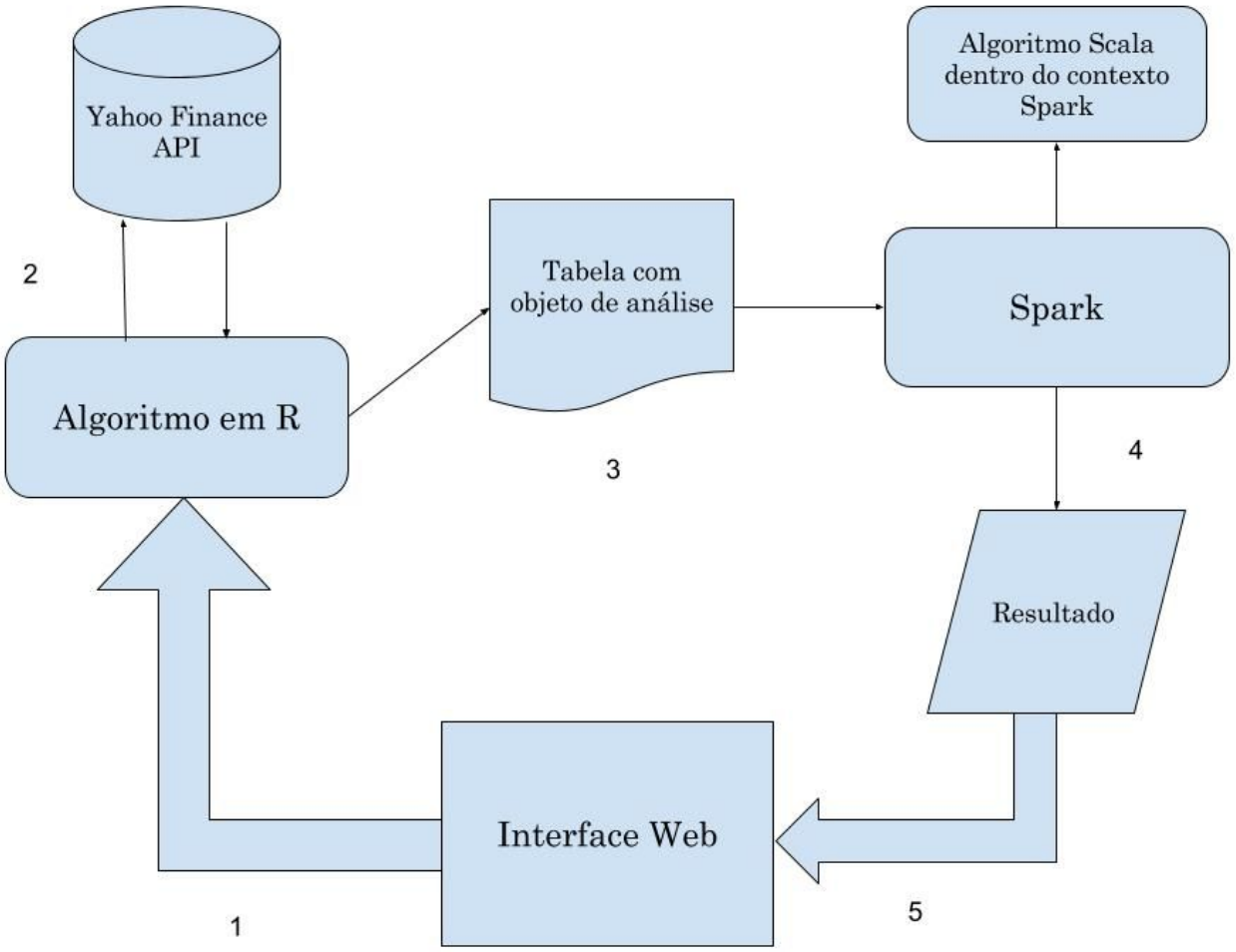
“Rule No.1: Never lose money. Rule No.2: Never forget rule No.1.”
Warren Buffet

UFRJ

Relatório Final do Trabalho de Big Data
Análise de Risco de ativos usando backtesting

- **Membros**
  - **Leonardo Almeida**
  - **Matheus Hoffmann**
  - **Rafael Mitre**

Overview do Sistema de Análise de Risco



Fluxo de Informações

1- Usuário escolhe o período sob análise: Data de fim e range de dias. Essas informações são os argumentos de entrada do script em R.

2 - Código em R faz a requisição para API da Yahoo Finance das séries históricas de interesse.

3 - Uma tabela, em formato CSV, é gerada com os dados de interesse, maior/menor preço da cotação e preço de início e fim.

4 - É enviada uma submissão ao spark para rodar no nó mestre a aplicação em Scala, que contém os algoritmos de análise e receberá como argumento o arquivo gerado pelo código R.

5 - O resultado da operação é lido via JavaScript e a biblioteca Chart.js plota os valores na interface para que o usuário veja o resultado.

Integração

A partir do momento que o usuário envia as opções da análise é disparado um script shell com todos os comandos necessários para análise.

```
$Rscript aquisition.R <Data fim> <Range>
$spark-submit --class "mddApp" --master local[4]
target/scala-2.11/simple-project_2.11-1.0.jar  "./../data/serieHistorica.csv"
```

***Lembrando que Data fim e range são os parametros escolhidos pelo usuário**

Algoritmos de Análise

Máximo Dropdown - Diferença entre maior e menor valor de cotação durante todo o períod

```
def mddCalc(df: DataFrame) : Integer = {
val princeDay_max = df.agg(max(df("High"))).first().getInt(0);
val princeDay_min = df.agg(min(df("Low"))).first().getInt(0);

return (princeDay_max - princeDay_min);
}
```

Desvio padrão - Variações em torno da média.

```
def desvPadraoCalc(df: DataFrame) : Double = {

val stdDev = df.agg(stddev_samp("High")).first.getDouble(0)
return (stdDev)
}
```

Média - Valor esperado da cotação.

```
def average(df: DataFrame) : Double = {
val avg = df.select(sum("High")/count("High")).first().getDouble(0)
return (avg)
}
```

Processamento Spark

É feito usando DataFrames, junto com as funções da biblioteca SQL. Usamos os módulos SQL dado que o objeto de análise está em formato tabular (CSV), então seria mais conveniente realizar as consultas em um dado estruturado.

```
def readPriceFromCsvData(path: String) : DataFrame = {
val spark = SparkSession.builder().master("local[*]").appName("Mdd
app").getOrCreate();
val esquema = StructType (
StructField("Index", IntegerType) ::
StructField("Open", IntegerType) ::
StructField("High", IntegerType) ::
```

```
StructField("Low", IntegerType) ::
StructField("Close", IntegerType) ::
StructField("Volume", IntegerType) ::
StructField("Adjusted", IntegerType) :: Nil )

val df_mdd = spark.sqlContext.read.schema(esquema).format("csv").option("header
"true").option("inferSchema", "true").load(path);

return (df_mdd);
```

Participação dos Componentes

1a parte - 18/04

Leonardo: Elaboração de requisitos funcionais/não funcionais.
Rafael: Contextualização e apresentação de métricas de risco.
Matheus: Definição de ferramentas utilizadas e pesquisa de formas de análise.

2a parte - 30/05

Leonardo: Implementação do algoritmo já no contexto do spark que calcula máximo dropdown.
Rafael: Algoritmo em R que puxa os dados API da Yahoo Finance.
Matheus: Elaboração da interface com parâmetros de entrada e "Hello World" da Google Chart API e código PHP que gera formato aceito automaticamente pela API.

3a parte - 27/06

Leonardo: Melhorias na interface, comunicação back/front e implementação do node serve
Suporte ao Rafael no Spark
Rafael: Implementação de novos algoritmos de métricas de risco: Média, Desvio Padrão, média de retornos e coeficiente de risco.
Matheus: Melhorias no código em R para integrar com a interface, suporte ao Rafael no Spark e elaboração deste documento.

Melhorias futuras

- **Deixar os dados armazenados no servidor e ir atualizando regularmente e/ou carregar dad de séries históricas pelo próprio Scala, sem necessidade do R, tornando o sistema mais simples.**

- **Melhoria do layout da interface, mais opções de parâmetros de entrada, principalmente de ativos.**

- **Métricas mais complexas de análise de risco: Curtose, distribuição Kernel e simulações de Monte Carlo e indicativo de previsão de risco futuro.**

Hospedagem e versionamento:

https://github.com/aracytopterm/AR_mercado_financeiro

- **/app**

Front end e node server.

- **/doc**

Documentação: Tese de referência inicial, Relatório Inicial e Relatório Final

- **/src**

/data : Algoritmos em R - v1: Estático para testar análise no Spark. v2: Integrado com front/back.
/mdd/src/main/scala : App Scala que realiza a análise das séries históricas.
/R-analysis : Códigos em R da tese de referência que não conseguiram ser rodados no SparkR.