Elements of Statistical Learning Practice

Brittney Bailey

29 December, 2020

1 Independent outcomes

1.1 From `tidymodels`: *Get Started*

1.1.1 Loading packages and data

The `tidymodels` package relies mostly on `parsnip` package for functions to specify and train models with different "engines."

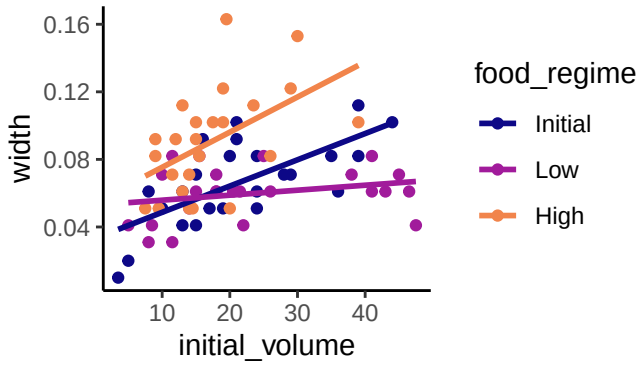
```
library(tidymodels) #parsnip

# helper packages
library(readr) # import data
library(broom.mixed) # convert bayesiam models to tidy tibbles
library(dotwhisker) # visualizing regression results
```

> use the data from Constable (1993) to explore how three different feeding regimes affect the size of sea urchins over time. The initial size of the sea urchins at the beginning of the experiment probably affects how big they grow as they are fed.

The urchins dataset only has three variables: food regime (Initial, Low, High), initial volume (ml), and final suture width (mm, probably?).

```
urchins <- read_csv("https://tidymodels.org/start/models/urchins.csv") %>%
  # Change the names to be a little more verbose
  setNames(c("food_regime", "initial_volume", "width")) %>%
  # Factors are very helpful for modeling, so we convert one column
  mutate(food_regime = factor(food_regime, levels = c("Initial", "Low", "High")))
```

```
ggplot(urchins,
       aes(x = initial_volume,
           y = width,
           group = food_regime,
           col = food_regime)) +
  geom_point() +
  geom_smooth(method = lm, se = FALSE) +
  scale_color_viridis_d(option = "plasma", end = .7)
```




```
#> `geom_smooth()` using formula 'y ~ x'
```

1.1.2 Model fitting

1. `_type_(args)`: Use `parsnip` package to specify *type* of model (`boost_tree`, `decision_tree()`, `linear_reg()`, `logistic_reg()`, `mars()`, `mlp()`, `multinom()`, `nearest_neighbor()`, `null_model()`, `rand_forest()`, `surv_reg()`, `svm_poly()`, `svm_rbf()`)
   - `rand_forest`: `mtry` = number of predictors, `trees` = number of trees, `min_n` = min number of points in a node
   - `logistic_reg`/`linear_reg`: `penalty` = total amount of regularization for glmnet, keras, spark, `mixture` = mixture amounts of different types of regularization
   - `decision_tree`: `cost_complexity` = the cost/complexity or C_p used by CART models in `rpart`, `tree_depth` = maximum depth of tree in rpart and spark, `min_n` = min number of points in a node
2. `set_mode()`: Specify the *mode* of the model (e.g., classification, regression, censored regression, risk regression, clustering)
   - `rand_forest`: "unknown", "regression", "classification"
   - `logistic_reg`: "classification"
   - `linear_reg`: "regression"
   - `decision_tree`: "unknown", "regression", "classification"
3. `set_engine()`: Specify the *engine* for the model, which is the package or external method used to fit the model (can use `shown_engines("_TYPE_")` to see current set)
   - `rand_forest`: "ranger", "randomForest", "spark" (use `fit_xy()`)
   - `logistic_reg`/`linear_reg`: "glm"/"lm", "glmnet", "stan", "spark", "keras"
   - `decision_tree`: "rpart", "C5.0", "spark"
4. `fit(y ~ x1 + x2, data = _DATA_)`: fit prescribed model

```
# specify model
lm_spec <- linear_reg() %>%
  set_mode("regression") %>% # default, can be deleted
  set_engine("lm")

# fit model
lm_fit <-  lm_spec %>%
  fit(width ~ initial_volume * food_regime, data = urchins)

# summarize model
summary(lm_fit)
```

```
             Length Class      Mode
    lvl        0    -none-     NULL
    spec       5    linear_reg list
    fit       13    lm         list
```

```
preproc  1     -none-     list
elapsed  5     proc_time  numeric
```

```
tidy(lm_fit)
```

```
# A tibble: 6 x 5
  term                           estimate std.error statistic  p.value
  <chr>                             <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)                     0.0331    0.00962      3.44  0.00100
2 initial_volume                  0.00155   0.000398     3.91  0.000222
3 food_regimeLow                  0.0198    0.0130       1.52  0.133
4 food_regimeHigh                 0.0214    0.0145       1.47  0.145
5 initial_volume:food_regimeLow  -0.00126   0.000510    -2.47  0.0162
6 initial_volume:food_regimeHigh  0.000525  0.000702     0.748 0.457
```

1.2 From ISLR Lab 8.3: Decision Trees (p. 323)

```
# load packages
library(tree)
library(ISLR)
```

```
# based on http://www.rebeccabarter.com/blog/2020-03-25_machine_learning/
library(tidymodels)
library(workflows)
library(tune)
library(modeldata)
# from tidymodels
library(readr)
library(broom.mixed)
library(dotwhisker)
```

1.2.1 Carseats dataset

The *Carseats* dataset has 400 stores of where 11 variables were measured on car seat sales: 3 categorical and 8 quantitative. We want to predict whether the sales are high, meaning greater than 8 thousand at a store.

We start by creating the `highsales` binary variable.

```
# create binary sales variable (make sure it's a factor)
carseats <- mutate(Carseats,
                   highsales = factor(ifelse(Sales <= 8, "no", "yes")))
```

```
# split data
set.seed(52)
# run splitting function
carseats.split <- rsample::initial_split(carseats, prop = 0.75)

# get training set
carseats.train <- training(carseats.split)

# get test set
carseats.test <- testing(carseats.split)

# get cross-validated version of training set
carseats.cv <- vfold_cv(carseats.train)
```

We can then fit a single classification tree using `tree()` in the `tree` package:

```
# fit classification tree
tree.carseats <- tree(highsales~ . - Sales, data = carseats)
summary(tree.carseats)
```

```
Classification tree:
tree(formula = highsales ~ . - Sales, data = carseats)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
[6] "Advertising" "Age"         "US"
Number of terminal nodes:  27
Residual mean deviance:  0.4575 = 170.7 / 373
Misclassification error rate: 0.09 = 36 / 400
```

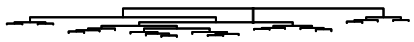
Note the following:

- `.-Sales` means *regress on everything except the Sales variable*;
- the default splitting method is based on the *deviance* (alternative: `gini`), $-2\sum_m \sum_k n_{mk} \log(\hat{p}_{mk})$ where n_{mk} is the number of observations in the mth terminal node that belong to class k; small deviance is good; and
- the misclassification error rate provided is the *training* error rate
- the residual mean deviance is the deviance divided by $n - |T_0|$

```
# specify the model inputs (and any preprocessing)
carseats.recipe <- recipe(highsales ~ . , data = carseats) %>%
  step_rm(Sales)

# prep data
carseats.prepped <- prep(carseats.recipe, carseats) %>%
  juice()

# specify the algorithm
carseats.tree <-
  decision_tree() %>%
  set_engine("rpart") %>% # cost_complexity (cp)
  set_mode("classification")

# put model and recipes together into workflow
carseats.workflow <-
  workflow() %>%
  add_model(carseats.tree)

# tune parameters if needed (not needed here)

# finalize workflow after tuning parameters (if needed)

# get training model
carseats.tree.fit <- last_fit(carseats.workflow, carseats.split)
```

We can display the tree using `plot()`

```
plot(tree.carseats)
```



2 Correlated binary outcomes

2.1 Depression data

```
data("DepressionDemo")
```

Simulated dataset of a randomized clinical trial (N = 150) to illustrate fitting of (G)LMM trees.

A data frame containing 150 observations on 6 variables:

`depression` numeric. Continuous treatment outcome variable (range: 3-16, M = 9.12, SD = 2.66).

`treatment` factor. Binary treatment variable.

`cluster` factor. Indicator for cluster with 10 levels.

`age` numeric. Continuous partitioning variable (range: 18-69, M = 45, SD = 9.56).

`anxiety` numeric. Continuous partitioning variable (range: 3-18, M = 10.26, SD = 3.05).

`duration` numeric. Continuous partitioning variable (range: 1-17, M = 6.97, SD = 2.90).

`depression_bin` factor. Binarized treatment outcome variable (0 = recovered, 1 = not recovered).

The data were generated such that the duration and anxiety covariates characterized three subgroups with differences in treatment effects. The cluster variable was used to introduce a random intercept that should be accounted for. The treatment outcome is an index of depressive symptomatology.