M2 Liyang Peng personal EDA

February 8, 2026

#1.Import data from github

[1]:
```
!wget https://raw.githubusercontent.com/rfordatascience/tidytuesday/main/data/
↪2023/2023-01-10/PFW_2021_public.csv
!wget https://raw.githubusercontent.com/rfordatascience/tidytuesday/main/data/
↪2023/2023-01-10/PFW_count_site_data_public_2021.csv
```

```
--2026-02-08 14:04:27--  https://raw.githubusercontent.com/rfordatascience/tidyt
uesday/main/data/2023/2023-01-10/PFW_2021_public.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.108.133, 185.199.109.133, 185.199.110.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 14704853 (14M) [text/plain]
Saving to: 'PFW_2021_public.csv'

PFW_2021_public.csv 100%[===================>]  14.02M  --.-KB/s    in 0.1s

2026-02-08 14:04:27 (129 MB/s) - 'PFW_2021_public.csv' saved [14704853/14704853]

--2026-02-08 14:04:27--  https://raw.githubusercontent.com/rfordatascience/tidyt
uesday/main/data/2023/2023-01-10/PFW_count_site_data_public_2021.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)…
185.199.109.133, 185.199.111.133, 185.199.110.133, …
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.109.133|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 32596253 (31M) [text/plain]
Saving to: 'PFW_count_site_data_public_2021.csv'

PFW_count_site_data 100%[===================>]  31.09M   201MB/s    in 0.2s

2026-02-08 14:04:28 (201 MB/s) - 'PFW_count_site_data_public_2021.csv' saved
[32596253/32596253]
```

#2.EDA for PFW_2021_public.csv

##(1)Description of the dataset about PFW_2021_public.csv

###Description The dataset records specific event information from bird observations, with the core comprising the species observed and their numbers during each survey. It includes the time of each observation (year, month, day), geographical location (latitude, longitude, administrative region), effort expended by the observer (observation duration, snow depth), and a unique identifier and quality control status for each observation record. This dataset constitutes an observational-level record, directly reflecting 'when, where, and how many birds of what species were observed'.

##(2)Data structure check

[2]:
```
#1. Data size
import pandas as pd
df = pd.read_csv("/content/PFW_2021_public.csv")
print("Rows, Columns:", df.shape)
```

```
Rows, Columns: (100000, 22)
```

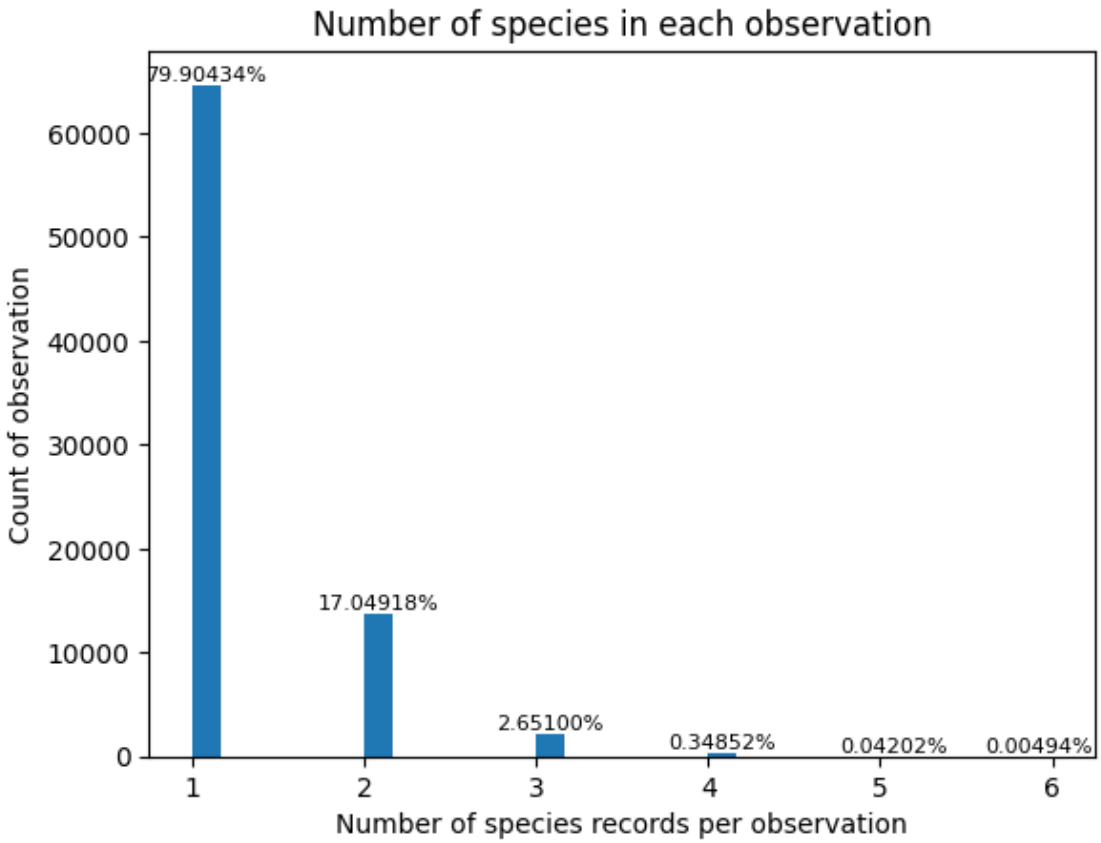
[3]:
```
#2. The number of key IDs
print("Number of sites (loc_id):", df["loc_id"].nunique())
print("Number of checklists (sub_id):", df["sub_id"].nunique())
print("Number of species (species_code):", df["species_code"].nunique())
```

```
Number of sites (loc_id): 15287
Number of checklists (sub_id): 80913
Number of species (species_code): 361
```

[4]:
```
#3. Observe the distribution of how many distinct species are recorded within␣
 ↪each checklist.
obs_per_checklist = df.groupby("sub_id").size()
print("\nNumber of species in each observation (summary):")
print(obs_per_checklist.describe())

# Plot
import numpy as np
import matplotlib.pyplot as plt
data = obs_per_checklist.values
total = len(data)
counts, bins, patches = plt.hist(data, bins=30)
plt.xlabel("Number of species records per observation")
plt.ylabel("Count of observation")
plt.title("Number of species in each observation")

for c, p in zip(counts, patches):
    if c > 0:
        percent = c / total * 100
        plt.text(
            p.get_x() + p.get_width() / 2,
            p.get_height(),
```

```
            f"{percent:.5f}%",
            ha="center",
            va="bottom",
            fontsize=8
        )
plt.show()
```

```
Number of species in each observation (summary):
count    80913.000000
mean         1.235895
std          0.510112
min          1.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          6.000000
dtype: float64
```

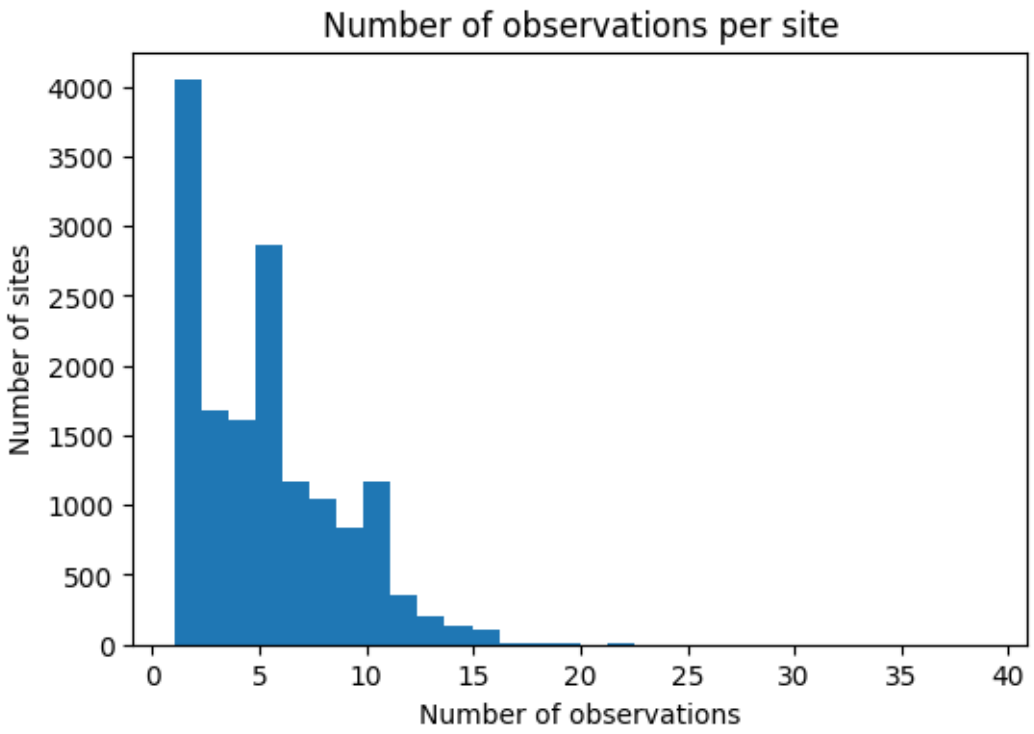


The distribution of the number of species observed in each checklist is shown in the figure above.

On average, the number observations per checklist was 80,913.

[5]:
```
#4. How many checklists correspond to each location?
checklists_per_site = df.groupby("loc_id")["sub_id"].nunique()
print("Number of observations per site (summary):")
print(checklists_per_site.describe())

#Plot
plt.figure(figsize=(6,4))
plt.hist(checklists_per_site, bins=30)

plt.xlabel("Number of observations")
plt.ylabel("Number of sites")
plt.title("Number of observations per site")

plt.show()
```

```
Number of observations per site (summary):
count    15287.000000
mean         5.292929
std          3.525224
min          1.000000
25%          2.000000
50%          5.000000
75%          8.000000
max         39.000000
Name: sub_id, dtype: float64
```



The distribution of the number of checklists provided by each site is shown in the figure above. On average, each site provides 5.29 checklists.

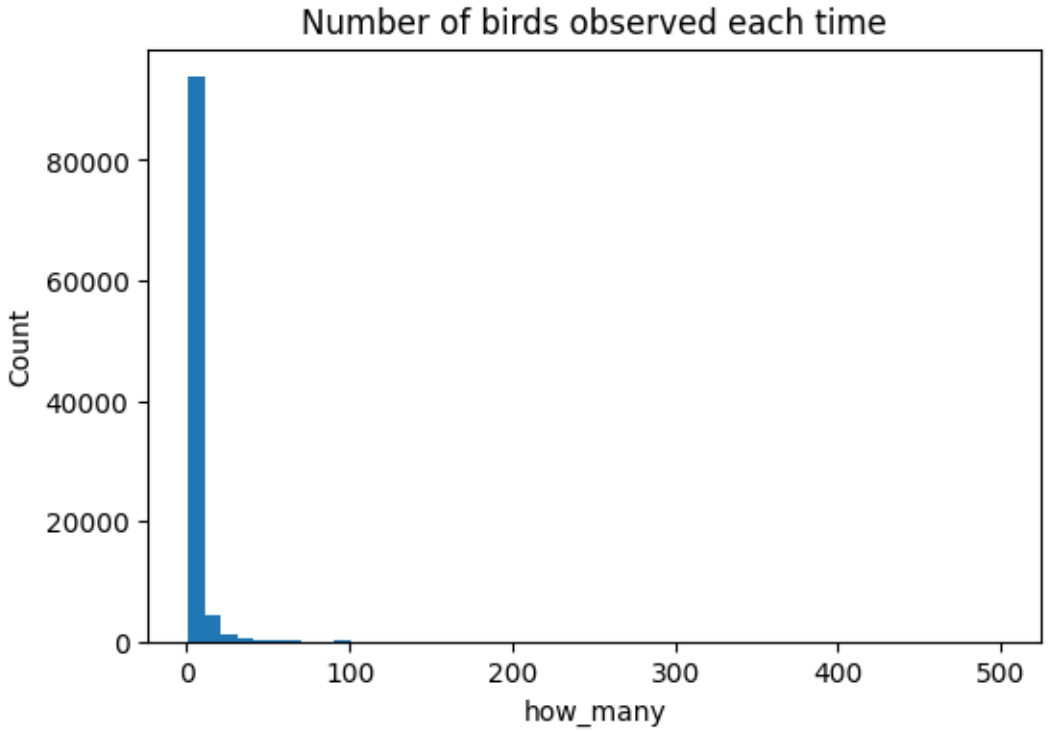
[6]:
```
#5. Check some locations contribute a large number of checklists (at the top of␣
 ↪the list)
print("\nTop 10 sites by number of checklists:")
print(checklists_per_site.sort_values(ascending=False).head(10))
```

```
Top 10 sites by number of checklists:
loc_id
L3149107     39
L8142493     36
L3165662     34
L3275296     33
L5181157     32
L12707422    31
L12753440    30
L12797130    26
L13273022    23
L874984      23
Name: sub_id, dtype: int64
```

These sites rank among the top ten in the submission checklist, suggesting that they may host

active bird populations or be frequented by researchers.

[7]:
```
# Number of birds observed each time.
import matplotlib.pyplot as plt
plt.figure(figsize=(6,4))
plt.hist(df["how_many"], bins=50)
plt.xlabel("how_many")
plt.ylabel("Count")
plt.title("Number of birds observed each time")
plt.show()
```



##(3)Data quality check and preprocessing

[8]:
```
# 1. Missing value rate check
missing_rate = df.isna().mean().sort_values(ascending=False)
print("Missing rate:")
print(missing_rate)
```

```
Missing rate:
snow_dep_atleast      0.08876
entry_technique       0.03433
effort_hrs_atleast    0.00161
longitude             0.00000
latitude              0.00000
```

```
loc_id                0.00000
sub_id                0.00000
subnational1_code     0.00000
obs_id                0.00000
Month                 0.00000
PROJ_PERIOD_ID        0.00000
species_code          0.00000
Day                   0.00000
Year                  0.00000
valid                 0.00000
how_many              0.00000
reviewed              0.00000
day1_am               0.00000
day2_am               0.00000
day1_pm               0.00000
day2_pm               0.00000
Data_Entry_Method     0.00000
dtype: float64
```

[9]:
```
#2. Missing value fixing
df["snow_dep_atleast"] = df["snow_dep_atleast"].fillna(0)
df["entry_technique"] = df["entry_technique"].fillna("Unknown")
df["effort_hrs_atleast"] = df["effort_hrs_atleast"].
  ↪fillna(df["effort_hrs_atleast"].median())
```

It is generally assumed that not writing down an estimated snow depth means there was no snowfall. Set the missing observation method category variable to a new value, "unknown".

[10]:
```
#3. Double check the missing values
missing_rate = df.isna().mean().sort_values(ascending=False)
print("Missing rate:")
print(missing_rate)
```

```
Missing rate:
loc_id                0.0
latitude              0.0
longitude             0.0
subnational1_code     0.0
entry_technique       0.0
sub_id                0.0
obs_id                0.0
Month                 0.0
Day                   0.0
Year                  0.0
PROJ_PERIOD_ID        0.0
species_code          0.0
how_many              0.0
valid                 0.0
```

```
reviewed              0.0
day1_am               0.0
day1_pm               0.0
day2_am               0.0
day2_pm               0.0
effort_hrs_atleast    0.0
snow_dep_atleast      0.0
Data_Entry_Method     0.0
dtype: float64
```

[11]:
```python
#4. Remove unverified rows (valid = 0)
print("Before:", len(df))
df = df[df["valid"] == 1]
print("After :", len(df))
```

```
Before: 100000
After : 99344
```

[12]:
```python
#5. Distinguishing variable types
df = pd.read_csv("/content/PFW_2021_public.csv")

continuous = []
categorical = []
date_like = []

for col in df.columns:
    s = df[col]

    # 1. Explicit date-related fields (by column name)
    if col.lower() in ["year", "month", "day"]:
        date_like.append(col)
        continue

    # 2. Object (string) columns -> categorical
    if s.dtype == "object":
        categorical.append(col)
        continue

    # 3. Numeric columns: decide by number of unique values
    # Small number of unique values usually indicates categorical variables (e.
 ↪g., 0/1, codes)
    nunique = s.nunique(dropna=True)

    if nunique <= 10:
        categorical.append(col)
    else:
        continuous.append(col)
```

```
print("Continuous numeric variables:")
print(continuous)

print("\nCategorical variables:")
print(categorical)

print("\nDate-related fields (to be combined into a timestamp):")
print(date_like)
```

```
Continuous numeric variables:
['latitude', 'longitude', 'how_many']

Categorical variables:
['loc_id', 'subnational1_code', 'entry_technique', 'sub_id', 'obs_id',
'PROJ_PERIOD_ID', 'species_code', 'valid', 'reviewed', 'day1_am', 'day1_pm',
'day2_am', 'day2_pm', 'effort_hrs_atleast', 'snow_dep_atleast',
'Data_Entry_Method']

Date-related fields (to be combined into a timestamp):
['Month', 'Day', 'Year']
```

[55]:
```
#6. Transform time info into timestamp
time_cols = ["Year","Month","Day"]
if all(c in df.columns for c in time_cols):
    df["timestamp"] = pd.to_datetime(df[time_cols], errors="coerce")
    df = df.drop(columns=time_cols)
else:
    print("Year / Month / Day have transformed to timestamp")
```

```
Year / Month / Day have transformed to timestamp
```

[14]:
```
#7. Outlier detection for continuous numeric variables
df[["latitude","longitude","how_many"]].quantile([0.001, 0.01, 0.99, 0.999]).T
```

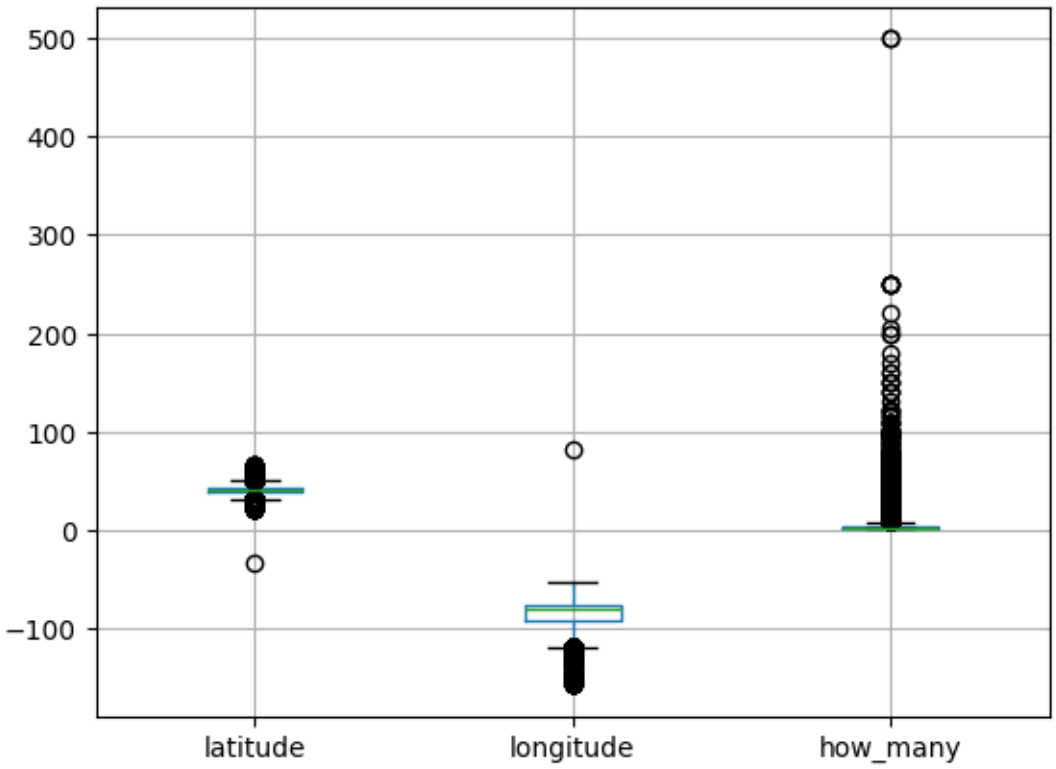
[14]:

	0.001	0.010	0.990	0.999
latitude	26.234041	29.483422	52.109427	61.110595
longitude	-149.750568	-123.495533	-68.323545	-62.180577
how_many	1.000000	1.000000	30.000000	75.000000

[15]:
```
cols = ["latitude", "longitude", "how_many"]

for col in cols:
    s = df[col].dropna()

    q1 = s.quantile(0.25)
    q3 = s.quantile(0.75)
```

```
    iqr = q3 - q1

    lower = q1 - 1.5 * iqr
    upper = q3 + 1.5 * iqr

    n_out = ((s < lower) | (s > upper)).sum()
    print(f"{col}: {n_out} outliers")
df[["latitude","longitude","how_many"]].boxplot()
plt.show()
```

```
latitude: 4971 outliers
longitude: 9948 outliers
how_many: 8516 outliers
```



Most continuous numerical values are acceptable.

[16]:
```
# Check for extreme values
cols = ["latitude", "longitude", "how_many"]

extreme_points = []

for col in cols:
```

```
    s = df[col]

    q1 = s.quantile(0.25)
    q3 = s.quantile(0.75)
    iqr = q3 - q1

    lower = q1 - 1.5 * iqr
    upper = q3 + 1.5 * iqr

    out = df[(s < lower) | (s > upper)].copy()

    if out.empty:
        continue

    out["extreme_distance"] = out[col].apply(
        lambda x: max(lower - x, x - upper)
    )

    extreme_points.append(
        out.sort_values("extreme_distance", ascending=False).head(1)
    )

extreme_df = pd.concat(extreme_points)

extreme_df[["loc_id", "latitude", "longitude", "how_many"]]
```

[16]:

	loc_id	latitude	longitude	how_many
4853	L10143380	-33.137551	81.826172	1
4853	L10143380	-33.137551	81.826172	1
50182	L12735656	43.084770	-70.884349	500

Delete the observation data from the islands on the west coast of the South Pacific

[17]:
```
before = len(df)
df = df[df["loc_id"] != "L10143380"]
after = len(df)
print("Removed rows:", before - after)
```

```
Removed rows: 1
```

##(4)Spatial distribution trends of different bird species

[18]:
```
# Check area
df[["latitude", "longitude"]].agg(["min", "max"])
```

[18]:

	latitude	longitude
min	21.422854	-157.949318
max	65.519890	-52.733096

[19]:
```
# Select one of the speices_code in next block
df["species_code"].unique()
```

[19]:
```
array(['amegfi', 'moudov', 'tuftit', 'houspa', 'balori', 'norcar',
       'pinsis', 'brnthr', 'whtspa', 'pingro', 'rewbla', 'brncre',
       'dowwoo', 'orejun', 'yerwar', 'eursta', 'annhum', 'whcspa',
       'whbnut', 'coshum', 'reshaw', 'haiwoo', 'orcwar', 'bkcchi',
       'easblu', 'amerob', 'y00033', 'pilwoo', 'varthr', 'rebnut',
       'chispa', 'daejun', 'rebwoo', 'stejay', 'blujay', 'bnhcow',
       'carwre', 'herthr', 'rethaw', 'chbchi', 'houfin', 'calqua',
       'carchi', 'monpar', 'foxspa', 'eucdov', 'mouchi', 'pinwar',
       'amecro', 'amtspa', 'normoc', 'rocpig1', 'sonspa', 'norfli',
       'lesgol', 'grycat', 'spotow', 'gilwoo', 'gryjay', 'caltow',
       'paibun', 'yebsap', 'coohaw', 'btywar', 'comrav', 'wlswar',
       'cowscj1', 'borchi2', 'pygnut', 'juntit1', 'bewwre', 'purfin',
       'eastow', 'comgra', 'bkbmag1', 'rufhum', 'comred', 'hoared',
       'clanut', 'wiltur', 'yetwar', 'larspa', 'whwdov', 'acowoo',
       'robgro', 'fiscro', 'blkpho', 'wooscj2', 'phaino', 'nutman',
       'bushti', 'bnhnut', 'abetow', 'allhum', 'mallar2', 'incdov',
       'shshaw', 'norpar', 'evegro', 'brebla', 'fiespa', 'rebsap',
       'linspa', 'ruckin', 'brthum', 'blctit4', 'norcro', 'easpho',
       'bbwduc', 'mallar3', 'rthhum', 'gocspa', 'nutwoo', 'casfin',
       'turvul', 'cedwax', 'wooduc', 'rehwoo', 'comyel', 'gofwoo',
       'cantow', 'scaqua', 'snobun', 'grhowl', 'brbhum', 'baleag',
       'towwar', 'yesfli', 'rufgro', 'slcjun', 'hergul', 'cangoo',
       'oaktit', 'labwoo', 'wesblu', 'gockin', 'rusbla', 'limpki',
       'cubthr', 'pinjay', 'towsol', 'bawwar', 'pisjun', 'savspa',
       'houwre', 'selasp', 'greegr', 'bkchum', 'gryvir', 'batpig1',
       'palwar', 'eutspa', 'lobthr', 'brdowl', 'norgos', 'bktgna',
       'grbher3', 'verdin', 'bohwax', 'gbbgul', 'gamqua', 'rocwre',
       'botgra', 'y00327', 'budger', 'clcspa', 'greroa', 'lucwar',
       'rinphe', 'altori', 'pyrrhu', 'britit', 'rucspa', 'amekes',
       'wesmea', 'woosto', 'hummin', 'norhar', 'casvir', 'mexjay4',
       'cogdov', 'buggna', 'norshr4', 'hutvir', 'chirav', 'audwar',
       'scoori', 'grnjay', 'saypho', 'hooori', 'yebmag', 'wesowl1',
       'killde', 'ruwspa', 'treswa', 'harspa', 'myrwar', 'y00475',
       'glwgul', 'whiibi', 'snoegr', 'grbher', 'ovenbi1', 'resfli',
       'sumtan', 'indbun', 'norbob', 'bkhgro', 'bktspa', 'belkin1',
       'westan', 'y00226', 'hawk', 'calhum', 'redcro', 'brespa', 'pibgre',
       'osprey', 'grtgra', 'whhwoo', 'gyhjun', 'rewbul', 'ribgul',
       'rosfin', 'hoomer', 'sancra', 'crithr', 'brocow', 'musduc3',
       'woothr', 'buffle', 'buhvir', 'calthr', 'easowl1', 'yelwar',
       'gcrfin', 'rorpar', 'peflov', 'yerwar3', 'wrenti', 'grcfly',
       'cacwre', 'libher', 'sparro1', 'audori', 'grekis', 'astfly',
       'reblei', 'barswa', 'norcar1', 'blkvul', 'norhar2', 'mutswa',
       'olispa', 'amekes1', 'wilsap', 'benthr', 'accipi', 'blackb',
       'loeowl', 'doccor', 'musduc', 'merlin', 'reshaw4', 'swtkit',
```

```
       'belvir', 'vigswa', 'foxsp1', 'gnttow', 'whevir', 'commer',
       'greegr2', 'commyn', 'easmea', 'moublu', 'shtgro', 'chwwid',
       'winwre3', 'canwre', 'purmar', 'whwcro', 'whwjun', 'lazbun',
       'daejun1', 'reshaw5', 'pitwhy', 'yehbla', 'brwhaw', 'y00322',
       'gilfli', 'pacwre1', 'egygoo', 'blugrb1', 'caskin', 'hrshaw',
       'anhing', 'comgol', 'lewwoo', 'scatan', 'foxsp2', 'monpar1',
       'revbul', 'magwar', 'wesgul', 'swaspa', 'lessca', 'nswowl',
       'x00004', 'redpol', 'comgal1', 'swahaw', 'rebjun1', 'y00227',
       'buwtea', 'sonspa2', 'canvas', 'camwar', 'spbori', 'laugul',
       'yetvir', 'bulori', 'amered', 'whtdov', 'prawar', 'orcori',
       'wesjay', 'bkrfin', 'rewbla1', 'cliswa', 'louwat', 'motduc',
       'chemun', 'amecoo', 'ambduc', 'dickci', 'gadwal', 'rinduc', 'sora',
       'spodov', 'neocor', 'brnowl', 'hoowar', 'sagthr', 'clcrob',
       'swiwhe1', 'sagspa', 'scrjay', 'weskin', 'zebdov', 'y00471',
       'gryfly', 'sheowl', 'rocpig', 'vichum', 'logshr', 'x00195',
       'naswar', 'paired', 'rolhaw', 'x00422', 'wilsni1'], dtype=object)
```

[20]:

```
!pip -q install plotly

import json
import pandas as pd
import numpy as np
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from urllib.request import urlopen

# Settings
# Choose ONE species to search
SPECIES = "acowoo"   # <- change to your target species_code

# 0 Load observation data
df = pd.read_csv("/content/PFW_2021_public.csv")
df = df[df["valid"] == 1].copy()

# subnational1_code like "US-CA", "CA-ON", sometimes "XX"
df = df[df["subnational1_code"].notna()]
df = df[df["subnational1_code"].str.contains("-", na=False)]
df = df[~df["subnational1_code"].str.startswith("XX", na=False)]


# 1 Load North America admin-1 boundaries (US states + Canada provinces/
 ↪territories) via Natural Earth GeoJSON
#    (Worldwide file; we'll filter to US/CA)
geojson_url = "https://raw.githubusercontent.com/nvkelso/natural-earth-vector/
 ↪master/geojson/ne_50m_admin_1_states_provinces.geojson"
with urlopen(geojson_url) as f:
    gj = json.load(f)
```

```
# Keep only US/CA features that have iso_3166_2 (e.g., "US-CA", "CA-ON")
features = []
for feat in gj["features"]:
    props = feat.get("properties", {})
    iso2 = props.get("iso_3166_2")
    if iso2 and (iso2.startswith("US-") or iso2.startswith("CA-")):
        features.append(feat)

na_geojson = {"type":"FeatureCollection", "features":features}


# 2 Assign each state/province to the 6 regions (edit if you want different␣
 ↪grouping)
region_map = {
    # --- US ---
    "US-AK":"Far North",

    "US-WA":"Northwest","US-OR":"Northwest","US-ID":"Northwest","US-MT":
 ↪"Northwest",

    "US-CA":"Southwest","US-NV":"Southwest","US-AZ":"Southwest","US-NM":
 ↪"Southwest","US-UT":"Southwest","US-CO":"Southwest",

    "US-ND":"Central","US-SD":"Central","US-NE":"Central","US-KS":"Central",
    "US-MN":"Central","US-IA":"Central","US-MO":"Central","US-WI":"Central",
    "US-IL":"Central","US-MI":"Central","US-IN":"Central","US-OH":"Central",

    "US-ME":"Northeast","US-NH":"Northeast","US-VT":"Northeast","US-MA":
 ↪"Northeast","US-RI":"Northeast","US-CT":"Northeast",
    "US-NY":"Northeast","US-NJ":"Northeast","US-PA":"Northeast",

    "US-DE":"Southeast","US-MD":"Southeast","US-DC":"Southeast","US-VA":
 ↪"Southeast","US-WV":"Southeast","US-NC":"Southeast","US-SC":"Southeast",
    "US-GA":"Southeast","US-FL":"Southeast","US-KY":"Southeast","US-TN":
 ↪"Southeast","US-AL":"Southeast","US-MS":"Southeast",
    "US-AR":"Southeast","US-LA":"Southeast","US-OK":"Southeast","US-TX":
 ↪"Southeast",

    # --- Canada (rough grouping to match the example-style regions) ---
    "CA-YT":"Far North","CA-NT":"Far North","CA-NU":"Far North",

    "CA-BC":"Northwest",

    "CA-AB":"Central","CA-SK":"Central","CA-MB":"Central",
```

```
    "CA-ON":"Northeast","CA-QC":"Northeast","CA-NB":"Northeast","CA-NS":
 ↪"Northeast","CA-PE":"Northeast","CA-NL":"Northeast"
}
df["region"] = df["subnational1_code"].map(region_map).fillna("Central")

region_order = ["Far␣
 ↪North","Northwest","Southwest","Central","Northeast","Southeast"]
region_to_z = {r:i for i,r in enumerate(region_order)}
region_colors = ["#7D5BA6", "#16B6C6", "#E78AC3", "#33A02C", "#F28E2B",␣
 ↪"#D62728"]  # 6 discrete colors

# Region bubble anchor points (rough centers; include Canada)
region_centers = {
    "Far North":  (64.0, -110.0),
    "Northwest":  (53.0, -125.0),
    "Southwest":  (34.0, -112.0),
    "Central":    (49.0,  -97.0),
    "Northeast":  (46.0,  -71.0),
    "Southeast":  (32.5,  -84.0),
}


# 3 Metrics (ALL time; spatial only)
#    Left: % feeders visited = (# unique sites where species appears) / (#␣
 ↪unique sites) by region
#    Right: avg flock size when present (mean how_many), aggregated by region␣
 ↪(site-level then region-level)

# total unique sites by region
sites_all = df.groupby("region")["loc_id"].nunique().rename("n_sites_all")

# species-present subset
df_sp = df[df["species_code"] == SPECIES].copy()
sites_sp = df_sp.groupby("region")["loc_id"].nunique().rename("n_sites_sp")

metrics = pd.concat([sites_all, sites_sp], axis=1).fillna(0)
metrics["pct_feeders_visited"] = np.where(metrics["n_sites_all"] > 0,
                                          metrics["n_sites_sp"] /␣
 ↪metrics["n_sites_all"] * 100, 0)

# avg flock size: mean within site -> mean across sites
loc_mean = df_sp.groupby(["region","loc_id"])["how_many"].mean().
 ↪rename("loc_mean_how_many").reset_index()
avg_flock = loc_mean.groupby("region")["loc_mean_how_many"].mean().
 ↪rename("avg_flock_size")
```

```
metrics = metrics.join(avg_flock).fillna({"avg_flock_size":0}).reset_index()
metrics["lat"] = metrics["region"].map(lambda r: region_centers[r][0])
metrics["lon"] = metrics["region"].map(lambda r: region_centers[r][1])


# 4 Admin-1 polygons colored by region (US states + Canada provinces/
 ↪territories)
# Build a table of all admin-1 units we can map
admin_units = []
for feat in na_geojson["features"]:
    iso2 = feat["properties"]["iso_3166_2"]  # "US-CA" / "CA-ON"
    admin_units.append(iso2)
admin_units = pd.Series(admin_units, name="subnational1_code").to_frame()

admin_units["region"] = admin_units["subnational1_code"].map(region_map).
 ↪fillna("Central")
admin_units["z"] = admin_units["region"].map(region_to_z)


# 5 Plot
fig = make_subplots(
    rows=1, cols=2,
    specs=[[{"type":"choropleth"}, {"type":"choropleth"}]],
    subplot_titles=("Percent Feeders Visited", "Average Flock Size")
)

for col_i, metric_col, label_fmt, size_expr in [
    (1, "pct_feeders_visited", lambda x: f"{x:.1f}%", lambda x: np.clip(x, 10,␣
 ↪60)),
    (2, "avg_flock_size",      lambda x: f"{x:.3g}",  lambda x: np.clip(x*8,␣
 ↪10, 60)),
]:
    fig.add_trace(
        go.Choropleth(
            geojson=na_geojson,
            featureidkey="properties.iso_3166_2",
            locations=admin_units["subnational1_code"],
            z=admin_units["z"],
            zmin=0, zmax=5,
            colorscale=region_colors,
            showscale=False,
            marker_line_width=0.6
        ),
        row=1, col=col_i
    )

    fig.add_trace(
```

```
        go.Scattergeo(
            lon=metrics["lon"], lat=metrics["lat"],
            mode="markers+text",
            text=metrics[metric_col].map(label_fmt),
            textposition="middle center",
            marker=dict(size=size_expr(metrics[metric_col]), opacity=0.35),
            showlegend=False
        ),
        row=1, col=col_i
    )

fig.update_geos(
    scope="north america",
    projection_type="albers",

)



fig.update_layout(
    title=f"{SPECIES} - US + Canada  2020-11-13 to 2021-04-30 ",
    height=620, width=1250,
    margin=dict(l=10, r=10, t=70, b=10)
)

fig.show()
```

##(5)Time distribution trends of different bird species

[21]:
```
# Check timeline
time_cols = ["Year","Month","Day"]
if all(c in df.columns for c in time_cols):
    df["timestamp"] = pd.to_datetime(df[time_cols], errors="coerce")
    df = df.drop(columns=time_cols)
else:
    print("Year / Month / Day have transformed to timestamp")

print("Min timestamp:", df["timestamp"].min())
print("Max timestamp:", df["timestamp"].max())
```

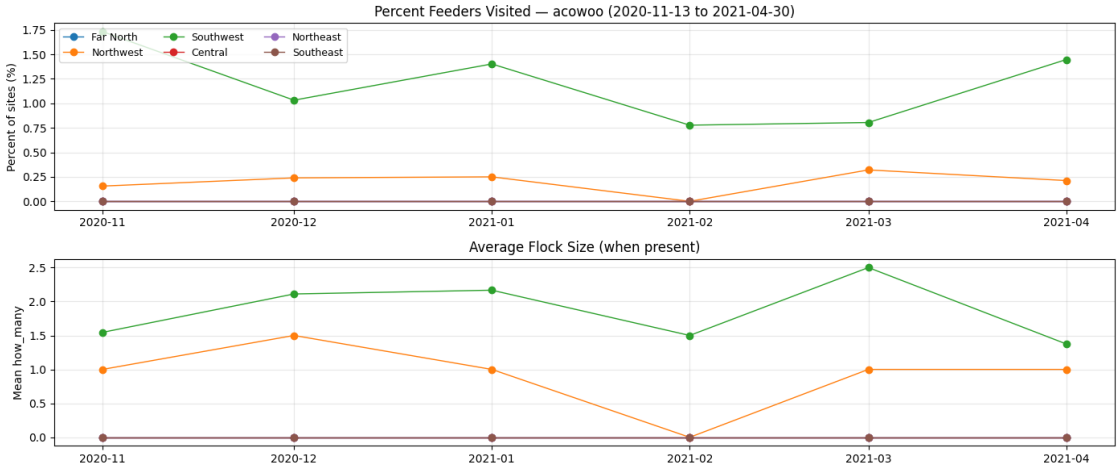
```
Min timestamp: 2020-11-13 00:00:00
Max timestamp: 2021-04-30 00:00:00
```

[22]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
# Settings

SPECIES = "acowoo"    # change to your species_code
FREQ = "M"        # "M"=monthly, "W"=weekly, "D"=daily


# Load and clean

df = pd.read_csv("/content/PFW_2021_public.csv")
df = df[df["valid"] == 1].copy()

df["date"] = pd.to_datetime(df[["Year", "Month", "Day"]], errors="coerce")
df = df.dropna(subset=["date"])

df = df[df["subnational1_code"].notna()]
df = df[df["subnational1_code"].str.contains("-", na=False)]
df = df[df["subnational1_code"] != "XX-"].copy()


# Region mapping (US + Canada)

region_map = {
    # US
    "US-AK": "Far North",
    "US-WA": "Northwest", "US-OR": "Northwest", "US-ID": "Northwest", "US-MT":␣
 ↪"Northwest",
    "US-CA": "Southwest", "US-NV": "Southwest", "US-AZ": "Southwest", "US-NM":␣
 ↪"Southwest", "US-UT": "Southwest", "US-CO": "Southwest",
    "US-ND": "Central", "US-SD": "Central", "US-NE": "Central", "US-KS":␣
 ↪"Central",
    "US-MN": "Central", "US-IA": "Central", "US-MO": "Central", "US-WI":␣
 ↪"Central",
    "US-IL": "Central", "US-MI": "Central", "US-IN": "Central", "US-OH":␣
 ↪"Central",
    "US-ME": "Northeast", "US-NH": "Northeast", "US-VT": "Northeast", "US-MA":␣
 ↪"Northeast",
    "US-RI": "Northeast", "US-CT": "Northeast", "US-NY": "Northeast", "US-NJ":␣
 ↪"Northeast", "US-PA": "Northeast",
    "US-DE": "Southeast", "US-MD": "Southeast", "US-DC": "Southeast", "US-VA":␣
 ↪"Southeast", "US-WV": "Southeast",
    "US-NC": "Southeast", "US-SC": "Southeast", "US-GA": "Southeast", "US-FL":␣
 ↪"Southeast",
    "US-KY": "Southeast", "US-TN": "Southeast", "US-AL": "Southeast", "US-MS":␣
 ↪"Southeast",
```

```
    "US-AR": "Southeast", "US-LA": "Southeast", "US-OK": "Southeast", "US-TX":␣
 ↪"Southeast",

    # Canada
    "CA-YT": "Far North", "CA-NT": "Far North", "CA-NU": "Far North",
    "CA-BC": "Northwest",
    "CA-AB": "Central", "CA-SK": "Central", "CA-MB": "Central",
    "CA-ON": "Northeast", "CA-QC": "Northeast", "CA-NB": "Northeast",
    "CA-NS": "Northeast", "CA-PE": "Northeast", "CA-NL": "Northeast",
}

region_order = ["Far North", "Northwest", "Southwest", "Central", "Northeast",␣
 ↪"Southeast"]
df["region"] = df["subnational1_code"].map(region_map).fillna("Central")


# Time bucketing

if FREQ == "M":
    df["t"] = df["date"].dt.to_period("M").dt.to_timestamp()
elif FREQ == "W":
    df["t"] = df["date"].dt.to_period("W").apply(lambda p: p.start_time)
else:
    df["t"] = df["date"].dt.floor("D")


# Metric 1: Percent feeders visited (site coverage)
#   numerator: unique sites where species appears (by time & region)
#   denominator: unique sites observed (by time & region)

sites_all = df.groupby(["t", "region"])["loc_id"].nunique().
 ↪rename("n_sites_all")

df_sp = df[df["species_code"] == SPECIES].copy()
sites_sp = df_sp.groupby(["t", "region"])["loc_id"].nunique().
 ↪rename("n_sites_sp")

m = pd.concat([sites_all, sites_sp], axis=1).fillna(0)
m["pct_feeders_visited"] = np.where(
    m["n_sites_all"] > 0,
    (m["n_sites_sp"] / m["n_sites_all"]) * 100,
    0
)


# Metric 2: Average flock size when present
#   compute mean how_many within site first, then average across sites
```

```
loc_mean = (
    df_sp.groupby(["t", "region", "loc_id"])["how_many"]
    .mean()
    .rename("loc_mean_how_many")
    .reset_index()
)

avg_flock = (
    loc_mean.groupby(["t", "region"])["loc_mean_how_many"]
    .mean()
    .rename("avg_flock_size")
)

m = m.join(avg_flock).fillna({"avg_flock_size": 0}).reset_index()

p1 = m.pivot(index="t", columns="region", values="pct_feeders_visited").
 ↪reindex(columns=region_order)
p2 = m.pivot(index="t", columns="region", values="avg_flock_size").
 ↪reindex(columns=region_order)


# Plot
start_date = df["date"].min().date()
end_date = df["date"].max().date()

plt.figure(figsize=(14, 6))

ax1 = plt.subplot(2, 1, 1)
for r in region_order:
    ax1.plot(p1.index, p1[r], marker="o", linewidth=1)
ax1.set_title(f"Percent Feeders Visited - {SPECIES} ({start_date} to␣
 ↪{end_date})")
ax1.set_ylabel("Percent of sites (%)")
ax1.grid(True, alpha=0.3)
ax1.legend(region_order, ncol=3, fontsize=9, loc="upper left")

ax2 = plt.subplot(2, 1, 2, sharex=ax1)
for r in region_order:
    ax2.plot(p2.index, p2[r], marker="o", linewidth=1)
ax2.set_title("Average Flock Size (when present)")
ax2.set_ylabel("Mean how_many")
ax2.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()
```



##(6)Frequency distribution of different bird species in different months

[23]:
```
import pandas as pd

# Load and basic cleaning

df = pd.read_csv("/content/PFW_2021_public.csv")

df = df[df["valid"] == 1].copy()

df = df[df["subnational1_code"].notna()]
df = df[df["subnational1_code"].str.contains("-", na=False)]
df = df[df["subnational1_code"] != "XX"]


# Build date and month
df["date"] = pd.to_datetime(df[["Year","Month","Day"]], errors="coerce")
df = df.dropna(subset=["date"])
df["month"] = df["date"].dt.to_period("M").astype(str)


# Region mapping
region_map = {
    "US-AK":"Far North",

    "US-WA":"Northwest","US-OR":"Northwest","US-ID":"Northwest","US-MT":
 ↪"Northwest",

    "US-CA":"Southwest","US-NV":"Southwest","US-AZ":"Southwest","US-NM":
 ↪"Southwest","US-UT":"Southwest","US-CO":"Southwest",
```

```
    "US-ND":"Central","US-SD":"Central","US-NE":"Central","US-KS":"Central",
    "US-MN":"Central","US-IA":"Central","US-MO":"Central","US-WI":"Central",
    "US-IL":"Central","US-MI":"Central","US-IN":"Central","US-OH":"Central",

    "US-ME":"Northeast","US-NH":"Northeast","US-VT":"Northeast","US-MA":
 ↪"Northeast",
    "US-RI":"Northeast","US-CT":"Northeast","US-NY":"Northeast","US-NJ":
 ↪"Northeast","US-PA":"Northeast",

    "US-DE":"Southeast","US-MD":"Southeast","US-DC":"Southeast","US-VA":
 ↪"Southeast","US-WV":"Southeast",
    "US-NC":"Southeast","US-SC":"Southeast","US-GA":"Southeast","US-FL":
 ↪"Southeast",
    "US-KY":"Southeast","US-TN":"Southeast","US-AL":"Southeast","US-MS":
 ↪"Southeast",
    "US-AR":"Southeast","US-LA":"Southeast","US-OK":"Southeast","US-TX":
 ↪"Southeast",

    "CA-YT":"Far North","CA-NT":"Far North","CA-NU":"Far North",
    "CA-BC":"Northwest",
    "CA-AB":"Central","CA-SK":"Central","CA-MB":"Central",
    "CA-ON":"Northeast","CA-QC":"Northeast","CA-NB":"Northeast",
    "CA-NS":"Northeast","CA-PE":"Northeast","CA-NL":"Northeast"
}

df["region"] = df["subnational1_code"].map(region_map).fillna("Central")


# Count species frequency
# per month × region
#
counts = (
    df
    .groupby(["month", "region", "species_code"])
    .size()
    .reset_index(name="n_records")
)

# Top 5 species per month × region

top5 = (
    counts
    .sort_values(["month", "region", "n_records"], ascending=[True, True,␣
 ↪False])
    .groupby(["month", "region"])
    .head(5)
```

```
    .reset_index(drop=True)
)

from IPython.display import display
pd.set_option("display.max_rows", 200)
pd.set_option("display.max_columns", 50)
pd.set_option("display.width", 200)

display(top5)
```

	month	region	species_code	n_records
0	2020-11	Central	norcar	174
1	2020-11	Central	blujay	167
2	2020-11	Central	dowwoo	167
3	2020-11	Central	bkcchi	160
4	2020-11	Central	whbnut	151
5	2020-11	Far North	dowwoo	5
6	2020-11	Far North	pingro	4
7	2020-11	Far North	bkbmag1	3
8	2020-11	Far North	rebnut	3
9	2020-11	Far North	bkcchi	2
10	2020-11	Northeast	bkcchi	344
11	2020-11	Northeast	blujay	331
12	2020-11	Northeast	whbnut	320
13	2020-11	Northeast	daejun	290
14	2020-11	Northeast	norcar	287
15	2020-11	Northwest	bkcchi	77
16	2020-11	Northwest	norfli	69
17	2020-11	Northwest	daejun	66
18	2020-11	Northwest	houfin	52
19	2020-11	Northwest	rebnut	52
20	2020-11	Southeast	norcar	190
21	2020-11	Southeast	carwre	143
22	2020-11	Southeast	tuftit	143
23	2020-11	Southeast	moudov	142
24	2020-11	Southeast	carchi	140
25	2020-11	Southwest	daejun	76
26	2020-11	Southwest	houfin	75
27	2020-11	Southwest	whcspa	45
28	2020-11	Southwest	lesgol	42
29	2020-11	Southwest	annhum	38
30	2020-12	Central	dowwoo	260
31	2020-12	Central	daejun	257
32	2020-12	Central	bkcchi	254
33	2020-12	Central	norcar	242
34	2020-12	Central	blujay	236
35	2020-12	Far North	rebnut	7

36	2020-12	Far North	pingro	6
37	2020-12	Far North	bkcchi	5
38	2020-12	Far North	borchi2	4
39	2020-12	Far North	bkbmag1	3
40	2020-12	Northeast	bkcchi	572
41	2020-12	Northeast	blujay	481
42	2020-12	Northeast	dowwoo	480
43	2020-12	Northeast	moudov	465
44	2020-12	Northeast	daejun	452
45	2020-12	Northwest	daejun	109
46	2020-12	Northwest	bkcchi	108
47	2020-12	Northwest	houfin	85
48	2020-12	Northwest	annhum	84
49	2020-12	Northwest	pinsis	76
50	2020-12	Southeast	norcar	296
51	2020-12	Southeast	tuftit	258
52	2020-12	Southeast	moudov	242
53	2020-12	Southeast	dowwoo	232
54	2020-12	Southeast	houfin	224
55	2020-12	Southwest	houfin	126
56	2020-12	Southwest	daejun	108
57	2020-12	Southwest	whcspa	69
58	2020-12	Southwest	lesgol	58
59	2020-12	Southwest	pinsis	57
60	2021-01	Central	dowwoo	312
61	2021-01	Central	daejun	269
62	2021-01	Central	norcar	267
63	2021-01	Central	houspa	256
64	2021-01	Central	bkcchi	248
65	2021-01	Far North	rebnut	9
66	2021-01	Far North	comred	6
67	2021-01	Far North	bkcchi	4
68	2021-01	Far North	dowwoo	4
69	2021-01	Far North	borchi2	2
70	2021-01	Northeast	bkcchi	569
71	2021-01	Northeast	whbnut	559
72	2021-01	Northeast	dowwoo	534
73	2021-01	Northeast	daejun	511
74	2021-01	Northeast	blujay	484
75	2021-01	Northwest	bkcchi	113
76	2021-01	Northwest	daejun	112
77	2021-01	Northwest	rebnut	98
78	2021-01	Northwest	annhum	87
79	2021-01	Northwest	spotow	80
80	2021-01	Southeast	norcar	339
81	2021-01	Southeast	tuftit	288
82	2021-01	Southeast	carwre	286
83	2021-01	Southeast	amegfi	281

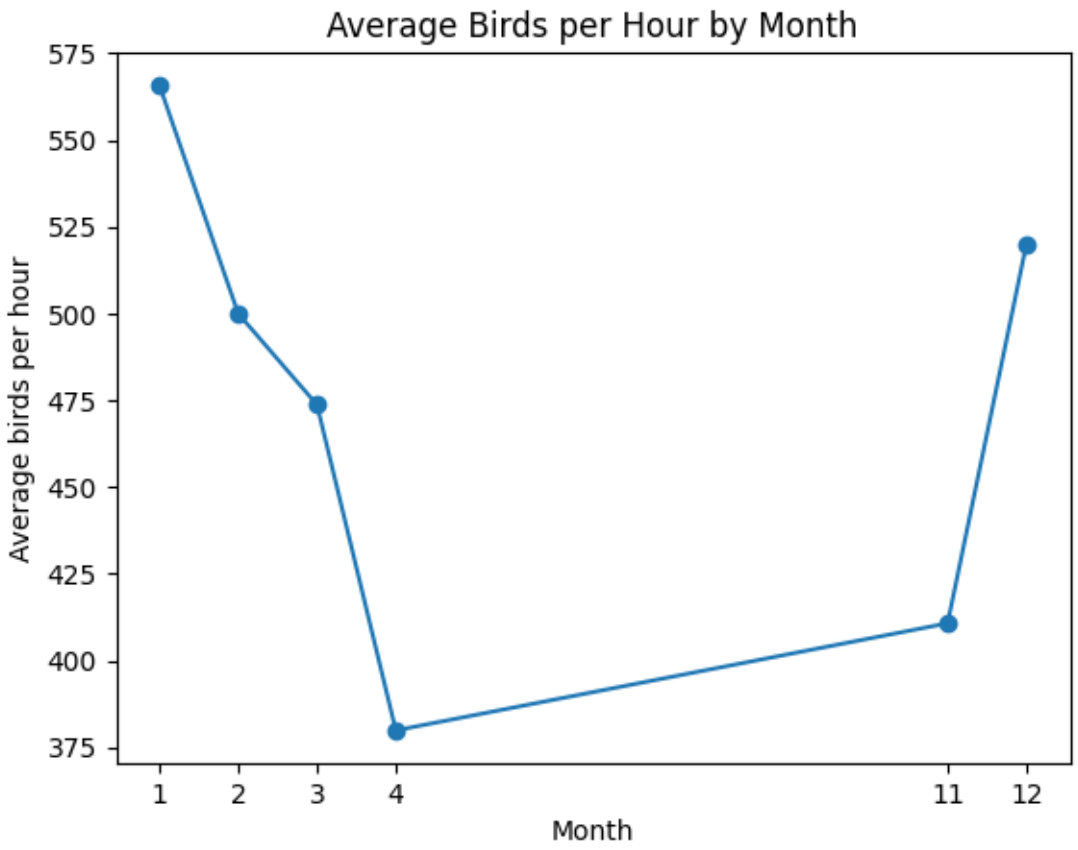
84	2021-01	Southeast	houfin	265
85	2021-01	Southwest	houfin	129
86	2021-01	Southwest	daejun	101
87	2021-01	Southwest	lesgol	70
88	2021-01	Southwest	moudov	70
89	2021-01	Southwest	whcspa	60
90	2021-02	Central	dowwoo	304
91	2021-02	Central	norcar	246
92	2021-02	Central	daejun	235
93	2021-02	Central	houspa	210
94	2021-02	Central	whbnut	202
95	2021-02	Far North	bkcchi	9
96	2021-02	Far North	pingro	5
97	2021-02	Far North	rebnut	5
98	2021-02	Far North	comred	4
99	2021-02	Far North	dowwoo	3
100	2021-02	Northeast	bkcchi	516
101	2021-02	Northeast	daejun	471
102	2021-02	Northeast	norcar	456
103	2021-02	Northeast	dowwoo	439
104	2021-02	Northeast	blujay	400
105	2021-02	Northwest	daejun	98
106	2021-02	Northwest	bkcchi	88
107	2021-02	Northwest	annhum	75
108	2021-02	Northwest	norfli	73
109	2021-02	Northwest	pinsis	66
110	2021-02	Southeast	norcar	331
111	2021-02	Southeast	carwre	246
112	2021-02	Southeast	houfin	236
113	2021-02	Southeast	amegfi	214
114	2021-02	Southeast	blujay	205
115	2021-02	Southwest	houfin	86
116	2021-02	Southwest	daejun	79
117	2021-02	Southwest	whcspa	67
118	2021-02	Southwest	moudov	54
119	2021-02	Southwest	houspa	53
120	2021-03	Central	dowwoo	228
121	2021-03	Central	houfin	228
122	2021-03	Central	norcar	224
123	2021-03	Central	daejun	210
124	2021-03	Central	bkcchi	198
125	2021-03	Far North	rebnut	10
126	2021-03	Far North	bkcchi	6
127	2021-03	Far North	bkbmag1	4
128	2021-03	Far North	dowwoo	4
129	2021-03	Far North	pingro	3
130	2021-03	Northeast	norcar	478
131	2021-03	Northeast	dowwoo	445

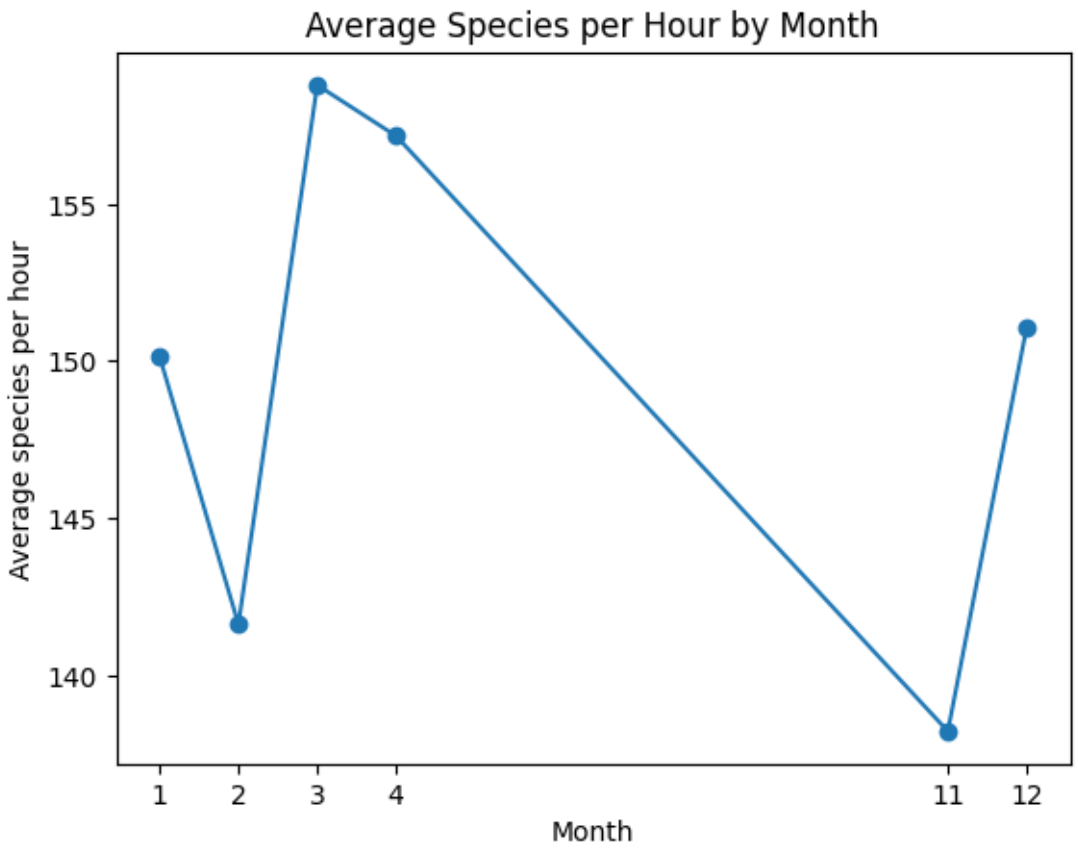
132	2021-03	Northeast	bkcchi	438
133	2021-03	Northeast	moudov	428
134	2021-03	Northeast	daejun	427
135	2021-03	Northwest	daejun	83
136	2021-03	Northwest	houfin	66
137	2021-03	Northwest	bkcchi	65
138	2021-03	Northwest	spotow	51
139	2021-03	Northwest	stejay	51
140	2021-03	Southeast	norcar	270
141	2021-03	Southeast	carwre	243
142	2021-03	Southeast	moudov	228
143	2021-03	Southeast	carchi	220
144	2021-03	Southeast	houfin	213
145	2021-03	Southwest	houfin	95
146	2021-03	Southwest	daejun	80
147	2021-03	Southwest	whcspa	69
148	2021-03	Southwest	lesgol	64
149	2021-03	Southwest	eucdov	53
150	2021-04	Central	dowwoo	158
151	2021-04	Central	norcar	151
152	2021-04	Central	houfin	148
153	2021-04	Central	moudov	146
154	2021-04	Central	houspa	140
155	2021-04	Far North	rebnut	4
156	2021-04	Far North	comrav	3
157	2021-04	Far North	borchi2	2
158	2021-04	Far North	daejun	2
159	2021-04	Far North	amerob	1
160	2021-04	Northeast	moudov	345
161	2021-04	Northeast	norcar	336
162	2021-04	Northeast	bkcchi	307
163	2021-04	Northeast	dowwoo	297
164	2021-04	Northeast	amegfi	280
165	2021-04	Northwest	daejun	65
166	2021-04	Northwest	pinsis	59
167	2021-04	Northwest	bkcchi	56
168	2021-04	Northwest	spotow	39
169	2021-04	Northwest	amerob	38
170	2021-04	Southeast	norcar	207
171	2021-04	Southeast	moudov	154
172	2021-04	Southeast	tuftit	143
173	2021-04	Southeast	rebwoo	134
174	2021-04	Southeast	blujay	125
175	2021-04	Southwest	houfin	82
176	2021-04	Southwest	moudov	43
177	2021-04	Southwest	eucdov	41
178	2021-04	Southwest	houspa	38
179	2021-04	Southwest	cowscj1	34

##(7)The impact of effort bias on the results

[50]:
```
import pandas as pd
import matplotlib.pyplot as plt

# 1) Load data
df = pd.read_csv("/content/PFW_2021_public.csv")

# 2) Keep valid observations + avoid division by 0
df = df[(df["valid"] == 1) & (df["effort_hrs_atleast"] > 0)]

# 3) Aggregate to checklist level (sub_id)
# Each sub_id = one checklist (one survey session)
checklist = (
    df.groupby("sub_id")
      .agg(
          total_birds=("how_many", "sum"),          # total birds seen in the␣
 ↪checklist
          n_species=("species_code", "nunique"),    # number of unique species␣
 ↪in the checklist
          effort_hours=("effort_hrs_atleast", "first"),
          Month=("Month", "first")
      )
      .reset_index()
)

# 4) Standardize by effort (control effort bias)
checklist["birds_per_hour"] = checklist["total_birds"] /␣
 ↪checklist["effort_hours"]
checklist["species_per_hour"] = checklist["n_species"] /␣
 ↪checklist["effort_hours"]

# 5) Monthly mean trends
monthly = (
    checklist.groupby("Month")[["birds_per_hour", "species_per_hour"]]
             .mean()
             .reset_index()
             .sort_values("Month")
)

# 6) Plot 1: Average birds per hour by month
plt.figure()
plt.plot(monthly["Month"], monthly["birds_per_hour"], marker="o")
plt.xlabel("Month")
plt.ylabel("Average birds per hour")
plt.title("Average Birds per Hour by Month")
plt.xticks(sorted(monthly["Month"].unique()))
```

```
plt.show()

# 7) Plot 2: Average species per hour by month
plt.figure()
plt.plot(monthly["Month"], monthly["species_per_hour"], marker="o")
plt.xlabel("Month")
plt.ylabel("Average species per hour")
plt.title("Average Species per Hour by Month")
plt.xticks(sorted(monthly["Month"].unique()))
plt.show()
```





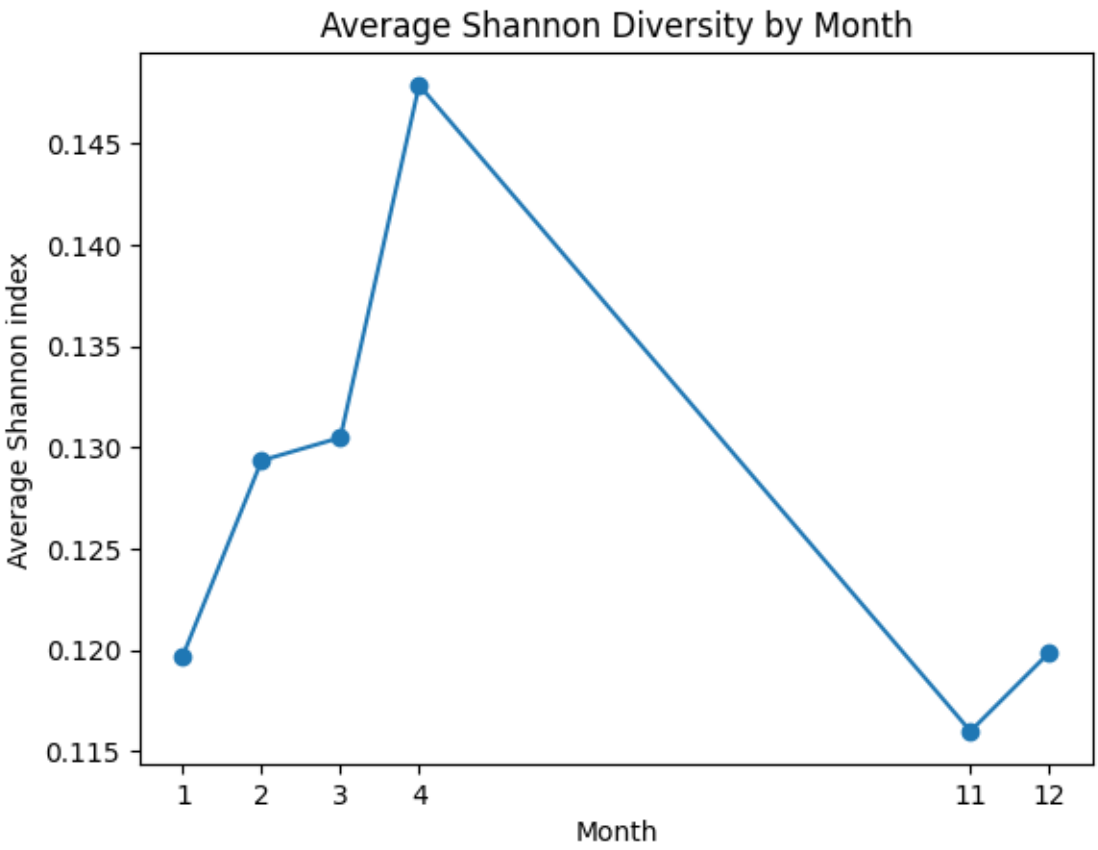
To control for observation effort, we normalize both bird counts and species richness by survey duration and analyze birds per hour and species per hour instead of raw counts.

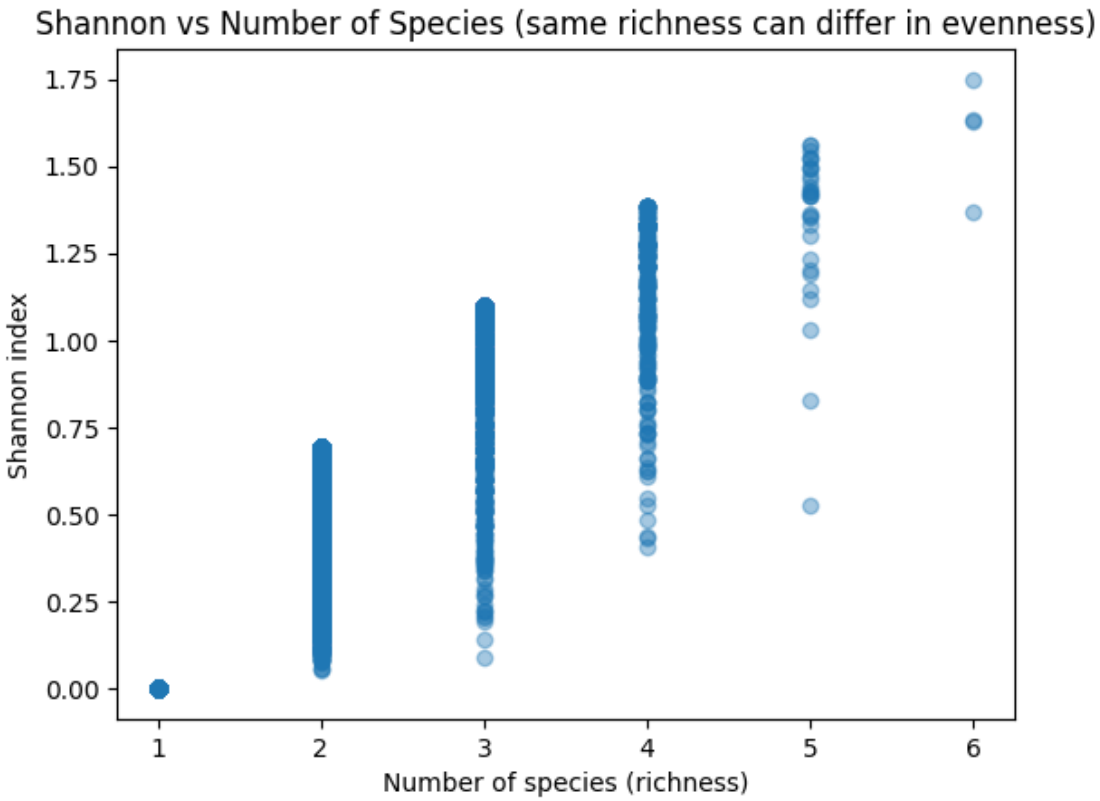
##(8)Species richness vs. diversity (Shannon index)

[52]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


# 1) Load & basic cleaning
df = pd.read_csv("/content/PFW_2021_public.csv")

# Keep valid + remove zero/negative effort if you later want "per hour" metrics
df = df[(df["valid"] == 1)].copy()

# how_many might have missing; treat missing as 0 (or drop)
df["how_many"] = pd.to_numeric(df["how_many"], errors="coerce").fillna(0)

# 2) Build species counts per checklist
#    Each row: (sub_id, species_code) total individuals in that checklist
```

```
sp_counts = (
    df.groupby(["sub_id", "species_code"], as_index=False)
      .agg(species_count=("how_many", "sum"),
           Month=("Month", "first"),
           effort_hours=("effort_hrs_atleast", "first"))
)

# total individuals per checklist
totals = (
    sp_counts.groupby("sub_id", as_index=False)
             .agg(total_birds=("species_count", "sum"))
)

sp_counts = sp_counts.merge(totals, on="sub_id", how="left")

# avoid divide-by-zero: drop checklists with 0 total birds
sp_counts = sp_counts[sp_counts["total_birds"] > 0].copy()

# p_i = species proportion within checklist
sp_counts["p_i"] = sp_counts["species_count"] / sp_counts["total_birds"]


# 3) Shannon & Simpson per checklist
#    Shannon: -Σ p_i * ln(p_i)
#    Simpson (Gini-Simpson): 1 - Σ p_i^2

diversity = (
    sp_counts.groupby("sub_id", as_index=False)
             .agg(
                 Month=("Month", "first"),
                 effort_hours=("effort_hours", "first"),
                 n_species=("species_code", "nunique"),
                 shannon=("p_i", lambda x: -np.sum(x * np.log(x))),
                 simpson=("p_i", lambda x: 1 - np.sum(x**2)),
                 total_birds=("total_birds", "first")
             )
)

# (Optional) If you also want effort-standardized richness
diversity = diversity[(diversity["effort_hours"] > 0)].copy()
diversity["species_per_hour"] = diversity["n_species"] /␣
 ↪diversity["effort_hours"]
diversity["birds_per_hour"] = diversity["total_birds"] /␣
 ↪diversity["effort_hours"]

# 4) Visualization A: Monthly mean trends
```

```
monthly = (
    diversity.groupby("Month")[["shannon", "simpson", "n_species"]]
             .mean()
             .reset_index()
             .sort_values("Month")
)

plt.figure()
plt.plot(monthly["Month"], monthly["shannon"], marker="o")
plt.xlabel("Month")
plt.ylabel("Average Shannon index")
plt.title("Average Shannon Diversity by Month")
plt.xticks(sorted(monthly["Month"].unique()))
plt.show()



# 5) Visualization B: Why it matters
#    Same n_species can have different Shannon

plt.figure()
plt.scatter(diversity["n_species"], diversity["shannon"], alpha=0.4)
plt.xlabel("Number of species (richness)")
plt.ylabel("Shannon index")
plt.title("Shannon vs Number of Species (same richness can differ in evenness)")
plt.show()
```


Average Shannon Diversity by Month
Average Shannon index
0.145
0.140
0.135
0.130
0.125
0.120
0.115
1
2
3
4
11
12
Month



For the same species richness, the Shannon index varies substantially, indicating that community evenness differs across checklists.

#3.EDA for PFW_count_site_data_public_2021

##(1)Description of the dataset about PFW_count_site_data_public_2021.csv

###Description The dataset describes the static environmental and facility characteristics of bird observation sites, with a core focus on habitat types and human management practices at survey locations. It details each site's courtyard type, surrounding habitat composition (such as woodland, grassland, and water bodies), vegetation and water availability, feeder types and quantities, alongside factors potentially influencing bird activity (such as the frequency of cats, dogs, and squirrels) and feeding habits. This dataset comprises site-level attribute data, serving to contextualise the environmental and anthropogenic factors underlying bird observation results.

##(2)Data structure check

[24]:
```
import pandas as pd
df = pd.read_csv("/content/PFW_count_site_data_public_2021.csv")
print("Rows, Columns:", df.shape)
```

```
Rows, Columns: (254355, 62)
```

[25]:
```
df.dtypes
```

[25]:
```
loc_id                   object
proj_period_id           object
yard_type_pavement      float64
yard_type_garden        float64
yard_type_landsca       float64
yard_type_woods         float64
yard_type_desert        float64
hab_dcid_woods          float64
hab_evgr_woods          float64
hab_mixed_woods         float64
hab_orchard             float64
hab_park                float64
hab_water_fresh         float64
hab_water_salt          float64
hab_residential         float64
hab_industrial          float64
hab_agricultural        float64
hab_desert_scrub        float64
hab_young_woods         float64
hab_swamp               float64
hab_marsh               float64
evgr_trees_atleast      float64
evgr_shrbs_atleast      float64
dcid_trees_atleast      float64
dcid_shrbs_atleast      float64
fru_trees_atleast       float64
cacti_atleast           float64
brsh_piles_atleast      float64
water_srcs_atleast      float64
bird_baths_atleast      float64
nearby_feeders          float64
squirrels               float64
cats                    float64
dogs                    float64
humans                  float64
housing_density         float64
fed_yr_round            float64
fed_in_jan              float64
fed_in_feb              float64
fed_in_mar              float64
fed_in_apr              float64
fed_in_may              float64
fed_in_jun              float64
fed_in_jul              float64
fed_in_aug              float64
fed_in_sep              float64
fed_in_oct              float64
```

```
fed_in_nov                      float64
fed_in_dec                      float64
numfeeders_suet                 float64
numfeeders_ground               float64
numfeeders_hanging              float64
numfeeders_platfrm              float64
numfeeders_humming              float64
numfeeders_water                float64
numfeeders_thistle              float64
numfeeders_fruit                float64
numfeeders_hopper               float64
numfeeders_tube                 float64
numfeeders_other                float64
population_atleast              float64
count_area_size_sq_m_atleast    float64
dtype: object
```

[26]:
```python
import pandas as pd

df = pd.read_csv("/content/PFW_count_site_data_public_2021.csv")

# Show unique values (and counts) for each column
for col in df.columns:
    print("=" * 60)
    print(f"Column: {col}")
    print("Number of unique values:", df[col].nunique(dropna=True))

    # Show value counts (including NaN)
    print(df[col].value_counts(dropna=False))
```

```
============================================================
Column: loc_id
Number of unique values: 91105
loc_id
L36031       33
L33888       33
L22111       33
L33282       32
L33190       32
             ..
L34380        1
L34378        1
L34376        1
L3437550      1
L13437971     1
Name: count, Length: 91105, dtype: int64
============================================================
```

```
Column: proj_period_id
Number of unique values: 33
proj_period_id
PFW_2021    13222
PFW_2020    11262
PFW_2015    10516
PFW_2016    10484
PFW_2017    10346
PFW_2019    10320
PFW_2018    10257
PFW_2014    10193
PFW_2003     9496
PFW_2002     9342
PFW_2001     9198
PFW_2004     9173
PFW_2005     9120
PFW_2000     8651
PFW_2013     8357
PFW_1999     7199
PFW_2012     7001
PFW_2011     6930
PFW_2006     6688
PFW_2008     6669
PFW_1998     6605
PFW_2007     6500
PFW_2010     6221
PFW_2009     6164
PFW_1995     5917
PFW_1997     5669
PFW_1996     5470
PFW_1990     5013
PFW_1994     4760
PFW_1993     4543
PFW_1989     4492
PFW_1992     4320
PFW_1991     4257
Name: count, dtype: int64
============================================================
Column: yard_type_pavement
Number of unique values: 2
yard_type_pavement
0.0    185146
NaN     68877
1.0       332
Name: count, dtype: int64
============================================================
Column: yard_type_garden
Number of unique values: 2
```

```
yard_type_garden
0.0    178433
NaN     68282
1.0      7640
Name: count, dtype: int64
============================================================
Column: yard_type_landsca
Number of unique values: 2
yard_type_landsca
1.0    156716
NaN     63017
0.0     34622
Name: count, dtype: int64
============================================================
Column: yard_type_woods
Number of unique values: 2
yard_type_woods
1.0    113397
0.0     73330
NaN     67628
Name: count, dtype: int64
============================================================
Column: yard_type_desert
Number of unique values: 2
yard_type_desert
0.0    184109
NaN     68782
1.0      1464
Name: count, dtype: int64
============================================================
Column: hab_dcid_woods
Number of unique values: 2
hab_dcid_woods
0.0    102271
1.0    101970
NaN     50114
Name: count, dtype: int64
============================================================
Column: hab_evgr_woods
Number of unique values: 2
hab_evgr_woods
0.0    148782
NaN     61690
1.0     43883
Name: count, dtype: int64
============================================================
Column: hab_mixed_woods
Number of unique values: 2
```

```
hab_mixed_woods
1.0    139497
0.0     72454
NaN     42404
Name: count, dtype: int64
============================================================
Column: hab_orchard
Number of unique values: 2
hab_orchard
0.0    157880
NaN     79599
1.0     16876
Name: count, dtype: int64
============================================================
Column: hab_park
Number of unique values: 2
hab_park
NaN    116534
0.0     75252
1.0     62569
Name: count, dtype: int64
============================================================
Column: hab_water_fresh
Number of unique values: 2
hab_water_fresh
1.0    126263
0.0     80833
NaN     47259
Name: count, dtype: int64
============================================================
Column: hab_water_salt
Number of unique values: 2
hab_water_salt
0.0    168232
NaN     77087
1.0      9036
Name: count, dtype: int64
============================================================
Column: hab_residential
Number of unique values: 2
hab_residential
1.0    188337
NaN     38110
0.0     27908
Name: count, dtype: int64
============================================================
Column: hab_industrial
Number of unique values: 2
```

```
hab_industrial
0.0    149401
NaN     61649
1.0     43305
Name: count, dtype: int64
============================================================
Column: hab_agricultural
Number of unique values: 2
hab_agricultural
0.0    102198
NaN     78850
1.0     73307
Name: count, dtype: int64
============================================================
Column: hab_desert_scrub
Number of unique values: 2
hab_desert_scrub
0.0    160631
NaN     77001
1.0     16723
Name: count, dtype: int64
============================================================
Column: hab_young_woods
Number of unique values: 2
hab_young_woods
0.0    111130
NaN     82494
1.0     60731
Name: count, dtype: int64
============================================================
Column: hab_swamp
Number of unique values: 2
hab_swamp
NaN    115086
0.0     98318
1.0     40951
Name: count, dtype: int64
============================================================
Column: hab_marsh
Number of unique values: 2
hab_marsh
0.0    134140
NaN     91140
1.0     29075
Name: count, dtype: int64
============================================================
Column: evgr_trees_atleast
Number of unique values: 5
```

```
evgr_trees_atleast
4.0     65586
1.0     64503
11.0    47960
NaN     29181
0.0     26482
3.0     20643
Name: count, dtype: int64
============================================================
Column: evgr_shrbs_atleast
Number of unique values: 5
evgr_shrbs_atleast
4.0     62251
1.0     58883
11.0    40571
0.0     40089
NaN     33592
3.0     18969
Name: count, dtype: int64
============================================================
Column: dcid_trees_atleast
Number of unique values: 5
dcid_trees_atleast
11.0    80081
4.0     73134
1.0     40674
NaN     27709
3.0     23886
0.0      8871
Name: count, dtype: int64
============================================================
Column: dcid_shrbs_atleast
Number of unique values: 5
dcid_shrbs_atleast
11.0    67245
4.0     66882
1.0     46353
NaN     32109
0.0     20997
3.0     20769
Name: count, dtype: int64
============================================================
Column: fru_trees_atleast
Number of unique values: 5
fru_trees_atleast
1.0     79452
NaN     56583
4.0     50128
```

```
0.0     48973
11.0    17113
3.0      2106
Name: count, dtype: int64
============================================================
Column: cacti_atleast
Number of unique values: 5
cacti_atleast
0.0     185240
NaN      56066
1.0       7921
4.0       3051
11.0      1833
3.0        244
Name: count, dtype: int64
============================================================
Column: brsh_piles_atleast
Number of unique values: 5
brsh_piles_atleast
1.0     109465
0.0      75401
NaN      48434
4.0      15521
11.0      3473
3.0       2061
Name: count, dtype: int64
============================================================
Column: water_srcs_atleast
Number of unique values: 4
water_srcs_atleast
0.0     127107
NaN      64915
1.0      59846
4.0       1307
11.0      1180
Name: count, dtype: int64
============================================================
Column: bird_baths_atleast
Number of unique values: 4
bird_baths_atleast
1.0     138813
NaN      58343
0.0      49776
4.0       6586
11.0       837
Name: count, dtype: int64
============================================================
Column: nearby_feeders
```

```
Number of unique values: 2
nearby_feeders
0.0    126739
1.0     98130
NaN     29486
Name: count, dtype: int64
============================================================
Column: squirrels
Number of unique values: 2
squirrels
1.0    190362
0.0     45323
NaN     18670
Name: count, dtype: int64
============================================================
Column: cats
Number of unique values: 2
cats
1.0    120345
0.0     98507
NaN     35503
Name: count, dtype: int64
============================================================
Column: dogs
Number of unique values: 2
dogs
1.0    117754
0.0     99215
NaN     37386
Name: count, dtype: int64
============================================================
Column: humans
Number of unique values: 2
humans
1.0    185365
0.0     41883
NaN     27107
Name: count, dtype: int64
============================================================
Column: housing_density
Number of unique values: 4
housing_density
3.0    77098
2.0    70068
1.0    67977
NaN    23893
4.0    15319
Name: count, dtype: int64
```

```
============================================================
Column: fed_yr_round
Number of unique values: 2
fed_yr_round
NaN    173165
1.0     63463
0.0     17727
Name: count, dtype: int64
============================================================
Column: fed_in_jan
Number of unique values: 2
fed_in_jan
1.0    182951
NaN     57691
0.0     13713
Name: count, dtype: int64
============================================================
Column: fed_in_feb
Number of unique values: 2
fed_in_feb
1.0    182534
NaN     57788
0.0     14033
Name: count, dtype: int64
============================================================
Column: fed_in_mar
Number of unique values: 2
fed_in_mar
1.0    181902
NaN     57856
0.0     14597
Name: count, dtype: int64
============================================================
Column: fed_in_apr
Number of unique values: 2
fed_in_apr
1.0    173608
NaN     59035
0.0     21712
Name: count, dtype: int64
============================================================
Column: fed_in_may
Number of unique values: 2
fed_in_may
1.0    147446
NaN     62200
0.0     44709
Name: count, dtype: int64
```

```
============================================================
Column: fed_in_jun
Number of unique values: 2
fed_in_jun
1.0    133787
NaN     64002
0.0     56566
Name: count, dtype: int64
============================================================
Column: fed_in_jul
Number of unique values: 2
fed_in_jul
1.0    128508
NaN     64739
0.0     61108
Name: count, dtype: int64
============================================================
Column: fed_in_aug
Number of unique values: 2
fed_in_aug
1.0    129258
NaN     64669
0.0     60428
Name: count, dtype: int64
============================================================
Column: fed_in_sep
Number of unique values: 2
fed_in_sep
1.0    142574
NaN     62945
0.0     48836
Name: count, dtype: int64
============================================================
Column: fed_in_oct
Number of unique values: 2
fed_in_oct
1.0    164050
NaN     60159
0.0     30146
Name: count, dtype: int64
============================================================
Column: fed_in_nov
Number of unique values: 2
fed_in_nov
1.0    182886
NaN     57438
0.0     14031
Name: count, dtype: int64
```

```
============================================================
Column: fed_in_dec
Number of unique values: 2
fed_in_dec
1.0    182980
NaN     57613
0.0     13762
Name: count, dtype: int64
============================================================
Column: numfeeders_suet
Number of unique values: 37
numfeeders_suet
1.0        104526
2.0         64283
0.0         25758
3.0         22933
NaN         21627
4.0          9137
5.0          3223
6.0          1933
7.0           354
8.0           264
10.0          102
9.0            85
11.0           41
12.0           33
13.0           10
14.0           10
21.0            5
15.0            4
20.0            3
33.0            2
30.0            2
16.0            2
23.0            2
25.0            2
34.0            1
32.0            1
28.0            1
117.0           1
31.0            1
45.0            1
14850.0         1
17.0            1
22.0            1
54.0            1
24.0            1
48.0            1
```

```
19.0            1
64.0            1
Name: count, dtype: int64
============================================================
Column: numfeeders_ground
Number of unique values: 34
numfeeders_ground
1.0     86178
0.0     58067
NaN     41908
2.0     40861
3.0     16830
4.0      5828
5.0      2233
6.0      1828
7.0       202
8.0       173
10.0       80
9.0        44
11.0       32
12.0       31
20.0       10
14.0        9
15.0        9
16.0        7
13.0        5
30.0        3
18.0        2
22.0        2
28.0        1
19.0        1
40.0        1
33.0        1
75.0        1
76.0        1
17.0        1
67.0        1
63.0        1
32.0        1
21.0        1
23.0        1
42.0        1
Name: count, dtype: int64
============================================================
Column: numfeeders_hanging
Number of unique values: 32
numfeeders_hanging
NaN     150162
```

```
2.0      26154
1.0      20231
3.0      20169
4.0      12553
0.0       8559
6.0       8058
5.0       6844
7.0        578
8.0        402
10.0       177
9.0        163
12.0        68
11.0        51
14.0        45
15.0        42
13.0        24
20.0        18
16.0        16
17.0         9
22.0         5
19.0         4
23.0         3
18.0         3
21.0         3
25.0         3
40.0         3
30.0         2
55.0         2
27.0         1
29.0         1
28.0         1
45.0         1
Name: count, dtype: int64
============================================================
Column: numfeeders_platfrm
Number of unique values: 35
numfeeders_platfrm
1.0      85001
0.0      73831
NaN      45545
2.0      32275
3.0      10824
4.0       3940
5.0       1396
6.0       1045
7.0        190
8.0        106
9.0         56
```

```
10.0        49
11.0        22
12.0        17
20.0        10
14.0         8
19.0         6
13.0         4
23.0         4
18.0         3
16.0         3
17.0         2
24.0         2
21.0         2
15.0         2
25.0         2
28.0         1
31.0         1
22.0         1
48.0         1
111.0        1
27.0         1
32.0         1
62.0         1
100.0        1
65.0         1
Name: count, dtype: int64
============================================================
Column: numfeeders_humming
Number of unique values: 30
numfeeders_humming
0.0     138311
NaN      62393
1.0      29109
2.0      15240
3.0       5210
4.0       2225
5.0        862
6.0        580
7.0        142
8.0        112
10.0        55
12.0        35
9.0         25
11.0        18
15.0         6
13.0         4
20.0         4
30.0         3
```

```
32.0         3
19.0         3
21.0         3
22.0         2
25.0         2
26.0         1
40.0         1
16.0         1
24.0         1
33.0         1
18.0         1
14.0         1
28.0         1
Name: count, dtype: int64
============================================================
Column: numfeeders_water
Number of unique values: 16
numfeeders_water
NaN     155125
0.0      46780
1.0      36744
2.0      10180
3.0       3416
4.0       1171
5.0        459
6.0        420
8.0         21
7.0         13
10.0         7
12.0         6
11.0         4
13.0         3
9.0          3
16.0         2
20.0         1
Name: count, dtype: int64
============================================================
Column: numfeeders_thistle
Number of unique values: 22
numfeeders_thistle
NaN     179792
1.0      37189
0.0      22191
2.0      10966
3.0       2703
4.0        852
5.0        300
6.0        255
```

```
7.0         34
8.0         29
12.0        12
11.0         8
9.0          7
10.0         6
18.0         2
22.0         2
21.0         1
32.0         1
16.0         1
50.0         1
20.0         1
17.0         1
14.0         1
Name: count, dtype: int64
============================================================
Column: numfeeders_fruit
Number of unique values: 28
numfeeders_fruit
0.0      137869
NaN       98995
1.0       14776
2.0        2015
3.0         375
4.0         176
6.0          41
5.0          39
10.0         14
8.0          14
12.0         13
9.0           3
11.0          3
18.0          3
7.0           3
60.0          2
30.0          2
24.0          1
56.0          1
13.0          1
19.0          1
27.0          1
15.0          1
40.0          1
82.0          1
20.0          1
112.0         1
75.0          1
```

```
87.0          1
Name: count, dtype: int64
============================================================
Column: numfeeders_hopper
Number of unique values: 31
numfeeders_hopper
NaN      136248
1.0       45839
0.0       30364
2.0       24078
3.0        9807
4.0        4301
5.0        1850
6.0         872
7.0         340
8.0         287
10.0        106
9.0          93
12.0         55
11.0         43
13.0         18
14.0         16
15.0         10
16.0          7
18.0          4
17.0          3
21.0          2
43.0          2
23.0          1
32.0          1
30.0          1
44.0          1
200.0         1
211.0         1
24.0          1
35.0          1
31.0          1
25.0          1
Name: count, dtype: int64
============================================================
Column: numfeeders_tube
Number of unique values: 42
numfeeders_tube
NaN      124165
1.0       41599
2.0       36403
3.0       20216
0.0       11448
```

```
4.0       10465
5.0        4575
6.0        2484
7.0        1079
8.0         769
9.0         372
10.0        316
12.0        134
11.0         99
15.0         56
13.0         38
14.0         37
18.0         19
16.0         15
20.0         12
17.0          9
28.0          5
22.0          5
21.0          4
19.0          4
32.0          3
41.0          3
31.0          3
24.0          2
23.0          2
25.0          2
66.0          1
50.0          1
61.0          1
62.0          1
51.0          1
104.0         1
30.0          1
35.0          1
43.0          1
42.0          1
26.0          1
44.0          1
Name: count, dtype: int64
============================================================
Column: numfeeders_other
Number of unique values: 30
numfeeders_other
NaN     162542
0.0      58417
1.0      21069
2.0       7572
3.0       2583
```

```
4.0       1072
5.0        437
6.0        276
7.0        102
8.0        100
10.0        61
9.0         41
11.0        18
12.0        17
14.0        11
15.0         8
16.0         6
20.0         5
18.0         4
13.0         3
27.0         1
33.0         1
24.0         1
17.0         1
19.0         1
25.0         1
99.0         1
40.0         1
98.0         1
32.0         1
58.0         1
Name: count, dtype: int64
============================================================
Column: population_atleast
Number of unique values: 5
population_atleast
1.0         70941
5001.0      62911
25001.0     50869
NaN         29384
100001.0    25778
100000.0    14472
Name: count, dtype: int64
============================================================
Column: count_area_size_sq_m_atleast
Number of unique values: 4
count_area_size_sq_m_atleast
100.01    106987
1.01       65072
375.01     48712
NaN        26622
0.01        6962
Name: count, dtype: int64
```

##(3)Data quality check and preprocessing

Group the columns

[27]:
```
import pandas as pd

df = pd.read_csv("/content/PFW_count_site_data_public_2021.csv")

# Manually categorize each variable

#  Feeders identifiers
id_cols = ["loc_id", "proj_period_id"]

#  Yard / site type variables
yard_cols = [c for c in df.columns if c.startswith("yard_type_")]

#  Habitat type variables
hab_cols = [c for c in df.columns if c.startswith("hab_")]

#  Vegetation and environmental count variables (minimum counts, *_atleast)
environment_cols = [c for c in df.columns if c.endswith("_atleast")] + [
    "nearby_feeders", "housing_density"
]

#  Disturbance / human and animal activity variables
disturbance_cols = ["squirrels", "cats", "dogs", "humans"]

#  Feeding behavior over time (year-round and monthly indicators)
feeding_time_cols = ["fed_yr_round"] + [
    c for c in df.columns if c.startswith("fed_in_")
]

#  Feeder type and quantity variables
feeder_cols = [c for c in df.columns if c.startswith("numfeeders_")]


# Store all variable groups in a dictionary for convenient EDA
feature_groups = {
    "identifiers": id_cols,
    "yard_type": yard_cols,
    "habitat": hab_cols,
    "environment": environment_cols,
    "disturbance": disturbance_cols,
    "feeding_time": feeding_time_cols,
    "num_diff_types_feeder": feeder_cols
}

# Quick check: number of variables in each group
```

```
for k, v in feature_groups.items():
    print(k, len(v))
```

```
identifiers 2
yard_type 5
habitat 14
environment 13
disturbance 4
feeding_time 13
num_diff_types_feeder 11
```

[28]:
```
print("identifiers", id_cols)
print("yard:", yard_cols)
print("hab:", hab_cols)
print("environment:", environment_cols)
print("disturbance:", disturbance_cols)
print("feeding time:", feeding_time_cols)
print("num_diff_types_feeder:", feeder_cols)
```

```
identifiers ['loc_id', 'proj_period_id']
yard: ['yard_type_pavement', 'yard_type_garden', 'yard_type_landsca',
'yard_type_woods', 'yard_type_desert']
hab: ['hab_dcid_woods', 'hab_evgr_woods', 'hab_mixed_woods', 'hab_orchard',
'hab_park', 'hab_water_fresh', 'hab_water_salt', 'hab_residential',
'hab_industrial', 'hab_agricultural', 'hab_desert_scrub', 'hab_young_woods',
'hab_swamp', 'hab_marsh']
environment: ['evgr_trees_atleast', 'evgr_shrbs_atleast', 'dcid_trees_atleast',
'dcid_shrbs_atleast', 'fru_trees_atleast', 'cacti_atleast',
'brsh_piles_atleast', 'water_srcs_atleast', 'bird_baths_atleast',
'population_atleast', 'count_area_size_sq_m_atleast', 'nearby_feeders',
'housing_density']
disturbance: ['squirrels', 'cats', 'dogs', 'humans']
feeding time: ['fed_yr_round', 'fed_in_jan', 'fed_in_feb', 'fed_in_mar',
'fed_in_apr', 'fed_in_may', 'fed_in_jun', 'fed_in_jul', 'fed_in_aug',
'fed_in_sep', 'fed_in_oct', 'fed_in_nov', 'fed_in_dec']
num_diff_types_feeder: ['numfeeders_suet', 'numfeeders_ground',
'numfeeders_hanging', 'numfeeders_platfrm', 'numfeeders_humming',
'numfeeders_water', 'numfeeders_thistle', 'numfeeders_fruit',
'numfeeders_hopper', 'numfeeders_tube', 'numfeeders_other']
```

Check missing value rate by group

[29]:
```
# Missing value summary by variable group
group_missing_summary = {}

for group_name, cols in feature_groups.items():
    sub_df = df[cols]
```

```
    group_missing_summary[group_name] = pd.DataFrame({
        "missing_count": sub_df.isna().sum(),
        "missing_rate": (sub_df.isna().sum() / len(df)).round(4)
    }).sort_values("missing_rate", ascending=False)

# Print missing values for each group
for group_name, miss_df in group_missing_summary.items():
    print("=" * 80)
    print(f"Group: {group_name}")
    display(miss_df)
```

```
================================================================================
Group: identifiers
```

	missing_count	missing_rate
loc_id	0	0.0
proj_period_id	0	0.0

```
================================================================================
Group: yard_type
```

	missing_count	missing_rate
yard_type_pavement	68877	0.2708
yard_type_desert	68782	0.2704
yard_type_garden	68282	0.2685
yard_type_woods	67628	0.2659
yard_type_landsca	63017	0.2478

```
================================================================================
Group: habitat
```

	missing_count	missing_rate
hab_park	116534	0.4582
hab_swamp	115086	0.4525
hab_marsh	91140	0.3583
hab_young_woods	82494	0.3243
hab_orchard	79599	0.3129
hab_agricultural	78850	0.3100
hab_water_salt	77087	0.3031
hab_desert_scrub	77001	0.3027
hab_evgr_woods	61690	0.2425
hab_industrial	61649	0.2424
hab_dcid_woods	50114	0.1970
hab_water_fresh	47259	0.1858
hab_mixed_woods	42404	0.1667
hab_residential	38110	0.1498

```
================================================================================
Group: environment
```

	missing_count	missing_rate

```
water_srcs_atleast                   64915        0.2552
bird_baths_atleast                   58343        0.2294
fru_trees_atleast                    56583        0.2225
cacti_atleast                        56066        0.2204
brsh_piles_atleast                   48434        0.1904
evgr_shrbs_atleast                   33592        0.1321
dcid_shrbs_atleast                   32109        0.1262
nearby_feeders                       29486        0.1159
population_atleast                   29384        0.1155
evgr_trees_atleast                   29181        0.1147
dcid_trees_atleast                   27709        0.1089
count_area_size_sq_m_atleast         26622        0.1047
housing_density                      23893        0.0939
```

```
================================================================================
Group: disturbance
```

```
           missing_count  missing_rate
dogs               37386        0.1470
cats               35503        0.1396
humans             27107        0.1066
squirrels          18670        0.0734
```

```
================================================================================
Group: feeding_time
```

```
              missing_count  missing_rate
fed_yr_round         173165        0.6808
fed_in_jul            64739        0.2545
fed_in_aug            64669        0.2542
fed_in_jun            64002        0.2516
fed_in_sep            62945        0.2475
fed_in_may            62200        0.2445
fed_in_oct            60159        0.2365
fed_in_apr            59035        0.2321
fed_in_mar            57856        0.2275
fed_in_feb            57788        0.2272
fed_in_jan            57691        0.2268
fed_in_dec            57613        0.2265
fed_in_nov            57438        0.2258
```

```
================================================================================
Group: num_diff_types_feeder
```

```
                    missing_count  missing_rate
numfeeders_thistle         179792        0.7069
numfeeders_other           162542        0.6390
numfeeders_water           155125        0.6099
numfeeders_hanging         150162        0.5904
numfeeders_hopper          136248        0.5357
numfeeders_tube            124165        0.4882
```

```
numfeeders_fruit            98995       0.3892
numfeeders_humming          62393       0.2453
numfeeders_platfrm          45545       0.1791
numfeeders_ground           41908       0.1648
numfeeders_suet             21627       0.0850
```

[30]:
```python
# Fixing missing value by group rules

# yard_type -> fill with 0
df[yard_cols] = df[yard_cols].fillna(0)

# habitat -> fill with 0
df[hab_cols] = df[hab_cols].fillna(0)

# environment -> fill with mode (column-wise)
for col in environment_cols:
    if df[col].isna().any():
        mode_val = df[col].mode(dropna=True)
        if len(mode_val) > 0:
            df[col] = df[col].fillna(mode_val[0])

# disturbance -> fill with mode
for col in disturbance_cols:
    if df[col].isna().any():
        mode_val = df[col].mode(dropna=True)
        if len(mode_val) > 0:
            df[col] = df[col].fillna(mode_val[0])

# feeding_time -> fill with 0
df[feeding_time_cols] = df[feeding_time_cols].fillna(0)

# num_diff_types_feeder -> fill with 0
df[feeder_cols] = df[feeder_cols].fillna(0)
```

[31]:
```python
# Double check the missing value summary by variable group
group_missing_summary = {}

for group_name, cols in feature_groups.items():
    sub_df = df[cols]

    group_missing_summary[group_name] = pd.DataFrame({
        "missing_count": sub_df.isna().sum(),
        "missing_rate": (sub_df.isna().sum() / len(df)).round(4)
    }).sort_values("missing_rate", ascending=False)

# Print missing values for each group
for group_name, miss_df in group_missing_summary.items():
```

```
    print("=" * 80)
    print(f"Group: {group_name}")
    display(miss_df)
```

```
================================================================================
Group: identifiers
```

	missing_count	missing_rate
loc_id	0	0.0
proj_period_id	0	0.0

```
================================================================================
Group: yard_type
```

	missing_count	missing_rate
yard_type_pavement	0	0.0
yard_type_garden	0	0.0
yard_type_landsca	0	0.0
yard_type_woods	0	0.0
yard_type_desert	0	0.0

```
================================================================================
Group: habitat
```

	missing_count	missing_rate
hab_dcid_woods	0	0.0
hab_evgr_woods	0	0.0
hab_mixed_woods	0	0.0
hab_orchard	0	0.0
hab_park	0	0.0
hab_water_fresh	0	0.0
hab_water_salt	0	0.0
hab_residential	0	0.0
hab_industrial	0	0.0
hab_agricultural	0	0.0
hab_desert_scrub	0	0.0
hab_young_woods	0	0.0
hab_swamp	0	0.0
hab_marsh	0	0.0

```
================================================================================
Group: environment
```

	missing_count	missing_rate
evgr_trees_atleast	0	0.0
evgr_shrbs_atleast	0	0.0
dcid_trees_atleast	0	0.0
dcid_shrbs_atleast	0	0.0
fru_trees_atleast	0	0.0
cacti_atleast	0	0.0
brsh_piles_atleast	0	0.0

```
water_srcs_atleast                        0          0.0
bird_baths_atleast                        0          0.0
population_atleast                        0          0.0
count_area_size_sq_m_atleast              0          0.0
nearby_feeders                            0          0.0
housing_density                           0          0.0
```

```
================================================================================
Group: disturbance
```

```
           missing_count  missing_rate
squirrels              0           0.0
cats                   0           0.0
dogs                   0           0.0
humans                 0           0.0
```

```
================================================================================
Group: feeding_time
```

```
              missing_count  missing_rate
fed_yr_round              0           0.0
fed_in_jan                0           0.0
fed_in_feb                0           0.0
fed_in_mar                0           0.0
fed_in_apr                0           0.0
fed_in_may                0           0.0
fed_in_jun                0           0.0
fed_in_jul                0           0.0
fed_in_aug                0           0.0
fed_in_sep                0           0.0
fed_in_oct                0           0.0
fed_in_nov                0           0.0
fed_in_dec                0           0.0
```

```
================================================================================
Group: num_diff_types_feeder
```

```
                    missing_count  missing_rate
numfeeders_suet                 0           0.0
numfeeders_ground               0           0.0
numfeeders_hanging              0           0.0
numfeeders_platfrm              0           0.0
numfeeders_humming              0           0.0
numfeeders_water                0           0.0
numfeeders_thistle              0           0.0
numfeeders_fruit                0           0.0
numfeeders_hopper               0           0.0
numfeeders_tube                 0           0.0
numfeeders_other                0           0.0
```

##(4)Group distribution exploration

[32]:
```
group_distribution = []

for group_name, cols in feature_groups.items():
    sub = df[cols]

    group_distribution.append({
        "group": group_name,
        "n_variables": len(cols),
        "avg_missing_rate": sub.isna().mean().mean(),
        "avg_zero_ratio": (sub == 0).mean().mean(),
        "avg_skewness": sub.skew(numeric_only=True).mean()
    })

group_distribution_df = pd.DataFrame(group_distribution)
group_distribution_df
```

[32]:

	group	n_variables	avg_missing_rate	avg_zero_ratio	avg_skewness
0	identifiers	2	0.0	0.000000	NaN
1	yard_type	5	0.0	0.780190	9.187830
2	habitat	14	0.0	0.732510	1.588023
3	environment	13	0.0	0.260128	2.757490
4	disturbance	4	0.0	0.280050	-1.100964
5	feeding_time	13	0.0	0.396378	-0.446492
6	num_diff_types_feeder	11	0.0	0.639799	61.318063

1) Yard type distribution

[33]:
```
print("yard:", yard_cols)
yard_summary = pd.DataFrame({
    "count_1": (df[yard_cols] == 1).sum(),
    "ratio_1_overall": (df[yard_cols] == 1).mean()
})

yard_summary
```

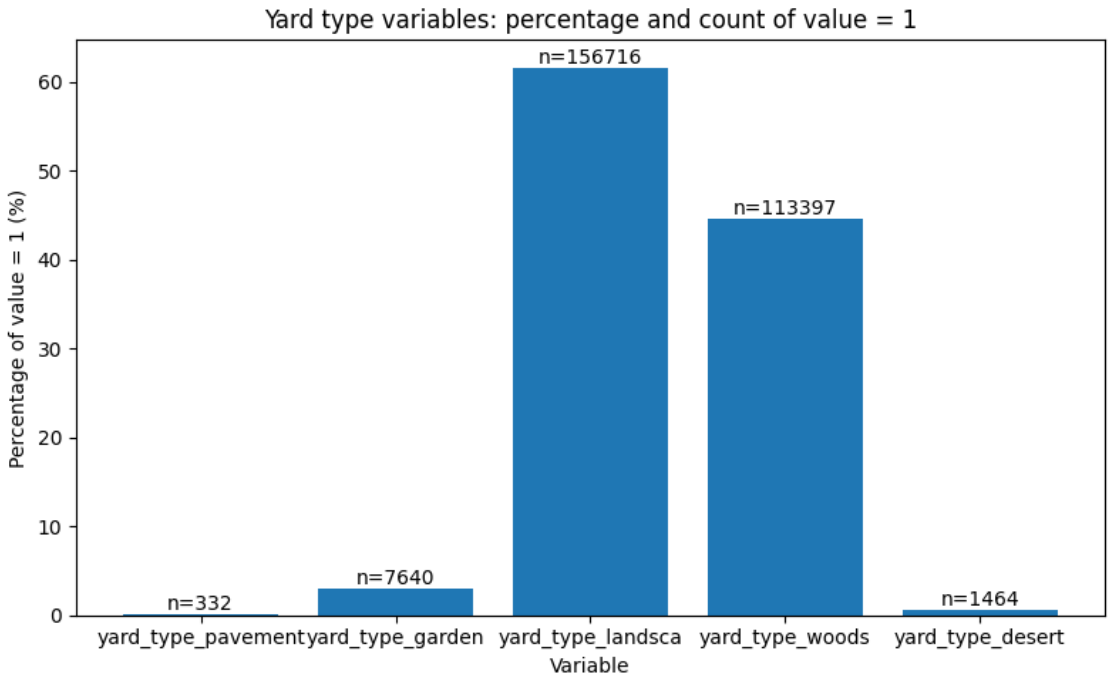
```
yard: ['yard_type_pavement', 'yard_type_garden', 'yard_type_landsca',
'yard_type_woods', 'yard_type_desert']
```

[33]:

	count_1	ratio_1_overall
yard_type_pavement	332	0.001305
yard_type_garden	7640	0.030037
yard_type_landsca	156716	0.616131
yard_type_woods	113397	0.445822
yard_type_desert	1464	0.005756

[34]:

```
import matplotlib.pyplot as plt
import pandas as pd

# Convert ratio to percentage
percent = yard_summary["ratio_1_overall"] * 100

# One single chart, using matplotlib only (no custom colors)
plt.figure(figsize=(8, 5))
bars = plt.bar(yard_summary.index, percent)

plt.xlabel("Variable")
plt.ylabel("Percentage of value = 1 (%)")
plt.title("Yard type variables: percentage and count of value = 1")

plt.xticks(rotation=0, ha="center")

# Annotate each bar with the corresponding count_1
for bar, cnt in zip(bars, yard_summary["count_1"]):
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        bar.get_height(),
        f"n={cnt}",
        ha="center",
        va="bottom"
    )

plt.tight_layout()
plt.show()
```



2.Habitite distribution

[35]:
```
print("hab:", hab_cols)
hab_summary = pd.DataFrame({
    "count_1": (df[hab_cols] == 1).sum(),
    "ratio_1_overall": (df[hab_cols] == 1).mean()
})

hab_summary
```

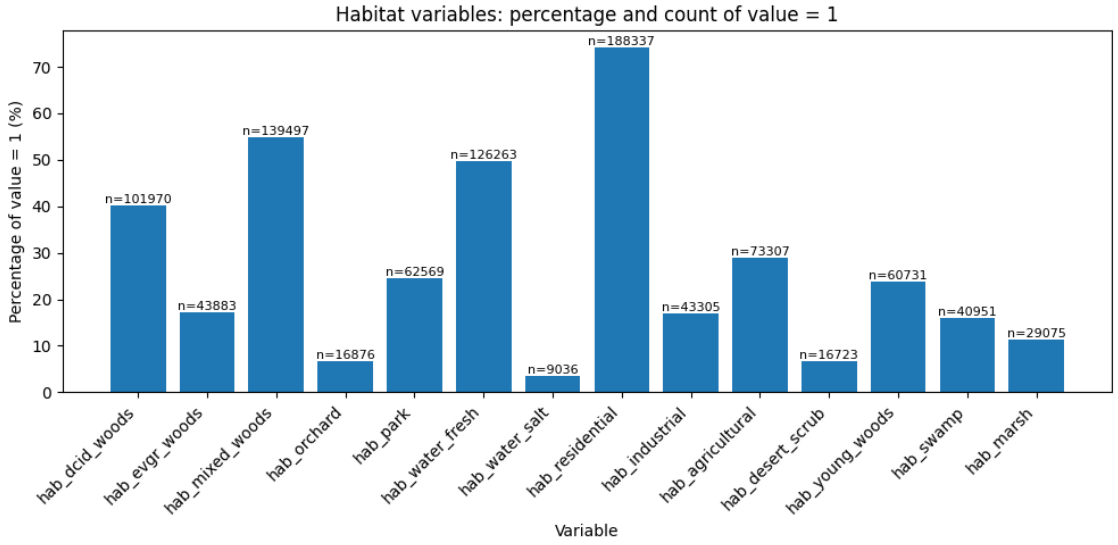
```
hab: ['hab_dcid_woods', 'hab_evgr_woods', 'hab_mixed_woods', 'hab_orchard',
'hab_park', 'hab_water_fresh', 'hab_water_salt', 'hab_residential',
'hab_industrial', 'hab_agricultural', 'hab_desert_scrub', 'hab_young_woods',
'hab_swamp', 'hab_marsh']
```

[35]:

	count_1	ratio_1_overall
hab_dcid_woods	101970	0.400896
hab_evgr_woods	43883	0.172527
hab_mixed_woods	139497	0.548434
hab_orchard	16876	0.066348
hab_park	62569	0.245991
hab_water_fresh	126263	0.496405
hab_water_salt	9036	0.035525
hab_residential	188337	0.740449
hab_industrial	43305	0.170254

```
hab_agricultural     73307          0.288207
hab_desert_scrub     16723          0.065747
hab_young_woods      60731          0.238765
hab_swamp            40951          0.160999
hab_marsh            29075          0.114309
```

[36]:
```python
import matplotlib.pyplot as plt

percent = hab_summary["ratio_1_overall"] * 100

plt.figure(figsize=(10, 5))
bars = plt.bar(hab_summary.index, percent)

plt.xlabel("Variable")
plt.ylabel("Percentage of value = 1 (%)")
plt.title("Habitat variables: percentage and count of value = 1")
plt.xticks(rotation=45, ha="right")

for bar, cnt in zip(bars, hab_summary["count_1"]):
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        bar.get_height(),
        f"n={cnt}",
        ha="center",
        va="bottom",
        fontsize=8
    )

plt.tight_layout()
plt.show()
```



3.Environmental variable distribution

[37]:
```
print("environment:", environment_cols)
for col in environment_cols:
    print("=" * 60)
    print(f"Column: {col}")
    print(df[col].value_counts(dropna=False))
```

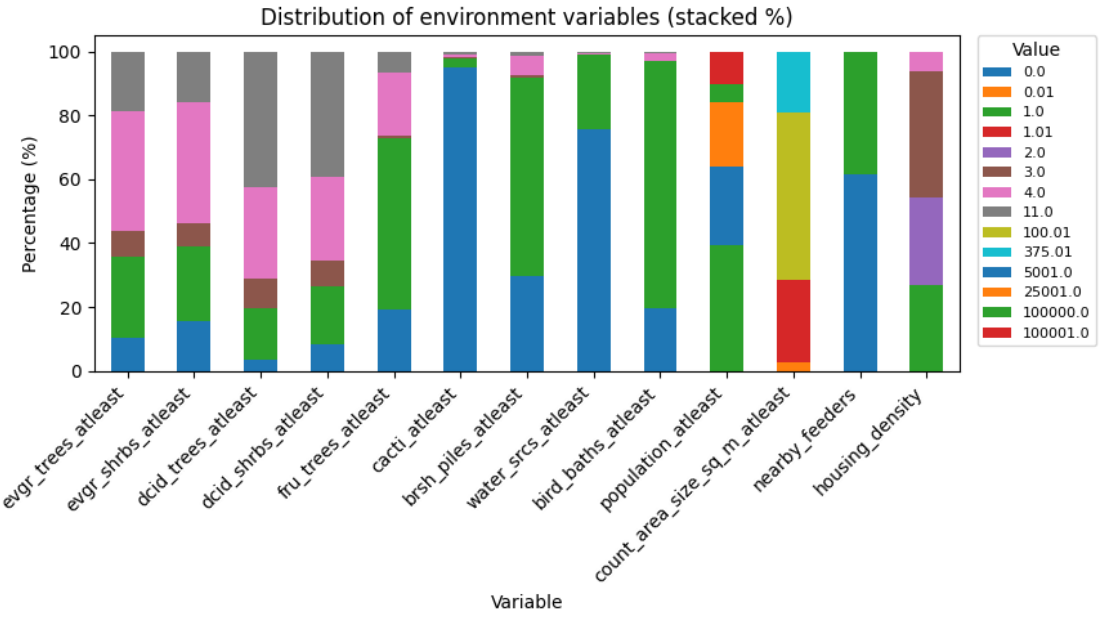
```
environment: ['evgr_trees_atleast', 'evgr_shrbs_atleast', 'dcid_trees_atleast',
'dcid_shrbs_atleast', 'fru_trees_atleast', 'cacti_atleast',
'brsh_piles_atleast', 'water_srcs_atleast', 'bird_baths_atleast',
'population_atleast', 'count_area_size_sq_m_atleast', 'nearby_feeders',
'housing_density']
============================================================
Column: evgr_trees_atleast
evgr_trees_atleast
4.0     94767
1.0     64503
11.0    47960
0.0     26482
3.0     20643
Name: count, dtype: int64
============================================================
Column: evgr_shrbs_atleast
evgr_shrbs_atleast
4.0     95843
1.0     58883
11.0    40571
0.0     40089
3.0     18969
Name: count, dtype: int64
============================================================
Column: dcid_trees_atleast
dcid_trees_atleast
11.0    107790
4.0      73134
1.0      40674
3.0      23886
0.0       8871
Name: count, dtype: int64
============================================================
Column: dcid_shrbs_atleast
dcid_shrbs_atleast
11.0    99354
4.0     66882
```

```
1.0     46353
0.0     20997
3.0     20769
Name: count, dtype: int64
============================================================
Column: fru_trees_atleast
fru_trees_atleast
1.0     136035
4.0      50128
0.0      48973
11.0     17113
3.0       2106
Name: count, dtype: int64
============================================================
Column: cacti_atleast
cacti_atleast
0.0     241306
1.0       7921
4.0       3051
11.0      1833
3.0        244
Name: count, dtype: int64
============================================================
Column: brsh_piles_atleast
brsh_piles_atleast
1.0     157899
0.0      75401
4.0      15521
11.0      3473
3.0       2061
Name: count, dtype: int64
============================================================
Column: water_srcs_atleast
water_srcs_atleast
0.0     192022
1.0      59846
4.0       1307
11.0      1180
Name: count, dtype: int64
============================================================
Column: bird_baths_atleast
bird_baths_atleast
1.0     197156
0.0      49776
4.0       6586
11.0       837
Name: count, dtype: int64
============================================================
```

```
Column: population_atleast
population_atleast
1.0         100325
5001.0       62911
25001.0      50869
100001.0     25778
100000.0     14472
Name: count, dtype: int64
============================================================
Column: count_area_size_sq_m_atleast
count_area_size_sq_m_atleast
100.01    133609
1.01       65072
375.01     48712
0.01        6962
Name: count, dtype: int64
============================================================
Column: nearby_feeders
nearby_feeders
0.0    156225
1.0     98130
Name: count, dtype: int64
============================================================
Column: housing_density
housing_density
3.0    100991
2.0     70068
1.0     67977
4.0     15319
Name: count, dtype: int64
```

[38]:
```python
import matplotlib.pyplot as plt

# Keep *_atleast variables + add nearby_feeders and housing_density
cols_to_plot = [c for c in environment_cols if c.endswith("_atleast")] + [
    "nearby_feeders",
    "housing_density"
]

# Build percentage distribution table
dist_table = (
    df[cols_to_plot]
    .apply(lambda s: s.value_counts(normalize=True))
    .fillna(0)
    * 100
)
```

```
# Sort the value levels on the x-axis
dist_table = dist_table.sort_index()

# Plot
ax = dist_table.T.plot(
    kind="bar",
    stacked=True,
    figsize=(11, 5)
)

plt.xlabel("Variable")
plt.ylabel("Percentage (%)")
plt.title("Distribution of environment variables (stacked %)")
plt.xticks(rotation=45, ha="right")

# Move legend outside the plot (right side)
plt.legend(
    title="Value",
    bbox_to_anchor=(1.02, 1),
    loc="upper left",
    borderaxespad=0,
    fontsize=8
)

# Leave space on the right for the legend
plt.tight_layout(rect=[0, 0, 0.8, 1])
plt.show()
```



4.Disturbance distribution

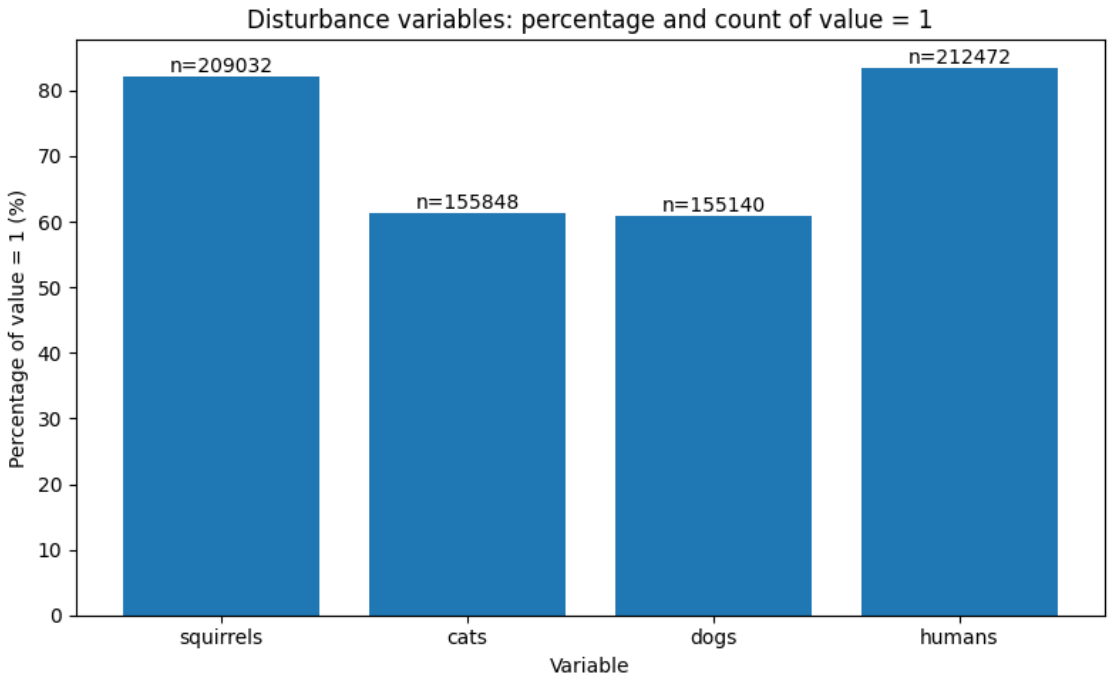
[39]:
```
print("disturbance:", disturbance_cols)
disturbance_summary = pd.DataFrame({
    "count_1": (df[disturbance_cols] == 1).sum(),
    "ratio_1_overall": (df[disturbance_cols] == 1).mean()
})
disturbance_summary
```

```
disturbance: ['squirrels', 'cats', 'dogs', 'humans']
```

[39]:

	count_1	ratio_1_overall
squirrels	209032	0.821812
cats	155848	0.612718
dogs	155140	0.609935
humans	212472	0.835336

[40]:
```
import matplotlib.pyplot as plt

# Convert ratio to percentage
percent = disturbance_summary["ratio_1_overall"] * 100

# One single chart, using matplotlib only (no custom colors)
plt.figure(figsize=(8, 5))
bars = plt.bar(disturbance_summary.index, percent)

plt.xlabel("Variable")
plt.ylabel("Percentage of value = 1 (%)")
plt.title("Disturbance variables: percentage and count of value = 1")



plt.xticks(rotation=0, ha="center")
# Annotate each bar with the corresponding count_1
for bar, cnt in zip(bars, disturbance_summary["count_1"]):
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        bar.get_height(),
        f"n={cnt}",
        ha="center",
        va="bottom"
    )

plt.tight_layout()
plt.show()
```



5.Feed time distribution

[41]:
```
print("feeding time:", feeding_time_cols)
feeding_time_summary = pd.DataFrame({
    "count_1": (df[feeding_time_cols] == 1).sum(),
    "ratio_1_overall": (df[feeding_time_cols] == 1).mean()
})

feeding_time_summary
```

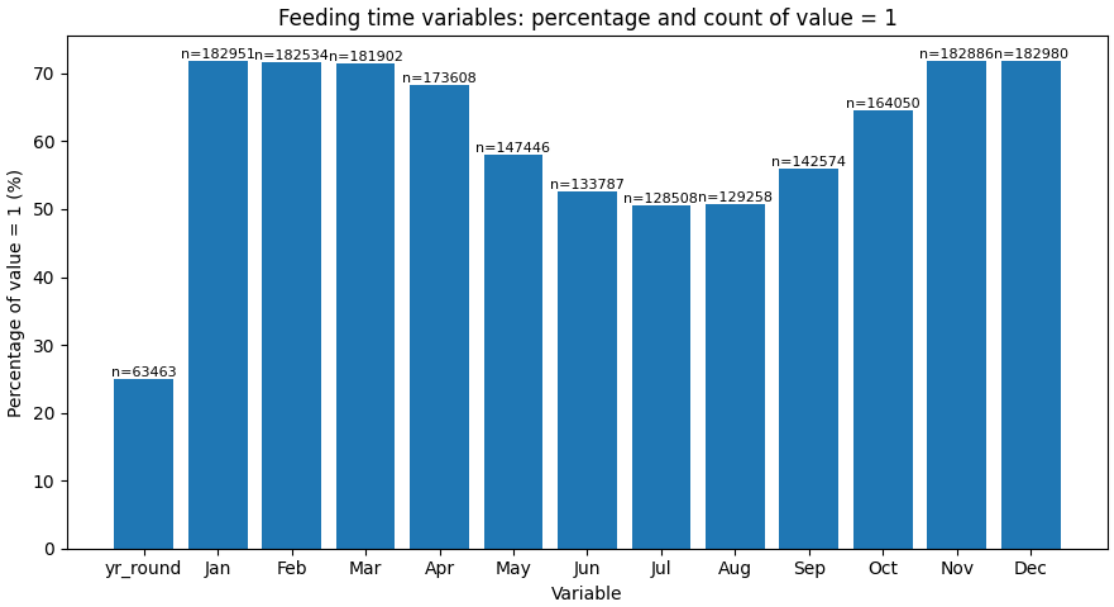
```
feeding time: ['fed_yr_round', 'fed_in_jan', 'fed_in_feb', 'fed_in_mar',
'fed_in_apr', 'fed_in_may', 'fed_in_jun', 'fed_in_jul', 'fed_in_aug',
'fed_in_sep', 'fed_in_oct', 'fed_in_nov', 'fed_in_dec']
```

[41]:

	count_1	ratio_1_overall
fed_yr_round	63463	0.249506
fed_in_jan	182951	0.719274
fed_in_feb	182534	0.717635
fed_in_mar	181902	0.715150
fed_in_apr	173608	0.682542
fed_in_may	147446	0.579686
fed_in_jun	133787	0.525985
fed_in_jul	128508	0.505231
fed_in_aug	129258	0.508180
fed_in_sep	142574	0.560532

```
fed_in_oct      164050          0.644965
fed_in_nov      182886          0.719019
fed_in_dec      182980          0.719388
```

[42]:
```python
import matplotlib.pyplot as plt

percent = feeding_time_summary["ratio_1_overall"] * 100

plt.figure(figsize=(9, 5))
bars = plt.bar(feeding_time_summary.index, percent)

plt.xlabel("Variable")
plt.ylabel("Percentage of value = 1 (%)")
plt.title("Feeding time variables: percentage and count of value = 1")

# build short labels for x-axis
short_labels = []
for name in feeding_time_summary.index:
    if name == "fed_yr_round":
        short_labels.append("yr_round")
    else:
        # fed_in_jan -> jan -> Jan
        short_labels.append(name.replace("fed_in_", "").capitalize())

plt.xticks(
    ticks=range(len(feeding_time_summary.index)),
    labels=short_labels,
    rotation=0,
    ha="center"
)


for bar, cnt in zip(bars, feeding_time_summary["count_1"]):
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        bar.get_height(),
        f"n={cnt}",
        ha="center",
        va="bottom",
        fontsize=8
    )

plt.tight_layout()
plt.show()
```



6.Distribution of the number of different types of feeders

[43]:
```
print("num_diff_types_feeder:", feeder_cols)
for col in feeder_cols:
    print("=" * 60)
    print(f"Column: {col}")
    print(df[col].value_counts(dropna=False))
```

```
num_diff_types_feeder: ['numfeeders_suet', 'numfeeders_ground',
'numfeeders_hanging', 'numfeeders_platfrm', 'numfeeders_humming',
'numfeeders_water', 'numfeeders_thistle', 'numfeeders_fruit',
'numfeeders_hopper', 'numfeeders_tube', 'numfeeders_other']
============================================================
Column: numfeeders_suet
numfeeders_suet
1.0        104526
2.0         64283
0.0         47385
3.0         22933
4.0          9137
5.0          3223
6.0          1933
7.0           354
8.0           264
10.0          102
9.0            85
11.0           41
12.0           33
```

```
13.0           10
14.0           10
21.0            5
15.0            4
20.0            3
33.0            2
30.0            2
16.0            2
23.0            2
25.0            2
34.0            1
32.0            1
28.0            1
117.0           1
31.0            1
45.0            1
14850.0         1
17.0            1
22.0            1
54.0            1
24.0            1
48.0            1
19.0            1
64.0            1
Name: count, dtype: int64
============================================================
Column: numfeeders_ground
numfeeders_ground
0.0     99975
1.0     86178
2.0     40861
3.0     16830
4.0      5828
5.0      2233
6.0      1828
7.0       202
8.0       173
10.0       80
9.0        44
11.0       32
12.0       31
20.0       10
14.0        9
15.0        9
16.0        7
13.0        5
30.0        3
18.0        2
```

```
22.0        2
28.0        1
19.0        1
40.0        1
33.0        1
75.0        1
76.0        1
17.0        1
67.0        1
63.0        1
32.0        1
21.0        1
23.0        1
42.0        1
Name: count, dtype: int64
============================================================
Column: numfeeders_hanging
numfeeders_hanging
0.0     158721
2.0      26154
1.0      20231
3.0      20169
4.0      12553
6.0       8058
5.0       6844
7.0        578
8.0        402
10.0       177
9.0        163
12.0        68
11.0        51
14.0        45
15.0        42
13.0        24
20.0        18
16.0        16
17.0         9
22.0         5
19.0         4
23.0         3
18.0         3
21.0         3
25.0         3
40.0         3
30.0         2
55.0         2
27.0         1
29.0         1
```

```
28.0         1
45.0         1
Name: count, dtype: int64
============================================================
Column: numfeeders_platfrm
numfeeders_platfrm
0.0      119376
1.0       85001
2.0       32275
3.0       10824
4.0        3940
5.0        1396
6.0        1045
7.0         190
8.0         106
9.0          56
10.0         49
11.0         22
12.0         17
20.0         10
14.0          8
19.0          6
13.0          4
23.0          4
18.0          3
16.0          3
17.0          2
24.0          2
21.0          2
15.0          2
25.0          2
28.0          1
31.0          1
22.0          1
48.0          1
111.0         1
27.0          1
32.0          1
62.0          1
100.0         1
65.0          1
Name: count, dtype: int64
============================================================
Column: numfeeders_humming
numfeeders_humming
0.0     200704
1.0      29109
2.0      15240
```

```
3.0       5210
4.0       2225
5.0        862
6.0        580
7.0        142
8.0        112
10.0        55
12.0        35
9.0         25
11.0        18
15.0         6
13.0         4
20.0         4
30.0         3
32.0         3
19.0         3
21.0         3
22.0         2
25.0         2
26.0         1
40.0         1
16.0         1
24.0         1
33.0         1
18.0         1
14.0         1
28.0         1
Name: count, dtype: int64
============================================================
Column: numfeeders_water
numfeeders_water
0.0     201905
1.0      36744
2.0      10180
3.0       3416
4.0       1171
5.0        459
6.0        420
8.0         21
7.0         13
10.0         7
12.0         6
11.0         4
13.0         3
9.0          3
16.0         2
20.0         1
Name: count, dtype: int64
```

```
============================================================
Column: numfeeders_thistle
numfeeders_thistle
0.0     201983
1.0      37189
2.0      10966
3.0       2703
4.0        852
5.0        300
6.0        255
7.0         34
8.0         29
12.0        12
11.0         8
9.0          7
10.0         6
18.0         2
22.0         2
21.0         1
32.0         1
16.0         1
50.0         1
20.0         1
17.0         1
14.0         1
Name: count, dtype: int64
============================================================
Column: numfeeders_fruit
numfeeders_fruit
0.0      236864
1.0       14776
2.0        2015
3.0         375
4.0         176
6.0          41
5.0          39
10.0         14
8.0          14
12.0         13
9.0           3
11.0          3
18.0          3
7.0           3
60.0          2
30.0          2
24.0          1
56.0          1
13.0          1
```

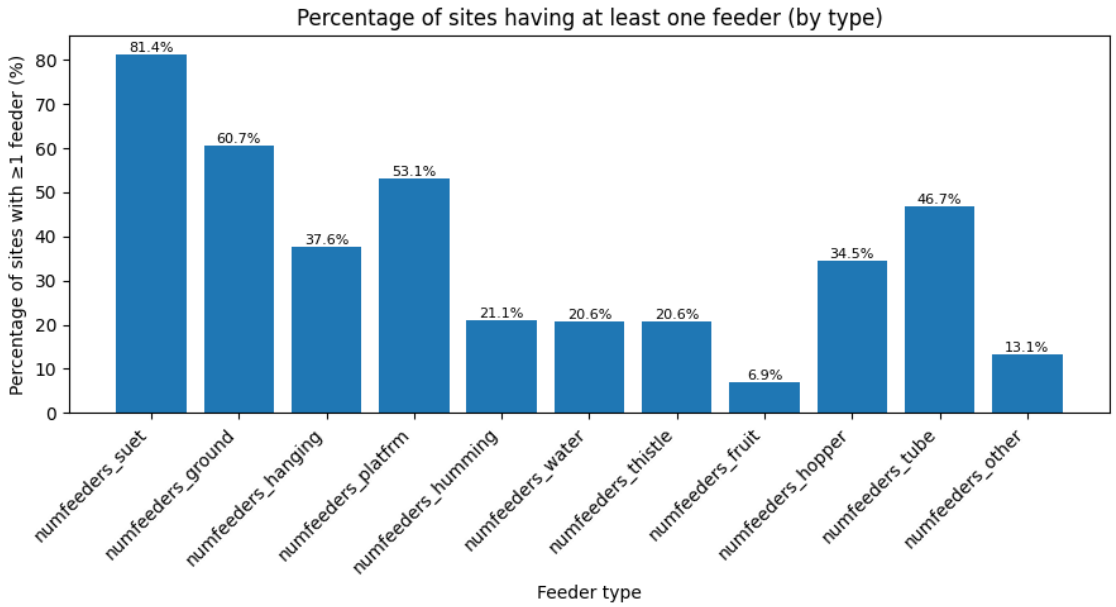
```
19.0          1
27.0          1
15.0          1
40.0          1
82.0          1
20.0          1
112.0         1
75.0          1
87.0          1
Name: count, dtype: int64
============================================================
Column: numfeeders_hopper
numfeeders_hopper
0.0      166612
1.0       45839
2.0       24078
3.0        9807
4.0        4301
5.0        1850
6.0         872
7.0         340
8.0         287
10.0        106
9.0          93
12.0         55
11.0         43
13.0         18
14.0         16
15.0         10
16.0          7
18.0          4
17.0          3
21.0          2
43.0          2
23.0          1
32.0          1
30.0          1
44.0          1
200.0         1
211.0         1
24.0          1
35.0          1
31.0          1
25.0          1
Name: count, dtype: int64
============================================================
Column: numfeeders_tube
numfeeders_tube
```

```
0.0      135613
1.0       41599
2.0       36403
3.0       20216
4.0       10465
5.0        4575
6.0        2484
7.0        1079
8.0         769
9.0         372
10.0        316
12.0        134
11.0         99
15.0         56
13.0         38
14.0         37
18.0         19
16.0         15
20.0         12
17.0          9
28.0          5
22.0          5
21.0          4
19.0          4
32.0          3
41.0          3
31.0          3
24.0          2
23.0          2
25.0          2
66.0          1
50.0          1
61.0          1
62.0          1
51.0          1
104.0         1
30.0          1
35.0          1
43.0          1
42.0          1
26.0          1
44.0          1
Name: count, dtype: int64
============================================================
Column: numfeeders_other
numfeeders_other
0.0     220959
1.0      21069
```

```
2.0       7572
3.0       2583
4.0       1072
5.0        437
6.0        276
7.0        102
8.0        100
10.0        61
9.0         41
11.0        18
12.0        17
14.0        11
15.0         8
16.0         6
20.0         5
18.0         4
13.0         3
27.0         1
33.0         1
24.0         1
17.0         1
19.0         1
25.0         1
99.0         1
40.0         1
98.0         1
32.0         1
58.0         1
Name: count, dtype: int64
```

[44]:
```python
import matplotlib.pyplot as plt
import pandas as pd

has_feeder_ratio = (df[feeder_cols] >= 1).mean() * 100

plt.figure(figsize=(9,5))
bars = plt.bar(has_feeder_ratio.index, has_feeder_ratio.values)

plt.ylabel("Percentage of sites with  1 feeder (%)")
plt.xlabel("Feeder type")
plt.title("Percentage of sites having at least one feeder (by type)")

plt.xticks(rotation=45, ha="right")

for bar, val in zip(bars, has_feeder_ratio.values):
    plt.text(
        bar.get_x() + bar.get_width()/2,
```

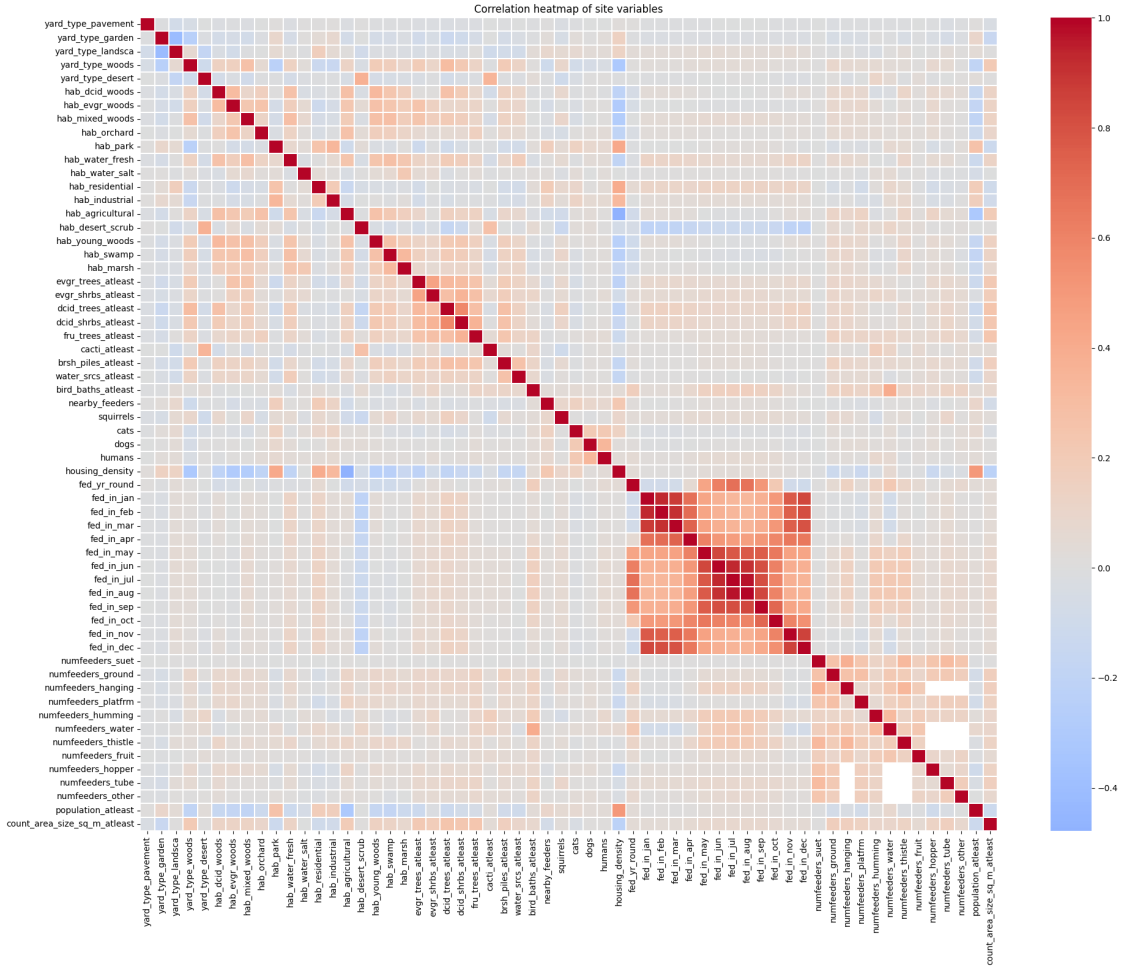
```
        bar.get_height(),
        f"{val:.1f}%",
        ha="center",
        va="bottom",
        fontsize=8
    )

plt.tight_layout()
plt.show()
```



##(5)Correlation heatmap

[53]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_csv("/content/PFW_count_site_data_public_2021.csv")

df_num = df.select_dtypes(include="number")

corr = df_num.corr()

plt.figure(figsize=(20, 16))
sns.heatmap(
    corr,
    cmap="coolwarm",
    center=0,
```

```
    linewidths=0.1
)

plt.title("Correlation heatmap of site variables")
plt.tight_layout()
plt.show()
```



#4.EDA after combining tables

Merge tables by loc_id

[60]:
```
import pandas as pd


# df1: observation results

df1 = pd.read_csv("/content/PFW_2021_public.csv")
```

```
# keep only valid observations
df1 = df1[df1["valid"] == 1].copy()

# make sure we only keep the 2021 season
df1 = df1[df1["Year"] == 2021].copy()

# aggregate to checklist level first
sub_level = (
    df1.groupby(["loc_id", "sub_id"], as_index=False)
       .agg(
           n_species=("species_code", "nunique"),
           total_birds=("how_many", "sum"),
           Year=("Year", "first")
       )
)

# aggregate to site level
site_level = (
    sub_level.groupby("loc_id", as_index=False)
             .agg(
                 avg_species=("n_species", "mean"),
                 avg_birds=("total_birds", "mean"),
                 Year=("Year", "first")
             )
)


# df2: habitat / yard data

df2 = pd.read_csv("/content/PFW_count_site_data_public_2021.csv")

# print columns once to check real names
print("df2 columns:", df2.columns.tolist())

# find the proj_period_id column (case-insensitive)
cols_lower = {c.strip().lower(): c for c in df2.columns}
proj_col = cols_lower.get("proj_period_id")

if proj_col is None:
    raise KeyError("Column proj_period_id is not found in df2.")

# keep only PFW_2021 to match df1 (Year = 2021)
df2[proj_col] = df2[proj_col].astype(str).str.strip()
df2_2021 = df2[df2[proj_col] == "PFW_2021"].copy()


# merge by loc_id
```

```
merged = site_level.merge(
    df2_2021,
    on="loc_id",
    how="left"
)

# quick alignment check
print("Number of sites in df1:", site_level["loc_id"].nunique())
print("Number of sites in df2 (PFW_2021):", df2_2021["loc_id"].nunique())
print("Number of matched sites:", merged[proj_col].notna().sum())

merged.head()
```

```
df2 columns: ['loc_id', 'proj_period_id', 'yard_type_pavement',
'yard_type_garden', 'yard_type_landsca', 'yard_type_woods', 'yard_type_desert',
'hab_dcid_woods', 'hab_evgr_woods', 'hab_mixed_woods', 'hab_orchard',
'hab_park', 'hab_water_fresh', 'hab_water_salt', 'hab_residential',
'hab_industrial', 'hab_agricultural', 'hab_desert_scrub', 'hab_young_woods',
'hab_swamp', 'hab_marsh', 'evgr_trees_atleast', 'evgr_shrbs_atleast',
'dcid_trees_atleast', 'dcid_shrbs_atleast', 'fru_trees_atleast',
'cacti_atleast', 'brsh_piles_atleast', 'water_srcs_atleast',
'bird_baths_atleast', 'nearby_feeders', 'squirrels', 'cats', 'dogs', 'humans',
'housing_density', 'fed_yr_round', 'fed_in_jan', 'fed_in_feb', 'fed_in_mar',
'fed_in_apr', 'fed_in_may', 'fed_in_jun', 'fed_in_jul', 'fed_in_aug',
'fed_in_sep', 'fed_in_oct', 'fed_in_nov', 'fed_in_dec', 'numfeeders_suet',
'numfeeders_ground', 'numfeeders_hanging', 'numfeeders_platfrm',
'numfeeders_humming', 'numfeeders_water', 'numfeeders_thistle',
'numfeeders_fruit', 'numfeeders_hopper', 'numfeeders_tube', 'numfeeders_other',
'population_atleast', 'count_area_size_sq_m_atleast']
Number of sites in df1: 13787
Number of sites in df2 (PFW_2021): 13222
Number of matched sites: 10246
```

[60]:

	loc_id	avg_species	avg_birds	Year	proj_period_id	yard_type_pavement	yard_type_garden	yard_type_landsca	yard_type_woods	yard_type_desert	hab_dcid_woods	hab_evgr_woods
0	L100032	1.500000	2.000000	2021	PFW_2021	0.0	0.0	1.0	1.0	0.0	1.0	0.0
1	L100057	1.500000	8.333333	2021	PFW_2021	0.0	0.0	1.0	0.0	0.0	0.0	0.0
2	L10007061	1.000000	1.500000	2021	PFW_2021	0.0	0.0	1.0	1.0	0.0	1.0	0.0
3	L10011691	1.111111	5.222222	2021	NaN	NaN						

```
NaN                NaN              NaN               NaN             NaN
NaN
4   L1001179     1.000000  15.000000  2021            NaN                 NaN
NaN                NaN              NaN               NaN             NaN
NaN
```

```
   hab_mixed_woods  hab_orchard  hab_park  hab_water_fresh  hab_water_salt
hab_residential  hab_industrial  hab_agricultural  hab_desert_scrub
hab_young_woods  hab_swamp  hab_marsh  \
0              1.0          0.0       0.0              1.0             0.0
1.0             1.0               0.0               0.0             0.0
0.0        0.0
1              1.0          0.0       0.0              0.0             0.0
1.0             0.0               0.0               0.0             0.0
0.0        0.0
2              0.0          0.0       0.0              1.0             0.0
1.0             0.0               1.0               0.0             0.0
0.0        0.0
3              NaN          NaN       NaN              NaN             NaN
NaN             NaN               NaN               NaN             NaN
NaN        NaN
4              NaN          NaN       NaN              NaN             NaN
NaN             NaN               NaN               NaN             NaN
NaN        NaN
```

```
   evgr_trees_atleast  …  fed_in_jan  fed_in_feb  fed_in_mar  fed_in_apr
fed_in_may  fed_in_jun  fed_in_jul  fed_in_aug  fed_in_sep  fed_in_oct
fed_in_nov  fed_in_dec  numfeeders_suet  \
0                11.0  …         1.0         1.0         1.0         1.0
1.0         1.0         1.0         1.0         1.0         1.0         1.0
1.0              1.0
1                 4.0  …         1.0         1.0         1.0         1.0
1.0         1.0         1.0         1.0         1.0         1.0         1.0
1.0              1.0
2                11.0  …         1.0         1.0         1.0         1.0
1.0         1.0         1.0         1.0         1.0         1.0         1.0
1.0              1.0
3                 NaN  …         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN              NaN
4                 NaN  …         NaN         NaN         NaN         NaN
NaN         NaN         NaN         NaN         NaN         NaN         NaN
NaN              NaN
```

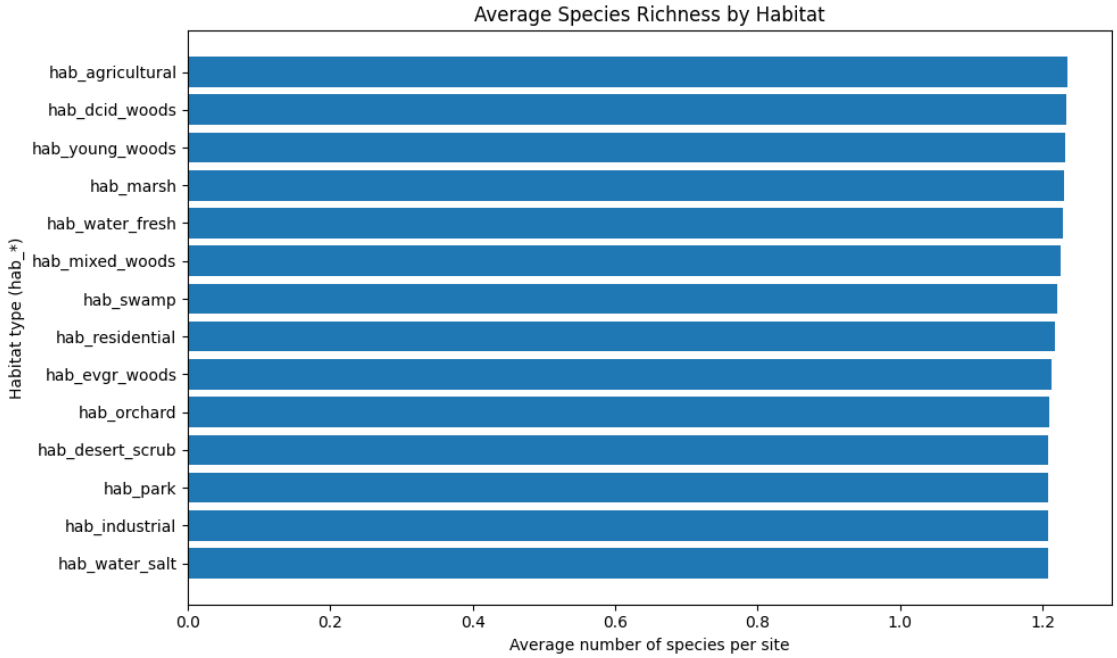
```
   numfeeders_ground  numfeeders_hanging  numfeeders_platfrm  numfeeders_humming
numfeeders_water  numfeeders_thistle  numfeeders_fruit  numfeeders_hopper
numfeeders_tube  numfeeders_other  \
```

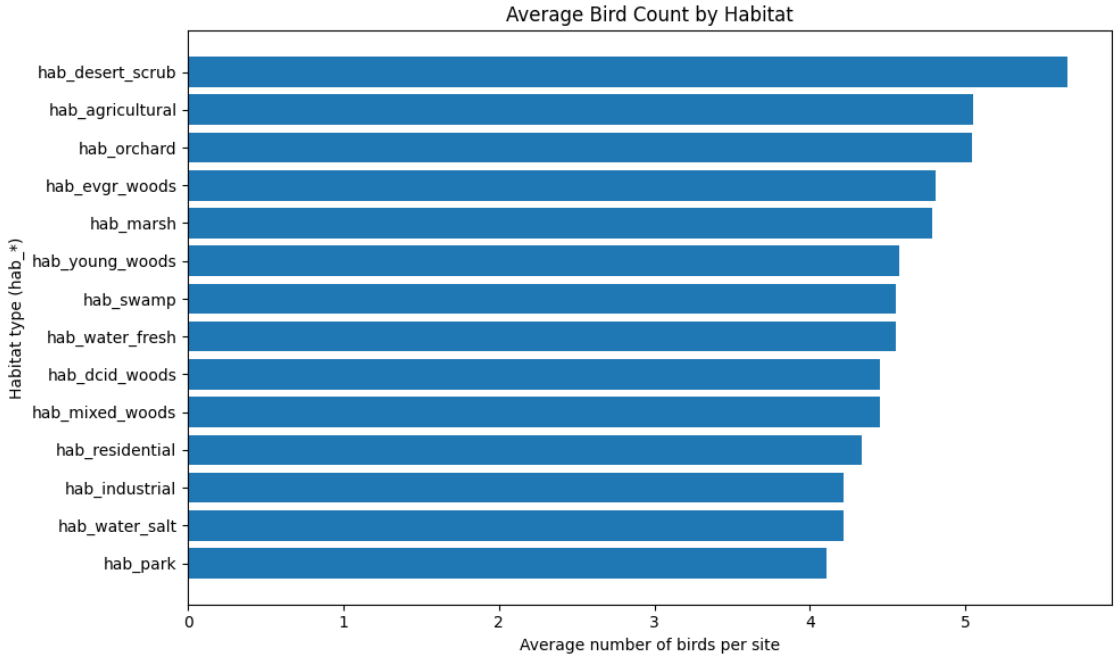
```
0                1.0                NaN                2.0                0.0
NaN                 NaN                0.0                1.0                4.0
2.0
1                3.0                NaN                2.0                1.0
NaN                 NaN                0.0                4.0                2.0
0.0
2                0.0                NaN                1.0                0.0
NaN                 NaN                0.0                0.0                2.0
0.0
3                NaN                NaN                NaN                NaN
NaN                 NaN                NaN                NaN                NaN
NaN
4                NaN                NaN                NaN                NaN
NaN                 NaN                NaN                NaN                NaN
NaN

   population_atleast  count_area_size_sq_m_atleast
0              5001.0                        100.01
1              5001.0                        375.01
2              5001.0                          1.01
3                 NaN                           NaN
4                 NaN                           NaN

[5 rows x 65 columns]
```

[70]:
```
merged.to_csv("/content/merged_on_site.csv", index=False)
```

[71]:
```
import pandas as pd
import matplotlib.pyplot as plt


# 1) Load merged dataset

merged_path = "/content/merged_on_site.csv"
merged = pd.read_csv(merged_path)


# 2) Identify habitat columns
hab_cols = [c for c in merged.columns if c.lower().startswith("hab_")]


# 3) Build habitat summary
rows = []

for col in hab_cols:
    sub = merged[merged[col] == 1]
```

```
    rows.append({
        "habitat": col,
        "avg_species": sub["avg_species"].mean(),
        "avg_birds": sub["avg_birds"].mean(),
        "n_sites": sub.shape[0]
    })

hab_summary = pd.DataFrame(rows)

# Sort for nicer plots
hab_summary_species = hab_summary.sort_values("avg_species", ascending=True)
hab_summary_birds   = hab_summary.sort_values("avg_birds", ascending=True)


# 4) Plot: average species by habitat

plt.figure(figsize=(10, 6))
plt.barh(hab_summary_species["habitat"],
         hab_summary_species["avg_species"])
plt.xlabel("Average number of species per site")
plt.ylabel("Habitat type (hab_*)")
plt.title("Average Species Richness by Habitat")
plt.tight_layout()
plt.show()


# 5) Plot: average birds by habitat
plt.figure(figsize=(10, 6))
plt.barh(hab_summary_birds["habitat"],
         hab_summary_birds["avg_birds"])
plt.xlabel("Average number of birds per site")
plt.ylabel("Habitat type (hab_*)")
plt.title("Average Bird Count by Habitat")
plt.tight_layout()
plt.show()
```





The average number of species across different habitats was nearly identical, clustering around approximately 1.2 overall. This indicates that habitat type had no discernible effect on species richness within this dataset.

The average number of birds observed differed markedly between habitats. Sites within desert

scrub, agricultural land, and orchards recorded higher average bird counts, whereas parks, industrial estates, and saline water bodies exhibited comparatively lower average bird numbers. This indicates that habitat type strongly influences bird abundance.

Merge by each checklist (observation)

[72]:
```
import pandas as pd


# 1. load data
df1 = pd.read_csv("/content/PFW_2021_public.csv")
df2 = pd.read_csv("/content/PFW_count_site_data_public_2021.csv")


# 2. make column names case-insensitive

df1.columns = df1.columns.str.strip().str.lower()
df2.columns = df2.columns.str.strip().str.lower()


# 3. build proj_period_id in df1
#    to match df2 (e.g. PFW_2021)

if "year" not in df1.columns:
    raise KeyError("df1 does not contain column 'Year'.")

df1["proj_period_id"] = "PFW_" + df1["year"].astype(int).astype(str)


# 4. make sure join keys exist

required_cols_df1 = {"loc_id", "proj_period_id"}
required_cols_df2 = {"loc_id", "proj_period_id"}

if not required_cols_df1.issubset(df1.columns):
    raise KeyError("df1 is missing join keys.")

if not required_cols_df2.issubset(df2.columns):
    raise KeyError("df2 is missing join keys.")

# 5. merge (keep all df1 rows)

merged_on_checklist = df1.merge(
    df2,
    on=["loc_id", "proj_period_id"],
    how="left",
    suffixes=("", "_site")
)
```

```
# 6. quick check
print("rows in df1:", df1.shape[0])
print("rows after merge:", merged_obs_site.shape[0])

merged_on_checklist.head()
```

```
rows in df1: 100000
rows after merge: 100000
```

[72]:

```
      loc_id   latitude   longitude subnational1_code           entry_technique
sub_id         obs_id  month  day  year proj_period_id species_code  how_many
valid  reviewed  day1_am  day1_pm  \
0    L981010  52.129760 -122.135470             CA-BC  POSTCODE LAT/LONG LOOKUP
S83206450  OBS1092604618      3    4  2021       PFW_2021       amegfi        20
1         0        1        0
1   L3161698  43.832207 -123.092405             US-OR       /GOOGLE_MAP/ZOOM:18
S78031190  OBS1036509564     12   19  2020       PFW_2020       moudov        11
1         0        1        1
2  L13210778  39.721470  -75.933660             US-MD       /GOOGLE_MAP/ZOOM:15
S81318993  OBS1073386105      2   13  2021       PFW_2021       tuftit         2
1         0        1        1
3  L13258348  42.217874  -83.672300             US-MI       /GOOGLE_MAP/ZOOM:15
S79251313  OBS1051702542      1   13  2021       PFW_2021       houspa         2
1         0        1        1
4    L149639  32.749921  -79.941582             US-SC           PointMaker1.0_2
S79183993  OBS1050809672      1   11  2021       PFW_2021       balori        10
1         0        1        1

   day2_am  day2_pm  effort_hrs_atleast  snow_dep_atleast data_entry_method
yard_type_pavement  yard_type_garden  yard_type_landsca  …  fed_in_jan
fed_in_feb  fed_in_mar  fed_in_apr  fed_in_may  \
0        1        0               1.001              5.0     PFW Web 4.1.4
0.0              0.0               1.0  …         1.0         1.0
1.0         1.0         1.0
1        1        1               1.001              0.0     PFW Web 4.1.4
NaN              NaN               NaN  …         NaN         NaN
NaN         NaN         NaN
2        1        1               8.001              5.0     PFW Web 4.1.4
0.0              0.0               1.0  …         1.0         1.0
1.0         1.0         1.0
3        1        1               4.001              0.0     PFW Web 4.1.4
0.0              0.0               1.0  …         1.0         1.0
1.0         1.0         1.0
4        1        1               1.001              0.0     PFW Web 4.1.4
0.0              0.0               1.0  …         1.0         1.0
```

```
1.0         1.0         1.0

   fed_in_jun  fed_in_jul  fed_in_aug  fed_in_sep  fed_in_oct  fed_in_nov
fed_in_dec  numfeeders_suet  numfeeders_ground  numfeeders_hanging
numfeeders_platfrm  numfeeders_humming  \
0         1.0         1.0         1.0         1.0         1.0         1.0
1.0              4.0               0.0               NaN                 0.0
1.0
1         NaN         NaN         NaN         NaN         NaN         NaN
NaN              NaN               NaN               NaN                 NaN
NaN
2         1.0         1.0         1.0         1.0         1.0         1.0
1.0              5.0               0.0               NaN                 0.0
2.0
3         1.0         1.0         1.0         1.0         1.0         1.0
1.0              3.0               2.0               NaN                 1.0
0.0
4         1.0         1.0         1.0         1.0         1.0         1.0
1.0              4.0               1.0               NaN                 1.0
1.0

   numfeeders_water  numfeeders_thistle  numfeeders_fruit  numfeeders_hopper
numfeeders_tube  numfeeders_other  population_atleast
count_area_size_sq_m_atleast
0               NaN                 NaN               0.0                 0.0
2.0               0.0                 1.0                          100.01
1               NaN                 NaN               NaN                 NaN
NaN               NaN                 NaN                             NaN
2               NaN                 NaN               0.0                 3.0
3.0               0.0              5001.0                          100.01
3               NaN                 NaN               0.0                 1.0
2.0               0.0              5001.0                          100.01
4               NaN                 NaN               8.0                 1.0
3.0               3.0             25001.0                          375.01

[5 rows x 82 columns]
```

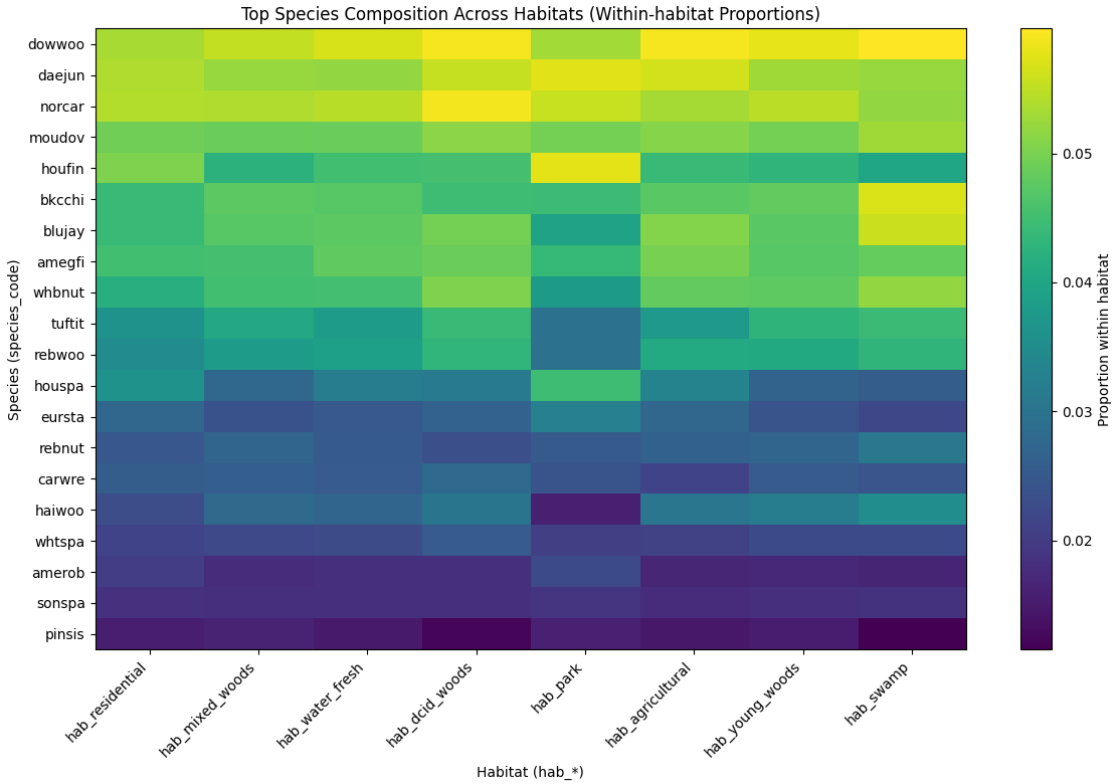
[74]:
```
merged_on_checklist.to_csv("/content/merged_on_checklist.csv", index=False)
```

[77]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# 1) Load observation-level merged dataset
# If your file is in /content, change the path accordingly.
path = "/content/merged_on_checklist.csv"   # <-- change if needed
df = pd.read_csv(path)
```

```python
# Make column names case-insensitive
df.columns = df.columns.str.strip().str.lower()

# Basic sanity checks
required = {"loc_id", "species_code"}
missing = required - set(df.columns)
if missing:
    raise KeyError(f"Missing required columns: {missing}")


# 2) Identify habitat columns
hab_cols = [c for c in df.columns if c.startswith("hab_")]
if len(hab_cols) == 0:
    raise KeyError("No habitat columns found (hab_*).")

# Optional: keep only valid observations if the column exists
if "valid" in df.columns:
    df = df[df["valid"] == 1].copy()


# 3) Choose top K habitats by number of observations (hab == 1)
K = 8   # number of habitats to compare
hab_sizes = {h: int((df[h] == 1).sum()) for h in hab_cols}
top_habs = sorted(hab_sizes, key=hab_sizes.get, reverse=True)[:K]

print("Top habitats by #observations:")
for h in top_habs:
    print(h, hab_sizes[h])


# 4) Choose top N species overall within selected habitats
N = 20  # number of species to show

df_sel = df[df[top_habs].eq(1).any(axis=1)].copy()
top_species = df_sel["species_code"].value_counts().head(N).index.tolist()

print("\nTop species shown:", top_species[:10], "...")


# 5) Build a species x habitat matrix of within-habitat proportions
#    proportion = count(species in habitat) / total observations in habitat

mat = []
for h in top_habs:
    sub = df[df[h] == 1]
    denom = len(sub)
```

```
    counts = sub["species_code"].value_counts()
    row = [(counts.get(sp, 0) / denom) if denom > 0 else 0 for sp in␣
 ↪top_species]
    mat.append(row)

# mat currently: habitats x species -> convert to species x habitats
mat = np.array(mat).T

heat_df = pd.DataFrame(mat, index=top_species, columns=top_habs)


# 6) Plot heatmap (matplotlib only)

plt.figure(figsize=(12, 8))
im = plt.imshow(heat_df.values, aspect="auto")  # no manual colors

plt.colorbar(im, label="Proportion within habitat")

plt.xticks(ticks=np.arange(len(top_habs)), labels=top_habs, rotation=45,␣
 ↪ha="right")
plt.yticks(ticks=np.arange(len(top_species)), labels=top_species)

plt.title("Top Species Composition Across Habitats (Within-habitat␣
 ↪Proportions)")
plt.xlabel("Habitat (hab_*)")
plt.ylabel("Species (species_code)")
plt.tight_layout()
plt.show()
```

```
Top habitats by #observations:
hab_residential 65424
hab_mixed_woods 49144
hab_water_fresh 43372
hab_dcid_woods 34277
hab_park 25728
hab_agricultural 24740
hab_young_woods 23478
hab_swamp 15653

Top species shown: ['dowwoo', 'daejun', 'norcar', 'moudov', 'houfin', 'bkcchi',
'blujay', 'amegfi', 'whbnut', 'tuftit'] …
```


Top Species Composition Across Habitats (Within-habitat Proportions)
dowwoo
daejun
norcar
moudov
houfin
bkcchi
blujay
amegfi
whbnut
tuftit
rebwoo
houspa
eursta
rebnut
carwre
haiwoo
whtspa
amerob
sonspa
pinsis
Species (species_code)
hab_residential
hab_mixed_woods
hab_water_fresh
hab_dcid_woods
hab_park
hab_agricultural
hab_young_woods
hab_swamp
Habitat (hab_*)
0.05
0.04
0.03
0.02
Proportion within habitat