type = "posterior")
rpart	rpart	predict(obj, type = "prob")
Weka	RWeka	predict(obj, type = "probability")
logitboost	LogitBoost	predict(obj, type = "raw", nIter)

blatantly stolen from Max Kuhn

parsnip

www.rstudio.com

The goals of **parsnip** is...

- Decouple the **model classification** from the **computational engine**
- Separate the definition of a model from its evaluation
- Harmonize argument names
- Make consistent predictions (always tibbles with na.omit=FALSE)

```r
model_glm <- glm(am ~ disp + drat + qsec, data = mtcars,
                 family = "binomial")
```

```r
library(parsnip)
model_glm <- logistic_reg(mode = "classification") %>%
  set_engine("glm")

model_glm
```

```
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

```r
library(parsnip)
model_glm <- logistic_reg(mode = "classification") %>%
  set_engine("glm")

model_glm
```

```
## Logistic Regression Model Specification (classification)
##
## Computational engine: glm
```

```r
fit_glm <- model_glm %>%
  fit(factor(am) ~ disp + drat + qsec, data = mtcars)
```

```r
library(parsnip)
model_glmnet <- logistic_reg(mode = "classification") %>%
  set_engine("glmnet")
model_glmnet
```

```
## Logistic Regression Model Specification (classification)
##
## Computational engine: glmnet
```

```r
fit_glmnet <- model_glmnet %>%
  fit(factor(am) ~ disp + drat + qsec, data = mtcars)
```

Using both formula and x/y

Formula

```
fit_glm <- model_glm %>%
  fit(factor(am) ~ ., data = mtcars)
```

x/y

```
fit_glm <- model_glm %>%
  fit_xy(x = as.matrix(mtcars[, c("disp", "drat", "qsec")]),
         y = factor(mtcars[, "am"]),
         data = mtcars)
```

Tidy prediction

```
predict(fit_glm, mtcars)
```

```
## # A tibble: 32 x 1
##    .pred_class
##    <fct>
##  1 1
##  2 1
##  3 1
##  4 0
##  5 0
##  6 0
##  7 0
##  8 0
##  9 0
## 10 0
## # … with 22 more rows
```

Consider now that we wanted to model a more advanded relation ship between variables

```
fit_glm <- model_glm %>%
  fit(factor(am) ~ poly(mpg, 3) + pca(disp:wt)[1] + pca(disp:wt)[2] + pca(disp:wt)[3],
      data = mtcars)
```

Consider now that we wanted to model a more advanded relation ship between variables

```
fit_glm <- model_glm %>%
  fit(factor(am) ~ poly(mpg, 3) + pca(disp:wt)[1] + pca(disp:wt)[2] + pca(disp:wt)[3],
      data = mtcars)
```

- Not all inline functions can be used with formulas
- Having to run some calculations many many times
- Connected to the model, calculations are not saved between models

Post by Max Kuhn about the bad sides of formula
https://rviews.rstudio.com/2017/03/01/the-r-formula-method-the-bad-parts/

recipes

www.rstudio.com

Preprocessing steps

Some of things you may need to deal with before you can start modeling

- Same unit (center and scale)
- Remove correlation (filter and PCA extraction)
- Missing data (imputation)
- Dummy varibles
- Zero Variance

Same units

```
library(recipes)
car_rec <- recipe(mpg ~ ., mtcars) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())
```

PCA

```
library(recipes)
car_rec <- recipe(mpg ~ ., mtcars) %>%
  step_pca(all_predictors(), threshold = 0.8)
```

Any combination of steps

```
car_rec <- recipe(mpg ~ ., mtcars) %>%
  step_knnimpute(drat, wt, neighbors = 5) %>%
  step_zv(all_predictors()) %>%
  step_pca(all_predictors(), threshold = 0.8)
```

```r
library(recipes)
car_rec <- recipe(mpg ~ ., mtcars) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())
```

```r
library(recipes)
car_rec <- recipe(mpg ~ ., mtcars) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())
```

```r
car_rec
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         10
##
## Operations:
##
## Centering for all_predictors()
## Scaling for all_predictors()
```

```r
library(recipes)
car_rec <- recipe(mpg ~ ., mtcars) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

car_preped <- prep(car_rec, training = mtcars)
```

```r
library(recipes)
car_rec <- recipe(mpg ~ ., mtcars) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

car_preped <- prep(car_rec, training = mtcars)
```

```r
bake(car_preped, new_data = mtcars)
```

```r
library(recipes)
car_rec <- recipe(mpg ~ ., mtcars) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

car_preped <- prep(car_rec, training = mtcars)
```

```r
bake(car_preped, new_data = mtcars)
```

```
## # A tibble: 32 x 11
##      mpg    cyl    disp     hp   drat       wt   qsec     vs     am   gear
##    <dbl>  <dbl>   <dbl>  <dbl>  <dbl>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
##  1  21   -0.105 -0.571  -0.535  0.568 -0.610   -0.777 -0.868  1.19   0.424
##  2  21   -0.105 -0.571  -0.535  0.568 -0.350   -0.464 -0.868  1.19   0.424
##  3  22.8 -1.22  -0.990  -0.783  0.474 -0.917    0.426  1.12   1.19   0.424
##  4  21.4 -0.105  0.220  -0.535 -0.966 -0.00230  0.890  1.12  -0.814 -0.932
##  5  18.7  1.01   1.04    0.413 -0.835  0.228   -0.464 -0.868 -0.814 -0.932
##  6  18.1 -0.105 -0.0462 -0.608 -1.56   0.248    1.33   1.12  -0.814 -0.932
##  7  14.3  1.01   1.04    1.43  -0.723  0.361   -1.12  -0.868 -0.814 -0.932
##  8  24.4 -1.22  -0.678  -1.24   0.175 -0.0278   1.20   1.12  -0.814  0.424
##  9  22.8 -1.22  -0.726  -0.754  0.605 -0.0687   2.83   1.12  -0.814  0.424
## 10  19.2 -0.105 -0.509  -0.345  0.605  0.228    0.253  1.12  -0.814  0.424
## # … with 22 more rows, and 1 more variable: carb <dbl>
```

```r
library(recipes)
car_rec <- recipe(mpg ~ ., mtcars) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

car_preped <- prep(car_rec, training = mtcars)
```

```r
juice(car_preped)
```

```
## # A tibble: 32 x 11
##       cyl    disp     hp   drat       wt   qsec     vs     am   gear   carb
##     <dbl>   <dbl>  <dbl>  <dbl>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
##  1 -0.105 -0.571  -0.535  0.568 -0.610   -0.777 -0.868  1.19   0.424  0.735
##  2 -0.105 -0.571  -0.535  0.568 -0.350   -0.464 -0.868  1.19   0.424  0.735
##  3 -1.22  -0.990  -0.783  0.474 -0.917    0.426  1.12   1.19   0.424 -1.12
##  4 -0.105  0.220  -0.535 -0.966 -0.00230  0.890  1.12  -0.814 -0.932 -1.12
##  5  1.01   1.04    0.413 -0.835  0.228   -0.464 -0.868 -0.814 -0.932 -0.503
##  6 -0.105 -0.0462 -0.608 -1.56   0.248    1.33   1.12  -0.814 -0.932 -1.12
##  7  1.01   1.04    1.43  -0.723  0.361   -1.12  -0.868 -0.814 -0.932  0.735
##  8 -1.22  -0.678  -1.24   0.175 -0.0278   1.20   1.12  -0.814  0.424 -0.503
##  9 -1.22  -0.726  -0.754  0.605 -0.0687   2.83   1.12  -0.814  0.424 -0.503
## 10 -0.105 -0.509  -0.345  0.605  0.228    0.253  1.12  -0.814  0.424  0.735
## # … with 22 more rows, and 1 more variable: mpg <dbl>
```

```
recipe -> prepare -> bake/juice

(define) -> (estimate) -> (apply)
```

Types of data splitting

- Random
- By date
- By outcome
    - Classification: within class
    - regression: within quantile

Training and Testing sets

```
library(rsample)

car_preped <- prep(car_rec, training = mtcars)
```

rsample

www.rstudio.com

Training and Testing sets

```
library(rsample)

set.seed(4595)

# These slides were almost finished and I didn't want to change the data in all the other slides
big_mtcars <- rerun(10, mtcars) %>%
  bind_rows()

data_split <- initial_split(big_mtcars, strata = "mpg", p = 0.80)

# Training and test data
cars_train <- training(data_split)
cars_test  <- testing(data_split)

car_prep <- prep(car_rec, training = cars_train)

# Preprocessed data
cars_train_p <- juice(car_prep)
cars_test_p <-bake(car_prep, new_data = cars_test)
```

Cross-Validating (sneak peak)

```
set.seed(1234)
cv_splits <- vfold_cv(
  data = big_mtcars,
  v = 10,
  strata = "mpg"
)

cv_splits
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 2
##    splits           id
##    <list>           <chr>
##  1 <split [288/32]> Fold01
##  2 <split [288/32]> Fold02
##  3 <split [288/32]> Fold03
##  4 <split [288/32]> Fold04
##  5 <split [288/32]> Fold05
##  6 <split [288/32]> Fold06
##  7 <split [288/32]> Fold07
##  8 <split [288/32]> Fold08
##  9 <split [288/32]> Fold09
## 10 <split [288/32]> Fold10
```

```r
car_form <- mpg ~ disp + qsec + cyl
# Fit on a single analysis resample
fit_model <- function(split, spec) {
  fit(
    object = nearest_neighbor() %>% set_engine("kknn"),
    formula = car_form,
    data = analysis(split) # <- pull out training set
  )
}
# For each resample, call fit_model()
cv_splits <- cv_splits %>%
  mutate(models_knn = map(splits, fit_model, spec_lm),
         )
cv_splits
```

```
## #  10-fold cross-validation using stratification
## # A tibble: 10 x 3
##    splits            id     models_knn
##  * <list>            <chr>  <list>
##  1 <split [288/32]> Fold01 <fit[+]>
##  2 <split [288/32]> Fold02 <fit[+]>
##  3 <split [288/32]> Fold03 <fit[+]>
##  4 <split [288/32]> Fold04 <fit[+]>
##  5 <split [288/32]> Fold05 <fit[+]>
##  6 <split [288/32]> Fold06 <fit[+]>
##  7 <split [288/32]> Fold07 <fit[+]>
##  8 <split [288/32]> Fold08 <fit[+]>
##  9 <split [288/32]> Fold09 <fit[+]>
## 10 <split [288/32]> Fold10 <fit[+]>
```

yardstick

www.rstudio.com

```r
library(yardstick)
head(two_class_example)
```

```
##    truth      Class1       Class2 predicted
## 1 Class2 0.003589243 0.9964107574    Class2
## 2 Class1 0.678621054 0.3213789460    Class1
## 3 Class2 0.110893522 0.8891064779    Class2
## 4 Class1 0.735161703 0.2648382969    Class1
## 5 Class2 0.016239960 0.9837600397    Class2
## 6 Class1 0.999275071 0.0007249286    Class1
```

```r
library(yardstick)
head(two_class_example)
```

```
##    truth      Class1       Class2 predicted
## 1 Class2 0.003589243 0.9964107574    Class2
## 2 Class1 0.678621054 0.3213789460    Class1
## 3 Class2 0.110893522 0.8891064779    Class2
## 4 Class1 0.735161703 0.2648382969    Class1
## 5 Class2 0.016239960 0.9837600397    Class2
## 6 Class1 0.999275071 0.0007249286    Class1
```

```r
metrics(two_class_example, truth = truth, estimate = predicted)
```

```
## # A tibble: 2 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.838
## 2 kap      binary         0.675
```

```r
library(yardstick)
head(two_class_example)
```

```
##    truth      Class1       Class2 predicted
## 1 Class2 0.003589243 0.9964107574    Class2
## 2 Class1 0.678621054 0.3213789460    Class1
## 3 Class2 0.110893522 0.8891064779    Class2
## 4 Class1 0.735161703 0.2648382969    Class1
## 5 Class2 0.016239960 0.9837600397    Class2
## 6 Class1 0.999275071 0.0007249286    Class1
```

```r
accuracy(two_class_example, truth = truth, estimate = predicted)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.838
```

```r
library(yardstick)
head(two_class_example)
```

```
##    truth      Class1       Class2 predicted
## 1 Class2 0.003589243 0.9964107574    Class2
## 2 Class1 0.678621054 0.3213789460    Class1
## 3 Class2 0.110893522 0.8891064779    Class2
## 4 Class1 0.735161703 0.2648382969    Class1
## 5 Class2 0.016239960 0.9837600397    Class2
## 6 Class1 0.999275071 0.0007249286    Class1
```

```r
j_index(two_class_example, truth = truth, estimate = predicted)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 j_index binary         0.673
```

And many more!!

```r
library(yardstick)
head(two_class_example)
```

```
##    truth      Class1       Class2 predicted
## 1 Class2 0.003589243 0.9964107574    Class2
## 2 Class1 0.678621054 0.3213789460    Class1
## 3 Class2 0.110893522 0.8891064779    Class2
## 4 Class1 0.735161703 0.2648382969    Class1
## 5 Class2 0.016239960 0.9837600397    Class2
## 6 Class1 0.999275071 0.0007249286    Class1
```

```r
conf_mat(two_class_example, truth = truth, estimate = predicted)
```

```
##           Truth
## Prediction Class1 Class2
##     Class1    227     50
##     Class2     31    192
```

```
roc_curve(two_class_example, truth = truth, Class1) %>%
  autoplot()
```