Relatório de Dados (não fidedignos ou de baixa representatividade)

Relatório CGU (2016) & Portaria de Consolidação Nº 01/2017

O principal objetivo deste relatório é encontrar se há e por quais motivos ocorrem as divergências de dados entre as etapas do ciclo da assistência farmacêutica **pertinentes ao MS/DAF**, que se resumem ao ciclo da assistência farmacêutica no mesmo: (execução-DAF) **programação**, (acompanhamento-DLOG) **aquisição**, (acompanhamento-DLOG) **distribuição** e (acompanhamento-Estados/Municípios) **dispensação**.

***Obs.:** O que está sendo entregue de concreto neste estudo é um script que ao rodar é capaz de informar os desfechos abaixo atendendo à demanda da CGU:
Unidade Auditada: SECR. DE CIENCIA, TECNO. E INSUMOS ESTRATEGIC
Exercício: 2015
Município: Brasília - DF
Relatório no: 201600604
UCI Executora: SFC/DS/CGSAU - Coordenação-Geral de Auditoria da Área de Saúde

Os *outcomes* (desfechos) esperados deste estudo são:

1. Identificar os principais pontos de falhas/divergências do **ciclo da informação**;
2. **Munir o DAF** para ser capaz de responder, de forma estatística/padrão, em **qual etapa** do processo **ocorreu o erro** em caso de **desabastecimento** com motivo aparente;
3. **Criar hipóteses para prover massa de dados** que possa criar um **cenário melhor para o treino/aprendizado da Inteligência Artificial** de Compra e Distribuição. Isto porque os dados atuais não são capazes de ter significância estatística coerente entre as etapas do processo;

Métricas para a metodologia científica e a qualidade do estudo:

1. Eliminação dos possíveis vieses (erro de lógica ou computacional ou ainda parcialidade):
   A. extração dos dados - comparação com extração de dados de outras bases(InRad);
   B. manipulação dos dados (cálculos, transformações matemáticas e etc.);
   C. parcialidade da análise dos dados;
2. Estatísticas acerca da PORTARIA DE CONSOLIDAÇÃO Nº 1, DE 28 DE SETEMBRO DE 2017 (https://bvsms.saude.gov.br/bvs/saudelegis/gm/2017/prc0001_03_10_2017.html#ART193).
3. Comparação dos dados das bases do MS/DAF com bases externas para validação estatística (Estudo de Caso Padrão ouro Horus - Alagoas)).

Dados Suplementares:

1. Externo ao MS/DAF:
   A. Internacional:
      a. Organização Mundial da Saúde (**OMS**):
         i. Taxa/Estatística mundial de consumo de medicamentos por tipo/principio ativo/doença
         ii. Taxa/Estatística mundial de acometimento por doenças
   B. Nacional:
      a. Agência Nacional de Águas (**ANA**):
         i. Umidade Relativa do Ar (URA)
         ii. Temperatura média por município
         iii. Porcentagem Água e Esgoto Inadequado
         iv. Porcentagem Coleta de Lixo Inadequada

   b. Instituto Brasileiro de Geografia e Estatística (**IBGE**):
      i. População Projetada (CENSOs)
      ii. Índice de Desenvolvimento Humano por Município (IDHM)
   c. Instituto de Pesquisa Econômica Aplicada (**IPEA**):
      i. Índice de Vulnerabilidade Social (IVS)
      ii. IVS - Capital Humano
      iii. IVS - Infraestrutura
      iv. IVS - Renda e Trabalho
   d. Firjan (**SENAI/SESI/IEL/CIRJ**)
      i. Índice FIRJAN de Desenvolvimento Municipal (IFDM)
      ii. IFDM - Desenvolvimento Social
      iii. IFDM - Educação
      iv. IFDM - Renda
      v. IFDM - Saúde
2. Interno ao MS, porém externo ao DAF:
   A. DenaSUS:
      a. *Dados pertinentes principalmente ao FarmPop, porém o consumo de medicamentos similares da Coordenação Básica é uma métrica estatística para ambos (FarmPop e Compra Centralizada)
   B. DLOG:
      a. Dados sobre o processo de compra (volume efetivamente adquirido);
      b. Dados sobre o processo de entrega nos Centros de Distribuições - CDs (volume efetivamente recebido);
      c. Dados sobre pedido de envio por parte dos Estados/Municípios (volume efetivamente entregue);
   C. Fundo Nacional de Saúde (**FNS**):
      a. Valores financeiros efetivamente executados;
      b. Preços efetivamente praticados;
   D. SIOPS - Tesouro/**SIAF**/SIOP:
      a. Batimento dos valores **FNS** com os valores **SIOPS/SIAF**

(técnico) Load Dependencies

In [1]:

```
%matplotlib inline
%config InlineBackend.figure_format = 'retina'

import json
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import pymysql
from pandas.io import sql as psql
from sqlalchemy import create_engine
```

(técnico)Load Data from CSV

In [2]:

```
ws_legend = '(WS) Dispensação'
horus_legend = '(HÓRUS) Dispensação'

# Production Path
# arimaset_path = '/content/drive/My Drive/DAF-USP/Preditivo/Colab Notebooks/data/arimaset-quetiapina-alagoas.csv'
# microset_path = '/content/drive/My Drive/DAF-USP/Preditivo/Colab Notebooks/data/microset-quetiapina-alagoas.csv'

wsset_path = 'datasets/_daf-ws-dataset.csv'
horusset_path = 'datasets/_daf-horus-dataset.csv'
horus_gaesiset_path = 'datasets/_daf-horus-gaesi-dataset.csv'

h_ceafset_path = 'datasets/_daf-horus-ceaf.csv'
h_cgafb_cgafme_path = 'datasets/_daf-horus-cgafb-cgafme.csv'

wsset_df = pd.read_csv(wsset_path, index_col=1)
wsset_df.QTD = wsset_df.QTD.astype(float)
wsset_df.QTD = pd.to_numeric(wsset_df.QTD, downcast='float')

horus_gaesiset_df = pd.read_csv(horus_gaesiset_path, index_col=1)
horus_gaesiset_df.QTD = horus_gaesiset_df.QTD.astype(float)
horus_gaesiset_df.QTD = pd.to_numeric(horus_gaesiset_df.QTD, downcast='float')

# horusset_df = pd.read_csv(horusset_path, sep=';', warn_bad_lines=False, error_bad_lines=False)
# wsset_df['DT'] = pd.to_datetime(wsset_df['DT'], format='%Y%m', errors='ignore')
# horusset_df['DT'] = pd.to_datetime(horusset_df['DT'], format='%Y%m', errors='ignore')
```

Conjunto de Dados WebService

Quantidade de dispensação por medicamento (CATMAT) por municipio por mês (exercício)

In [3]:

```
wsset_df.head()
```

Out[3]:

NU_CATMAT	DT	QTD	MUNICIPIO
BR0233632U0062	01-2011	3.0	310970
BR0233632U0062	01-2011	4.0	510515
BR0266597	01-2011	60.0	330420
BR0266597	01-2011	120.0	330285
BR0266597	01-2011	150.0	421190

Conjunto de Dados Hórus

Quantidade de dispensação por medicamento (CATMAT) por municipio por mês (exercício) por Coordenação (1. CGAFB-CGAFME e 2. CEAF) e ORIGIN = h (hórus).

In [4]:

```
horus_gaesiset_df.head()
```

Out[4]:

NU_CATMAT	DT	QTD	MUNICIPIO	COORD	ORIGIN
BR0233632U0062	01-2011	3.0	310970	CGAFB-CGAFME	h
BR0233632U0062	01-2011	4.0	510515	CGAFB-CGAFME	h
BR0266597	01-2011	60.0	330420	CGAFB-CGAFME	h
BR0266597	01-2011	120.0	330285	CGAFB-CGAFME	h
BR0266597	01-2011	150.0	421190	CGAFB-CGAFME	h

In [5]:

```
# Checking if there is only horus
horus_gaesiset_df.groupby('ORIGIN').mean()
```

Out[5]:

ORIGIN	QTD	MUNICIPIO
h	1244.950317	306760.589049

In [6]:

```
h_ceafset_df = pd.read_csv(h_ceafset_path, sep=';', warn_bad_lines=False, error_
bad_lines=False)
h_cgafb_cgafme_df = pd.read_csv(h_cgafb_cgafme_path, sep=';', warn_bad_lines=Fal
se, error_bad_lines=False)

horusset_daf_df = h_ceafset_df.merge(h_cgafb_cgafme_df, how='outer').fillna(0) #
pd.concat([h_ceafset_df,h_cgafb_cgafme_df], axis=1)
horusset_daf_df
```

Out[6]:

	NU_COMPETENCIA_MOVIMENTACAO	NU_CATMAT	CO_IBGE_MOVIMENTACAO	C
0	201106.0	BR0271391U0042	530010.0	
1	201106.0	BR0271392U0042	530010.0	
2	201106.0	BR0272782U0042	530010.0	
3	201106.0	BR0272793U0042	530010.0	
4	201106.0	BR0288641U0042	530010.0	
...	...	...	...	
6392074	201910.0	BR0314517-5 INATIVO	351870.0	
6392075	201910.0	BR0314517-5 INATIVO	354410.0	
6392076	201910.0	BR0314517-5 INATIVO	411030.0	
6392077	201910.0	BR0314517-5 INATIVO	411960.0	
6392078	201910.0	BR0392113 (INATIVO)	260560.0	

6392079 rows × 14 columns

In [7]:

```
h_ceafset_df.head()
```

Out[7]:

	NU_COMPETENCIA_MOVIMENTACAO	NU_CATMAT	CO_IBGE_MOVIMENTACAO	QT_CNS
0	201106.0	BR0271391U0042	530010.0	
1	201106.0	BR0271392U0042	530010.0	
2	201106.0	BR0272782U0042	530010.0	
3	201106.0	BR0272793U0042	530010.0	
4	201106.0	BR0288641U0042	530010.0	

In [8]:

```
h_cgafb_cgafme_df.head()
```

Out[8]:

	NU_COMPETENCIA_MOVIMENTACAO	NU_CATMAT	CO_IBGE_MOVIMENTACAO	QT_CNS_PA(
0	201004.0	BR0267416	351380.0	
1	201004.0	BR0268082	351380.0	
2	201004.0	BR0268859	351380.0	
3	201004.0	BR0269963	351380.0	
4	201004.0	BR0270557	351380.0	

In [9]:

```
horusset_daf_df.head()
```

Out[9]:

	NU_COMPETENCIA_MOVIMENTACAO	NU_CATMAT	CO_IBGE_MOVIMENTACAO	QT_CNS
0	201106.0	BR0271391U0042	530010.0	
1	201106.0	BR0271392U0042	530010.0	
2	201106.0	BR0272782U0042	530010.0	
3	201106.0	BR0272793U0042	530010.0	
4	201106.0	BR0288641U0042	530010.0	

Primeira Validação - Vieses de Extração

Esta validação visa evidenciar se há massa significativa de falhas na extração e carregamento dos dados do MS/DAF (WS & Horus). Para tal, será utilizado como base uma massa de dados extraída do HC - InRad para comparação estatística de pontos de falha na extração:

Caso	Registros	Falhas	Taxa de Falha
AGENDADOS	14.959	0	0%
CUSTO_POR_SETOR	7.920	0	0%
EPIDEMIOLOGICO	250.024	13	0,0052%
PACIENTES	169.930	0	0%
PRODUCAO	217.251	34	0,016%

(técnico)WS Failure Records

In [10]:

```
wsset_df_size = len(wsset_df.index)
ws_original_records = sum(1 for line in open(wsset_path))

header_row = 1
ws_total_records = ws_original_records - header_row
ws_failure_records = ws_total_records - wsset_df_size
print('Total records: ', ws_total_records)
print('Failure records: ', ws_failure_records)
print('Failure records rate: ' + str((ws_failure_records/ws_original_records)*10
0) + '%')
```

```
Total records:  8733063
Failure records:  0
Failure records rate: 0.0%
```

(técnico)Horus Failure Records

In [11]:

```
horusset_daf_df_size = len(horusset_daf_df.index)

h_original_records = sum(1 for line in open(h_ceafset_path))
h_original_records = h_original_records + sum(1 for line in open(h_cgafb_cgafme_
path))

header_row = 1
h_total_records = h_original_records - header_row
h_failure_records = h_total_records - horusset_daf_df_size
daf_percent_failure = (h_failure_records*100)/h_original_records

print('DAF - Rows unimportable (csv process)')
print('Total records: ', h_total_records)
print('Failure records: ', h_failure_records)
print('Failure records rate: ' + '{0:.2f}'.format(daf_percent_failure) + '%')
```

```
DAF - Rows unimportable (csv process)
Total records:  6392080
Failure records:  1
Failure records rate: 0.00%
```

In [12]:

```
horusset_daf_nan_amount = np.sum(pd.isna(horusset_daf_df.QT_MOVIMENTACAO.values
))
daf_percent_nan = (horusset_daf_nan_amount*100)/h_total_records

print('DAF - Rows NaN (processing importable data)')
print('Total records: ', h_total_records)
print('Failure NaN records: ', horusset_daf_nan_amount)
print('Failure NaN records rate: ' + '{0:.2f}'.format(daf_percent_nan) + '%')
```

```
DAF - Rows NaN (processing importable data)
Total records:  6392080
Failure NaN records:  0
Failure NaN records rate: 0.00%
```

In [13]:

```
h_gaesi_daf_df_size = len(horus_gaesiset_df.index)
h_gaesi_original_records = sum(1 for line in open(horus_gaesiset_path))

h_gaesi_total_records = h_gaesi_original_records - header_row
h_gaesi_failure_records = h_gaesi_total_records - h_gaesi_daf_df_size

gaesi_percent_failure = (h_gaesi_failure_records*100)/h_gaesi_total_records

print('GAESI - Rows unimportable (csv process)')
print('Total records: ', h_gaesi_total_records)
print('Failure records: ', h_gaesi_failure_records)
print('Failure records rate: ' + '{0:.2f}'.format(gaesi_percent_failure) + '%')
```

```
GAESI - Rows unimportable (csv process)
Total records:  6377289
Failure records:  0
Failure records rate: 0.00%
```

In [14]:

```
horusset_gaesi_nan_amount = np.sum(pd.isna(horus_gaesiset_df.QTD.values))
gaesi_percent_nan = (horusset_gaesi_nan_amount*100)/h_gaesi_total_records

print('DAF - Rows NaN (processing importable data)')
print('Total records: ', h_gaesi_total_records)
print('Failure NaN records: ', horusset_gaesi_nan_amount)
print('Failure NaN records rate: ' + '{0:.2f}'.format(gaesi_percent_nan) + '%')
```

```
DAF - Rows NaN (processing importable data)
Total records:  6377289
Failure NaN records:  0
Failure NaN records rate: 0.00%
```

In [15]:

```
# pd.DataFrame().head()
pd.options.display.float_format = '{:,.2f}'.format
diff_df = pd.DataFrame(columns=["DataSource", "Total Records", "Failure Records"
, "Failure Rate(%)", "Failure NaN", "NaN Rate(%)", "HealthyRecords"],
                       data=[['DAF', h_total_records, h_failure_records, daf_per
cent_failure, horusset_daf_nan_amount, daf_percent_nan, h_total_records-horusset
_daf_nan_amount],
                             ['GAESI', h_gaesi_total_records, h_gaesi_failure_re
cords, gaesi_percent_failure, horusset_gaesi_nan_amount, gaesi_percent_nan, h_ga
esi_total_records-horusset_gaesi_nan_amount]])
diff_df.head()
```

Out[15]:

	DataSource	Total Records	Failure Records	Failure Rate(%)	Failure NaN	NaN Rate(%)	HealthyRecords
0	DAF	6392080	1	0.00	0	0.00	6392080
1	GAESI	6377289	0	0.00	0	0.00	6377289

In [16]:

```
print('GAESI improved DAF Data: ', diff_df['HealthyRecords'][1] - diff_df['Healt
hyRecords'][0])
```

```
GAESI improved DAF Data:  -14791
```

Resultado - Primeira Validação - Vieses de Extração

Como é possível visualizar na última tabela, logo acima, o trabalho do GAESI obteve 0% de falhas de importação, frente a 7,22% de falha do DAF.
Posteriormente o GAESI obteve 0% de **falha** NaN (valores esperados como numérico e não eram) frente a **86,01%** de falha do DAF.
O trabalho do **GAESI recuperou** 5.482.774 (13,99%) dos dados do DAF.

Segunda Validação - Vieses de Manipulação dos Dados

Após a manipulação para recuperar os dados, é possível ver que os **dados do GAESI representam 99,76%** dos dados do MS/DAF, com **14.791 registros com perda total** (não sendo aptos a serem recuperados). Ou seja, o GAESI recuperou todos os dados viáveis e ignorou os dados inviáveis de serem recuperados. Vale ressaltar que estes **14.791 registros com perda total** representam dados informados de forma totalmente não coerente com as especificações do MS, ou seja, **os municípios enviaram dados sujos (qualquer coisa aleatória de qualquer jeito)**.

Obs.: estas métricas se referem ao Hórus, uma vez que os dados do WS foram extraídos pelo GAESI e representa 100% de extração.

Imparcialidade na análise dos dados

Haja visto que o trabalho foi desenvolvido utilizando uma metodologica científica na qual foi levado em consideração os pontos de falha de ambas as partes (ministério da saúde, estados e municípios), este resultado é isento de parcialidade.

(técnico)Load Data (County & UFs)

In [17]:

```
# Load Municipios & UFs
municipios_json = {}
with open('data/municipios.json', encoding='utf-8-sig') as json_file:
    municipios_json = json.load(json_file)

ufs_json = {}
with open('data/ufs.json', encoding='utf-8-sig') as json_file:
    ufs_json = json.load(json_file)

ufs_id = {}
ufs_id_i = {}

for uf in ufs_json:
    ufs_id[str(uf['codigo_uf'])] = uf['uf']
    ufs_id_i[uf['codigo_uf']] = uf['uf']
```

(técnico)Database Connection

In [18]:

```
db_connection_str = 'mysql+pymysql://root:@localhost:3306/datasource_developmen
t'
db_connection = create_engine(db_connection_str)
```

In [19]:

```
amount_counties_query = 'SELECT COUNT(DISTINCT CO_MUNICIPIO_IBGE_EST) AS MUNICIP
IOS FROM datasource_development.bnafar_dispensations;'
db_disp_count_counties = pd.read_sql(amount_counties_query, con=db_connection)
```

In [20]:

```
amount_uf_counties_query = 'SELECT SUBSTRING(CO_MUNICIPIO_IBGE_EST, 1, 2) AS UF,
COUNT(DISTINCT CO_MUNICIPIO_IBGE_EST) AS MUNICIPIOS FROM datasource_development.
bnafar_dispensations GROUP BY UF;'
db_disp_count_uf_counties = pd.read_sql(amount_uf_counties_query, con=db_connect
ion)
```

In [21]:

```
brazil_total_counties = 5570
missing_counties = brazil_total_counties - db_disp_count_counties['MUNICIPIOS'][
0]
db_disp_count_counties_br = None
db_disp_count_counties_br = db_disp_count_counties.append(pd.DataFrame({"MUNICIP
IOS":[missing_counties]}))
db_disp_count_counties_br['TIPO'] = ['MS Recebe Dados', 'MS às Cegas']
db_disp_count_counties_br
```

Out[21]:

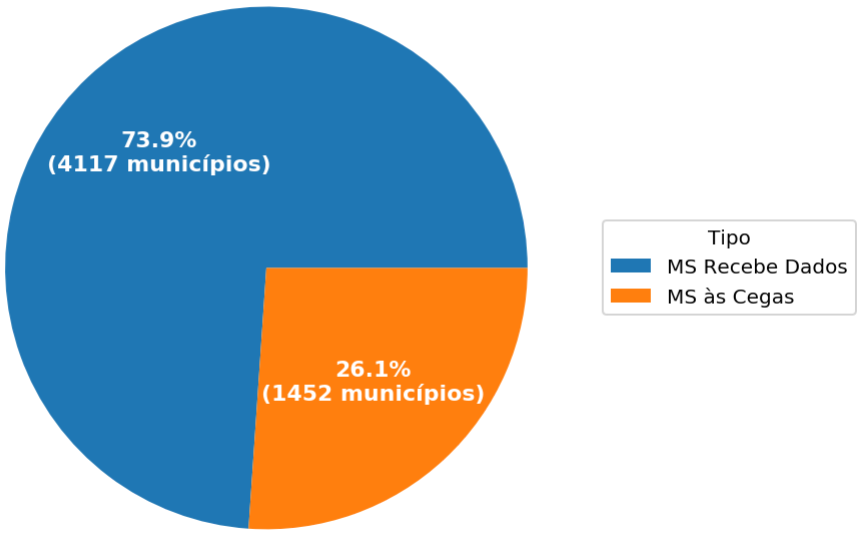
	MUNICIPIOS	TIPO
0	4117	MS Recebe Dados
0	1453	MS às Cegas

Quantidade de municípios (nível nacional) visíveis para o DAF

In [22]:

```
fig, ax = plt.subplots(figsize=(12, 6), subplot_kw=dict(aspect="equal"))

data = db_disp_count_counties_br['MUNICIPIOS']
types = db_disp_count_counties_br['TIPO']


def func(pct, allvals):
    absolute = int(pct/100.*np.sum(allvals))
    return "{:.1f}%\n({:d} municípios)".format(pct, absolute)


wedges, texts, autotexts = ax.pie(data, autopct=lambda pct: func(pct, data),
                                  textprops=dict(color="w"))

ax.legend(wedges, types,
          title="Tipo",
          loc="center left",
          bbox_to_anchor=(1, 0, 0.5, 1))

plt.setp(autotexts, size=11, weight="bold")

ax.set_title("Representatividade dos Dados (municípios Brasil vs municípios MS)"
)

plt.show()
```

Representatividade dos Dados (municípios Brasil vs municípios MS)



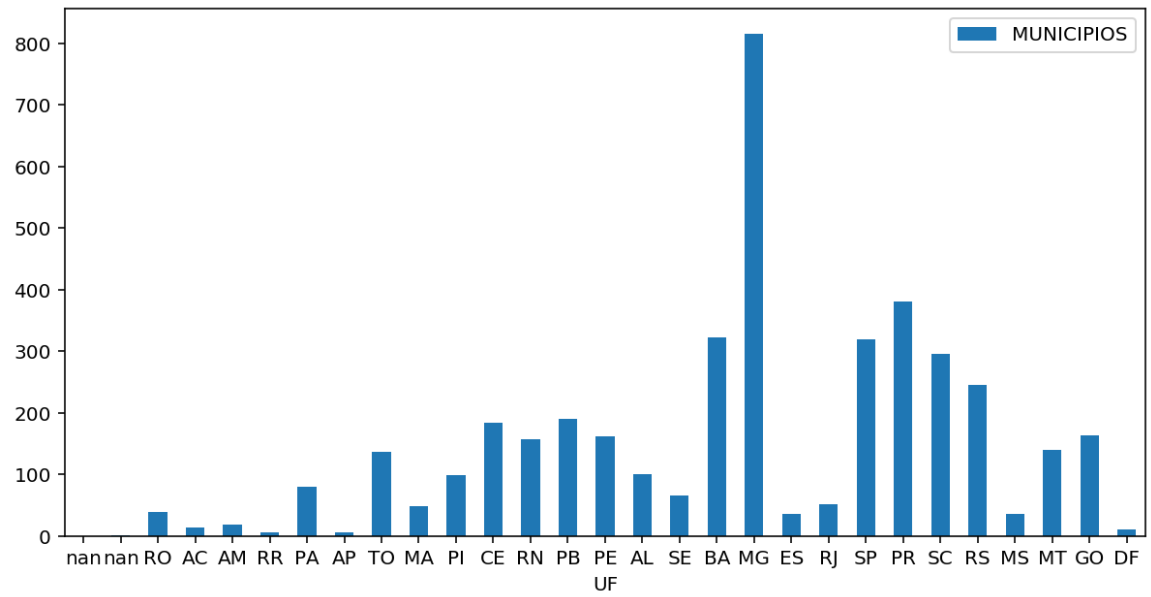
Quantidade de municípios por UF visíveis para o DAF

In [23]:

```
db_disp_count_uf_counties_sig = db_disp_count_uf_counties.copy()
db_disp_count_uf_counties_sig['UF'] = db_disp_count_uf_counties_sig['UF'].map(uf
s_id)
db_disp_count_uf_counties_sig.plot.bar(x='UF', y='MUNICIPIOS', rot=0, figsize=(1
0,5))
```

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1584b8b90>
```



Definição do Padrão Ouro Hórus & WS

A definição do Padrão Ouro se refere a:

1. Nível de UF (segundo tabela abaixo):
   A. Hórus = Ceará e Alagoas (empate técnico)
   B. WS = Santa Catarina

***Obs.:** utilizamos o Padrão Ouro Hórus (Alagoas) para os testes em nível de município pelo fato do ARIMA, único algoritmo já implantado e em uso para a programação automatizada, atuar somente em municípios que usam Hórus e somente para os medicamentos da coordenação Componente Especializado da Assistência Farmacêutica.

In [35]:

```
br_uf_municipios = {}
for municipio in municipios_json:
    cuf = municipio['codigo_uf']
    if cuf in br_uf_municipios:
        br_uf_municipios[cuf] = br_uf_municipios[cuf] + 1
    else:
        br_uf_municipios[cuf] = 1

ms_uf_municipios = {}
for ufbr in ufs_json:
    municipios = 0
    for ufms, m in zip(db_disp_count_uf_counties_sig['UF'], db_disp_count_uf_cou
nties_sig['MUNICIPIOS']):
        if ufms == ufbr['uf']:
            if ufms == 'DF':
                municipios = 1
            else:
                municipios = m
            break
    ms_uf_municipios[ufbr['codigo_uf']] = municipios
ms_uf_municipios

# s_ufs = pd.Series(ufs_id, dtype='U2')
s_ms_uf_municipios = pd.Series(ms_uf_municipios)
s_br_uf_municipios = pd.Series(br_uf_municipios)
municipios_br_vs_ms = pd.DataFrame({'Municipios BR': s_br_uf_municipios, 'Munici
pios MS': s_ms_uf_municipios})
municipios_br_vs_ms['UF'] = municipios_br_vs_ms.index.to_series().map(ufs_id_i)

# municipios_br_vs_ms['Abrangência (CNS - beneficiários)'] = ((municipios_br_vs_
ms['Municipios MS']*100)/municipios_br_vs_ms['Municipios BR'])
municipios_br_vs_ms['Abrangência (Municípios)'] = ((municipios_br_vs_ms['Municip
ios MS']*100)/municipios_br_vs_ms['Municipios BR'])
municipios_br_vs_ms.sort_values(by=['Abrangência (Municípios)'], ascending=False
)
```

Out[35]:

	Municipios BR	Municipios MS	UF	Abrangência (Municípios)
53	1	1	DF	100.00
42	295	295	SC	100.00
23	184	183	CE	99.46
27	102	101	AL	99.02
51	141	139	MT	98.58
17	139	136	TO	97.84
31	853	816	MG	95.66
41	399	381	PR	95.49
24	167	157	RN	94.01
28	75	66	SE	88.00
26	185	162	PE	87.57
25	223	190	PB	85.20
29	417	323	BA	77.46
11	52	39	RO	75.00
52	246	163	GO	66.26
12	22	14	AC	63.64
33	92	52	RJ	56.52
15	144	79	PA	54.86
35	645	319	SP	49.46
43	497	245	RS	49.30
32	78	35	ES	44.87
50	79	35	MS	44.30
22	224	99	PI	44.20
14	15	5	RR	33.33
16	16	5	AP	31.25
13	62	19	AM	30.65
21	217	48	MA	22.12

Desempate (AL vs CE) por CNES

O desempate para o empate técnico entre AL e CE se dará pela representatividade da quantidade de UBS participantes (fornecendo dados) ao MS/DAF.

***Obs.:** Os dados de CNES aqui apresentados são frutos de uma [análise exploratória da base de CNES (http://localhost:8888/notebooks/automated-programming/CNES%20BD.ipynb#Exploratory-Analysis-Columns)](http://localhost:8888/notebooks/automated-programming/CNES%20BD.ipynb#Exploratory-Analysis-Columns).

In [25]:

```
print('CO_IBGE & NU_COMPETENCIA_MOVIMENTACAO as CATEGORICAL')
horusset_daf_df = horusset_daf_df.dropna(subset=['CO_IBGE_MOVIMENTACAO'])
# horusset_daf_df.reset_index(drop=True)
horusset_daf_df.NU_COMPETENCIA_MOVIMENTACAO = horusset_daf_df.NU_COMPETENCIA_MOV
IMENTACAO.astype(int)
horusset_daf_df.NU_COMPETENCIA_MOVIMENTACAO = horusset_daf_df.NU_COMPETENCIA_MOV
IMENTACAO.astype(str)
horusset_daf_df.CO_IBGE_MOVIMENTACAO = horusset_daf_df.CO_IBGE_MOVIMENTACAO.asty
pe(int)
horusset_daf_df.CO_IBGE_MOVIMENTACAO = horusset_daf_df.CO_IBGE_MOVIMENTACAO.asty
pe(str)
horusset_daf_df.head()
```

```
CO_IBGE & NU_COMPETENCIA_MOVIMENTACAO as CATEGORICAL
```

Out[25]:

	NU_COMPETENCIA_MOVIMENTACAO	NU_CATMAT	CO_IBGE_MOVIMENTACAO	QT_CNS
0	201106	BR0271391U0042	530010	
1	201106	BR0271392U0042	530010	
2	201106	BR0272782U0042	530010	
3	201106	BR0272793U0042	530010	
4	201106	BR0288641U0042	530010	

Agrupamento CNES (Alagoas e Ceará)

In [36]:

```
# URL Download Data https://www.ibge.gov.br/apps/populacao/projecao/jsdados/al_faixas.js
ce_uf_ibge = '23'
al_uf_ibge = '27'
competencia = '201909'

ce_pop_2018_ibge = 9018764
al_pop_2018_ibge = 3307532

ce_counties = horusset_daf_df[horusset_daf_df['CO_IBGE_MOVIMENTACAO'].str.startswith(ce_uf_ibge)]
al_counties = horusset_daf_df[horusset_daf_df['CO_IBGE_MOVIMENTACAO'].str.startswith(al_uf_ibge)]

ce_counties_competencia = ce_counties[ce_counties['NU_COMPETENCIA_MOVIMENTACAO'] == competencia]
al_counties_competencia = al_counties[al_counties['NU_COMPETENCIA_MOVIMENTACAO'] == competencia]

ce_cns_amount = ce_counties_competencia.QT_CNS_PACIENTE.sum()
al_cns_amount = al_counties_competencia.QT_CNS_PACIENTE.sum()

ce_counties_competencia = ce_counties_competencia[ce_counties_competencia['DS_ORIGEM'] == 'CGAFB-CGAFME']
al_counties_competencia = al_counties_competencia[al_counties_competencia['DS_ORIGEM'] == 'CGAFB-CGAFME']

ce_counties_competencia_cp = ce_counties_competencia.copy()
ce_grp = ce_counties_competencia_cp.groupby(['CO_IBGE_MOVIMENTACAO'])['QT_CNES'].max().reset_index(name='COUNTS')
ce_cnes_amount = ce_grp.COUNTS.sum()

al_counties_competencia_cp = al_counties_competencia.copy()
al_grp = al_counties_competencia_cp.groupby(['CO_IBGE_MOVIMENTACAO'])['QT_CNES'].max().reset_index(name='COUNTS')
al_cnes_amount = al_grp.COUNTS.sum()

ce_pop_at_rate = (ce_cns_amount*100)/ce_pop_2018_ibge
al_pop_at_rate = (al_cns_amount*100)/al_pop_2018_ibge

ce_total_cnes = 591
al_total_cnes = 211

ce_cnes_rate = (ce_cnes_amount*100)/ce_total_cnes
al_cnes_rate = (al_cnes_amount*100)/al_total_cnes

ce_mun = municipios_br_vs_ms['Abrangência (Municípios)'][int(ce_uf_ibge)]
al_mun = municipios_br_vs_ms['Abrangência (Municípios)'][int(al_uf_ibge)]

ce_al_df = pd.DataFrame(columns=["Exercicio", "UF", "Pop. IBGE", "Pop. Horus (CNS)", "TotalCNES", "Qtd. CNES Horus", "Abrangência Pop. (CNS)", "Abrangência (Municípios)", "Abrangência CNES"],
                        data=[[competencia, 'CEARÁ', ce_pop_2018_ibge, ce_cns_amount, ce_total_cnes, ce_cnes_amount, ce_pop_at_rate, ce_mun, ce_cnes_rate],
                              [competencia, 'ALAGOAS', al_pop_2018_ibge, al_cns_amount, al_total_cnes, al_cnes_amount, al_pop_at_rate, al_mun, al_cnes_rate]])

ce_al_df_printable = ce_al_df.copy()
ce_al_df_printable["Pop. IBGE"] = ce_al_df_printable["Pop. IBGE"].apply(lambda x
```

```
: "{:,}".format(x))
ce_al_df_printable["Pop. Horus (CNS)"] = ce_al_df_printable["Pop. Horus (CNS)"].
apply(lambda x : "{:,}".format(x))
ce_al_df_printable["Qtd. CNES Horus"] = ce_al_df_printable["Qtd. CNES Horus"].ap
ply(lambda x : "{:,}".format(x))
ce_al_df_printable["TotalCNES"] = ce_al_df_printable["TotalCNES"].apply(lambda x
: "{:,}".format(x))
ce_al_df_printable["Abrangência Pop. (CNS)"] = ce_al_df_printable["Abrangência P
op. (CNS)"].apply(lambda x : '{0:.2f}'.format(x) + '%')
ce_al_df_printable["Abrangência (Municípios)"] = ce_al_df_printable["Abrangência
(Municípios)"].apply(lambda x : '{0:.2f}'.format(x) + '%')
ce_al_df_printable["Abrangência CNES"] = ce_al_df_printable["Abrangência CNES"].
apply(lambda x : '{0:.2f}'.format(x) + '%')
ce_al_df_printable.head()
```

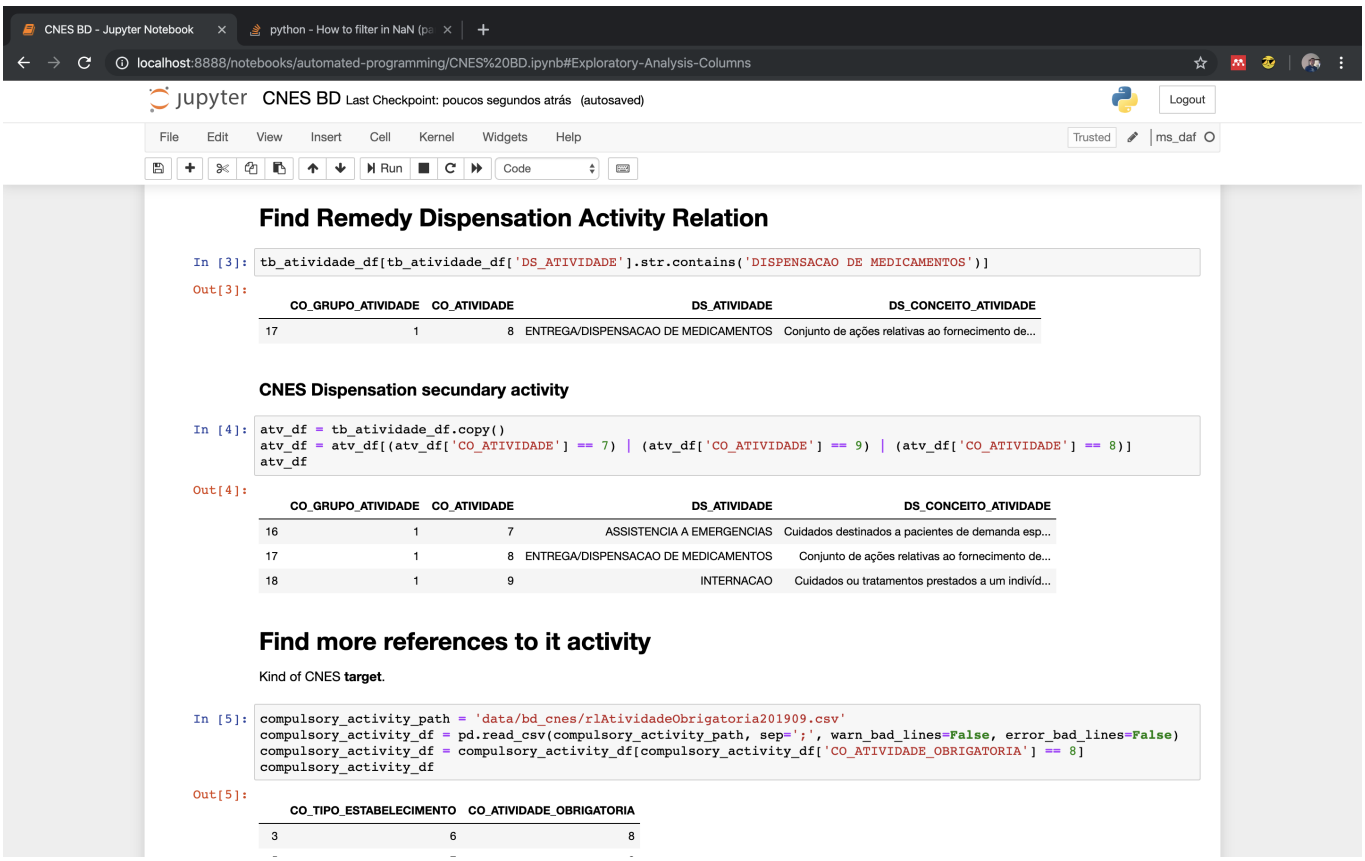
Out[36]:

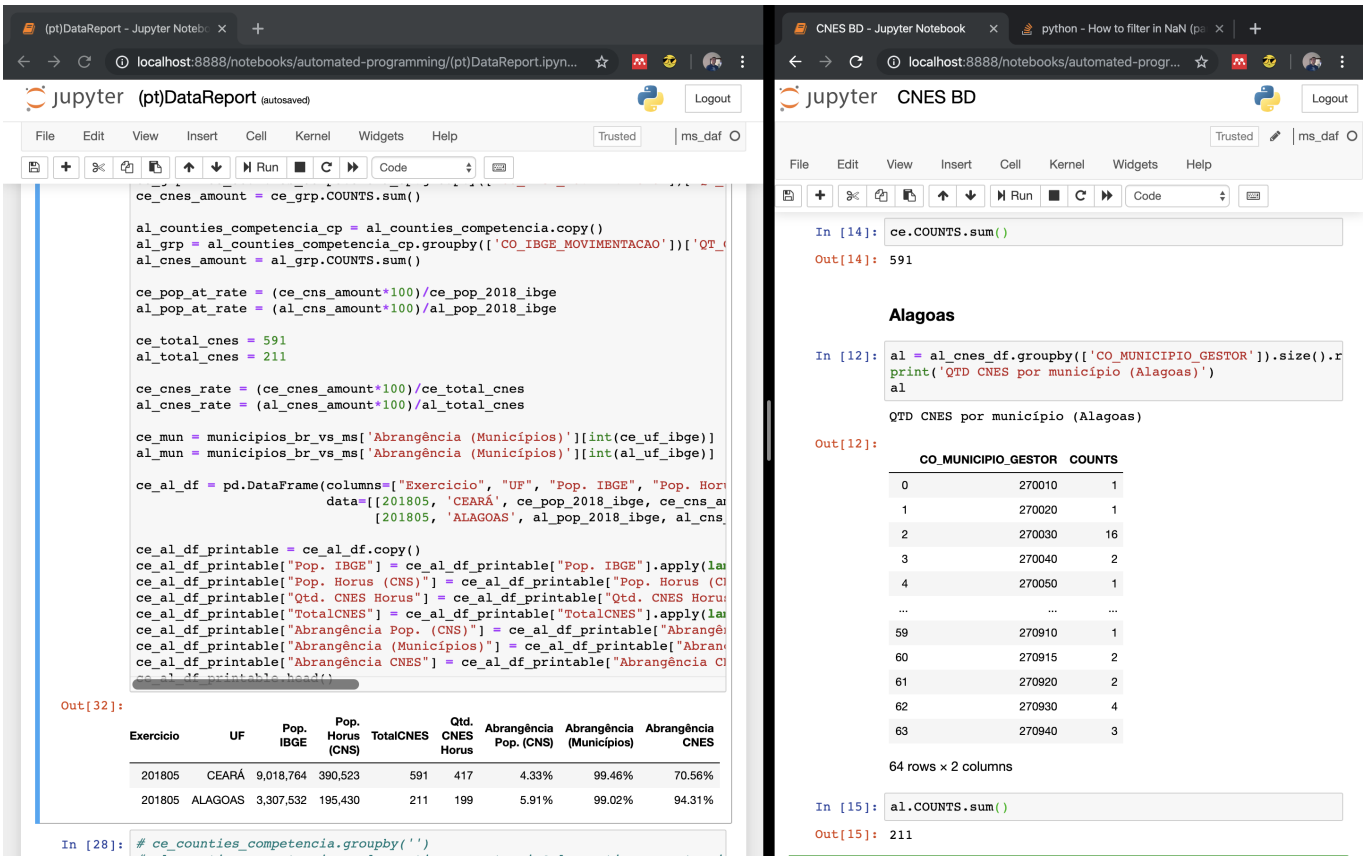
	Exercicio	UF	Pop. IBGE	Pop. Horus (CNS)	TotalCNES	Qtd. CNES Horus	Abrangência Pop. (CNS)	Abrangência (Municípios)	Abrangência (CNES)
0	201909	CEARÁ	9,018,764	390,523	591	417	4.33%	99.46%	70.56%
1	201909	ALAGOAS	3,307,532	195,430	211	199	5.91%	99.02%	94.31%

Análise Exploratória - Representatividade/Densidade CNES

Somente 3 tipos de CNES dispensam medicamento segundo as regras do DAF:

CO_ATIVIDADE - 7 - ASSISTENCIA A EMERGENCIAS Cuidados destinados a pacientes de demanda esp...
CO_ATIVIDADE - 8 - ENTREGA/DISPENSACAO DE MEDICAMENTOS Conjunto de ações relativas ao fornecimento de...
CO_ATIVIDADE - 9 - INTERNACAO Cuidados ou tratamentos prestados a um indivíd...





```
ce_cnes_amount = ce_grp.COUNTS.sum()

al_counties_competencia_cp = al_counties_competencia.copy()
al_grp = al_counties_competencia_cp.groupby(['CO_IBGE_MOVIMENTACAO'])['QT_
al_cnes_amount = al_grp.COUNTS.sum()

ce_pop_at_rate = (ce_cns_amount*100)/ce_pop_2018_ibge
al_pop_at_rate = (al_cns_amount*100)/al_pop_2018_ibge

ce_total_cnes = 591
al_total_cnes = 211

ce_cnes_rate = (ce_cnes_amount*100)/ce_total_cnes
al_cnes_rate = (al_cnes_amount*100)/al_total_cnes

ce_mun = municipios_br_vs_ms['Abrangência (Municípios)'][int(ce_uf_ibge)]
al_mun = municipios_br_vs_ms['Abrangência (Municípios)'][int(al_uf_ibge)]

ce_al_df = pd.DataFrame(columns=["Exercicio", "UF", "Pop. IBGE", "Pop. Hor
                        data=[[201805, 'CEARÁ', ce_pop_2018_ibge, ce_cns_a
                              [201805, 'ALAGOAS', al_pop_2018_ibge, al_cns_

ce_al_df_printable = ce_al_df.copy()
ce_al_df_printable["Pop. IBGE"] = ce_al_df_printable["Pop. IBGE"].apply(la
ce_al_df_printable["Pop. Horus (CNS)"] = ce_al_df_printable["Pop. Horus (C
ce_al_df_printable["Qtd. CNES Horus"] = ce_al_df_printable["Qtd. CNES Horu
ce_al_df_printable["TotalCNES"] = ce_al_df_printable["TotalCNES"].apply(la
ce_al_df_printable["Abrangência Pop. (CNS)"] = ce_al_df_printable["Abrangê
ce_al_df_printable["Abrangência (Municípios)"] = ce_al_df_printable["Abran
ce_al_df_printable["Abrangência CNES"] = ce_al_df_printable["Abrangência C
```

Out[32]:

Exercicio	UF	Pop. IBGE	Pop. Horus (CNS)	TotalCNES	Qtd. CNES Horus	Abrangência Pop. (CNS)	Abrangência (Municípios)	Abrangência CNES
201805	CEARÁ	9,018,764	390,523	591	417	4.33%	99.46%	70.56%
201805	ALAGOAS	3,307,532	195,430	211	199	5.91%	99.02%	94.31%

In [28]:
```
# ce_counties_competencia.groupby('')
```

In [14]:
```
ce.COUNTS.sum()
```

Out[14]: 591

Alagoas

In [12]:
```
al = al_cnes_df.groupby(['CO_MUNICIPIO_GESTOR']).size().r
print('QTD CNES por município (Alagoas)')
al
```

```
QTD CNES por município (Alagoas)
```

Out[12]:

	CO_MUNICIPIO_GESTOR	COUNTS
0	270010	1
1	270020	1
2	270030	16
3	270040	2
4	270050	1
...	...	...
59	270910	1
60	270915	2
61	270920	2
62	270930	4
63	270940	3

64 rows × 2 columns

In [15]:
```
al.COUNTS.sum()
```

Out[15]: 211

Macro Análise - Estudo de Caso - CEAF - Quetiapina - Alagoas

Quetiapina (BR0272832U0042) é um medicamento usado para tratar Esquizofrenia. O valor ajustado (última coluna) é considerando que não foi informado o cálculo pela unidade farmacotécnica, mas sim por caixas (no caso da quetiapina vêm sempre caixas de 30 unidades).

Fonte: OMS
Divergência Estoque vs. Dispensação

Hórus-CEAF

Estado	População	Doença	Med.	Est. (1% pop. OMS)	Est. Qtd. Medicam. (OMS)	ARIMA (05/17)	ARIMA 300MG sobra	Disp. Hórus	Horus Ajust
ACRE	769.265	Esquizofrenia	Quetiapina	7.692	230.760	151.198,90	-2.667,76	130	3.900
DF	3.013.000	Esquizofrenia	Quetiapina	30.130	903.900	3.674,8	0	860	25.800
ALAGOAS	3.322.000	Esquizofrenia	Quetiapina	33.220	996.600	252.881,26	0	6654	199.620

WS-Todos

Estado	População	Doença	Med.	Est. (1% pop.)	Total Mun.	Mun. Hórus	Mun. WS	Est. Qtd. Medicam.	ARIMA	ARIMA sobra
PARANÁ	11.350.000	Esquizofrenia	Quetiapina	113.500	399	177(44%)	357(89%)	3.405.000	0	0
SANTA CAT.	6.727.000 2.018.100	Esquizofrenia	Quetiapina	67.270	295	62(21%)	295(100%)		0	0

Resultado:

- (Hórus-CEAF) **ARIMA** média desabastecimento de **1/30 a 1/32 (70%)**. **Dispensações defasagem** de **99,90%** e com ajuste: **97,14%**
- (WS-todos) **ARIMA** não enxerga/opera/calcula. **Dispensações defasagem** de **90,19%** e com ajuste: **-194,17%**