# DSC Analysis

Technical Guide

Distance Sample Count — Wildlife Survey Analysis Pipeline

Generated July 24, 2026

Workflow id: **dsc_analysis**

# 1. Overview

The **dsc_analysis** workflow ingests Distance Sample Count (DSC) wildlife survey data from EarthRanger and produces structured, analysis-ready datasets for use in population density modelling. The workflow supports **multiple surveys** in a single run, each defined by its own EarthRanger connection, patrol type, survey time window, and transect spatial group. Each survey is further split automatically into its **detected activity periods**, so a site visited in e.g. both February and June produces two independent, period-tagged output sets rather than one combined one.

For each survey period the workflow delivers:

- A **metadata CSV** — survey-level observations including transect IDs, observer counts, team members, and event types
- A **field effort CSV** — per-team-member daily distance travelled, duration, and man-hours, derived from EarthRanger subject GPS tracks
- An **analysis data CSV** — wildlife observation events enriched with off-transect distances, orthogonal distances, estimated animal positions, and satellite-derived environmental covariates (NDVI, terrain slope)
- An **events GeoPackage** — spatial point layer of wildlife observations with key distance sampling geometry fields
- A **transect areas GeoPackage** — visited transect corridors (buffered) labelled with mean NDVI and slope values from Google Earth Engine
- A **transect lines GeoPackage** — visited transect centrelines (unbuffered), reprojected to EPSG:4326

## Output summary

| Output type | Format | Description |
|---|---|---|
| {survey}_{period}_analysis_metadata | CSV | Survey event metadata: transect IDs, team members, observer counts, event types |
| {survey}_{period}_field_effort | CSV | Per-team-member daily distance, duration, and man-hours from GPS tracks |
| {survey}_{period}_analysis_data | CSV | Wildlife observations with distance fields, animal position estimates, NDVI, and slope covariates |
| {survey}_{period}_events | GeoPackage | Spatial point layer of wildlife observation events |
| {survey}_{period}_transect_areas | GeoPackage | Visited transect corridors (buffered) with NDVI and slope labels (EPSG:4326) |
| {survey}_{period}_transect_lines | GeoPackage | Visited transect centrelines (unbuffered) in EPSG:4326 |

**Note:** {survey} is the survey name as defined in the connection configuration and {period} is the detected activity period label (e.g. 2026_02). All six output file sets are produced independently for each survey period.

# 2. Dependencies

## 2.1 Python packages

The workflow declares six versioned packages from the Ecoscope prefix.dev channels:

| Package | Version | Channel |
|---|---|---|
| ecoscope-platform | >=2.15.0, <2.16.0 | ecoscope-workflows |
| ecoscope-workflows-ext-custom | 0.1.0rc14.* | ecoscope-workflows-custom |
| ecoscope-workflows-ext-ste | 0.0.0rc1.* | ecoscope-workflows-custom |
| ecoscope-workflows-ext-distance-sample-counts | 1.0.0.* | ecoscope-workflows-custom |
| pydeck | 0.9.2 | conda-forge |
| opentelemetry-sdk | >=1.20.0, <2.0.0 | conda-forge |

**Note:** Earlier revisions of this workflow additionally pinned separate ecoscope-workflows-core / ecoscope-workflows-ext-ecoscope packages plus site-specific extensions (ecoscope-workflows-ext-big-life, -ext-mnc, -ext-ate). These have been consolidated into the single ecoscope-platform package and removed where no longer required.

## 2.2 Google Earth Engine connection

A Google Earth Engine (GEE) service account connection is required (**set_gee_connection**). The GEE client is used to build satellite imagery composites and label transect lines with environmental covariates. The same GEE client is shared across all surveys in a single run.

## 2.3 EarthRanger connections

One EarthRanger connection is required *per survey*. Each entry in **connection_config** pairs an EarthRanger server and patrol type ID with one or more survey definitions (survey name, time window, and transect spatial group ID). Connections are split and distributed to parallel per-survey pipelines via **split_connection_configs**.

## 2.4 EarthRanger data requirements

The workflow expects the following event types to be present in EarthRanger for each patrol:

| Event type | Role |
|---|---|
| distancecountpatrol_rep | Survey metadata event — carries transect ID, team members, and observer count |
| distance_count_patrol_metadata | Alternative metadata event type (also retained during metadata filtering) |
| distancecountwildlife_rep | Wildlife observation event — carries species, total count, distance to centre, radial angle, and juvenile count |

**Note:** Field effort computation additionally requires that team members recorded on patrol metadata events (Team Members / reported_by) match named subjects/trackers in EarthRanger, so their GPS observations can be retrieved and converted into daily distance and duration statistics.

# 3. Input Configuration

## 3.1 Workflow-level parameters

| Parameter | Description |
|---|---|
| workflow_details | Human-readable name and optional description for this workflow run — used for dashboard registration |
| time_range | Required on all Ecoscope workflows. Used for timestamp display and UTC conversion only — does not filter which patrol events are fetched. Set broadly to cover all surveys in the run. Each survey's own time window controls data fetching. |
| gee_client | Google Earth Engine service account connection |
| connection_config | Array of EarthRanger connection entries, one per survey (see Section 3.2) |

## 3.2 Connection config entry (per survey)

Each entry in **connection_config** defines one EarthRanger site and one or more surveys to run against it:

| Field | Type | Description |
|---|---|---|
| earthranger.server | Connection | EarthRanger data source connection |
| earthranger.patrol_type_id | String | Numeric or UUID identifier for the patrol type containing DSC surveys |
| surveys[].surveyName | String | Unique name for this survey (used as output filename prefix) |
| surveys[].time_range | Object | Survey-specific data fetch window. Patrol events within this since / until range are retrieved from EarthRanger for this survey. This is the field that controls which data is pulled — not the top-level time_range. |
| surveys[].erSpatialTransectsGroupId | String | EarthRanger spatial group ID that contains the transect line features |

# 4. Data Ingestion

## 4.1 Connection splitting

The combined **connection_config** is split into individual per-survey connection objects via **split_connection_configs**. All subsequent steps that require per-survey data use **mapvalues** to fan out over this list in parallel.

## 4.2 Patrol events

For each survey, **fetch_patrol_events** retrieves all patrols matching the configured patrol type ID within the survey time window. The resulting patrol DataFrame is indexed on the **id** column via **set_dataframe_index** to enable subsequent join operations.

## 4.3 Survey period splitting

Before any further processing, each survey's patrol events are passed through **split_survey_by_period**, which detects the distinct activity periods within the fetched patrol events (e.g. a site visited in February and again in June) and splits them into independent, period-keyed branches. **unpack_period_connection_survey** and **unpack_period_patrol_events** then extract the connection/survey config and patrol events for each period, and **get_survey_period_label** derives the period label (e.g. **2026_02**) used in output filenames. From this point on, every downstream step fans out over survey *periods* rather than surveys.

## 4.4 Patrol transects

Transect lines are fetched per survey period via **fetch_transects**, using the spatial group ID specified in the survey's connection config. Because a survey's transect geometry does not change between periods, the same transects are fetched once per period (duplicated across periods of the same survey) so that each period's output is self-contained. The transects are GeoDataFrames in EPSG:4326 as returned by EarthRanger.

## 4.5 Survey observation events

Individual wildlife observation events are fetched from the patrol event IDs produced by period splitting using **fetch_events** (chunk_size: 75). This two-step approach — fetch patrols first, then fetch the events within them — is required because the EarthRanger API does not support direct patrol-type filtering on the events endpoint.

## 4.6 EarthRanger server name

The EarthRanger server name (site identifier) is retrieved per survey period via **get_server_name** and zipped with the survey event DataFrame. It is passed to **process_events_details** as the **client** argument so that EarthRanger field IDs can be resolved to human-readable display names. The domain name is retrieved separately via a second **get_server_name** call for use in filename/traceability construction.

# 5. Event Processing Pipeline

## 5.1 Metadata event branch

Survey metadata events (**distancecountpatrol_rep** and **distance_count_patrol_metadata**) are processed through a dedicated branch to produce the *analysis metadata* output:

| Step | Task | Purpose |
|---|---|---|
| 1 | filter_row_values | Retain only metadata event types from the raw event fetch |
| 2 | parse_df_point | Extract latitude and longitude columns from the GeoDataFrame geometry |
| 3 | normalize_json_column (reported_by) | Flatten the reported_by JSON field into flat columns |
| 4 | join_list_column (reported_by__name) | Collapse list values in reported_by__name to a comma-separated string |
| 5 | process_events_details | Resolve EarthRanger field IDs to display names (map_to_titles: true, ordered: true) |
| 6 | normalize_json_column (event_details) | Flatten the event_details JSON into flat columns |
| 7 | map_columns | Retain and rename key columns: time, event_type, serial_number, latitude, longitude, reported_by, Transect ID, Team Members, Number of Observers |
| 8 | parse_list_column / join_list_column (Team Members) | Parse and flatten Team Members list into a comma-separated string |
| 9 | drop_null_columns | Drop any columns that are entirely null after normalisation |

The resulting DataFrame is persisted as **{survey_name}_{period}_analysis_metadata.csv**.

## 5.2 Wildlife observation event branch

Wildlife observation events (**distancecountwildlife_rep**) pass through a separate normalisation branch before being merged with patrol-level metadata:

| Step | Task | Purpose |
|---|---|---|
| 1 | select_columns (event_details) | Extract only the event_details column from the raw event fetch for merging with the patrol DataFrame |
| 2 | merge_two_dataframes (left join on index) | Join the patrol DataFrame with the event_details column using the patrol id index |
| 3 | convert_column_timezone | Convert the time column to UTC using the timezone from the global time_range (used for display/conversion only, not for data filtering) |
| 4 | normalize_json_column (event_details) | Flatten the merged event_details JSON into flat columns |

| Step | Task | Purpose |
|---|---|---|
| 5 | map_columns (patrol rename) | Rename flattened distance sampling fields to standardised names (see table below) |
| 6 | format_text_column (transect_id, lower) | Lowercase transect_id so it matches the lowercased transect names used downstream for name-based transect matching |
| 7 | bfill_within_patrols / ffill_within_patrols | Backward- then forward-fill transect_id and num_observers within each patrol (group_col: patrol_serial_number) to propagate metadata event values to wildlife observation rows |
| 8 | filter_row_values (distancecountwildlife_rep) | Retain only wildlife observation events after fill propagation |
| 9 | add_constant_column (survey_id) | Stamp each row with the survey+period base name (e.g. olaremotorogi_2026_02) as a survey_id identifier |

## 5.3 Field name mapping

The **map_columns** step in the wildlife observation branch renames the flattened EarthRanger field codes to analysis-ready column names:

| Source field (EarthRanger) | Target column |
|---|---|
| event_details__distancecountpatrol_numberofobservers | num_observers |
| event_details__distancecountpatrol_teammembers | Team Members |
| event_details__distancecountpatrol_transectid | transect_id |
| event_details__distancecountwildlife_distancetocentre | dist_to_centre |
| event_details__distancecountwildlife_numberofjuveniles | num_juveniles |
| event_details__distancecountwildlife_radialangle | radialangle |
| event_details__distancecountwildlife_species | species |
| event_details__distancecountwildlife_totalcount | totalcount |
| event_details__Transect_ID (alt. form) | transect_id |
| event_details__Team_members (alt. form) | Team Members |
| event_details__Number_of_observers (alt. form) | num_observers |

**Note:** The mapping includes both the snake_case EarthRanger internal names and alternative title-case forms to handle variation across EarthRanger server configurations. raise_if_not_found is set to false so that missing columns are silently ignored.

## 5.4 Field effort computation

In parallel with the metadata and wildlife observation branches, **compute_field_effort** derives a daily field-effort summary per team member from the metadata event DataFrame (post drop_null_columns) and the survey's EarthRanger connection:

| Step | Description |
|---|---|
| 1. Roster reconstruction | Team Members (or reported_by as a fallback) is parsed per event and grouped by recorder and survey_date to build a daily team roster. If no member names are recorded, a placeholder roster of size 3 (DEFAULT_TEAM_SIZE) is generated from the recorder so that a team size estimate is still produced. |
| 2. Subject resolution | Roster member names are matched against EarthRanger subjects (get_subjects) to resolve subject IDs. If no match is found for individual members, the recorder name is matched against subjects as a fallback. |
| 3. Observation window | Each survey_date is assigned a fixed daily observation window (04:00–13:00 UTC) used to bound the GPS observation fetch for that date. |
| 4. Track retrieval | Subject GPS observations are fetched per survey_date via get_subject_observations and converted to relocations, then to trajectories (process_relocations / relocations_to_trajectory), filtered to segments between 0.1 m and 100 km, 1 second and 6 hours, and 1–100 km/h. |
| 5. Distance/duration summary | Trajectory segments are summarised per subject/day (summarize_df) into total distance (km) and total duration (h), then merged back onto the roster. |
| 6. Man-hours | man_hours is computed as team_size × duration for each subject/day row. |

The resulting DataFrame — columns survey_date, id, name, team_size, distance, duration, man_hours — is persisted as **{survey_name}_{period}_field_effort.csv**.

> **Note:** If no GPS observations are returned for a survey period, compute_field_effort returns an empty DataFrame with the expected columns rather than raising an error, so downstream persistence is skipped gracefully for that period.

# 6. Transect Processing

## 6.1 CRS conversion to UTM

Distance sampling calculations require a metric coordinate reference system. The UTM zone is estimated automatically from the transect geometry via **estimate_utm_crs**. Both the transect GeoDataFrame and the wildlife observation GeoDataFrame are then reprojected to this UTM CRS via **reproject_gdf** before any distance calculations are performed.

## 6.2 Transect simplification and buffering

Before spatial intersection tests, the transect lines are prepared as follows:

| Step | Task | Parameters | Purpose |
|---|---|---|---|
| 1 | merge_transect_lines | — | Merge all transect line segments for the survey into a unified GeoDataFrame keyed by transect name |
| 2 | simplify_transects | tolerance: 50 m | Reduce transect vertex count while preserving shape — improves intersection performance |
| 3 | buffer_transects | distance: 500 m, cap_style: flat, single_sided: false, resolution: 5 | Create a 500 m bilateral corridor around each transect for event intersection testing |

## 6.3 Event–transect intersection filter

Wildlife observation events are tested against the buffered transect corridors via **flag_events_intersecting_transect**. This step adds an **intersects_transect** boolean column and records which transect each event falls within (transect_id_column: transect_id, transect_name_column: name, index_column: id).

Only transects visited by at least one intersecting event are retained for the final output via **filter_visited_transects**. This removes transects that were planned but not walked during the survey period.

## 6.4 Re-projection to EPSG:4326 for export

After intersection filtering, visited transects are reprojected back to EPSG:4326 (**reproject_gdf**, target_crs: "epsg:4326") before satellite imagery labeling and GeoPackage export.

> **Note:** The visited, buffered corridor geometry from this step is exported as the transect_areas GeoPackage (Section 9). A second, independent branch filters the pre-buffer simplified centrelines (from Section 6.2, step 2) down to the same visited transect names (filter_transect_lines_by_visited) and reprojects them to EPSG:4326 for export as the transect_lines GeoPackage — the same visited subset, but with line rather than polygon geometry.

# 7. Geometric Calculations

## 7.1 Off-transect distance (observer position)

The perpendicular distance from the *observer's recorded GPS position* to the nearest transect centreline is calculated via **add_off_transect_distance** and stored in the column **off_transect_dist**. Events where the computed distance equals –**1** (indicating that no matching transect was found) are removed via **filter_rows** (op: "ne", value: –1).

## 7.2 Estimated animal position

The true animal position is estimated from the observer position, the recorded radial angle (**radialangle**), and the recorded distance from observer to detected animal (**dist_to_centre**) via **estimate_animal_positions**. The original observer geometry is preserved first by **add_orig_geometry** into the column **orig_geometry**; the main geometry column is then replaced with the estimated animal position.

## 7.3 Orthogonal distance (animal position)

After the animal position is estimated, the perpendicular distance from the *estimated animal position* to the transect centreline is computed via a second call to **add_off_transect_distance** and stored as **ortho_dist**. This is the key detection distance used in distance sampling density estimation models.

Events with ortho_dist == –1 are again removed via **filter_rows** before the data proceeds to imagery labeling.

## 7.4 Distance calculation summary

| Column | Task | Geometry used | Meaning |
|---|---|---|---|
| off_transect_dist | add_off_transect_distance (1st call) | Observer GPS position | Distance from observer to transect centreline — QA check |
| orig_geometry | add_orig_geometry | Observer GPS position | Preserved original observer geometry before position estimation |
| geometry | estimate_animal_positions | Computed from radialangle + dist_to_centre | Estimated true animal location |
| ortho_dist | add_off_transect_distance (2nd call) | Estimated animal position | Perpendicular distance from animal to transect — used in DSC models |

# 8. Satellite Imagery Labeling

Each transect line is labelled with two satellite-derived environmental covariates using Google Earth Engine. These covariates are commonly used as predictors in distance sampling detection function models.

## 8.1 HLS NDVI

A Harmonized Landsat Sentinel-2 (HLS) NDVI composite is built per survey via **build_hls_ndvi_image**:

| Parameter | Value | Notes |
|---|---|---|
| ndvi_window_days | null | No fixed window — uses the full image archive available up to the survey start date |
| max_cloud_cover | 30 % | Scenes with more than 30 % cloud cover are excluded from the composite |
| ndvi_band_name | NDVI_HSL | Output band name stored in the labelled transect column |

The NDVI image is evaluated over each transect's geometry and the mean pixel value within each transect is computed via **label_features_with_image_stat** (scale: 30 m, reducer_key: mean, out_column: NDVI_HSL). The survey start date is added as a **img_date_hsl_ndvi** column for traceability.

## 8.2 Terrain slope

A slope layer is derived from the USGS SRTM 1-arc-second DEM (**USGS/SRTMGL1_003**) via **build_slope_image**. The mean slope value within each transect is then computed via a second call to **label_features_with_image_stat** (scale: 30 m, reducer_key: mean, out_column: slope).

## 8.3 Imagery labeling workflow

The transects are converted to a Google Earth Engine FeatureCollection via **to_ee_feature_collection** before labeling. The period's minimum patrol event date (**get_survey_min_date**) is both added as a column to the transect DataFrame and passed to the NDVI image builder as the **since** parameter to anchor the composite date.

## 8.4 Final transect column selection

After labeling, transects are trimmed to the columns required for merging and export:

| Column | In transect_areas.gpkg | Included in merge | Description |
|---|---|---|---|
| name | Yes | Yes | Transect identifier (matches transect_id in patrol events) |
| img_date_hsl_ndvi | Yes | Yes | Period min. patrol date used as NDVI image anchor |
| NDVI_HSL | Yes | Yes | Mean HLS NDVI value along transect |
| slope | Yes | Yes | Mean terrain slope along transect |

| Column | In transect_areas.gpkg | Included in merge | Description |
|---|---|---|---|
| geometry | Yes | No | Transect corridor geometry (excluded from CSV merge) |

**Note:** The separately exported transect_lines.gpkg (Section 6.4) carries only the transect name and unbuffered line geometry — it is not labelled with NDVI_HSL, slope, or img_date_hsl_ndvi.

## 9. Output Files

All outputs are written to **$ECOSCOPE_WORKFLOWS_RESULTS**. Six file sets are produced for each survey period. {survey} is the survey name defined in the connection config and {period} is the detected activity period label (e.g. 2026_02).

| File | Format | Description |
|---|---|---|
| {survey}_{period}_analysis_metadata.csv | CSV | Survey metadata events: transect IDs, team members, observer counts, event types, lat/lon |
| {survey}_{period}_field_effort.csv | CSV | Per-team-member daily field effort: survey_date, id, name, team_size, distance (km), duration (h), man_hours |
| {survey}_{period}_analysis_data.csv | CSV | Wildlife observation events with all distance sampling fields: species, totalcount, num_juveniles, dist_to_centre, radialangle, off_transect_dist, ortho_dist, survey_id, orig_geometry (WKT), estimated geometry (WKT), NDVI_HSL, slope, img_date_hsl_ndvi, intersects_transect, transect_id, num_observers, time, serial_number, patrol_id, patrol_serial_number |
| {survey}_{period}_events.gpkg | GeoPackage | Spatial point layer of wildlife observations. Columns: serial_number, transect_id, dist_to_centre, ortho_dist, intersects_transect, geometry |
| {survey}_{period}_transect_areas.gpkg | GeoPackage | Visited transect corridors — buffered polygons (EPSG:4326) with environmental covariates: name, img_date_hsl_ndvi, NDVI_HSL, slope, geometry |
| {survey}_{period}_transect_lines.gpkg | GeoPackage | Visited transect centrelines — unbuffered lines (EPSG:4326), the same visited subset as transect_areas but without environmental covariates |

**Note:** The _events GeoPackage contains a subset of columns optimised for spatial QA and GIS workflows. The _analysis_data CSV contains the full column set including all covariates and is the primary input for distance sampling density estimation models. If a survey period has no visited transects or no field-effort GPS data, the corresponding file(s) for that period are simply not produced rather than raising an error.

# 10. Workflow Execution Logic

## 10.1 Skip conditions

All tasks share a global default skip policy defined in **task-instance-defaults**:

- **any_is_empty_df** — skips the task if any upstream DataFrame dependency is empty
- **any_dependency_skipped** — skips the task if any upstream task was itself skipped

This means that if a survey returns no patrol events or no transects, all downstream tasks for that survey branch are skipped gracefully without raising an error. Many **groupbykey** steps additionally set **any_keyed_iterables_are_skips** (unpack_depth: 1) so that if either input iterable for a given key was itself a skip, that key's pairing is dropped instead of propagating a malformed pair downstream.

> **Note:** The final publish-path aggregation step (Section 9 / dsc_publish handoff) deliberately omits any_keyed_iterables_are_skips: with many independent survey periods, it is normal for some individual period to have nothing persisted (e.g. no visited transects that period) while others succeed. groupbykey's own SkippedDependencyFallback already drops just the skipped entries and keeps the rest, so using the stricter check there would discard every other period's real output.

## 10.2 Per-survey-period fan-out with mapvalues

Every data-bearing step is executed once per survey *period* via the **mapvalues** directive. The fan-out list originates from **split_connection_configs**, is then exploded per detected activity period by **split_survey_by_period**, and flows through all subsequent steps. Key fan-out points:

| Step | mapvalues source |
|---|---|
| fetch_patrol_events | split_connection_config.return |
| period_split (activity period detection) | zip_conn_patrol_events.return |
| fetch_patrol_transects | period_connection_survey.return |
| fetch_events_from_ids | period_split.return |
| All event processing steps | Chained from fetch_events_from_ids |
| compute_field_effort | zip_connection_metadata.return |
| Transect CRS / distance steps | Chained from fetch_patrol_transects |
| GEE labeling steps | Chained from reproject_transects |
| All persist steps | Chained via zip_filename_* groupbykey |

## 10.3 groupbykey — multi-input coordination

**groupbykey** (the successor to the older zip_groupbykey task) is used throughout the workflow to combine two or more per-period iterables into paired tuples before feeding them into a multi-argument mapvalues step. Key uses:

- **zip_conn_patrol_events** — pairs split_connection_config with set_patrol_index to feed period_split
- **zip_conn_survey_df** — pairs the event DataFrame with the EarthRanger server name to feed process_events_details

- **zip_connection_metadata** — pairs period_connection_survey with the metadata summary table to feed compute_field_effort
- **zip_patrol_transects** — pairs patrol events (UTM) with transects (UTM) to feed add_off_transect_distance
- **zip_aoi_min_date** — pairs the EE FeatureCollection with the period's minimum patrol date to feed build_hls_ndvi_image
- **zip_lines_visited** — pairs the pre-buffer simplified transect lines with the visited-transects list to feed filter_transect_lines_by_visited
- **zip_filename_*** — pairs dynamically constructed filenames with DataFrames to feed persist_df

## 10.4 Dynamic filename construction

Output filenames are constructed at runtime with **join_with_underscore**. The survey name and period label are first combined into a shared base name (**combine_survey_period_name**, e.g. olaremotorogi_2026_02) that is then joined with a fixed suffix per output:

| Suffix | Output file purpose |
|---|---|
| analysis_metadata | Metadata events CSV |
| field_effort | Field effort summary CSV |
| analysis_data | Wildlife observations CSV |
| events | Events GeoPackage |
| transect_areas | Visited transect corridors (buffered) GeoPackage |
| transect_lines | Visited transect centrelines (unbuffered) GeoPackage |

## 10.5 Fill propagation for metadata fields

Patrol events in EarthRanger are recorded sequentially: a metadata event (distancecountpatrol_rep) carrying the transect ID and observer count typically appears once per transect walk, while multiple wildlife observation events (distancecountwildlife_rep) follow it. Because both event types share the same patrol_serial_number, the workflow uses backward fill (**bfill_within_patrols**) followed by forward fill (**ffill_within_patrols**) to propagate the transect_id and num_observers values to every row within the patrol before filtering to wildlife observation events only. transect_id is lowercased immediately beforehand so it matches the lowercased transect names used for name-based transect matching.

## 10.6 Publish handoff (prepared, not yet wired in)

The workflow builds a text widget (**create_text_widget_single_view**, title: "Files ready to publish") listing every persisted transects and patrol-events GeoPackage path across all surveys and periods, intended for a reviewer to copy into a companion **dsc_publish** workflow. As of this revision the widget is built but not yet attached to the dashboard (**overall_dashboard.widgets** is empty, with the widget reference commented out) — it is prepared for future wiring.

## 11. Software Versions

| Package | Version pinned |
|---|---|
| ecoscope-platform | >=2.15.0, <2.16.0 |
| ecoscope-workflows-ext-custom | 0.1.0rc14.* |
| ecoscope-workflows-ext-ste | 0.0.0rc1.* |
| ecoscope-workflows-ext-distance-sample-counts | 1.0.0.* |
| pydeck | 0.9.2 |
| opentelemetry-sdk | >=1.20.0, <2.0.0 |

**Note:** All packages are resolved from the prefix.dev Ecoscope conda channels. The wildcard patch-version pin (.*) allows bug-fix releases to be picked up automatically while keeping minor and major versions locked.