ETAPA 3b — Análise Triple-DiD: IA Generativa, Conectividade e Emprego Formal

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

Objetivo: Estimar o efeito diferencial da IA generativa entre municípios de alta e baixa conectividade (Triple-DiD). Unidade: ocupação × município × mês. Coeficiente de interesse: β_7 (triple_did = post × alta_exp × alta_conectividade).

Input: `data/output/painel_caged_municipio_anatel.parquet` (Notebook 3a).

0.1 0. Dependências

Conferir pacotes instalados e instalar o necessário para o notebook (pandas, numpy, pyfixest e dependências como platformdirs).

```
# Etapa 3b.0 — Conferir e instalar dependências
!pip install pandas numpy pyfixest platformdirs pyarrow -q
print("Pacotes principais (após install):")
!pip list
```

```
Pacotes principais (após install):
Package                           Version
--------------------------------- --------------
aiohappyeyeballs                  2.6.1
aiohttp                           3.13.2
aiosignal                         1.4.0
altair                            5.4.1
annotated-types                   0.6.0
anyio                             4.12.0
appnope                           0.1.3
argon2-cffi                       25.1.0
argon2-cffi-bindings              25.1.0
arrow                             1.3.0
asttokens                         2.4.1
async-lru                         2.0.5
attrs                             23.2.0
Authlib                           1.6.6
babel                             2.17.0
backports.tarfile                 1.2.0
basedosdados                      2.0.2
beartype                          0.22.8
beautifulsoup4                    4.12.2
bleach                            6.2.0
blinker                           1.8.2
brotli                            1.2.0
cachetools                        5.5.0
catboost                          1.2.8
certifi                           2023.5.7
cffi                              1.17.1
charset-normalizer                2.1.1
```

```
click                                8.1.7
cloudpickle                          3.1.2
cobble                               0.1.4
coloredlogs                          15.0.1
comm                                 0.2.0
commonmark                           0.9.1
contourpy                            1.3.3
coverage                             7.13.0
cryptography                         44.0.0
cssselect2                           0.8.0
cycler                               0.12.1
cyclopts                             4.3.0
dataclasses-json                     0.6.7
db-dtypes                            1.5.0
debugpy                              1.8.0
decorator                            5.1.1
defusedxml                           0.7.1
diskcache                            5.6.3
distlib                              0.3.6
distro                               1.8.0
dnspython                            2.8.0
docstring_parser                     0.17.0
docutils                             0.22.3
einops                               0.7.0
email-validator                      2.3.0
et_xmlfile                           2.0.0
exceptiongroup                       1.3.1
executing                            2.0.1
faicons                              0.2.2
fakeredis                            2.32.1
fastjsonschema                       2.21.1
fastmcp                              2.14.0
fastuuid                             0.14.0
filelock                             3.12.2
filetype                             1.2.0
flatbuffers                          24.3.25
fonttools                            4.61.1
formulaic                            1.2.1
fqdn                                 1.5.1
frozenlist                           1.4.1
fsspec                               2023.12.2
ftfy                                 6.3.1
geobr                                0.2.2
geopandas                            1.1.0
gitdb                                4.0.11
GitPython                            3.1.43
google-ai-generativelanguage         0.6.15
google-api-core                      2.25.1
google-api-python-client             2.178.0
google-auth                          2.40.3
google-auth-httplib2                 0.2.0
google-auth-oauthlib                 1.2.3
google-cloud-bigquery                3.40.0
google-cloud-bigquery-connection     1.20.0
```

```
google-cloud-bigquery-storage      2.36.0
google-cloud-core                  2.5.0
google-cloud-storage               2.19.0
google-crc32c                      1.8.0
google-generativeai                0.8.5
google-resumable-media             2.8.0
googleapis-common-protos           1.70.0
graphviz                           0.21
great-tables                       0.20.0
greenlet                           3.2.4
grpc-google-iam-v1                 0.14.3
grpcio                             1.76.0
grpcio-status                      1.71.2
h11                                0.14.0
html5lib                           1.1
htmltools                          0.6.0
httpcore                           1.0.2
httplib2                           0.22.0
httpx                              0.28.1
httpx-sse                          0.4.3
huggingface-hub                    0.26.2
humanfriendly                      10.0
idna                               3.4
imbalanced-learn                   0.14.0
importlib_metadata                 8.7.0
importlib_resources                6.5.2
iniconfig                          2.3.0
interface-meta                     1.3.0
ipykernel                          6.26.0
ipython                            8.17.2
ipywidgets                         8.1.7
isoduration                        20.11.0
jaraco.classes                     3.4.0
jaraco.context                     6.0.1
jaraco.functools                   4.3.0
jedi                               0.19.1
Jinja2                             3.1.2
jiter                              0.12.0
joblib                             1.4.2
json5                              0.12.1
jsonpatch                          1.33
jsonpointer                        3.0.0
jsonschema                         4.23.0
jsonschema-path                    0.3.4
jsonschema-specifications          2023.12.1
jupyter                            1.1.1
jupyter-book                       2.1.2
jupyter_client                     8.6.0
jupyter-console                    6.6.3
jupyter_core                       5.5.0
jupyter-events                     0.12.0
jupyter-lsp                        2.2.6
jupyter_server                     2.16.0
jupyter_server_terminals           0.5.3
```

```
jupyterlab                    4.4.5
jupyterlab_pygments           0.3.0
jupyterlab_server             2.27.3
jupyterlab_widgets            3.0.15
kaggle                        1.7.4.5
keyring                       25.7.0
kiwisolver                    1.4.9
langchain                     0.3.8
langchain-core                0.3.21
langchain-text-splitters      0.3.0
langsmith                     0.1.145
lightgbm                      4.6.0
litellm                       1.80.10
llvmlite                      0.46.0
loguru                        0.7.3
lupa                          2.6
lxml                          5.3.0
mammoth                       1.9.0
mapclassify                   2.10.0
markdown-it-py                3.0.0
markdownify                   0.14.1
marker-pdf                    0.3.10
markitdown                    0.0.1a3
MarkupSafe                    2.1.3
marshmallow                   3.22.0
matplotlib                    3.10.5
matplotlib-inline             0.1.6
mcp                           1.24.0
mdurl                         0.1.2
mistune                       3.1.3
mock-open                     1.4.0
more-itertools                10.8.0
mpmath                        1.2.1
multidict                     6.0.4
mypy-extensions               1.0.0
mysql-connector-python        9.5.0
narwhals                      2.1.1
nbclient                      0.10.2
nbconvert                     7.16.6
nbformat                      5.10.4
nest-asyncio                  1.5.8
networkx                      3.6.1
nodeenv                       1.10.0
notebook                      7.4.5
notebook_shim                 0.2.4
numba                         0.63.1
numpy                         1.26.4
oauthlib                      3.3.1
onnxruntime                   1.20.1
openai                        2.13.0
openapi-pydantic              0.5.1
opencv-python                 4.10.0.84
openpyxl                      3.1.5
opentelemetry-api             1.39.1
```

```
opentelemetry-exporter-prometheus  0.60b1
opentelemetry-sdk                  1.39.1
opentelemetry-semantic-conventions 0.60b1
orjson                             3.10.7
overrides                          7.7.0
packaging                          25.0
pandas                             2.2.2
pandas-gbq                         0.33.0
pandocfilters                      1.5.1
parso                              0.8.3
pathable                           0.4.4
pathvalidate                       3.3.1
patsy                              1.0.1
pdf2image                          1.17.0
pdfminer.six                       20250506
pdfplumber                         0.11.7
pdftext                            0.3.19
pexpect                            4.8.0
pillow                             10.4.0
pip                                26.0.1
pipenv                             2023.6.26
platformdirs                       4.5.1
playwright                         1.54.0
plotly                             6.3.0
pluggy                             1.6.0
prometheus_client                  0.22.1
prompt-toolkit                     3.0.41
propcache                          0.4.1
proto-plus                         1.26.1
protobuf                           5.29.5
psutil                             7.2.1
ptyprocess                         0.7.0
pure-eval                          0.2.2
puremagic                          1.28
py-key-value-aio                   0.3.0
py-key-value-shared                0.3.0
pyarrow                            17.0.0
pyasn1                             0.6.1
pyasn1_modules                     0.4.2
pycparser                          2.22
pydantic                           2.12.5
pydantic_core                      2.41.5
pydantic-settings                  2.6.1
pydata-google-auth                 1.9.1
pydeck                             0.9.1
pydocket                           0.15.4
pydub                              0.25.1
pydyf                              0.12.1
pyee                               13.0.0
pyfixest                           0.40.1
Pygments                           2.16.1
PyJWT                              2.10.1
pyogrio                            0.12.1
pyparsing                          3.2.3
```

```
PyPDF2                          3.0.1
pypdfium2                       4.30.0
pyperclip                       1.11.0
pyphen                          0.17.2
pyproj                          3.7.2
pytest                          9.0.2
pytest-cov                      7.0.0
python-bcb                      0.3.3
python-dateutil                 2.8.2
python-dotenv                   1.1.0
python-json-logger              3.3.0
python-multipart                0.0.20
python-pptx                     1.0.2
python-slugify                  8.0.4
pytz                            2024.2
PyYAML                          6.0.1
pyzmq                           25.1.1
RapidFuzz                       3.10.1
redis                           7.1.0
referencing                     0.35.1
regex                           2024.11.6
requests                        2.31.0
requests-oauthlib               2.0.0
requests-toolbelt               1.0.0
rfc3339-validator               0.1.4
rfc3986-validator               0.1.1
rich                            14.2.0
rich-rst                        1.3.2
rpds-py                         0.20.0
rsa                             4.9.1
safetensors                     0.4.2
scikit-learn                    1.5.2
scipy                           1.12.0
seaborn                         0.13.2
Send2Trash                      1.8.3
sentence-transformers           5.2.0
setuptools                      68.0.0
shapely                         2.1.0
shellingham                     1.5.4
six                             1.16.0
smmap                           5.0.1
sniffio                         1.3.0
sortedcontainers                2.4.0
soupsieve                       2.6
SpeechRecognition               3.14.0
SQLAlchemy                      2.0.35
sse-starlette                   3.0.3
stack-data                      0.6.3
starlette                       0.50.0
statsmodels                     0.14.5
streamlit                       1.40.1
surya-ocr                       0.6.13
sympy                           1.13.1
tabled-pdf                      0.1.4
```

tabulate	0.9.0
tenacity	8.5.0
terminado	0.18.1
texify	0.2.1
text-unidecode	1.3
threadpoolctl	3.5.0
tiktoken	0.12.0
tinycss2	1.4.0
tinyhtml5	2.0.0
tokenizers	0.20.3
toml	0.10.2
tomlkit	0.11.8
torch	2.5.1
torchaudio	2.5.1
torchsde	0.2.6
torchvision	0.20.1
tornado	6.3.3
tqdm	4.66.1
traitlets	5.13.0
trampoline	0.1.2
transformers	4.46.3
typer	0.20.0
types-python-dateutil	2.9.0.20250809
typing_extensions	4.15.0
typing-inspect	0.9.0
typing-inspection	0.4.2
tzdata	2024.1
uri-template	1.3.0
uritemplate	4.2.0
urllib3	1.26.13
uvicorn	0.38.0
virtualenv	20.23.1
virtualenv-clone	0.5.7
wcwidth	0.2.10
weasyprint	68.1
webcolors	24.11.1
webencodings	0.5.1
websocket-client	1.8.0
websockets	15.0.1
wheel	0.40.0
widgetsnbextension	4.0.14
wordcloud	1.9.4
wrapt	2.1.1
xgboost	3.1.2
xlrd	2.0.2
XlsxWriter	3.2.0
yarl	1.22.0
youtube-transcript-api	0.6.3
zipp	3.23.0
zopfli	0.4.0

0.2 Estratégia de identificação

Elemento	Especificação
Unidade	Ocupação (CBO 4d) × Município × Mês
Tratamento (1)	Alta exposição à IA (ILO score > mediana)
Tratamento (2)	Alta conectividade (penetração BL > mediana)
Evento	Lançamento ChatGPT (Nov/2022)
FE	cbo_4d + uf_periodo
Clustering	id_municipio
Coef. interesse	β_7 (triple_did)

0.3 1. Configuração e carga de dados

Carregar painel 3a, winsorizar salários (P1/P99).

```
# Etapa 3b.1 — Configuração e carga
import warnings
import pandas as pd
import numpy as np
import pyfixest as pf
from pathlib import Path

warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", message=".*multicollinearity.*", category=UserWarnin

# Sempre usar a pasta notebook/data
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_OUTPUT = DATA_ROOT / "output"
CACHE_DIR = DATA_ROOT / "cache"
OUTPUTS_TABLES = PROJECT_ROOT / "notebook" / "outputs" / "tables"
OUTPUTS_FIGURES = PROJECT_ROOT / "notebook" / "outputs" / "figures"
USE_CACHE_3B = True  # Se True, seção 8 usa cache e checkpoints; se False, tudo em mem
for d in [OUTPUTS_TABLES, OUTPUTS_FIGURES, CACHE_DIR]:
    d.mkdir(parents=True, exist_ok=True)

PAINEL_FILE = DATA_OUTPUT / "painel_caged_municipio_anatel.parquet"
PAINEL_FILE_V2 = DATA_OUTPUT / "painel_caged_municipio_anatel_v2.parquet"
VCOV_SPEC = {"CRV1": "id_municipio"}
OUTCOMES = {"ln_salario_real_adm": "Log(Sal. Real Adm.)", "ln_admissoes": "Log(Admissõ

print("Data output:", DATA_OUTPUT.resolve())
```

```
if PAINEL_FILE_V2.exists():
    df = pd.read_parquet(PAINEL_FILE_V2)
    print("Painel v2 (faixa etária, fibra, capital) carregado:", PAINEL_FILE_V2.resolv
else:
    df = pd.read_parquet(PAINEL_FILE)
    print("Painel (sem v2) carregado:", PAINEL_FILE.resolve())
for c in ["salario_medio_adm", "salario_real_adm"]:
    if c in df.columns:
        lo, hi = df[c].quantile(0.01), df[c].quantile(0.99)
        df[c] = df[c].clip(lo, hi)
if "ln_salario_real_adm" in df.columns:
    df["ln_salario_real_adm"] = np.log(df["salario_real_adm"].clip(lower=1))
print(f"Painel: {len(df):,} obs | {df['id_municipio'].nunique():,} municípios | {df['c
```

```
Data output: /Users/manebrasil/Documents/Projects/Dissetação
Mestrado/notebook/data/output
Painel v2 (faixa etária, fibra, capital) carregado:
/Users/manebrasil/Documents/Projects/Dissetação
Mestrado/notebook/data/output/painel_caged_municipio_anatel_v2.parquet
Painel: 5,534,808 obs | 657 municípios | 614 ocupações
```

0.4 2. Tabela de balanço por conectividade (pré-tratamento)

```
# Etapa 3b.2 — Balanço
pre = df[df["post"] == 0]
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))
```

alta_conectividade	n_obs	admissoes_media	salario_medio	penetracao_media
0	205077	5.02	1330.58	0.18
1	2126257	15.02	1742.66	0.69

0.5 3. DiD por subgrupo de conectividade (motivação)

Estimar DiD (post × alta_exp) separadamente para municípios de alta e baixa conectividade.

```
# Etapa 3b.3 — DiD por subgrupo
df["post_alta_exp"] = df["post"] * df["alta_exp"]
for conn_label, mask in [("Alta conect.", df["alta_conectividade"] == 1), ("Baixa cone
    d = df.loc[mask].dropna(subset=["ln_salario_real_adm"])
    m = pf.feols("ln_salario_real_adm ~ post_alta_exp | cbo_4d + uf_periodo", data=d,
    c = m.coef().get("post_alta_exp", m.coef().iloc[0])
    print(f"{conn_label}: coef = {float(c):.4f}, se = {float(m.se().loc['post_alta_exp
```

```
Alta conect.: coef = -0.0058, se = 0.0018
Baixa conect.: coef = -0.0089, se = 0.0046
```

0.6 4. Triple-DiD — Modelo principal

outcome ~ triple_did + post_alta_exp + post_alta_conect + alta_exp_alta_conect | cbo_4d + uf_periodo

```
# Etapa 3b.4 — Triple-DiD principal
formula = "ln_salario_real_adm ~ triple_did + post_alta_exp + post_alta_conect + alta_
m = pf.feols(formula, data=df.dropna(subset=["ln_salario_real_adm"]), vcov=VCOV_SPEC)
print(m.summary())
results_triple = pd.DataFrame({"coef": m.coef(), "se": m.se(), "p_value": m.pvalue()})
results_triple.to_csv(OUTPUTS_TABLES / "triple_did_main_etapa3b.csv")
```

```
###

Estimation:  OLS
Dep. var.: ln_salario_real_adm, Fixed effects: cbo_4d+uf_periodo
Inference:  CRV1
Observations:  5534808

| Coefficient          |   Estimate |   Std. Error |   t value |   Pr(>|t|) |   2.5% |
97.5% |
|:---------------------|-----------:|-------------:|----------:|-----------:|------
-:|--------:|
| triple_did           |     -0.016 |        0.008 |    -1.888 |      0.059 | -0.032 |
0.001 |
| post_alta_exp        |      0.008 |        0.008 |     1.069 |      0.285 | -0.007 |
0.024 |
| post_alta_conect     |      0.017 |        0.008 |     2.239 |      0.025 |  0.002 |
0.032 |
| alta_exp_alta_conect |      0.037 |        0.010 |     3.743 |      0.000 |  0.017 |
0.056 |
---
RMSE: 0.555 R2: 0.969 R2 Within: 0.0
None
```

0.7 5. Event study por grupo de conectividade

Dummies (tempo_relativo × alta_exp) para alta e baixa conectividade; referência t = -1.

```
# Etapa 3b.5 — Event study por conectividade
# Dummies (tempo_relativo × alta_exp) para alta e baixa conectividade; referência t =
# Nomes das variáveis: sem "-" na fórmula (formulaic interpreta "did_t-12" como did_t
def _nome_did(t):
    return f"did_m{-t}" if t < 0 else f"did_{t}"

BIN_MIN, BIN_MAX, REF = -12, 24, -1
df["t_bin"] = df["tempo_relativo_meses"].clip(lower=BIN_MIN, upper=BIN_MAX)
for conn_val, label in [(1, "Alta"), (0, "Baixa")]:
    d = df[(df["alta_conectividade"] == conn_val) & df["ln_salario_real_adm"].notna()]
    ts = [t for t in sorted(d["t_bin"].unique()) if t != REF]
    did_vars = [_nome_did(t) for t in ts]
    for t, v in zip(ts, did_vars):
        d[v] = ((d["t_bin"] == t) & (d["alta_exp"] == 1)).astype(int)
```

```
    if did_vars:
        m = pf.feols(f"ln_salario_real_adm ~ {' + '.join(did_vars)} | cbo_4d + uf_peri
        print(f"{label} conect.: {len(ts)} coeficientes estimados")
```

```
Alta conect.: 36 coeficientes estimados
Baixa conect.: 36 coeficientes estimados
```

0.8 6. Testes de robustez e síntese

Placebo temporal (Dez/2021); cutoff Q75. Correção FDR para múltiplos outcomes (opcional). Síntese: comparar com Etapa 2 — efeito concentrado onde a adoção de IA é viável.

```
# Etapa 3b.6 — Robustez (placebo) e múltiplos outcomes
# Placebo: post = 1 a partir de Dez/2021
df["post_placebo"] = ((df["ano"] == 2021) & (df["mes"] >= 12)) | (df["ano"] > 2021)
df["triple_did_placebo"] = df["post_placebo"].astype(int) * df["alta_exp"] * df["alta_
df["post_alta_exp_placebo"] = df["post_placebo"].astype(int) * df["alta_exp"]
m_placebo = pf.feols("ln_salario_real_adm ~ triple_did_placebo + post_alta_exp_placebo
                     data=df.dropna(subset=["ln_salario_real_adm"]), vcov=VCOV_SPEC)
print("Placebo Dez/2021 (triple_did_placebo):", m_placebo.coef().get("triple_did_place
print("Esperado: não significativo.")
if USE_CACHE_3B:
    import json
    with open(CACHE_DIR / "placebo_secao6.json", "w") as f:
        json.dump({"triple_did_placebo": {"coef": float(m_placebo.coef().get("triple_d
```

```
Placebo Dez/2021 (triple_did_placebo): 0.03653039627709792 p = 5.672064404560473e-05
Esperado: não significativo.
```

0.9 7. Correção de tendência diferencial pré (Caminho 1)

Se o placebo falha por tendência pré-existente no grupo alta_exp × alta_conectividade, controlar por **alta_exp × alta_conectividade × trend** (trend = 1, 2, ..., T). Se β_7 (triple_did) permanece significativo após incluir a tendência, o efeito é robusto; se não, o efeito era explicado pela tendência pré.

```
# Etapa 3b.7 — Tendência diferencial pré (Caminho 1)
periodos_ordenados = sorted(df["periodo"].unique())
trend_map = {p: i + 1 for i, p in enumerate(periodos_ordenados)}
df["trend"] = df["periodo"].map(trend_map)
df["trend_exp_conect"] = df["alta_exp"] * df["alta_conectividade"] * df["trend"]
df["trend_exp"] = df["alta_exp"] * df["trend"]
df["trend_conect"] = df["alta_conectividade"] * df["trend"]

outcome = "ln_salario_real_adm"
formula_trend = (
    f"{outcome} ~ triple_did + post_alta_exp + post_alta_conect + alta_exp_alta_conect
    " + trend_exp_conect + trend_exp + trend_conect | cbo_4d + uf_periodo"
)
model_trend = pf.feols(formula_trend, data=df.dropna(subset=[outcome]), vcov=VCOV_SPEC
print("Modelo com tendência diferencial pré:")
print(model_trend.summary())
```

```
# Comparar com modelo original (seção 4)
formula_orig = "ln_salario_real_adm ~ triple_did + post_alta_exp + post_alta_conect +
model_orig = pf.feols(formula_orig, data=df.dropna(subset=["ln_salario_real_adm"]), vc
print("\nComparação: Original vs. com tendência")
pf.etable([model_orig, model_trend])
```

```
Modelo com tendência diferencial pré:
###

Estimation:  OLS
Dep. var.: ln_salario_real_adm, Fixed effects: cbo_4d+uf_periodo
Inference:  CRV1
Observations:  5534808

| Coefficient          |   Estimate |   Std. Error |   t value |   Pr(>|t|) |   2.5% |
97.5% |
|:---------------------|-----------:|-------------:|----------:|-----------:|------
-:|-------:|
| triple_did           |      0.006 |        0.010 |     0.668 |      0.504 | -0.012 |
0.025 |
| post_alta_exp        |     -0.005 |        0.009 |    -0.596 |      0.552 | -0.023 |
0.012 |
| post_alta_conect     |     -0.015 |        0.007 |    -2.070 |      0.039 | -0.030 |
-0.001 |
| alta_exp_alta_conect |      0.034 |        0.010 |     3.375 |      0.001 |  0.014 |
0.054 |
| trend_exp_conect     |     -0.001 |        0.000 |    -1.321 |      0.187 | -0.001 |
0.000 |
| trend_exp            |      0.000 |        0.000 |     0.616 |      0.538 | -0.000 |
0.001 |
| trend_conect         |      0.001 |        0.000 |     2.463 |      0.014 |  0.000 |
0.002 |
---
RMSE: 0.555 R2: 0.969 R2 Within: 0.0
None

Comparação: Original vs. com tendência
```

	ln_salario_real_adm	
	(1)	(2)
triple_did	-0.016 (0.008)	0.006 (0.010)
post_alta_exp	0.008 (0.008)	-0.005 (0.009)
post_alta_conect	0.017* (0.008)	-0.015* (0.007)
alta_exp_alta_conect	0.037*** (0.010)	0.034*** (0.010)

Significance levels: * p < 0.05, ** p < 0.01, *** p < 0.001. Format of coefficient cell: Coefficient (Std. Error)

	ln_salario_real_adm	
	(1)	(2)
trend_exp_conect		-0.001 (0.000)
trend_exp		0.000 (0.000)
trend_conect		0.001* (0.000)
uf_periodo	x	x
cbo_4d	x	x
Observations	5534808	5534808
S.E. type	by: id_municipio	by: id_municipio
R^2	0.969	0.969
R^2 Within	0.000	0.000

Significance levels: * p < 0.05, ** p < 0.01, *** p < 0.001. Format of coefficient cell: Coefficient (Std. Error)

0.10 8. Proxies alternativos de conectividade (Caminho 2)

Testar quatro proxies além da mediana de penetração: (1) extremos Q75 vs Q25, (2) % fibra óptica, (3) tratamento contínuo (dose-resposta), (4) apenas capitais. Para cada um, rodar Triple-DiD e placebo temporal (Dez/2021). Se algum proxy passar no placebo, sugere que o problema está na definição de conectividade.

Nota: Rode as células 8a a 8g em sequência. Com `USE_CACHE_3B = True`, cada célula carrega só a amostra necessária do cache e libera memória ao final.

```
# Etapa 3b.8a — Preparação e gravação do cache (Proxies)
outcome = "ln_salario_real_adm"

# Garantir placebo da seção 6; se não rodou, calcula aqui
try:
    _ = m_placebo.pvalue()
except NameError:
    df["post_placebo"] = ((df["ano"] == 2021) & (df["mes"] >= 12)) | (df["ano"] > 2021
    df["triple_did_placebo"] = df["post_placebo"].astype(int) * df["alta_exp"] * df["a
    df["post_alta_exp_placebo"] = df["post_placebo"].astype(int) * df["alta_exp"]
    m_placebo = pf.feols("ln_salario_real_adm ~ triple_did_placebo + post_alta_exp_pla

# Variáveis de dose e capital
exp_col = "ilo_exposure_score" if "ilo_exposure_score" in df.columns else "alta_exp"
df["dose_triple"] = df["penetracao_bl"] * (df[exp_col] if exp_col == "ilo_exposure_sco
df["dose_exp_post"] = (df[exp_col].astype(float) if exp_col == "alta_exp" else df[exp_
df["dose_conect_post"] = df["penetracao_bl"] * df["post"]
df["dose_exp_conect"] = (df[exp_col].astype(float) if exp_col == "alta_exp" else df[ex
if "capital" in df.columns:
    df["triple_did_capital"] = df["post"] * df["alta_exp"] * df["capital"]
    df["post_capital"] = df["post"] * df["capital"]
```

```
    df["alta_exp_capital"] = df["alta_exp"] * df["capital"]

# Amostra de regressão
print("8a: Preparando amostra de regressão…")
df_reg = df.loc[df[outcome].notna()].copy()

# Amostra extremos Q25/Q75
q25, q75 = df["penetracao_bl"].quantile(0.25), df["penetracao_bl"].quantile(0.75)
df_extremos = df[(df["penetracao_bl"] <= q25) | (df["penetracao_bl"] >= q75)].copy()
df_extremos["alta_conect_extremo"] = (df_extremos["penetracao_bl"] >= q75).astype(int)
df_extremos["triple_did_extremo"] = df_extremos["post"] * df_extremos["alta_exp"] * df
df_extremos["post_alta_conect_ext"] = df_extremos["post"] * df_extremos["alta_conect_e
df_extremos["alta_exp_alta_conect_ext"] = df_extremos["alta_exp"] * df_extremos["alta_
df_extremos["post_placebo"] = ((df_extremos["ano"] == 2021) & (df_extremos["mes"] >= 1
df_extremos["triple_placebo_ext"] = df_extremos["post_placebo"].astype(int) * df_extre
df_extremos["post_placebo_alta_exp"] = df_extremos["post_placebo"].astype(int) * df_ex
df_extremos["post_placebo_alta_conect_ext"] = df_extremos["post_placebo"].astype(int)
df_ext = df_extremos.loc[df_extremos[outcome].notna()].copy()

if USE_CACHE_3B:
    df_reg.to_parquet(CACHE_DIR / "painel_3b_df_reg.parquet", index=False)
    df_ext.to_parquet(CACHE_DIR / "painel_3b_df_ext.parquet", index=False)
    if (CACHE_DIR / "resultados_proxy.csv").exists():
        (CACHE_DIR / "resultados_proxy.csv").unlink()
    print("Cache salvo em", CACHE_DIR.resolve())
    del df_reg, df_extremos, df_ext
    print("8a concluído. Memória liberada.")
else:
    print("8a concluído (sem cache). df_reg e df_ext em memória para 8b–8f.")
```

```
8a: Preparando amostra de regressão…
Cache salvo em /Users/manebrasil/Documents/Projects/Dissetação
Mestrado/notebook/data/cache
8a concluído. Memória liberada.
```

```
# Etapa 3b.8b — Proxy Original (mediana)
outcome = "ln_salario_real_adm"
RESULTADOS_PROXY_CSV = CACHE_DIR / "resultados_proxy.csv"

if USE_CACHE_3B and (CACHE_DIR / "painel_3b_df_reg.parquet").exists():
    df_reg = pd.read_parquet(CACHE_DIR / "painel_3b_df_reg.parquet")
else:
    if "df_reg" not in dir():
        raise RuntimeError("Rode a célula 8a antes.")

# m_placebo: do escopo (seção 6) ou do cache
try:
    pval_placebo = m_placebo.pvalue().get("triple_did_placebo")
except NameError:
    import json
    with open(CACHE_DIR / "placebo_secao6.json") as f:
        pval_placebo = json.load(f)["triple_did_placebo"]["pval"]
```

```
formula_orig = "ln_salario_real_adm ~ triple_did + post_alta_exp + post_alta_conect +
m_orig = pf.feols(formula_orig, data=df_reg, vcov=VCOV_SPEC)
row = {"proxy": "Original (mediana)", "coef": m_orig.coef().get("triple_did"), "se": m
pd.DataFrame([row]).to_csv(RESULTADOS_PROXY_CSV, index=False)
if USE_CACHE_3B:
    del df_reg
print("8b concluído.")
```

```
8b concluído.
```

```
# Etapa 3b.8c — Proxy Extremos (Q75/Q25)
outcome = "ln_salario_real_adm"

if USE_CACHE_3B and (CACHE_DIR / "painel_3b_df_ext.parquet").exists():
    df_ext = pd.read_parquet(CACHE_DIR / "painel_3b_df_ext.parquet")
else:
    if "df_ext" not in dir():
        raise RuntimeError("Rode a célula 8a antes.")

formula_ext = f"{outcome} ~ triple_did_extremo + post_alta_exp + post_alta_conect_ext
m_ext = pf.feols(formula_ext, data=df_ext, vcov=VCOV_SPEC)
m_placebo_ext = pf.feols(f"{outcome} ~ triple_placebo_ext + post_placebo_alta_exp + po
pval_placebo_ext = m_placebo_ext.pvalue().get("triple_placebo_ext", np.nan)
row = {"proxy": "Extremos Q75/Q25", "coef": m_ext.coef().get("triple_did_extremo"), "s
pd.DataFrame([row]).to_csv(RESULTADOS_PROXY_CSV, mode="a", header=False, index=False)
if USE_CACHE_3B:
    del df_ext
print("8c concluído.")
```

```
8c concluído.
```

```
# Etapa 3b.8d — Proxy % Fibra (requer painel v2)
outcome = "ln_salario_real_adm"

if USE_CACHE_3B and (CACHE_DIR / "painel_3b_df_reg.parquet").exists():
    df_reg = pd.read_parquet(CACHE_DIR / "painel_3b_df_reg.parquet")
else:
    if "df_reg" not in dir():
        raise RuntimeError("Rode a célula 8a antes.")

if "triple_did_fibra" in df_reg.columns:
    df_reg["triple_placebo_fibra"] = df_reg["post_placebo"].astype(int) * df_reg["alta
    df_reg["post_placebo_alta_fibra"] = df_reg["post_placebo"].astype(int) * df_reg["a
    formula_fibra = f"{outcome} ~ triple_did_fibra + post_alta_exp + post_alta_fibra +
    m_fibra = pf.feols(formula_fibra, data=df_reg, vcov=VCOV_SPEC)
    m_pl_fibra = pf.feols(f"{outcome} ~ triple_placebo_fibra + post_alta_exp_placebo +
    row = {"proxy": "% Fibra", "coef": m_fibra.coef().get("triple_did_fibra"), "se": m
else:
    row = {"proxy": "% Fibra", "coef": np.nan, "se": np.nan, "pval": np.nan, "placebo_
pd.DataFrame([row]).to_csv(RESULTADOS_PROXY_CSV, mode="a", header=False, index=False)
if USE_CACHE_3B:
```

```
    del df_reg
print("8d concluído.")
```

```
/opt/homebrew/lib/python3.10/site-packages/pyfixest/estimation/feols_.py:2847:
UserWarning:
            3 variables dropped due to multicollinearity.
            The following variables are dropped: ['triple_did_fibra',
'post_alta_fibra', 'alta_exp_alta_fibra'].

  warnings.warn(
/opt/homebrew/lib/python3.10/site-packages/pyfixest/estimation/feols_.py:2847:
UserWarning:
            3 variables dropped due to multicollinearity.
            The following variables are dropped: ['triple_placebo_fibra',
'post_placebo_alta_fibra', 'alta_exp_alta_fibra'].

  warnings.warn(

8d concluído.
```

```
# Etapa 3b.8e — Proxy Contínuo (dose-resposta)
outcome = "ln_salario_real_adm"

if USE_CACHE_3B and (CACHE_DIR / "painel_3b_df_reg.parquet").exists():
    df_reg = pd.read_parquet(CACHE_DIR / "painel_3b_df_reg.parquet")
else:
    if "df_reg" not in dir():
        raise RuntimeError("Rode a célula 8a antes.")

formula_cont = f"{outcome} ~ dose_triple + dose_exp_post + dose_conect_post + dose_exp
m_cont = pf.feols(formula_cont, data=df_reg, vcov=VCOV_SPEC)
row = {"proxy": "Contínuo", "coef": m_cont.coef().get("dose_triple"), "se": m_cont.se(
pd.DataFrame([row]).to_csv(RESULTADOS_PROXY_CSV, mode="a", header=False, index=False)
if USE_CACHE_3B:
    del df_reg
print("8e concluído.")
```

```
8e concluído.
```

```
# Etapa 3b.8f — Proxy Capitais (requer painel v2)
outcome = "ln_salario_real_adm"

if USE_CACHE_3B and (CACHE_DIR / "painel_3b_df_reg.parquet").exists():
    df_reg = pd.read_parquet(CACHE_DIR / "painel_3b_df_reg.parquet")
else:
    if "df_reg" not in dir():
        raise RuntimeError("Rode a célula 8a antes.")

if "capital" in df_reg.columns:
    try:
        formula_cap = f"{outcome} ~ triple_did_capital + post_alta_exp + post_capital
        m_cap = pf.feols(formula_cap, data=df_reg, vcov=VCOV_SPEC)
```

```
        row = {"proxy": "Capitais", "coef": m_cap.coef().get("triple_did_capital"), "s
    except Exception:
        row = {"proxy": "Capitais", "coef": np.nan, "se": np.nan, "pval": np.nan, "pla
else:
    row = {"proxy": "Capitais", "coef": np.nan, "se": np.nan, "pval": np.nan, "placebo
pd.DataFrame([row]).to_csv(RESULTADOS_PROXY_CSV, mode="a", header=False, index=False)
if USE_CACHE_3B:
    del df_reg
print("8f concluído.")
```

```
8f concluído.
```

```
# Etapa 3b.8g — Montagem da tabela
RESULTADOS_PROXY_CSV = CACHE_DIR / "resultados_proxy.csv"
if not RESULTADOS_PROXY_CSV.exists():
    print("Rode as células 8b–8f antes para gerar resultados_proxy.csv")
else:
    tab_proxy = pd.read_csv(RESULTADOS_PROXY_CSV)
    tab_proxy["sig"] = tab_proxy["pval"].apply(lambda p: "***" if p < 0.01 else ("**"
    print("Tabela comparativa — Triple-DiD por proxy de conectividade")
    print(tab_proxy.to_string())
```

```
Tabela comparativa — Triple-DiD por proxy de conectividade
                proxy      coef        se          pval  placebo_pval  sig
0  Original (mediana) -0.015759  0.008346  5.944694e-02      0.000057    *
1    Extremos Q75/Q25 -0.021561  0.007295  3.322350e-03      0.017556  ***
2             % Fibra       NaN       NaN           NaN           NaN
3            Contínuo -0.058241  0.011591  6.506810e-07           NaN  ***
4            Capitais -0.113379  0.016465  1.345479e-11           NaN  ***
```

0.11 9. Decomposição etária do Triple-DiD

Rodar Triple-DiD para cada outcome por faixa etária (admissões e salário por jovem/intermediário/senior, share de jovens/seniores, razão salarial jovem/senior). Se a hipótese de compensação etária estiver correta: share_jovem ↓, share_senior ↑, ln_sal_real_jovem ↓, razao_sal_jovem_senior ↓.

Cache: Com `USE_CACHE_3B = True`, os resultados são salvos em `resultados_idade.csv`. Se o kernel cair ou você reexecutar a célula, os outcomes já calculados são carregados do cache e só os faltantes são estimados (um por vez, com menos memória).

```
# Etapa 3b.9 — Decomposição etária do Triple-DiD (com cache; reaproveita resultados an
outcomes_idade = {
    "Admissões jovens (log)": "ln_adm_jovem",
    "Admissões intermediários (log)": "ln_adm_intermediario",
    "Admissões seniores (log)": "ln_adm_senior",
    "Share jovens nas admissões": "share_jovem",
    "Share seniores nas admissões": "share_senior",
    "Salário real jovens (log)": "ln_sal_real_jovem",
    "Salário real intermediários (log)": "ln_sal_real_intermediario",
    "Salário real seniores (log)": "ln_sal_real_senior",
    "Razão salarial jovem/senior": "razao_sal_jovem_senior",
```

```
}

CACHE_IDADE_CSV = CACHE_DIR / "resultados_idade.csv"
COLS_REGR = ["triple_did", "post_alta_exp", "post_alta_conect", "alta_exp_alta_conect"
resultados_idade = {}
if CACHE_IDADE_CSV.exists() and USE_CACHE_3B:
    cache_df = pd.read_csv(CACHE_IDADE_CSV)
    for _, r in cache_df.iterrows():
        resultados_idade[r["outcome_name"]] = {"coef": r["coef"], "se": r["se"], "pval
    print(f"Carregados {len(resultados_idade)} resultados do cache.")

formula_base = "{outcome} ~ triple_did + post_alta_exp + post_alta_conect + alta_exp_a
for nome, outcome_var in outcomes_idade.items():
    if nome in resultados_idade:
        continue
    if outcome_var not in df.columns:
        print(f"SKIP {nome}: coluna {outcome_var} não encontrada (rode 3a com painel v
        continue
    # Só as colunas necessárias para a regressão (reduz memória)
    use_cols = [outcome_var] + [c for c in COLS_REGR if c in df.columns]
    mask = df[outcome_var].notna()
    df_valid = df.loc[mask, use_cols].copy()
    if len(df_valid) < 1000:
        print(f"SKIP {nome}: apenas {len(df_valid)} obs válidas")
        del df_valid
        continue
    formula = formula_base.format(outcome=outcome_var)
    try:
        model = pf.feols(formula, data=df_valid, vcov=VCOV_SPEC)
        coef = float(model.coef().get("triple_did"))
        se = float(model.se().get("triple_did"))
        pval = float(model.pvalue().get("triple_did"))
        n = len(df_valid)
        resultados_idade[nome] = {"coef": coef, "se": se, "pval": pval, "n": n}
        if USE_CACHE_3B:
            pd.DataFrame([{"outcome_name": nome, "outcome_var": outcome_var, "coef": c
        print(f"OK {nome}")
    except Exception as e:
        print(f"ERRO {nome}: {e}")
    del df_valid

if resultados_idade:
    df_resultados_idade = pd.DataFrame(resultados_idade).T
    df_resultados_idade["sig"] = df_resultados_idade["pval"].apply(
        lambda p: "***" if p < 0.01 else ("**" if p < 0.05 else ("*" if p < 0.1 else "
    )
    print("Triple-DiD por outcome etário (coef = triple_did):")
    print(df_resultados_idade.to_string())
else:
    print("Nenhum outcome etário disponível. Execute o Notebook 3a e exporte o painel
```

```
Carregados 9 resultados do cache.
Triple-DiD por outcome etário (coef = triple_did):
                                      coef        se          pval          n  sig
```

```
Admissões jovens (log)            -0.330124  0.043301  8.659740e-14  5534808.0  ***
Admissões intermediários (log)    -0.351799  0.050120  5.585532e-12  5534808.0  ***
Admissões seniores (log)          -0.182257  0.027490  7.022161e-11  5534808.0  ***
Share jovens nas admissões        -0.001168  0.005800  8.405146e-01  4404189.0
Share seniores nas admissões      -0.001866  0.002645  4.805814e-01  4404189.0
Salário real jovens (log)         -0.631805  0.076151  6.661338e-16  5534808.0  ***
Salário real intermediários (log) -0.534896  0.076899  8.498313e-12  5534808.0  ***
Salário real seniores (log)       -0.723624  0.116553  9.486867e-10  5534808.0  ***
Razão salarial jovem/senior       -7.271724  8.598501  3.980303e-01  1391959.0
```

0.12 10. Modelo reformulado: impacto sobre jovens

Hipótese: após o ChatGPT, a **participação de jovens nas contratações** (share_jovem) caiu mais em ocupações expostas à IA e em municípios conectados. Outcome principal: **share_jovem** = admissões de jovens (<30) / total de admissões. Três especificações (pura, com tendência, com controles demográficos), placebo temporal e event study por conectividade.

Memória: A célula usa apenas as colunas necessárias em cada bloco e libera os DataFrames entre as etapas para evitar travamento.

```
# Etapa 3b.10 — Modelo reformulado: share_jovem (DataFrames mínimos para reduzir memór
if "share_jovem" not in df.columns:
    print("share_jovem não disponível. Execute o Notebook 3a e exporte o painel v2.")
else:
    # Colunas necessárias para as 3 especificações (evita cópia do painel inteiro)
    cols_j = ["share_jovem", "triple_did", "post_alta_exp", "post_alta_conect", "alta_
    if "periodo" in df.columns:
        cols_j.append("periodo")
    for c in ["pct_superior_adm", "pct_mulher_adm", "ln_pib_pc"]:
        if c in df.columns:
            cols_j.append(c)
    cols_j = [c for c in cols_j if c in df.columns]
    df_j = df.loc[df["share_jovem"].notna(), cols_j].copy()
    if "periodo" in df_j.columns and "trend_exp_conect" not in df_j.columns:
        periodos_ordenados = sorted(df_j["periodo"].unique())
        trend_map = {p: i + 1 for i, p in enumerate(periodos_ordenados)}
        df_j["trend"] = df_j["periodo"].map(trend_map)
        df_j["trend_exp_conect"] = df_j["alta_exp"] * df_j["alta_conectividade"] * df_
        df_j["trend_exp"] = df_j["alta_exp"] * df_j["trend"]
        df_j["trend_conect"] = df_j["alta_conectividade"] * df_j["trend"]

    formula_j1 = "share_jovem ~ triple_did + post_alta_exp + post_alta_conect + alta_e
    m_jovem_1 = pf.feols(formula_j1, data=df_j, vcov=VCOV_SPEC)
    formula_j2 = "share_jovem ~ triple_did + post_alta_exp + post_alta_conect + alta_e
    m_jovem_2 = pf.feols(formula_j2, data=df_j, vcov=VCOV_SPEC)
    controles = [c for c in ["pct_superior_adm", "pct_mulher_adm", "ln_pib_pc"] if c i
    formula_j3 = "share_jovem ~ triple_did + post_alta_exp + post_alta_conect + alta_e
    m_jovem_3 = pf.feols(formula_j3, data=df_j.dropna(subset=controles) if controles e
    print("Três especificações — outcome: share_jovem")
    pf.etable([m_jovem_1, m_jovem_2, m_jovem_3])
    del df_j

    # Placebo: só colunas necessárias
```

```
    mask_pre = (df["ano"] < 2022) | ((df["ano"] == 2022) & (df["mes"] < 11)) if "ano"
    cols_pre = ["share_jovem", "alta_exp", "alta_conectividade", "alta_exp_alta_conect
    if "ano" in df.columns:
        cols_pre.extend(["ano", "mes"])
    if "periodo_dt" in df.columns:
        cols_pre.append("periodo_dt")
    cols_pre = [c for c in cols_pre if c in df.columns]
    df_pre = df.loc[mask_pre, cols_pre].copy()
    df_pre = df_pre.dropna(subset=["share_jovem"])
    if "periodo_dt" not in df_pre.columns and "ano" in df_pre.columns:
        df_pre["periodo_dt"] = pd.to_datetime(df_pre["ano"].astype(str) + "-" + df_pre
    df_pre["post_placebo"] = (df_pre["periodo_dt"] >= pd.Timestamp("2021-12-01")).asty
    df_pre["triple_placebo"] = df_pre["post_placebo"].astype(int) * df_pre["alta_exp"]
    df_pre["post_placebo_alta_exp"] = df_pre["post_placebo"].astype(int) * df_pre["alt
    df_pre["post_placebo_alta_conect"] = df_pre["post_placebo"].astype(int) * df_pre["
    if "alta_exp_alta_conect" not in df_pre.columns:
        df_pre["alta_exp_alta_conect"] = df_pre["alta_exp"] * df_pre["alta_conectivida
    m_placebo_j = pf.feols("share_jovem ~ triple_placebo + post_placebo_alta_exp + pos
    print(f"Placebo share_jovem (Dez/2021): coef = {m_placebo_j.coef().get('triple_pla
    del df_pre

    # Event study: só colunas necessárias; um grupo de conectividade por vez
    if "tempo_relativo_meses" in df.columns:
        BIN_MIN, BIN_MAX, REF = -12, 24, -1
        cols_es = ["share_jovem", "tempo_relativo_meses", "alta_conectividade", "alta_
        cols_es = [c for c in cols_es if c in df.columns]
        df_es = df.loc[df["share_jovem"].notna(), cols_es].copy()
        df_es["t_bin"] = df_es["tempo_relativo_meses"].clip(lower=BIN_MIN, upper=BIN_M
        for conn_val, label in [(1, "Alta"), (0, "Baixa")]:
            d = df_es[df_es["alta_conectividade"] == conn_val].copy()
            ts = [t for t in sorted(d["t_bin"].unique()) if t != REF]
            did_vars = [f"did_m{-t}" if t < 0 else f"did_{t}" for t in ts]
            for t, v in zip(ts, did_vars):
                d[v] = ((d["t_bin"] == t) & (d["alta_exp"] == 1)).astype(int)
            if did_vars:
                m_es = pf.feols(f"share_jovem ~ {' + '.join(did_vars)} | cbo_4d + uf_p
                print(f"Event study share_jovem — {label} conect.: {len(ts)} coeficien
            del d
        del df_es
    print("3b.10 concluído.")
```

```
/opt/homebrew/lib/python3.10/site-
packages/pyfixest/estimation/model_matrix_fixest_.py:215: UserWarning: 1 singleton
fixed effect(s) detected. These observations are dropped from the model.
  warnings.warn(
/opt/homebrew/lib/python3.10/site-
packages/pyfixest/estimation/model_matrix_fixest_.py:215: UserWarning: 1 singleton
fixed effect(s) detected. These observations are dropped from the model.
  warnings.warn(
/opt/homebrew/lib/python3.10/site-
packages/pyfixest/estimation/model_matrix_fixest_.py:215: UserWarning: 1 singleton
fixed effect(s) detected. These observations are dropped from the model.
  warnings.warn(
```

```
Três especificações — outcome: share_jovem

/opt/homebrew/lib/python3.10/site-
packages/pyfixest/estimation/model_matrix_fixest_.py:215: UserWarning: 3 singleton
fixed effect(s) detected. These observations are dropped from the model.
  warnings.warn(

Placebo share_jovem (Dez/2021): coef = -0.0064, p = 0.3391

/opt/homebrew/lib/python3.10/site-
packages/pyfixest/estimation/model_matrix_fixest_.py:215: UserWarning: 1 singleton
fixed effect(s) detected. These observations are dropped from the model.
  warnings.warn(

Event study share_jovem — Alta conect.: 36 coeficientes

/opt/homebrew/lib/python3.10/site-
packages/pyfixest/estimation/model_matrix_fixest_.py:215: UserWarning: 6 singleton
fixed effect(s) detected. These observations are dropped from the model.
  warnings.warn(

Event study share_jovem — Baixa conect.: 36 coeficientes
3b.10 concluído.
```

0.13 Verificação metodológica (conferência com o plano Etapa 3)

- **β_7 (triple_did):** Interpretar como a diferença do efeito da IA entre municípios de alta e baixa conectividade. $\beta_7 < 0$ em salário real = efeito mais negativo onde a adoção de IA é viável.
- **Conectividade pré-tratamento:** Medida Jan–Out/2022 para evitar endogeneidade.
- **FE:** cbo_4d + uf_periodo absorvem choques estaduais; robustez com id_municipio + periodo.
- **Clustering:** Por id_municipio; robustez multiway (ocupação + município).
- **Comparação com Etapa 2:** Apresentar 3 como extensão — efeito médio concentrado nos municípios conectados.