CRCA Lab — Analytics MVP

Protótipo reprodutível para validação da Fase 0.

Este notebook demonstra uma análise mínima voltada a **processos**, e não a pessoas. Todos os registros são sintéticos, anonimizados e agregáveis. Não carregue microdados de beneficiários, contatos, documentos confidenciais, dados de saúde, renda individual, imagens identificáveis ou dados de crianças e adolescentes.

Decisão apoiada: avaliar se a Fase 0 possui condições mínimas de governança e se os processos contam com rastreabilidade suficiente para uma revisão humana da Diretoria. O notebook não aprova automaticamente a continuidade do piloto e não produz ranking individual.

Como executar no Google Colab

1. Faça upload deste arquivo em `Arquivo > Fazer upload de notebook`.
2. Selecione `Ambiente de execução > Executar tudo`.
3. Revise os quadros e o arquivo `relatorio_validacao.json` gerado na sessão.
4. Para compartilhar, use `Arquivo > Salvar uma cópia no Drive` e defina o acesso como **Restrito** ou **Leitor**, conforme a política do projeto.

Dependências: `pandas`, `numpy` e `matplotlib`, já disponíveis no Colab.

```
import json
import unicodedata
from pathlib import Path

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

pd.set_option('display.max_columns', None)
SEMENTE = 20260731
rng = np.random.default_rng(SEMENTE)
print(f'Reprodutibilidade ativada — semente: {SEMENTE}')
```

1. Regras de governança e escopo

Os campos abaixo traduzem as condições da decisão recomendada: finalidade delimitada, dados mínimos, revisão humana, proibição de ranking individual, orçamento-teto e responsabilidade da Diretoria.

```
PROIBIDOS = {
    'nome', 'cpf', 'telefone', 'email', 'saude', 'renda_individual',
    'imagem_identificavel', 'dado_de_crianca', 'dado_de_adolescente'
}
CONDICOES = {
    'finalidade_delimitada', 'dados_minimos', 'revisao_humana',
    'sem_ranking_individual', 'orcamento_teto', 'diretoria_responsavel'
}
CAMPOS_NECESSARIOS = {
    'registro_sintetico', 'projeto', 'processo', 'data_atividade',
    'data_validacao', 'campos_validos', 'campos_esperados',
    'evidencia_vinculada', 'retrabalho', 'custo_agregado',
    'reutilizacao', 'adocao', 'decisao_apoiada'
}

config = {
    'finalidade_delimitada': True,
    'dados_minimos': True,
    'revisao_humana': True,
    'sem_ranking_individual': True,
    'orcamento_teto': True,
    'diretoria_responsavel': True,
}
config
```

2. Base sintética do piloto

Cada linha representa um registro operacional sintético de processo. Não há identificação de pessoa, beneficiário, equipe ou território específico. Os valores não descrevem resultados observados do CRCA Lab.

```
processos = ['planejamento_de_pautas', 'producao_e_edicao', 'publicacao_e_ac
n_por_processo = 12
n = len(processos) * n_por_processo

df = pd.DataFrame({
    'registro_sintetico': [f'SYN-{i:03d}' for i in range(1, n + 1)],
    'projeto': 'CRCA Lab — piloto',
    'processo': np.repeat(processos, n_por_processo),
    'data_atividade': pd.Timestamp('2026-07-01') + pd.to_timedelta(rng.integ
    'campos_esperados': rng.integers(6, 10, n),
    'evidencia_vinculada': rng.choice([0, 1], size=n, p=[0.18, 0.82]),
    'retrabalho': rng.choice([0, 1], size=n, p=[0.72, 0.28]),
    'custo_agregado': rng.integers(90, 351, n),
    'reutilizacao': rng.choice([0, 1], size=n, p=[0.42, 0.58]),
    'adocao': rng.choice([0, 1], size=n, p=[0.30, 0.70]),
    'decisao_apoiada': rng.choice([0, 1], size=n, p=[0.35, 0.65]),
})
df['campos_validos'] = [rng.integers(max(1, esperado - 2), esperado + 1) for
df['data_validacao'] = df['data_atividade'] + pd.to_timedelta(rng.integers(1
```

```
df = df[[
    'registro_sintetico', 'projeto', 'processo', 'data_atividade', 'data_val
    'campos_validos', 'campos_esperados', 'evidencia_vinculada', 'retrabalho
    'custo_agregado', 'reutilizacao', 'adocao', 'decisao_apoiada'
]]
df.head()
```

3. Validação da Fase 0

A função bloqueia campos proibidos, registra lacunas de governança, verifica estrutura mínima, duplicidades, ausências, faixas válidas e coerência temporal. O resultado `fase0_apta` indica apenas que as salvaguardas declaradas estão presentes; a decisão é humana.

```
def normalizar(nome):
    texto = unicodedata.normalize('NFKD', str(nome)).encode('ASCII', 'ignore
    return texto.lower().strip().replace(' ', '_')

def validar(config, df):
    faltantes_governanca = sorted(CONDICOES - {k for k, v in config.items()
    colunas_normalizadas = {normalizar(c) for c in df.columns}
    bloqueados = sorted(PROIBIDOS & colunas_normalizadas)
    if bloqueados:
        raise ValueError(f'Campos pessoais ou sensíveis proibidos: {bloquead

    campos_ausentes = sorted(CAMPOS_NECESSARIOS - set(df.columns))
    if campos_ausentes:
        raise ValueError(f'Campos mínimos ausentes: {campos_ausentes}')

    data_atividade = pd.to_datetime(df['data_atividade'], errors='coerce')
    data_validacao = pd.to_datetime(df['data_validacao'], errors='coerce')
    prazo_invalido = int((data_validacao < data_atividade).sum())
    campos_invalidos = int(((df['campos_validos'] < 0) | (df['campos_validos
    binarios = ['evidencia_vinculada', 'retrabalho', 'reutilizacao', 'adocao
    valores_binarios_invalidos = {c: int((~df[c].isin([0, 1, True, False])).

    return {
        'fase0_apta': not faltantes_governanca and not bloqueados and not ca
        'faltantes_governanca': faltantes_governanca,
        'linhas': int(len(df)),
        'duplicadas': int(df.duplicated().sum()),
        'ausencias_por_campo': {k: int(v) for k, v in df.isna().sum().items(
        'datas_invalidas': prazo_invalido,
        'campos_invalidos': campos_invalidos,
        'valores_binarios_invalidos': valores_binarios_invalidos,
    }

validacao = validar(config, df)
pd.DataFrame([validacao]).T.rename(columns={0: 'resultado'})
```

4. Indicadores agregados por processo

Alcance não é tratado como impacto. Os indicadores apoiam a leitura de qualidade operacional: completude, prazo, rastreabilidade, retrabalho, custo agregado, reutilização, adoção e decisões apoiadas.

```
def indicadores(df):
    x = df.copy()
    x['completude'] = x['campos_validos'] / x['campos_esperados']
    x['prazo_dias'] = (pd.to_datetime(x['data_validacao']) - pd.to_datetime(
    return x.groupby(['projeto', 'processo'], as_index=False).agg(
        registros=('registro_sintetico', 'size'),
        completude_mediana=('completude', 'median'),
        prazo_mediano_dias=('prazo_dias', 'median'),
        rastreabilidade=('evidencia_vinculada', 'mean'),
        retrabalho=('retrabalho', 'mean'),
        custo_agregado=('custo_agregado', 'sum'),
        reutilizacao=('reutilizacao', 'mean'),
        adocao=('adocao', 'mean'),
        decisoes_apoiadas=('decisao_apoiada', 'sum'),
    )

indicadores_agregados = indicadores(df)
indicadores_agregados.round(4)
```

```
grafico = indicadores_agregados.set_index('processo')[['rastreabilidade', 'a
ax = grafico.plot(kind='bar', figsize=(10, 5), color=['#0F3B24', '#8AB833',
ax.set_title('Indicadores agregados por processo — base sintética')
ax.set_xlabel('Processo')
ax.set_ylabel('Proporção')
ax.set_ylim(0, 1)
ax.legend(['Rastreabilidade', 'Adoção', 'Reutilização'], frameon=False)
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

5. Relatório para revisão humana

O arquivo JSON organiza o resultado em formato reutilizável. A regra de interpretação permanece: **comparar processos; nunca pessoas**.

```
def exportar(config, df, arquivo='relatorio_validacao.json'):
    saida = {
        'metadados': {
            'base': 'dados sintéticos',
            'semente': SEMENTE,
            'finalidade': 'Avaliar condições mínimas da Fase 0 e qualidade d
        },
```

```
            'validacao': validar(config, df),
            'indicadores_por_processo': indicadores(df).round(4).to_dict('record
            'regra': 'Comparar processos; nunca pessoas.',
            'limite': 'Resultados sintéticos não são evidência de impacto observ
            'decisao': 'Submeter a leitura e a continuidade do piloto à revisão
        }
        Path(arquivo).write_text(json.dumps(saida, ensure_ascii=False, indent=2,
        return saida

    relatorio = exportar(config, df)
    print('Arquivo gerado:', Path('relatorio_validacao.json').resolve())
    print(json.dumps(relatorio['validacao'], ensure_ascii=False, indent=2))
```

6. Teste de salvaguarda

A célula abaixo demonstra que a presença de um campo proibido interrompe a análise. Execute-a apenas para testar o mecanismo; ela não altera a base original.

```
    teste_bloqueado = df.assign(email='nao_utilizar@exemplo.invalid')
    try:
        validar(config, teste_bloqueado)
    except ValueError as erro:
        print('Salvaguarda funcionando:', erro)
```

Declaração de uso responsável de IA generativa

Este código foi elaborado com apoio de IA generativa, sob revisão humana. A ferramenta não é autora, não substitui análise, decisão metodológica, verificação de fontes ou responsabilidade pela redação final. Foram usados apenas dados sintéticos neste notebook. Resultados simulados não devem ser apresentados como observados.

Prompts de referência: usar somente conteúdo fornecido; separar fatos, metas, hipóteses e lacunas; não inferir atributos pessoais; interromper diante de dados pessoais ou sensíveis; listar evidência, limitação e responsável humano.