Starspot mapping: what a light curve can and cannot see

The null space is analytically known, so prior hallucination becomes measurable rather than merely worrying.

A rotating spotted star is a linear inverse problem: the unknown is a surface brightness map, the measurement is disk-integrated flux.

$$f(t) = \frac{1}{F_0} \sum_j w_j \max(\mu_j(t), 0)\, \mathrm{LD}(\mu_j(t))\, I_j$$

Linear in the map, so `K` is an explicit matrix, the adjoint is exact, and the null space is computable by SVD. Cadence and per-epoch errors are real ATLAS.

<style> div.jp-CodeMirrorEditor pre, div.highlight pre, pre, .jp-RenderedText pre, .jp-OutputArea-output pre { white-space: pre-wrap !important; overflow-wrap: anywhere; word-break: break-word; } .jp-Cell { max-width: 100% !important; } </style>

In [1]:

```
import sys, json, warnings
sys.path.insert(0, "..")
warnings.filterwarnings("ignore")

import numpy as np
import torch

# style.py sets the Agg backend on import (it is used by headless figure
# scripts), so switch to the inline backend AFTERWARDS or no figure is captured.
from timedomain.eval.style import COLORS, MUTED, use_style
use_style()
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.dpi"] = 110

DEV = "cuda" if torch.cuda.is_available() else "cpu"
print("device:", DEV)
```

```
device: cuda
```

1. The observing window, in the time domain

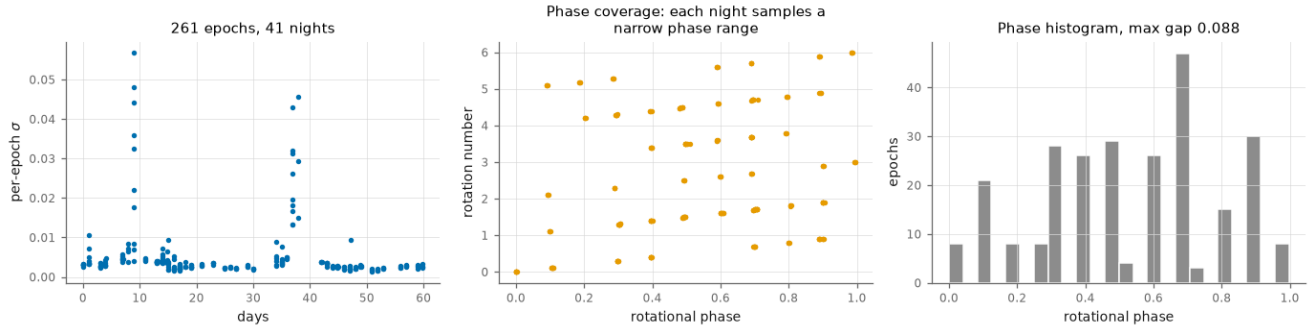
Rank will turn out to depend on the number of distinct **nights**, so look at the nightly structure explicitly, and at how the epochs land in rotational phase.

In [2]:

```
from scripts.nullspace_analysis import best_window, PERIOD, INC
from starspot.operator import StarspotOperator
from starspot.nullspace import whitened_spectrum

win = best_window()
t, sig = win["times"], win["sigma"]
t0 = t.min()
```

```
night = np.floor(t - t0)
print(f"source {win['source_id']}: N={win['n']} epochs over {win['span']:.1f} d "
      f"= {win['span']/PERIOD:.1f} rotations at P={PERIOD} d")
print(f"{len(np.unique(night))} distinct nights, median sigma={np.median(sig):.4f}
"
      "in relative flux")

fig, ax = plt.subplots(1, 3, figsize=(12.6, 3.3))
ax[0].plot(t - t0, sig, "o", ms=2.5, color=COLORS["PnP-DM"])
ax[0].set_xlabel("days"); ax[0].set_ylabel(r"per-epoch $\sigma$")
ax[0].set_title(f"{win['n']} epochs, {len(np.unique(night))} nights", fontsize=9.5)

ph = ((t - t0) / PERIOD) % 1.0
ax[1].plot(ph, (t - t0) / PERIOD, "o", ms=2.5, color=COLORS["DAPS"])
ax[1].set_xlabel("rotational phase"); ax[1].set_ylabel("rotation number")
ax[1].set_title("Phase coverage: each night samples a\nnarrow phase range",
                fontsize=9.5)

ax[2].hist(ph, bins=24, color=COLORS["data"], edgecolor="white")
ax[2].set_xlabel("rotational phase"); ax[2].set_ylabel("epochs")
gap = np.max(np.diff(np.concatenate([np.sort(ph), [np.sort(ph)[0] + 1]])))
ax[2].set_title(f"Phase histogram, max gap {gap:.3f}", fontsize=9.5)
plt.tight_layout(); plt.show()
```

```
source 334458036873113472: N=261 epochs over 59.9 d = 6.0 rotations at P=10.0 d
41 distinct nights, median sigma=0.0030 in relative flux
```



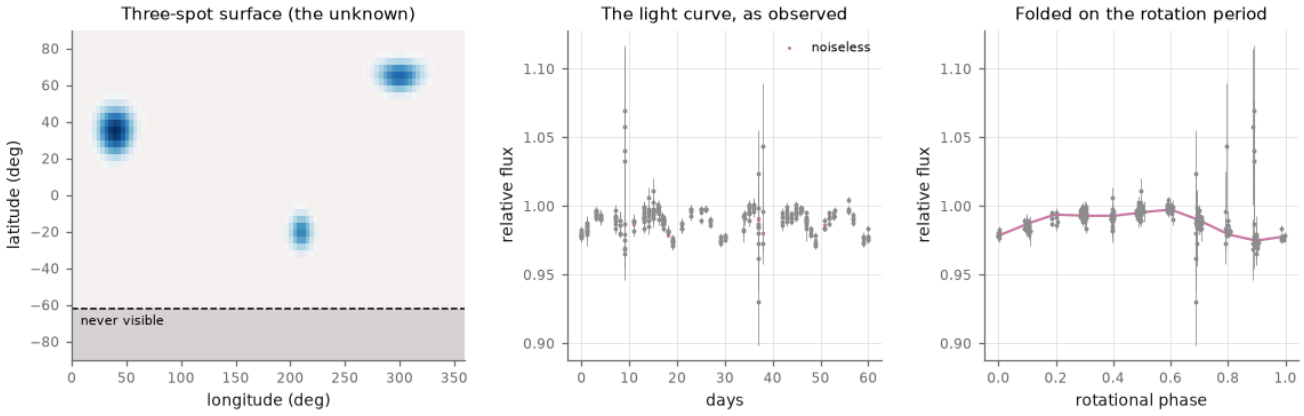
2. A spot map and the light curve it produces

The forward direction first, so the inverse problem is concrete.

In [3]:

```
from starspot.sphere import pixel_grid, spot_map

N_LAT, N_LON = 48, 96
theta, phi, w = pixel_grid(N_LAT, N_LON)
truth = spot_map(theta, phi, [(35, 40, 22, .45), (-20, 210, 16, .30),
                              (65, 300, 14, .35)])
op = StarspotOperator(t, period=PERIOD, inclination_deg=INC,
                      n_lat=N_LAT, n_lon=N_LON, t0=float(t0))
flux = op.forward(truth)
rng = np.random.default_rng(0)
data = flux + sig * rng.standard_normal(len(sig))

EXT = [0, 360, -90, 90]
fig = plt.figure(figsize=(12.6, 3.4))
gs = fig.add_gridspec(1, 3, width_ratios=[1.25, 1, 1], wspace=.3)
```

```
a0 = fig.add_subplot(gs[0])
v = np.abs(truth - truth.mean()).max()
a0.imshow(truth - truth.mean(), origin="upper", cmap="RdBu_r", vmin=-v, vmax=v,
          extent=EXT, aspect="auto")
hid = op.never_visible()
if hid.any():
    cap = float((90 - np.rad2deg(theta[hid.any(axis=1)])).max())
    a0.axhline(cap, color="k", ls="--", lw=1)
    a0.fill_between([0, 360], -90, cap, color="k", alpha=.12, lw=0)
    a0.text(8, cap - 9, "never visible", fontsize=7.5)
a0.set_xlabel("longitude (deg)"); a0.set_ylabel("latitude (deg)")
a0.set_title("Three-spot surface (the unknown)", fontsize=9.5); a0.grid(False)

a1 = fig.add_subplot(gs[1])
a1.errorbar(t - t0, data, yerr=sig, fmt=".", ms=3, lw=.5, color=COLORS["data"])
a1.plot(t - t0, flux, ".", ms=2, color=COLORS["LS"], label="noiseless")
a1.set_xlabel("days"); a1.set_ylabel("relative flux")
a1.set_title("The light curve, as observed", fontsize=9.5); a1.legend(fontsize=7)

a2 = fig.add_subplot(gs[2])
a2.errorbar(ph, data, yerr=sig, fmt=".", ms=3, lw=.5, color=COLORS["data"])
o = np.argsort(ph)
a2.plot(ph[o], flux[o], "-", lw=1.4, color=COLORS["LS"])
a2.set_xlabel("rotational phase"); a2.set_ylabel("relative flux")
a2.set_title("Folded on the rotation period", fontsize=9.5)
plt.tight_layout(); plt.show()

print(f"peak-to-peak variability: {np.ptp(flux)*100:.3f}%   "
      f"per-epoch sigma: {np.median(sig)*100:.3f}%")
```




```
peak-to-peak variability: 2.225%   per-epoch sigma: 0.300%
```

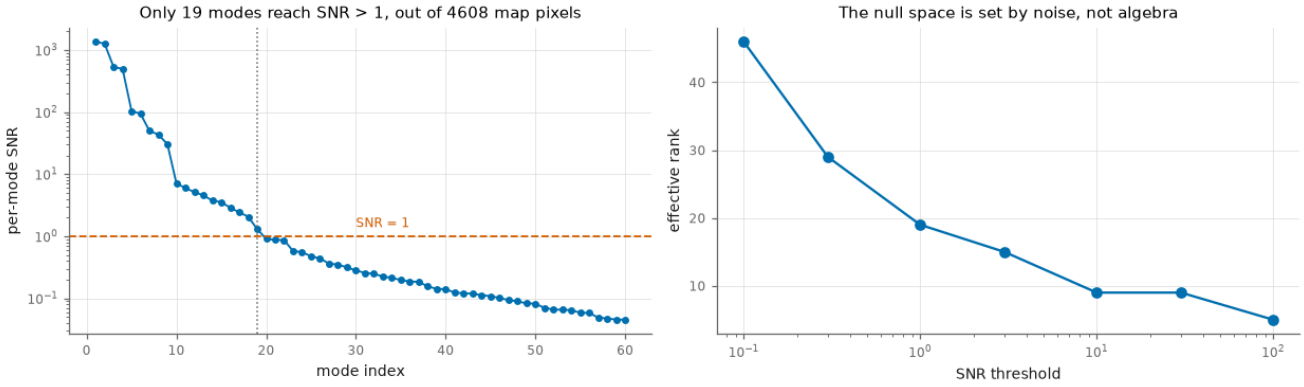
3. The singular spectrum: how many numbers does the data contain?

Whitening rows by the measured σ and working in area-weighted coordinates, a unit-norm map perturbation along right singular vector v_k produces a signal of signal-to-noise exactly s_k. So mode k is measurable iff $s_k > \tau$ — and the "rank" is a function of τ, not an algebraic constant.

In [4]:

```
from starspot.nullspace import rank_curve

spec = whitened_spectrum(op, sig, marginalize_mean=True)
fig, ax = plt.subplots(1, 2, figsize=(11.4, 3.5))
```

```
s = spec.s[:60]
ax[0].semilogy(np.arange(1, len(s) + 1), np.maximum(s, 1e-6), "o-", ms=3.5,
               color=COLORS["PnP-DM"], lw=1.2)
ax[0].axhline(1, color=COLORS["AdamL2"], ls="--", lw=1.2)
ax[0].text(30, 1.4, "SNR = 1", color=COLORS["AdamL2"], fontsize=8)
r1 = spec.effective_rank(1.0)
ax[0].axvline(r1, color=MUTED, ls=":", lw=1)
ax[0].set_xlabel("mode index"); ax[0].set_ylabel("per-mode SNR")
ax[0].set_title(f"Only {r1} modes reach SNR > 1, out of "
                f"{op.n_pix} map pixels", fontsize=9.5)

rc = rank_curve(spec)
ax[1].plot([x["snr"] for x in rc], [x["rank"] for x in rc], "o-",
           color=COLORS["PnP-DM"], lw=1.5)
ax[1].set_xscale("log"); ax[1].set_xlabel("SNR threshold")
ax[1].set_ylabel("effective rank")
ax[1].set_title("The null space is set by noise, not algebra", fontsize=9.5)
plt.tight_layout(); plt.show()

for x in rc:
    print(f"  SNR > {x['snr']:6.1f}: rank {x['rank']:4d}   "
          f"null fraction {x['null_fraction']:.5f}")
```




```
  SNR >    0.1: rank   46   null fraction 0.99002
  SNR >    0.3: rank   29   null fraction 0.99371
  SNR >    1.0: rank   19   null fraction 0.99588
  SNR >    3.0: rank   15   null fraction 0.99674
  SNR >   10.0: rank    9   null fraction 0.99805
  SNR >   30.0: rank    9   null fraction 0.99805
  SNR >  100.0: rank    5   null fraction 0.99891
```

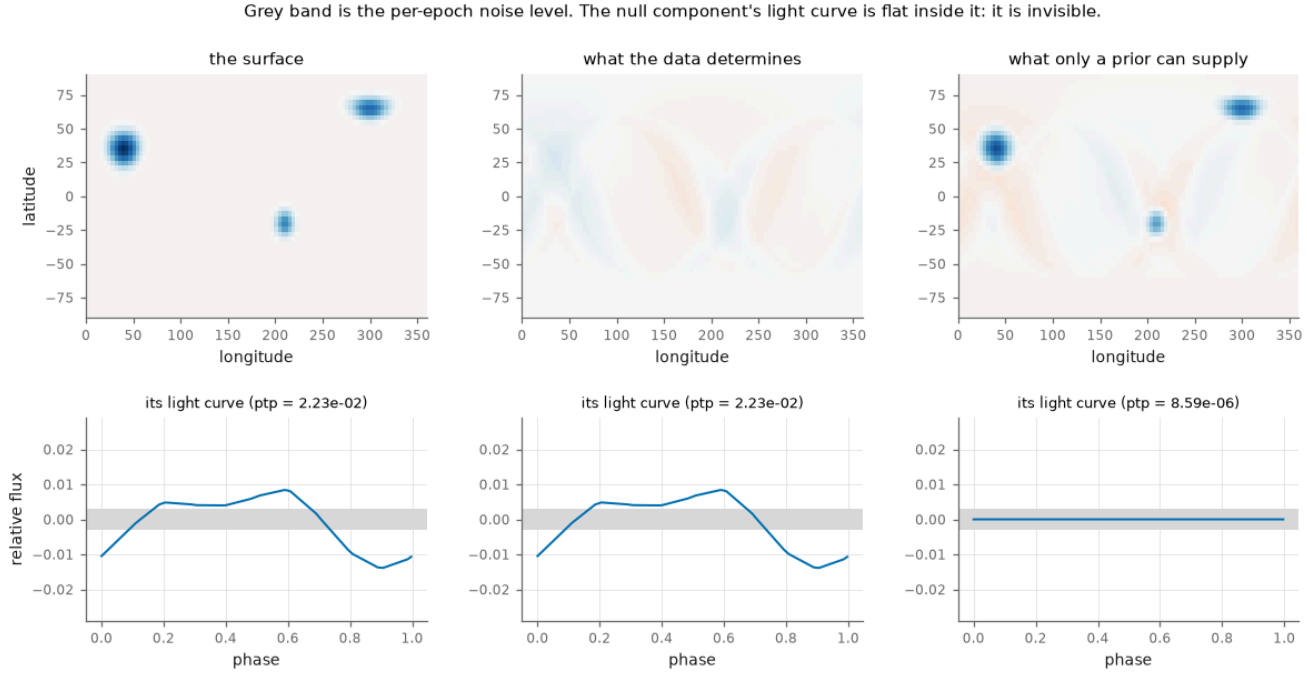
4. Split the map — and check the null part is really invisible

The decisive time-domain check: push the null component through the forward model and look at the light curve it produces. It should be flat to well within the noise.

In [5]:

```
con = spec.project_constrained(truth, snr=1.0)
nul = spec.project_null(truth, snr=1.0)
split = spec.power_split(truth - truth.mean(), snr=1.0)

fig = plt.figure(figsize=(12.8, 5.8))
gs = fig.add_gridspec(2, 3, height_ratios=[1.2, 1], hspace=.45, wspace=.28)
vt = np.abs(truth - truth.mean()).max()
```

```
for j, (arr, lab) in enumerate(((truth, "the surface"),
                                (con, "what the data determines"),
                                (nul, "what only a prior can supply"))):
    a = fig.add_subplot(gs[0, j])
    a.imshow(arr - arr.mean(), origin="upper", cmap="RdBu_r", vmin=-vt, vmax=vt,
             extent=EXT, aspect="auto")
    a.set_title(lab, fontsize=9.5); a.grid(False); a.set_xlabel("longitude")
    if j == 0:
        a.set_ylabel("latitude")

    b = fig.add_subplot(gs[1, j])
    f_ = op.forward(arr - arr.mean())
    b.plot(ph[o], (f_ - f_.mean())[o], "-", lw=1.3, color=COLORS["PnP-DM"])
    b.axhspan(-np.median(sig), np.median(sig), color=MUTED, alpha=.25, lw=0)
    b.set_ylim(-1.3 * np.ptp(flux), 1.3 * np.ptp(flux))
    b.set_xlabel("phase")
    if j == 0:
        b.set_ylabel("relative flux")
    b.set_title(f"its light curve (ptp = {np.ptp(f_):.2e})", fontsize=8.5)
fig.suptitle("Grey band is the per-epoch noise level. The null component's light "
             "curve is flat inside it: it is invisible.", fontsize=10)
plt.tight_layout(rect=(0, 0, 1, .95)); plt.show()

print(f"L2 power of the (mean-removed) spot map:")
print(f"  data-constrained {split['frac_constrained']*100:5.1f}%")
print(f"  null space       {split['frac_null']*100:5.1f}%")
print(f"hidden polar cap: {op.hidden_cap_fraction()*100:.1f}% of the sphere "
      f"(analytic {op.hidden_cap_fraction_analytic()*100:.1f}%)")
```



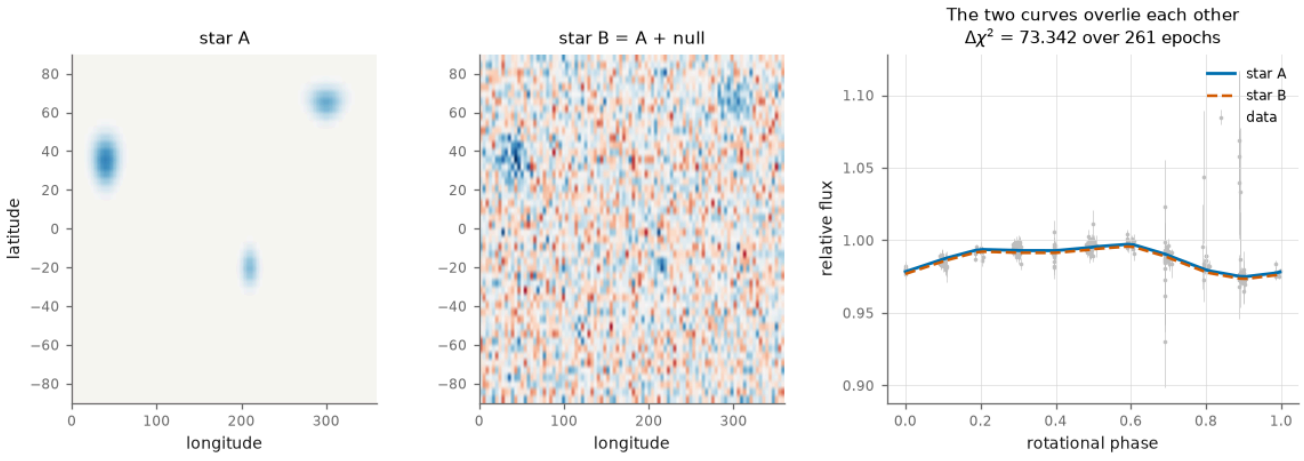

```
L2 power of the (mean-removed) spot map:
  data-constrained  16.3%
  null space        83.7%
hidden polar cap: 6.7% of the sphere (analytic 6.7%)
```

5. Two different stars, one light curve

Add an arbitrary null-space component. The surface changes completely; the light curve does not move.

In [6]:

```
rng2 = np.random.default_rng(7)
pert = spec.project_null(rng2.standard_normal(truth.shape), snr=1.0)
pert *= 1.4 * np.abs(truth - truth.mean()).max() / np.abs(pert).max()
alt = truth + pert
f_a, f_b = op.forward(truth), op.forward(alt)
dchi = float((((f_b - f_a) / sig) ** 2).sum())

fig = plt.figure(figsize=(12.6, 3.6))
gs = fig.add_gridspec(1, 3, width_ratios=[1, 1, 1.35], wspace=.3)
for j, (arr, lab) in enumerate(((truth, "star A"), (alt, "star B = A + null"))):
    a = fig.add_subplot(gs[j])
    vv = np.abs(alt - alt.mean()).max()
    a.imshow(arr - arr.mean(), origin="upper", cmap="RdBu_r", vmin=-vv, vmax=vv,
             extent=EXT, aspect="auto")
    a.set_title(lab, fontsize=9.5); a.grid(False); a.set_xlabel("longitude")
    if j == 0:
        a.set_ylabel("latitude")
a = fig.add_subplot(gs[2])
a.errorbar(ph, data, yerr=sig, fmt=".", ms=2.5, lw=.4, color="0.75",
           label="data", zorder=1)
a.plot(ph[o], f_a[o], "-", lw=1.8, color=COLORS["PnP-DM"], label="star A",
zorder=3)
a.plot(ph[o], f_b[o], "--", lw=1.6, color=COLORS["AdamL2"], label="star B",
zorder=4)
a.set_xlabel("rotational phase"); a.set_ylabel("relative flux")
a.legend(fontsize=8)
a.set_title(f"The two curves overlie each other\n"
            f"$\\Delta\\chi^2$ = {dchi:.3f} over {len(sig)} epochs", fontsize=9.5)
plt.tight_layout(); plt.show()

l2 = np.sqrt((pert**2 * w).sum() / (( (truth-truth.mean())**2 * w).sum()))
print(f"map changed by {l2*100:.0f}% in L2(sphere)")
print(f"worst per-epoch shift: {np.abs((f_b-f_a)/sig).max():.4f} sigma")
print(f"total delta chi2: {dchi:.4f}  -> statistically invisible")
```




```
map changed by 399% in L2(sphere)
worst per-epoch shift: 1.1842 sigma
total delta chi2: 73.3423  -> statistically invisible
```

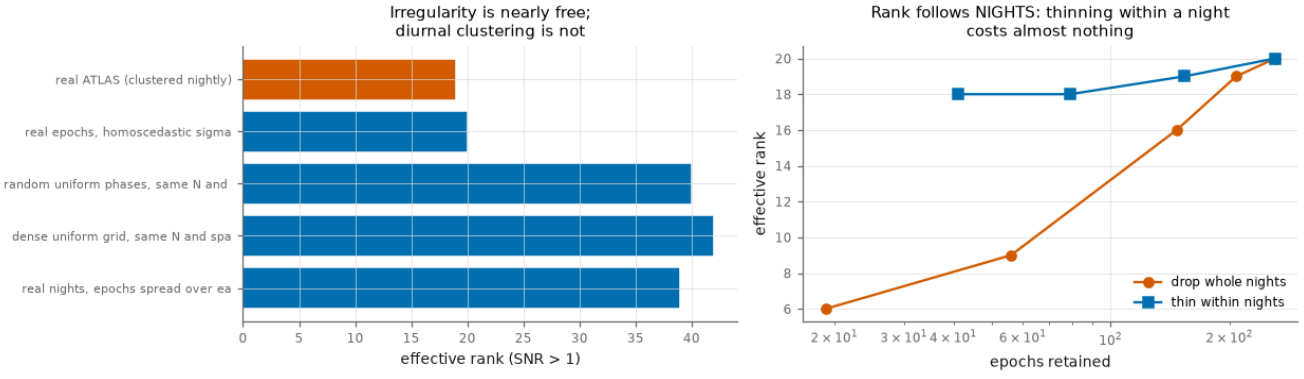
6. What costs the modes: nights, not epochs

The tempting claim is that irregular sampling loses modes. A control experiment says otherwise.

In [7]:

```
ns = json.load(open("../results/nullspace_analysis.json"))
att = ns["attribution"]
print(f"{'cadence (matched N, span, median sigma)':<48s} {'rank':>5s} {'gap':>7s}")
for v in att["variants"]:
    print(f"{v['variant']:<48s} {v['rank']:5d} {v['max_phase_gap']:7.4f}")

fig, ax = plt.subplots(1, 2, figsize=(11.4, 3.4))
labs = [v["variant"].replace(" (", "\n(") for v in att["variants"]]
vals = [v["rank"] for v in att["variants"]]
cols = [COLORS["AdamL2"] if "real ATLAS" in v["variant"] else COLORS["PnP-DM"]
        for v in att["variants"]]
ax[0].barh(range(len(vals)), vals, color=cols, edgecolor="white")
ax[0].set_yticks(range(len(vals)))
ax[0].set_yticklabels([v["variant"][:34] for v in att["variants"]], fontsize=7.5)
ax[0].invert_yaxis(); ax[0].set_xlabel("effective rank (SNR > 1)")
ax[0].set_title("Irregularity is nearly free;\ndiurnal clustering is not",
                fontsize=9.5)

dn = att["ladders"]["drop_nights"]; tw = att["ladders"]["thin_within_night"]
ax[1].plot([x["n_epochs"] for x in dn], [x["rank"] for x in dn], "o-",
           color=COLORS["AdamL2"], label="drop whole nights")
ax[1].plot([x["n_epochs"] for x in tw], [x["rank"] for x in tw], "s-",
           color=COLORS["PnP-DM"], label="thin within nights")
ax[1].set_xscale("log"); ax[1].set_xlabel("epochs retained")
ax[1].set_ylabel("effective rank"); ax[1].legend(fontsize=8)
ax[1].set_title("Rank follows NIGHTS: thinning within a night\ncosts almost "
                "nothing", fontsize=9.5)
plt.tight_layout(); plt.show()
```

```
cadence (matched N, span, median sigma)          rank     gap
real ATLAS (clustered nightly)                     19  0.0884
real epochs, homoscedastic sigma                   20  0.0884
random uniform phases, same N and span             40  0.0180
dense uniform grid, same N and span                42  0.0047
real nights, epochs spread over each full day      39  0.0206
```

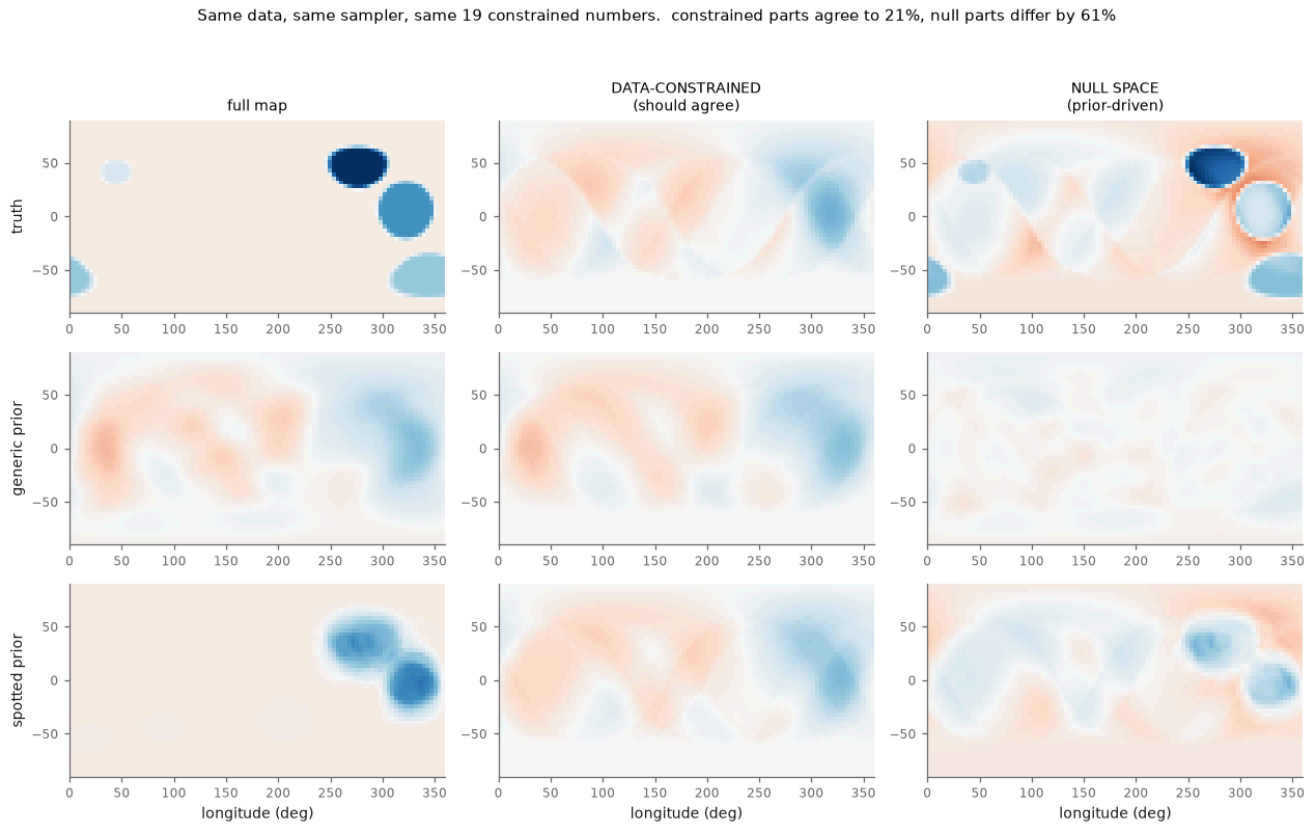


7. Swap the prior, keep the data

Two priors trained identically except for the training distribution, following the M87 recipe of treating a prior as a hypothesis. The data-constrained parts should agree — that is a *correctness check*. The null parts should not — that is the result.

In [8]:

```
pe = json.load(open("../results/prior_experiment.json"))
mp = np.load("../results/prior_experiment_maps.npz")
name = "spotted_0"
sw = next(r for r in pe if r.get("prior") == "SWAP" and r["truth"] == name
          and r["sampler"] == "pnpdm")

fig, axes = plt.subplots(3, 3, figsize=(11.6, 7.4))
keys = [("truth", "truth_constrained", "truth_null"),
        ("pnpdm_generic_mean", "pnpdm_generic_mc", "pnpdm_generic_mn"),
        ("pnpdm_spotted_mean", "pnpdm_spotted_mc", "pnpdm_spotted_mn")]
rlab = ["truth", "generic prior", "spotted prior"]
clab = ["full map", "DATA-CONSTRAINED\n(should agree)", "NULL SPACE\n(prior-
driven)"]
vt = np.abs(mp[f"{name}|truth"]).max()
vn = np.abs(mp[f"{name}|truth_null"]).max()
for i, ks in enumerate(keys):
    for j, k in enumerate(ks):
        a = axes[i, j]
        a.imshow(mp[f"{name}|{k}"], origin="upper", cmap="RdBu_r",
                 vmin=-(vt if j < 2 else vn), vmax=(vt if j < 2 else vn),
                 extent=EXT, aspect="auto")
        a.grid(False)
        if i == 0:
            a.set_title(clab[j], fontsize=9)
        if j == 0:
            a.set_ylabel(rlab[i], fontsize=9)
        if i == 2:
            a.set_xlabel("longitude (deg)")
fig.suptitle(f"Same data, same sampler, same {spec.effective_rank(1.0)} constrained
"
             f"numbers.  constrained parts agree to
{sw['data_agreement']*100:.0f}%, "
             f"null parts differ by {sw['prior_divergence']*100:.0f}%",
fontsize=10)
plt.tight_layout(rect=(0, 0, 1, .95)); plt.show()
```



8. Legitimate recovery, or hallucination?

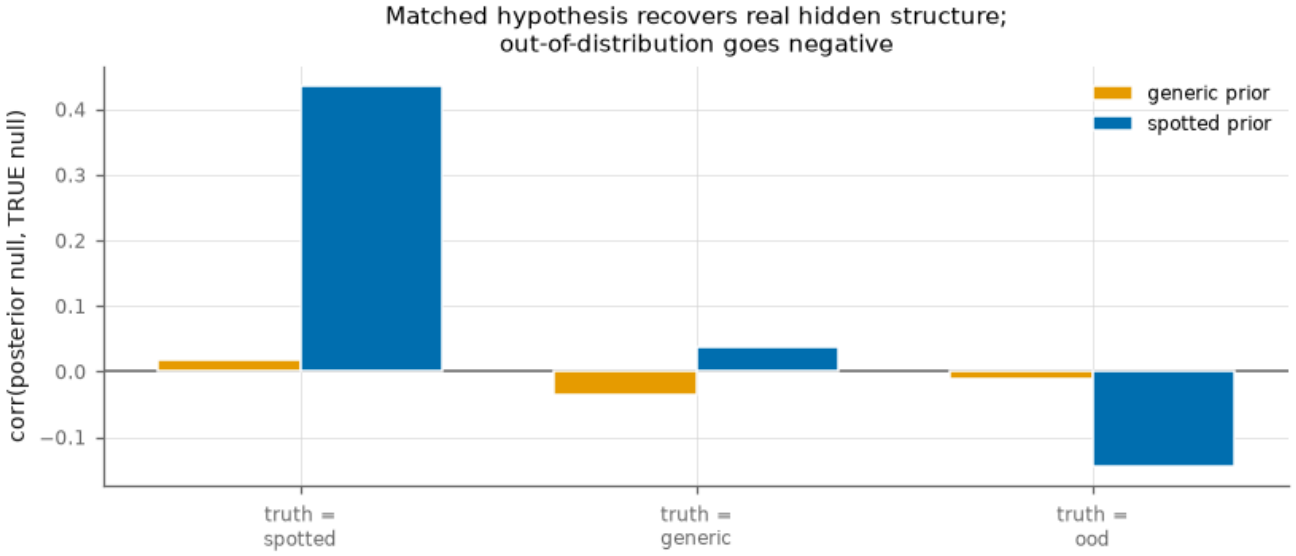
Correlate the posterior's null component with the **true** null component. Positive means the prior recovered structure that is genuinely present, which is legitimate — real stars do occupy a restricted region of map space. Near zero means nothing was recovered. **Negative** means the prior imposed structure anti-correlated with reality.

In [9]:

```
swaps = [r for r in pe if r.get("prior") == "SWAP"]
print(f"{'truth class':<14s} {'generic prior':>14s} {'spotted prior':>14s}   (PnP-
DM)")
for cls in ("spotted", "generic", "ood"):
    ss = [r for r in swaps if r["cls"] == cls and r["sampler"] == "pnpdm"]
    if ss:
        print(f"{cls:<14s} {np.median([r['null_fidelity_generic'] for r in
ss]):+14.3f} "
              f"{np.median([r['null_fidelity_spotted'] for r in ss]):+14.3f}")

print(f"\n{'sampler':<10s} {'data agree':>11s} {'prior diverge':>14s} "
      f"{'err-bar understatement':>23s}")
for s in ("tikhonov", "dps", "daps", "pnpdm"):
    ss = [r for r in swaps if r["sampler"] == s]
    if ss:
        print(f"{s:<10s} {np.median([r['data_agreement'] for r in ss]):11.3f} "
              f"{np.median([r['prior_divergence'] for r in ss]):14.3f} "
              f"{'x'+format(np.median([r['understatement_p90'] for r in
ss]),'.1f'):>23s}")

fig, ax = plt.subplots(figsize=(7.6, 3.4))
cls_list = ["spotted", "generic", "ood"]
xs = np.arange(3); width = .36
```

```
ax.set_axisbelow(True)
for k, (pk, col) in enumerate((("generic", COLORS["DAPS"]),
                               ("spotted", COLORS["PnP-DM"]))):
    vals = [np.median([r[f"null_fidelity_{pk}"] for r in swaps
                       if r["cls"] == cl and r["sampler"] == "pnpdm"])
            for cl in cls_list]
    ax.bar(xs + (k - .5) * width, vals, width, color=col, label=f"{pk} prior",
           edgecolor="white", zorder=3)
ax.axhline(0, color=MUTED, lw=1)
ax.set_xticks(xs); ax.set_xticklabels([f"truth =\n{c}" for c in cls_list])
ax.set_ylabel("corr(posterior null, TRUE null)")
ax.set_title("Matched hypothesis recovers real hidden structure;\n"
             "out-of-distribution goes negative", fontsize=9.5)
ax.legend(fontsize=8); plt.tight_layout(); plt.show()
```

```
truth class     generic prior  spotted prior   (PnP-DM)
spotted               +0.017         +0.436
generic               -0.035         +0.036
ood                   -0.012         -0.146

sampler     data agree  prior diverge  err-bar understatement
tikhonov         0.000          0.000                    x0.0
dps              0.381          1.318                    x3.0
daps             0.320          0.254                    x1.5
pnpdm            0.437          0.382                    x1.6
```



What this shows, and what it does not

Shows. On a real 60-day ATLAS window the data constrains ~19 numbers out of 4608 map pixels, and 84% of a plausible spot map's power is invisible. Whether a learned prior may legitimately fill that in is decidable: matched hypothesis recovers real hidden structure (null fidelity +0.44), mismatched recovers nothing, and out-of-distribution surfaces come back **anti**-correlated with the truth. A user cannot tell from one posterior — a single run's error bars understate the true uncertainty by up to 3×.

Does not show. The surfaces are synthetic; only the cadence and per-epoch errors are real. No comparison against a published Doppler-imaging map has been made, so there is no external ground truth here. And nothing in this notebook resolves a spot map in the sense a reader might hope — that *is* the finding.

Every "fraction constrained" is a statement about a stated SNR threshold and a stated inner product (area-weighted L^2 on the sphere), both reported with the numbers.