A recipe for recipes

eRum 2018 @ Budapest

Edwin Thoen

May 15, 2018

The `recipes` package

Kuhn & Wickham 2017

"Preprocessing Tools to Create Design Matrices"

Define the steps you'll take to go from raw data to the analysis set.

Store these steps into a procedure (a recipe), and apply it on new data.

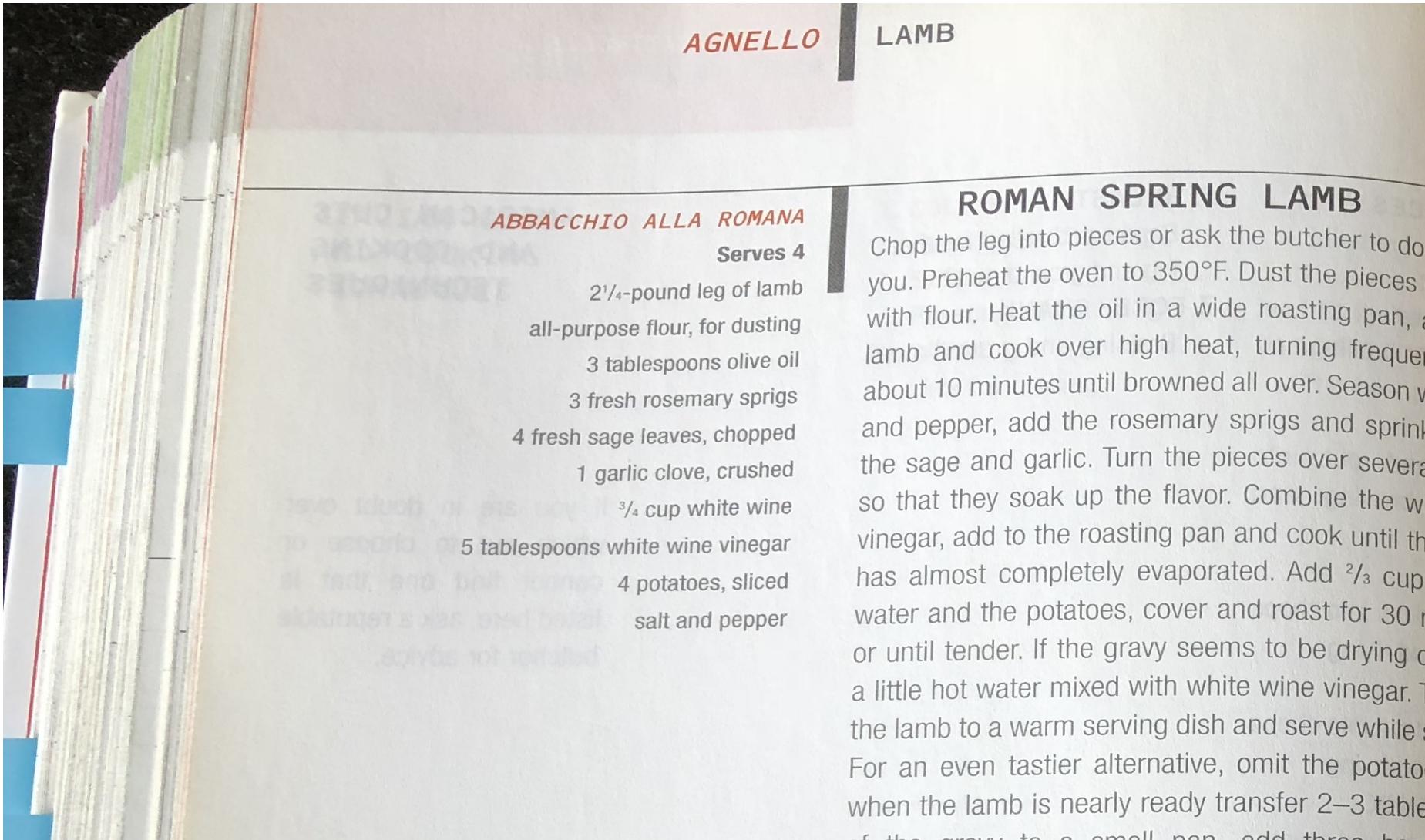
You focus on the best processing steps, `recipes` does the bookkeeping.

What to expect?

1. A short introduction of `recipes`
2. The recipe for `recipes`
3. Example(s) in which the recipe for `recipes` is applied

The `recipes` package

A `recipe` is the specification of an intent, separate the planning from the doing.



Creating a recipe

```
train_set <- mtcars[1:20, c("am", "disp", "hp")]
test_set  <- mtcars[21:32, c("am", "disp", "hp")]
rec <- recipe(train_set, am ~ .)
```

We have defined the roles here, `am` is the outcome, `disp` and `hp` are the predictors.

Adding steps to the recipe

```
rec_with_steps <- rec %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())
```

This is specifying the intent, we didn't do anything on the data yet.

Preparing the recipe with prep

prep acquires all the necesarry information on the training set.

```
(rec_prepped <- rec_with_steps %>% prep())
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##    outcome           1
##  predictor           2
##
## Training data contained 20 data points and no missing da
##
## Operations:
##
## Centering for disp, hp [trained]
## Scaling for disp, hp [trained]
```

Preparing the data with bake

```
train_final <- bake(rec_prepped, train_set)
test_final  <- bake(rec_prepped, test_set)
```

The statistics to center and scale are learned on the `train_set` and applied to the `test_set`.

```
head(test_final)
```

```
## # A tibble: 6 x 3
##      am        disp          hp
##   <dbl>       <dbl>       <dbl>
## 1     0 -0.8833893 -0.6534422
## 2     0  0.6524338  0.2300383
## 3     0  0.5437853  0.2300383
## 4     0  0.9007730  1.8136356
## 5     0  1.2888031  0.6467745
## 6     1 -1.2023500 -1.1701950
```

Get information with `tidy`

Gives information about the steps in a data frame.

```
tidy(rec_prepped)
```

```
## # A tibble: 2 x 5
##   number operation   type trained  skip
##    <int>     <chr>  <chr>   <lgl> <lgl>
## 1      1      step center    TRUE FALSE
## 2      2      step  scale    TRUE FALSE
```

Why am I talking about `recipes`?

Added the `check` framework together with Max.

A `check` does not change the data in any way, it tests assumptions and will break `bake` if these are not met.

```
rec2 <- recipe(train_set) %>% check_missing(everything()) %
test_set[1, 1] <- NA
train_baked <- bake(rec2, train_set)
test_baked  <- bake(rec2, test_set)
```

```
## Error: The following columns contain missing values: `an
```

Building youw own steps and checks

Fully leverage package structure.

For your own preparations and to contribute to the package.

Challenge, delve a little deeper into the package inner workings.

S3 classes in `recipes`

A recipe itself is of class `recipe`.

All the `steps` and `checks` available have their own subclass. Each with their own `prep` and `bake` functions.

The recipe gathers all the objects of different subclasses.

`prep.recipe` and `bake.recipe` call the `prep` and `bake` methods of its steps and checks.

Create a custom `step` or `check`

A full step or check comprises:

- the function that is called to add to the recipe
- constructor to create new objects of the subclass
- `prep` method
- `bake` method
- `print` method
- `tidy` method

A recipe for `recipes`

My preferred way to create a new `step` or `check`:

1. Write a function that does the baking action. Don't bother about the `recipes` package yet.
2. Recognize which arguments need to be provided upfront and which are estimated by `prep`.
3. Write the constructor.
4. Write the `<step_name>_step` or `<check_name>_check` function.
5. Write the `prep` method.
6. Write the `bake` method.
7. Write the `print` method.
8. Write the `tidy` method.

Resources

Within `recipes` you'll find a number of helper functions.

Clone the source code from https://github.com/topepo/recipes to access them.

On https://github.com/EdwinTh/recipe_for_recipes you will find a skeleton for new steps.

Example: A range check

Assure that the range of a numeric variable in a new set is approximately equal to the range of the variable in the train set.

Throw informative error when on one or both ends the new variable exceeds the original range plus some slack.

1) the function for this is:

```
range_check_func <- function(x,
                             lower,
                             upper,
                             slack_prop = 0.05,
                             colname = "x") {
  min_x <- min(x); max_x <- max(x); slack <- (upper - lower
  if (min_x < (lower - slack) & max_x > (upper + slack)) {
    stop("min ", colname, " is ", min_x, ", lower bound is
         "\n", "max x is ", max_x, ", upper bound is ", upp
         call. = FALSE)
  } else if (min_x < (lower - slack)) {
    stop("min ", colname, " is ", min_x, ", lower bound is
         call. = FALSE)
  } else if (max_x > (upper + slack)) {
    stop("max ", colname, " is ", max_x, ", upper bound is
         call. = FALSE)
  }
}
```

2) thinking about the arguments

`slack_prop` is an argument provided by the user.

`lower` and `upper` should be calculated by the `prep.check_range` method.

3) the constructor

```
check_range_new <-
  function(terms = NULL,
           role  = NA,
           trained = FALSE,
           lower   = NULL,
           upper   = NULL,
           slack_prop = NULL) {
    check(subclass = "range",
          terms    = terms,
          role     = role,
          trained  = trained,
          lower    = lower,
          upper    = upper,
          slack_prop = slack_prop)
  }
```

4) the function to add it to the recipe

```
check_range <-
  function(recipe,
           ...,
           role = NA,
           trained = FALSE,
           lower   = NULL,
           upper   = NULL,
           slack_prop = 0.05) {
    add_check(
      recipe,
      check_range_new(
        terms   = ellipse_check(...),
        role    = role,
        trained = trained,
        lower   = lower,
        upper   = upper,
        slack_prop = slack_prop
      )
```

5) the prep method

```
prep.check_range <-
  function(x,
           training,
           info = NULL,
           ...) {
    col_names <- terms_select(x$terms, info = info)
    lower_vals <- vapply(training[ ,col_names], min, c(min
                         na.rm = TRUE)
    upper_vals <- vapply(training[ ,col_names], max, c(max
                         na.rm = TRUE)
    check_range_new(
      x$terms,
      role = x$role,
      trained = TRUE,
      lower   = lower_vals,
      upper   = upper_vals,
      slack_prop = x$slack_prop
    )
```

6) the bake method

```
bake.check_range <- function(object,
                             newdata,
                             ...) {
  col_names <- names(object$lower)
  for (i in seq_along(col_names)) {
    colname <- col_names[i]
    range_check_func(newdata[[ colname ]],
                     object$lower[colname],
                     object$upper[colname],
                     object$slack_prop,
                     colname)
  }
  as_tibble(newdata)
}
```

7) the print method

```
print.check_range <-
  function(x, width = max(20, options()$width - 30), ...) {
    cat("Checking range of ", sep = "")
    printer(names(x$lower), x$terms, x$trained, width = wid
    invisible(x)
  }
```

8) the tidy method

```
tidy.check_range <- function(x, ...) {
  if (is_trained(x)) {
    res <- tibble(terms = x$columns)
  } else {
    res <- tibble(terms = sel2char(x$terms))
  }
  res
}
```

Put it in practise

```
df1 <- data_frame(x = -1:1)
df2 <- data_frame(x = -2:2)
recipe(df1) %>% check_range(x) %>% prep() %>% bake(df2)
```

```
## Error: min x is -2, lower bound is -1.1
## max x is 2, upper bound is 1.1
```

Resources

Slides and the skeleton can be found here:

https://github.com/EdwinTh/recipe_for_recipes

The source code for recipes is maintained here:

https://github.com/topepo/recipes/

Thorough introduction by Max Kuhn to the package:

https://www.rstudio.com/resources/webinars/
creating-and-preprocessing-a-design-matrix-with-recipes/

Thank you!

edwinthoen@gmail.com

@edwin_thoen

github.com/EdwinTh