Python

Folgert Karsdorp and Mike Kestemont

Jan 11, 2024

CONTENTS

1 Data preprocessing **3**
1.1 Abundance Data 3
1.2 Incidence Data 4

2 Estimating Loss with Copia **7**
2.1 Loading the Data 7
2.2 Species Richness 8
2.3 Bootstrapped Estimation 8

3 Rarefaction - Extrapolation **11**

Copia is a statistical software package for estimating diversity and richness on the basis of abundance data. The package contains several bias-correcting richness estimators, such as the Chao1 and the Jacknife estimator.

Because this package targets so-called "abundance" data from ecology, it has been named using the corresponding Latin term copia. The artwork for the logo has been kindly contributed by Lauren Fonteyn. The depicted goat is a mythological reference to the cornucopia or "horn of plenty", the legendary horn of the goat Amaltheia, who fed the infant Zeus with her milk.

Copia is licensed under BSD3. The source is on GitHub.

Currently implemented diversity estimation methods

- Empirical Richness: empirical species richness of an assemblage;
- Chao1: Chao1 estimate of bias-corrected species richness;
- "Improved" iChao1 estimate of bias-corrected species richness;
- Egghe & Proot: Egghe & Proot estimate of bias-corrected species richness;
- ACE: ACE estimate of bias-corrected species richness (Chao & Lee 1992);
- Jacknife: Jackknife estimate of bias-corrected species richness;
- Minsample: Observed population size added to the minimum additional sampling estimate;

Installation

Copia requires Python 3.11 or greater. To install Copia, execute the following command in your terminal:

```
python3 -m pip install copia
```

Development

If you wish to help developing Copia, download the source files from GitHub and install the package using the following command:

```
pip install -e .
```

Citation

Copia is developed by Folgert Karsdorp and Mike Kestemont. If you wish to cite Copia, please cite one of the following papers:

```
@article{doi:10.1126/science.abl7655,
  author = {Mike Kestemont  and Folgert Karsdorp
            and Elisabeth de Bruijn and Matthew Driscoll
            and Katarzyna A. Kapitan and Pádraig Ó Macháin
            and Daniel Sawyer  and Remco Sleiderink and Anne Chao},
```

(continues on next page)

(continued from previous page)

```bibtex
  title = {Forgotten books: The application of unseen species models to
           the survival of culture},
  journal = {Science},
  volume = {375},
  number = {6582},
  pages = {765-769},
  year = {2022},
  doi = {10.1126/science.abl7655},
  URL = {https://www.science.org/doi/abs/10.1126/science.abl7655}}

@inproceedings{karsdorpDarkNumbersModeling2023,
  title = {Dark Numbers: Modeling the Historical Vulnerability to Arrest in Brussels␣
↪(1879-1880) Using Demographic Predictors},
  booktitle = {Proceedings of the DHBenelux Conference 2023},
  author = {Karsdorp, Folgert and Kestemont, Mike and De Koster, Margo},
  year = {2023},
  address = {Brussels}
}

@inproceedings{karsdorpWhatShallWe2022,
  title = {What Shall We Do With the Unseen Sailor? Estimating the Size of the Dutch␣
↪East India Company Using an Unseen Species Model},
  booktitle = {Proceedings of the Computational Humanities Research Conference, 2022},
  author = {Karsdorp, Folgert and Wevers, Melvin and {van Lottum}, Jelle},
  year = {2022},
  pages = {189--197},
  address = {Antwerp},
}

@inproceedings{karsdorpIntroducingFunctionalDiversity2022,
  title = {Introducing Functional Diversity: A Novel Approach to Lexical Diversity in␣
↪(Historical) Corpora},
  booktitle = {Proceedings of the Computational Humanities Research Conference, 2022},
  author = {Karsdorp, Folgert and Manjavacas, Enrique and Fonteyn, Lauren},
  year = {2022},
  pages = {114--126},
  publisher = {CEUR-WS},
  address = {Antwerp},
  langid = {english},
  file = {/Users/folgert/Zotero/storage/9H2Z4PFR/Karsdorp et al. - Introducing␣
↪Functional Diversity A Novel Approach.pdf}
}
```

CHAPTER

ONE

DATA PREPROCESSING

Two primary data types are gathered to evaluate species richness diversity. The first is individual-based abundance data. This data type focuses on individuals as the sampling units, noting the count of each species observed. The second type is sample-based incidence data, where the sampling units are specified areas or intervals, such as quadrats, plots, transects, or time intervals. In this case, the data records simply the presence or absence of each species within these units, as outlined by Colwell et al. (2012).

In the 'copia' software package, these data types are represented through the `AbundanceData` and `IncidenceData` objects, respectively.

For data analysis, all functions within 'copia' require input data to be formatted as instances of either `AbundanceData` or `IncidenceData`. To facilitate this, the `copia.data.to_copia_dataset()` function is designed to convert a collection of observations into a structured 'copia' dataset. This process is compatible with both abundance and incidence data types. Below, we provide various examples to illustrate how datasets can be created and initialized using this function.

1.1 Abundance Data

Now, let's focus on handling abundance data. Suppose you have gathered a list of species observations. To effectively utilize these observations in the 'copia' software, you can organize them into a copia dataset in the following manner:

```
from copia.data import to_copia_dataset

observations = 'duck', 'duck', 'eagle', 'dove', 'dove', 'dove', 'hawk'
ds = to_copia_dataset(observations, data_type="abundance", input_type="observations")
ds
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 1
----> 1 from copia.data import to_copia_dataset
      3 observations = 'duck', 'duck', 'eagle', 'dove', 'dove', 'dove', 'hawk'
      4 ds = to_copia_dataset(observations, data_type="abundance", input_type=
 ↪"observations")

ModuleNotFoundError: No module named 'copia.data'
```

The `AbundanceData` object is designed to transform your observations into a structured array. This array enumerates the counts for each unique species identified in your dataset. Additionally, it computes basic statistical measures, including the count of singletons (f1) and doubletons (f2). These metrics are crucial for numerous estimation functions within the Copia framework.

In scenarios where you're working with count data directly, as opposed to raw observations, constructing a similar data object is straightforward:

```
import numpy as np

counts = np.array([3, 2, 1, 1])
ds = to_copia_dataset(counts, data_type="abundance", input_type="counts")
ds
```

```
AbundanceData(S_obs=4, f1=2, f2=1, n=7, counts=array([3, 2, 1, 1]))
```

Similarly, we can use a Pandas `Series` or `DataFrame` object to construct the dataset:

```
import pandas as pd

counts = pd.Series([3, 2, 1, 1], index=["dove", "duck", "eagle", "hawk"])
counts
```

```
dove     3
duck     2
eagle    1
hawk     1
dtype: int64
```

```
ds = to_copia_dataset(counts, data_type="abundance", input_type="counts")
ds
```

```
AbundanceData(S_obs=4, f1=2, f2=1, n=7, counts=array([3, 2, 1, 1]))
```

1.2 Incidence Data

Moving on, let's explore the handling of incidence data in Copia. Similar to abundance data, you can input either raw observation data or count data. There are two ways to input raw incidence observations:

1.2.1 1. Observation Matrix

This method is suitable for data presented in a matrix format, which can be either a NumPy array or a pandas DataFrame. In this matrix, rows (or columns) correspond to unique items, while columns (or rows) signify unique locations. A non-zero entry in the matrix indicates the occurrence of an item at a specific location.

```
observation_matrix = np.array([[1, 0], [0, 1], [1, 1]])
ds = to_copia_dataset(observation_matrix, data_type="incidence",
                      input_type='observation_matrix', n_sampling_units=3)
ds
```

```
IncidenceData(S_obs=3, f1=2, f2=1, n=4, counts=array([1, 1, 2]), T=3)
```

1.2.2 2. Observation List

This approach is ideal for data structured as a list, tuple, dict, or NumPy array consisting of (item, location) pairs. It can also be a pandas DataFrame with designated columns for items and locations.

```
observation_list = [
    ('item1', 'loc1'), ('item2', 'loc2'),
    ('item3', 'loc1'), ('item3', 'loc2')
]

ds = to_copia_dataset(observation_list, data_type="incidence",
                      input_type='observation_list', n_sampling_units=3)
ds
```

```
IncidenceData(S_obs=3, f1=2, f2=1, n=4, counts=array([1, 1, 2]), T=3)
```

One could also provide a Pandas DataFrame object and specify the columns holding the items and locations respectively:

```
observation_df = pd.DataFrame(observation_list, columns=['item', 'location'])
observation_df
```

```
    item location
0  item1     loc1
1  item2     loc2
2  item3     loc1
3  item3     loc2
```

```
ds = to_copia_dataset(
    observation_list, data_type="incidence",
    input_type='observation_list',
    location_column='location',
    index_column='item',
    n_sampling_units=3)
ds
```

```
IncidenceData(S_obs=3, f1=2, f2=1, n=4, counts=array([1, 1, 2]), T=3)
```

In addition to raw observations, Copia also supports the use of count data for analyzing incidence. You have the flexibility to provide this data in various formats:

1. List or Array: Simple count data can be supplied as a list or an array, where the counts are directly enumerated.
2. Pandas `Series` or `DataFrame`: For a more comprehensive analysis, you can opt for a detailed format by using a pandas `Series` or `DataFrame`. This allows for a richer representation of the data, including additional attributes and more complex structures.

The following examples illustrate how count data for incidence can be incorporated into Copia:

```
counts = [2, 1, 1]

ds = to_copia_dataset(
    counts, data_type='incidence', input_type='counts', n_sampling_units=3)
ds
```

```
IncidenceData(S_obs=3, f1=2, f2=1, n=4, counts=array([2, 1, 1]), T=3)
```

```
df = pd.DataFrame([
    {'item': 'item1', 'count': 1},
    {'item': 'item2', 'count': 1},
    {'item': 'item3', 'count': 2}])

ds = to_copia_dataset(
    counts, data_type='incidence',
    input_type='counts',
    index_column='item',
    count_column='count',
    n_sampling_units=3)
ds
```

```
IncidenceData(S_obs=3, f1=2, f2=1, n=4, counts=array([2, 1, 1]), T=3)
```

For more comprehensive details and specific guidance on utilizing the functions within Copia, we strongly encourage you to refer to our detailed documentation. This resource will provide you with in-depth explanations and additional examples to enhance your understanding and usage of the software.

Copia offers a versatile and robust framework that efficiently accommodates both abundance and incidence data types. It's important to reiterate that for successful analysis and estimation within Copia, your data must be formatted as an instance of either `AbundanceData` or `IncidenceData`. Adhering to this requirement ensures accurate processing and effective utilization of Copia's capabilities.

CHAPTER

TWO

ESTIMATING LOSS WITH COPIA

The data for this first part of the tutorial is sourced from the study by Kestemont & Karsdorp (2019), which marked our initial exploration into the realm of unseen medieval literature using statistical methods. The study addresses a key challenge in historical literature research: the incomplete transmission of sources over centuries, leading to a potential underestimation of the literary diversity of the past. By employing eco-diversity statistical methods on Middle Dutch chivalric epics, the study suggests that previous estimates of text loss might have been too conservative. These methods unveil that up to half, or possibly more, of the original texts may have been lost over time, highlighting the need for quantitative tools to adjust our understanding of historical literary diversity.

2.1 Loading the Data

We will utilize the Pandas library to load our dataset for analysis. The following section provides the necessary code and instructions to successfully load and prepare the data for our exploration using Copia.

```
import pandas as pd

df = pd.read_csv("../datasets/dum.csv", sep=';')
df.columns = 'work', 'signature'
df.head()
```

```
                 work                                          signature
0            Aiol (2)                       Breda, Arch. Begijnhof, z.s.
1  Alexanders geesten  München, Bayerische Staatsbibliotheek, Cod. ge...
2  Alexanders geesten                                   Gent, UB, 2749,6
3  Alexanders geesten                              Leiden, UB, Ltk. 1196
4  Alexanders geesten    Donaueschingen, Fürstl. Fürstenb. Hofbibl., 173
```

The Chao1 estimator formula necessitates the provision of two key parameters from our collection: the number of items occurring once (f_1) and the number of items occurring twice (f_2). These parameters are pivotal for estimating the unseen diversity in our dataset. Copia comes with a utility function designed to streamline the conversion of our list of works into abundance or incidence data, aligning with the requirements of the different estimators (see the tutorial on loading the data for more details).

```
from copia.data import to_copia_dataset

ds = to_copia_dataset(
    df, data_type="abundance", input_type="observations", index_column="work")
ds
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[2], line 1
----> 1 from copia.data import to_copia_dataset
      3 ds = to_copia_dataset(
      4     df, data_type="abundance", input_type="observations", index_column=
 ↪"work")
      5 ds

ModuleNotFoundError: No module named 'copia.data'
```

2.2 Species Richness

We utilize Copia's `copia.estimators` module to estimate unbiased diversity using one of the available estimators. All estimators can be accessed through a single entry point, the `diversity()` function:

```
from copia.estimators import diversity

round(diversity(ds, method='chao1'))
```

```
148
```

Copia includes a variety of estimators for your convenience:

```
for method in ('chao1', 'ichao1', 'ace', 'jackknife', 'egghe_proot'):
    print(method, '->', round(diversity(ds, method=method)))
```

```
chao1 -> 148
ichao1 -> 166
ace -> 148
jackknife -> 173
egghe_proot -> 165
```

2.3 Bootstrapped Estimation

The Chao1 estimator provides a lower bound estimate of the unseen species count. However, this estimate itself has an associated range, defined by a lower and upper bound, which represents the confidence interval around the estimate. Two common approaches to obtaining this confidence interval include analytical solutions or bootstrap procedures. In Copia, the bootstrap procedure is implemented to derive these bounds. To enable this feature, set the CI parameter to True when calling the function.

```
D = diversity(ds, method='chao1', CI=True, n_iter=100, n_jobs=1)
D
```

```
100
 ↪%|
 ↪100/100 [00:01<00:00, 78.38it/s]
```

```
est    148.012975
lci    102.645897
uci    225.651810
Name: 0, dtype: float64
```

CHAPTER

THREE

RAREFACTION - EXTRAPOLATION

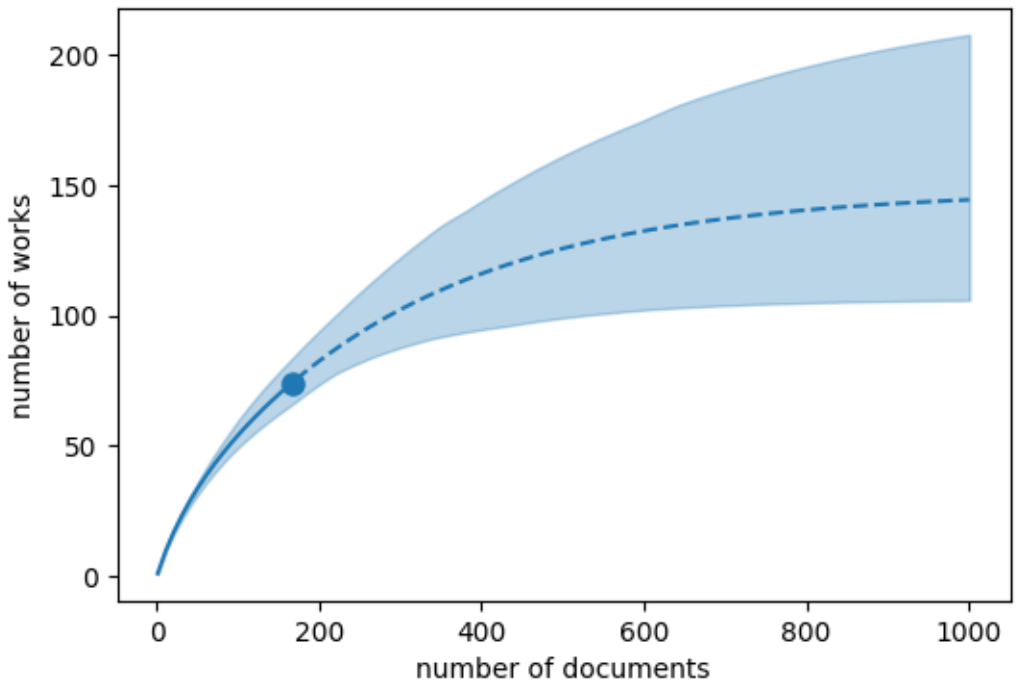
Rarefaction - Extrapolation, or Species Accumulation curves elucidate the relationship between the number of unique species and the increment in sample size (i.e., the number of observations). These curves offer a sort of peak into the future: they show how many unique species we are expected to find if we were to continue sampling more items – into infinity. With Copia, you can compute this relationship effortlessly using the `species_accumulation()` function. Examine the following code snippet for a practical demonstration:

```
import pandas as pd

from copia.data import to_copia_dataset
from copia.rarefaction_extrapolation import species_accumulation

df = pd.read_csv("../datasets/dum.csv", sep=';')
df.columns = 'work', 'signature'

ds = to_copia_dataset(
    df, data_type="abundance", input_type="observations", index_column="work")

accumulation = species_accumulation(ds, max_steps=1000, n_iter=100, step_size=10)
accumulation
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 3
      1 import pandas as pd
----> 3 from copia.data import to_copia_dataset
      4 from copia.rarefaction_extrapolation import species_accumulation
      6 df = pd.read_csv("../datasets/dum.csv", sep=';')

ModuleNotFoundError: No module named 'copia.data'
```

We can visualize the species accumulation using Copia's `accumulation_curve()` function. The following code snippet demonstrates how to plot the accumulation curve:

```
import matplotlib.pyplot as plt
from copia.plot import accumulation_curve

fig, ax = plt.subplots(figsize=(6, 4))

accumulation_curve(
    ds, accumulation, ax=ax, xlabel="number of documents", ylabel="number of works");
```



In the plotted curve, the blue dot represents the empirical species richness. To its left, the interpolated accumulation of species with increased sampling is depicted (the rarefaction part). On the right, a projection into the future is illustrated, showing the anticipated discovery of new species or works with further document sampling (the extrapolation part). Notably, the curve begins to flatten around 1000 documents, suggesting that beyond this point, no new unseen works are expected to be found according to the estimator.