

# Trabalho de Aplicação - Rio São Francisco

Yugo Oyama NUSP: 9297784

14/11/2021

## Introdução

Este trabalho consiste na aplicação de técnicas estatísticas abordadas no curso de Estatística em Altas Dimensões - 2021 em dois bancos de dados que serão explicados detalhadamente mais a frente.

## Chuva e vazão no Rio São Francisco

### Dados

Os dados armazenados no arquivo `dados_rio_sf.rdata` são referentes a medições de vazão e chuva em estações localizadas na região do rio São Francisco.

Dentro deste arquivo, há três variáveis:

`treino_sf`: conjunto de treino com as medições de vazão e precipitação nas estações consideradas; `estacoes`: com informações sobre as estações que coletam os dados; `teste_sf`: conjunto com dados que serão usados para avaliar a performance dos modelos preditivos;

### Dados de treinamento

O dataframe `dados_treino` contém 1717 linhas e 83 colunas. Cada linha contém medições semanais de vazão em diferentes estações fluviométricas sobre o rio São Francisco e de chuva em estações pluviométricas em regiões próximas do rio. Se a estação é fluviométrica, o dado armazenado é a vazão média registrada na semana correspondente. Caso a estação seja pluviométrica, o dado registrado é a chuva acumulada naquela semana. A primeira coluna, chamada `Y`, contém a medida de vazão na estação fluviométrica de código 46998000, que corresponde à estação em que se tem interesse em fazer previsão. A vazão na coluna `Y` corresponde à medição na semana seguinte às demais medições da mesma linha.

### Estações

O dataframe `estações` contém informações sobre cada uma das estações que aparecem no dataframe `treino_sf`. Essas informações estão disponibilizadas apenas a título de curiosidade e não devem ser utilizadas para construir os modelos preditivos. Por exemplo, latitude e longitude, além do tipo da estação (fluviométrica ou pluviométrica) e outros detalhes técnicos de cada uma delas.

### Objetivo

O objetivo desta análise é propor um modelo que consiga realizar boas previsões para as medições de vazão na estação 46998000 (coluna `Y`), dadas as medições tomadas nas estações do sistema na semana anterior (demais colunas).

### Técnicas

Para esse projeto serão usados modelos lineares com regularização, modelos baseados em árvores, e redes neurais.

Para comparar os modelos entre si, será utilizado o erro absoluto médio (MAE) e ao final de cada método testado, será adicionado o resultado a uma tabela comparativa.

### Parâmetros

Para o ajuste do modelo, foi definido que o conjunto de treino seria dividido em conjunto de treino e validação na proporção 8:2, ou seja, 20% do conjunto originalmente de treino foi usado para validação. Foi definida também uma semente por questão de reprodutibilidade (1234).

### Modelo Média

De forma a ter uma base para comparar, foi construído um modelo simples que considera todos os valores da estação 46998000, calcula a média e sempre prevê que a próxima medição será a média.

Temos os resultados abaixo.

```
##      modelo erro_dentro erro_fora
## 1  Media      1229.247  1117.671
```

### Lasso

Para criar um modelo de regressão com a penalização lasso, foi utilizada a biblioteca glmnet. Com ela, dentro da amostra de treino, foi ajustado o modelo de regressão com penalização lasso utilizando validação cruzada. Em seguida, foi testado o modelo obtido no conjunto de validação e calculado o respectivo erro dentro e fora.

Para a escolha do lambda, iniciou-se uma sequência: 0.01, 0.1, 1, 2, 10, 25, 50, 75, 100 e permitiu que a função cv.glmnet escolhesse o melhor lambda. Existem dois lambdas que são popularmente usados: o que minimiza o erro gerado pela validação cruzada e o no qual o erro não ultrapassa um desvio padrão do melhor modelo. Com isso, foram testados modelos com cada um dos lambdas e calculados os erros dentro e fora respectivos de cada um.

Ainda como modelo alternativo, foi definido que a função cv.glmnet escolhesse 30 lambdas diferentes e ajustasse um modelo.

Os resultados obtidos podem ser observados na tabela a seguir.

```
##              modelo erro_dentro erro_fora
## 1              Media      1229.247  1117.671
## 2      Lasso - lambda minimo      159.3055  167.9468
## 3  Lasso - lambda 1 desvio padrao      164.7565  171.2722
## 4      Lasso 2- lambda minimo      159.4736  167.9468
## 5  Lasso 2- lambda 1 desvio padrao      165.0675  171.5473
```

Percebemos que em relação ao modelo média, há uma expressiva melhora tanto no erro dentro como no erro fora.

Em relação ao uso do lambda mínimo ou lambda 1 desvio padrão, os resultados são muito semelhantes, sendo que o lambda mínimo obteve um desempenho um pouco melhor.

Em relação a aos modelos com a lista de lambdas fornecidas e com os 30 lambdas escolhidos pela função, os resultados obtidos foram muito semelhantes.

### Modelo de Floresta Aleatória

Foi escolhido o modelo de Floresta Aleatória já que dessa forma existe variabilidade na escolha das variáveis explicativas e na escolha das observações para a construção de cada árvore. Essa variabilidade por sua vez pode levar a uma melhor predição no conjunto de validação e no de teste.

Para realizar o ajuste do modelo, foi utilizado a função randomForest do pacote randomForest.

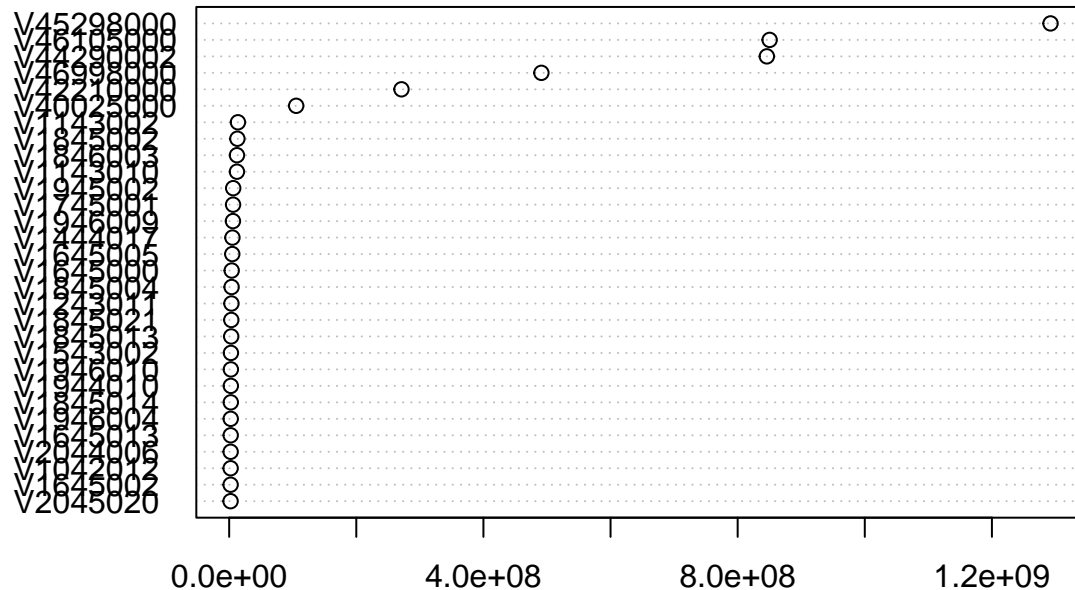
Como os nomes das estações são numéricas foi necessário fazer uma pequena alteração para que o comando `randomForest` funcionasse.

A seguir, podemos ver as cinco estações mais importantes e as cinco menos importantes para a predição.

```
##          IncNodePurity
## V45298000    1292229623
## V46105000     850297391
## V44290002     845926098
## V46998000     491202165
## V42210000     271091061
## V40025000     105445025
```

```
##          IncNodePurity
## V1343008      845293.7
## V1944011      840024.1
## V1544018      737681.6
## V1242015      733611.5
## V1444005      672340.8
```

Ainda, é possível visualizar melhor no gráfico a seguir a importância de cada estação.



Percebemos claramente pelo gráfico que as estações 45298000,46105000,44290002,46998000,42210000,40025000 são as estações que mais contribuem para a previsão. As outras estações por outro lado contribuem pouco e de forma semelhante para a previsão.

##	modelo	erro_dentro	erro_fora
## 1	Media	1229.2470	1117.6707
## 2	Lasso - lambda minimo	159.3055	167.9468
## 3	Lasso - lambda 1 desvio padrao	164.7565	171.2722
## 4	Lasso 2- lambda minimo	159.4736	167.9468
## 5	Lasso 2- lambda 1 desvio padrao	165.0675	171.5473
## 6	Floresta Aleatoria	52.7926	113.4956

Novamente, percebemos uma melhora significativa em relação ao modelo média.

Em relação ao modelo de lasso, percebemos melhora expressiva no erro dentro e melhora significativa no erro fora.

## Redes neurais

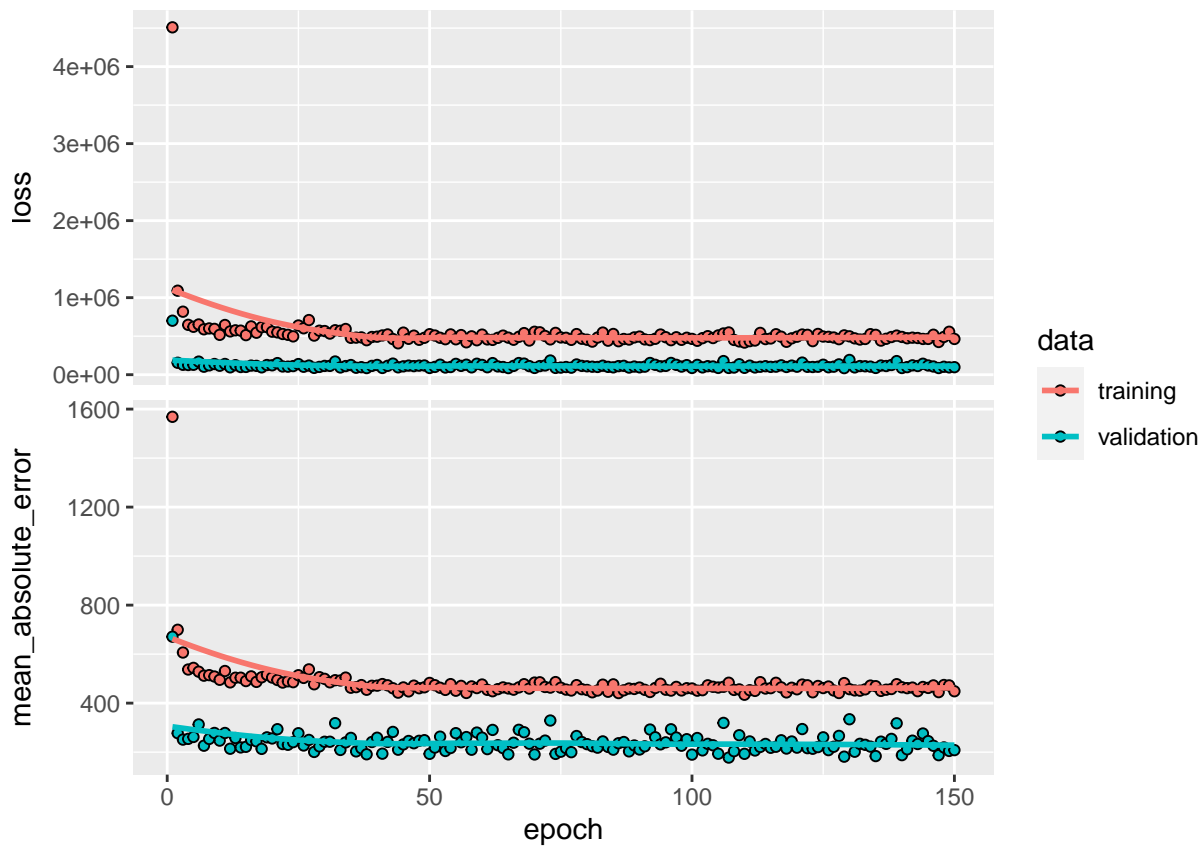
Para o modelo de redes neurais, foi utilizada a função de ativação ReLU ao invés da sigmoid por ser computacionalmente mais rápida.

Vamos considerar um modelo com apenas uma camada oculta e 41 unidades, por corresponder a 50% da quantidade de camadas de entrada e um modelo com 3 camadas ocultas com 60,40,20 unidades respectivamente por corresponderem a aproximadamente 75%, 50%, 25%.

Foram considerados 40% de supressão, 150 epochs e tamanho de batch = 32.

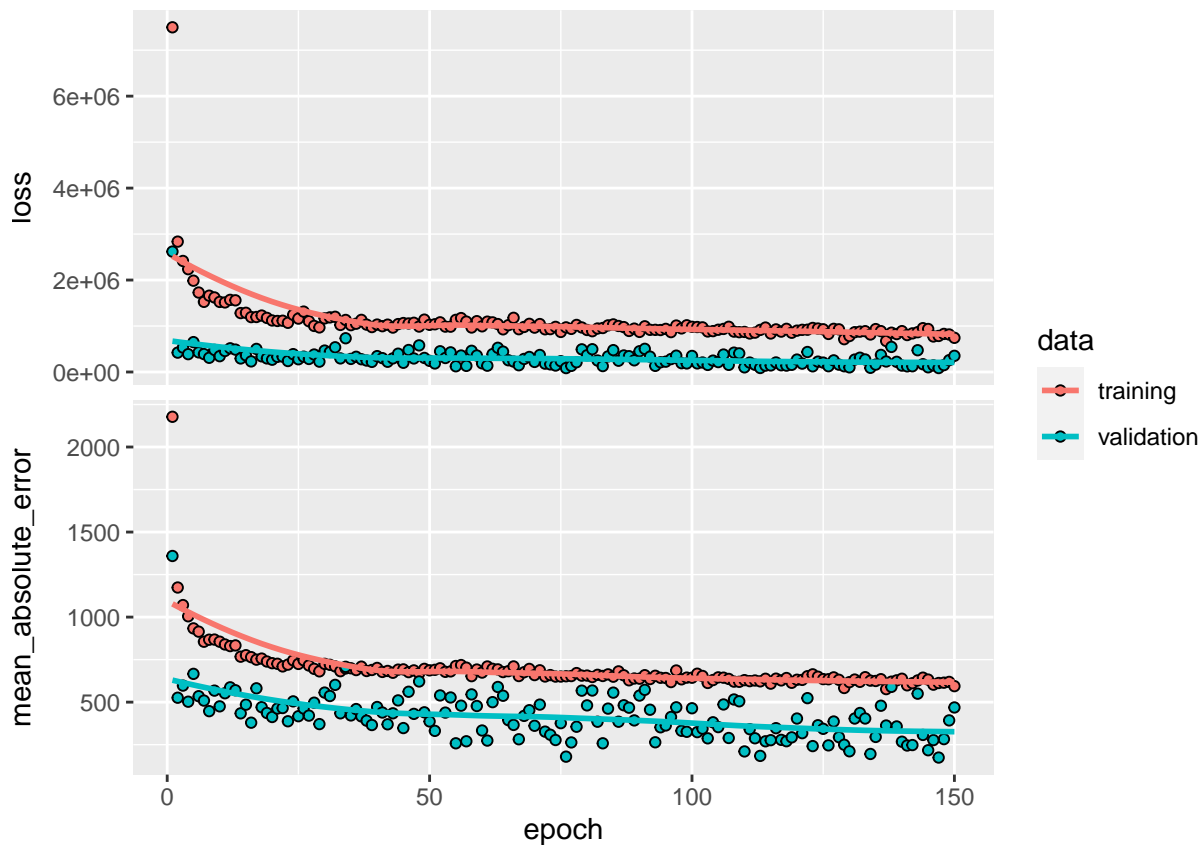
```
## Loaded Tensorflow version 2.5.0
```

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



Podemos observar os resultados a seguir.

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



Podemos observar os resultados a seguir.

##	modelo	erro_dentro	erro_fora
## 1	Media	1229.2470	1117.6707
## 2	Lasso - lambda minimo	159.3055	167.9468
## 3	Lasso - lambda 1 desvio padrao	164.7565	171.2722
## 4	Lasso 2- lambda minimo	159.4736	167.9468
## 5	Lasso 2- lambda 1 desvio padrao	165.0675	171.5473
## 6	Floresta Aleatoria	52.7926	113.4956
## 7	Rede Neural	185.6514	208.2235
## 8	Rede Neural 2	469.4136	468.7537

Percebemos que aumentar o numero de camadas ocultas não melhorou o erro de previsão do modelo. Pelo contrário, piorou-a.

Dentre modelos, o que teve melhor desempenho tanto no erro dentro quanto no erro fora foi o modelo de floresta aleatória. Dessa forma, a previsão para os dados de teste foi feita utilizando o modelo de floresta aleatória.