📖 CurrentDataset.md

Visualize the current dataset

Lets Draw...

- ✅ number of train and test assets that were sampled for each of the target species
  - `TrainTestSplitForCurrentDataset.png`
- ✅ Asset source by species
  - `AssetSourceForCurrentDatasetLine.png`
  - `AssetSourceForCurrentDatasetSplit.png`
  - `AssetCountsPerSpeciesBySourceDualBar.png`
- ✅ Unique checklists & users per species
  - `UniqueChecklistsUsersPerSpecies.png`
- ✅ Number of months represented per species
  - `MonthsRepresentedPerSpecies.png`
- ✅ Unique country counts by species
  - `UniqueCountriesBySource.png`
- ✅ Song / call `TAG_*` counts, aggregate
  - `TagCountsCurrentDatasetBySource.png`
- ✅ Priority counts for current dataset
- ✅ ProtocolId counts for current dataset
- ✅ point map of splits for each asset
  - ✅ faux density map of sources
- ~~for grab all `reportAs` & `speciesCode` values for the current 54440 assets;~~
  - ~~print out taxanomic level using eBird taxonamy / issf / spuh~~
  - ~~make a bar plot to these splits~~
  - ~~can taxon splits be used to diversify selection?~~

setup:

```
import pandas as pd
import os
import numpy as np
from matplotlib import pyplot as plt
import datetime
import json
import csv
import random
```

```
import cartopy.crs as ccrs
import math
```

Plot configurations

```
# Plot configurations
# Set matplotlib font size
SMALL_SIZE = 12
MEDIUM_SIZE = 14
BIGGER_SIZE = 18
EXTRACHUNGUS_SIZE = 22

plt.rc('font', size=BIGGER_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=EXTRACHUNGUS_SIZE)    # fontsize of the axes title
plt.rc('axes', labelsize=BIGGER_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=MEDIUM_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=BIGGER_SIZE)    # legend fontsize
plt.rc('figure', titlesize=EXTRACHUNGUS_SIZE)  # fontsize of the figure title
```

file paths to import

```
## file paths:
# this csv contains a variety of additional project info, in addition to the species codes field `code`:
TARGET_SPECIES_FP = "./v2/us_ca_dataset_bird_species_info.csv"

# this is the v2 dataset, e.g. `assets_df`:
PREVIOUS_ASSETS_FP = "./v2/complete_project_assets.pkl"

# these json files only contain asset ids by split:
PREVIOUS_TRAIN_ASSETS_IDS_FP = "./v2/complete_train_asset_ids.json"
PREVIOUS_TEST_ASSETS_IDS_FP = "./v2/complete_test_asset_ids.json"

EBIRD_TAXONOMY_FP = "./clements/eBird_Taxonomy_v2019.csv"
```

ℹ️ *load required files*

```
# required files for asset_selection, eg target species:
categories_df = pd.read_csv(TARGET_SPECIES_FP)
target_species = categories_df['code']

# set a variable for `original_target_species`, so we can keep track of target species losses:
original_target_species = target_species

# we'll use the previous project assets csv as a sanity check,
# e.g. compare current export with the last for dupes, missing entries etc.
assets_df = pd.read_pickle("v2/complete_project_assets_20210418.pkl")

with open(PREVIOUS_TRAIN_ASSETS_IDS_FP) as f:
    prev_train_ids = json.load(f)

with open(PREVIOUS_TEST_ASSETS_IDS_FP) as f:
    prev_test_ids = json.load(f)



print("`assets_df`: Loaded in %d assets from previous `v2` model dataset" % (assets_df.shape[0],))
print("`categories_df`: Loaded in %d columns from `%s`" % (categories_df.shape[1], TARGET_SPECIES_FP))
print("`target_species` & `original_target_species`: Loaded in %d target species from `%s`" % (target_species.shape[0]
```

```
`assets_df`: Loaded in 54440 assets from previous `v2` model dataset
`categories_df`: Loaded in 10 columns from `./v2/us_ca_dataset_bird_species_info.csv`
`target_species` & `original_target_species`: Loaded in 562 target species from `./v2/us_ca_dataset_bird_species_info.
```

ℹ️ *Process the taxonomy file as dictionaries*

```
## Process the taxonomy file

# Keep a dict mapping species code to its info
all_species_code_map = {}

# this array will contain those species codes we will use as class labels
target_species_codes = []

# this dict will map a target species code to its subspecies/forms
target_species_code_to_issf = {}

# this dict will be used to determine if an image labeled with given species code can be included in the dataset
# this dict will map issf to their corresponding species
species_code_to_target_species_code = {}
species_sci_name_to_species_code = {}

with open(EBIRD_TAXONOMY_FP, encoding='utf-8-sig') as f:

    reader = csv.DictReader(f)

    for row in reader:

        taxon_order = row['TAXON_ORDER']
        category = row['CATEGORY']
        species_code = row['SPECIES_CODE']
        primary_com_name = row['PRIMARY_COM_NAME']
        sci_name = row['SCI_NAME']
        order = row['ORDER1']
        family = row['FAMILY']
        species_group = row['SPECIES_GROUP']
        report_as = row['REPORT_AS']

        taxon = {
            "taxon_order" : int(taxon_order),
            "category" : category,
            "species_code" : species_code,
            "primary_com_name" : primary_com_name,
            "sci_name" : sci_name,
            "order" : order,
            "family" : family,
            "species_group" : species_group,
            "report_as" : report_as
        }

        all_species_code_map[species_code] = taxon

        if category == 'species':

            target_species_codes.append(species_code)
            species_code_to_target_species_code[species_code] = species_code

            assert sci_name not in species_sci_name_to_species_code, str(taxon)
            species_sci_name_to_species_code[sci_name] = species_code

            # Initialize an empty array
            target_species_code_to_issf[species_code] = []

        # Map Identifiable subspecies or group of subspecies back up to species.
        if category == 'issf':
```

```
            species_name = " ".join(sci_name.split()[:2])
            species_code_to_target_species_code[species_code] = species_sci_name_to_species_code[species_name]
            target_species_code_to_issf[species_sci_name_to_species_code[species_name]].append(species_code)

total_num_targets = len(target_species_codes)
print("Found %d target categories" % total_num_targets)
```

```
Found 10721 target categories
```

ℹ️ *Load & evaluate the taxonomy file as dataframe*

```
ebird_taxonomy_df = pd.read_csv(EBIRD_TAXONOMY_FP)
```

```
print("`ebird_taxonomy_df`: Loaded in %d rows from `%s`" % (ebird_taxonomy_df.shape[0], EBIRD_TAXONOMY_FP))
print("... found %d species level identifiers: " % (ebird_taxonomy_df['CATEGORY'].unique().shape[0]))
for identifier in ebird_taxonomy_df['CATEGORY'].unique():
    print('  - ' + identifier)
```

```
`ebird_taxonomy_df`: Loaded in 16513 rows from `./clements/eBird_Taxonomy_v2019.csv`
... found 8 species level identifiers:
  - species
  - slash
  - issf
  - spuh
  - hybrid
  - domestic
  - form
  - intergrade
```

ℹ️ *lets see if we added / removed any target species*

```
# let see if we added / removed any target species
# lets confirm this is wht we want at sound id standup
prev_species_set = set(assets_df['reportAs'])
current_target_species_set = set(categories_df['code'])

print("The current model dataset has %d species represented" % (len(prev_species_set)))
print("...we currently are looking for %d species " % (len(current_target_species_set)))

print("...we removed the following species: ")
for sp in prev_species_set - current_target_species_set:
    print(sp)

print("...we added the following species: ")
if not current_target_species_set - prev_species_set:
    print('0 added species!')
    for sp in current_target_species_set - prev_species_set:
        print(sp)
```

```
The current model dataset has 569 species represented
...we currently are looking for 562 species
...we removed the following species:
owl1
amaant1
whwsco2
chivir1
harduc
```

```
sursco
rosspo1
...we added the following species:
0 added species!
```

ℹ️ *Make sure we arn't trying to target any non-species codes:*

```
# Make sure we arn't trying use non-species:
non_species = dict()

print('checking `target_species` list...')
for species in target_species:
    species_cat = ebird_taxonomy_df[ebird_taxonomy_df['SPECIES_CODE']==species]['CATEGORY']
    if species_cat.max() != 'species':
        # if this is the case, figure out why this is a target species:
        print(species + ' is not a species!  it is a ' + species_cat.max())
        non_species[species] = species_cat.max()

# all is well:
if not len(non_species):
    print('all %d codes in `target_species` are legit species, hurray!  (not "spuh" etc)' % (len(target_species)))

# all isn't well:
assert len(non_species) == 0, "A target species is not actually a species!!"
```

```
checking `target_species` list...
all 562 codes in `target_species` are legit species, hurray!  (not "spuh" etc)
```

```
# Make sure we arn't trying use non-species in assets_df;
# if we are, share this info with the group
non_species = dict()

print('checking `assets_df`...')
for species in assets_df['reportAs']:
    species_cat = ebird_taxonomy_df[ebird_taxonomy_df['SPECIES_CODE']==species]['CATEGORY']
    if species_cat.max() != 'species':
        # if this is the case, figure out why this is a target species:
        print(species + ' is not a species!  it is a ' + species_cat.max())
        non_species[species] = species_cat.max()

# all is well:
if not len(non_species):
    print('all %d codes in `target_species` are legit species, hurray!  (not "spuh" etc)' % (len(target_species)))
```

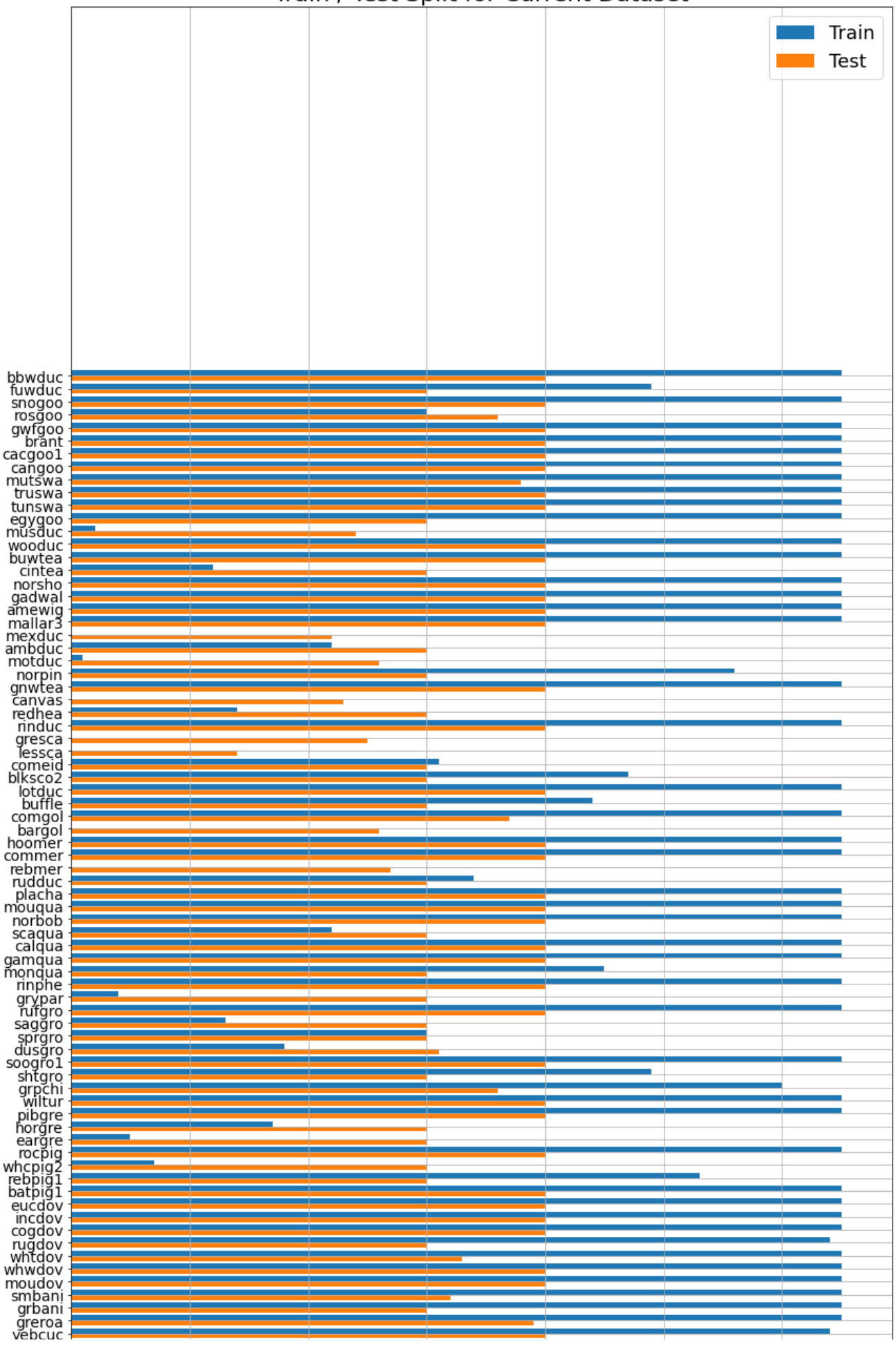
```
checking `assets_df`...
owl1 is not a species!  it is a spuh
```

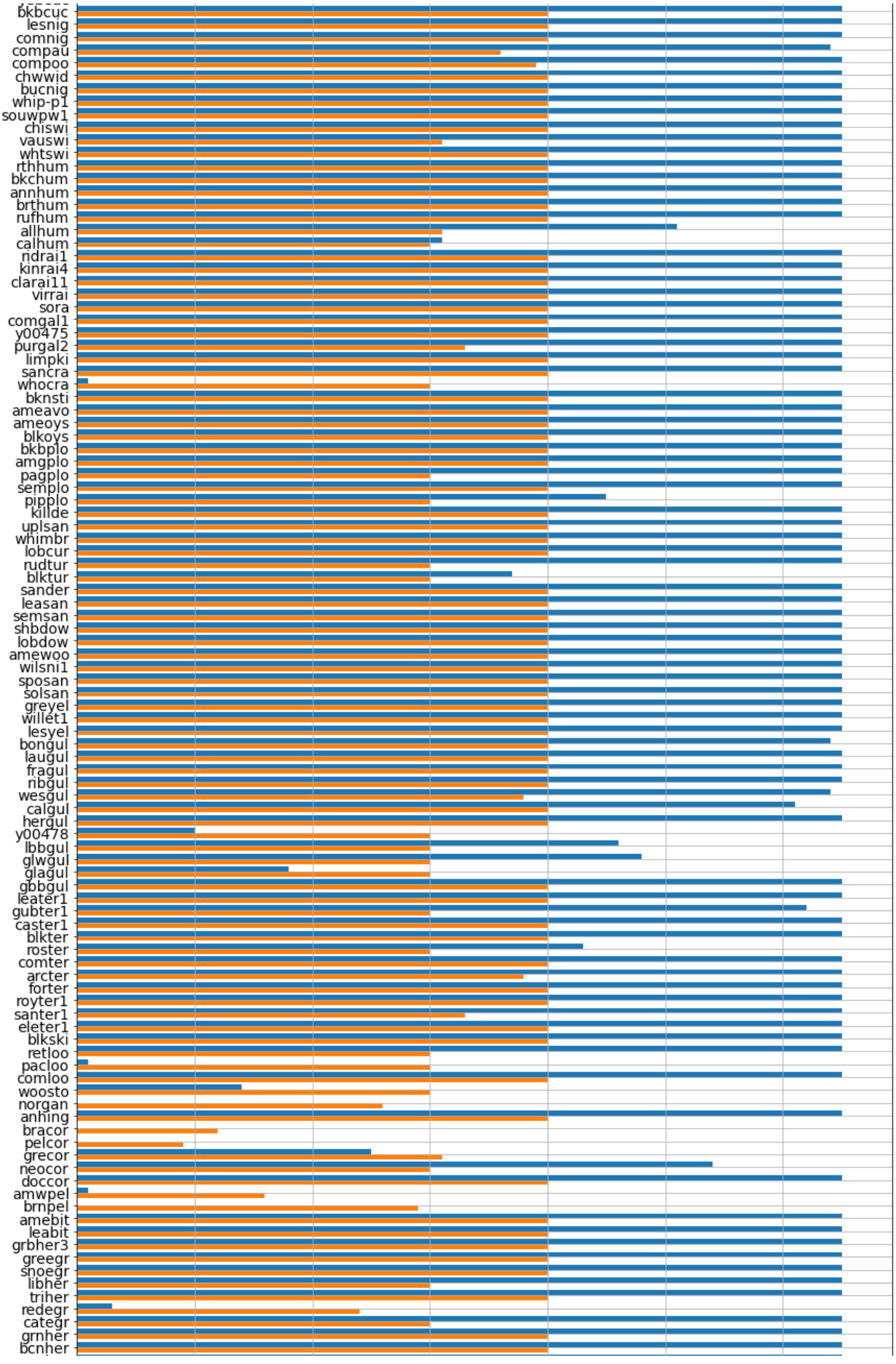
Plot Data!

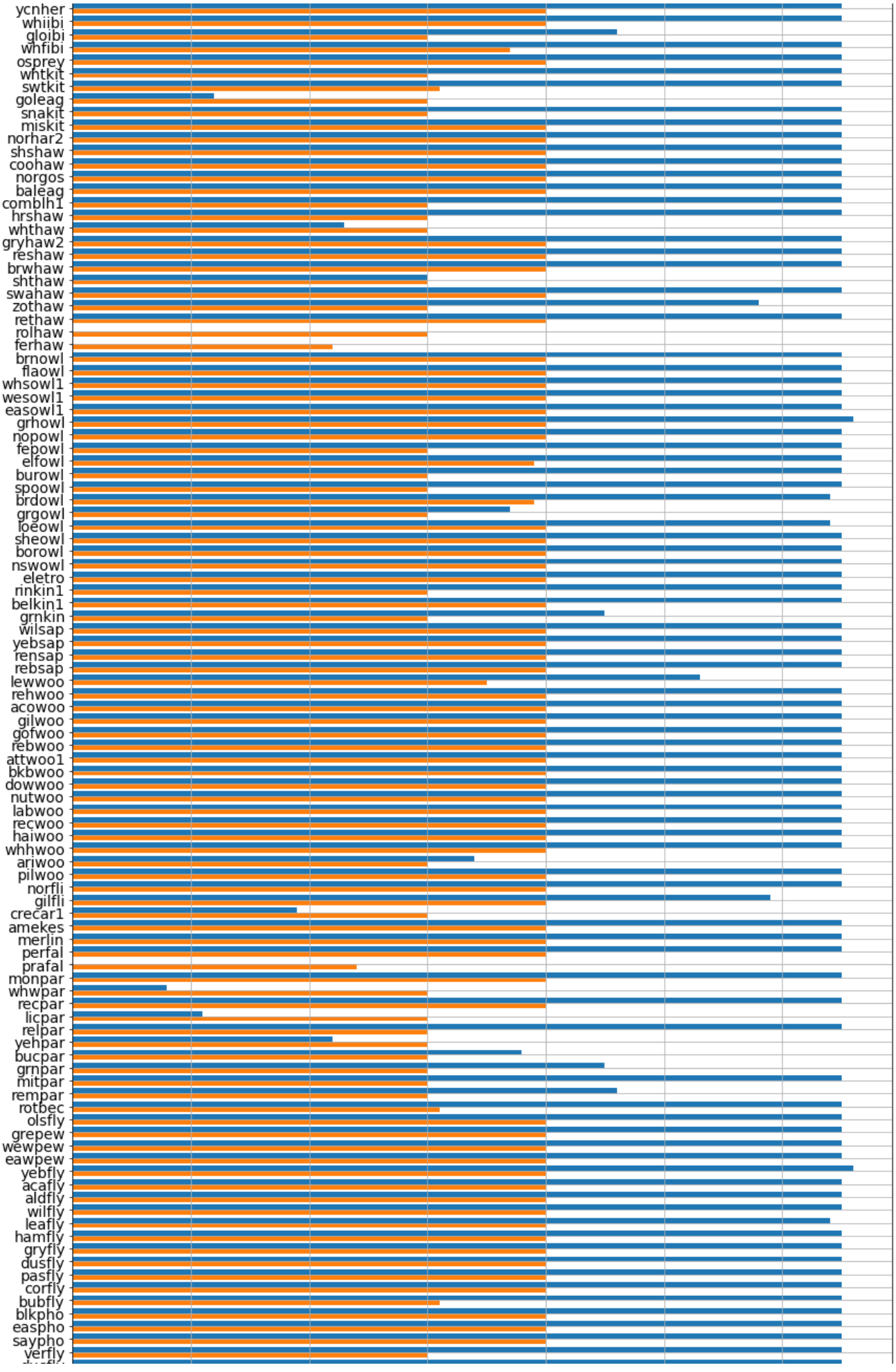
📈 dataset test / train asset counts per species: twin bar plot

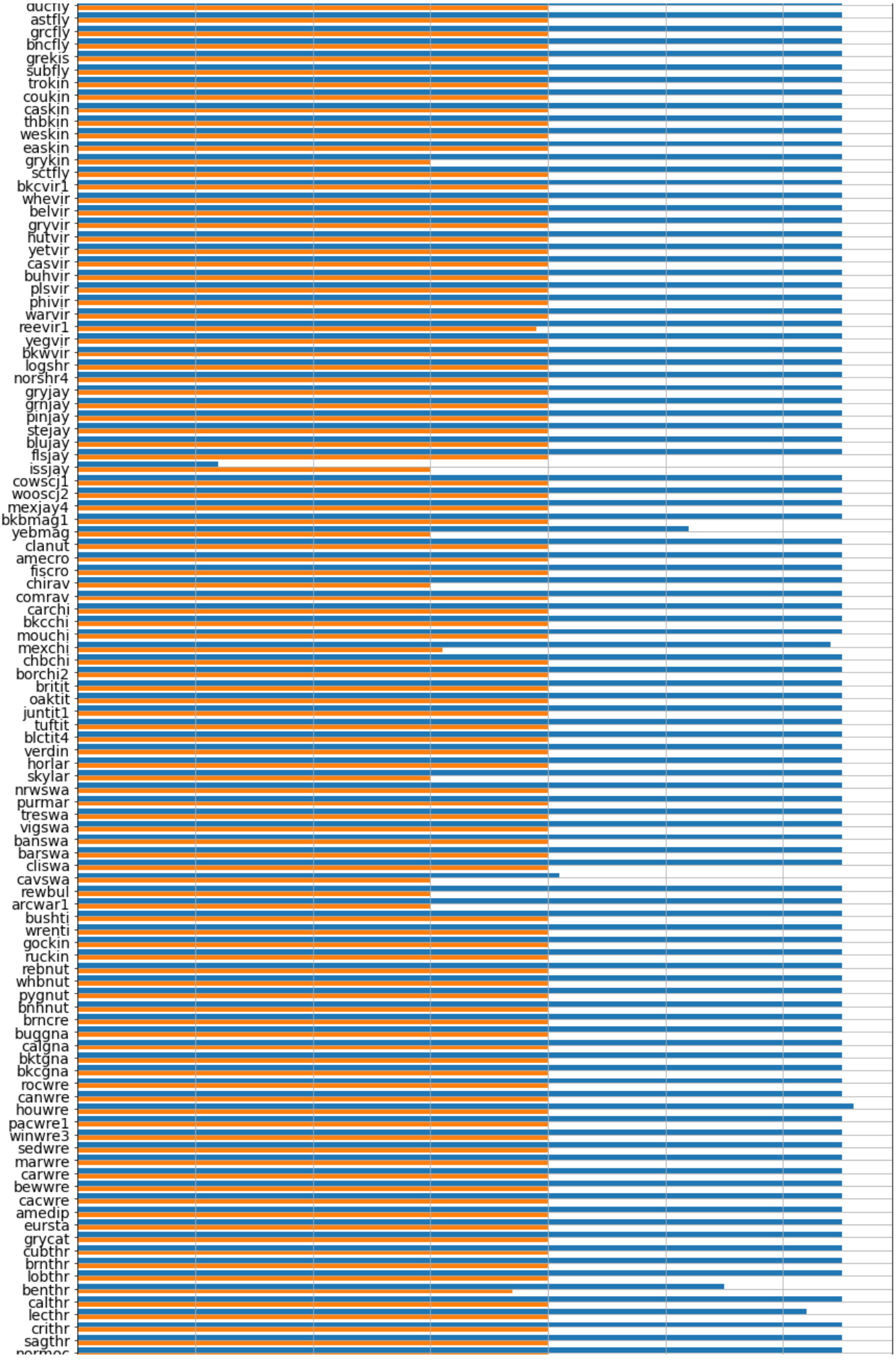
```
# plot number of train and test assets that were sampled for each of the target species
def plot_train_test_asset_counts(prev_assets_df=assets_df,
                                 prev_train_ids=prev_train_ids,
                                 prev_test_ids=prev_test_ids,
                                 title='Train / Test Split for Current Dataset'):
    prev_train_df = prev_assets_df[prev_assets_df['id'].isin(prev_train_ids)]
    prev_test_df = prev_assets_df[prev_assets_df['id'].isin(prev_test_ids)]
```

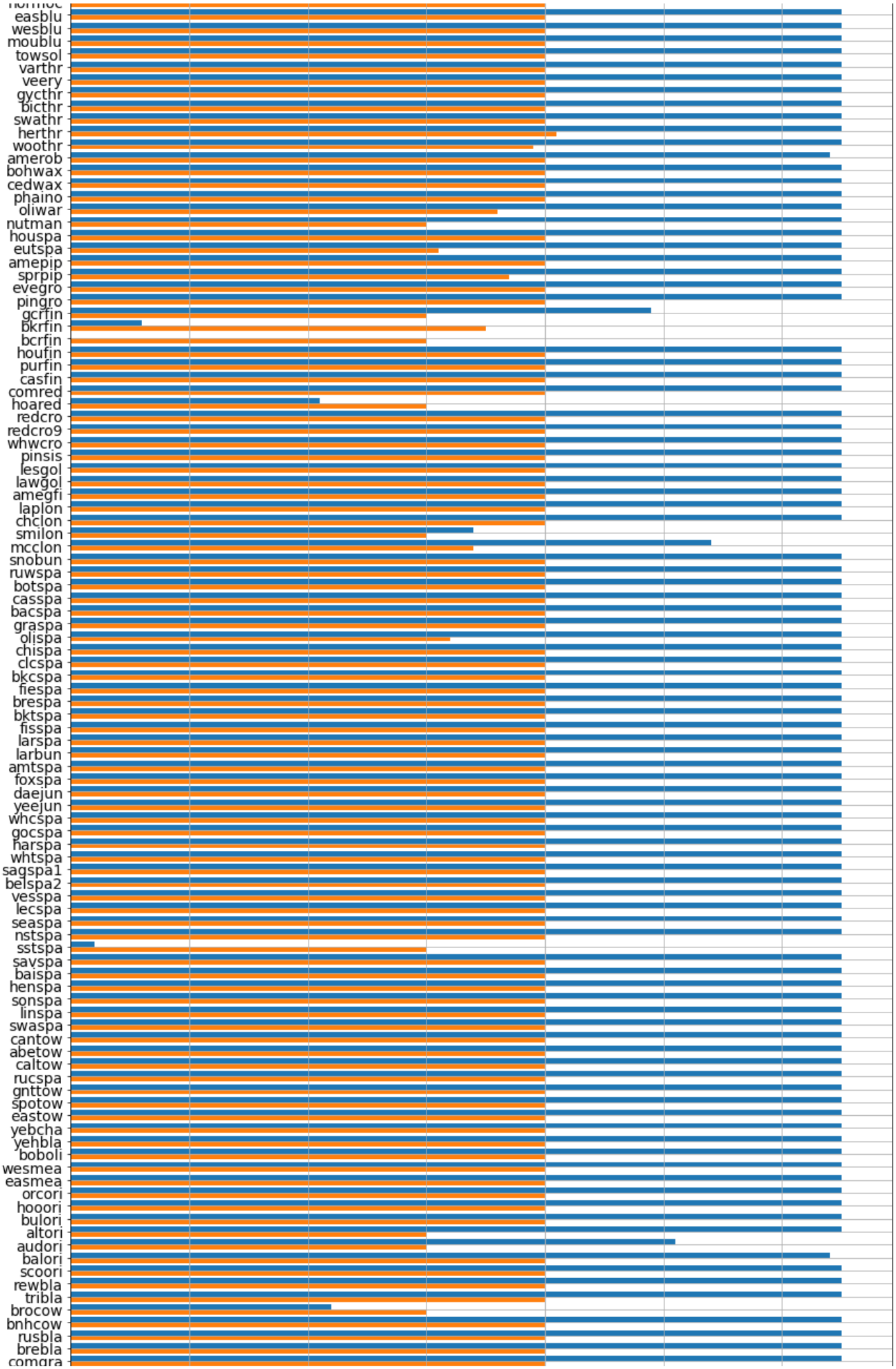
```
    train_counts  = []
    test_counts = []

    for s in target_species:
        species_df_test = prev_test_df[prev_test_df['reportAs']==s]
        species_df_train = prev_train_df[prev_train_df['reportAs']==s]
        train_counts.append(species_df_train.shape[0])
        test_counts.append(species_df_test.shape[0])

    labels = np.array(target_species)
    train_array = np.array(train_counts)
    test_array = np.array(test_counts)

    fig, ax = plt.subplots(1)
    fig.set_figwidth(14)
    fig.set_figheight(140)
    width =.4
    y_pos = np.arange(labels.shape[0])[::-1]

    ax.set_yticks(y_pos)
    ax.set_yticklabels(labels)

    plt.barh(y_pos + width/2, train_array, width, label='Train')
    plt.barh(y_pos - width/2, test_array, width, label='Test')

    plt.legend()
    plt.grid()
    plt.title(title)
    plt.xlabel('Asset count')
    plt.savefig('./output/TrainTestSplitForCurrentDataset.png')
    plt.savefig('./output/TrainTestSplitForCurrentDataset.pdf')

plot_train_test_asset_counts()
```

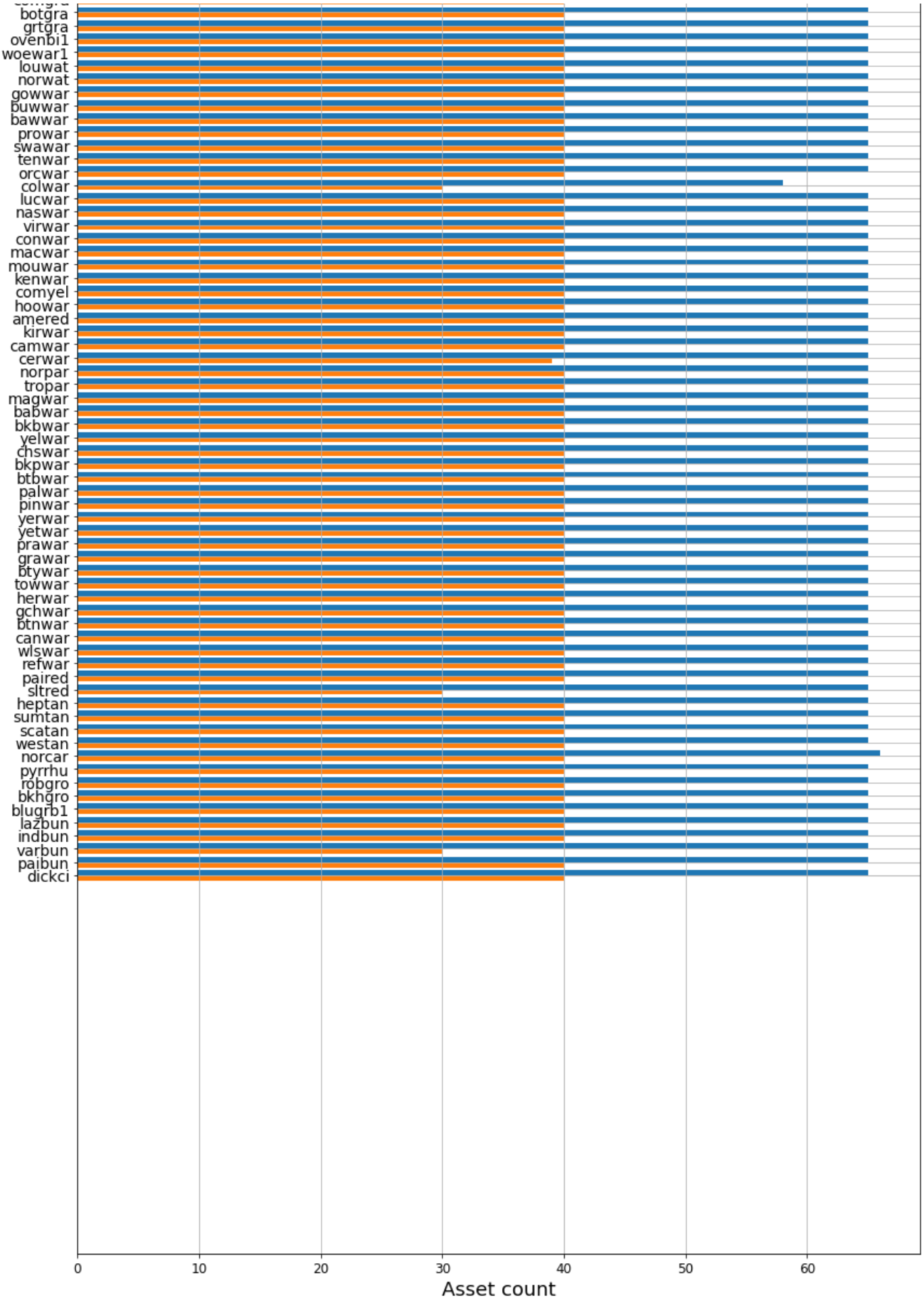

Train / Test Split for Current Dataset
Train
Test
bbwduc
fuwduc
snogoo
rosgoo
gwfgoo
brant
cacgoo1
cangoo
mutswa
truswa
tunswa
egygoo
musduc
wooduc
buwtea
cintea
norsho
gadwal
amewig
mallar3
mexduc
ambduc
motduc
norpin
gnwtea
canvas
redhea
rinduc
gresca
lessca
comeid
blksco2
lotduc
buffle
comgol
bargol
hoomer
commer
rebmer
rudduc
placha
mouqua
norbob
scaqua
calqua
gamqua
monqua
rinphe
grypar
rufgro
saggro
sprgro
dusgro
soogro1
shtgro
grpchi
wiltur
pibgre
horgre
eargre
rocpig
whcpig2
rebpig1
batpig1
eucdov
incdov
cogdov
rugdov
whtdov
whwdov
moudov
smbani
grbani
greroa
yebcuc


bkbcuc
lesnig
comnig
compau
compoo
chwwid
bucnig
whip-p1
souwpw1
chiswi
vauswi
whtswi
rthhum
bkchum
annhum
brthum
rufhum
allhum
calhum
ridrai1
kinrai4
clarai11
virrai
sora
comgal1
y00475
purgal2
limpki
sancra
whocra
bknsti
ameavo
ameoys
blkoys
bkbplo
amgplo
pagplo
semplo
pipplo
killde
uplsan
whimbr
lobcur
rudtur
blktur
sander
leasan
semsan
shbdow
lobdow
amewoo
wilsni1
sposan
solsan
greyel
willet1
lesyel
bongul
laugul
fragul
ribgul
wesgul
calgul
hergul
y00478
lbbgul
glwgul
glagul
gbbgul
leater1
gubter1
caster1
blkter
roster
comter
arcter
forter
royter1
santer1
eleter1
blkski
retloo
pacloo
comloo
woosto
norgan
anhing
bracor
pelcor
grecor
neocor
doccor
amwpel
brnpel
amebit
leabit
grbher3
greegr
snoegr
libher
triher
redegr
categr
grnher
bcnher


ycnher
whiibi
gloibi
whfibi
osprey
whtkit
swtkit
goleag
snakit
miskit
norhar2
shshaw
coohaw
norgos
baleag
comblh1
hrshaw
whthaw
gryhaw2
reshaw
brwhaw
shthaw
swahaw
zothaw
rethaw
rolhaw
ferhaw
brnowl
flaowl
whsowl1
wesowl1
easowl1
grhowl
nopowl
fepowl
elfowl
burowl
spoowl
brdowl
grgowl
loeowl
sheowl
borowl
nswowl
eletro
rinkin1
belkin1
grnkin
wilsap
yebsap
rensap
rebsap
lewwoo
rehwoo
acowoo
gilwoo
gofwoo
rebwoo
attwoo1
bkbwoo
dowwoo
nutwoo
labwoo
recwoo
haiwoo
whhwoo
ariwoo
pilwoo
norfli
gilfli
crecar1
amekes
merlin
perfal
prafal
monpar
whwpar
recpar
licpar
relpar
yehpar
bucpar
grnpar
mitpar
rempar
rotbec
olsfly
grepew
wewpew
eawpew
yebfly
acafly
aldfly
wilfly
leafly
hamfly
gryfly
dusfly
pasfly
corfly
bubfly
blkpho
easpho
saypho
verfly

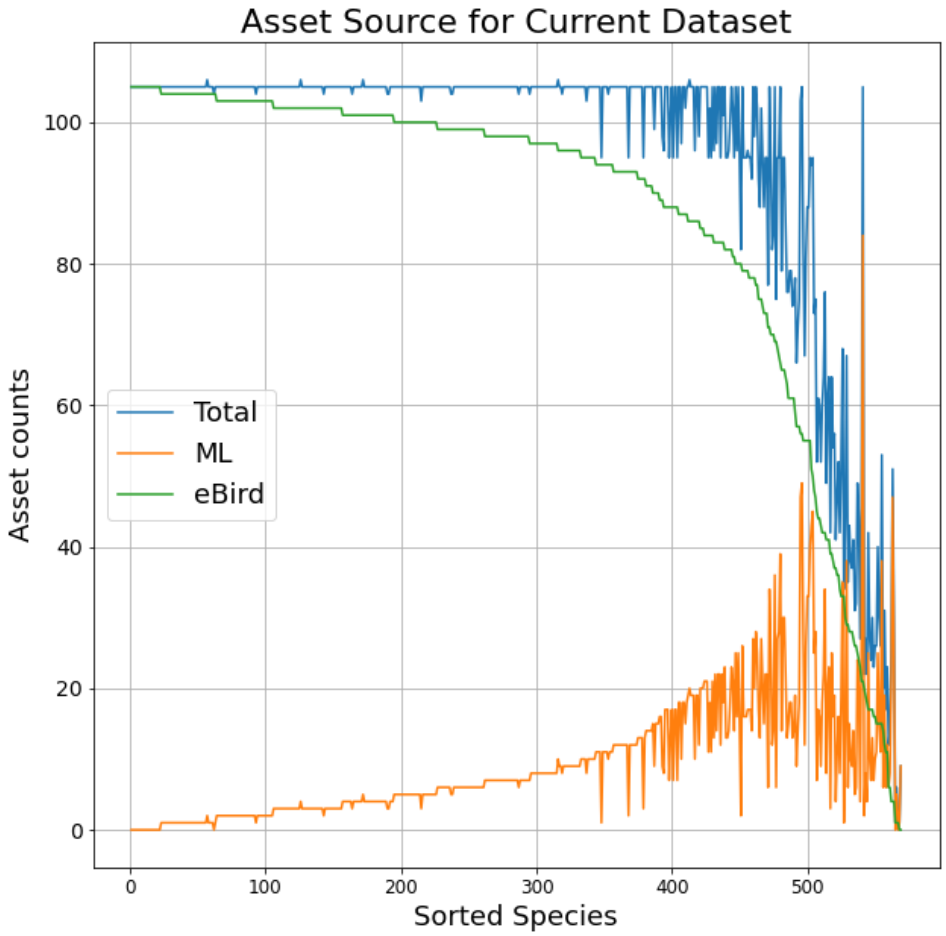

ducfly
astfly
grcfly
bncfly
grekis
subfly
trokin
coukin
caskin
thbkin
weskin
easkin
grykin
sctfly
bkcvir1
whevir
belvir
gryvir
hutvir
yetvir
casvir
buhvir
plsvir
phivir
warvir
reevir1
yegvir
bkwvir
logshr
norshr4
gryjay
grnjay
pinjay
stejay
blujay
flsjay
issjay
cowscj1
wooscj2
mexjay4
bkbmag1
yebmag
clanut
amecro
fiscro
chirav
comrav
carchi
bkcchi
mouchi
mexchi
chbchi
borchi2
britit
oaktit
juntit1
tuftit
blctit4
verdin
horlar
skylar
nrwswa
purmar
treswa
vigswa
banswa
barswa
cliswa
cavswa
rewbul
arcwar1
bushti
wrenti
gockin
ruckin
rebnut
whbnut
pygnut
bnhnut
brncre
buggna
calgna
bktgna
bkcgna
rocwre
canwre
houwre
pacwre1
winwre3
sedwre
marwre
carwre
bewwre
cacwre
amedip
eursta
grycat
cubthr
brnthr
lobthr
benthr
calthr
lecthr
crithr
sagthr


easblu
wesblu
moublu
towsol
varthr
veery
gycthr
bicthr
swathr
herthr
woothr
amerob
bohwax
cedwax
phaino
oliwar
nutman
houspa
eutspa
amepip
sprpip
evegro
pingro
gcrfin
bkrfin
bcrfin
houfin
purfin
casfin
comred
hoared
redcro
redcro9
whwcro
pinsis
lesgol
lawgol
amegfi
laplon
chclon
smilon
mcclon
snobun
ruwspa
botspa
casspa
bacspa
graspa
olispa
chispa
clcspa
bkcspa
fiespa
brespa
bktspa
fisspa
larspa
larbun
amtspa
foxspa
daejun
yeejun
whcspa
gocspa
harspa
whtspa
sagspa1
belspa2
vesspa
lecspa
seaspa
nstspa
sstspa
savspa
baispa
henspa
sonspa
linspa
swaspa
cantow
abetow
caltow
rucspa
gnttow
spotow
eastow
yebcha
yehbla
boboli
wesmea
easmea
orcori
hooori
bulori
altori
audori
balori
scoori
rewbla
tribla
brocow
bnhcow
rusbla
brebla
comgra



📈 dataset eBird / ML asset counts: line plot

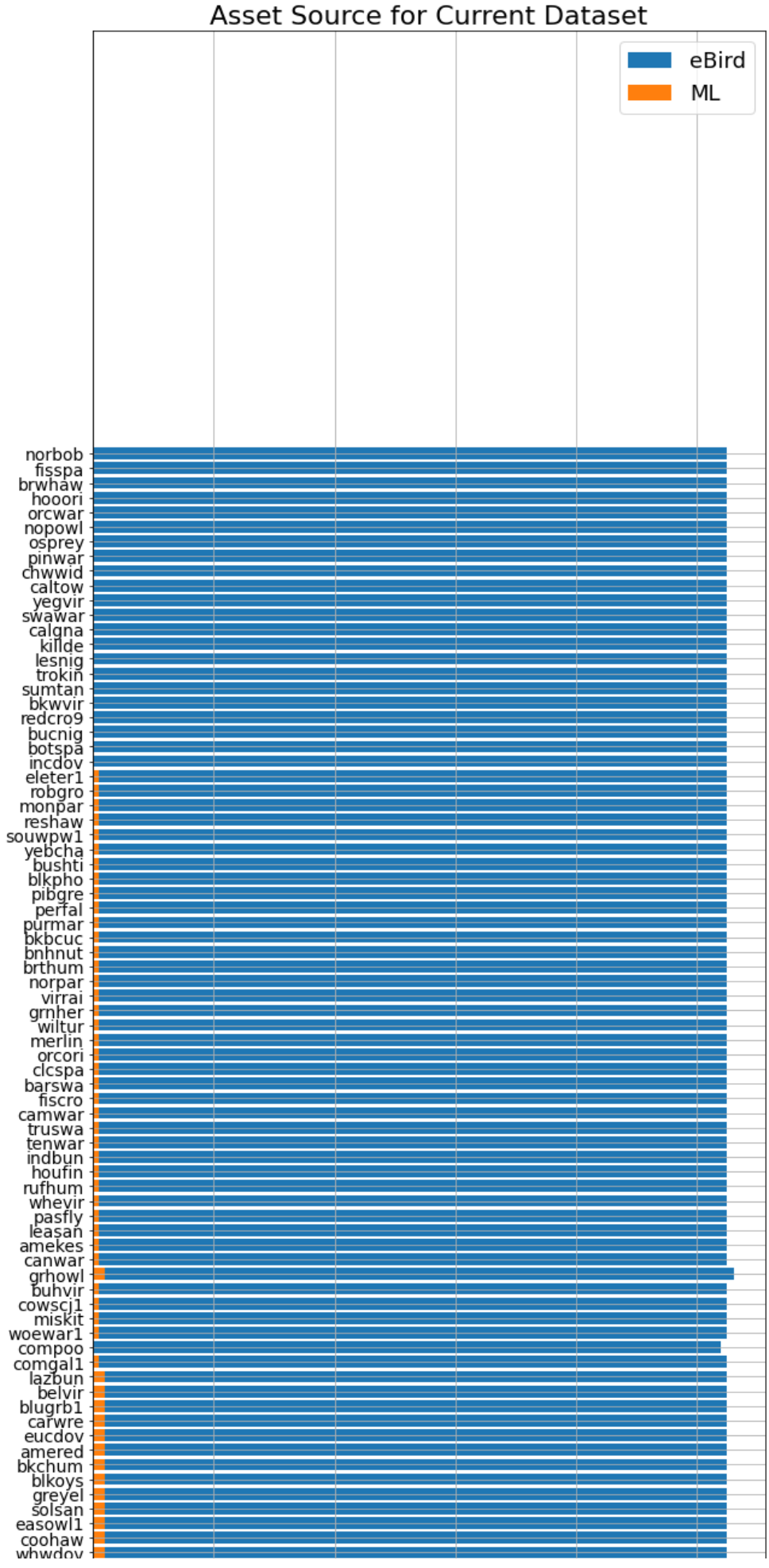
```
def chart_totals(assets_df=assets_df,
                 title='Asset Source for Current Dataset'):

    """
    3 lines;
    - number of assets sampled from eBird
    - number of assets sampled from ML
    - total assets
    """

    target_species = assets_df['reportAs'].unique()
    ml_counts  = []
    ebird_counts = []

    for s in target_species:
        species_df = assets_df[assets_df['reportAs']==s]
        ml_counts.append(species_df[species_df['source']=='ml'].shape[0])
        ebird_counts.append(species_df[species_df['source']=='ebird'].shape[0])

    labels = np.array(target_species)
    ebird_array = np.array(ebird_counts)
    ml_array = np.array(ml_counts)

    # sort based off ml_array:
    idx = np.argsort(ebird_array)[::-1]

    labels = labels[idx]
    ebird_array = ebird_array[idx]
    ml_array = ml_array[idx]
    total_array = ebird_array + ml_array
    fig1 = plt.figure(figsize=(10,10))
    plt.plot(np.arange(labels.shape[0])+1, total_array, label='Total')
    plt.plot(np.arange(labels.shape[0])+1, ml_array, label='ML')
    plt.plot(np.arange(labels.shape[0])+1, ebird_array, label='eBird')

    #plt.loglog()
    plt.grid()
    plt.legend()
    plt.xlabel('Sorted Species')
    plt.ylabel('Asset counts')
    plt.title(title)
    plt.savefig('./output/AssetSourceForCurrentDatasetLine.png')
    plt.savefig('./output/AssetSourceForCurrentDatasetLine.pdf')


chart_totals()
```

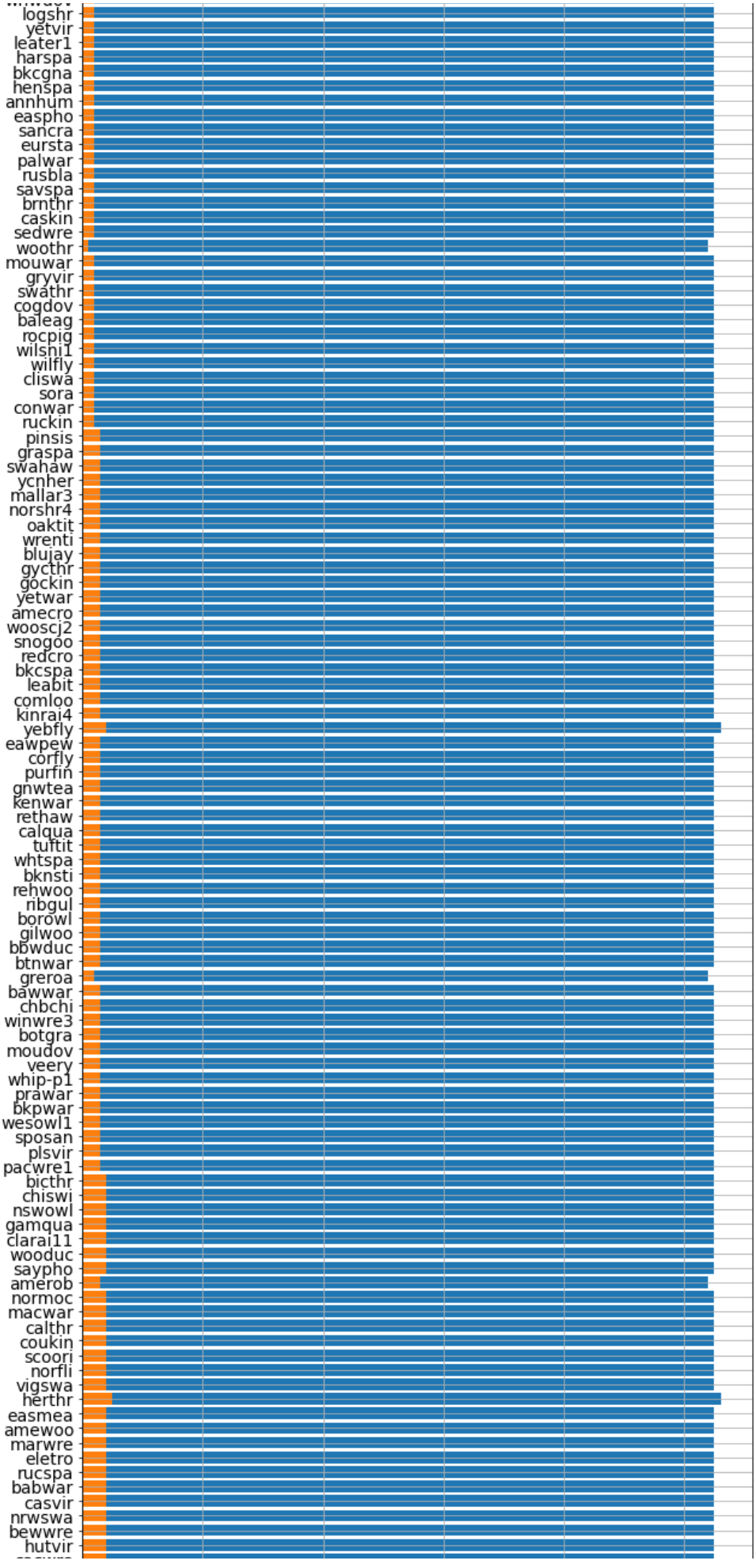


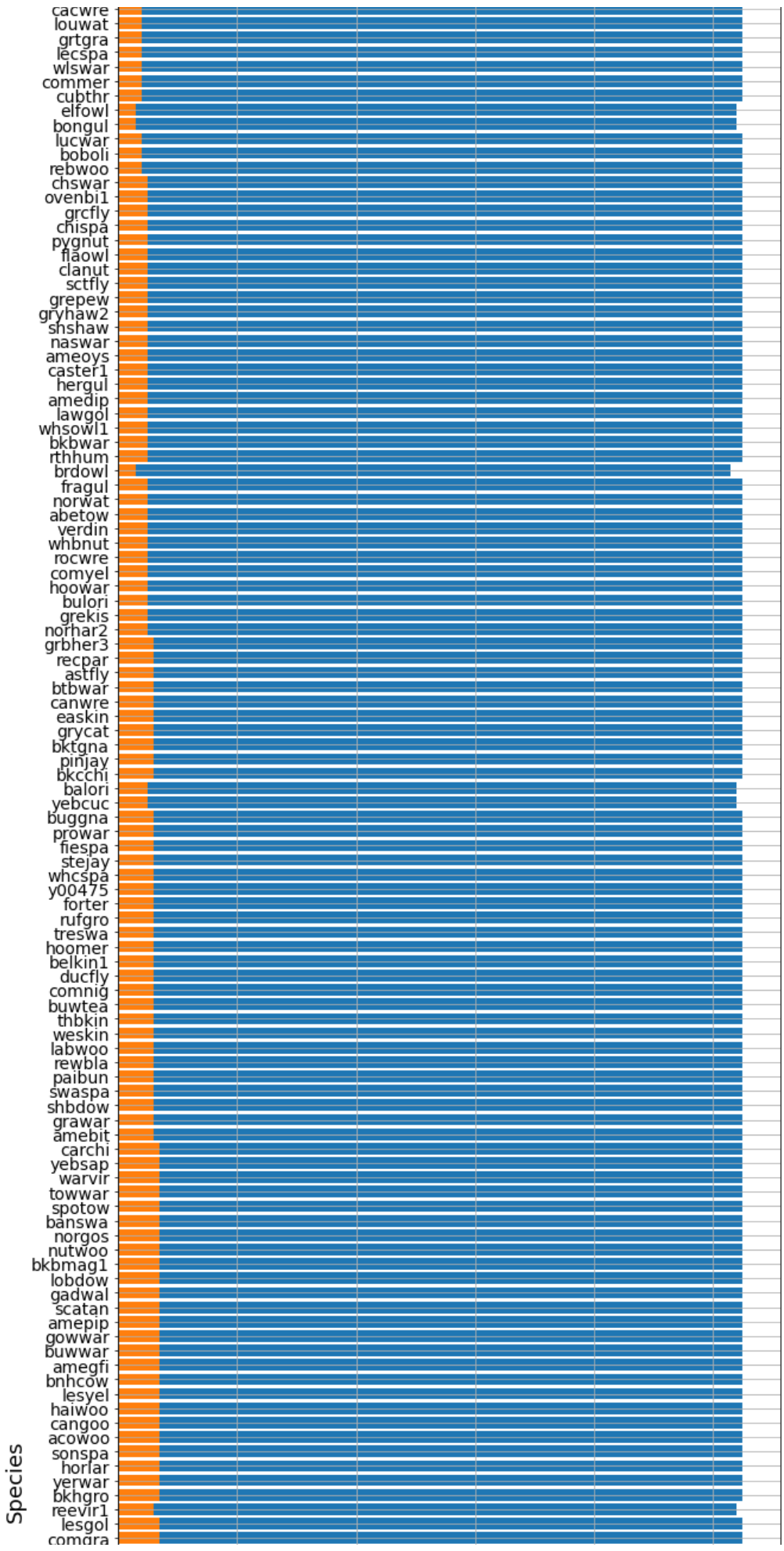
📈 dataset eBird / ML asset counts: inline bar plot

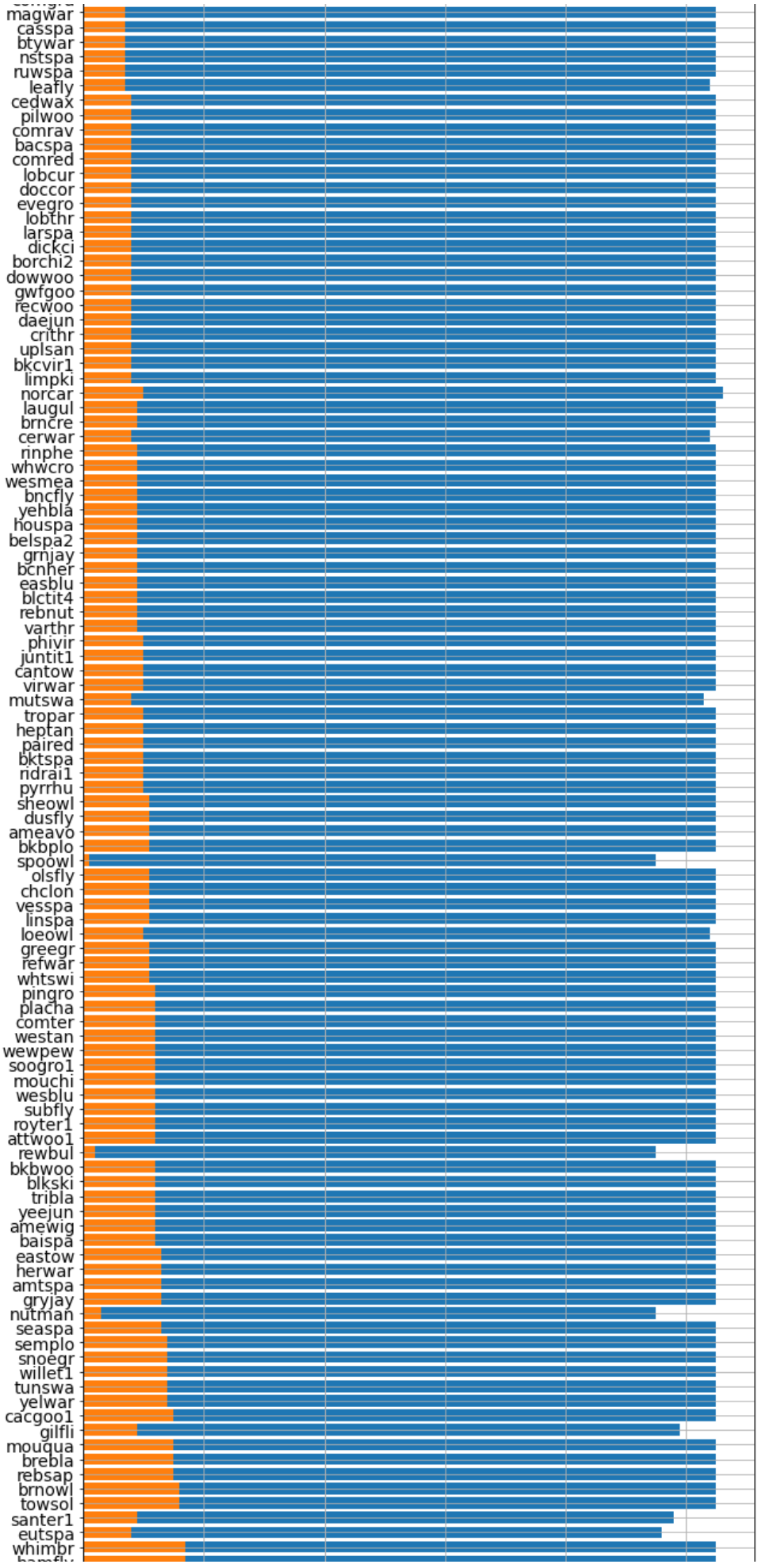
```
def bar_totals(assets_df=assets_df,
               title='Asset Source for Current Dataset'):

    """
    2 bars;
    - number of assets sampled from eBird
    - number of assets sampled from ML
    """

    target_species = assets_df['reportAs'].unique()
    ml_counts  = []
    ebird_counts = []

    for s in target_species:
        species_df = assets_df[assets_df['reportAs']==s]
        ml_counts.append(species_df[species_df['source']=='ml'].shape[0])
        ebird_counts.append(species_df[species_df['source']=='ebird'].shape[0])

    labels = np.array(target_species)
    ebird_array = np.array(ebird_counts)
    ml_array = np.array(ml_counts)

    # sort based off ml_array:
    idx = np.argsort(ebird_array)

    labels = labels[idx]
    ebird_array = ebird_array[idx]
    ml_array = ml_array[idx]
    total_array = ebird_array + ml_array
```

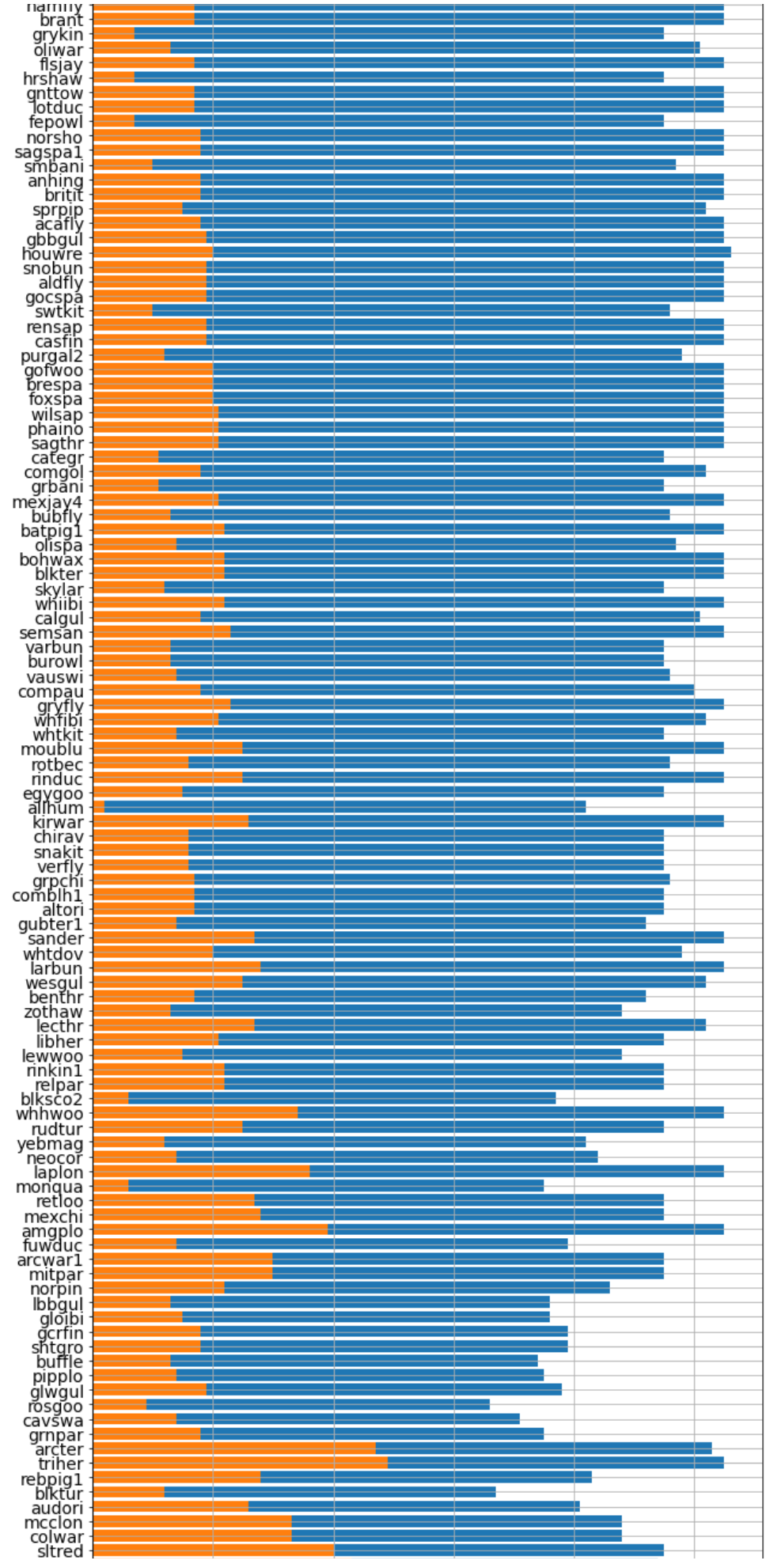
```
    fig, ax = plt.subplots()
    fig.set_figwidth(10)
    fig.set_figheight(140)
    #ax.set_xticklabels(labels)

    plt.barh(labels, total_array, label='eBird')
    plt.barh(labels, ml_array, label='ML')
    #plt.barh(labels, ebird_array, label='Missing Assets')

    plt.grid()
    plt.legend()
    plt.title(title)
    plt.ylabel('Species')
    plt.xlabel('Asset counts by Source')
    plt.savefig('./output/AssetSourceForCurrentDatasetSplitBar.png')
    plt.savefig('./output/AssetSourceForCurrentDatasetSplitBar.pdf')

bar_totals()
```

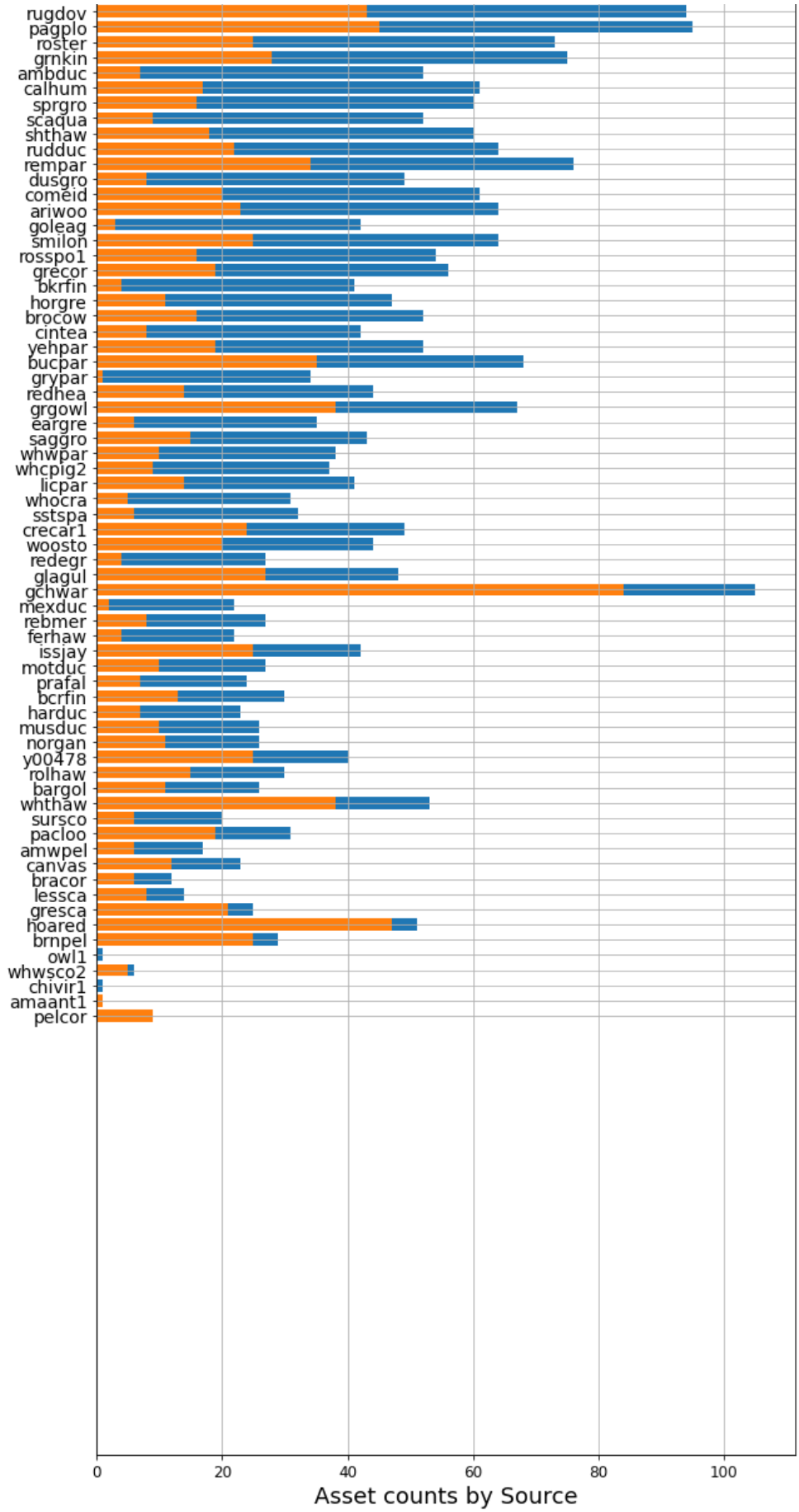

Asset Source for Current Dataset
eBird
ML
norbob
fisspa
brwhaw
hoori
orcwar
nopowl
osprey
pinwar
chwwid
caltow
yegvir
swawar
calgna
killde
lesnig
trokin
sumtan
bkwvir
redcro9
bucnig
botspa
incdov
eleter1
robgro
monpar
reshaw
souwpw1
yebcha
bushti
blkpho
pibgre
perfal
purmar
bkbcuc
bnhnut
brthum
norpar
virrai
grnher
wiltur
merlin
orcori
clcspa
barswa
fiscro
camwar
truswa
tenwar
indbun
houfin
rufhum
whevir
pasfly
leasan
amekes
canwar
grhowl
buhvir
cowscj1
miskit
woewar1
compoo
comgal1
lazbun
belvir
blugrb1
carwre
eucdov
amered
bkchum
blkoys
greyel
solsan
easowl1
coohaw
whwdov


logshr
yetvir
leater1
harspa
bkcgna
henspa
annhum
easpho
sancra
eursta
palwar
rusbla
savspa
brnthr
caskin
sedwre
woothr
mouwar
gryvir
swathr
cogdov
baleag
rocpig
wilsni1
wilfly
cliswa
sora
conwar
ruckin
pinsis
graspa
swahaw
ycnher
mallar3
norshr4
oaktit
wrenti
blujay
gycthr
gockin
yetwar
amecro
wooscj2
snogoo
redcro
bkcspa
leabit
comloo
kinrai4
yebfly
eawpew
corfly
purfin
gnwtea
kenwar
rethaw
calqua
tuftit
whtspa
bknsti
rehwoo
ribgul
borowl
gilwoo
bbwduc
btnwar
greroa
bawwar
chbchi
winwre3
botgra
moudov
veery
whip-p1
prawar
bkpwar
wesowl1
sposan
plsvir
pacwre1
bicthr
chiswi
nswowl
gamqua
clarai11
wooduc
saypho
amerob
normoc
macwar
calthr
coukin
scoori
norfli
vigswa
herthr
easmea
amewoo
marwre
eletro
rucspa
babwar
casvir
nrwswa
bewwre
hutvir


Species
cacwre
louwat
grtgra
lecspa
wlswar
commer
cubthr
elfowl
bongul
lucwar
boboli
rebwoo
chswar
ovenbi1
grcfly
chispa
pygnut
flaowl
clanut
sctfly
grepew
gryhaw2
shshaw
naswar
ameoys
caster1
hergul
amedip
lawgol
whsowl1
bkbwar
rthhum
brdowl
fragul
norwat
abetow
verdin
whbnut
rocwre
comyel
hoowar
bulori
grekis
norhar2
grbher3
recpar
astfly
btbwar
canwre
easkin
grycat
bktgna
pinjay
bkcchi
balori
yebcuc
buggna
prowar
fiespa
stejay
whcspa
y00475
forter
rufgro
treswa
hoomer
belkin1
ducfly
comnig
buwtea
thbkin
weskin
labwoo
rewbla
paibun
swaspa
shbdow
grawar
amebit
carchi
yebsap
warvir
towwar
spotow
banswa
norgos
nutwoo
bkbmag1
lobdow
gadwal
scatan
amepip
gowwar
buwwar
amegfi
bnhcow
lesyel
haiwoo
cangoo
acowoo
sonspa
horlar
yerwar
bkhgro
reevir1
lesgol
comgra

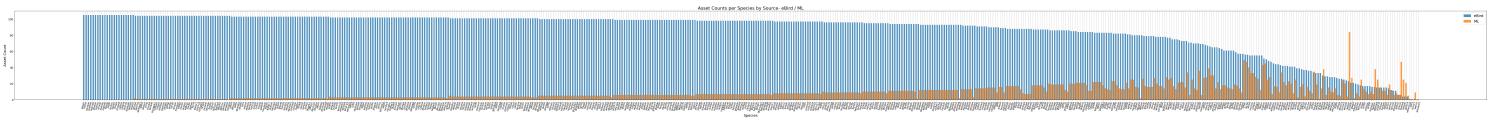

magwar
casspa
btywar
nstspa
ruwspa
leafly
cedwax
pilwoo
comrav
bacspa
comred
lobcur
doccor
evegro
lobthr
larspa
dickci
borchi2
dowwoo
gwfgoo
recwoo
daejun
crithr
uplsan
bkcvir1
limpki
norcar
laugul
brncre
cerwar
rinphe
whwcro
wesmea
bncfly
yehbla
houspa
belspa2
grnjay
bcnher
easblu
blctit4
rebnut
varthr
phivir
juntit1
cantow
virwar
mutswa
tropar
heptan
paired
bktspa
ridrai1
pyrrhu
sheowl
dusfly
ameavo
bkbplo
spoowl
olsfly
chclon
vesspa
linspa
loeowl
greegr
refwar
whtswi
pingro
placha
comter
westan
wewpew
soogro1
mouchi
wesblu
subfly
royter1
attwoo1
rewbul
bkbwoo
blkski
tribla
yeejun
amewig
baispa
eastow
herwar
amtspa
gryjay
nutman
seaspa
semplo
snoegr
willet1
tunswa
yelwar
cacgoo1
gilfli
mouqua
brebla
rebsap
brnowl
towsol
santer1
eutspa
whimbr


hamfly
brant
grykin
oliwar
fisjay
hrshaw
gnttow
lotduc
fepowl
norsho
sagspa1
smbani
anhing
britit
sprpip
acafly
gbbgul
houwre
snobun
aldfly
gocspa
swtkit
rensap
casfin
purgal2
gofwoo
brespa
foxspa
wilsap
phaino
sagthr
categr
comgol
grbani
mexjay4
bubfly
batpig1
olispa
bohwax
blkter
skylar
whiibi
calgul
semsan
varbun
burowl
vauswi
compau
gryfly
whfibi
whtkit
moublu
rotbec
rinduc
egygoo
alfhum
kirwar
chirav
snakit
verfly
grpchi
comblh1
altori
gubter1
sander
whtdov
larbun
wesgul
benthr
zothaw
lecthr
libher
lewwoo
rinkin1
relpar
blksco2
whhwoo
rudtur
yebmag
neocor
laplon
monqua
retloo
mexchi
amgplo
fuwduc
arcwar1
mitpar
norpin
lbbgul
gloibi
gcrfin
shtgro
buffle
pipplo
glwgul
rosgoo
cavswa
grnpar
arcter
triher
rebpig1
blktur
audori
mcclon
colwar
sltred



📈 dataset eBird / ML asset counts: horizontal bar plot

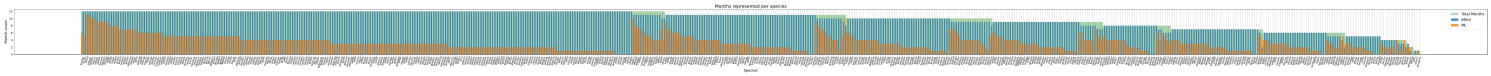
```
"""
2 bars per species: the number of assets sampled from eBird, and the number of assets sampled from ML.
"""

target_species = assets_df['reportAs'].unique()
ml_counts  = []
ebird_counts = []

for s in target_species:
    species_df = assets_df[assets_df['reportAs']==s]
    ml_counts.append(species_df[species_df['source']=='ml']['id'].unique().shape[0])
    ebird_counts.append(species_df[species_df['source']=='ebird']['id'].unique().shape[0])

ebird_counts, ml_counts, target_species = map(np.array, zip(*sorted(zip(ebird_counts, ml_counts, target_species))))

ebird_counts = np.array(ebird_counts)
ml_counts = np.array(ml_counts)

y_pos = np.arange(target_species.shape[0])[::-1] # have the "first" category be at the "top"
width = 0.6
fig1 = plt.figure(figsize=(130,8))
ax1 = fig1.add_subplot(1,1,1)

# ax3.axes.xaxis.set_visible(False)

#rects3 = ax3.bar(y_pos, both_counts, width*2, label='Total Months', color='green', alpha=0.3)   # test "below"
rects1 = ax1.bar(y_pos + width/2, ebird_counts, width, label='eBird', alpha=0.8) # train "first"
rects2 = ax1.bar(y_pos - width/2, ml_counts, width, label='ML', alpha=0.8)   # test "below"

ax1.set_ylabel('Asset Count')

ax1.set_xlabel('Species')
ax1.set_title('Asset Counts per Species by Source- eBird / ML')
ax1.set_xticks(y_pos)
plt.xticks(rotation = 75)
ax1.set_xticklabels(target_species)
ax1.legend()
ax1.grid(axis='x')
plt.savefig('./output/AssetCountsPerSpeciesBySourceHorzDualBar.png')
plt.savefig('./output/AssetCountsPerSpeciesBySourceHorzDualBar.pdf')
```



📈 unique Months by Species

```
"""
1 bar per species: the number of unique months represented in the selected assets (max is 12)
"""
target_species = assets_df['reportAs'].unique()
months_counts_ebird = []
months_counts_ml = []
months_counts_both = []

for s in target_species:
    species_df = assets_df[assets_df['reportAs']==s]
    months_counts_ebird.append(species_df[species_df['source']=='ebird']['obsMonth'].unique().shape[0])
    months_counts_ml.append(species_df[species_df['source']=='ml']['obsMonth'].unique().shape[0])
    months_counts_both.append(species_df['obsMonth'].unique().shape[0])

months_counts_both, months_counts_ebird, months_counts_ml, target_species = map(np.array, zip(*sorted(zip(months_count
np
```

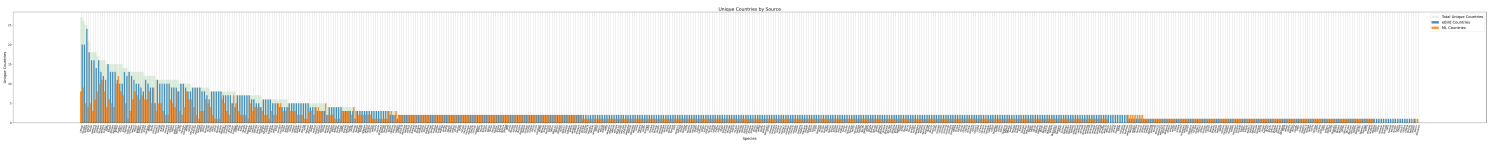
```
ebird_counts = np.array(months_counts_ebird).clip(max=12)
ml_counts = np.array(months_counts_ml).clip(max=12)
both_counts = np.array(months_counts_both).clip(max=12)

y_pos = np.arange(target_species.shape[0])[::-1] # have the "first" category be at the "top"
width = 0.6
fig3 = plt.figure(figsize=(130,4))
ax3 = fig3.add_subplot(1,1,1)

# ax3.axes.xaxis.set_visible(False)

rects3 = ax3.bar(y_pos, both_counts, width*2, label='Total Months', color='green', alpha=0.3)   # test "below"
rects1 = ax3.bar(y_pos + width/2, ebird_counts, width, label='eBird', alpha=0.8) # train "first"
rects2 = ax3.bar(y_pos - width/2, ml_counts, width, label='ML', alpha=0.8)   # test "below"

ax3.set_ylabel('Month count')

ax3.set_xlabel('Species')
ax3.set_title('Months represented per species')
ax3.set_xticks(y_pos)
plt.xticks(rotation = 75)
ax3.set_xticklabels(target_species)
ax3.legend()
ax3.grid(axis='x')
plt.savefig('./output/MonthsRepresentedPerSpecies.png')
plt.savefig('./output/MonthsRepresentedPerSpecies.pdf')
```



📈 unique Countries by Species

```
"""
bar plot of the country values (in aggregate as well as per species)
"""
target_species = assets_df['reportAs'].unique()
months_counts_ebird = []
months_counts_ml = []
months_counts_both = []

for s in target_species:
    species_df = assets_df[assets_df['reportAs']==s]
    months_counts_ebird.append(species_df[species_df['source']=='ebird']['countryName'].unique().shape[0])
    months_counts_ml.append(species_df[species_df['source']=='ml']['countryName'].unique().shape[0])
    months_counts_both.append(species_df['countryName'].unique().shape[0])

months_counts_both, months_counts_ebird, months_counts_ml, target_species = map(np.array, zip(*sorted(zip(months_count

ebird_counts = np.array(months_counts_ebird)
ml_counts = np.array(months_counts_ml)
both_counts = np.array(months_counts_both)

y_pos = np.arange(target_species.shape[0])[::-1] # have the "first" category be at the "top"
width = 0.6
fig3 = plt.figure(figsize=(130,10))
ax3 = fig3.add_subplot(1,1,1)

# ax3.axes.xaxis.set_visible(False)

rects3 = ax3.bar(y_pos, both_counts, width*2, label='Total Unique Countries', color='green', alpha=0.1)   # test "belo
rects1 = ax3.bar(y_pos + width/2, ebird_counts, width, label='eBird Countries', alpha=0.8) # train "first"
rects2 = ax3.bar(y_pos - width/2, ml_counts, width, label='ML Countries', alpha=0.8)   # test "below"

ax3.set_ylabel('Unique Countries')
```

```
ax3.set_xlabel('Species')
ax3.set_title('Unique Countries by Source')
ax3.set_xticks(y_pos)
plt.xticks(rotation = 75)
ax3.set_xticklabels(target_species)
ax3.legend()
ax3.grid(axis='x')
plt.savefig('./output/UniqueCountriesBySource.png')
plt.savefig('./output/UniqueCountriesBySource.pdf')
```



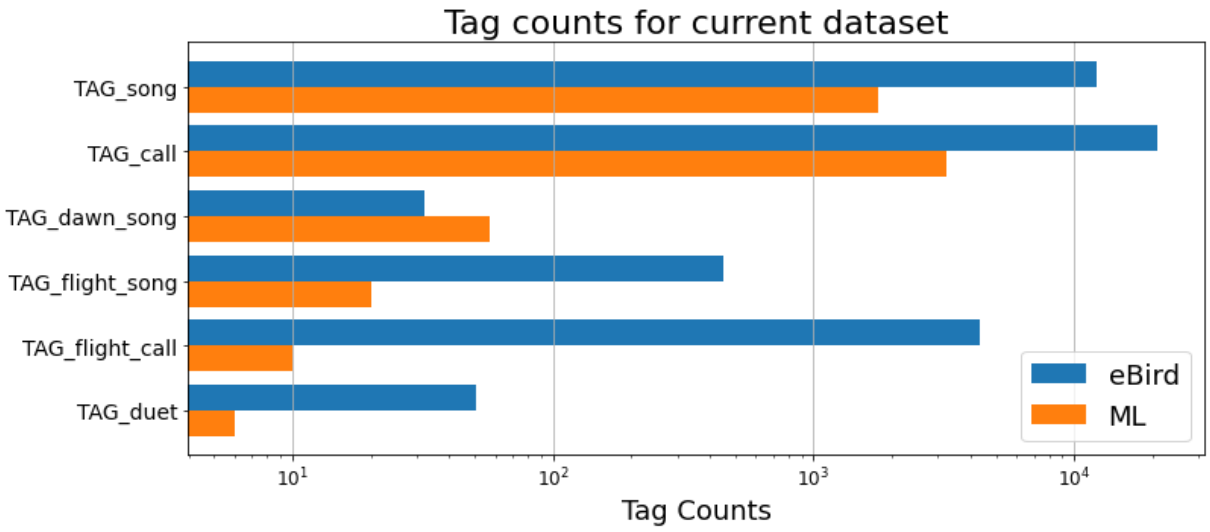
📈 unique Checklists & Users by Species

```
"""
2 bars per species: the number of unique checklists, and the number of unique users.
"""

target_species = assets_df['reportAs'].unique()
userId_counts  = []
subId_counts = []

for s in target_species:
    species_df = assets_df[assets_df['reportAs']==s]
    userId_counts.append(species_df['userId'].unique().shape[0])
    subId_counts.append(species_df['subId'].unique().shape[0])

subId_counts, userId_counts, target_species = map(np.array, zip(*sorted(zip(subId_counts, userId_counts, target_specie

subId_counts = np.array(subId_counts)
userId_counts = np.array(userId_counts)

y_pos = np.arange(target_species.shape[0])[::-1] # have the "first" category be at the "top"
width = 0.6
fig2 = plt.figure(figsize=(130,4))
ax2 = fig2.add_subplot(1,1,1)

# ax3.axes.xaxis.set_visible(False)

#rects3 = ax3.bar(y_pos, both_counts, width*2, label='Total Months', color='green', alpha=0.3)   # test "below"
rects1 = ax2.bar(y_pos + width/2, subId_counts, width, label='Checklists', alpha=0.8) # train "first"
rects2 = ax2.bar(y_pos - width/2, userId_counts, width, label='Users', alpha=0.8)   # test "below"

ax2.set_ylabel('# Unique')

ax2.set_xlabel('Species')
ax2.set_title('Unique Checklists & Users per Species')
ax2.set_xticks(y_pos)
plt.xticks(rotation = 75)
ax2.set_xticklabels(target_species)
ax2.legend()
ax2.grid(axis='x')
plt.savefig('./nb_exports/UniqueChecklistsUsersPerSpecies.png')
```

📈 tag Counts for current dataset

```
def tag_stuff(df=assets_df, title=''):

    preferred_tags = [
```

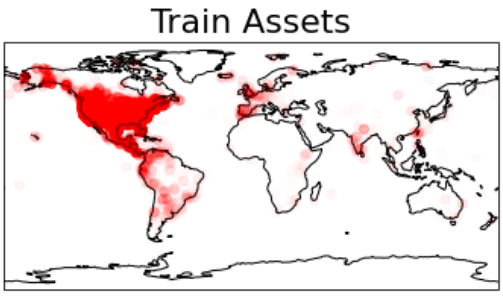
```
        'TAG_song',
        'TAG_call',
        'TAG_dawn_song',
        'TAG_flight_song',
        'TAG_flight_call',
        'TAG_duet'
    ]

    target_species = df['reportAs'].unique()

    preferred_tag_counts_ebird = []
    preferred_tag_counts_ml = []

    for tag in preferred_tags:
        preferred_tag_counts_ebird.append(df[df['source']=='ebird'][tag].sum())
        preferred_tag_counts_ml.append(df[df['source']!='ebird'][tag].sum())

    ebird_counts = np.array(preferred_tag_counts_ebird)
    ml_counts = np.array(preferred_tag_counts_ml)
    labels = np.array(preferred_tags)
    y_pos = np.arange(labels.shape[0])[::-1] # have the "first" category be at the "top"
    width = 0.4
    fig = plt.figure(figsize=(12, 5))
    ax = fig.add_subplot(1,1,1)
    rects1 = ax.barh(y_pos + width/2, ebird_counts, width, label='eBird') # train "first"
    rects2 = ax.barh(y_pos - width/2, ml_counts, width, label='ML')   # test "below"
    ax.set_xlabel('Tag Counts')
    ax.set_title(title)
    ax.set_yticks(y_pos)
    ax.set_yticklabels(labels)
    ax.legend()
    ax.semilogx()
    ax.grid(axis='x')
    plt.savefig('./output/TagCountsCurrentDatasetBySource.png')
    plt.savefig('./output/TagCountsCurrentDatasetBySource.pdf')

tag_stuff(df=assets_df, title='Tag counts for current dataset')
```

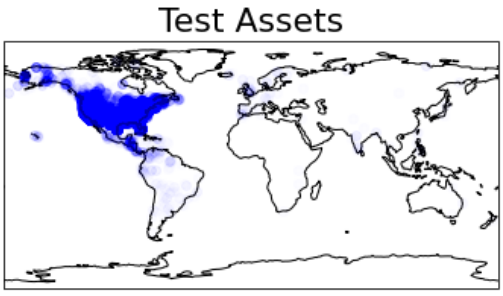


🗺️ test / train locations for current dataset

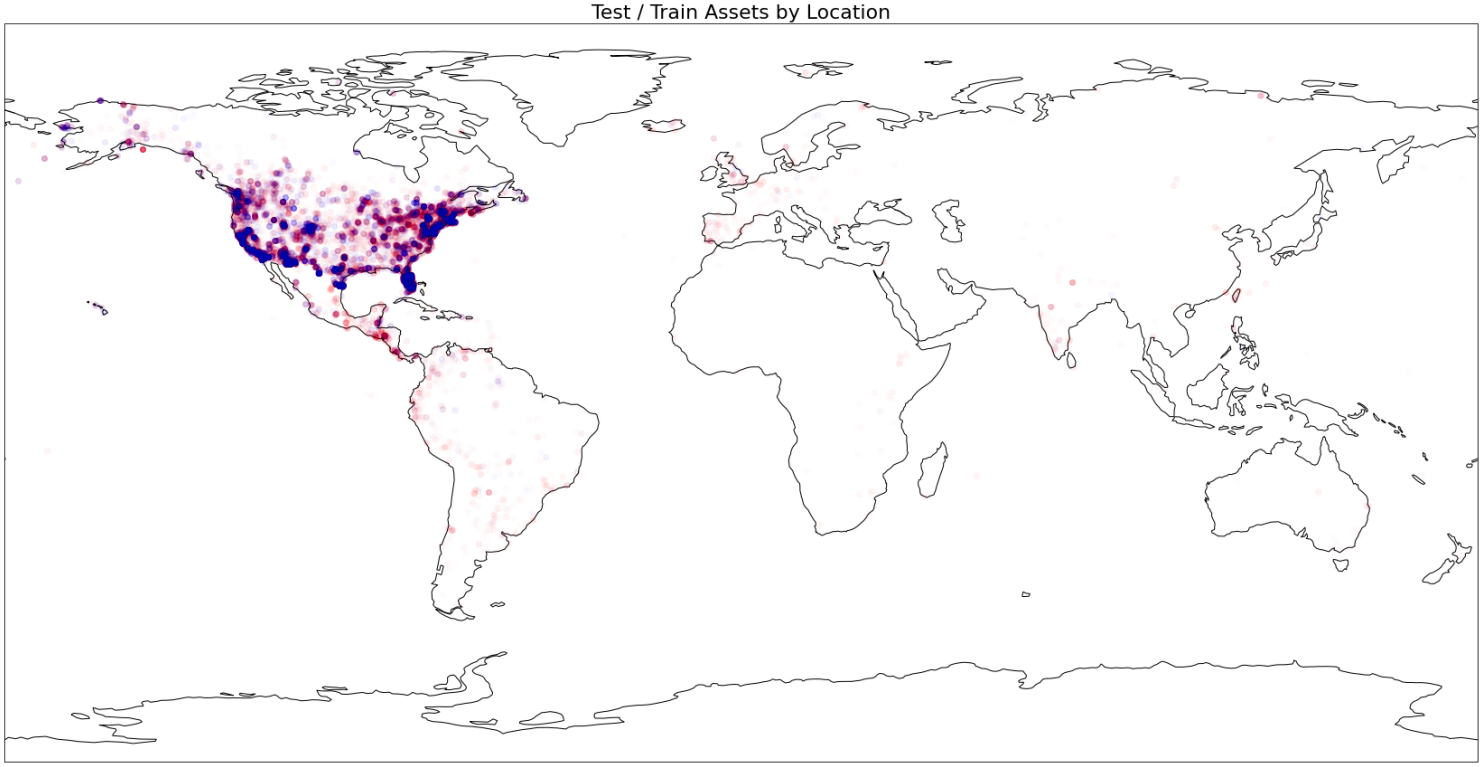
```
# split test / train:
test_df = assets_df[assets_df['id'].isin(prev_test_ids)]
train_df = assets_df[assets_df['id'].isin(prev_train_ids)]

# make seperate mini maps:
```

```
# Train:
fig = plt.figure(figsize=(6, 3))
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines()
ax.plot(train_df['longitude'], train_df['latitude'], 'ro', alpha=0.01)
ax.set_global()
_ = ax.set_title("Train Assets")
plt.savefig('./output/TrainAssetsMapSmall.png')
plt.savefig('./output/TrainAssetsMapSmall.pdf')

# Test:
fig = plt.figure(figsize=(6, 3))
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines()
ax.plot(test_df['longitude'], test_df['latitude'], 'bo', alpha=0.01)
ax.set_global()
_ = ax.set_title("Test Assets")
plt.savefig('./output/TestAssetsMapSmall.png')
plt.savefig('./output/TestAssetsMapSmall.pdf')


# make a big map of both:

fig = plt.figure(figsize=(30, 15))
ax = plt.axes(projection=ccrs.PlateCarree())
ax.coastlines()
ax.plot(train_df['longitude'], train_df['latitude'], 'ro', alpha=0.01)
ax.plot(test_df['longitude'], test_df['latitude'], 'bo', alpha=0.01)
ax.set_global()

_ = ax.set_title("Test / Train Assets by Location")

plt.savefig('./output/TestTrainAssetsMapBig.png')
plt.savefig('./output/TestTrainAssetsMapBig.pdf')
```

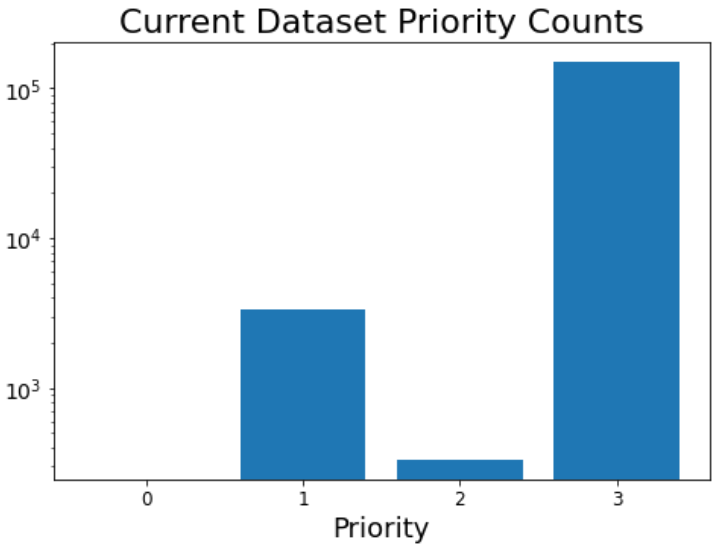




Test / Train Assets by Location



```
# get priority counts
def get_priority_counts(assets_df=assets_df):

    dist_count = assets_df['priority'].unique()
    priority_cts_df = pd.DataFrame()

    for p in dist_count:
        if not math.isnan(p):
            ct = assets_df[assets_df['priority']==p]['priority'].sum()
            priority_cts_df.loc[p, 1] = ct

    fig = plt.figure()
    ax = fig.add_axes([0,0,1,1])
    ax.ticklabel_format(useOffset=False, style='plain')
    ax.semilogy()
    priority = priority_cts_df.index.tolist()
    counts = priority_cts_df[1].tolist()
    ax.set_xticks(priority)
    ax.set_xticks(priority)

    ax.set_xlabel('Priority')
    ax.set_title('Current Dataset Priority Counts')
    ax.bar(priority, counts)
    plt.show()
    plt.savefig('./output/CurrentDatasetPriorityCounts.png')
    plt.savefig('./output/CurrentDatasetPriorityCounts.pdf')

    return priority_cts_df

get_priority_counts()
```



<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

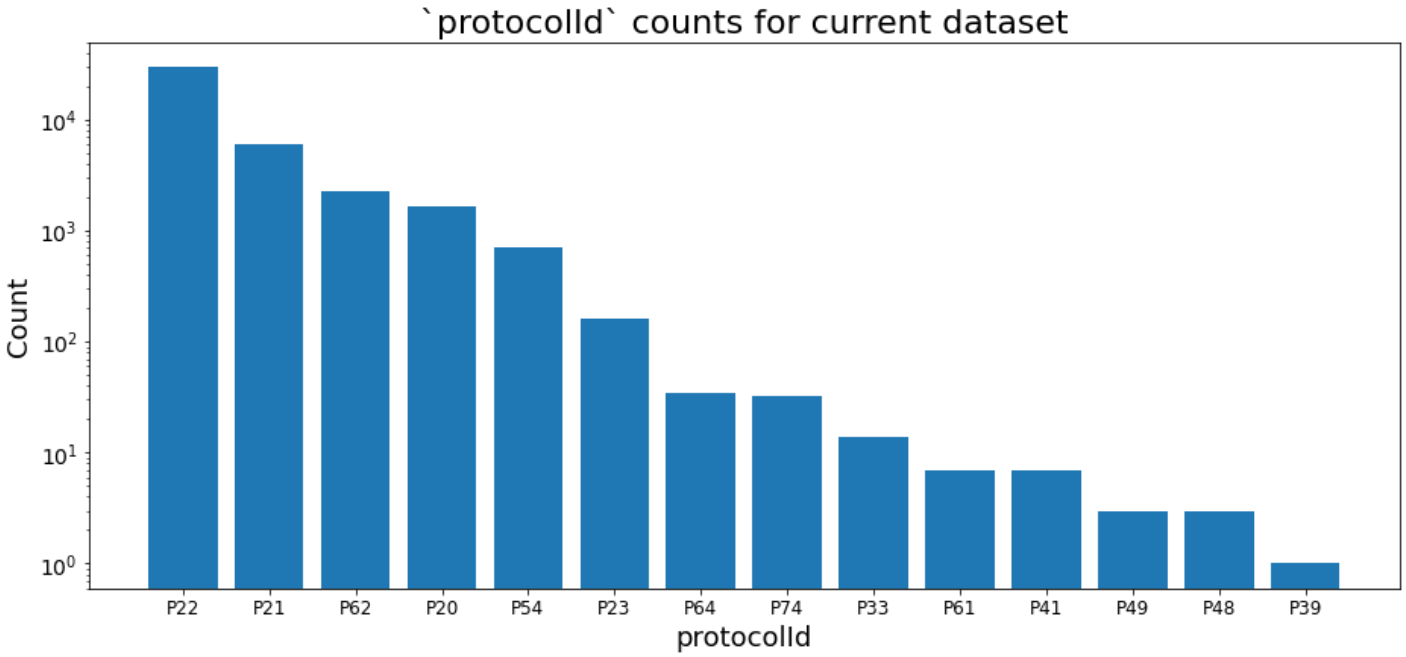
	1
3.0	150846.0
2.0	334.0
1.0	3322.0
0.0	0.0

```
<Figure size 432x288 with 0 Axes>
```

`protocolId` counts for current dataset

```
# get priority counts

protocolIds = assets_df['protocolId'].unique()
protocolIds_cts_df = pd.DataFrame()

for p in protocolIds:
    try:
        if math.isnan(p):
            continue
    except:
        pass

    ct = assets_df[assets_df['protocolId']==p]['protocolId'].shape[0]
    protocolIds_cts_df.loc[p, 1] = ct


fig = plt.figure(figsize=(12, 5))
```

```
ax = fig.add_axes([0,0,1,1])

priority = protocolIds_cts_df.index.tolist()
counts = protocolIds_cts_df[1].tolist()

counts, priority = map(np.array, zip(*sorted(zip(counts, priority))))

ax.semilogy()
ax.set_xlabel('protocolId')
ax.set_ylabel('Count')

ax.set_title('`protocolId` counts for current dataset')
ax.bar(priority[::-1], counts[::-1])
plt.show()

plt.savefig('./output/CurrentDatasetProtocolIdCounts.png')
plt.savefig('./output/CurrentDatasetProtocolIdCounts.pdf')
```




```
<Figure size 432x288 with 0 Axes>
```