# Herbarium 2022 - FGVC9

The Herbarium 2022: Flora of North America is a part of a project of the New York Botanical Garden funded by the National Science Foundation to build tools to identify novel plant species around the world. The dataset strives to represent all known vascular plant taxa in North America, using images gathered from 60 different botanical institutions around the world.

## Data Analysis:

The training and test sets contain images of herbarium specimens from 15,501 species of vascular plants. Each image contains exactly one specimen. The text labels on the specimen images have been blurred to remove category information in the image.

# Import Libraries

```
import pandas as pd
import numpy as np
from addict import Dict
import json
import matplotlib.pyplot as plt
import seaborn as sns
import os
import cv2
import random
```

```
!pip install -q addict
```

```
def load_image(path="/content/gdrive/MyDrive/Kaggle/train_images", channels=cv2.COLOR_
    if os.path.exists(path):
        image = cv2.imread(path)
        image = cv2.cvtColor(image, channels)
        image = np.asarray(image)
        return image
    else:
        raise Exception(f"Path '{path}' doesn't exist.")
```

```
def get_institutions_data_frame(institutions):
    all_ids, all_codes = [], []

    for institution in institutions:
        all_ids.append(institution["institution_id"])
```

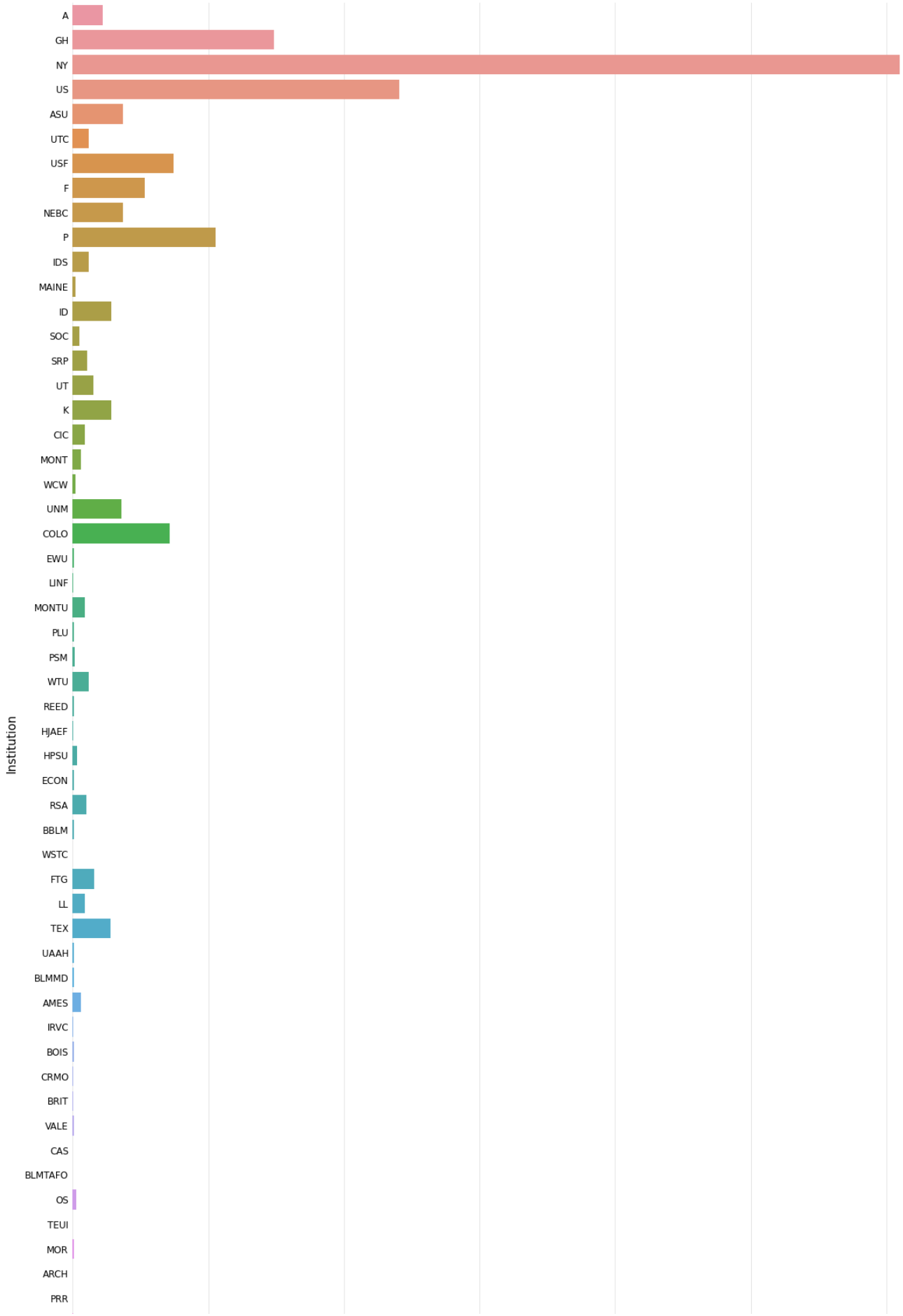
```
        all_codes.append(institution["collectionCode"])

    data_frame = pd.DataFrame({
        "institution_id": all_ids,
        "institution": all_codes,
    })

    return data_frame


def get_genera_data_frame(genera):
    all_ids, all_genuses = [], []
    for _ in genera:
        all_ids.append(_["genus_id"])
        all_genuses.append(_["genus"])

    data_frame = pd.DataFrame({
        "genus_id": all_ids,
        "genus": all_genuses
    })

    return data_frame


def get_categories_data_frame(categories):
    all_ids, all_names, all_families, all_genuses, all_species, all_authors = [], [],

    for category in categories:
        all_ids.append(category["category_id"])
        all_names.append(category["scientificName"])
        all_families.append(category["family"])
        all_genuses.append(category["genus"])
        all_species.append(category["species"])
        all_authors.append(category["authors"])

    data_frame = pd.DataFrame({
        "category_id": all_ids,
        "category": all_names,
        "family": all_families,
        "genus": all_genuses,
        "species": all_species,
        "authors": all_authors
    })

    return data_frame


def get_annotations_data_frame(annotations):
    all_genuses, all_institutions, all_categories, all_images = [], [], [], []

    for annotation in annotations:
```

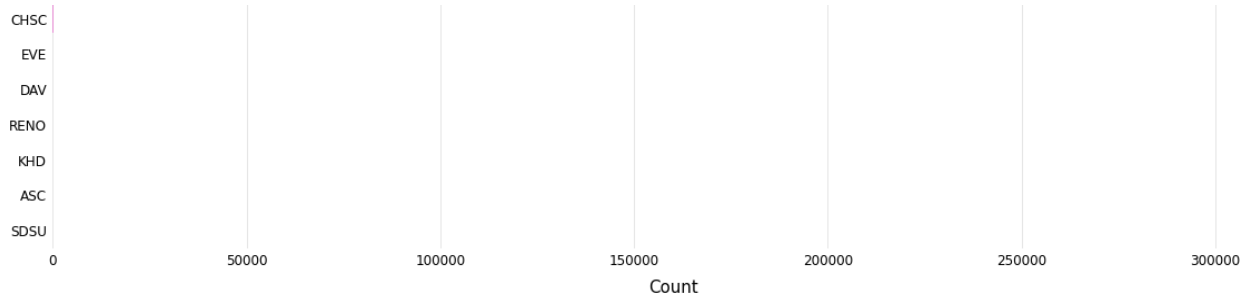
```
        all_genuses.append(annotation["genus_id"])
        all_institutions.append(annotation["institution_id"])
        all_categories.append(annotation["category_id"])
        all_images.append(annotation["image_id"])


    data_frame = pd.DataFrame({
        "genus": all_genuses,
        "institution_id": all_institutions,
        "category_id": all_categories,
        "image_id": all_images,
    })


    return data_frame


def get_images_data_frame(images):
    all_ids, all_pathes = [], []
    for image in images:
        all_ids.append(image["image_id"])
        all_pathes.append(image["file_name"])

    data_frame = pd.DataFrame({
        "image_id": all_ids,
        "image_path": all_pathes,
    })

    return data_frame


def get_meta_data_frame(meta):
    annotations = get_annotations_data_frame(meta["annotations"])
    annotations = annotations.drop("genus", axis=1)
    images = get_images_data_frame(meta["images"])
    categories = get_categories_data_frame(meta["categories"])
    genera = get_genera_data_frame(meta["genera"])
    institutions = get_institutions_data_frame(meta["institutions"])

    data_frame = annotations.merge(images, on="image_id")
    data_frame = data_frame.merge(categories, on="category_id")
    data_frame = data_frame.merge(institutions, on="institution_id")

    data_frame = data_frame.drop(["image_id", "category_id", "institution_id"], axis=1

    return data_frame


def create_submission(ids, predictions, path="submission.csv"):
    submission = pd.DataFrame({
        "Id": ids,
```

```
        "Predicted": predictions,
    })

    submission.to_csv(path, index=False)
    return submission

def read_json(path):
    with open(path, "r", encoding="utf-8") as file:
        data = json.loads(file.read())
    return data


def plot_category_images(data_frame, category, title="Title", rows=1, columns=5, backg
    fig = plt.figure(figsize=(columns*3, rows*5))
    fig.set_facecolor(background_color)
    data_frame = data_frame[data_frame["category"] == category]
    images = rows * columns
    genuses = ", ".join(data_frame["genus"].unique())
    families = ", ".join(data_frame["family"].unique())
    species = ", ".join(data_frame["species"].unique())

    for i in range(images):
        index = random.randint(0, len(data_frame)-1)
        sample = data_frame.iloc[index]
        image_path = sample["image_path"]
        image = load_image(image_path)
        filename = image_path.split("/")[-1]

        description = f"Families: {families}\nGenuses: {genuses}\nSpecies: {species}"

        ax = fig.add_subplot(rows, columns, i+1)
        ax.set_facecolor(background_color)
        ax.imshow(image)
        ax.set_xlabel(filename, color="#000", fontsize=14, y=1)
        ax.xaxis.set_tick_params(labelsize=0, size=0)
        ax.yaxis.set_visible(False)
        hide_spines(ax)

    fig.suptitle(title, fontsize=25, fontweight="bold", fontfamily="serif", horizontal
    fig.text(s=description, x=0.01, y=0.97, fontsize=17, fontfamily="serif", horizonta
    fig.tight_layout(h_pad=5, w_pad=2)
    fig.show()

def hide_spines(ax, spines=["top", "right", "left", "bottom"]):
    for spine in spines:
        ax.spines[spine].set_visible(False)


pathes = Dict({
    "train_meta": "../input/herbarium-2022-fgvc9/train_metadata.json",
    "train_images": "../input/herbarium-2022-fgvc9/train_images",
```

```
    "test_meta": "../input/herbarium-2022-fgvc9/test_metadata.json",
    "test_images": "../input/herbarium-2022-fgvc9/test_images",
    "sample_submission": "../input/herbarium-2022-fgvc9/sample_submission.csv",
})


train_meta = read_json("/content/gdrive/MyDrive/Kaggle/train_metadata.json")
train = get_meta_data_frame(train_meta)
train["image_path"] = train["image_path"].apply(lambda _: os.path.join(pathes.train_ir


test_meta = read_json("/content/gdrive/MyDrive/Kaggle/test_metadata.json")
test = get_images_data_frame(test_meta)
test["image_path"] = test["image_path"].apply(lambda _: os.path.join(pathes.test_image


sample_submission = pd.read_csv("/content/gdrive/MyDrive/Kaggle/sample_submission.csv'


fig = plt.figure(figsize=(20, 35))
fig.set_facecolor("#fff")
ax = fig.add_subplot()
ax.set_facecolor("#fff")
ax.grid(color="lightgrey", alpha=0.7, axis="both", zorder=0)
sns.countplot(y="institution", data=train, ax=ax, zorder=2)
ax.xaxis.set_tick_params(labelsize=12, size=0, pad=5)
ax.yaxis.set_tick_params(labelsize=12, size=0, pad=5)
ax.set_ylabel("Institution", fontsize=15, fontweight="normal", labelpad=10)
ax.set_xlabel("Count", fontsize=15, fontweight="normal", labelpad=10)
hide_spines(ax)

ax.set_title("Institution Distribution", fontsize=25, fontweight="bold", fontfamily="s
fig.show()
```

## Institution Distribution

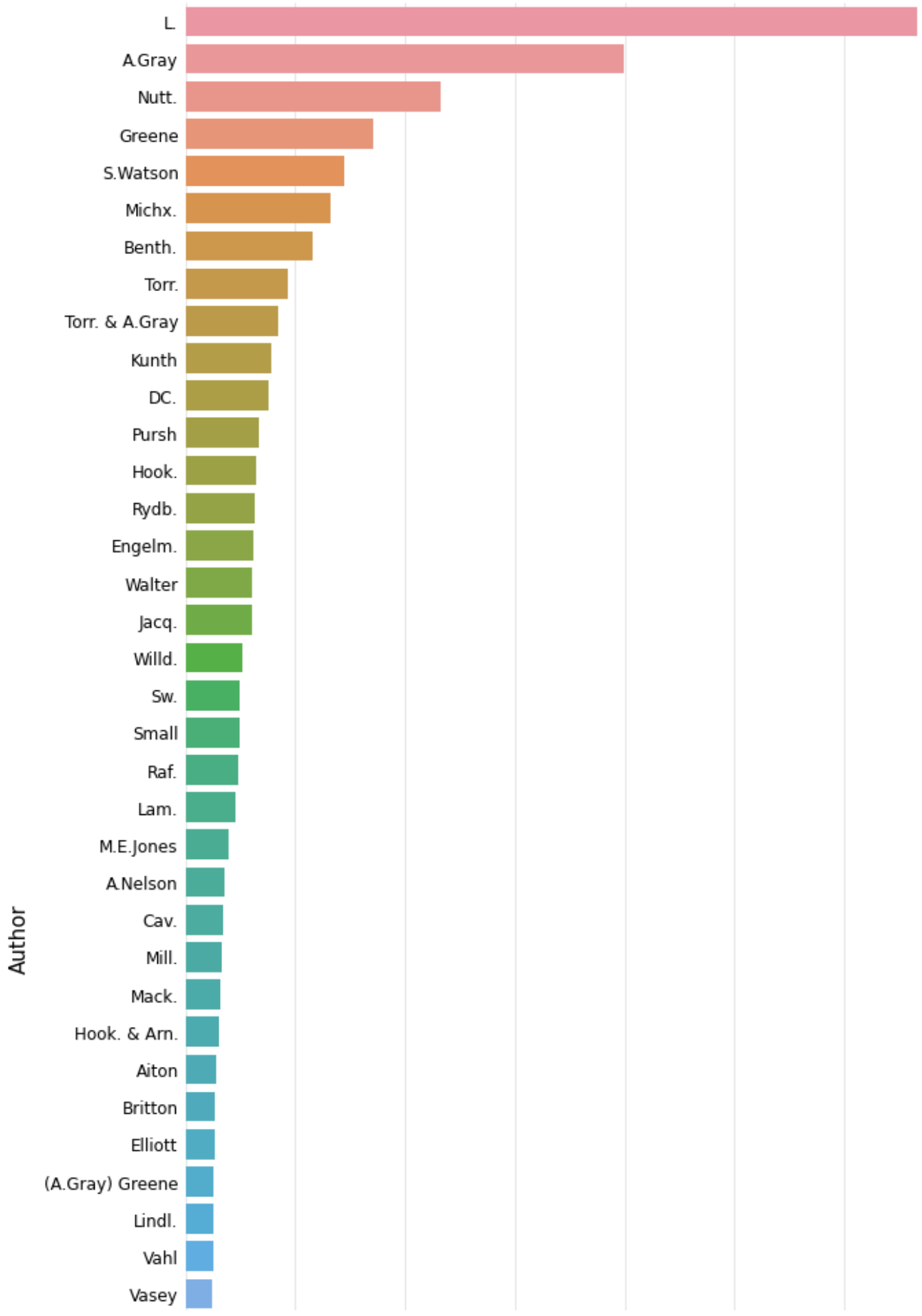


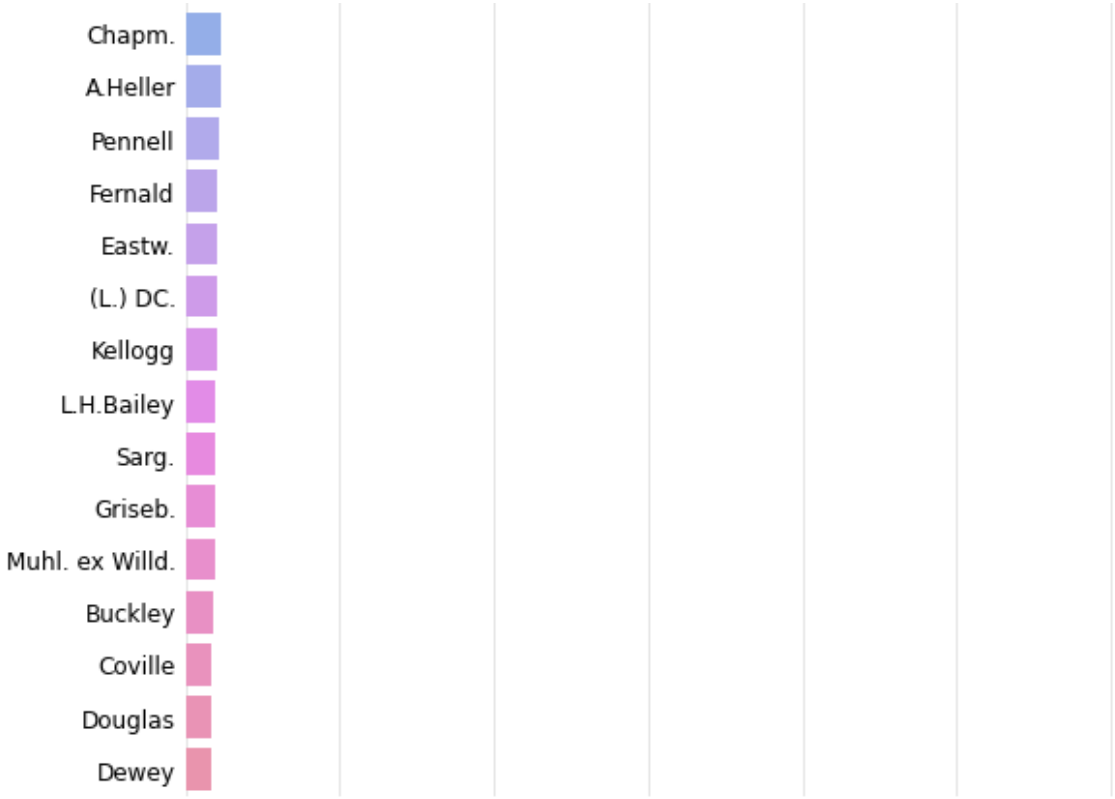



## Top 50 Authors Distribution

```
k = 50
topk_authors = train["authors"].value_counts()[:k]

fig = plt.figure(figsize=(10, 25))
fig.set_facecolor("#fff")
ax = fig.add_subplot()
ax.set_facecolor("#fff")
ax.grid(color="lightgrey", alpha=0.7, axis="both", zorder=0)
sns.barplot(x=topk_authors.values, y=topk_authors.index, ax=ax, zorder=2)
ax.xaxis.set_tick_params(labelsize=12, size=0, pad=5)
ax.yaxis.set_tick_params(labelsize=12, size=0, pad=5)
ax.set_ylabel("Author", fontsize=15, fontweight="normal", labelpad=10)
ax.set_xlabel("Author's works", fontsize=15, fontweight="normal", labelpad=10)
hide_spines(ax)

ax.set_title(f"Top {k} Authors Distribution", fontsize=25, fontweight="bold", fontfami
fig.show()
```

# Top 50 Authors Distribution

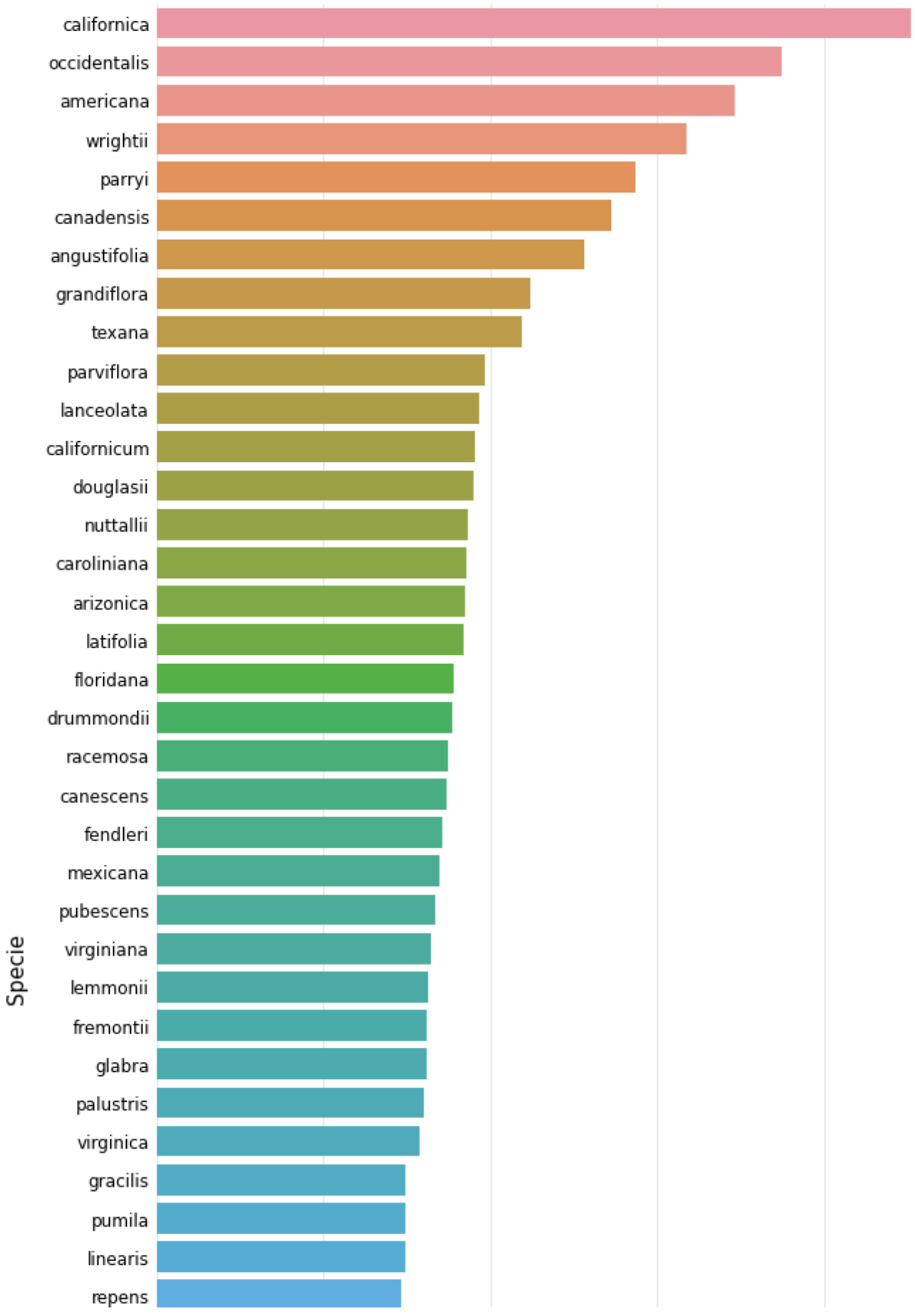


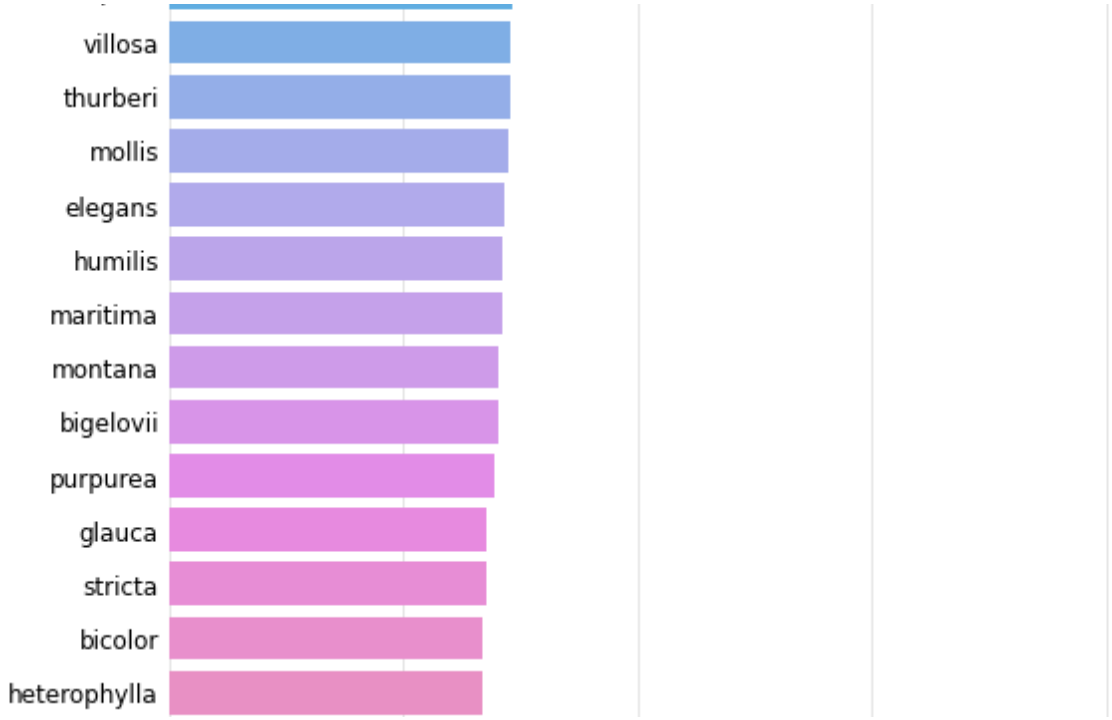



## Top 50 species Distribution

```
k = 50
topk_authors = train["species"].value_counts()[:k]

fig = plt.figure(figsize=(10, 25))
fig.set_facecolor("#fff")
ax = fig.add_subplot()
ax.set_facecolor("#fff")
ax.grid(color="lightgrey", alpha=0.7, axis="both", zorder=0)
sns.barplot(x=topk_authors.values, y=topk_authors.index, ax=ax, zorder=2)
ax.xaxis.set_tick_params(labelsize=12, size=0, pad=5)
ax.yaxis.set_tick_params(labelsize=12, size=0, pad=5)
ax.set_ylabel("Specie", fontsize=15, fontweight="normal", labelpad=10)
ax.set_xlabel("Count", fontsize=15, fontweight="normal", labelpad=10)
hide_spines(ax)

ax.set_title(f"Top {k} Species Distribution", fontsize=25, fontweight="bold", fontfami
fig.show()
```

# Top 50 Species Distribution

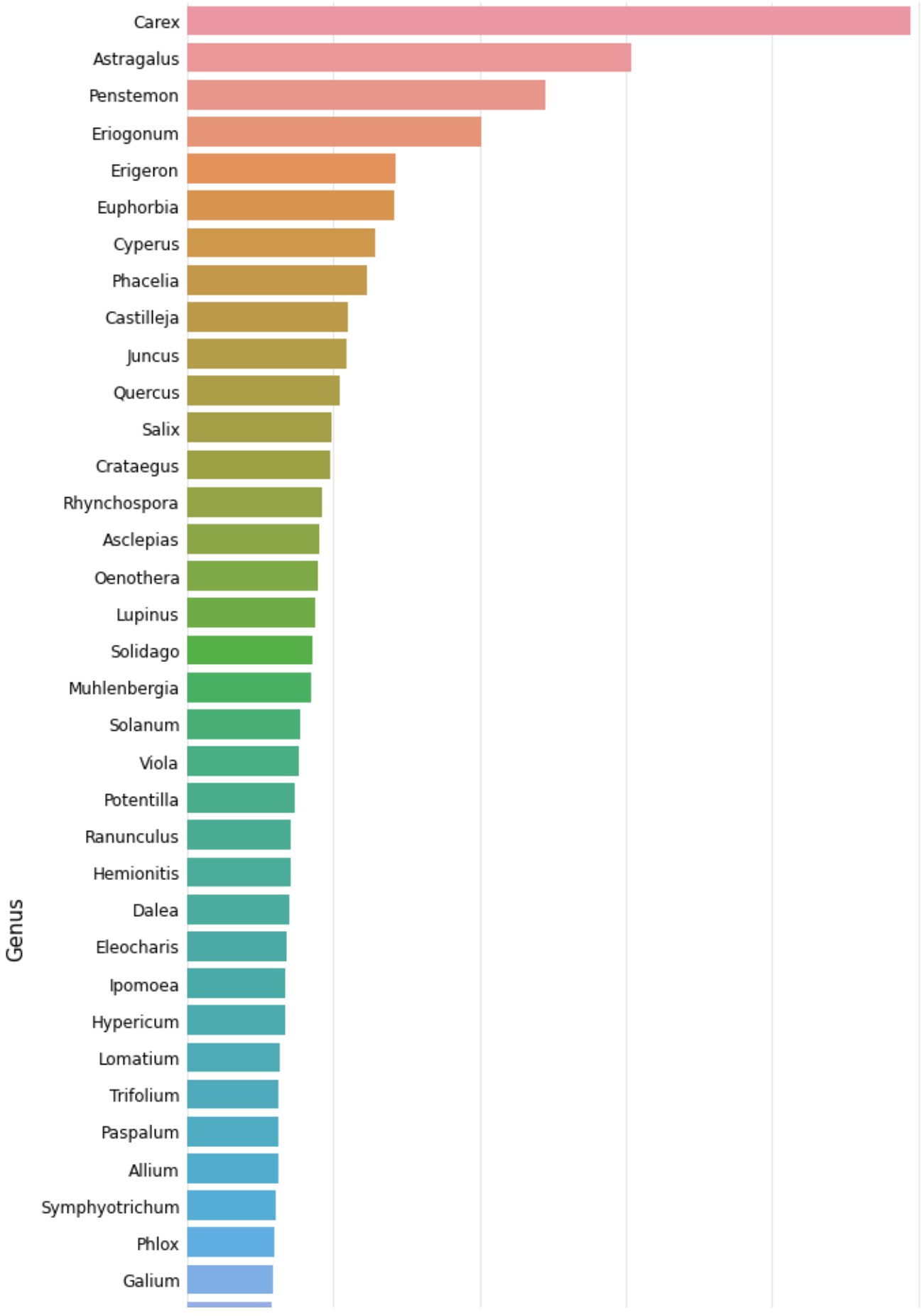


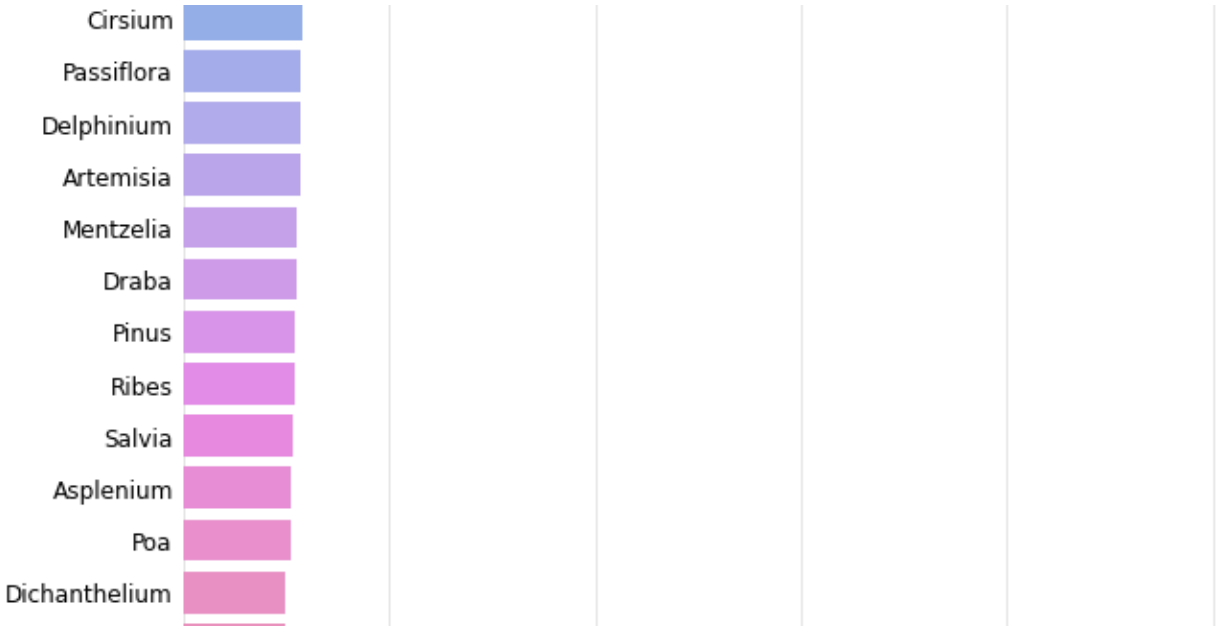



## Top 50 genuses Distribution


```
k = 50
topk_authors = train["genus"].value_counts()[:k]

fig = plt.figure(figsize=(10, 25))
fig.set_facecolor("#fff")
ax = fig.add_subplot()
ax.set_facecolor("#fff")
ax.grid(color="lightgrey", alpha=0.7, axis="both", zorder=0)
sns.barplot(x=topk_authors.values, y=topk_authors.index, ax=ax, zorder=2)
ax.xaxis.set_tick_params(labelsize=12, size=0, pad=5)
ax.yaxis.set_tick_params(labelsize=12, size=0, pad=5)
ax.set_ylabel("Genus", fontsize=15, fontweight="normal", labelpad=10)
ax.set_xlabel("Count", fontsize=15, fontweight="normal", labelpad=10)
hide_spines(ax)

ax.set_title(f"Top {k} Genuses Distribution", fontsize=25, fontweight="bold", fontfami
fig.show()
```

# Top 50 Genuses Distribution

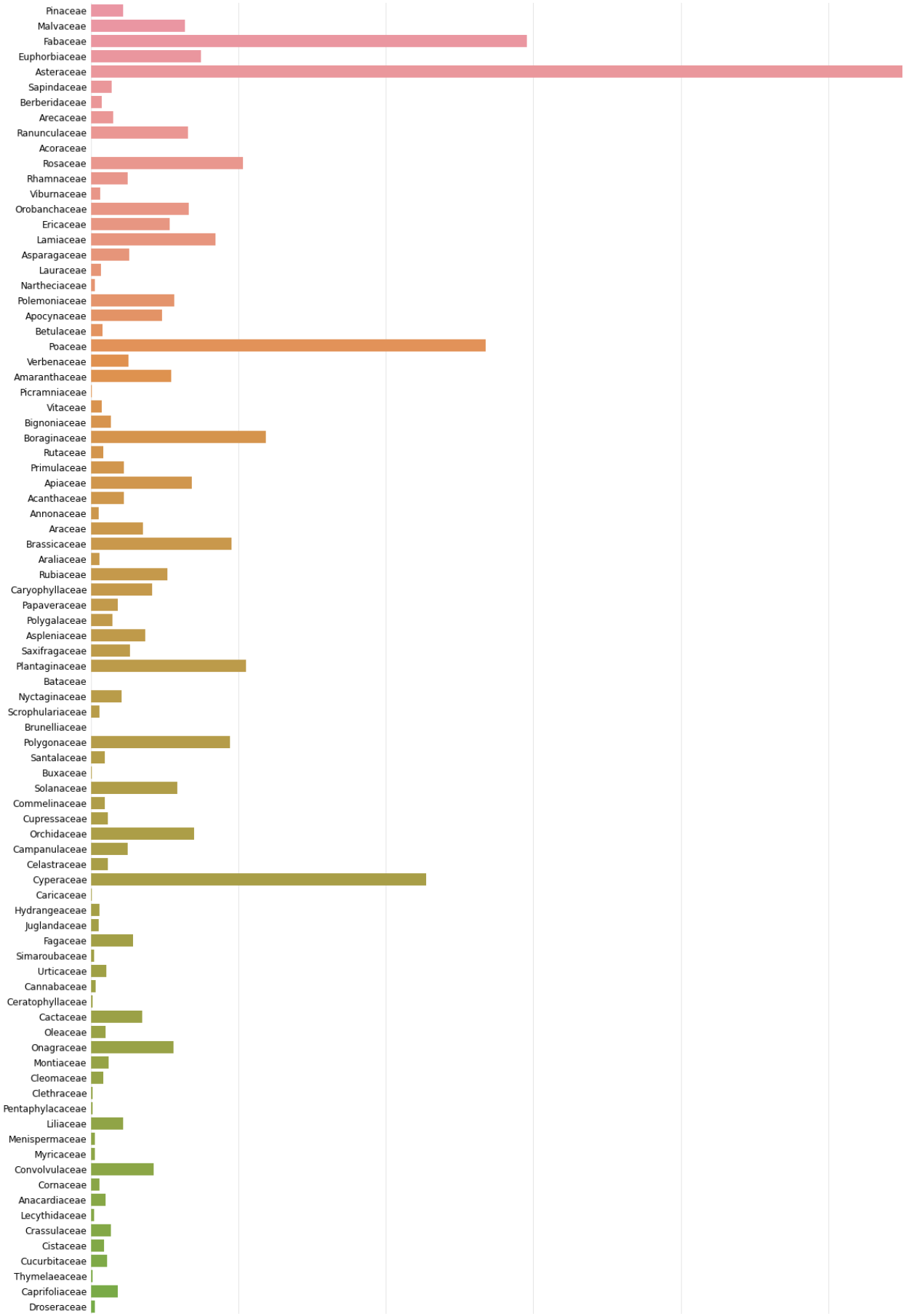


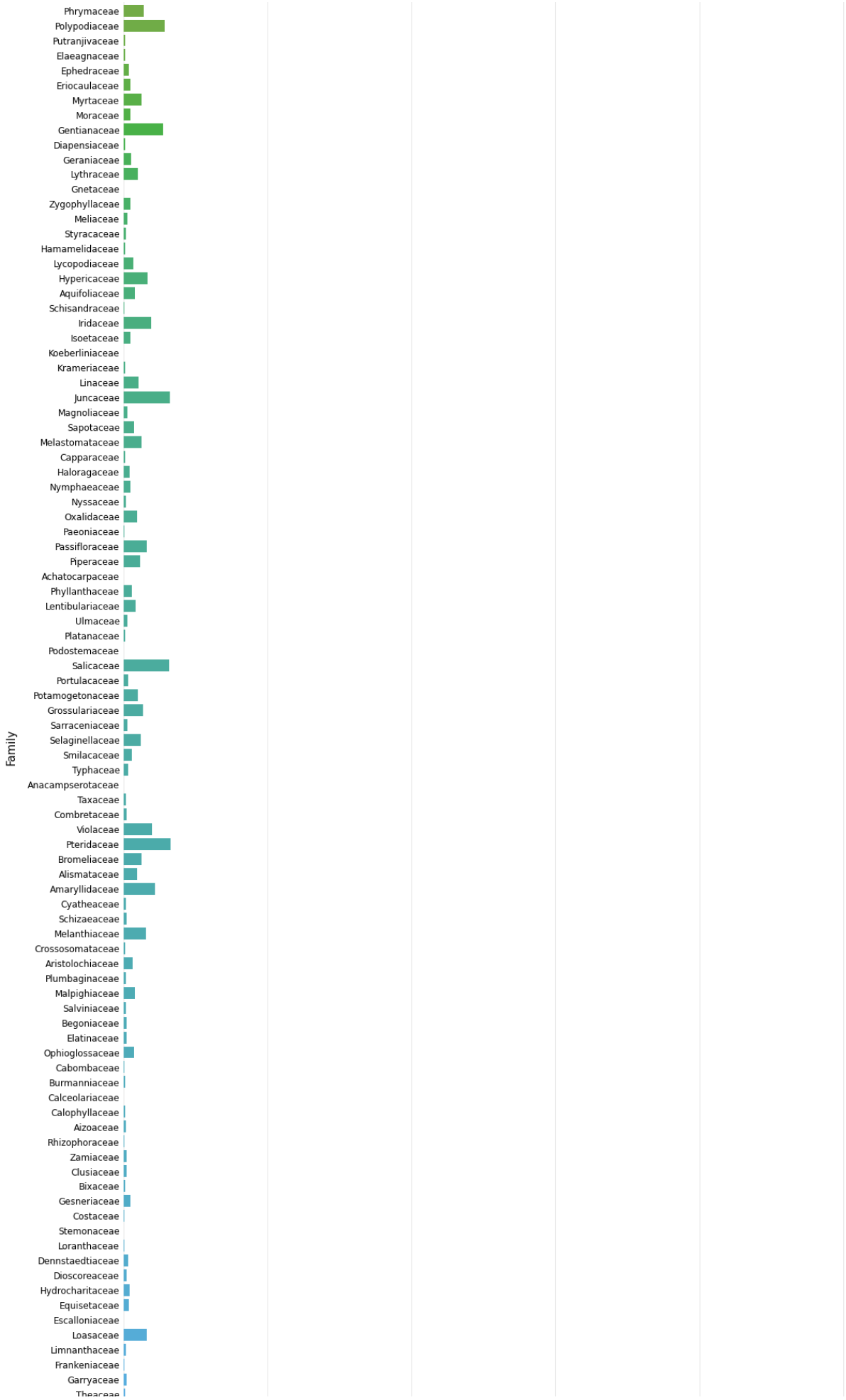


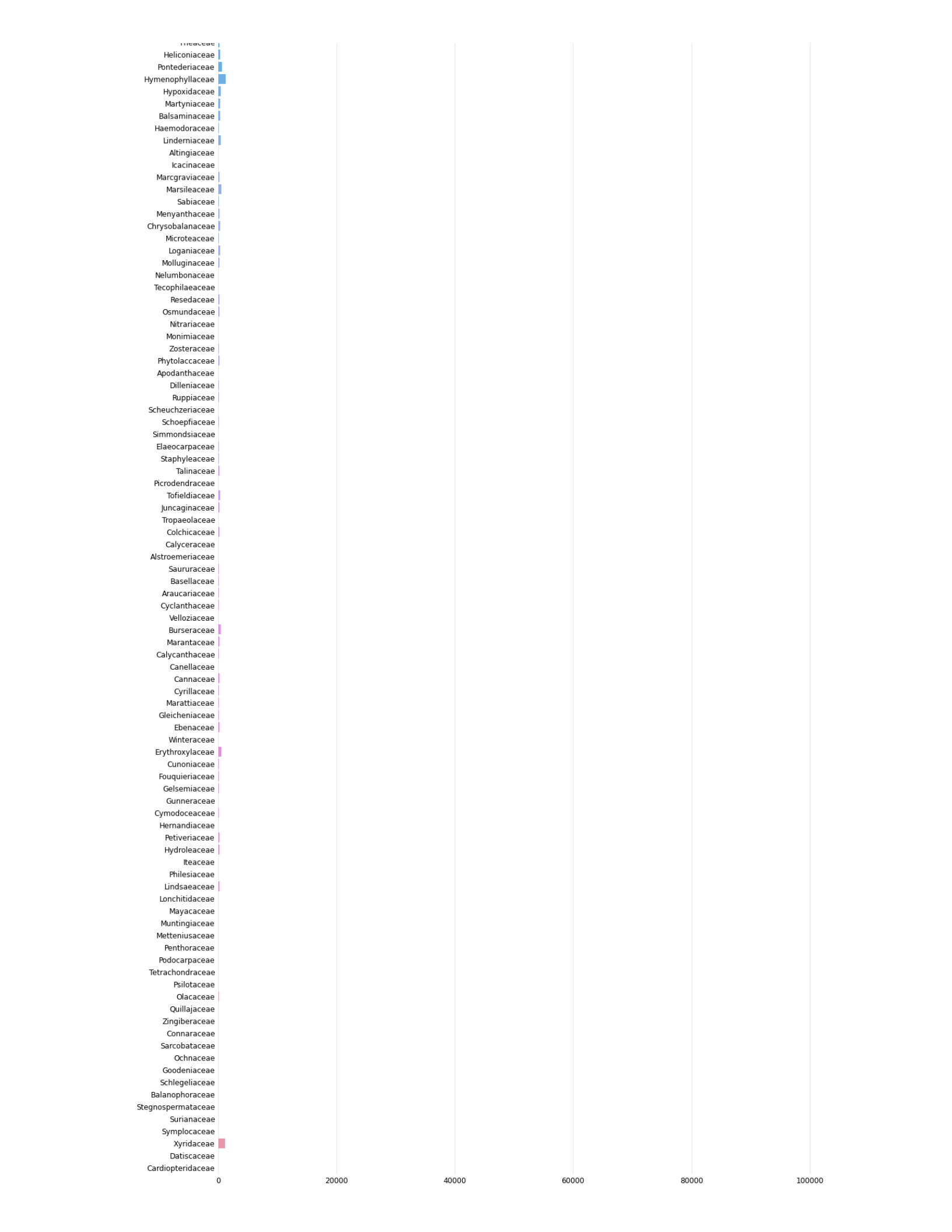

## Family Distribution

```
fig = plt.figure(figsize=(20, 100))
fig.set_facecolor("#fff")
ax = fig.add_subplot()
ax.set_facecolor("#fff")
ax.grid(color="lightgrey", alpha=0.7, axis="both", zorder=0)
sns.countplot(y="family", data=train, ax=ax, zorder=2)
ax.xaxis.set_tick_params(labelsize=12, size=0, pad=5)
ax.yaxis.set_tick_params(labelsize=12, size=0, pad=5)
ax.set_ylabel("Family", fontsize=15, fontweight="normal", labelpad=10)
ax.set_xlabel("Count", fontsize=15, fontweight="normal", labelpad=10)
hide_spines(ax)

ax.set_title("Family Distribution", fontsize=25, fontweight="bold", fontfamily="serif'
fig.show()
```

## Family Distribution




Family
Phrymaceae
Polypodiaceae
Putranjivaceae
Elaeagnaceae
Ephedraceae
Eriocaulaceae
Myrtaceae
Moraceae
Gentianaceae
Diapensiaceae
Geraniaceae
Lythraceae
Gnetaceae
Zygophyllaceae
Meliaceae
Styracaceae
Hamamelidaceae
Lycopodiaceae
Hypericaceae
Aquifoliaceae
Schisandraceae
Iridaceae
Isoetaceae
Koeberliniaceae
Krameriaceae
Linaceae
Juncaceae
Magnoliaceae
Sapotaceae
Melastomataceae
Capparaceae
Haloragaceae
Nymphaeaceae
Nyssaceae
Oxalidaceae
Paeoniaceae
Passifloraceae
Piperaceae
Achatocarpaceae
Phyllanthaceae
Lentibulariaceae
Ulmaceae
Platanaceae
Podostemaceae
Salicaceae
Portulacaceae
Potamogetonaceae
Grossulariaceae
Sarraceniaceae
Selaginellaceae
Smilacaceae
Typhaceae
Anacampserotaceae
Taxaceae
Combretaceae
Violaceae
Pteridaceae
Bromeliaceae
Alismataceae
Amaryllidaceae
Cyatheaceae
Schizaeaceae
Melanthiaceae
Crossosomataceae
Aristolochiaceae
Plumbaginaceae
Malpighiaceae
Salviniaceae
Begoniaceae
Elatinaceae
Ophioglossaceae
Cabombaceae
Burmanniaceae
Calceolariaceae
Calophyllaceae
Aizoaceae
Rhizophoraceae
Zamiaceae
Clusiaceae
Bixaceae
Gesneriaceae
Costaceae
Stemonaceae
Loranthaceae
Dennstaedtiaceae
Dioscoreaceae
Hydrocharitaceae
Equisetaceae
Escalloniaceae
Loasaceae
Limnanthaceae
Frankeniaceae
Garryaceae
Theaceae


Heliconiaceae
Pontederiaceae
Hymenophyllaceae
Hypoxidaceae
Martyniaceae
Balsaminaceae
Haemodoraceae
Linderniaceae
Altingiaceae
Icacinaceae
Marcgraviaceae
Marsileaceae
Sabiaceae
Menyanthaceae
Chrysobalanaceae
Microteaceae
Loganiaceae
Molluginaceae
Nelumbonaceae
Tecophilaeaceae
Resedaceae
Osmundaceae
Nitrariaceae
Monimiaceae
Zosteraceae
Phytolaccaceae
Apodanthaceae
Dilleniaceae
Ruppiaceae
Scheuchzeriaceae
Schoepfiaceae
Simmondsiaceae
Elaeocarpaceae
Staphyleaceae
Talinaceae
Picrodendraceae
Tofieldiaceae
Juncaginaceae
Tropaeolaceae
Colchicaceae
Calyceraceae
Alstroemeriaceae
Saururaceae
Basellaceae
Araucariaceae
Cyclanthaceae
Velloziaceae
Burseraceae
Marantaceae
Calycanthaceae
Canellaceae
Cannaceae
Cyrillaceae
Marattiaceae
Gleicheniaceae
Ebenaceae
Winteraceae
Erythroxylaceae
Cunoniaceae
Fouquieriaceae
Gelsemiaceae
Gunneraceae
Cymodoceaceae
Hernandiaceae
Petiveriaceae
Hydroleaceae
Iteaceae
Philesiaceae
Lindsaeaceae
Lonchitidaceae
Mayacaceae
Muntingiaceae
Metteniusaceae
Penthoraceae
Podocarpaceae
Tetrachondraceae
Psilotaceae
Olacaceae
Quillajaceae
Zingiberaceae
Connaraceae
Sarcobataceae
Ochnaceae
Goodeniaceae
Schlegeliaceae
Balanophoraceae
Stegnospermataceae
Surianaceae
Symplocaceae
Xyridaceae
Datiscaceae
Cardiopteridaceae
0
20000
40000
60000
80000
100000

Count

## Top 50 categories Distribution

```
k = 50
topk_authors = train["category"].value_counts()[:k]

fig = plt.figure(figsize=(10, 25))
fig.set_facecolor("#fff")
ax = fig.add_subplot()
ax.set_facecolor("#fff")
ax.grid(color="lightgrey", alpha=0.7, axis="both", zorder=0)
sns.barplot(x=topk_authors.values, y=topk_authors.index, ax=ax, zorder=2)
ax.xaxis.set_tick_params(labelsize=12, size=0, pad=5)
ax.yaxis.set_tick_params(labelsize=12, size=0, pad=5)
ax.set_ylabel("Specie", fontsize=15, fontweight="normal", labelpad=10)
ax.set_xlabel("Count", fontsize=15, fontweight="normal", labelpad=10)
hide_spines(ax)

ax.set_title(f"Top {k} Categories Distribution", fontsize=25, fontweight="bold", font
fig.show()
```