

Supplementary Text

This supplementary text contains copy/paste of the following: (1) The prompt for the final Codex adversarial review of the analysis code written by Claude; (2) The findings from that final adversarial review reported by Codex; (3) The summary of the final correction made by Claude in response to that final adversarial review; and (4) The summary of the full reproduction of the analysis from scratch by Claude following that final correction.

1. Text of *codex_review_2026-09-15_comprehensive_prompt.md* :

You are conducting a read-only adversarial code review of a climate-attribution figure-reproduction repository at this path. Do not modify any files -- this is analysis only; report findings back in your final message.

What this repository does

It reproduces 10 figures (`figure1\_central\_europe\_2023.png`, `figure2\_2000\_2024.png`, `figure3\_2000\_2024.png`, `figureS1\_2000\_2024.png` through `figureS7\_2000\_2024.png`, all in `figures/`, generated by `scripts/plot\_figure1.py` through `plot\_figureS7.py` via their `run\_plot\_figure\*.sh` wrappers) comparing CNN-predicted regional temperatures against observed ERA5 temperatures for annual hottest 3-day heat events across 3 European regions (northern/central/southern Europe), using Z500-pattern analogue matching. The main method is Running-mean Z500 analogues (bw=14d baseline window, sw=28d event-centered search window) with absolute temperature; several supplemental figures also check sensitivity to alternative Z500-field variants (raw, JJA-mean-removed, normalized/standardized) and temperature representations (bias-corrected, calendar-day anomaly). There is a companion manuscript that is the authoritative source for scientific methodology/rationale/results - this repo's `README.md` and docstrings are meant to cover only what's needed to run the code (see the "docs-hygiene" section below for the precise carve-out).

12 Python scripts + their Slurm/shell wrappers form a documented from-scratch reproduction pipeline (`README.md`'s "Reproducing from scratch" section, steps 1-5) even though they are not directly imported by the 10 plotting scripts - e.g. `annual\_analogue\_analysis.py`, `bias\_correct\_predictions.py`, `extract\_raw\_z500\_regions.py`, `compute\_jja\_mean\_anomaly.py`, `compute\_running\_mean\_anomaly.py`, `compare\_raw\_vs\_normalized\_analogues.py`, `compare\_runningmean\_vs\_normalized\_analogues.py`, `anomaly\_convert\_predictions.py`, `compute\_fixed\_period\_comparison.py`, `find\_z500\_analogues.py` (imported), `extract\_temperature\_comparison.py` (imported). ****These are not dead code**** - "not imported by the plotting scripts" is not the same as "unreachable" in this repo, since the README's reproduction steps invoke them directly.

Extensive review history - please read this before flagging anything, to avoid re-flagging settled items

This repo has been through many rounds of Codex adversarial review already (2026-08-12 x4 rounds, 2026-08-28 x4 rounds, 2026-09-11, 2026-09-12 x5 rounds, 2026-09-13 x9 rounds, 2026-09-14 x5 rounds, 2026-09-15 x6 rounds so far). Real bugs found and fixed across these rounds include (non-exhaustive,

but covers the major categories so you don't need to re-derive them): a leap-year ``.dayofyear`` misalignment bug in the climatology/analogue-day-matching code (fixed via a custom ``.noleap_dayofyear()`` used everywhere calendar-day comparison matters); an invalid sign-flip-frequency bootstrap p-value replaced with a proper permutation test, later replaced again with a `**.block-bootstrap**` (block length from the significant-autocorrelation length of the OLS residuals, implemented once in ``.scripts/bootstrap_utils.py`` and shared by all 3 regression call sites: ``.plot_figure2.py``, ``.plot_region_detail_grid.py``, ``.plot_observed_diff_vs_gmt.py``); a median-of-differences vs. difference-of-medians bug in event-vs-analogue temperature-difference calculations; a ``.status`` column bug where a success row could silently fail a ``.status=='success'`` filter when the source CSV lacked that column and any other row failed; missing day-count/date-set/finite-value validation in several data loaders (now standardized via ``.validate_year_coverage()`` in ``.plot_analogue_statistics.py``, used by every figure loader that filters to a year range, and per-loader day-count/date-set/finite checks); GMT-difference computation evolved from a start-date-only lookup, to a 50/50 start+end average, to (current, final) a `**.true daily-weighted mean**` across every day in the period (``.plot_analogue_statistics.get_gmt_for_period()`` - ``.pd.date_range`` + per-day ``.get_gmt_for_date()`` lookup + ``.np.mean()``); a real visible copy-paste legend mislabel in Figure 1 panel D (``.plot_figureS6.py::draw_combined_panel()``); a fully-removed "random-period baseline" analysis path (was a relic of an earlier approach never used by any of the 10 figures - ``.random_analogue_analysis.py`` and all ``.--event-type`/`.event_type`` branches deleted, along with 39 tracked result-data files); a "fail loud rather than silently pool fewer regions/rows/years than documented" pattern rolled out to every figure loader; and a substantial dead-code trim of 6 mixed-purpose files that had unreachable standalone ``.main()`/`.create_plot()`/`.create_grid()`` functions left over from when they were run directly (2 were merged into their sole caller and deleted; 4 were trimmed in place to just their still-imported functions).

`**.Additional fixes from the most recent rounds (please don't re-flag these):**` ``.bootstrap_utils.py::block_bootstrap_regression()`` gained upfront length/finiteness validation on x/y (previously accepted non-finite input silently); ``.compute_running_mean_anomaly.py`/`.compute_jja_mean_anomaly.py`/`.bias_correct_predictions.py`` all gained a finiteness check on their baseline/reference/prediction values (previously only validated date coverage, not the values themselves), and all their "unique day" checks now normalize timestamps to calendar dates before deduping (previously deduped on raw timestamps, so two sub-daily timestamps on the same day could mask a missing day) - this same calendar-date-normalization fix was also applied to ``.plot_figureS6.py::member_vec()``, ``.plot_figure1.py``'s ``.load_z500_data()`/`.load_temp_stats_and_ensemble()`/`.load_absolute_tmean_map()``, ``.annual_analogue_analysis.py::validate_period_coverage()``, and ``.extract_temperature_comparison.py`` - `**.all "unique day"/timestamp-deduplication checks in this repository now normalize to calendar dates first; this entire category is closed, do not flag any instance of it again**`. ``.plot_figure1.py::load_event_row()`` now raises if multiple rows match the requested year (previously silently took the first). ``.plot_figureS1.py::load_regional_data()`/`.create_plot()``'s Python-API defaults now match their CLI default (``.representation='absolute'``, previously mismatched at ``.anomaly``). ``.plot_figure1.py``'s panels A-D and E-G now cross-validate they describe the same analogue-selection dates (mirroring ``.figure3_panel_data.py``'s pre-existing cross-validation), and this same cross-validation was extended to ``.plot_figureS2.py`` (all 12 grid cells) and

``plot_region_detail_grid.py`/`plot_figureS3.py`` (both temperature representations). ``annual_analogue_analysis.py`` now refuses to silently overwrite an existing summary file with a different `--n-days`` than what generated it, and ``plot_figure1.py``'s subtitle derives its displayed event duration and analogue-pool years dynamically (from the data and from ``compare_runningmean_vs_normalized_analogues.ANALOGUE_START`/`ANALOGUE_END`` respectively) instead of hardcoding `"3-day"/"1979-1999"`. Several sibling loaders (``plot_figureS1.py``, ``plot_predicted_vs_observed_absolute.py``, ``plot_observed_diff_vs_gmt.py``, ``plot_region_detail_grid.py::draw_panel()``) gained both ``dropna()`` AND an ``np.isfinite()`` mask on every column they render (previously some had only one or the other, or checked only a subset of the rendered columns). ``bias_correct_predictions.py`` and ``anomaly_convert_predictions.py`` now reject an empty `--output-suffix`` at argument-parsing time - previously an empty suffix would make the output path identical to the source prediction file, risking an actual overwrite of the original non-regenerable bundled input. ``plot_figureS1.py::create_pooled_panel()`` and ``plot_figureS2.py::draw_cell()`` gained the same `n<2/zero-variance`` guard before ``np.corrcoef()`` that ``figure3_region_overlay.py`` already had. ``plot_analogue_statistics.py::load_gmt_data()``'s within-month consistency check now explicitly verifies ``df['gmt']`` is finite first, since ``groupby(...).nunique()`/`.`first()`` both silently drop NaN by default. ``plot_analogue_statistics.py::validate_year_coverage()`` now rejects non-integral years explicitly instead of silently truncating them via ``int(y)``. ``annual_analogue_analysis.py::find_hottest_period_in_year()``'s year-filtered DataFrame now gets ``.reset_index(drop=True)`` before its positional-arithmetic-via-``.loc`` logic, so it's correct regardless of the caller's index structure rather than relying on an undocumented assumption about it. ``anomaly_convert_predictions.py`` gained the same JJA-date-coverage/finiteness validation its sibling ``bias_correct_predictions.py`` already has. ``compare_runningmean_vs_normalized_analogues.py`` gained the same z500-record-coverage check ``annual_analogue_analysis.py`` already has (a custom `--raw-z500-file`` narrower than the requested analogue-search range previously silently narrowed the candidate pool with no indication). All stale ``2025-12-31`` date references (the Z500 data's real end date is ``2024-10-31``) have been corrected across ``annual_analogue_analysis.py``'s docstring/epilog examples, ``run_annual_analogue_analysis.sh``'s usage comment, and ``run_annual_analogue_analysis.slurm``'s default-value comment/default. ``bias_correct_predictions.py`` now checks ``predicted_tmean`` for finiteness before quantile-mapping (previously only the ERA5 reference side was checked), matching the equivalent check already in ``anomaly_convert_predictions.py``. ``plot_analogue_statistics.py::validate_year_coverage()`` previously skipped all range/gap checking whenever either ``start_year`` or ``end_year`` was ``None`` (e.g. a one-sided call with ``start_year=2000, end_year=None`` didn't verify the years actually started at 2000); it now derives any missing bound from the observed years' own min/max and still requires a gapless run - this closes the gap for both one-sided and all-years-mode calls. ``figure3_region_overlay.py``'s docstring's median-nonlinearity identity (previously stated as the degenerate ``median(pred_diff) != median(pred_diff) - median(cf_diff)``) is now correctly ``median(pred_diff - cf_diff) != median(pred_diff) - median(cf_diff)`` - documentation-only, the code was already correct. ``plot_region_detail_grid.py::draw_panel()`` (S3-S5's shared panel renderer) gained the same zero-variance guard before ``np.corrcoef()`` that ``plot_figureS1.py`/`plot_figureS2.py`/`figure3_region_overlay.py`` already had - previously a constant predictor or response would render ``r=nan``. ``plot_figureS2.py``'s title previously fell back to "All Years"

whenever either CLI bound was omitted, mislabeling legitimate one-sided selections (e.g. `--start-year` alone); it now derives the displayed year range from the actual filtered data, matching `plot_figureS1.py`'s convention. `annual_analogue_analysis.py::find_hottest_period_in_year()` previously computed its rolling-mean/period selection via positional arithmetic that assumed adjacent rows are adjacent calendar days, relying entirely on an undocumented assumption that its caller pre-sorts by time - it now sorts its own input by `time` before proceeding, so it's correct regardless of caller behavior (its sole current caller already sorted first, so this was inert in practice).

`extract_temperature_comparison.py::create_comparison_dataframe()` previously merged observed/predicted values on exact timestamps while the coverage validation upstream normalizes both sides to calendar dates before comparing them - a mismatched intraday timestamp convention (e.g. obs at noon, predictions at midnight) could pass that validation yet leave every `predicted_*` column NaN for that day after the merge; it now merges on a `calendar-date-normalized` key instead. `extract_raw_z500_regions.py`'s `--end-month` previously accepted any integer (a nonpositive value silently skipped all of `end_year` with no gap for the existing checks to catch); it's now validated to 1-12 at argument-parsing time. `annual_analogue_analysis.py`'s `summary-overwrite` guard previously only compared `--n-days` against an existing summary file, missing the analogue search pool (start/end), season bounds, exclusion buffer, and which `--pred-file` was used - none of which the CSV itself records. A rerun changing any of these but keeping the same year range could silently overwrite the canonical summary with an incompatible analysis (e.g. a bias-corrected `--pred-file` run without `--output-dir` overwriting the absolute-temperature summary, since dates stay identical). Fixed by persisting all 7 result-defining parameters in a JSON sidecar (`<summary>.meta.json`) next to the CSV, checked on every subsequent run; a pre-existing summary with no sidecar yet falls back to the original `n_days-only` check, then gets one written going forward.

`find_z500_analogues.py`/`annual_analogue_analysis.py`'s `--exclude-buffer` and `compare_runningmean_vs_normalized_analogues.py`'s `--search-window-days` previously accepted negative integers (a negative exclude-buffer could flip `exclude_start` past `exclude_end`, disabling the exclusion entirely and letting the target period become its own "analogue" with correlation 1.0; a negative search-window-days rejected every candidate and wrote an empty results CSV to the real output path before exiting nonzero) - all 3 now reject negative values at argument-parsing time, matching the existing `compute_running_mean_anomaly.py::--window-days` precedent. `scripts/logs/` (used by 3 Slurm wrappers' `--output`/`--error`) is now tracked via `scripts/logs/.gitkeep` - it was previously an empty, untracked directory that a fresh checkout would be missing entirely.

`compare_runningmean_vs_normalized_analogues.py`'s docstring previously claimed an out-of-JJA analogue "silently produces NaN downstream" - corrected, since `_validate_period_coverage()` and the round-17 calendar-date-merge fix both make this raise on a date-coverage mismatch instead. `compare_raw_vs_normalized_analogues.py`/`compare_runningmean_vs_normalized_analogues.py` previously never wrote `n_days` into their output rows, so `plot_figureS1.py`'s cross-region event-duration consistency check always compared 3 hardcoded fallback values instead of the actual event durations for this data source - both now include `n_days` (already computed internally), which propagates through `compute_fixed_period_comparison.py`'s full-row copy with no other changes needed. `annual_analogue_analysis.py`'s `overwrite-guard` metadata sidecar (round 18) omitted `region` - two runs sharing a custom `--output-dir`/`year-range`/`--pred-file` but different `--region` could silently overwrite each other undetected; `region` is now

included. ``extract_temperature_comparison.py``'s ``load_observed_temps()``/``load_predicted_temps()``, ``plot_figure1.py``'s ``load_temp_stats_and_ensemble()``, and ``plot_figureS6.py``'s ``member_vec()`` all previously selected data using raw, non-normalized timestamps - a non-midnight timestamp on the final day of a period would fail the raw upper-bound comparison and be silently excluded from selection itself, before any calendar-date normalization logic downstream even ran; all 4 now select on calendar-date-normalized bounds instead.

****Deliberately NOT bugs - please do not re-flag these:****

- Methodology choices inherited from the published companion paper (Callahan et al. 2026, *Nature Climate Change*) that a reviewer might otherwise flag as arbitrary - e.g. the exact Z500 geographic box per region, the 1981-2010 baseline period, analogue selection by correlation-then-RMSE-tiebreak, the +/-28-day event-centered search window. These are inherited design choices, not code bugs.

- ``get_gmt_for_date()``/``get_gmt_for_period()`` look up GMT by year+month against ``data/ERA5/ML_inputs/{region}/gmt_ERA5.csv``, whose ``gmt`` column is itself a ****trailing 12-month mean**** (from the Callahan et al. replication materials) - so a `"ΔGMT_annual"` label refers to that 12-month window, not "one value per calendar year." Confirmed by direct data inspection (identical across all 3 regions' copies to ~1e-13 float noise) - not a bug.

- ``find_z500_analogues.py``'s ``calculate_pattern_correlation()``/``calculate_rmse()`` require full finite (non-NaN) coverage from both compared maps (returns NaN otherwise, handled downstream) - a deliberate choice (an earlier per-pair-NaN-masking design was intentionally replaced with this), and moot in practice since the real bundled Running-mean Z500 data has zero NaN cells across all 3 regions.
- A land-only absolute-temperature map with permanently-NaN ocean cells (``plot_figure1.py::load_absolute_tmean_map()``) is expected - do not flag missing/NaN cells there as a data-quality bug.

- Figures rendering a "No data" placeholder used to be a mixed pattern but has since been made uniformly fail-loud (raise/exit 1) across all 10 figures for missing/incomplete region-year data - if you see a loader that still silently tolerates a smaller-than-requested N without raising, that IS worth flagging (it would be a regression from the now-established pattern), but the ***general design choice*** to fail loud (vs. e.g. render a placeholder) is settled, not up for debate.

- A known, still-open, low-priority gap: the block-bootstrap permutation test's p-value degenerates to ``p=1.000`` for a constant predictor (zero-variance x) - real gap, but inert on the actual 25-year GMT/correlation records in this repo (never actually constant). You may mention it if you rediscover it, but it's already tracked and not a new finding.

- The 12 from-scratch producer scripts/wrappers listed above being unreachable from the 10 plotting scripts' import graph is expected, not dead code (see above).

- ``plot_region_detail_grid.py::load_panel_data()`` still supports 4 `row_labels` (Raw/JJA-mean-removed/Running-mean/Normalized), a GMT-lookup branch, and a 4th metric column, but its only current caller (``plot_figureS3.py``, transitively also S4/S5 via ``create_grid()``) always fixes ``row_label='Running-mean'``, always passes ``include_gmt=False``, and only plots the first 3 metrics - so those other branches are unreachable. This was explicitly raised to the user and they chose to leave it as-is (mirrors the original pre-dead-code-trim grid's generality rather than being orphaned dead code) - do not re-flag it.

- ****Confirmed false positive (verified via simulation, not just re-derivation)****: ``bootstrap_utils.py::_block_permutation()``'s final block is

shorter than `block\_length` when `n` isn't evenly divisible (e.g. 25 years at block_length=2 gives twelve 2-year blocks + one 1-year block), which was flagged as producing an invalid permutation null and reportedly changing Figure 2/S5's p-values under an alternative "keep the singleton fixed" scheme. A null-hypothesis Monte Carlo calibration check (2000 synthetic autocorrelated-but-x-independent datasets) showed the current implementation's p-values are correctly uniformly distributed under the null (~5% rejection at alpha=0.05, mean p~0.5) - and that the proposed alternative scheme is **equally** well-calibrated, no better. Both are valid resampling schemes; the real-data p-value shift when switching between them is expected Monte Carlo variation between two legitimate schemes, not evidence of bias in the current one. Do not re-flag this without a new calibration argument, not just a different point-estimate diff.

****Docs-hygiene carve-out**** (a full pass for this already happened on 2026-09-13; only flag genuinely new instances, and respect this distinction): README.md and docstrings should describe only how to run the current code (usage, CLI flags, file paths, run-order dependencies, data provenance/citations) - not scientific methodology/rationale, not results narrative, not historical/changelog narrative about prior versions of this code. However, inline comments AND docstrings that explain a non-obvious ****implementation-correctness invariant**** needed to safely maintain this specific code (e.g. why a bootstrap resampling function returns what it returns, why a leap-year day-of-year mapping is needed, why a specific scalar must be subtracted in a specific order because median isn't linear) are a ****different, legitimate category**** and should NOT be flagged just for being detailed or explaining "why" - only flag them if they contain actual scientific-methodology justification (why this analysis approach is scientifically correct/chosen) or historical narrative, not implementation mechanics.

Your task - focus on exactly these 3 dimensions

1. ****Accuracy and consistency of calculations and statistical tests.**** Check every place a statistic is computed (correlation, RMSE, bootstrap CI/p-value, regression slope, GMT difference, day-of-year difference, quantile mapping, bias correction) for correctness and for consistency with its sibling implementations elsewhere in the codebase (e.g. do all 3 `block\_bootstrap\_regression()` call sites use it identically; do all loaders that filter to a year range apply `validate\_year\_coverage()`; does every stats box report the same definition of r/p/N that its sibling figures do; are there any remaining copy-pasted statistical implementations that could silently diverge). Pay particular attention to edge cases (small N, ties, NaN handling, degenerate/constant inputs) and to whether a recently-changed shared function (e.g. `get\_gmt\_for\_period()`, `validate\_year\_coverage()`, `bootstrap\_utils.py`) is actually applied everywhere it should be, not just at some call sites.

2. ****Code bugs.**** General correctness issues: off-by-one errors, incorrect NaN/finite handling, silent failures, exceptions that would crash on realistic (not just current bundled) input, resource leaks, exit-code/status-column consistency, import errors, unreachable or dead code introduced by a recent change, inconsistent parameter defaults between duplicated code paths.

3. ****Consistency of docstrings and README.**** Beyond the docs-hygiene scope question above: does every docstring/README claim about current behavior (CLI flags, file paths, importers, outputs, script structure, what a function

returns, which figures use what) actually match the current code? Flag anything stale/inaccurate, but respect the "deliberately not bugs" and "docs-hygiene carve-out" sections above.

For each finding, give: file:line, a concrete description with a failure scenario where applicable, which of the 3 categories it falls into, severity (High/Medium/Low), and **whether it currently affects the actual delivered results** - i.e. does it change any of the 10 bundled figures or any currently-committed data/results file as things stand today, or is it purely hypothetical (a real gap in the code that would only matter for different/future/malformed input, but is inert on the real bundled data right now)? Say so explicitly for each finding, don't leave it implied.

After the findings, include a **verification summary**: what you actually checked and confirmed clean (e.g. specific recomputations that matched, specific consistency checks that passed, specific files/call sites you traced and found correct) - not just "no other issues found," but the concrete things you verified. This review is read-only - do not edit files.

2. Text of *codex_review_2026-09-15_comprehensive_findings.md*:

Findings

1.

[scripts/plot_figure1.py:61] (/oak/stanford/groups/omramom/group_members/omramom/attribution_verification_release/scripts/plot_figure1.py:61) and [scripts/plot_figure1.py:144] (/oak/stanford/groups/omramom/group_members/omramom/attribution_verification_release/scripts/plot_figure1.py:144) - ``load_z500_data()`` and ``load_absolute_tmean_map()`` still select periods with raw timestamp slices. Their bounds are midnight ``Timestamp`` objects, so a complete dataset timestamped later in each day—such as daily values at noon—excludes the final calendar day. The subsequent day-count validation then raises even though every requested calendar date is present. This is inconsistent with the calendar-date-normalized selection recently applied to the temperature and ensemble loaders.

Category: Code bug

Severity: Low

Delivered-results impact: None. The bundled Z500 and temperature-map inputs use compatible midnight timestamps, so all ten committed figures are unaffected. This only affects replacement/future inputs using a different intraday convention, and it fails loudly rather than silently changing results.

No new accuracy/statistical-test defect or README/docstring inconsistency was found.

Verification summary

- Independently recomputed all numerical temperature statistics in the 36 committed summary CSVs—9 annual-analysis summaries and 27 Z500-variant/temperature-representation summaries, covering 900 region-year rows. Observed means, per-member period means, ensemble min/median/max/std, and paired event-minus-analogue differences matched within a maximum absolute error of ``3.55e-15``.

- Recomputed 600 stored Z500 statistics across the normalized, raw, JJA-mean-removed, and Running-mean selections. Every selected-period pattern correlation and RMSE matched; maximum absolute error was ``2.22e-16``.
- Re-ran the complete normalized analogue ranking for the 2023 Central Europe example. The best analogue remained 1994-07-01, matching the committed result.
- Verified all three ``*_ANOM.nc`` files exactly equal ``predicted_tmean - seasonal_cycle``; maximum error was zero. All copied-through variables were byte/value identical.
- Recomputed every member of all three quantile-mapped prediction files using ``quantile_map_member()``. The results matched exactly, and copied-through variables were unchanged.
- Exercised the delivered-data loaders and cross-source validation for Figures S1, S2, S3-S5, 3, S6, and S7. Every region/cell contained exactly 2000-2024 once each, all rendered values were finite, and all event/analogue date cross-checks passed.
- Traced all three regression call sites. Each uses the same ``block_bootstrap_regression()``, orders data through ``year_sort_order()``, reports the permutation-test p-value, and uses the shared bootstrap-median line/slope convention.
- Confirmed all GMT period differences go through the daily-weighted ``get_gmt_for_period()`` implementation, and all calendar-day comparisons use the no-leap mapping.
- Checked README paths, figure descriptions, run order, CLI examples, wrapper arguments, generated filenames, and importer relationships against the current code and repository contents.
- No repository files were edited. The pre-existing modification to ``codex_review_2026-09-15_comprehensive_prompt.md`` remained untouched.

3. Text of *round21_fix_summary_2026-09-15.md*:

All 10 figures byte-identical.

****Round 21 (round 8 of 2026-09-15) fix complete and verified:****

- ****`plot\_figure1.py::load\_z500\_data()` and `load\_absolute\_tmean\_map()`**** - both now select on calendar-date-normalized bounds instead of raw timestamps, matching round 20's fix to the sibling function in the same file. Unit-tested ``load_z500_data()`` against a synthetic noon-timestamped repro case (now correctly includes all 3 days). ``load_absolute_tmean_map()`` confirmed working directly against real data (Central Europe 2023).

All 10 delivered figures regenerated and byte-identical.

4. Text of *from_scratch_rebuild_verification_2026-09-15_round2.md*:

Full from-scratch rebuild complete and verified (2nd rebuild of the day, run after round 21). Summary:

****Rebuild process:**** Cloned the tracked repo (``8ede93f``) to ``$SCRATCH``, staged only the true raw/supplied inputs (1142 files: raw ERA5 monthlies, per-region GMT/temperature/normalized-Z500 files, raw CNN predictions, shapefile), cleared all tracked ``results/``/``figures/``, then ran the full 5-step README pipeline end-to-end (60+ script invocations, zero failures).

****Verification results:****

- ****All 10 delivered figures****: byte-identical.
- ****Result CSVs****: 9 of 45 identical outright; the other 36 (all under ``results/z500_{raw,jja_anomaly,runningmean}_vs_normalized/``) came back "different." Investigated programmatically (pandas column-diff, not just byte comparison): every one of the 36 differed **only** by a new ``n_days`` column, with every other value byte-identical.
- ****Derived NetCDF data****: all identical except the 3 regional ``z500_cday_raw_ERA5.nc`` files, confirmed via xarray to be the same incidental encoding-noise pattern seen in the first rebuild (data values and time coordinates exactly identical).

****Root cause of the 36 CSV differences:**** round 20 (earlier the same day) added an ``n_days`` column to ``compare_raw_vs_normalized_analogues.py``'s and ``compare_runningmean_vs_normalized_analogues.py``'s output rows, and verified the fix by smoke-testing to ``/tmp`` and confirming all 10 figures stayed byte-identical (correctly - the only consumer's fallback already produced the same effective value). But the real tracked ``results/`` files were never regenerated into their committed paths, so they were stale relative to what current code would actually produce. This from-scratch rebuild is what surfaced it.

****Fix applied:**** regenerated all 36 affected CSVs directly into their tracked paths (re-running the same producer scripts against the real data), then re-verified programmatically that the only change versus the previously committed content is the new ``n_days`` column (uniformly ``3``) - every other value is byte-identical. Re-ran all 10 figures afterward and confirmed they remain byte-identical. Committed as ``87a4e9a``.

The tracked repo is now fully in sync with current code - no results are affected by any of the 21 review rounds to date, and the tracked ``results/`` directory now reflects exactly what current code produces.