Recipe Preparation and Cook Time Calculator

## Primary Aim

The aim of this project is to create a function that calculates the time of preparation and cooking based on the ingredients that the user will enter. The interface will be an RShiny app that prompts the use for the ingredients. The app will then estimate preparation and cooking time.

I will use the data set below to find the relationship between ingredients and preparation time an cooking time. Once this relationship has been modeled, the app will be able to calculate the prep and cook time for the ingredients the user enters.

I will use tidytext to analyze the recipe data. Then I will use tidymodels to get a mathematical model of the relationships between the ingredients listed and the outcome times.

The user interface will be written using RShiny.

I will use the data described below to fit a model predicting preparation and cooking time. The package tidytext will be used to create the features. The package tidymodels will be used to create the model and fit the model on the ingredients the user enters.

### The Data

The data is consisting of 255 recipes from India which each recipes contains some variables which are (name of the Recipes,Ingredients,Diet,Preparation time, cooking time,Flavor, Course,State(the origin of the food),region(location)).

## Proof of Concept

The steps to complete this analysis require:

* Find a training data set (indian\_food.csv)
* Explore the training data set to find good features and descriptions (given below)
* Use tidytext to create features from the ingredients listed in the training data set.
* Use the features created and tidymodels to fit a model.
* test the model
* write an RShiny interface that askes the user for a list of ingredient
* Take the ingredients given and then use the fitted model to calculate the prep and cooking time for that recepe.

### Data Exploration

This is the training data set.

d <- read.csv("indian\_food.csv")  
d %>% head

## name  
## 1 Balu shahi  
## 2 Boondi  
## 3 Gajar ka halwa  
## 4 Ghevar  
## 5 Gulab jamun  
## 6 Imarti  
## ingredients  
## 1 Maida flour, yogurt, oil, sugar  
## 2 Gram flour, ghee, sugar  
## 3 Carrots, milk, sugar, ghee, cashews, raisins  
## 4 Flour, ghee, kewra, milk, clarified butter, sugar, almonds, pistachio, saffron, green cardamom  
## 5 Milk powder, plain flour, baking powder, ghee, milk, sugar, water, rose water  
## 6 Sugar syrup, lentil flour  
## diet prep\_time cook\_time flavor\_profile course state region  
## 1 vegetarian 45 25 sweet dessert West Bengal East  
## 2 vegetarian 80 30 sweet dessert Rajasthan West  
## 3 vegetarian 15 60 sweet dessert Punjab North  
## 4 vegetarian 15 30 sweet dessert Rajasthan West  
## 5 vegetarian 15 40 sweet dessert West Bengal East  
## 6 vegetarian 10 50 sweet dessert West Bengal East

Summary of the data. what’s in the data set.

d %>%   
 select(-name, -ingredients) %>%   
 tbl\_summary()

## Table printed with {flextable}, not {gt}. Learn why at  
## http://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html  
## To suppress this message, include `message = FALSE` in the code chunk header.

| Characteristic | N = 2551 |
| --- | --- |
| diet |  |
| non vegetarian | 29 (11%) |
| vegetarian | 226 (89%) |
| prep\_time | 10 (10, 20) |
| cook\_time | 30 (20, 40) |
| flavor\_profile |  |
| -1 | 29 (11%) |
| bitter | 4 (1.6%) |
| sour | 1 (0.4%) |
| spicy | 133 (52%) |
| sweet | 88 (35%) |
| course |  |
| dessert | 85 (33%) |
| main course | 129 (51%) |
| snack | 39 (15%) |
| starter | 2 (0.8%) |
| state |  |
| -1 | 24 (9.4%) |
| Andhra Pradesh | 10 (3.9%) |
| Assam | 21 (8.2%) |
| Bihar | 3 (1.2%) |
| Chhattisgarh | 1 (0.4%) |
| Goa | 3 (1.2%) |
| Gujarat | 35 (14%) |
| Haryana | 1 (0.4%) |
| Jammu & Kashmir | 2 (0.8%) |
| Karnataka | 6 (2.4%) |
| Kerala | 8 (3.1%) |
| Madhya Pradesh | 2 (0.8%) |
| Maharashtra | 30 (12%) |
| Manipur | 2 (0.8%) |
| Nagaland | 1 (0.4%) |
| NCT of Delhi | 1 (0.4%) |
| Odisha | 7 (2.7%) |
| Punjab | 32 (13%) |
| Rajasthan | 6 (2.4%) |
| Tamil Nadu | 20 (7.8%) |
| Telangana | 5 (2.0%) |
| Tripura | 1 (0.4%) |
| Uttar Pradesh | 9 (3.5%) |
| Uttarakhand | 1 (0.4%) |
| West Bengal | 24 (9.4%) |
| region |  |
|  | 1 (0.4%) |
| -1 | 13 (5.1%) |
| Central | 3 (1.2%) |
| East | 31 (12%) |
| North | 49 (19%) |
| North East | 25 (9.8%) |
| South | 59 (23%) |
| West | 74 (29%) |
| 1Statistics presented: n (%); median (IQR) | |

count the NO. of recopies per state

d %>%   
 count(state) %>%   
 arrange(desc(n))

## state n  
## 1 Gujarat 35  
## 2 Punjab 32  
## 3 Maharashtra 30  
## 4 -1 24  
## 5 West Bengal 24  
## 6 Assam 21  
## 7 Tamil Nadu 20  
## 8 Andhra Pradesh 10  
## 9 Uttar Pradesh 9  
## 10 Kerala 8  
## 11 Odisha 7  
## 12 Karnataka 6  
## 13 Rajasthan 6  
## 14 Telangana 5  
## 15 Bihar 3  
## 16 Goa 3  
## 17 Jammu & Kashmir 2  
## 18 Madhya Pradesh 2  
## 19 Manipur 2  
## 20 Chhattisgarh 1  
## 21 Haryana 1  
## 22 Nagaland 1  
## 23 NCT of Delhi 1  
## 24 Tripura 1  
## 25 Uttarakhand 1

plot the relationship btw perp time and cook time by course

d %>%   
 ggplot(aes(x=prep\_time, y = cook\_time))+  
 geom\_point()+  
 geom\_smooth()+  
 facet\_wrap(~course)

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : span too small. fewer data values than degrees of freedom.

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : at 119.4

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : radius 0.36

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : all data on boundary of neighborhood. make span bigger

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : pseudoinverse used at 119.4

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : neighborhood radius 0.6

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : reciprocal condition number 1

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : at 240.6

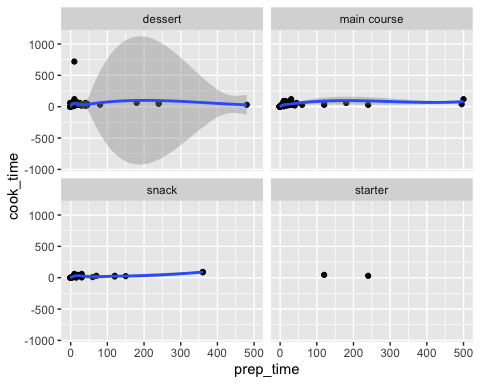
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : radius 0.36

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : all data on boundary of neighborhood. make span bigger

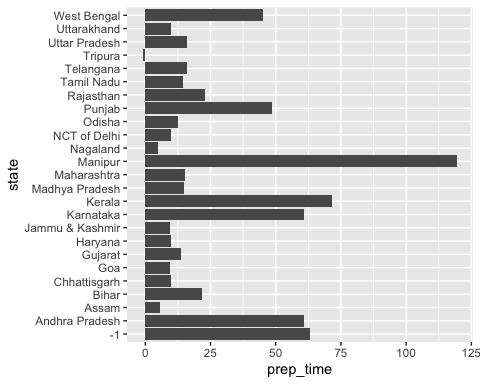
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : There are other near singularities as well. 0.36

## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : zero-width neighborhood. make span bigger  
  
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =  
## parametric, : zero-width neighborhood. make span bigger

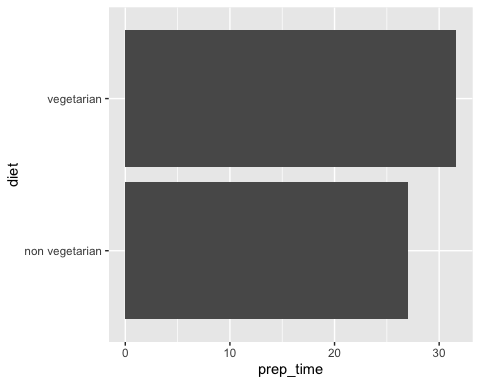
## Warning: Computation failed in `stat\_smooth()`:  
## NA/NaN/Inf in foreign function call (arg 5)

 the average of prep time by state

d %>%   
   
 ggplot(aes(x = prep\_time, y = state))+  
 geom\_bar(stat = 'summary', fun = mean)

 average prep time by diet

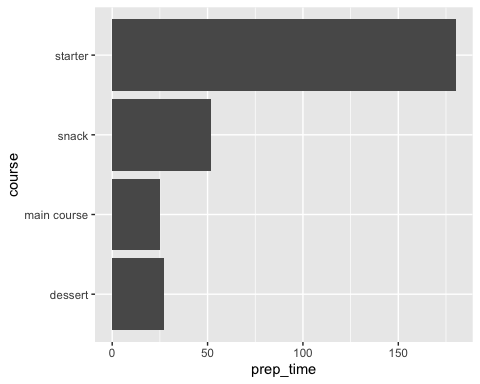
d %>%   
 ggplot(aes(x = prep\_time, y = diet))+  
 geom\_bar(stat = "summary", fun = mean)



d %>%   
 ggplot(aes(x = prep\_time, y = course))+  
 geom\_bar(stat = "summary", fun.y = "mean")

## Warning: Ignoring unknown parameters: fun.y

## No summary function supplied, defaulting to `mean\_se()`



### Feature Extraction

Start of feature extraction froom the ingredients list. This shows how individual ingredients can be extracted to features.

d %>%   
 select(name,prep\_time, cook\_time, ingredients) %>%   
 mutate(ind = 1) %>%   
 separate\_rows(ingredients, sep = ',') %>%   
 pivot\_wider(names\_from = 'ingredients', values\_from = "ind") %>%   
 mutate(across(is\_double, ~if\_else(is.na(.x),0,1)))

## Warning: Problem with `mutate()` input `..1`.  
## ℹ Predicate functions must be wrapped in `where()`.  
##   
## # Bad  
## data %>% select(is\_double)  
##   
## # Good  
## data %>% select(where(is\_double))  
##   
## ℹ Please update your code.  
## This message is displayed once per session.  
## ℹ Input `..1` is `across(is\_double, ~if\_else(is.na(.x), 0, 1))`.

## Warning: Predicate functions must be wrapped in `where()`.  
##   
## # Bad  
## data %>% select(is\_double)  
##   
## # Good  
## data %>% select(where(is\_double))  
##   
## ℹ Please update your code.  
## This message is displayed once per session.

## # A tibble: 255 x 429  
## name prep\_time cook\_time `Maida flour` ` yogurt` ` oil` ` sugar`  
## <chr> <int> <int> <dbl> <dbl> <dbl> <dbl>  
## 1 Balu… 45 25 1 1 1 1  
## 2 Boon… 80 30 0 0 0 1  
## 3 Gaja… 15 60 0 0 0 1  
## 4 Ghev… 15 30 0 0 0 1  
## 5 Gula… 15 40 0 0 0 1  
## 6 Imar… 10 50 0 0 0 0  
## 7 Jale… 10 50 0 0 0 0  
## 8 Kaju… 10 20 0 0 0 1  
## 9 Kala… 20 30 0 0 0 1  
## 10 Kheer 10 40 0 0 0 1  
## # … with 245 more rows, and 422 more variables: `Gram flour` <dbl>, `  
## # ghee` <dbl>, Carrots <dbl>, ` milk` <dbl>, ` cashews` <dbl>, `  
## # raisins` <dbl>, Flour <dbl>, ` kewra` <dbl>, ` clarified butter` <dbl>, `  
## # almonds` <dbl>, ` pistachio` <dbl>, ` saffron` <dbl>, ` green  
## # cardamom` <dbl>, `Milk powder` <dbl>, ` plain flour` <dbl>, ` baking  
## # powder` <dbl>, ` water` <dbl>, ` rose water` <dbl>, `Sugar syrup` <dbl>, `  
## # lentil flour` <dbl>, Maida <dbl>, ` corn flour` <dbl>, ` baking  
## # soda` <dbl>, ` vinegar` <dbl>, ` curd` <dbl>, ` turmeric` <dbl>, `  
## # cardamom` <dbl>, Cashews <dbl>, Milk <dbl>, ` cottage cheese` <dbl>, `  
## # rice` <dbl>, ` dried fruits` <dbl>, Yogurt <dbl>, ` nuts` <dbl>, `Refined  
## # flour` <dbl>, ` besan` <dbl>, ` powdered sugar` <dbl>, ` yoghurt` <dbl>,  
## # `Firm white pumpkin` <dbl>, ` kitchen lime` <dbl>, ` alum powder` <dbl>,  
## # Rice <dbl>, `Condensed milk` <dbl>, ` spices` <dbl>, Semolina <dbl>,  
## # Khoa <dbl>, ` coconut` <dbl>, ` molu leaf` <dbl>, `Corn flour` <dbl>, ` dry  
## # fruits` <dbl>, Chhena <dbl>, Sugar <dbl>, ` chenna cheese` <dbl>, `  
## # cream` <dbl>, ` lemon juice` <dbl>, ` coconut flakes` <dbl>, Chenna <dbl>,  
## # ` condensed milk` <dbl>, ` fried milk power` <dbl>, ` sugar syrup` <dbl>,  
## # Yoghurt <dbl>, ` refined flour` <dbl>, ` fennel seeds` <dbl>, `Besan  
## # flour` <dbl>, ` jaggery` <dbl>, ` flour` <dbl>, `Rice flour` <dbl>, ` wheat  
## # flour` <dbl>, ` sweetened milk` <dbl>, ` reduced milk` <dbl>, ` vegetable  
## # oil` <dbl>, ` elachi` <dbl>, Besan <dbl>, ` cardamom powder` <dbl>, `  
## # cashews and raisins` <dbl>, ` jaggery syrup` <dbl>, Peanuts <dbl>, `  
## # Sugar` <dbl>, ` Dharwadi buffalo milk` <dbl>, `Loaf bread` <dbl>, `  
## # salt` <dbl>, ` semolina` <dbl>, `Wheat flour` <dbl>, `Black lentils` <dbl>,  
## # ` mung bean` <dbl>, ` skimmed milk powder` <dbl>, ` chickpeas` <dbl>,  
## # `Chana dal` <dbl>, Apricots <dbl>, `Vermicelli pudding` <dbl>, `  
## # banana` <dbl>, ` khus-khus seeds` <dbl>, Cucumber <dbl>, ` rava` <dbl>, `  
## # jaggery ` <dbl>, ` maida flour` <dbl>, Curd <dbl>, Fish <dbl>, `  
## # potol` <dbl>, ` tomato` <dbl>, …

## Challenges and To-Do:

* in some point I will have to create more data sets of recipes to make the modal better
* what I will do if the user enters ingredient that are not in the data set.
* what other info besides ingredient should the user require to enter
* To fix the error that I will encounter in my code.
* Finding online cook book with more recipes to add them in my training data.

## To Do

* add data set (10\14) *extract features using tidytext (10\21)* Fit the models using tidymodals(10\28) *create Rshiny interface(11\5)* users testing the package (11\10)
* publish the package in the internet (11\13)