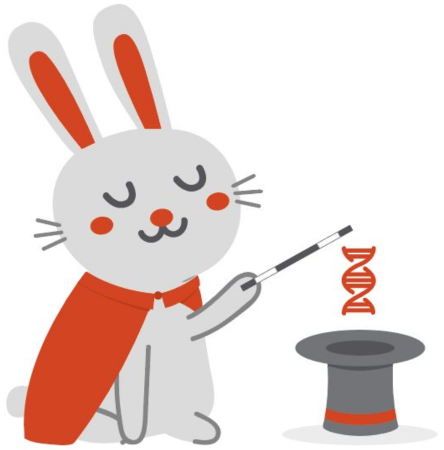
Abracadabra

Encontrando doenças raras com machine learning e bioinformática em Go



Vitor De Mario
twitter.com/vdemario

Quem sou eu

- 2 anos e meio de experiência em Go
- Alguns anos de Java
- Um pouco de Python
- Fascinado com computadores desde que me entendo por gente

Mendelics

- Abracadabra
- Machine learning

Como tudo começou

- CEO neurologista começou a programar sozinho

Por que Go?

- Praticamente um acidente
- Linguagem que o neurologista conseguiu sair do zero sozinho

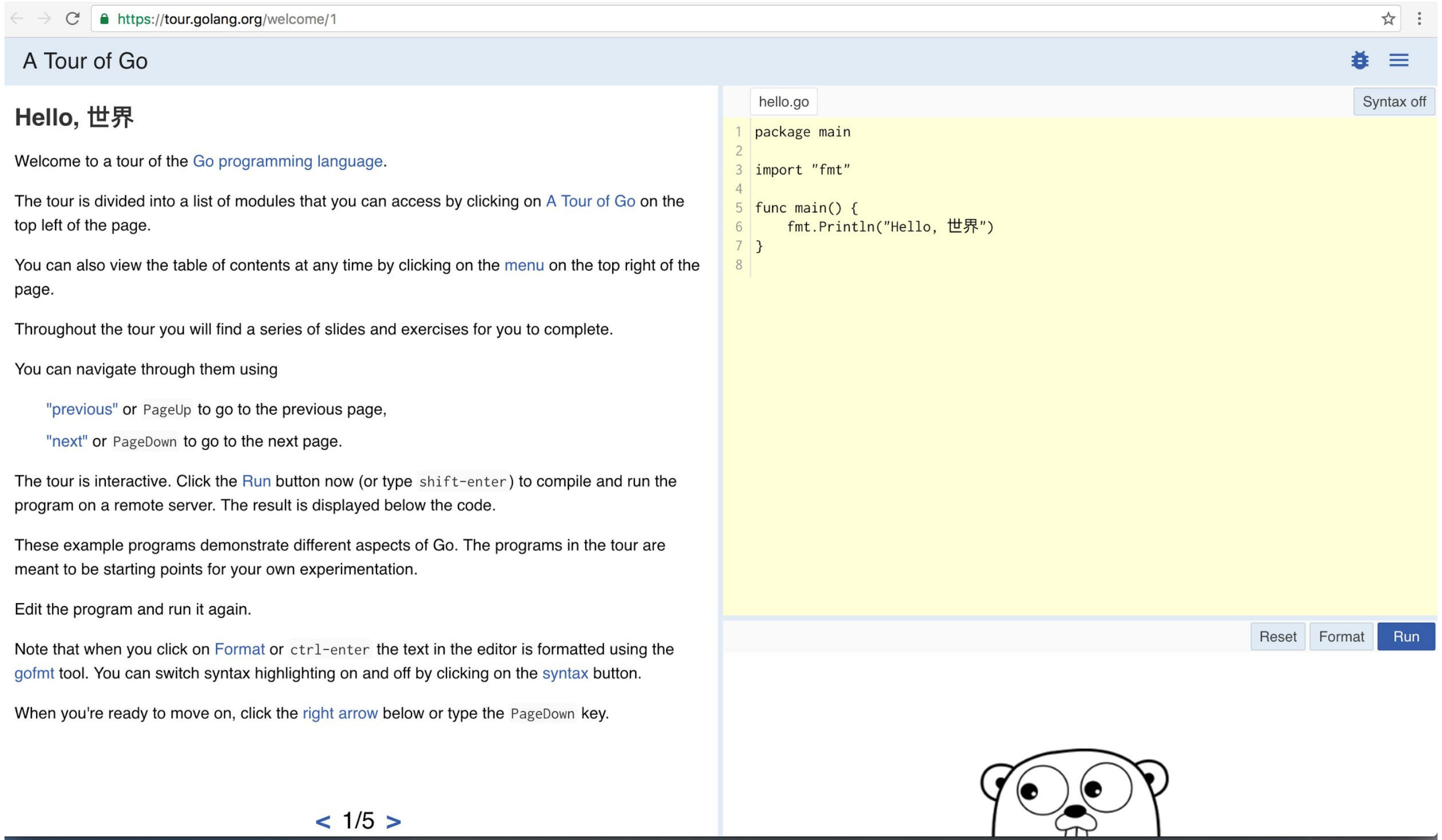
O monolito

Mas vamos ter que trabalhar ***nisso***?

Aprendendo Go

- Saindo do 0, golang.org tinha tudo que era necessário
- Instalação trivial
- A Tour of Go
- How to Write Go Code
- Em menos de uma semana produtivo

A Tour of Go - tour.golang.org



Hello, 世界

Welcome to a tour of the Go programming language.

The tour is divided into a list of modules that you can access by clicking on A Tour of Go on the top left of the page.

You can also view the table of contents at any time by clicking on the menu on the top right of the page.

Throughout the tour you will find a series of slides and exercises for you to complete.

You can navigate through them using

- "previous" or `PageUp` to go to the previous page,
- "next" or `PageDown` to go to the next page.

The tour is interactive. Click the Run button now (or type `shift-enter`) to compile and run the program on a remote server. The result is displayed below the code.

These example programs demonstrate different aspects of Go. The programs in the tour are meant to be starting points for your own experimentation.

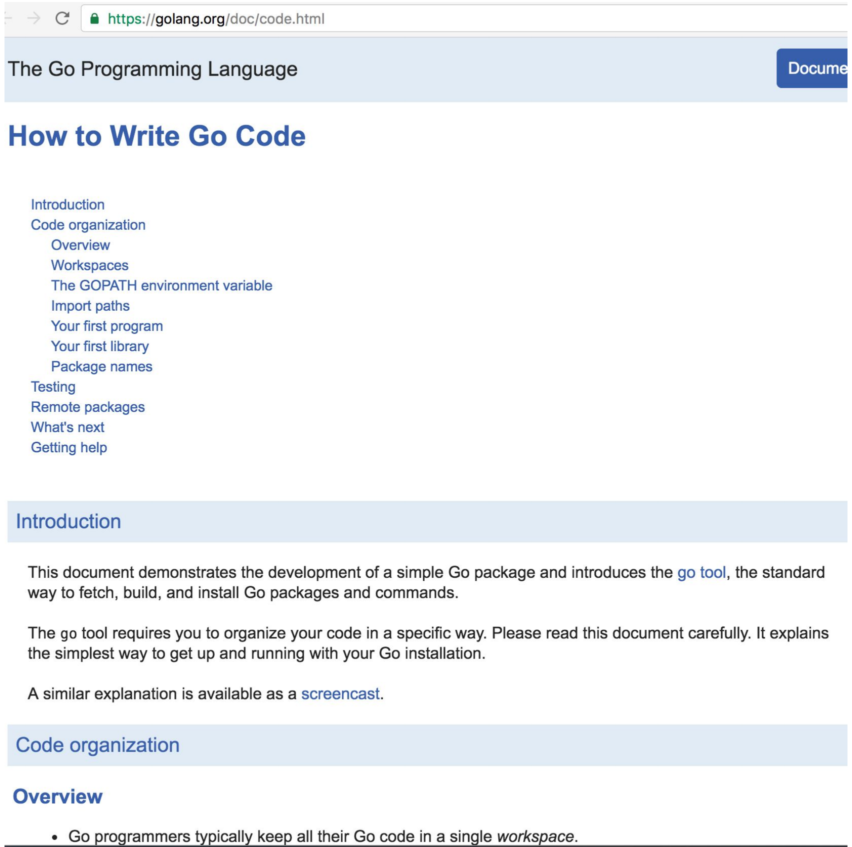
Edit the program and run it again.

Note that when you click on Format or `ctrl-enter` the text in the editor is formatted using the gofmt tool. You can switch syntax highlighting on and off by clicking on the syntax button.

When you're ready to move on, click the right arrow below or type the `PageDown` key.

```go
package main

import "fmt"

func main() {
    fmt.Println("Hello, 世界")
}
```

How to write Go Code



How to Write Go Code

Introduction

This document demonstrates the development of a simple Go package and introduces the go tool, the standard way to fetch, build, and install Go packages and commands.

The `go` tool requires you to organize your code in a specific way. Please read this document carefully. It explains the simplest way to get up and running with your Go installation.

A similar explanation is available as a screencast.

Code organization

Overview

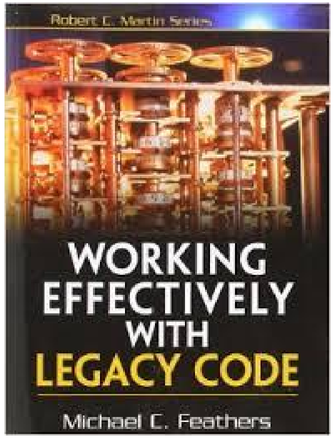
- Go programmers typically keep all their Go code in a single *workspace*.

Reorganização

- Pacotes
  - Cacoetes de outras linguagens
  - Exceções
  - Injeção de dependências
  - Idiomas que não se encaixavam

Reorganização

- Testes?
    - Neurologista nunca ouviu falar.
    - Working Effectively With Legacy Code
    - Mas acabou de nascer e já é legacy?



Problemas

- Muita memória
- Programa não subia
- Primeiras aventuras com profiling
  - *go tool pprof*
    - https://blog.golang.org/profiling-go-programs

Problemas

- Como lidar com o genoma humano inteiro?
  - 3+ bilhões de bases (ACTG)
  - 3 gigabases == gigabytes sendo carregados em memória assim que o programa subia
- Chrome + ABCD =

Soluções

- Não carregar genoma na memória e ler pedaços necessários direto do disco
- Pacote *os* da stdlib ao resgate!
    - *os.Open*
    - *map[Cromossomo]os.File*
    - *file.Seek*
- Segunda semana, ficou no ar meses

Crescendo

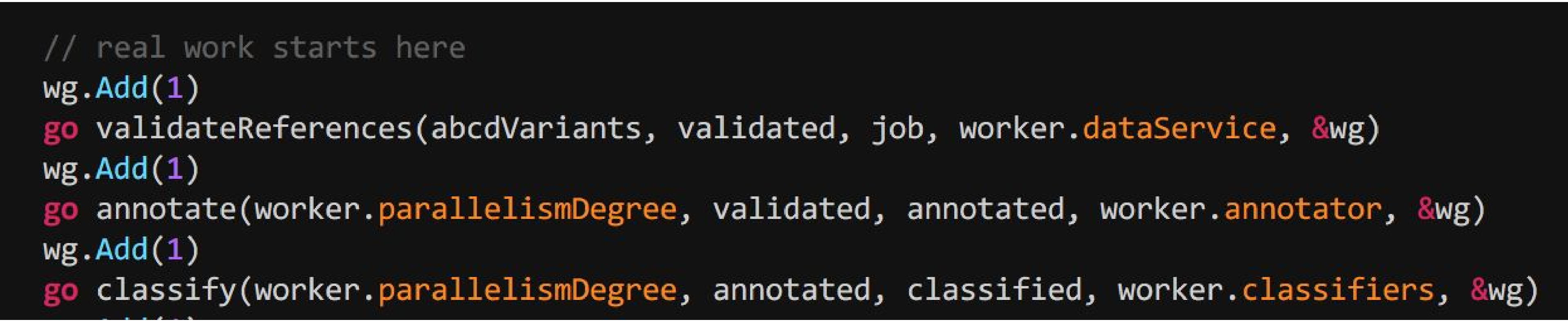
- Como consultar diversas bases?
- Partes direto do disco. Stdlib resolveu:
    - *os*
    - *encoding/csv*
    - *errors*
    - *io*
    - *bufio*

Crescendo

- Bancos de dados
  - MongoDB: https://gopkg.in/mgo.v2 - Gustavo Niemeyer
  - Facílimo de usar
- Nesse ponto tínhamos tudo na mão

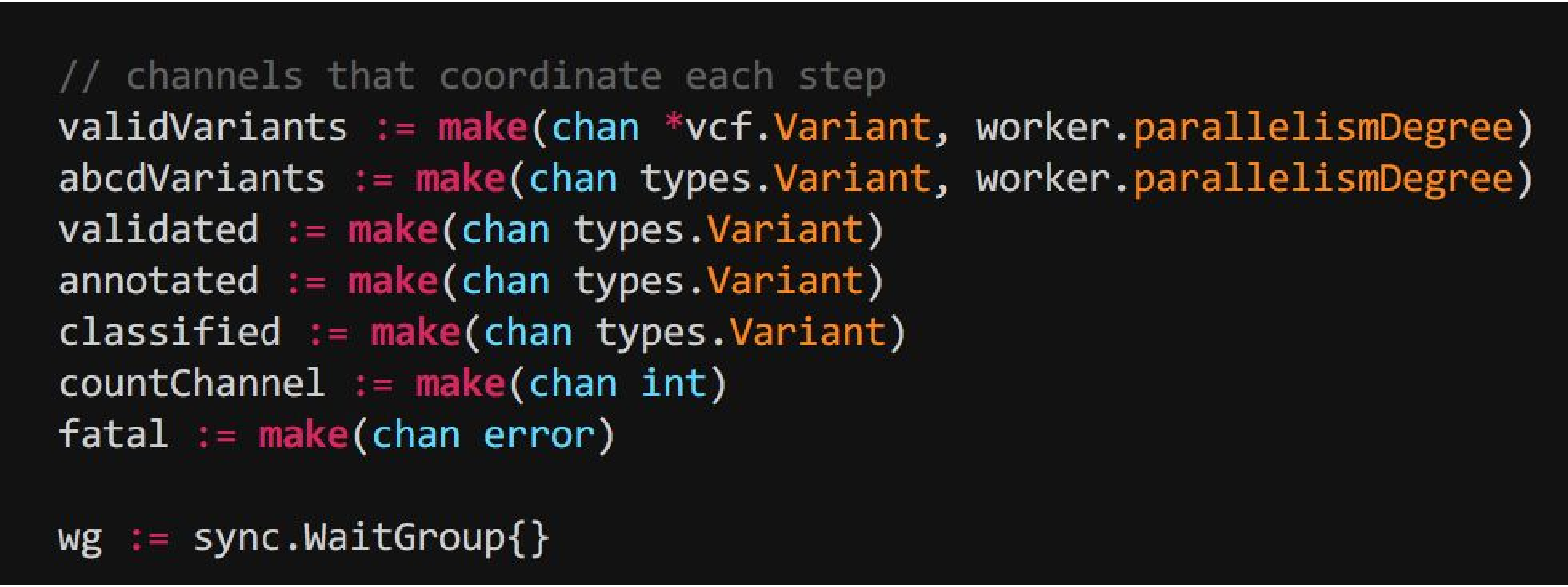
Concorrência

- Resolvido nativamente
- *Channels* e *goroutines* simples de entender



```
// real work starts here
wg.Add(1)
go validateReferences(abcdVariants, validated, job, worker.dataService, &wg)
wg.Add(1)
go annotate(worker.parallelismDegree, validated, annotated, worker.annotator, &wg)
wg.Add(1)
go classify(worker.parallelismDegree, annotated, classified, worker.classifiers, &wg)
```

Concorrência



```go
// channels that coordinate each step
validVariants := make(chan *vcf.Variant, worker.parallelismDegree)
abcdVariants := make(chan types.Variant, worker.parallelismDegree)
validated := make(chan types.Variant)
annotated := make(chan types.Variant)
classified := make(chan types.Variant)
countChannel := make(chan int)
fatal := make(chan error)

wg := sync.WaitGroup{}
```

Novas etapas

- Linguagem “saiu” da frente
- Em vez de perder tempo lutando com a linguagem ela nos permitiu pensar no nosso problema
- Dali pra frente:
    - Bioinformática, genética e computação

Amadurecendo

- Docker
- Microserviços
- Elasticsearch
- Múltiplas ferramentas de linha de comando
- Modularização
- Open source

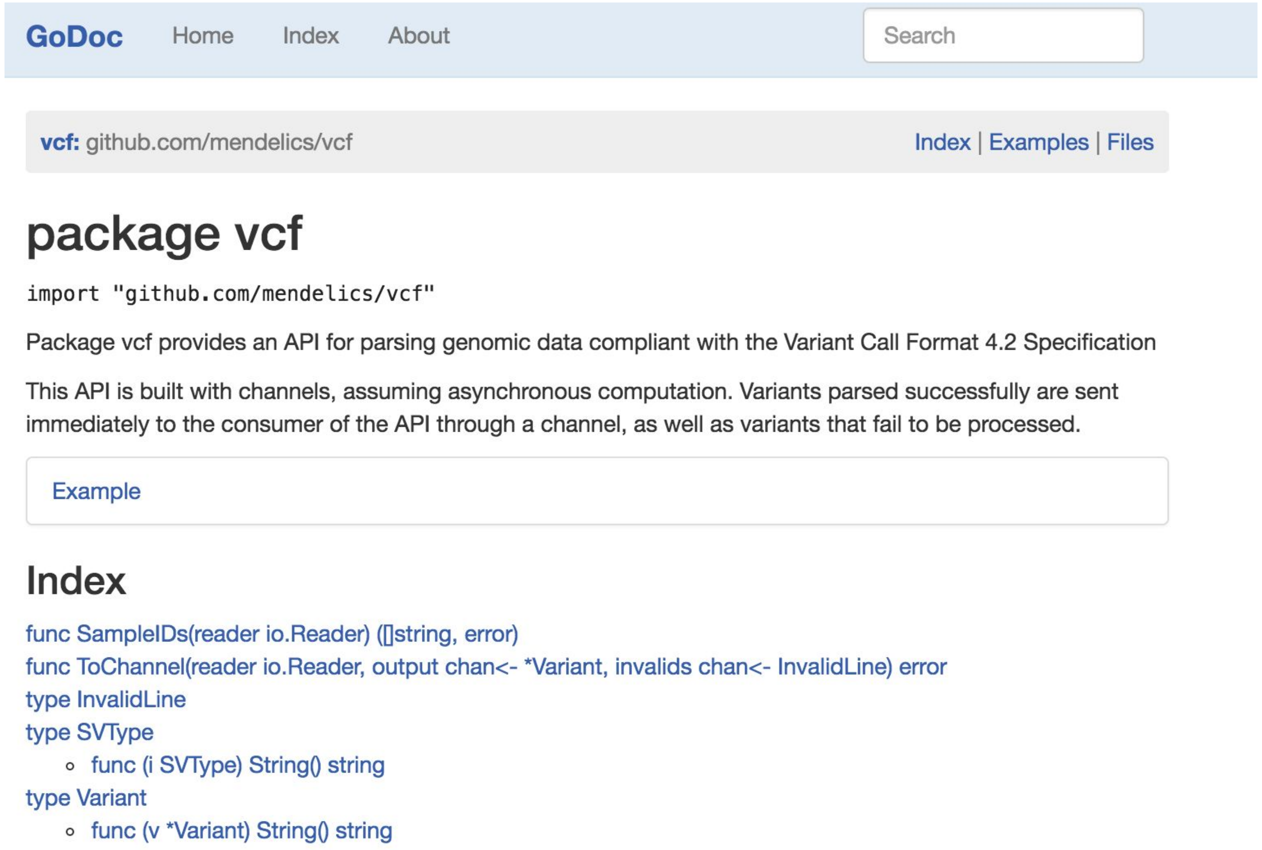
github.com/mendelics/garoa

- Apache Storm
- Encaixar funções como peças de LEGO
- Paralelização em larga escala
- Channels e goroutines
- 100% de CPU
- Não deu muito certo
- interface{} - oh no

github.com/mendelics/vcf

- Channels novamente
- Docs completos sem dificuldade
  - Incluindo examples
- Segue a especificação do formato VCF
  - io.Reader entra
  - Variantes saem

github.com/mendelics/vcf + GoDoc



vcf: github.com/mendelics/vcf

Index | Examples | Files

package vcf

```
import "github.com/mendelics/vcf"
```

Package vcf provides an API for parsing genomic data compliant with the Variant Call Format 4.2 Specification

This API is built with channels, assuming asynchronous computation. Variants parsed successfully are sent immediately to the consumer of the API through a channel, as well as variants that fail to be processed.

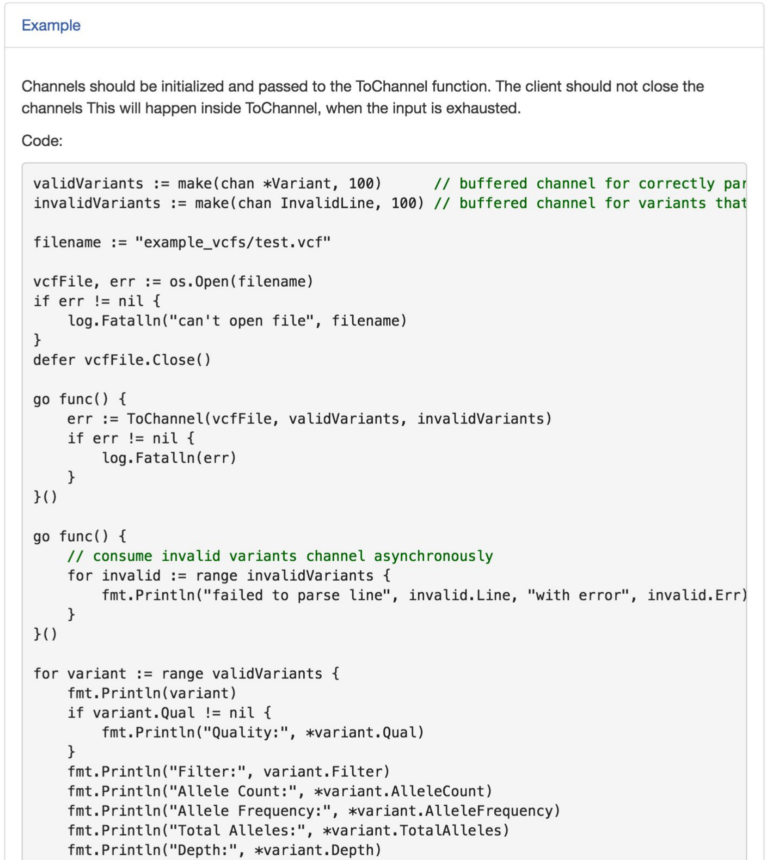
Example

Index

func SampleIDs(reader io.Reader) ([]string, error)
func ToChannel(reader io.Reader, output chan<- *Variant, invalids chan<- InvalidLine) error
type InvalidLine
type SVType
- func (i SVType) String() string

type Variant
- func (v *Variant) String() string

Exemplo é um teste

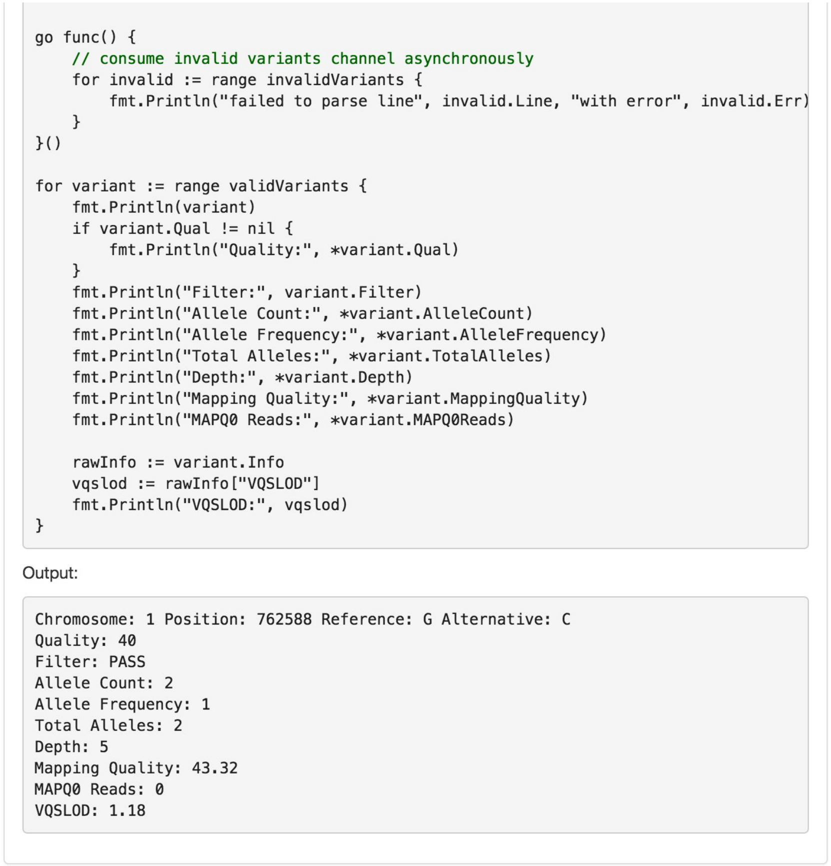


Example

Channels should be initialized and passed to the ToChannel function. The client should not close the channels This will happen inside ToChannel, when the input is exhausted.

Code:

```
validVariants := make(chan *Variant, 100)      // buffered channel for correctly par
invalidVariants := make(chan InvalidLine, 100) // buffered channel for variants that

filename := "example_vcfs/test.vcf"

vcfFile, err := os.Open(filename)
if err != nil {
    log.Fatalln("can't open file", filename)
}
defer vcfFile.Close()

go func() {
    err := ToChannel(vcfFile, validVariants, invalidVariants)
    if err != nil {
        log.Fatalln(err)
    }
}()

go func() {
    // consume invalid variants channel asynchronously
    for invalid := range invalidVariants {
        fmt.Println("failed to parse line", invalid.Line, "with error", invalid.Err)
    }
}()

for variant := range validVariants {
    fmt.Println(variant)
    if variant.Qual != nil {
        fmt.Println("Quality:", *variant.Qual)
    }
    fmt.Println("Filter:", variant.Filter)
    fmt.Println("Allele Count:", *variant.AlleleCount)
    fmt.Println("Allele Frequency:", *variant.AlleleFrequency)
    fmt.Println("Total Alleles:", *variant.TotalAlleles)
    fmt.Println("Depth:", *variant.Depth)
```



```
go func() {
    // consume invalid variants channel asynchronously
    for invalid := range invalidVariants {
        fmt.Println("failed to parse line", invalid.Line, "with error", invalid.Err)
    }
}()

for variant := range validVariants {
    fmt.Println(variant)
    if variant.Qual != nil {
        fmt.Println("Quality:", *variant.Qual)
    }
    fmt.Println("Filter:", variant.Filter)
    fmt.Println("Allele Count:", *variant.AlleleCount)
    fmt.Println("Allele Frequency:", *variant.AlleleFrequency)
    fmt.Println("Total Alleles:", *variant.TotalAlleles)
    fmt.Println("Depth:", *variant.Depth)
    fmt.Println("Mapping Quality:", *variant.MappingQuality)
    fmt.Println("MAPQ0 Reads:", *variant.MAPQ0Reads)

    rawInfo := variant.Info
    vqslod := rawInfo["VQSLOD"]
    fmt.Println("VQSLOD:", vqslod)
}
```

Output:

```
Chromosome: 1 Position: 762588 Reference: G Alternative: C
Quality: 40
Filter: PASS
Allele Count: 2
Allele Frequency: 1
Total Alleles: 2
Depth: 5
Mapping Quality: 43.32
MAPQ0 Reads: 0
VQSLOD: 1.18
```

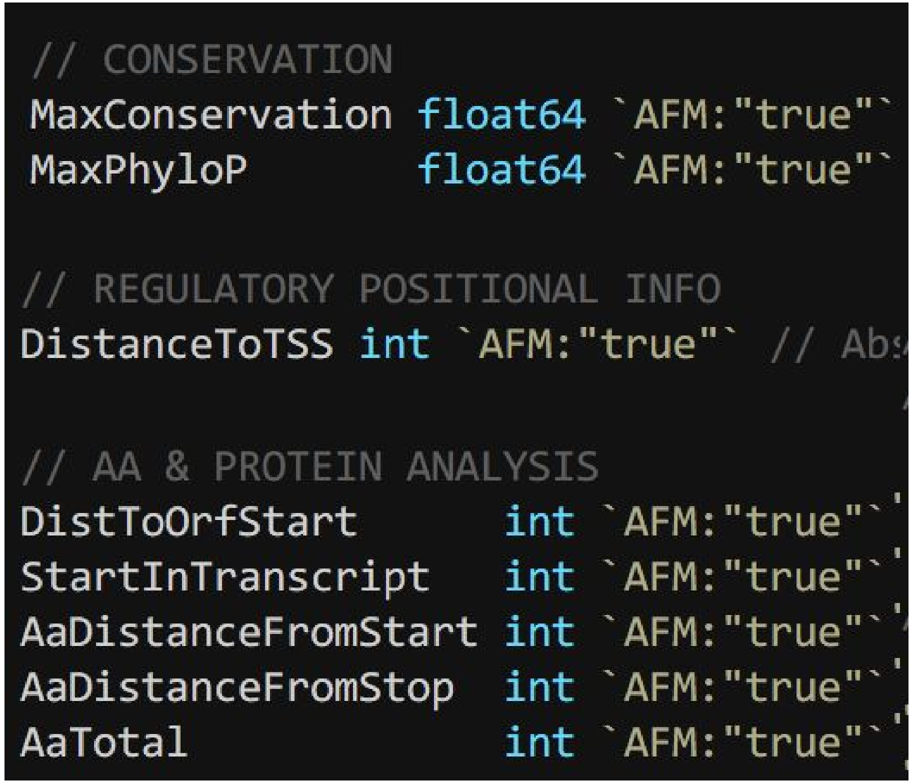
Machine Learning em Go

- Pulo do gato: prever mutações que causam doenças
- RandomForests™
- github.com/ryanbressler/CloudForest
- outros projetos na área:
    - https://github.com/sjwhitworth/golearn
    - https://github.com/avelino/awesome-go#machine-learning

Machine Learning em Go

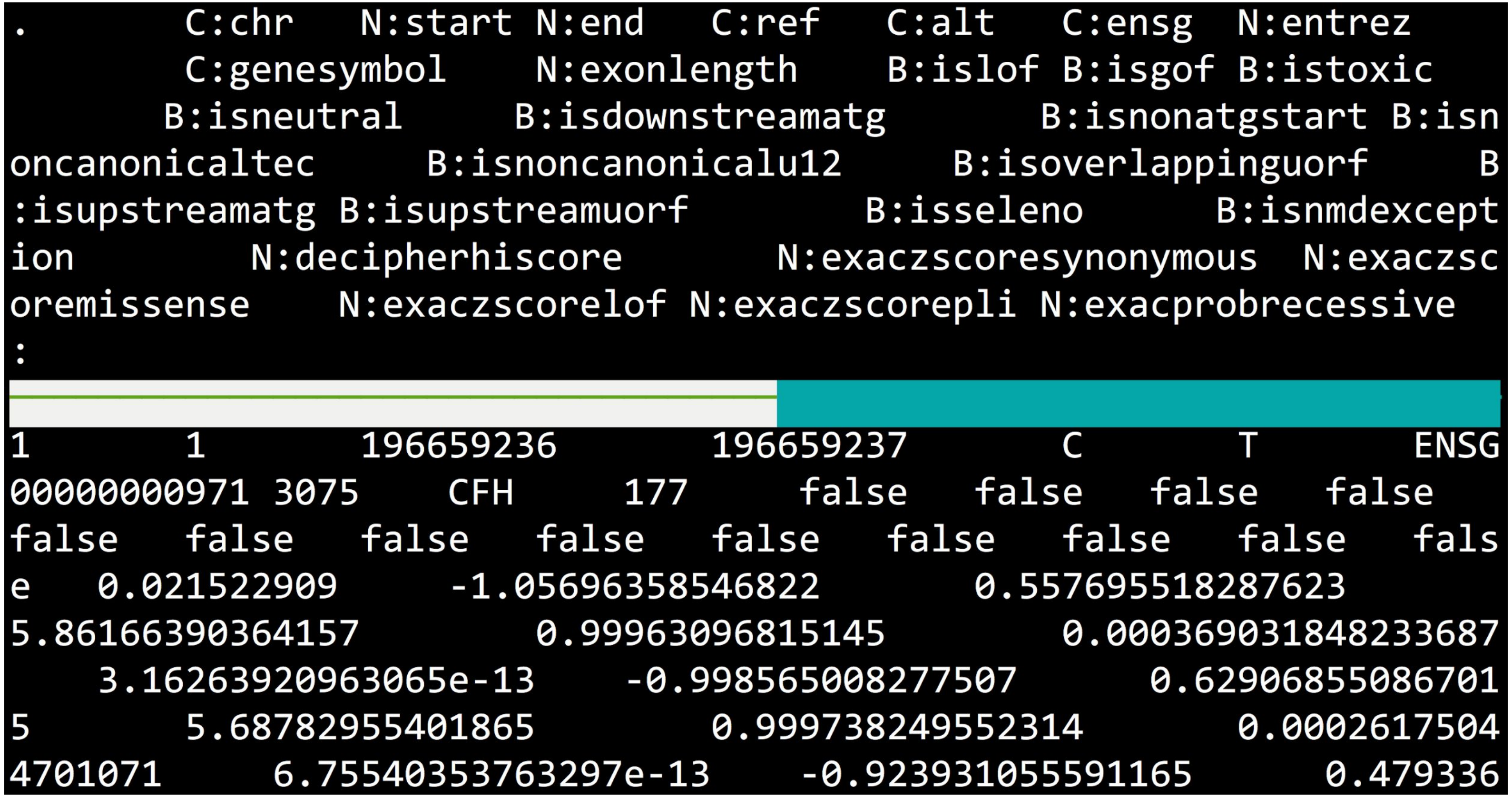
- Como passar dados do ABCD para o CloudForest?
  - Reflection
  - Struct Tags
- Mais uma vez, só stdlib

Reflection + Struct Tags



```
 // CONSERVATION
 MaxConservation float64 `AFM:"true"`
 MaxPhyloP       float64 `AFM:"true"`

// REGULATORY POSITIONAL INFO
DistanceToTSS int `AFM:"true"` // Ab

// AA & PROTEIN ANALYSIS
DistToOrfStart      int `AFM:"true"`
StartInTranscript   int `AFM:"true"`
AaDistanceFromStart int `AFM:"true"`
AaDistanceFromStop  int `AFM:"true"`
AaTotal             int `AFM:"true"`
```

Variante vira uma matriz

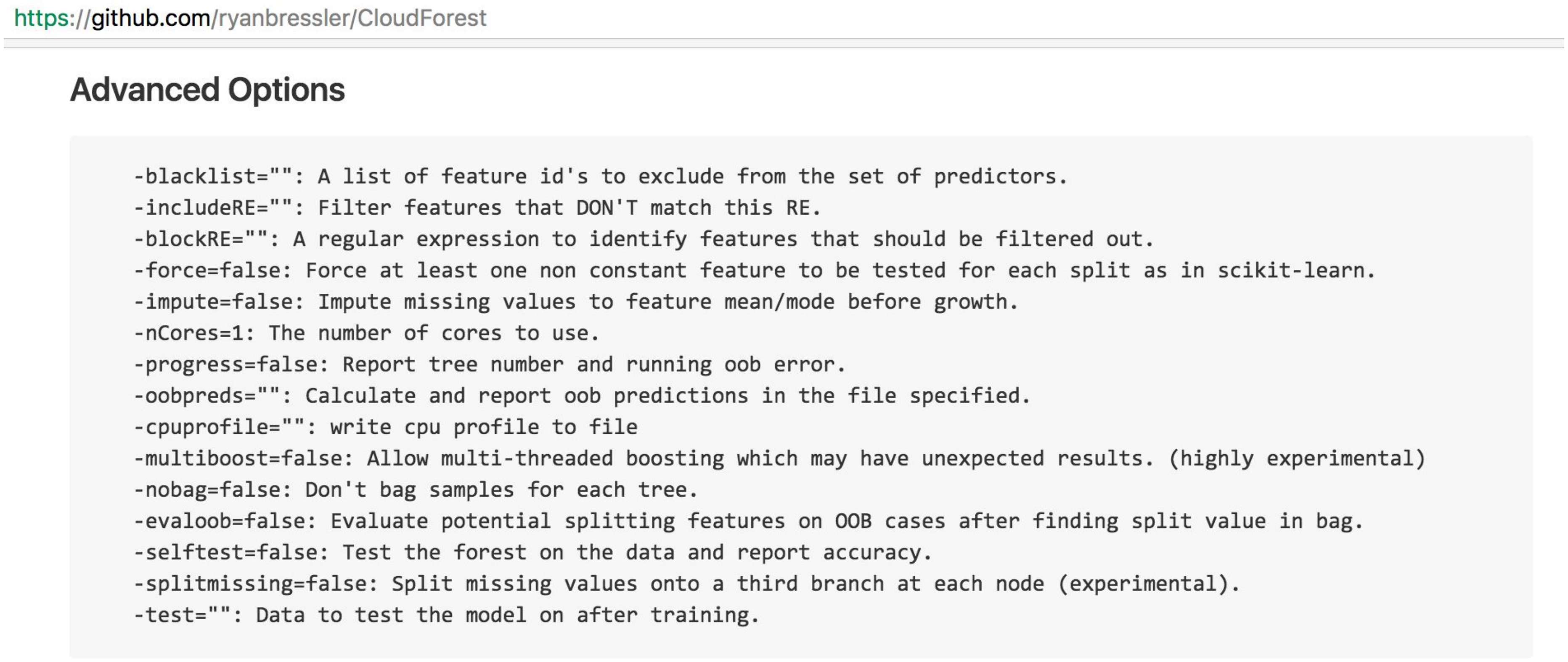


```
.       C:chr   N:start N:end   C:ref   C:alt   C:ensg  N:entrez
        C:genesymbol    N:exonlength    B:islof B:isgof B:istoxic
       B:isneutral     B:isdownstreamatg       B:isnonatgstart B:isn
oncanonicaltec     B:isnoncanonicalu12     B:isoverlappinguorf     B
:isupstreamatg B:isupstreamuorf        B:isseleno      B:isnmdexcept
ion          N:decipherhiscore       N:exaczscoresynonymous  N:exaczsc
oremissense    N:exaczscorelof N:exaczscorepli N:exacprobrecessive
:

1       1       196659236       196659237       C       T       ENSG
00000000971 3075    CFH     177     false   false   false   false
false   false   false   false   false   false   false   false   fals
e   0.021522909     -1.05696358546822       0.557695518287623
5.86166390364157        0.99963096815145        0.000369031848233687
    3.16263920963065e-13    -0.998565008277507      0.629068550867O1
5       5.68782955401865        0.999738249552314       0.0002617504
4701071     6.75540353763297e-13    -0.92393105559165       0.479336
```

Machine Learning em Go

- Tentativa e erro
- Construção de vários modelos
- CloudForest tem dezenas de flags

CloudForest



https://github.com/ryanbressler/CloudForest

Advanced Options

```
-blacklist="": A list of feature id's to exclude from the set of predictors.
-includeRE="": Filter features that DON'T match this RE.
-blockRE="": A regular expression to identify features that should be filtered out.
-force=false: Force at least one non constant feature to be tested for each split as in scikit-learn.
-impute=false: Impute missing values to feature mean/mode before growth.
-nCores=1: The number of cores to use.
-progress=false: Report tree number and running oob error.
-oobpreds="": Calculate and report oob predictions in the file specified.
-cpuprofile="": write cpu profile to file
-multiboost=false: Allow multi-threaded boosting which may have unexpected results. (highly experimental)
-nobag=false: Don't bag samples for each tree.
-evaloob=false: Evaluate potential splitting features on OOB cases after finding split value in bag.
-selftest=false: Test the forest on the data and report accuracy.
-splitmissing=false: Split missing values onto a third branch at each node (experimental).
-test="": Data to test the model on after training.
```

Machine Learning

- Mais de 90% de acerto
- Prever que uma variante é benigna quase sempre acerta
- Grande maioria não causa doença
- Filtro vai direto ao ponto
- Médico livre para lidar com os casos difíceis

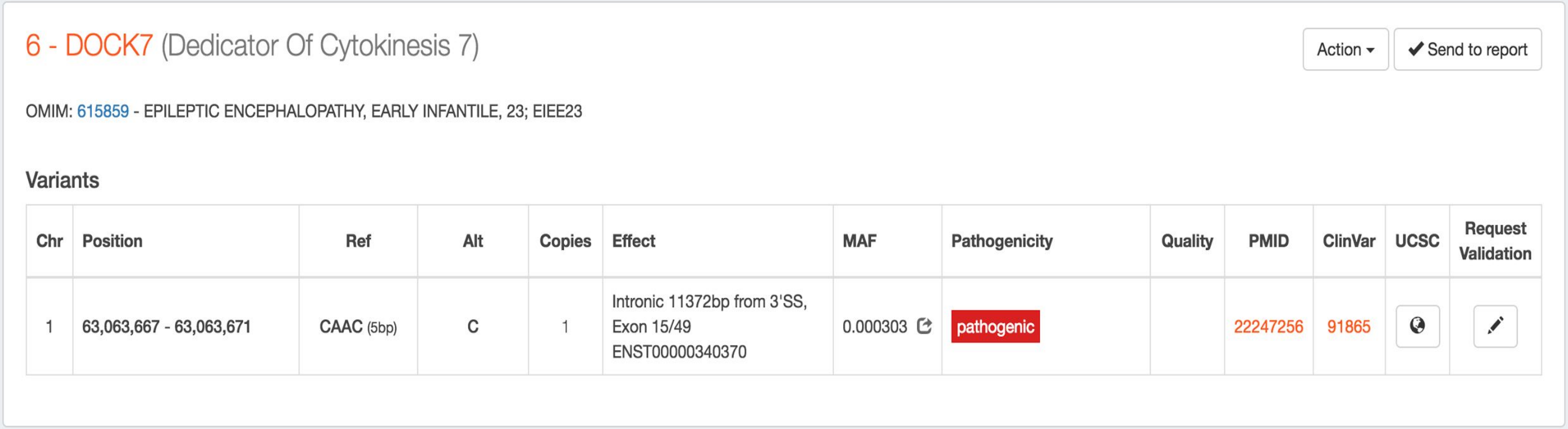
Evolução

- Segundo modelo de classificação
  - Lida mesmo com as áreas mais difíceis do genoma
- 100% dos pacientes passa pelo Abracadabra
- Quando médico vai para a análise já está tudo pronto

Hoje

- Infraestrutura toda baseada em Docker
- Máquinas lançadas on demand na Amazon para cada exame
- Começando a experimentar com Tensor Flow
- Mais de um ano de uso em produção, médicos se recusam a voltar atrás.

Hoje



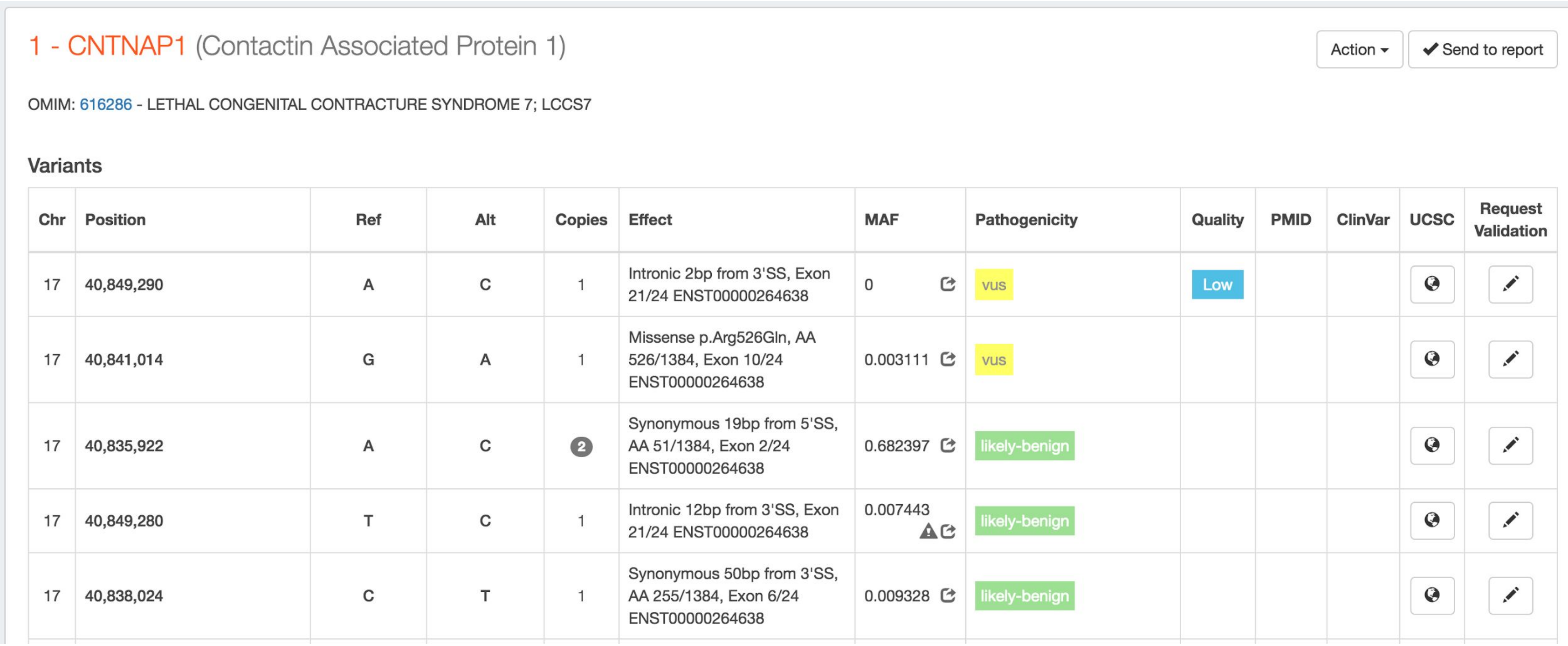
6 - DOCK7 (Dedicator Of Cytokinesis 7)

OMIM: 615859 - EPILEPTIC ENCEPHALOPATHY, EARLY INFANTILE, 23; EIEE23

Variants

Chr	Position	Ref	Alt	Copies	Effect	MAF	Pathogenicity	Quality	PMID	ClinVar	UCSC	Request Validation
1	63,063,667 - 63,063,671	CAAC (5bp)	C	1	Intronic 11372bp from 3'SS, Exon 15/49 ENST00000340370	0.000303	pathogenic		22247256	91865		

Hoje



1 - CNTNAP1 (Contactin Associated Protein 1)

OMIM: 616286 - LETHAL CONGENITAL CONTRACTURE SYNDROME 7; LCCS7

Variants

Chr	Position	Ref	Alt	Copies	Effect	MAF	Pathogenicity	Quality	PMID	ClinVar	UCSC	Request Validation
17	40,849,290	A	C	1	Intronic 2bp from 3'SS, Exon 21/24 ENST00000264638	0	vus	Low				
17	40,841,014	G	A	1	Missense p.Arg526Gln, AA 526/1384, Exon 10/24 ENST00000264638	0.003111	vus					
17	40,835,922	A	C	2	Synonymous 19bp from 5'SS, AA 51/1384, Exon 2/24 ENST00000264638	0.682397	likely-benign					
17	40,849,280	T	C	1	Intronic 12bp from 3'SS, Exon 21/24 ENST00000264638	0.007443	likely-benign					
17	40,838,024	C	T	1	Synonymous 50bp from 3'SS, AA 255/1384, Exon 6/24 ENST00000264638	0.009328	likely-benign					

Hoje

- Dados pré-processados levam a buscas precisas somente com resultados relevantes.
- Casos fáceis praticamente resolvidos pela máquina.
- Médico precisa analizar menos de 100 variantes de um total de 50 mil.

O que Go tornou possível

- Simples o suficiente para um médico sem experiência em programação começar
- Poderoso o suficiente para crescer
  - Microserviços
  - Containers Docker
  - Busca textual com Elasticsearch
  - Machine Learning

O que Go tornou possível

- Solução para a maior parte dos problemas já na stdlib e fácil de usar: manipulação de arquivos, parsing, sql, profiling, concorrência

O que Go tornou possível

- Comunidade forte e crescendo: mgo, vim-go, elastic, godep, caddy, gb, tsuru e por aí vai.
- Docker, Canonical, Hashicorp, Dropbox, Cloudflare, Uber, Globo.com, cada dia mais conversões.

Obrigado

Mendelics

twitter.com/vdemario

github.com/vdemario