tidymodels

EST0133 - Introdução à Modelagem de Big Data

Marcus Nunes
https://introbigdata.org/
https://marcusnunes.me/

Universidade Federal do Rio Grande do Norte

Introdução

Introdução

- O pacote `tidymodels` irá ajudar a organizar nosso fluxo de trabalho
- De modo geral, cada ferramenta que veremos daqui em diante está implementada em um pacote diferente
- Com o `tidymodels` podemos usar uma sintaxe similar para todas essas ferramentas, focando menos em aprender como cada uma é utilizada, e nos dedicando mais a interpretar os resultados obtidos

tidymodels

tidymodels

- O `tidymodels` não é um pacote em si
- Assim como o `tidyverse`, o `tidymodels` é uma coleção de pacotes com aplicações específicas
- Os dois pacotes partem dos mesmos princípios básicos para criarem fluxos de trabalho consistentes

tidymodels

Os princípios *tidy* são os seguintes:

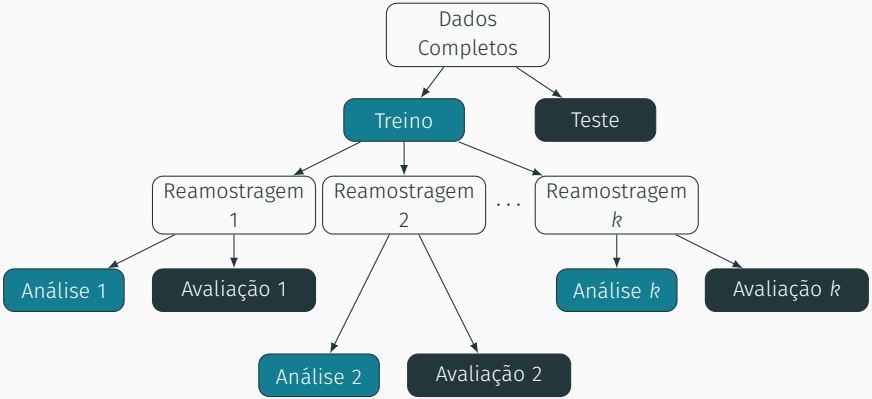
1. Reutilizar estruturas de dados existentes
2. Criar funções simples que utilizam *pipe* (`%>%`)
3. Adotar programação funcional
4. Feito para humanos

tidymodels

Os principais pacotes disponíveis dentro do `tidymodels` são:

- `rsample`: tipos diferentes de reamostragem
- `recipes`: transformações para pré-processamento de dados
- `parsnip`: uma interface comum para modelagem
- `yardstick`: medidas de desempenho do modelo

tidymodels


Dados Completos
Treino
Teste
Reamostragem 1
Reamostragem 2
...
Reamostragem k
Análise 1
Avaliação 1
Análise 2
Avaliação 2
Análise k
Avaliação k

Aplicação Tradicional

Aplicação Tradicional

- A seguir veremos como ajustar um modelo de regressão linear simples a um conjunto de dados
- É importante entendermos os passos do ajuste de um modelo que já conhecemos para depois expandirmos o método para outros tipos de modelagem

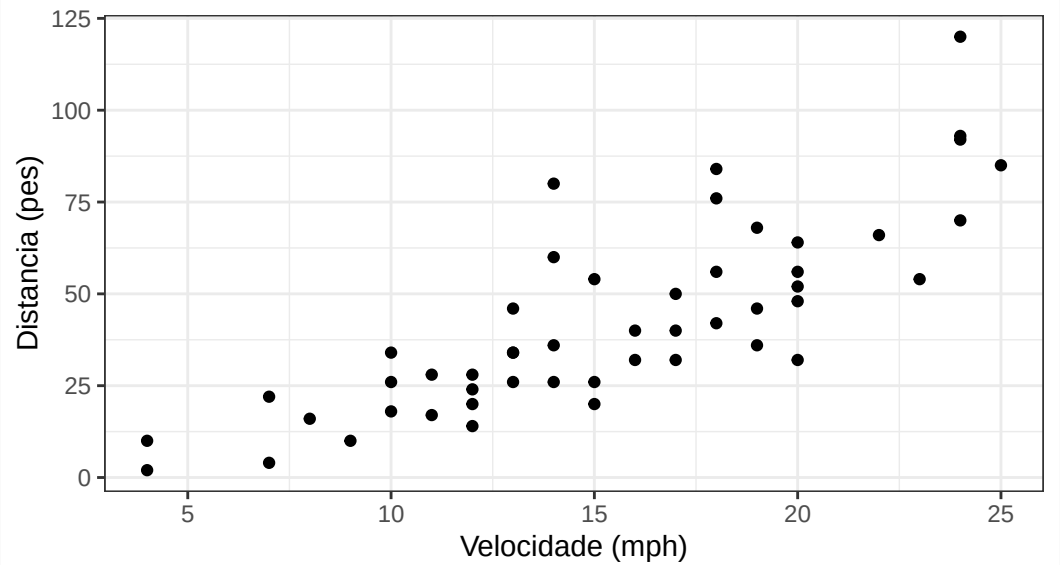
Aplicação Tradicional

- Nossa regressão será feita no conjunto de dados `cars`
- Ele possui apenas duas colunas
  - `speed`: velocidade do carro (milhas por hora)
  - `dist`: distância que o carro levou para parar completamente (pés)

Aplicação Tradicional

```
> library(tidymodels)
> theme_set(theme_bw())
>
> ggplot(cars, aes(x = speed, y = dist)) +
+   geom_point() +
+   labs(x = "Velocidade (mph)", y = "Distancia (pes)")
```

Aplicação Tradicional



Aplicação Tradicional

```
> # determinacao do software
>
> cars_lm <-
+   linear_reg() %>%
+   set_engine("lm")
>
> # ajuste do modelo
>
> cars_lm_fit <-
+   cars_lm %>%
+   fit(dist ~ speed,
+       data = cars)
```

Aplicação Tradicional

```
> # resultados
>
> cars_lm_fit

## parsnip model object
##
## Fit time:  13ms
##
## Call:
## stats::lm(formula = dist ~ speed, data = data)
##
## Coefficients:
## (Intercept)        speed
##     -17.579        3.932
```

Aplicação Tradicional

```
> # resultados
>
> tidy(cars_lm_fit)

## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   -17.6      6.76      -2.60 1.23e- 2
## 2 speed           3.93     0.416      9.46 1.49e-12
```

Aplicação Treino/Teste

Aplicação Treino/Teste

```
> # semente aleatoria
>
> set.seed(555)
>
> # 75% dos dados como treino
>
> cars_split <- initial_split(cars, prop = .75)
> cars_split

## <Analysis/Assess/Total>
## <38/12/50>
```

Aplicação Treino/Teste

```
> # criar os conjuntos de dados de treino e teste
>
> cars_treino <- training(cars_split)
> nrow(cars_treino)/nrow(cars)

## [1] 0.76

> cars_teste  <- testing(cars_split)
> nrow(cars_teste)/nrow(cars)

## [1] 0.24
```

Aplicação Treino/Teste

```
> # receita
>
> cars_rec <-
+   recipe(dist ~ speed,
+          data = cars_treino)
>
> cars_rec

## Data Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          1
```

Aplicação Treino/Teste

```
> # modelo
>
> cars_lm <-
+   linear_reg() %>%
+   set_engine("lm")
>
> cars_lm

## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

Aplicação Treino/Teste

```
> # criar workflow
>
> cars_wflow <-
+   workflow() %>%
+   add_recipe(cars_rec) %>%
+   add_model(cars_lm)
>
> # ajuste do modelo
>
> cars_lm_fit_treino <- fit(cars_wflow, cars_treino)
```

Aplicação Treino/Teste

```
> tidy(cars_lm_fit_treino)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   -16.3      7.32      -2.23 3.22e- 2
## 2 speed           3.90     0.473      8.25 8.23e-10
```

```
> tidy(cars_lm_fit)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   -17.6      6.76      -2.60 1.23e- 2
## 2 speed           3.93     0.416      9.46 1.49e-12
```

Aplicação Validação Cruzada

Aplicação Validação Cruzada

```
> # semente aleatoria
>
> set.seed(321)
>
> # divisao dos dados
>
> cars_treino_cv <- vfold_cv(cars_treino, v = 5)
```

Aplicação Validação Cruzada

```
> cars_treino_cv

## #  5-fold cross-validation
## # A tibble: 5 x 2
##   splits         id
##   <list>         <chr>
## 1 <split [30/8]> Fold1
## 2 <split [30/8]> Fold2
## 3 <split [30/8]> Fold3
## 4 <split [31/7]> Fold4
## 5 <split [31/7]> Fold5
```

Aplicação Validação Cruzada

```
> # modelo ajustado com validacao cruzada
>
> cars_lm_fit_cv <- fit_resamples(cars_wflow, cars_treino_cv)
```

Aplicação Validação Cruzada

```
> cars_lm_fit_cv

## # Resampling results
## # 5-fold cross-validation
## # A tibble: 5 x 4
##   splits         id    .metrics          .notes
##   <list>         <chr> <list>            <list>
## 1 <split [30/8]> Fold1 <tibble [2 x 4]> <tibble [0 x 1]>
## 2 <split [30/8]> Fold2 <tibble [2 x 4]> <tibble [0 x 1]>
## 3 <split [30/8]> Fold3 <tibble [2 x 4]> <tibble [0 x 1]>
## 4 <split [31/7]> Fold4 <tibble [2 x 4]> <tibble [0 x 1]>
## 5 <split [31/7]> Fold5 <tibble [2 x 4]> <tibble [0 x 1]>
```

Aplicação Validação Cruzada

```
> # resultados
>
> collect_metrics(cars_lm_fit_cv)

## # A tibble: 2 x 6
##   .metric .estimator   mean     n std_err .config
##   <chr>   <chr>       <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   14.6       5  2.14   Preprocessor1_Mod~
## 2 rsq     standard    0.663     5  0.0835 Preprocessor1_Mod~

> sqrt(mean((predict(cars_lm_fit$fit) - cars$dist)^2))

## [1] 15.06886
```

Aplicação Validação Cruzada

```
> # resultados no conjunto de teste
>
> resultado <-
+   cars_teste %>%
+   bind_cols(predict(cars_lm_fit_treino, cars_teste) %>%
+               rename(predicao_lm = .pred))
```

Aplicação Validação Cruzada

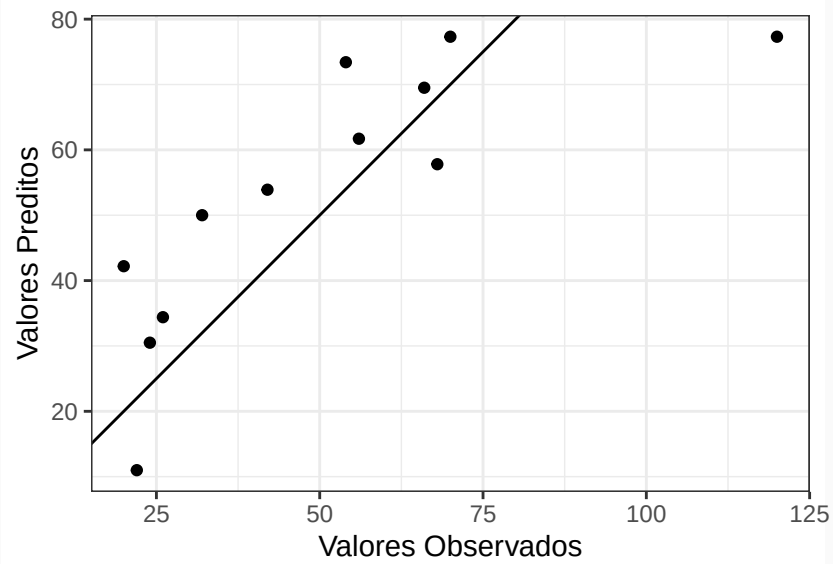
```
> # resultado final
>
> metrics(resultado,
+         truth = dist,
+         estimate = predicao_lm)

## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      17.3
## 2 rsq     standard       0.632
## 3 mae     standard      13.9
```

Aplicação Validação Cruzada

```
> # grafico final
>
> ggplot(resultado, aes(x = dist, y = predicao_lm)) +
+   geom_point() +
+   labs(x = "Valores Observados", y = "Valores Preditos") +
+   geom_abline(intercept = 0, slope = 1) +
+   coord_fixed()
```

Aplicação Validação Cruzada



Exercícios

Exercícios

O conjunto de dados `mpg` faz parte do pacote `ggplot2`. Ele possui informações a respeito de carros vendidos no mercado norte-americano. As variáveis são:

- `manufacturer`: fabricante
- `model`: modelo do carro
- `displ`: tamanho do motor em litros
- `year`: ano de fabricação
- `cyl`: número de cilindros
- `trans`: tipo de transmissão
- `drv`: tipo de tração
- `cty`: consumo na cidade em milhas por galão
- `hwy`: consumo na estrada em milhas por galão
- `fl`: tipo de combustível
- `class`: tipo de automóvel

Exercícios

1. Crie um novo objeto chamado `mpg2` com apenas as variáveis numéricas presentes no conjunto de dados
2. Encontre a variável com a maior correlação negativa com a variável `hwy`. Visualize essa relação em um gráfico de dispersão. Intuitivamente, a correlação entre estas variáveis faz sentido? Explique.
3. Crie conjuntos de treinamento e teste com o objeto `mpg2`. Reserve 80% das observações para o conjunto de treinamento.
4. Utilize a validação cruzada com 5 grupos para ajustar um modelo de regressão linear simples. Utilize `hwy` como variável resposta e a variável encontrada na pergunta 2 como preditora.

Exercícios

5. Verifique se o resultado do ajuste ficou aceitável, comparando os RMSE do modelo nos conjuntos de treinamento e teste.
6. Ajuste uma regressão linear múltipla neste conjunto de dados. Mantenha `hwy` como variável resposta e adicione as outras variáveis numéricas como variáveis preditoras. Padronize as variáveis preditoras no conjunto de teste com as funções `step_center`, `step_scale`, `prep` e `juice` do pacote `tidymodels`.
7. Como ficou o novo ajuste? Decida baseando-se em medidas relevantes nos conjuntos de treinamento e teste (transforme os dados do conjunto teste com a função `bake`) e em gráficos de diagnóstico.

tidymodels

EST0133 - Introdução à Modelagem de Big Data

Marcus Nunes
https://introbigdata.org/
https://marcusnunes.me/

Universidade Federal do Rio Grande do Norte