ETAPA 3a — Preparação do Painel CAGED × Município × Conectividade (Anatel)

Dissertação: Inteligência Artificial Generativa e o Mercado de Trabalho Brasileiro.

Aluno: Manoel Brasil Orlandi

0.1 Contextualização

A Etapa 2 estimou o efeito médio nacional da IA generativa sobre o emprego formal. O efeito médio mascara **heterogeneidade espacial**: a penetração de banda larga varia fortemente entre municípios. A Etapa 3 explora isso via **Triple-DiD**, adicionando a dimensão municipal de conectividade (Anatel). Hipótese: o efeito da IA é **amplificado** em municípios com alta conectividade.

0.2 Objetivo

Construir painel **ocupação (CBO 4d) × município × mês** com CAGED, exposição à IA (Etapa 2a) e índice de conectividade municipal. **Saída:** `data/output/painel_caged_municipio_anatel.parquet`.

0.3 Ficha Técnica dos Dados

Campo	CAGED	Anatel BLF	IBGE
Fonte	MTE / CAGED	Anatel / SCM	IBGE / Censo
Período	Jan/2021 – Jun/2025	2021–Out/2022 (pré-trat.)	2022
Unidade	Movimentação	Acesso banda larga	Município
Granularidade	Ocupação × Município × Mês	Município × mês	Município

0.4 Referências

- Autor & Dorn (2013); Hjort & Poulsen (2019); Goldfarb & Tucker (2019); Webb (2020); Felten et al. (2021).

0.5 1. Configuração do ambiente

Paths, parâmetros e dependências. Conectividade medida em período **pré-tratamento** (Jan–Out/2022) para evitar endogeneidade.

```
# Etapa 3a.1 — Configuração
import warnings
import pandas as pd
import numpy as np
from pathlib import Path
```

```
warnings.filterwarnings("ignore", category=FutureWarning)

# Sempre usar a pasta notebook/data (raiz = pasta que contém 'notebook')
def _find_project_root():
    p = Path.cwd().resolve()
    for _ in range(5):
        if (p / "notebook").is_dir():
            return p
        p = p.parent if p.parent != p else p
    return Path.cwd().resolve()
PROJECT_ROOT = _find_project_root()
DATA_ROOT = PROJECT_ROOT / "notebook" / "data"

DATA_INPUT     = DATA_ROOT / "input"
DATA_RAW       = DATA_ROOT / "raw"
DATA_PROCESSED = DATA_ROOT / "processed"
DATA_OUTPUT    = DATA_ROOT / "output"
OUTPUTS_TABLES = PROJECT_ROOT / "notebook" / "outputs" / "tables"
for d in [DATA_INPUT, DATA_RAW, DATA_PROCESSED, DATA_OUTPUT, OUTPUTS_TABLES]:
    d.mkdir(parents=True, exist_ok=True)

# Cache CAGED municipal: para forçar nova agregação (ex.: faixa etária), use USE_CAGED
USE_CAGED_CACHE = False
CAGED_CACHE_FILE = DATA_PROCESSED / "painel_caged_municipio.parquet"
PAINEL_CAGED_MUN = CAGED_CACHE_FILE

GCP_PROJECT_ID = "mestrado-pnad-2026"
ANO_INICIO, ANO_FIM = 2021, 2025
ANO_TRATAMENTO, MES_TRATAMENTO = 2022, 12
ANO_PRE_CONECT, MES_FIM_PRE_CONECT = 2022, 10
MIN_POPULACAO = 50_000
MIN_MOVIMENTACOES_PRE = 5

PAINEL_ETAPA2 = DATA_OUTPUT / "painel_caged_did_ready.parquet"
IPCA_FILE = DATA_PROCESSED / "ipca_mensal.parquet"
ANATEL_PRE_FILE = DATA_PROCESSED / "anatel_pre_tratamento.parquet"
IBGE_FILE = DATA_PROCESSED / "ibge_municipios.parquet"
CONECTIVIDADE_FILE = DATA_PROCESSED / "conectividade_municipal.parquet"
PAINEL_FINAL = DATA_OUTPUT / "painel_caged_municipio_anatel.parquet"
PAINEL_FINAL_V2 = DATA_OUTPUT / "painel_caged_municipio_anatel_v2.parquet"

print("Data root:", DATA_ROOT.resolve())
print("Cache CAGED:", CAGED_CACHE_FILE.resolve(), "(existe:", CAGED_CACHE_FILE.exists(
print("Configuração carregada. Período pré conectividade: até", f"{MES_FIM_PRE_CONECT}
```

```
Data root: /Users/manebrasil/Documents/Projects/Dissetação Mestrado/notebook/data
Cache CAGED: /Users/manebrasil/Documents/Projects/Dissetação
Mestrado/notebook/data/processed/painel_caged_municipio.parquet (existe: True ,
use_cache: False )
Configuração carregada. Período pré conectividade: até 10/2022
```

0.6 2. Anatel — Banda Larga Fixa

Download via BigQuery (Base dos Dados). Período pré-tratamento: 2021 e Jan–Out/2022. Agregado por município: média de acessos e % fibra.

```
# Etapa 3a.2 — Anatel (ou carregar do cache)
if ANATEL_PRE_FILE.exists():
    df_anatel = pd.read_parquet(ANATEL_PRE_FILE)
    print(f"Anatel carregado do cache: {len(df_anatel):,} municípios")
else:
    query_anatel = f'''
    SELECT ano, mes, id_municipio,
           SUM(acessos) AS total_acessos,
           SUM(CASE WHEN LOWER(SAFE_CAST(tecnologia AS STRING)) LIKE '%fibra%' THEN ac
    FROM `basedosdados.br_anatel_banda_larga_fixa.microdados`
    WHERE ano IN (2021, {ANO_PRE_CONECT}) AND (ano < {ANO_PRE_CONECT} OR mes <= {MES_F
    GROUP BY ano, mes, id_municipio
    '''
    try:
        from google.cloud import bigquery
        client = bigquery.Client(project=GCP_PROJECT_ID)
        df = client.query(query_anatel).to_dataframe(create_bqstorage_client=True)
    except Exception as e:
        import basedosdados as bd
        df = bd.read_sql(query_anatel, billing_project_id=GCP_PROJECT_ID)
    out = df.groupby("id_municipio").agg(
        media_acessos_pre=("total_acessos", "mean"),
        soma_acessos=("total_acessos", "sum"),
        soma_fibra=("acessos_fibra", "sum"),
    ).reset_index()
    out["pct_fibra_pre"] = out["soma_fibra"] / out["soma_acessos"].clip(lower=1)
    df_anatel = out[["id_municipio", "media_acessos_pre", "pct_fibra_pre"]]
    df_anatel.to_parquet(ANATEL_PRE_FILE, index=False)
    print(f"Anatel salvo: {len(df_anatel):,} municípios")
df_anatel.head()
```

```
Anatel carregado do cache: 5,570 municípios
```

	id_municipio	media_acessos_pre	pct_fibra_pre
0	1100015	1771.0	0.0
1	1100023	19923.5	0.0
2	1100031	169.636364	0.0
3	1100049	14243.272727	0.0
4	1100056	1107.636364	0.0

0.7 3. IBGE — Domicílios, PIB, População

População (2022), PIB municipal (2021), domicílios (Censo 2022 ou proxy) para denominador da penetração e filtro de porte.

```
# Etapa 3a.3 — IBGE
# Nota: tabela br_ibge_pib.municipio tem apenas id_municipio e pib; população vem de b
if IBGE_FILE.exists():
    df_ibge = pd.read_parquet(IBGE_FILE)
    print(f"IBGE carregado do cache: {len(df_ibge):,} municípios")
else:
    q_pop = "SELECT id_municipio, populacao FROM `basedosdados.br_ibge_populacao.munic
    q_pib = "SELECT id_municipio, pib FROM `basedosdados.br_ibge_pib.municipio` WHERE
    use_bq = False
    try:
        from google.cloud import bigquery
        client = bigquery.Client(project=GCP_PROJECT_ID)
        df_pop = client.query(q_pop).to_dataframe(create_bqstorage_client=True)
        df_pib = client.query(q_pib).to_dataframe(create_bqstorage_client=True)
        use_bq = True
    except Exception:
        import basedosdados as bd
        df_pop = bd.read_sql(q_pop, billing_project_id=GCP_PROJECT_ID)
        df_pib = bd.read_sql(q_pib, billing_project_id=GCP_PROJECT_ID)
    df_ibge = df_pop.merge(df_pib[["id_municipio", "pib"]], on="id_municipio", how="ou
    df_ibge["pib_per_capita"] = df_ibge["pib"] / df_ibge["populacao"].clip(lower=1)
    try:
        q_dom = "SELECT id_municipio, SUM(domicilios_particulares_ocupados) AS domicil
        if use_bq:
            df_dom = client.query(q_dom).to_dataframe(create_bqstorage_client=True)
        else:
            import basedosdados as bd
            df_dom = bd.read_sql(q_dom, billing_project_id=GCP_PROJECT_ID)
        df_ibge = df_ibge.merge(df_dom, on="id_municipio", how="left")
    except Exception:
        df_ibge["domicilios"] = (df_ibge["populacao"] / 3).round().clip(lower=1)
    if "domicilios" not in df_ibge.columns:
        df_ibge["domicilios"] = (df_ibge["populacao"] / 3).round().clip(lower=1)
    df_ibge = df_ibge[["id_municipio", "populacao", "domicilios", "pib", "pib_per_capi
    df_ibge.to_parquet(IBGE_FILE, index=False)
    print(f"IBGE salvo: {len(df_ibge):,} municípios")
df_ibge.head()
```

```
IBGE carregado do cache: 5,570 municípios
```

	id_municipio	populacao	domicilios	pib	pib_per_capita
0	1100015	21494	7699	734469000	34170.884898
1	1100023	96833	34784	3209761000	33147.387771
2	1100031	5351	1967	238412000	44554.66268
3	1100049	86887	31931	2792383000	32138.09891
4	1100056	15890	5876	743037000	46761.296413

0.8 4. Índice de Conectividade Municipal

Merge Anatel + IBGE. **penetracao_bl** = media_acessos_pre / domicilios. **alta_conectividade** = 1 se penetração > mediana. Filtrar municípios com população ≥ MIN_POPULACAO.

```
# Etapa 3a.4 — Conectividade
df_anatel = pd.read_parquet(ANATEL_PRE_FILE)
df_ibge = pd.read_parquet(IBGE_FILE)
for d in [df_anatel, df_ibge]:
    d["id_municipio"] = d["id_municipio"].astype(str).str.zfill(7)
df_conect = df_anatel.merge(df_ibge, on="id_municipio", how="inner")
df_conect["penetracao_bl"] = df_conect["media_acessos_pre"] / df_conect["domicilios"].
med = df_conect["penetracao_bl"].median()
df_conect["alta_conectividade"] = (df_conect["penetracao_bl"] > med).astype(int)
df_conect["conectividade_q75"] = (df_conect["penetracao_bl"] > df_conect["penetracao_b
df_conect["conectividade_q25"] = (df_conect["penetracao_bl"] > df_conect["penetracao_b
# % fibra óptica como proxy alternativo de qualidade (cutoff por mediana)
mediana_fibra = df_conect["pct_fibra_pre"].median()
df_conect["alta_fibra"] = (df_conect["pct_fibra_pre"] > mediana_fibra).astype(int)
df_conect = df_conect[df_conect["populacao"] >= MIN_POPULACAO].copy()
df_conect.to_parquet(CONECTIVIDADE_FILE, index=False)
print(f"Mediana penetração: {med:.4f}. Alta conect.: {df_conect['alta_conectividade'].
```

```
Mediana penetração: 0.2681. Alta conect.: 529 municípios. Total: 657
```

0.9 5. CAGED — Agregação por Ocupação × Município × Período

Reagregar microdados CAGED (data/raw/caged_{ano}.parquet) por **(cbo_4d, id_municipio, ano, mes)** com as mesmas métricas da Etapa 2a. Inclui sigla_uf e **decomposição por faixa etária** (jovem <30, intermediário 30–49, senior 50+): adm_jovem, adm_intermediario, adm_senior e salário médio por faixa. Processamento ano a ano para evitar OOM.

Nota: Se o cache foi gerado antes da inclusão da faixa etária, use na seção 1 `USE_CAGED_CACHE = False` e execute de novo, ou apague o arquivo em `notebook/data/processed/painel_caged_municipio.parquet`.

```
# Etapa 3a.5 — CAGED municipal (ou carregar do cache)
# Agregação por (cbo_4d, id_municipio, ano, mes) — mesmas métricas da Etapa 2a. Sexo:
import time
CODIGO_SEXO_MULHER = 3
CODIGOS_RACA_BRANCA, CODIGOS_RACA_NEGRA = [1], [2, 4]
IDADE_CORTE_JOVEM = 29
CODIGOS_ESCOLARIDADE_SUPERIOR = ["9", "10", "11", "12", "13"]
CNAE_SECOES_TECNOLOGICO = ["J"]
SETOR_TECNOLOGICO_LIMIAR = 0.5

def processar_ano_caged_mun(ano):
    """Agrega um ano de CAGED por (cbo_4d, id_municipio, ano, mes)."""
    t0 = time.time()
    df = pd.read_parquet(DATA_RAW / f"caged_{ano}.parquet")
    print(f"  [{ano}] Carregado: {len(df):,} registros ({time.time()-t0:.0f}s)")
    df["cbo_2002"] = df["cbo_2002"].astype(str).str.strip()
    df["cbo_4d"] = df["cbo_2002"].str[:4]
    df = df[df["cbo_4d"].str.len() == 4]
```

```
    df = df[df["cbo_4d"].str.isdigit()]
    df = df[~df["cbo_4d"].isin(["0000", "nan", ""])]
    df["is_mulher"] = (df["sexo"].astype(str) == str(CODIGO_SEXO_MULHER)).astype(float
    raca_str = df["raca_cor"].astype(str)
    df["is_branco"] = raca_str.isin([str(c) for c in CODIGOS_RACA_BRANCA]).astype(floa
    df["is_negro"] = raca_str.isin([str(c) for c in CODIGOS_RACA_NEGRA]).astype(float)
    df["is_jovem"] = (df["idade"] <= IDADE_CORTE_JOVEM).astype(float)
    df["is_intermediario"] = ((df["idade"] >= 30) & (df["idade"] < 50)).astype(float)
    df["is_senior"] = (df["idade"] >= 50).astype(float)
    df["is_superior"] = df["grau_instrucao"].astype(str).isin(CODIGOS_ESCOLARIDADE_SUP
    df["is_setor_tech"] = df["cnae_2_secao"].astype(str).str.strip().isin(CNAE_SECOES_
    df_adm = df[df["saldo_movimentacao"] == 1].copy()
    df_desl = df[df["saldo_movimentacao"] == -1]
    print(f"  [{ano}] Admissões: {len(df_adm):,} | Desligamentos: {len(df_desl):,}")
    df_adm["sal_mulher"] = np.where(df_adm["is_mulher"] == 1, df_adm["salario_mensal"]
    df_adm["sal_homem"] = np.where(df_adm["is_mulher"] == 0, df_adm["salario_mensal"],
    df_adm["sal_branco"] = np.where(df_adm["is_branco"] == 1, df_adm["salario_mensal"]
    df_adm["sal_negro"] = np.where(df_adm["is_negro"] == 1, df_adm["salario_mensal"],
    df_adm["sal_jovem"] = np.where(df_adm["is_jovem"] == 1, df_adm["salario_mensal"],
    df_adm["sal_naojovem"] = np.where(df_adm["is_jovem"] == 0, df_adm["salario_mensal"
    df_adm["sal_intermediario"] = np.where(df_adm["is_intermediario"] == 1, df_adm["sa
    df_adm["sal_senior"] = np.where(df_adm["is_senior"] == 1, df_adm["salario_mensal"]
    df_adm["sal_sup"] = np.where(df_adm["is_superior"] == 1, df_adm["salario_mensal"],
    df_adm["sal_med"] = np.where(df_adm["is_superior"] == 0, df_adm["salario_mensal"],
    df_adm["is_homem"] = 1 - df_adm["is_mulher"]
    grp = ["cbo_4d", "id_municipio", "ano", "mes"]
    painel_adm = df_adm.groupby(grp).agg(
        sigla_uf=("sigla_uf", "first"),
        admissoes=("saldo_movimentacao", "count"),
        salario_medio_adm=("salario_mensal", "mean"),
        idade_media_adm=("idade", "mean"),
        pct_mulher_adm=("is_mulher", "mean"),
        pct_superior_adm=("is_superior", "mean"),
        pct_branco_adm=("is_branco", "mean"),
        pct_negro_adm=("is_negro", "mean"),
        pct_jovem_adm=("is_jovem", "mean"),
        pct_tecnologico_adm=("is_setor_tech", "mean"),
        salario_medio_mulher=("sal_mulher", "mean"),
        salario_medio_homem=("sal_homem", "mean"),
        salario_medio_branco=("sal_branco", "mean"),
        salario_medio_negro=("sal_negro", "mean"),
        salario_medio_jovem=("sal_jovem", "mean"),
        salario_medio_naojovem=("sal_naojovem", "mean"),
        salario_medio_intermediario=("sal_intermediario", "mean"),
        salario_medio_senior=("sal_senior", "mean"),
        salario_medio_superior=("sal_sup", "mean"),
        salario_medio_medio=("sal_med", "mean"),
        admissoes_mulher=("is_mulher", "sum"),
        admissoes_homem=("is_homem", "sum"),
        admissoes_jovem=("is_jovem", "sum"),
        adm_jovem=("is_jovem", "sum"),
        adm_intermediario=("is_intermediario", "sum"),
        adm_senior=("is_senior", "sum"),
        admissoes_negro=("is_negro", "sum"),
```

```
    ).reset_index()
    mediana = df_adm.groupby(grp)["salario_mensal"].median().reset_index().rename(colu
    painel_adm = painel_adm.merge(mediana, on=grp, how="left")
    painel_desl = df_desl.groupby(grp).agg(
        desligamentos=("saldo_movimentacao", "count"),
        salario_medio_desl=("salario_mensal", "mean"),
    ).reset_index()
    p = painel_adm.merge(painel_desl, on=grp, how="outer").fillna(0)
    p["saldo"] = p["admissoes"] - p["desligamentos"]
    p["n_movimentacoes"] = p["admissoes"] + p["desligamentos"]
    p["setor_tecnologico"] = (p["pct_tecnologico_adm"] >= SETOR_TECNOLOGICO_LIMIAR).as
    p["periodo"] = p["ano"].astype(int).astype(str) + "-" + p["mes"].astype(int).astyp
    p["periodo_num"] = p["ano"].astype(int) * 100 + p["mes"].astype(int)
    p["post"] = (p["periodo_num"] >= ANO_TRATAMENTO * 100 + MES_TRATAMENTO).astype(int
    p["ln_admissoes"] = np.log(p["admissoes"] + 1)
    p["ln_desligamentos"] = np.log(p["desligamentos"] + 1)
    p["ln_salario_adm"] = np.log(p["salario_medio_adm"].clip(lower=1))
    p["cbo_2d"] = p["cbo_4d"].str[:2]
    for grp_name in ["mulher", "homem", "branco", "negro", "jovem", "naojovem", "super
        col = f"salario_medio_{grp_name}"
        if col in p.columns:
            p[f"ln_salario_{grp_name}"] = np.log(p[col].clip(lower=1))
    for grp_name in ["mulher", "homem", "jovem", "negro"]:
        col = f"admissoes_{grp_name}"
        if col in p.columns:
            p[f"ln_admissoes_{grp_name}"] = np.log(p[col].astype(float) + 1)
    print(f"  [{ano}] Painel: {len(p):,} linhas ({time.time()-t0:.0f}s)")
    del df, df_adm, df_desl
    return p

if USE_CAGED_CACHE and CAGED_CACHE_FILE.exists():
    df_caged = pd.read_parquet(CAGED_CACHE_FILE)
    print(f"CAGED municipal carregado do cache: {CAGED_CACHE_FILE.resolve()}")
    print(f"  {len(df_caged):,} linhas, {df_caged['id_municipio'].nunique():,} municíp
else:
    print("ETAPA 3a — Agregação CAGED por ocupação × município × período")
    t0 = time.time()
    paineis = [processar_ano_caged_mun(ano) for ano in range(ANO_INICIO, ANO_FIM + 1)]
    painel = pd.concat(paineis, ignore_index=True)
    painel["id_municipio"] = painel["id_municipio"].astype(str).str.zfill(7)
    # Garantir colunas texto como string para o PyArrow (evitar ArrowTypeError por tip
    for col in ["sigla_uf", "cbo_4d", "cbo_2d", "periodo"]:
        if col in painel.columns:
            painel[col] = painel[col].astype(str)
    painel.to_parquet(CAGED_CACHE_FILE, index=False)
    print(f"Total: {len(painel):,} linhas | {painel['cbo_4d'].nunique()} ocupações | {
    df_caged = painel
df_caged["id_municipio"] = df_caged["id_municipio"].astype(str).str.zfill(7)
df_caged.head()
```

```
ETAPA 3a — Agregação CAGED por ocupação × município × período
  [2021] Carregado: 36,554,795 registros (7s)
  [2021] Admissões: 19,703,604 | Desligamentos: 16,851,191
  [2021] Painel: 2,627,059 linhas (115s)
```

```
  [2022] Carregado: 42,475,516 registros (7s)
  [2022] Admissões: 22,243,441 | Desligamentos: 20,232,075
  [2022] Painel: 2,868,327 linhas (137s)
  [2023] Carregado: 44,485,982 registros (9s)
  [2023] Admissões: 22,982,161 | Desligamentos: 21,503,821
  [2023] Painel: 2,945,596 linhas (145s)
  [2024] Carregado: 48,996,040 registros (10s)
  [2024] Admissões: 25,336,277 | Desligamentos: 23,659,763
  [2024] Painel: 3,056,122 linhas (156s)
  [2025] Carregado: 26,312,103 registros (4s)
  [2025] Admissões: 13,763,059 | Desligamentos: 12,549,044
  [2025] Painel: 1,597,780 linhas (76s)
Total: 13,094,884 linhas | 629 ocupações | 5570 municípios | 54 períodos | salvo em
689s
```

	cbo_4d	id_municipio	ano	mes	sigla_uf	admissoes	salario_medio_adm	idade_media_adm	pct_mu
0	1010	1100015	2021	2	RO	1	1178.0	21.0	0.0
1	1010	1100023	2021	5	RO	1	1243.0	40.0	1.0
2	1010	1100056	2021	10	0	0	0.0	0.0	0.0
3	1010	1100205	2021	1	0	0	0.0	0.0	0.0
4	1010	1100205	2021	5	0	0	0.0	0.0	0.0

5 rows × 56 columns

0.10 6. Merge — Painel Tridimensional

Merge CAGED municipal + exposição (Etapa 2, uma linha por cbo_4d) + conectividade. Variáveis temporais (post, tempo_relativo_meses) e interações DDD (triple_did, post_alta_exp, post_alta_conect). Salário real com IPCA.

```
# Etapa 3a.6 — Merge
painel2 = pd.read_parquet(PAINEL_ETAPA2)
cols_exp = ["cbo_4d", "exposure_score_2d", "exposure_score_4d", "alta_exp", "alta_exp_
            "alta_exp_10", "alta_exp_25", "alta_exp_mediana", "quintil_exp",
            "grande_grupo_cbo", "grande_grupo_nome", "anthropic_automation_index", "is
cols_exp = [c for c in cols_exp if c in painel2.columns]
df_exposure = painel2[cols_exp].drop_duplicates("cbo_4d")

df = df_caged.merge(df_exposure, on="cbo_4d", how="inner")
df = df.merge(
    df_conect[["id_municipio", "penetracao_bl", "pct_fibra_pre", "alta_conectividade",
    on="id_municipio", how="inner"
)
df["periodo_dt"] = pd.to_datetime(df["periodo"] + "-01", errors="coerce")
df["post"] = (df["periodo_dt"] >= f"{ANO_TRATAMENTO}-{MES_TRATAMENTO:02d}-01").astype(
df["tempo_relativo_meses"] = (df["periodo_dt"].dt.year - ANO_TRATAMENTO) * 12 + (df["p
df["uf_periodo"] = df["sigla_uf"].astype(str) + "_" + df["periodo"].astype(str)
df["post_alta_exp"] = df["post"] * df["alta_exp"]
df["post_alta_conect"] = df["post"] * df["alta_conectividade"]
df["alta_exp_alta_conect"] = df["alta_exp"] * df["alta_conectividade"]
```

```
df["triple_did"] = df["post"] * df["alta_exp"] * df["alta_conectividade"]
# Interações para Triple-DiD com proxy fibra
df["post_alta_fibra"] = df["post"] * df["alta_fibra"]
df["alta_exp_alta_fibra"] = df["alta_exp"] * df["alta_fibra"]
df["triple_did_fibra"] = df["post"] * df["alta_exp"] * df["alta_fibra"]
# Dummy capital estadual (proxy extrema de conectividade para robustez no 3b)
capitais_ibge = ["1100205", "1200401", "1302603", "1400100", "1501402", "1600303", "17
df["capital"] = df["id_municipio"].astype(str).str.zfill(7).isin(capitais_ibge).astype

if IPCA_FILE.exists():
    df_ipca = pd.read_parquet(IPCA_FILE)
    df = df.merge(df_ipca[["ano", "mes", "indice"]], on=["ano", "mes"], how="left")
    df["salario_real_adm"] = df["salario_medio_adm"] * (100.0 / df["indice"].clip(lowe
else:
    df["salario_real_adm"] = df["salario_medio_adm"]
df["ln_salario_real_adm"] = np.log(df["salario_real_adm"].clip(lower=1))
df["ln_pib_pc"] = np.log(df["pib_per_capita"].clip(lower=1))

if MIN_MOVIMENTACOES_PRE > 0:
    pre = df["post"] == 0
    mov = df.loc[pre].groupby(["cbo_4d", "id_municipio"])["n_movimentacoes"].sum().res
    mov.columns = ["cbo_4d", "id_municipio", "mov_pre"]
    df = df.merge(mov, on=["cbo_4d", "id_municipio"], how="left")
    df = df[df["mov_pre"] >= MIN_MOVIMENTACOES_PRE].drop(columns=["mov_pre"])
print(f"Painel final: {len(df):,} linhas | {df['cbo_4d'].nunique()} ocupações | {df['i
```

```
Painel final: 5,534,808 linhas | 614 ocupações | 657 municípios | 54 períodos
```

0.11 7. Validação e Estatísticas Descritivas

Balanço pré por grupo de conectividade; distribuição de penetração; correlação penetração × PIB per capita.

```
# Etapa 3a.7 — Validação
pre = df[df["post"] == 0]
print("Balanço pré-tratamento por conectividade:")
print(pre.groupby("alta_conectividade").agg(
    n_obs=("cbo_4d", "count"),
    admissoes_media=("admissoes", "mean"),
    salario_medio=("salario_medio_adm", "mean"),
    penetracao_media=("penetracao_bl", "mean"),
).round(2))
print("\nCorrelação penetração × PIB per capita:", df_conect["penetracao_bl"].corr(df_
```

```
Balanço pré-tratamento por conectividade:
```

alta_conectividade				
0	205077	5.02	8230.56	0.18
1	2126257	15.02	3991.63	0.69

```
Correlação penetração × PIB per capita: 0.304
```

0.12 8. Exportação

Salvar painel final em parquet e tabela de conectividade em CSV.

```
# Etapa 3a.8 — Exportação
df.to_parquet(PAINEL_FINAL, index=False)
df_conect.to_csv(OUTPUTS_TABLES / "conectividade_municipal.csv", index=False)
print(f"Painel salvo: {PAINEL_FINAL} ({PAINEL_FINAL.stat().st_size/1e6:.1f} MB)")
print(f"Conectividade: {OUTPUTS_TABLES / 'conectividade_municipal.csv'}")
```

```
Painel salvo: /Users/manebrasil/Documents/Projects/Dissetação
Mestrado/notebook/data/output/painel_caged_municipio_anatel.parquet (635.3 MB)
Conectividade: /Users/manebrasil/Documents/Projects/Dissetação
Mestrado/notebook/outputs/tables/conectividade_municipal.csv
```

0.13 9. Decomposição por Faixa Etária e Exportação v2

Construir outcomes desagregados por faixa etária (jovem <30, intermediário 30–49, senior 50+) para o Triple-DiD etário no Notebook 3b: share de jovens/seniores nas admissões, log-admissões e log-salário real por faixa, razão salarial jovem/senior.

Nota: Células ocupação×município×mês sem admissões em alguma faixa terão NaN; regressões com outcomes por faixa no 3b terão menos observações (perda amostral esperada).

```
# Etapa 3a.9 — Decomposição por Faixa Etária e Re-exportação v2
colunas_faixa = ["adm_jovem", "adm_intermediario", "adm_senior", "salario_medio_jovem"
if not all(c in df.columns for c in colunas_faixa):
    print("AVISO: Painel sem colunas de faixa etária (adm_jovem, adm_intermediario, ad
    print("Rode a seção 5 com USE_CAGED_CACHE = False na seção 1, ou apague o cache:",
else:
    # adm_total = soma das três faixas (pode diferir de admissoes se houver missings d
    df["adm_total"] = df["adm_jovem"].fillna(0) + df["adm_intermediario"].fillna(0) +
    df["adm_total"] = df["adm_total"].replace(0, np.nan)
    df["share_jovem"] = df["adm_jovem"] / df["adm_total"]
    df["share_senior"] = df["adm_senior"] / df["adm_total"]
    df["ln_adm_jovem"] = np.log1p(df["adm_jovem"].fillna(0))
    df["ln_adm_intermediario"] = np.log1p(df["adm_intermediario"].fillna(0))
    df["ln_adm_senior"] = np.log1p(df["adm_senior"].fillna(0))
    # Salário real por faixa (deflator = indice IPCA, mesmo que ln_salario_real_adm)
    if "indice" in df.columns:
        deflator = df["indice"] / 100.0
    else:
        deflator = 1.0
    for faixa in ["jovem", "intermediario", "senior"]:
        col_sal = f"salario_medio_{faixa}"
        df[f"sal_real_{faixa}"] = df[col_sal] / deflator
        df[f"ln_sal_real_{faixa}"] = np.log(df[f"sal_real_{faixa}"].clip(lower=1))
    df["razao_sal_jovem_senior"] = df["sal_real_jovem"] / df["sal_real_senior"].replac
    # Re-exportar painel v2 com todas as variáveis (faixa etária, fibra, capital)
    df.to_parquet(PAINEL_FINAL_V2, index=False)
    print(f"Painel v2 salvo: {PAINEL_FINAL_V2} ({PAINEL_FINAL_V2.stat().st_size/1e6:.1
    print(f"Variáveis: share_jovem, share_senior, ln_adm_* por faixa, ln_sal_real_* po
```

```
Painel v2 salvo: /Users/manebrasil/Documents/Projects/Dissetação
Mestrado/notebook/data/output/painel_caged_municipio_anatel_v2.parquet (803.2 MB)
Variáveis: share_jovem, share_senior, ln_adm_* por faixa, ln_sal_real_* por faixa,
razao_sal_jovem_senior, alta_fibra, triple_did_fibra, capital
```