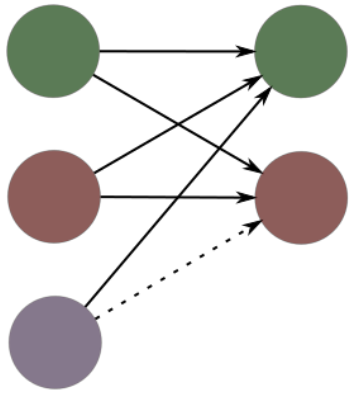


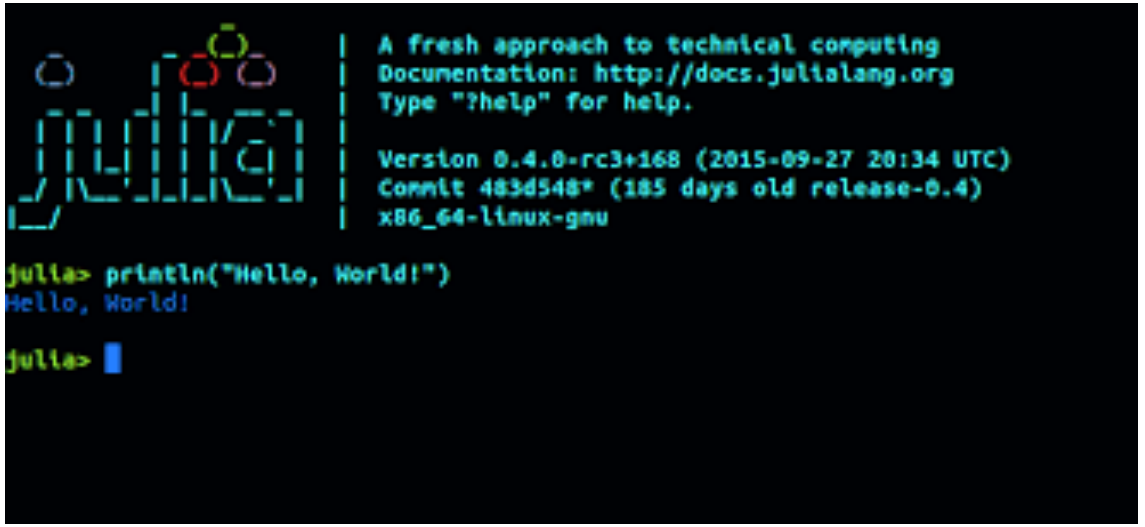
Sistemas de Recomendação utilizando Persa.jl

Filipe Braida

Julia Programming

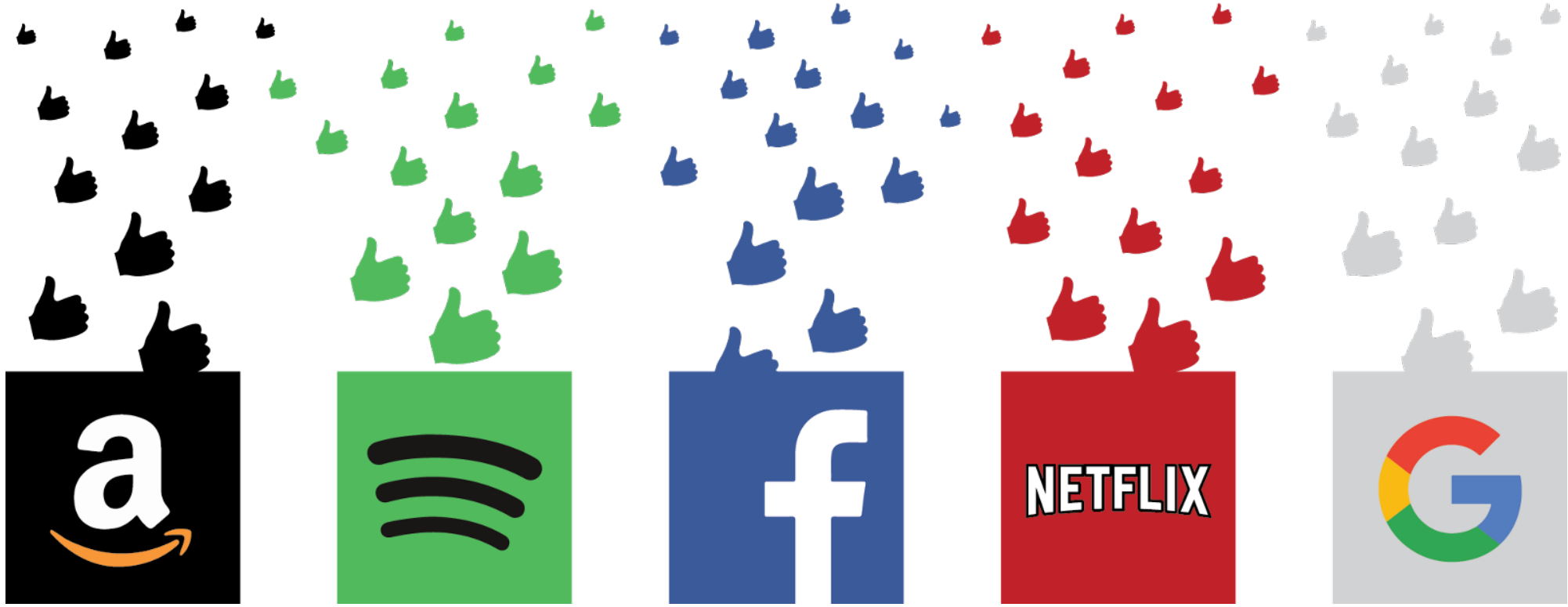
Hello World!

- Filipe Braida
  - Doutorado em Engenharia de Sistemas e Computação na UFRRJ
  - Professor da Universidade Federal Rural do Rio de Janeiro
  - http://filipe.braida.com.br



Sistemas de Recomendação

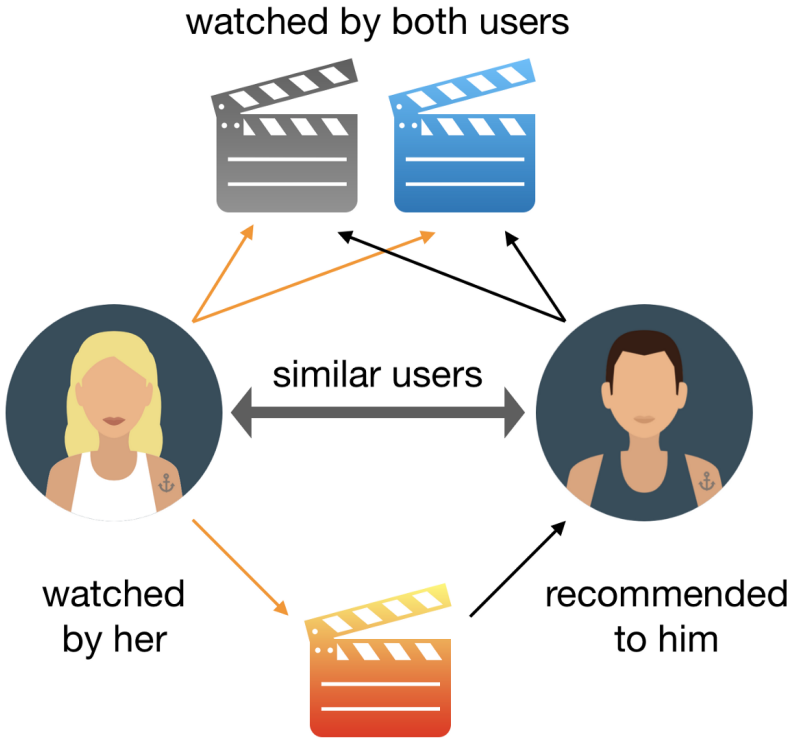
- Sistemas de Recomendação são uma realidade!



Fonte: https://thedatascientist.com/right-way-recommender-system-startup/

Sistemas de Recomendação

- Filtragem Colaborativa é uma das abordagens mais utilizadas em Sistemas de Recomendação
  - Utilizar as opiniões das outras pessoas para recomendação



Fonte: https://towardsdatascience.com/

Filtragem Colaborativa

- Possível representar através de uma matriz usuário-item
  - Essa matriz é esparsa!

	Titanic	Poderoso Chefão	Matrix
Filipe	4	∅	3
Orleans	4	5	5
Bruno	4	5	5
Fellipe	∅	5	∅

Filtragem Colaborativa

- Na Filtragem Colaborativa:

- Na Filtragem Colaborativa:

- O problema central:
  - Não é definida para todo o espaço UxI
  - Torna-se necessário extrapolá-la para todo o espaço!
  - A extrapolação, nesse caso, pode ser feita estimando a função

- O problema central:
  - Não é definida para todo o espaço UxI
  - Torna-se necessário extrapolá-la para todo o espaço!
  - A extrapolação, nesse caso, pode ser feita estimando a função

Contextualização

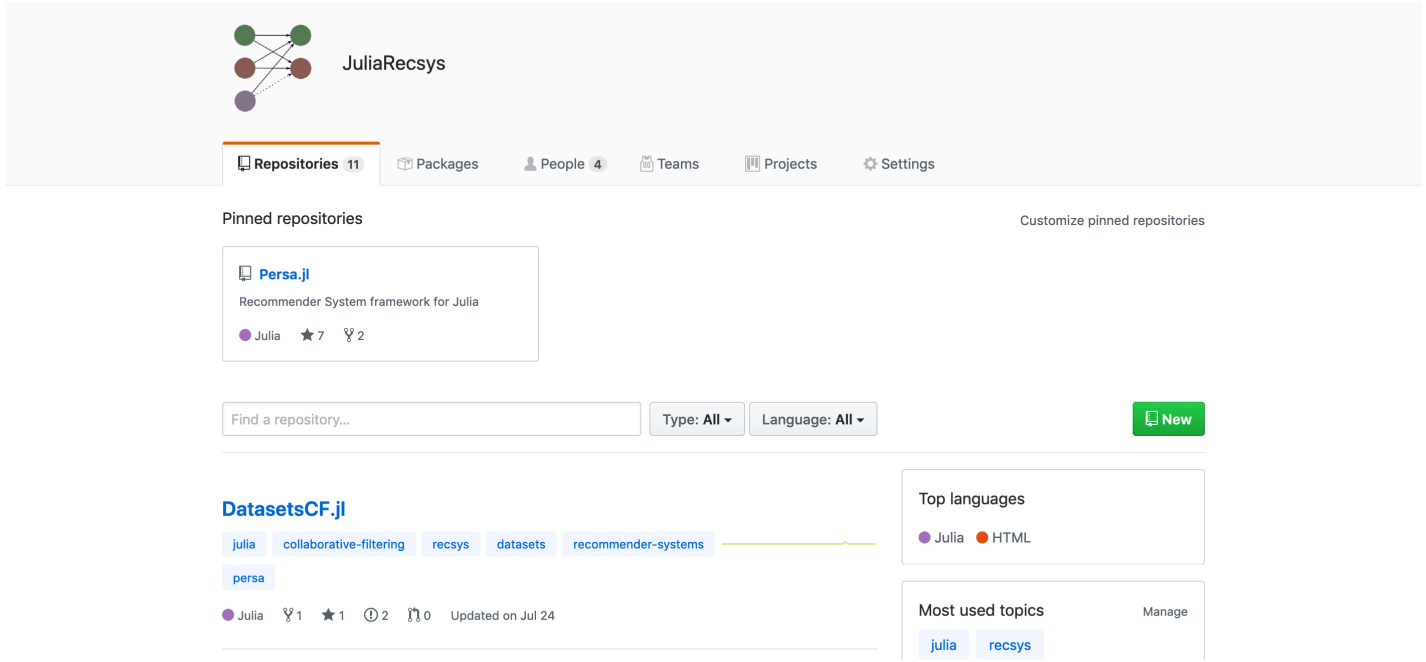
- The Netflix prize
  - $1 milhão de dólares para o aumento de 10% da acurácia
  - Desenvolvimento dos algoritmos preditivos para Filtragem Colaborativa
  - Período: 2006 – 2009
- Em 2010:
  - Inicio do mestrado
  - **Poucos** *datasets* disponíveis
  - **Nenhuma** implementação dos algoritmos
  - **Sem** uma padronização da metodologia experimental

Histórico de Versões

- 2010 até 2013 – MATLAB
- 2013 – Python
- 2013 até 2016 – Recsys.jl
- 2016 até 2018 – Persa.jl

JuliaRecsys

- JuliaRecsys é um ecossistema de pacotes para Recomendação
- Comunidade no GitHub: https://github.com/JuliaRecsys
- Principais Pacotes
  - Persa.jl
  - DatasetsCF.jl
  - ModelBased.jl
  - ...



DatasetsCF.jl

- Pacote com os principais *datasets* de Filtragem Colaborativa
  - Maior dificuldade está nas licenças dos *datasets*

```
using DatasetsCF
dataset = DatasetsCF.MovieLens()
```

```
Collaborative Filtering Dataset
  – # users: 943
  – # items: 1682
  – # ratings: 100000
  – Ratings Preference: [1, 2, 3, 4, 5]
```

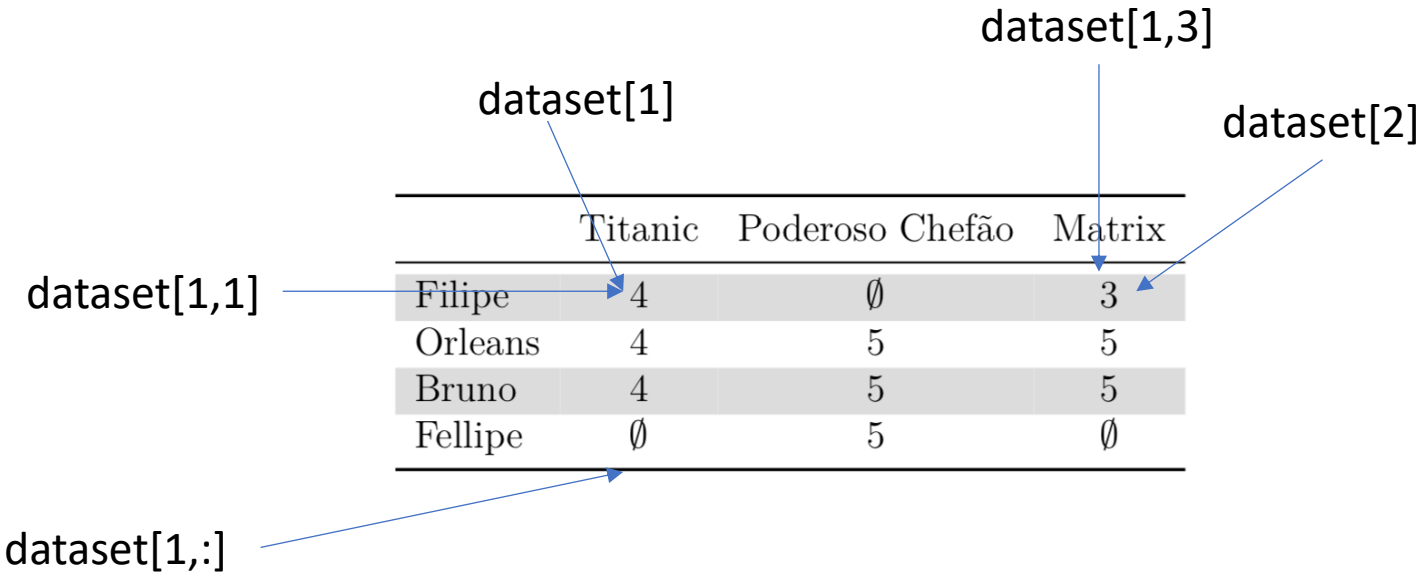
- Datasets:
  - MovieLens 100k
  - Movielens 1M

Persa.jl

- Pacote com o núcleo para Filtragem Colaborativa
- Possui as definições:
  - *Dataset*
  - Conjunto de Preferência
    - Avaliações
  - Modelo Preditivo

Persa.jl

- Persa.jl fornece uma navegação padronizada para o *dataset*
  - Internamente armazena os dados utilizando representação esparsa
    - Pode mudar caso for necessário
      - Ex: Armazenar em um banco de dados
  - O usuário pode utilizar a representação esparsa ou completa para acessar os dados



	Titanic	Poderoso Chefão	Matrix
Filipe	4	∅	3
Orleans	4	5	5
Bruno	4	5	5
Fellipe	∅	5	∅

Persa.jl

- Dependendo do acesso muda o retorno
  - Uma avaliação de um usuário sobre um item
    - Retorno é uma avaliação

```
dataset[1,1]    Rating: 5
```

  - A primeira avaliação de um sistema
    - Retorno é uma preferência de um usuário sobre um item

```
preference = dataset[2]    (user: 2, item: 1, rating: 4)

preference[1]    2
preference[2]    1
preference[3]    Rating: 4
```

Persa.jl

- Através da ortogonalidade temos:

```
preferences = dataset[1,:]
```

```
Persa.UserPreference{Int64}[272]
  (user: 1, item: 1, rating: 5)
  (user: 1, item: 2, rating: 3)
  (user: 1, item: 3, rating: 4)
  (user: 1, item: 4, rating: 3)
  (user: 1, item: 5, rating: 3)
  (user: 1, item: 6, rating: 5)
  (user: 1, item: 7, rating: 4)
  (user: 1, item: 8, rating: 1)
  (user: 1, item: 9, rating: 5)
  (user: 1, item: 10, rating: 3)
  ...
  (user: 1, item: 263, rating: 1)
```

Persa.jl

- Através da ortogonalidade temos:

```
for (u,v,r) in dataset
    print("User $u, Item $v, Rating $r")
end
```

```
for (u,v,r) in dataset[1,:]
    print("User $u, Item $v, Rating $r")
end
```

```
Persa.users(dataset)    943
Persa.items(dataset)    1682
length(dataset)    100000
size(dataset)    (943, 1682)
```

Persa.jl

- Conjunto de Preferência

```
struct Preference{T}
  possibles::Array{T, 1}
  min::T
  max::T
end
```

```
preference = dataset.preference   Ratings Preference: [1, 2, 3, 4, 5]

value = round(1.1, preference)   1

eltype(preference)   Int64
size(preference)   5
minimum(preference)   1
maximum(preference)   5
```

Persa.jl

- Grande diferencial do Persa.jl é a abstração do modelo
  - Deverá estender o tipo abstrato Persa.Model{T}
  - Deverá possuir três campos:
    - Quantidade de usuários
    - Quantidade de itens
    - Conjunto de Preferência
  - Deverá ter o método:

    Persa.predict(model::RandomModel, user::Int, item::Int)
  - Pode ter o método:

    Persa.train!(model::RandomModel, dataset::Persa.Dataset)

Persa.jl

- Exemplo:
  - Modelo Aleatório

```
using Random

mutable struct RandomModel{T} <: Persa.Model{T}
    preference::Persa.Preference{T}
    users::Int
    items::Int
end

RandomModel(dataset::Persa.Dataset) = RandomModel(dataset.preference, Persa.users(dataset), Persa.items(dataset))

Persa.predict(model::RandomModel, user::Int, item::Int) = rand(model.preference.possibles)
```

Persa.jl

- Exemplo:
  - Modelo Aleatório

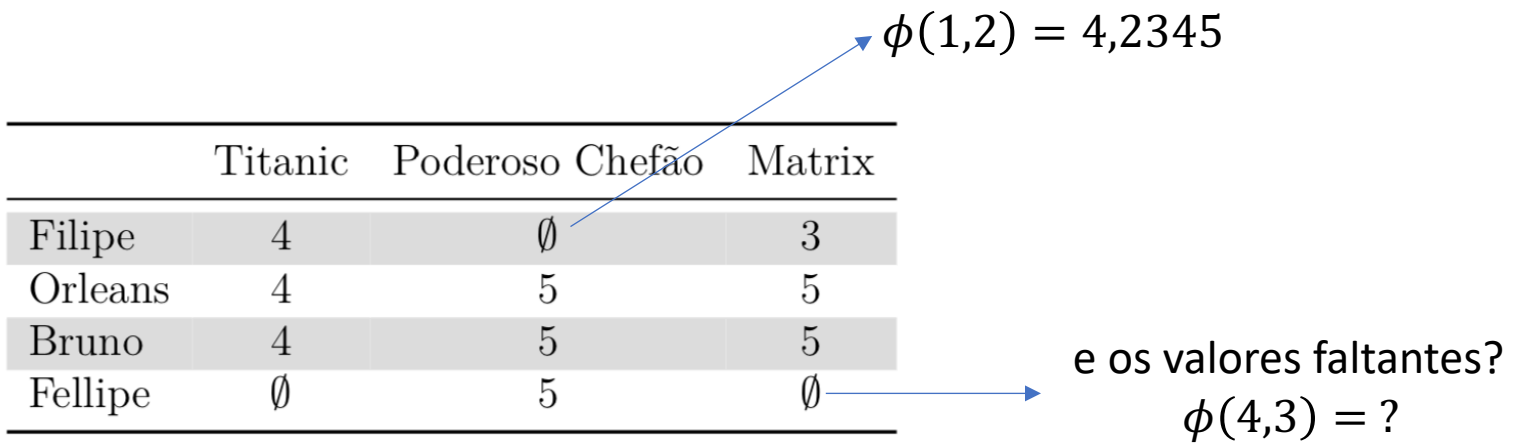
```
dataset = DatasetsCF.MovieLens()    > Collaborative Filtering Dataset

model = RandomModel(dataset)    > RandomModel
rating = model[1,1]    Rating: 1 (1)
```

```
ratings = model[1,:]    Vector{Persa.PredictRating{Int64}} with 1682 elements
                          Rating: 4 (4)
                          Rating: 4 (4)
                          Rating: 2 (2)
                          Rating: 5 (5)
                          Rating: 2 (2)
                          Rating: 5 (5)
                          Rating: 2 (2)
                          Rating: 5 (5)
                          Rating: 4 (4)
                          Rating: 2 (2)
                          Rating: 1 (1)
                          Rating: 3 (3)
```

O modelo é tratado exatamente igual um *Dataset!*

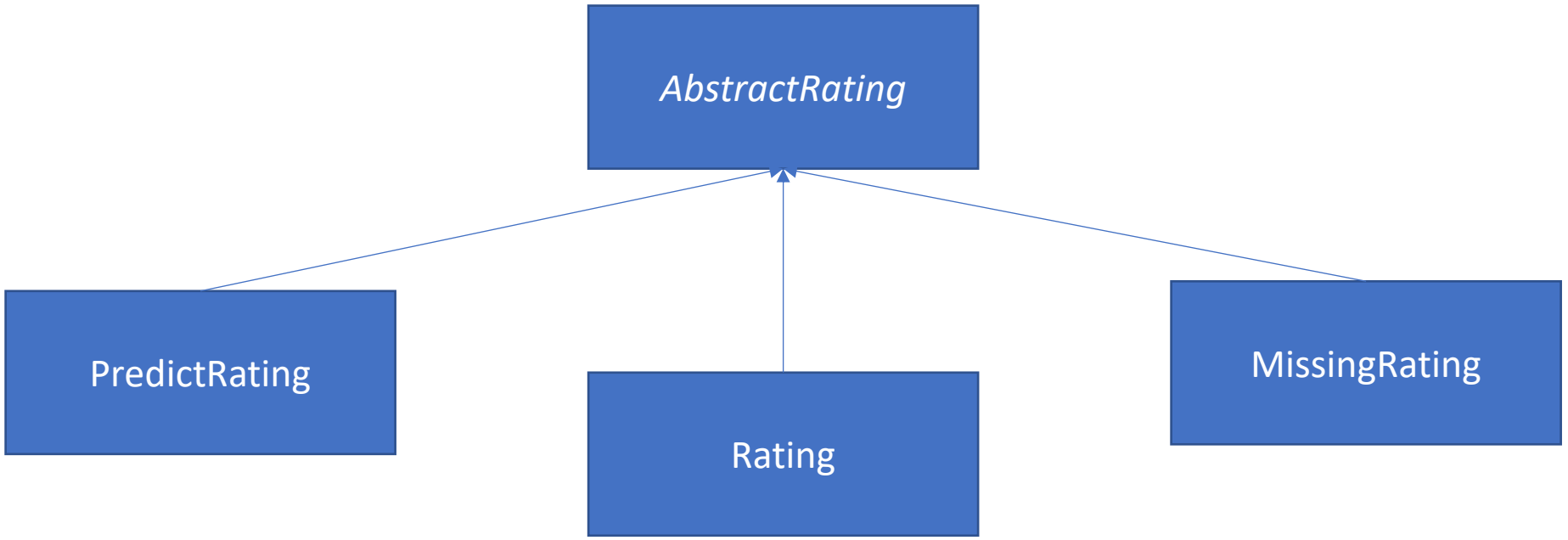
Persa.jl

- Existe uma diferença conceitual entre uma avaliação prevista e uma avaliação de um *Dataset*
  - Avaliação prevista pode assumir qualquer valor real entre o menor e o maior valor
  - Avaliação do dataset deve ser algum elemento do conjunto de preferência



Persa.jl

- Hierarquia de tipos:



Persa.jl

- O pacote fornece operações para as avaliações!

```
rating = model[1,1]   Rating: 3 (3)
rating + 1   4
```

```
rating = dataset[1,1]   Rating: 5
rating + 1.1   6.10
```

```
dataset[5,3]   Rating: missing
```

```
model[1,1] - model[1,1]   -1
model[1,1] / 2   0.500
model[1,1] * 2   2
```

Persa.jl

- No caso do PredictRating existe alguns detalhes

```
struct PredictRating{T <: Number} <: AbstractRating{T}
    value::Real
    target::T
    PredictRating(x::Real, preference::Preference{T}) where T <: Number = new{T}(correct(x, preference), round(x, preference))
end
```

```
Persa.predict(model::RandomModel, user::Int, item::Int) = 1.2   ✓
rating = model[1,1]   Rating: 1.2 (1)
rating.value   1.20
rating.target   1
```

Persa.jl

- O pacote verifica automaticamente a faixa de valores
  - Problema comum nos algoritmos baseados em memória!

```
Persa.predict(model::RandomModel, user::Int, item::Int) = 6
rating = model[1,1]   Rating: 5 (5)
rating.value   5
rating.target   5
```

Exemplo

- SVD by Funk
  - Algoritmo que mudou o rumo da Netflix Prize

$$\tilde{r}_{ui} = q_i^t p_u$$

$$\underset{q_*, p_*}{\arg\min} \sum_{(u,i,r) \in R} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

$$e_{ui} = r_{ui} - q_i^T p_u$$

$$q_i \leftarrow q_i + \gamma \times (e_{ui} p_u - \lambda q_i)$$

$$p_u \leftarrow p_u + \gamma \times (e_{ui} q_i - \lambda p_u)$$

https://sifter.org/~simon/journal/20061211.html

Exemplo

```
abstract type MatrixFactorization{T} <: Persa.Model{T}
end

mutable struct RegularizedSVD{T} <: MatrixFactorization{T}
    P::Array
    Q::Array
    preference::Persa.Preference{T}
    users::Int
    items::Int
end

const RSVD = RegularizedSVD

function RegularizedSVD(dataset::Persa.Dataset, features::Int)
    (users, items) = size(dataset)

    P = rand(users, features)
    Q = rand(items, features)

    return RegularizedSVD(P, Q, dataset.preference, Persa.users(dataset), Persa.items(dataset))
end
```

Exemplo

- Previsão da avaliação

$$\tilde{r}_{ui} = q_i^t p_u$$

```
Persa.predict(model::RegularizedSVD, user::Int, item::Int) = model.P[user, :]' * model.Q[item, :]
```

Exemplo

- Objetivo

$$\underset{q_*,p_*}{\arg\min} \sum_{(u,i,r)\in R} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

```
function objective(model::RegularizedSVD, dataset::Persa.Dataset, λ::Float64)
    total = 0

    for (u, v, r) in dataset
        total += (r - model[u, v])^2
        total += λ * (norm(model.P[u,:])^2 + norm(model.Q[v,:])^2)
    end

    return total
end
```

Exemplo

- Atualização

$$e_{ui} = r_{ui} - q_i^T p_u$$

$$q_i \leftarrow q_i + \gamma \times (e_{ui} p_u - \lambda q_i)$$

$$p_u \leftarrow p_u + \gamma \times (e_{ui} q_i - \lambda p_u)$$

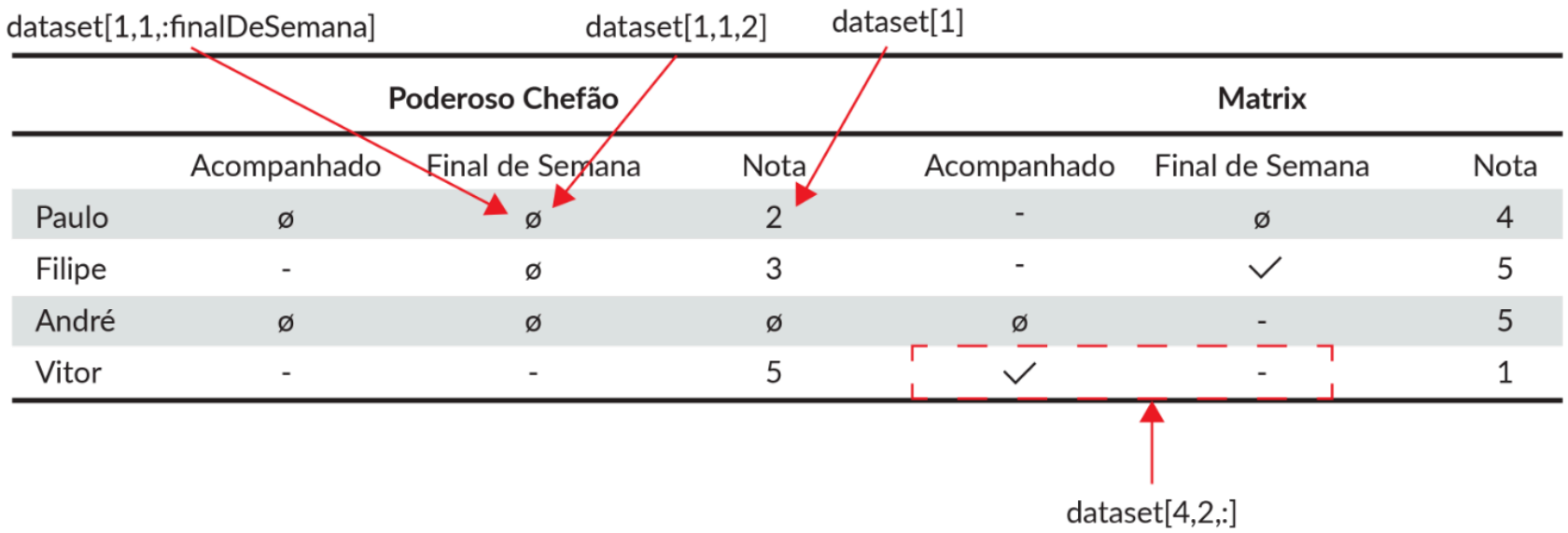
```
function update!(model::RegularizedSVD, dataset::Persa.Dataset, γ::Float64, λ::Float64)

    idx = shuffle(1:length(dataset))

    for i = 1:length(dataset)
        (u, v, r) = dataset[idx[i]]

        e = r - Persa.predict(model, u, v)

        P = model.P[u,:]
        Q = model.Q[v,:]

        model.P[u,:] += γ * (e .* Q .- λ .* P)
        model.Q[v,:] += γ * (e .* P .- λ .* Q)
    end
end
```

Outros Pacotes

- Existem outros pacotes em desenvolvimento:
  - EvaluationCF.jl
    - Mecanismo de avaliação experimental e métricas
  - ModelBasedCF.jl
    - Implementação de alguns algoritmos baseados em modelo
- Pacotes que estão na lista para serem atualizados:
  - NeuralCF.jl
  - COFILS.jl
  - Surprise.jl
  - ModelTune.jl
  - ...

Outros Pacotes

- Projeto Final que estende o Persa.jl adicionando o contexto
  - ContextCF.jl: Um Framework para CARS em Filtragem Colaborativa
  - Paulo Roberto Xavier Júnior



	Poderoso Chefão			Matrix		
	Acompanhado	Final de Semana	Nota	Acompanhado	Final de Semana	Nota
Paulo	ø	ø	2	-	ø	4
Filipe	-	ø	3	-	✓	5
André	ø	ø	ø	ø	-	5
Vitor	-	-	5	✓	-	1

Obrigado!

https://github.com/JuliaRecsys/