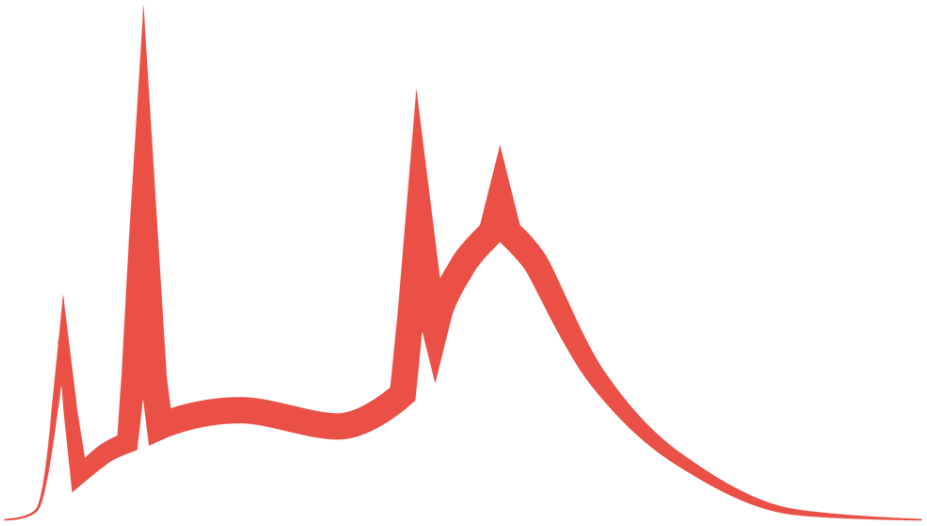

COLOUR - DATASETS

Colour - Datasets Documentation

Release 0.2.4

Colour Developers

Nov 13, 2023

CONTENTS

Colour science datasets for use with Colour or any Python package manipulating colours. The datasets are hosted in Zenodo under the Colour Science - Datasets community.

It is open source and freely available under the BSD-3-Clause terms.

CHAPTER

ONE

1.1 FEATURES

Colour - Datasets was created to overcome issues encountered frequently when trying to access or use colour science datasets:

- No straightforward ingestion path for dataset content.
- No simple loading mechanism for dataset content.
- Unavailability of the dataset, e.g. download url is down, dataset content is passed directly from hand to hand.
- No information regarding the definitive origination of the dataset.

Colour - Datasets offers all the above: it allows users to ingest and load colour science datasets with a single function call. The datasets information is hosted on Zenodo where the record for a dataset typically contain:

- An *urls.txt* file describing the urls to source the dataset files from.
- A copy of those files in the eventuality where the source files are not available or the content has changed without notice.
- Information about the authors, content and licensing.

When no explicit licensing information is available, the dataset adopts the **Other (Not Open)** licensing scheme, implying that assessing usage conditions is at the sole discretion of the users.

1.1 1.1.1 Examples

Colour - Datasets can also be used online with Google Colab.

Most of the objects are available from the `colour_datasets` namespace:

```
>>> import colour_datasets
```

The available datasets are listed with the `colour_datasets.datasets()` definition:

```
>>> print(colour_datasets.datasets())
```

```
colour-science-datasets
=======================

Datasets : 22
Synced   : 1
URL      : https://zenodo.org/communities/colour-science-datasets/

Datasets
--------
```

(continues on next page)

(continued from previous page)

```
[ ] 3269926 : Agfa IT8.7/2 Set - Marszalec (n.d.)
[ ] 3245883 : Camera Spectral Sensitivity Database - Jiang et al. (2013)
[ ] 3367463 : Constant Hue Loci Data - Hung and Berns (1995)
[ ] 3362536 : Constant Perceived-Hue Data - Ebner and Fairchild (1998)
[ ] 3270903 : Corresponding-Colour Datasets - Luo and Rhodes (1999)
[ ] 3269920 : Forest Colors - Jaaskelainen et al. (1994)
[ ] 4394536 : LUTCHI Colour Appearance Data - Luo and Rhodes (1997)
[x] 3245875 : Labsphere SRS-99-020 - Labsphere (2019)
[ ] 3269924 : Lumber Spectra - Hiltunen (n.d.)
[ ] 4051012 : Measured Commercial LED Spectra - Brendel (2020)
[ ] 3269918 : Munsell Colors Glossy (All) (Spectrofotometer Measured) - Orava (n.d.)
[ ] 3269916 : Munsell Colors Glossy (Spectrofotometer Measured) - Haanpalo (n.d.)
[ ] 3269914 : Munsell Colors Matt (AOTF Measured) - Hauta-Kasari (n.d.)
[ ] 3269912 : Munsell Colors Matt (Spectrofotometer Measured) - Hauta-Kasari (n.d.)
[ ] 3245895 : New Color Specifications for ColorChecker SG and Classic Charts - X-Rite␣
↪(2016)
[ ] 3252742 : Observer Function Database - Asano (2015)
[ ] 3269922 : Paper Spectra - Haanpalo (n.d.)
[ ] 6590768 : Physlight - Camera Spectral Sensitivity Curves - Winquist et al. (2022)
[ ] 3372171 : RAW to ACES Utility Data - Dyer et al. (2017)
[ ] 4642271 : Spectral Database of Commonly Used Cine Lighting - Karge et al. (2015)
[ ] 4297288 : Spectral Sensitivity Database - Zhao et al. (2009)
[ ] 4050598 : Spectral Upsampling Coefficient Tables - Jakob and Hanika. (2019)
```

A ticked checkbox means that the particular dataset has been synced locally. A dataset is loaded by using its unique number: *3245895*:

```
>>> print(colour_datasets.load("3245895").keys())
```

```
Pulling "New Color Specifications for ColorChecker SG and Classic Charts" record content..
↪.
Downloading "urls.txt" file: 8.19kB [00:01, 5.05kB/s]
Downloading "ColorChecker24_After_Nov2014.zip" file: 8.19kB [00:01, 6.52kB/s]
Downloading "ColorChecker24_Before_Nov2014.zip" file: 8.19kB [00:01, 7.66kB/s]
Downloading "ColorCheckerSG_After_Nov2014.zip" file: 8.19kB [00:01, 7.62kB/s]
Downloading "ColorCheckerSG_Before_Nov2014.zip" file: 8.19kB [00:00, 9.39kB/s]
Unpacking "/Users/kelsolaar/.colour-science/colour-datasets/3245895/dataset/
↪ColorCheckerSG_Before_Nov2014.zip" archive...
Unpacking "/Users/kelsolaar/.colour-science/colour-datasets/3245895/dataset/
↪ColorCheckerSG_After_Nov2014.zip" archive...
Unpacking "/Users/kelsolaar/.colour-science/colour-datasets/3245895/dataset/
↪ColorChecker24_After_Nov2014.zip" archive...
Unpacking "/Users/kelsolaar/.colour-science/colour-datasets/3245895/dataset/
↪ColorChecker24_Before_Nov2014.zip" archive...
odict_keys(['ColorChecker24 - After November 2014', 'ColorChecker24 - Before November 2014
↪', 'ColorCheckerSG - After November 2014', 'ColorCheckerSG - Before November 2014'])
```

Alternatively, a dataset can be loaded by using its full title: *New Color Specifications for ColorChecker SG and Classic Charts - X-Rite (2016)*

```
>>> print(colour_datasets.load("3245895").keys())
odict_keys(['ColorChecker24 - After November 2014', 'ColorChecker24 - Before November 2014
↪', 'ColorCheckerSG - After November 2014', 'ColorCheckerSG - Before November 2014'])
```

CHAPTER

TWO

1.2 USER GUIDE

2.1 User Guide

The user guide provides an overview of **Colour - Datasets** and explains important concepts and features, details can be found in the API Reference.

2.1.1 Installation Guide

Primary Dependencies

Colour - Datasets requires various dependencies in order to run:

- python >= 3.9, < 4
- cachetools
- colour-science >= 4.3
- imageio >= 2, < 3
- numpy >= 1.22, < 2
- scipy >= 1.8, < 2
- tqdm
- xlrd

Pypi

Once the dependencies are satisfied, **Colour - Datasets** can be installed from the Python Package Index by issuing this command in a shell:

```
pip install --user colour-datasets
```

The overall development dependencies are installed as follows:

```
pip install --user 'colour-datasets[development]'
```

2.1.2 Bibliography

Indirect References

Some extra references used in the codebase but not directly part of the public api:

- [OpenpyxlDevelopers19]

CHAPTER

THREE

1.3 API REFERENCE

3.1 API Reference

3.1.1 Colour - Datasets

Datasets & Dataset Loading

Loading a Dataset

`colour_datasets`

`load`(dataset)	Load given dataset: The dataset is pulled locally, i.e. synced if required and then its data is loaded.

colour_datasets.load

`colour_datasets.`**`load`**(*dataset: int | str*) → Any

Load given dataset: The dataset is pulled locally, i.e. synced if required and then its data is loaded.

Parameters
`dataset` (`int | str`) – Dataset id, i.e. the *Zenodo* record number or title.

Returns
Dataset data.

Return type
`object`

Examples

```
>>> len(load("3245883").keys())
28
>>> len(
...     load(
...         "Camera Spectral Sensitivity Database - " "Jiang et al. (2013)"
...     ).keys()
... )
...
28
```

Ancillary Objects

`colour_datasets.loaders`

DATASET_LOADERS	Dataset loaders ids and callables.

colour_datasets.loaders.DATASET_LOADERS

colour_datasets.loaders.**DATASET_LOADERS**: CanonicalMapping = **CanonicalMapping({'3252742': ..., '4051012': ..., '3372171': ..., '3362536': ..., '3367463': ..., '4050598': ..., '3245883': ..., '4642271': ..., '3245875': ..., '4394536': ..., '3270903': ..., '8314702': ..., '6590768': ..., '3245895': ..., '4297288': ..., '3269912': ..., '3269914': ..., '3269916': ..., '3269918': ..., '3269920': ..., '3269922': ..., '3269924': ..., '3269926': ...})**

Dataset loaders ids and callables.

AbstractDatasetLoader(record)	Define the base class for a dataset loader.

colour_datasets.loaders.AbstractDatasetLoader

class colour_datasets.loaders.**AbstractDatasetLoader**(*record:* Record)

Bases: ABC

Define the base class for a dataset loader.

This is an ABCMeta abstract class that must be inherited by sub-classes.

The sub-classes are expected to implement the colour_datasets.loaders.AbstractDatasetLoader.load() method that handles the syncing, parsing, conversion and return of the dataset content as a *Python* object.

Attributes

- colour_datasets.loaders.AbstractDatasetLoader.ID
- colour_datasets.loaders.AbstractDatasetLoader.record
- colour_datasets.loaders.AbstractDatasetLoader.id
- colour_datasets.loaders.AbstractDatasetLoader.content

Methods

- colour_datasets.loaders.AbstractDatasetLoader.__init__()
- colour_datasets.loaders.AbstractDatasetLoader.load()
- colour_datasets.loaders.AbstractDatasetLoader.sync()

Parameters
record (Record) – Dataset record.

ID: str = **'Undefined'**

Dataset record id, i.e. the *Zenodo* record number.

property record: Record

Getter property for the dataset record.

Returns
Dataset record.

Return type
colour_datasets.Record

property id: str

Getter property for the dataset id.

Returns
Dataset id.

Return type
str

property content: Any

Getter property for the dataset content.

Returns
Dataset content.

Return type
object

abstract load() → Any

Sync, parse, convert and return the dataset content as a *Python* object.

Returns
Dataset content as a *Python* object.

Return type
object

Notes

- Sub-classes are required to call `colour_datasets.loaders.AbstractDatasetLoader.sync()` method when they implement it, e.g. `super().sync()`.

sync()

Sync the dataset content, i.e. checks whether it is synced and pulls it if required.

Datasets

`colour_datasets.loaders`

Spectral Upsampling Coefficient Tables - Jakob and Hanika (2019)

DatasetLoader_Jakob2019()	Define the *Jakob and Hanika (2019) Spectral Upsampling Coefficient Tables* dataset loader.

colour_datasets.loaders.DatasetLoader_Jakob2019

class colour_datasets.loaders.**DatasetLoader_Jakob2019**

Bases: AbstractDatasetLoader

Define the *Jakob and Hanika (2019) Spectral Upsampling Coefficient Tables* dataset loader.

Attributes

- colour_datasets.loaders.DatasetLoader_Jakob2019.ID

Methods

- colour_datasets.loaders.DatasetLoader_Jakob2019.__init__()
- colour_datasets.loaders.DatasetLoader_Jakob2019.load()

References

[JH19]

ID: str = '4050598'

Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, LUT3D_Jakob2019]

Sync, parse, convert and return the *Jakob and Hanika (2019) Spectral Upsampling Coefficient Tables* dataset content.

Returns
Jakob and Hanika (2019) Spectral Upsampling Coefficient Tables dataset content.

Return type
dict

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Jakob2019()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
4
```

build_Jakob2019([load])	Singleton factory that builds the *Jakob and Hanika (2019) Spectral Upsampling Coefficient Tables* dataset loader.

colour_datasets.loaders.build_Jakob2019

colour_datasets.loaders.**build_Jakob2019**(*load: bool = True*) → *DatasetLoader_Jakob2019*

Singleton factory that builds the *Jakob and Hanika (2019) Spectral Upsampling Coefficient Tables* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *Jakob and Hanika (2019) Spectral Upsampling Coefficient Tables* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_Jakob2019

References

[JH19]

Agfa IT8.7/2 Set - Marszalec (n.d.)

DatasetLoader_AgfaIT872Set()	Defines the *University of Kuopio Agfa IT8.7/2 Set* dataset loader.

colour_datasets.loaders.DatasetLoader_AgfaIT872Set

class colour_datasets.loaders.**DatasetLoader_AgfaIT872Set**

Bases: DatasetLoader_KuopioUniversity

Defines the *University of Kuopio Agfa IT8.7/2 Set* dataset loader.

Attributes

- colour_datasets.loaders.Agfa IT8.7/2 Set.ID
- colour_datasets.loaders.Agfa IT8.7/2 Set.METADATA

Methods

- colour_datasets.loaders.Agfa IT8.7/2 Set.__init__()
- colour_datasets.loaders.Agfa IT8.7/2 Set.load()

References

[MUniversityoKuopio]

ID: str = '3269926'

Dataset record id, i.e. the *Zenodo* record number.

METADATA: ClassVar[Dict] = {('agfait872_mat', 'agfait872.mat'): ('agfa', SpectralShape(400, 700, 10), True, None)}

Mapping of paths and `colour_datasets.loaders.kuopio.MatFileMetadata_KuopioUniversity` class instances.

`build_AgfaIT872Set`([load])	Singleton factory that the builds *University of Kuopio Agfa IT8.7/2 Set* dataset loader.

colour_datasets.loaders.build_AgfaIT872Set

colour_datasets.loaders.**build_AgfaIT872Set**(*load: bool = True*) → DatasetLoader_KuopioUniversity

Singleton factory that the builds *University of Kuopio Agfa IT8.7/2 Set* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *University of Kuopio Agfa IT8.7/2 Set* dataset loader.

Return type
DatasetLoader_AgfaIT872Set

References

[MUniversityoKuopio]

Camera Dataset - Solomatov and Akkaynak (2023)

`DatasetLoader_Solomotav2023`()	Define the *Solomatov and Akkaynak (2023) Camera Dataset* dataset loader.

colour_datasets.loaders.DatasetLoader_Solomotav2023

class colour_datasets.loaders.**DatasetLoader_Solomotav2023**

Bases: `AbstractDatasetLoader`

Define the *Solomatov and Akkaynak (2023) Camera Dataset* dataset loader.

Attributes

- colour_datasets.loaders.DatasetLoader_Solomotav2023.ID

Methods

- colour_datasets.loaders.DatasetLoader_Solomotav2023.__init__()
- colour_datasets.loaders.DatasetLoader_Solomotav2023.load()

References

[]

ID: str = '8314702'

Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, Dict[str, RGB_CameraSensitivities]]

Sync, parse, convert and return the *Solomatov and Akkaynak (2023) Camera Dataset* dataset content.

Returns
Solomatov and Akkaynak (2023) Camera Dataset dataset content.

Return type
dict

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Solomotav2023()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
2
>>> len(dataset.content["Estimated"].keys())
1012
```

build_Solomotav2023([load])	Singleton factory that builds the *Solomatov and Akkaynak (2023) Camera Dataset* dataset loader.

colour_datasets.loaders.build_Solomotav2023

colour_datasets.loaders.**build_Solomotav2023**(*load: bool = True*) → *DatasetLoader_Solomotav2023*

Singleton factory that builds the *Solomatov and Akkaynak (2023) Camera Dataset* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *Solomatov and Akkaynak (2023) Camera Dataset* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_Solomotav2023

References

[]

Camera Spectral Sensitivity Database - Jiang et al. (2013)

DatasetLoader_Jiang2013()	Define the *Jiang et al. (2013) Camera Spectral Sensitivity Database* dataset loader.

colour_datasets.loaders.DatasetLoader_Jiang2013

class colour_datasets.loaders.**DatasetLoader_Jiang2013**

Bases: AbstractDatasetLoader

Define the *Jiang et al. (2013) Camera Spectral Sensitivity Database* dataset loader.

Attributes

- colour_datasets.loaders.DatasetLoader_Jiang2013.ID

Methods

- colour_datasets.loaders.DatasetLoader_Jiang2013.__init__()
- colour_datasets.loaders.DatasetLoader_Jiang2013.load()

References

[JLGS13]

ID: str = '3245883'

Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, RGB_CameraSensitivities]

Sync, parse, convert and return the *Jiang et al. (2013) Camera Spectral Sensitivity Database* dataset content.

Returns
Jiang et al. (2013) Camera Spectral Sensitivity Database dataset content.

Return type
dict

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Jiang2013()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
28
```

build_Jiang2013([load])	Singleton factory that builds the *Jiang et al. (2013) Camera Spectral Sensitivity Database* dataset loader.

colour_datasets.loaders.build_Jiang2013

colour_datasets.loaders.**build_Jiang2013**(*load: bool = True*) → *DatasetLoader_Jiang2013*

Singleton factory that builds the *Jiang et al. (2013) Camera Spectral Sensitivity Database* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *Jiang et al. (2013) Camera Spectral Sensitivity Database* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_Jiang2013

References

[JLGS13]

Constant Hue Loci Data - Hung and Berns (1995)

DatasetLoader_Hung1995()	Define the *Hung and Berns (1995) Constant Hue Loci Data* dataset loader.

colour_datasets.loaders.DatasetLoader_Hung1995

class colour_datasets.loaders.**DatasetLoader_Hung1995**

Bases: AbstractDatasetLoader

Define the *Hung and Berns (1995) Constant Hue Loci Data* dataset loader.

Attributes

- colour_datasets.loaders.DatasetLoader_Hung1995.ID

Methods

- colour_datasets.loaders.DatasetLoader_Hung1995.__init__()
- colour_datasets.loaders.DatasetLoader_Hung1995.load()

References

[HB95]

ID: str = '3367463'

Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, Dict[str, ConstantPerceivedHueColourMatches_Hung1995]]

Sync, parse, convert and return the *Hung and Berns (1995) Constant Hue Loci Data* dataset content.

Returns

Hung and Berns (1995) Constant Hue Loci Data dataset content.

Return type

dict

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Hung1995()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
6
```

build_Hung1995([load])	Singleton factory that builds the *Hung and Berns (1995) Constant Hue Loci Data* dataset loader.

colour_datasets.loaders.build_Hung1995

colour_datasets.loaders.**build_Hung1995**(*load: bool = True*) → *DatasetLoader_Hung1995*

Singleton factory that builds the *Hung and Berns (1995) Constant Hue Loci Data* dataset loader.

Parameters

load (bool) – Whether to load the dataset upon instantiation.

Returns

Singleton instance of the *Hung and Berns (1995) Constant Hue Loci Data* dataset loader.

Return type

colour_datasets.loaders.DatasetLoader_Hung1995

References

[HB95]

Constant Perceived-Hue Data - Ebner and Fairchild (1998)

DatasetLoader_Ebner1998()	Define the *Ebner and Fairchild (1998) Constant Perceived-Hue Data* dataset loader.

colour_datasets.loaders.DatasetLoader_Ebner1998

class colour_datasets.loaders.**DatasetLoader_Ebner1998**

Bases: AbstractDatasetLoader

Define the *Ebner and Fairchild (1998) Constant Perceived-Hue Data* dataset loader.

Attributes

- colour_datasets.loaders.DatasetLoader_Ebner1998.ID

Methods

- colour_datasets.loaders.DatasetLoader_Ebner1998.__init__()
- colour_datasets.loaders.DatasetLoader_Ebner1998.load()

References

[EF98]

ID: str = '3362536'

Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, Dict[int, ConstantPerceivedHueColourMatches_Ebner1998]]

Sync, parse, convert and return the *Ebner and Fairchild (1998) Constant Perceived-Hue Data* dataset content.

Returns

Ebner and Fairchild (1998) Constant Perceived-Hue Data* dataset content.

Return type

dict

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Ebner1998()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
1
```

build_Ebner1998([load])	Singleton factory that builds the *Ebner and Fairchild (1998) Constant Perceived-Hue Data* dataset loader.

colour_datasets.loaders.build_Ebner1998

colour_datasets.loaders.**build_Ebner1998**(*load: bool = True*) → *DatasetLoader_Ebner1998*

Singleton factory that builds the *Ebner and Fairchild (1998) Constant Perceived-Hue Data* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *Ebner and Fairchild (1998) Constant Perceived-Hue Data* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_Ebner1998

References

[EF98]

Corresponding-Colour Datasets - Luo and Rhodes (1999)

DatasetLoader_Luo1999()	Define the *Luo and Rhodes (1999) Corresponding-Colour Datasets* dataset loader.

colour_datasets.loaders.DatasetLoader_Luo1999

class colour_datasets.loaders.**DatasetLoader_Luo1999**

Bases: AbstractDatasetLoader

Define the *Luo and Rhodes (1999) Corresponding-Colour Datasets* dataset loader.

Attributes

- colour_datasets.loaders.DatasetLoader_Luo1999.ID

Methods

- colour_datasets.loaders.DatasetLoader_Luo1999.__init__()
- colour_datasets.loaders.DatasetLoader_Luo1999.load()

References

[Bre87], [LR99], [MMT76]

ID: str = '3270903'

Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, CorrespondingColourDataset_Luo1999]

Sync, parse, convert and return the *Luo and Rhodes (1999) Corresponding-Colour Datasets* dataset content.

Returns
Luo and Rhodes (1999) Corresponding-Colour Datasets dataset content.

Return type
dict

Notes

- *Brene.p6.dat* has only 11 samples while *Breneman (1987)* has 12 results.
- The illuminance in Lux for *Breneman (1987)* datasets given by *Luo and Rhodes (1999)* is in domain [50, 3870] while *Breneman (1987)* reports luminance in cd/m^2 in domain [15, 11100], i.e. [47, 34871.69] in Lux. The metadata has been corrected accordingly.
- The illuminance values, i.e. 14 and 40, for *McCann, McKee and Taylor (1976)* datasets given by *Luo and Rhodes (1999)* were not found in [MMT76]. The values in use are the average of both.

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Luo1999()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
37
```

build_Luo1999([load])	Singleton factory that the builds *Luo and Rhodes (1999) Corresponding-Colour Datasets* dataset loader.

colour_datasets.loaders.build_Luo1999

colour_datasets.loaders.**build_Luo1999**(*load: bool = True*) → *DatasetLoader_Luo1999*

Singleton factory that the builds *Luo and Rhodes (1999) Corresponding-Colour Datasets* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *Luo and Rhodes (1999) Corresponding-Colour Datasets* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_Luo1999

References

[Bre87], [LR99], [MMT76]

Forest Colors - Jaaskelainen et al. (1994)

DatasetLoader_ForestColors()	Defines the *University of Kuopio Forest Colors* dataset loader.

colour_datasets.loaders.DatasetLoader_ForestColors

class colour_datasets.loaders.**DatasetLoader_ForestColors**

Bases: DatasetLoader_KuopioUniversity

Defines the *University of Kuopio Forest Colors* dataset loader.

Attributes

- colour_datasets.loaders.Forest Colors.ID
- colour_datasets.loaders.Forest Colors.METADATA

Methods

- colour_datasets.loaders.Forest Colors.__init__()
- colour_datasets.loaders.Forest Colors.load()

References

[SUniversityoKuopio]

ID: str = '3269920'

Dataset record id, i.e. the *Zenodo* record number.

METADATA: ClassVar[Dict] = {('forest_matlab', 'birch.mat'): ('birch', SpectralShape(380, 850, 5), True, None), ('forest_matlab', 'pine.mat'): ('pine', SpectralShape(380, 850, 5), True, None), ('forest_matlab', 'spruce.mat'): ('spruce', SpectralShape(380, 850, 5), True, None)}

Mapping of paths and `colour_datasets.loaders.kuopio.MatFileMetadata_KuopioUniversity` class instances.

`build_ForestColors`([load])	Singleton factory that the builds *University of Kuopio Forest Colors* dataset loader.

colour_datasets.loaders.build_ForestColors

colour_datasets.loaders.**build_ForestColors**(*load: bool = True*) → DatasetLoader_KuopioUniversity

Singleton factory that the builds *University of Kuopio Forest Colors* dataset loader.

Parameters
load (`bool`) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *University of Kuopio Forest Colors* dataset loader.

Return type
DatasetLoader_ForestColors

References

[SUniversityoKuopio]

LUTCHI Colour Appearance Data - Luo and Rhodes (1997)

`DatasetLoader_Luo1997`()	Define the *Luo and Rhodes (1997) LUTCHI Colour Appearance Data* dataset loader.

colour_datasets.loaders.DatasetLoader_Luo1997

class colour_datasets.loaders.**DatasetLoader_Luo1997**

Bases: `AbstractDatasetLoader`

Define the *Luo and Rhodes (1997) LUTCHI Colour Appearance Data* dataset loader.

Attributes

- colour_datasets.loaders.DatasetLoader_Luo1997.ID

Methods

- colour_datasets.loaders.DatasetLoader_Luo1997.__init__()
- colour_datasets.loaders.DatasetLoader_Luo1997.load()

References

[LCR+91a], [LCR+91b], [LGR+93], [LR97]

ID: str = '4394536'

Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, ExperimentalGroupLuo1997]

Sync, parse, convert and return the *Luo and Rhodes (1997) LUTCHI Colour Appearance Data* dataset content.

Returns

Luo and Rhodes (1997) LUTCHI Colour Appearance Data dataset content.

Return type

dict

Notes

- The *cold65wnl* file located at the following url: https://web.archive.org/web/20031230164218/http://colour.derby.ac.uk/colour/info/lutchi/data/cold65wnl is empty. Mark Fairchild's archive located at the following url: http://www.rit-mcsl.org/fairchild/files/LUTCHI_Data.sit also contains an empty *cold65wnl* file. A single line break has been added to the original file so that it can be uploaded to *Zenodo*.
- The *BIT.p*.** files are effectively named *bit_p*.**.
- The *cola.l* file does not exist and is assumed to be named *colal.l*.
- The *Self-luminous* entry for *Table I: Summary of the experimental groups* is named *CRT* in the sub-sequent tables.
- The *mean4.p** and *col.rf.p** files should all have 40 samples, unexpectedly all the *col.rf.p** files have 41 samples. The first data rows are used as they are better correlated between the two datasets. The last row could be the experimental whitepoint.
- The *mean4.p7*, *mean4.p8*, *mean4.p9*, *mean4.p10*, *mean4.p11*, and *mean4.p12* files represent brightness experimental results.
- The *bit_p3.vis* file has 5 columns instead of 4 only the last 3 are accounted for.

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Luo1997()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
8
```

build_Luo1997([load])	Singleton factory that the builds *Luo and Rhodes (1997) LUTCHI Colour Appearance Data* dataset loader.

colour_datasets.loaders.build_Luo1997

colour_datasets.loaders.**build_Luo1997**(*load: bool = True*) → *DatasetLoader_Luo1997*

Singleton factory that the builds *Luo and Rhodes (1997) LUTCHI Colour Appearance Data* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *Luo and Rhodes (1997) LUTCHI Colour Appearance Data* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_Luo1997

References

[LCR+91a], [LCR+91b], [LGR+93], [LR97]

Labsphere SRS-99-020 - Labsphere (2019)

DatasetLoader_Labsphere2019()	Define the *Labsphere (2019) Labsphere SRS-99-020* dataset loader.

colour_datasets.loaders.DatasetLoader_Labsphere2019

class colour_datasets.loaders.**DatasetLoader_Labsphere2019**

Bases: AbstractDatasetLoader

Define the *Labsphere (2019) Labsphere SRS-99-020* dataset loader.

Attributes

- colour_datasets.loaders.DatasetLoader_Labsphere2019.ID

Methods

- colour_datasets.loaders.DatasetLoader_Labsphere2019.__init__()
- colour_datasets.loaders.DatasetLoader_Labsphere2019.load()

References

[Labsphere19]

ID: str = '3245875'

Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, SpectralDistribution]

Sync, parse, convert and return the *Labsphere (2019) Labsphere SRS-99-020* dataset content.

Returns
Labsphere (2019) Labsphere SRS-99-020 dataset content.

Return type
dict

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Labsphere2019()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
1
```

build_Labsphere2019([load])	Singleton factory that builds the *Labsphere (2019) Labsphere SRS-99-020* dataset loader.

colour_datasets.loaders.build_Labsphere2019

colour_datasets.loaders.**build_Labsphere2019**(*load: bool = True*) → *DatasetLoader_Labsphere2019*

Singleton factory that builds the *Labsphere (2019) Labsphere SRS-99-020* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *Labsphere (2019) Labsphere SRS-99-020* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_Labsphere2019

References

[Labsphere19]

Lumber Spectra - Hiltunen (n.d.)

DatasetLoader_LumberSpectra()	Defines the *University of Kuopio Lumber Spectra* dataset loader.

colour_datasets.loaders.DatasetLoader_LumberSpectra

class colour_datasets.loaders.**DatasetLoader_LumberSpectra**

Bases: DatasetLoader_KuopioUniversity

Defines the *University of Kuopio Lumber Spectra* dataset loader.

Attributes

- colour_datasets.loaders.Lumber Spectra.ID
- colour_datasets.loaders.Lumber Spectra.METADATA

Methods

- colour_datasets.loaders.Lumber Spectra.__init__()
- colour_datasets.loaders.Lumber Spectra.load()

References

[HUniversityoKuopioc]

ID: str = '3269924'

Dataset record id, i.e. the *Zenodo* record number.

METADATA: ClassVar[Dict] = {('lumber_matlab', 'aspenWb.mat'): ('aspenWb', SpectralShape(380, 2700, 1), True, None), ('lumber_matlab', 'aspenWp.mat'): ('aspenWp', SpectralShape(380, 2700, 1), True, None), ('lumber_matlab', 'birchWb.mat'): ('birchWb', SpectralShape(380, 2700, 1), True, None), ('lumber_matlab', 'birchWp.mat'): ('birchWp', SpectralShape(380, 2700, 1), True, None), ('lumber_matlab', 'pineWb.mat'): ('pineWb', SpectralShape(380, 2700, 1), True, None), ('lumber_matlab', 'pineWp.mat'): ('pineWp', SpectralShape(380, 2700, 1), True, None), ('lumber_matlab', 'spruceWb.mat'): ('spruceWb', SpectralShape(380, 2700, 1), True, None), ('lumber_matlab', 'spruceWp.mat'): ('spruceWp', SpectralShape(380, 2700, 1), True, None)}

Mapping of paths and colour_datasets.loaders.kuopio.MatFileMetadata_KuopioUniversity class instances.

build_LumberSpectra([load])	Singleton factory that the builds *University of Kuopio Lumber Spectra* dataset loader.

colour_datasets.loaders.build_LumberSpectra

colour_datasets.loaders.**build_LumberSpectra**(*load: bool = True*) → DatasetLoader_KuopioUniversity

Singleton factory that the builds *University of Kuopio Lumber Spectra* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *University of Kuopio Lumber Spectra* dataset loader.

Return type
DatasetLoader_LumberSpectra

References

[HUniversityoKuopioc]

Measured Commercial LED Spectra - Brendel (2020)

DatasetLoader_Brendel2020()	Define the *Brendel (2020) Measured Commercial LED Spectra* dataset loader.

colour_datasets.loaders.DatasetLoader_Brendel2020

class colour_datasets.loaders.**DatasetLoader_Brendel2020**

Bases: AbstractDatasetLoader

Define the *Brendel (2020) Measured Commercial LED Spectra* dataset loader.

Attributes

ID

Methods

load

References

[Bre20]

ID: str = '4051012'
Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, SpectralDistribution]
Sync, parse, convert and return the *Brendel (2020) Measured Commercial LED Spectra* dataset content.

Returns
Brendel (2020) Measured Commercial LED Spectra dataset content.

Return type
dict

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Brendel2020()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
29
```

build_Brendel2020([load])	Singleton factory that builds the *Brendel (2020) Measured Commercial LED Spectra* dataset loader.

colour_datasets.loaders.build_Brendel2020

colour_datasets.loaders.**build_Brendel2020**(*load: bool = True*) → *DatasetLoader_Brendel2020*

Singleton factory that builds the *Brendel (2020) Measured Commercial LED Spectra* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *Brendel (2020) Measured Commercial LED Spectra* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_Brendel2020

References

[Bre20]

Munsell Colors Glossy (All) (Spectrofotometer Measured) - Orava (n.d.)

DatasetLoader_MunsellColorsGlossyAllSpectrof	Defines the *University of Kuopio Munsell Colors Glossy (All) (Spectrofotometer Measured)* dataset loader.

colour_datasets.loaders.DatasetLoader_MunsellColorsGlossyAllSpectrofotometerMeasured

class colour_datasets.loaders.**DatasetLoader_MunsellColorsGlossyAllSpectrofotometerMeasured**

Bases: DatasetLoader_KuopioUniversity

Defines the *University of Kuopio Munsell Colors Glossy (All) (Spectrofotometer Measured)* dataset loader.

Attributes

- colour_datasets.loaders.Munsell Colors Glossy (All) (Spectrofotometer Measured).ID
- colour_datasets.loaders.Munsell Colors Glossy (All) (Spectrofotometer Measured).METADATA

Methods

- colour_datasets.loaders.Munsell Colors Glossy (All) (Spectrofotometer Measured).__init__()
- colour_datasets.loaders.Munsell Colors Glossy (All) (Spectrofotometer Measured).load()

References

[OUniversityoKuopio]

ID: str = '3269918'

Dataset record id, i.e. the *Zenodo* record number.

METADATA: ClassVar[Dict] = {('munsell380_780_1_glossy_mat', 'munsell380_780_1_glossy.mat'): ('X', SpectralShape(380, 780, 1), True, None)}

Mapping of paths and colour_datasets.loaders.kuopio.MatFileMetadata_KuopioUniversity class instances.

build_MunsellColorsGlossyAllSpectrofotometer	Singleton factory that the builds *University of Kuopio Munsell Colors Glossy (All) (Spectrofotometer Measured)* dataset loader.

colour_datasets.loaders.build_MunsellColorsGlossyAllSpectrofotometerMeasured

colour_datasets.loaders.**build_MunsellColorsGlossyAllSpectrofotometerMeasured**(*load: bool = True*) → DatasetLoader_KuopioUniversity

Singleton factory that the builds *University of Kuopio Munsell Colors Glossy (All) (Spectrofotometer Measured)* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *University of Kuopio Munsell Colors Glossy (All) (Spectrofotometer Measured)* dataset loader.

Return type
DatasetLoader_MunsellColorsGlossyAllSpectrofotometerMeasured

References

[OUniversityoKuopio]

Munsell Colors Glossy (Spectrofotometer Measured) - Haanpalo (n.d.)

DatasetLoader_MunsellColorsGlossySpectrofoto	Defines the *University of Kuopio Munsell Colors Glossy (Spectrofotometer Measured)* dataset loader.

colour_datasets.loaders.DatasetLoader_MunsellColorsGlossySpectrofotometerMeasured

class colour_datasets.loaders.**DatasetLoader_MunsellColorsGlossySpectrofotometerMeasured**

Bases: DatasetLoader_KuopioUniversity

Defines the *University of Kuopio Munsell Colors Glossy (Spectrofotometer Measured)* dataset loader.

Attributes

- colour_datasets.loaders.Munsell Colors Glossy (Spectrofotometer Measured).ID
- colour_datasets.loaders.Munsell Colors Glossy (Spectrofotometer Measured).METADATA

Methods

- colour_datasets.loaders.Munsell Colors Glossy (Spectrofotometer Measured).__init__()
- colour_datasets.loaders.Munsell Colors Glossy (Spectrofotometer Measured).load()

References

[HUniversityoKuopioa]

ID: str = '3269916'

Dataset record id, i.e. the *Zenodo* record number.

METADATA: ClassVar[Dict] = {('munsell400_700_10_mat', 'munsell400_700_10.mat'): ('munsell', SpectralShape(400, 700, 10), True, 'S')}

Mapping of paths and colour_datasets.loaders.kuopio.MatFileMetadata_KuopioUniversity class instances.

build_MunsellColorsGlossySpectrofotometerMea	Singleton factory that the builds *University of Kuopio Munsell Colors Glossy (Spectrofotometer Measured)* dataset loader.

colour_datasets.loaders.build_MunsellColorsGlossySpectrofotometerMeasured

colour_datasets.loaders.**build_MunsellColorsGlossySpectrofotometerMeasured**(*load: bool = True*) → DatasetLoader_KuopioUniversity

Singleton factory that the builds *University of Kuopio Munsell Colors Glossy (Spectrofotometer Measured)* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *University of Kuopio Munsell Colors Glossy (Spectrofotometer Measured)* dataset loader.

Return type
DatasetLoader_MunsellColorsGlossySpectrofotometerMeasured

References

[HUniversityoKuopioa]

Munsell Colors Matt (AOTF Measured) - Hauta-Kasari (n.d.)

DatasetLoader_MunsellColorsMattAOTFMeasured(	Defines the *University of Kuopio Munsell Colors Matt (AOTF Measured)* dataset loader.

colour_datasets.loaders.DatasetLoader_MunsellColorsMattAOTFMeasured

class colour_datasets.loaders.**DatasetLoader_MunsellColorsMattAOTFMeasured**

Bases: DatasetLoader_KuopioUniversity

Defines the *University of Kuopio Munsell Colors Matt (AOTF Measured)* dataset loader.

Attributes

- colour_datasets.loaders.Munsell Colors Matt (AOTF Measured).ID
- colour_datasets.loaders.Munsell Colors Matt (AOTF Measured).METADATA

Methods

- colour_datasets.loaders.Munsell Colors Matt (AOTF Measured).__init__()
- colour_datasets.loaders.Munsell Colors Matt (AOTF Measured).load()

References

[HautaKasariUniversityoKuopioa]

ID: str = '3269914'

Dataset record id, i.e. the *Zenodo* record number.

METADATA: ClassVar[Dict] = {('munsell400_700_5_mat', 'munsell400_700_5.mat'): ('munsell', SpectralShape(400, 700, 5), True, 'S')}

Mapping of paths and `colour_datasets.loaders.kuopio.MatFileMetadata_KuopioUniversity` class instances.

`build_MunsellColorsMattAOTFMeasured`([load])	Singleton factory that the builds *University of Kuopio Munsell Colors Matt (AOTF Measured)* dataset loader.

colour_datasets.loaders.build_MunsellColorsMattAOTFMeasured

colour_datasets.loaders.**build_MunsellColorsMattAOTFMeasured**(*load: bool = True*) → DatasetLoader_KuopioUniversity

Singleton factory that the builds *University of Kuopio Munsell Colors Matt (AOTF Measured)* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *University of Kuopio Munsell Colors Matt (AOTF Measured)* dataset loader.

Return type
DatasetLoader_MunsellColorsMattAOTFMeasured

References

[HautaKasariUniversityoKuopioa]

Munsell Colors Matt (Spectrofotometer Measured) - Hauta-Kasari (n.d.)

`DatasetLoader_MunsellColorsMattSpectrofotome`	Defines the *University of Kuopio Munsell Colors Matt (Spectrofotometer Measured)* dataset loader.

colour_datasets.loaders.DatasetLoader_MunsellColorsMattSpectrofotometerMeasured

class colour_datasets.loaders.**DatasetLoader_MunsellColorsMattSpectrofotometerMeasured**

Bases: `DatasetLoader_KuopioUniversity`

Defines the *University of Kuopio Munsell Colors Matt (Spectrofotometer Measured)* dataset loader.

Attributes

- colour_datasets.loaders.Munsell Colors Matt (Spectrofotometer Measured).ID
- colour_datasets.loaders.Munsell Colors Matt (Spectrofotometer Measured).METADATA

Methods

- colour_datasets.loaders.Munsell Colors Matt (Spectrofotometer Measured).__init__()
- colour_datasets.loaders.Munsell Colors Matt (Spectrofotometer Measured).load()

References

[HautaKasariUniversityoKuopiob]

ID: str = '3269912'

Dataset record id, i.e. the *Zenodo* record number.

METADATA: ClassVar[Dict] = {('munsell380_800_1_mat', 'munsell380_800_1.mat'): ('munsell', SpectralShape(380, 800, 1), True, 'S')}

Mapping of paths and colour_datasets.loaders.kuopio.MatFileMetadata_KuopioUniversity class instances.

build_MunsellColorsMattSpectrofotometerMeasu	Singleton factory that the builds *University of Kuopio Munsell Colors Matt (Spectrofotometer Measured)* dataset loader.

colour_datasets.loaders.build_MunsellColorsMattSpectrofotometerMeasured

colour_datasets.loaders.**build_MunsellColorsMattSpectrofotometerMeasured**(*load: bool = True*) → DatasetLoader_KuopioUniversity

Singleton factory that the builds *University of Kuopio Munsell Colors Matt (Spectrofotometer Measured)* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *University of Kuopio Munsell Colors Matt (Spectrofotometer Measured)* dataset loader.

Return type
DatasetLoader_MunsellColorsMattSpectrofotometerMeasured

References

[HautaKasariUniversityoKuopiob]

New Color Specifications for ColorChecker SG and Classic Charts - X-Rite (2016)

DatasetLoader_XRite2016()	Define the *X-Rite (2016) New Color Specifications for ColorChecker SG and Classic Charts* dataset loader.

colour_datasets.loaders.DatasetLoader_XRite2016

class colour_datasets.loaders.**DatasetLoader_XRite2016**

Bases: AbstractDatasetLoader

Define the *X-Rite (2016) New Color Specifications for ColorChecker SG and Classic Charts* dataset loader.

Attributes

- colour_datasets.loaders.DatasetLoader_XRite2016.ID

Methods

- colour_datasets.loaders.DatasetLoader_XRite2016.__init__()
- colour_datasets.loaders.DatasetLoader_XRite2016.load()

References

[XRite16]

ID: str = '3245895'

Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, ColourChecker]

Sync, parse, convert and return the *X-Rite (2016) New Color Specifications for ColorChecker SG and Classic Charts* dataset content.

Returns

X-Rite (2016) New Color Specifications for ColorChecker SG and Classic Charts dataset content.

Return type

dict

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_XRite2016()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
4
```

build_XRite2016([load])	Singleton factory that the builds *X-Rite (2016) New Color Specifications for ColorChecker SG and Classic Charts* dataset loader.

colour_datasets.loaders.build_XRite2016

colour_datasets.loaders.**build_XRite2016**(*load: bool = True*) → *DatasetLoader_XRite2016*

Singleton factory that the builds *X-Rite (2016) New Color Specifications for ColorChecker SG and Classic Charts* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *X-Rite (2016) New Color Specifications for ColorChecker SG and Classic Charts* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_XRite2016

References

[XRite16]

Observer Function Database - Asano (2015)

DatasetLoader_Asano2015()	Define the *Asano (2015) Observer Function Database* dataset loader.

colour_datasets.loaders.DatasetLoader_Asano2015

class colour_datasets.loaders.**DatasetLoader_Asano2015**

Bases: AbstractDatasetLoader

Define the *Asano (2015) Observer Function Database* dataset loader.

Attributes

- colour_datasets.loaders.DatasetLoader_Asano2015.ID

Methods

- colour_datasets.loaders.DatasetLoader_Asano2015.__init__()
- colour_datasets.loaders.DatasetLoader_Asano2015.load()
- colour_datasets.loaders.DatasetLoader_Asano2015.parse_workbook_Asano2015()

References

[Asa15]

ID: str = '3252742'

Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, Dict[int, Specification_Asano2015]]

Sync, parse, convert and return the *Asano (2015) Observer Function Database* dataset content.

Returns

Asano (2015) Observer Function Database dataset content.

Return type

dict

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Asano2015()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
2
```

static parse_workbook_Asano2015(*workbook: str*, *template: str*, *observers: tuple = (1, 10)*) → Dict[str, Dict]

Parse given *Asano (2015) Observer Function Database* workbook.

Parameters

- **workbook** (str) – *Asano (2015) Observer Function Database* workbook path.
- **template** (str) – Template used to create the *CMFS* names.
- **observers** (tuple) – Observers range.

Returns

Asano (2015) Observer Function Database workbook observer data.

Return type

dict

build_Asano2015([load])	Singleton factory that the builds *Asano (2015) Observer Function Database* dataset loader.

colour_datasets.loaders.build_Asano2015

colour_datasets.loaders.**build_Asano2015**(*load: bool = True*) → *DatasetLoader_Asano2015*

Singleton factory that the builds *Asano (2015) Observer Function Database* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *Asano (2015) Observer Function Database* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_Asano2015

References

[Asa15]

Paper Spectra - Haanpalo (n.d.)

DatasetLoader_PaperSpectra()	Defines the *University of Kuopio Paper Spectra* dataset loader.

colour_datasets.loaders.DatasetLoader_PaperSpectra

class colour_datasets.loaders.**DatasetLoader_PaperSpectra**

Bases: DatasetLoader_KuopioUniversity

Defines the *University of Kuopio Paper Spectra* dataset loader.

Attributes

- colour_datasets.loaders.Paper Spectra.ID
- colour_datasets.loaders.Paper Spectra.METADATA

Methods

- colour_datasets.loaders.Paper Spectra.__init__()
- colour_datasets.loaders.Paper Spectra.load()

References

[HUniversityoKuopiob]

ID: str = '3269922'

Dataset record id, i.e. the *Zenodo* record number.

METADATA: ClassVar[Dict] = {('paper_matlab', 'cardboardsce.mat'): ('cardboardsce', SpectralShape(400, 700, 10), True, None), ('paper_matlab', 'cardboardsci.mat'): ('cardboardsci', SpectralShape(400, 700, 10), True, None), ('paper_matlab', 'mirrorsci.mat'): ('mirrorsci', SpectralShape(400, 700, 10), True, None), ('paper_matlab', 'newsprintsce.mat'): ('newsprintsce', SpectralShape(400, 700, 10), True, None), ('paper_matlab', 'newsprintsci.mat'): ('newsprintsci', SpectralShape(400, 700, 10), True, None), ('paper_matlab', 'papersce.mat'): ('papersce', SpectralShape(400, 700, 10), True, None), ('paper_matlab', 'papersci.mat'): ('papersci', SpectralShape(400, 700, 10), True, None)}

Mapping of paths and `colour_datasets.loaders.kuopio.MatFileMetadata_KuopioUniversity` class instances.

`build_PaperSpectra`([load])	Singleton factory that the builds *University of Kuopio Paper Spectra* dataset loader.

colour_datasets.loaders.build_PaperSpectra

`colour_datasets.loaders.`**`build_PaperSpectra`**(*load: bool = True*) → DatasetLoader_KuopioUniversity

Singleton factory that the builds *University of Kuopio Paper Spectra* dataset loader.

Parameters
load (`bool`) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *University of Kuopio Paper Spectra* dataset loader.

Return type
DatasetLoader_PaperSpectra

References

[HUniversityoKuopiob]

Physlight - Camera Spectral Sensitivity Curves - Winquist et al. (2022)

`DatasetLoader_Winquist2022`()	Define the *Winquist et al. (2022) Physlight - Camera Spectral Sensitivity Curves* dataset /loader.

colour_datasets.loaders.DatasetLoader_Winquist2022

class `colour_datasets.loaders.`**`DatasetLoader_Winquist2022`**

Bases: `AbstractDatasetLoader`

Define the *Winquist et al. (2022) Physlight - Camera Spectral Sensitivity Curves* dataset /loader.

Attributes

ID

Methods

load

References

[WTWetaDigital22]

ID: str = '6590768'
Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, MultiSpectralDistributions_AMPAS]
Sync, parse, convert and return the *Winquist et al. (2022) Physlight - Camera Spectral Sensitivity Curves* dataset content.

Returns
Winquist et al. (2022) Physlight - Camera Spectral Sensitivity Curves dataset content.

Return type
dict

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Winquist2022()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
17
```

build_Winquist2022([load])	Singleton factory that builds the *Winquist et al. (2022) Physlight - Camera Spectral Sensitivity Curves* dataset loader.

colour_datasets.loaders.build_Winquist2022

colour_datasets.loaders.**build_Winquist2022**(*load: bool = True*) → *DatasetLoader_Winquist2022*
Singleton factory that builds the *Winquist et al. (2022) Physlight - Camera Spectral Sensitivity Curves* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *Winquist et al. (2022) Physlight - Camera Spectral Sensitivity Curves* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_Winquist2022

References

[WTWetaDigital22]

RAW to ACES Utility Data - Dyer et al. (2017)

DatasetLoader_Dyer2017()	Define the *Dyer et al. (2017) RAW to ACES Utility Data* dataset loader.

colour_datasets.loaders.DatasetLoader_Dyer2017

class colour_datasets.loaders.**DatasetLoader_Dyer2017**

Bases: AbstractDatasetLoader

Define the *Dyer et al. (2017) RAW to ACES Utility Data* dataset loader.

Attributes

- colour_datasets.loaders.DatasetLoader_Dyer2017.ID

Methods

- colour_datasets.loaders.DatasetLoader_Dyer2017.__init__()
- colour_datasets.loaders.DatasetLoader_Dyer2017.load()

References

[DFI+17]

ID: str = '3372171'

Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, Dict[str, SpectralDistribution_AMPAS | MultiSpectralDistributions_AMPAS]]

Sync, parse, convert and return the *Dyer et al. (2017) RAW to ACES Utility Data* dataset content.

Returns

Dyer et al. (2017) RAW to ACES Utility Data dataset content.

Return type

dict

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Dyer2017()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
4
```

build_Dyer2017([load])	Singleton factory that builds the *Dyer et al. (2017) RAW to ACES Utility Data* dataset loader.

colour_datasets.loaders.build_Dyer2017

colour_datasets.loaders.**build_Dyer2017**(*load: bool = True*) → *DatasetLoader_Dyer2017*

Singleton factory that builds the *Dyer et al. (2017) RAW to ACES Utility Data* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *Dyer et al. (2017) RAW to ACES Utility Data* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_Dyer2017

References

[DFI+17]

Spectral Database of Commonly Used Cine Lighting - Karge et al. (2015)

DatasetLoader_Zhao2009()	Define the *Zhao et al. (2009) Spectral Sensitivity Database* dataset loader.

colour_datasets.loaders.DatasetLoader_Zhao2009

class colour_datasets.loaders.**DatasetLoader_Zhao2009**

Bases: AbstractDatasetLoader

Define the *Zhao et al. (2009) Spectral Sensitivity Database* dataset loader.

Attributes

- colour_datasets.loaders.DatasetLoader_Zhao2009.ID

Methods

- colour_datasets.loaders.DatasetLoader_Zhao2009.__init__()
- colour_datasets.loaders.DatasetLoader_Zhao2009.load()

References

[ZKTI09]

ID: str = '4297288'

Dataset record id, i.e. the *Zenodo* record number.

load() → Dict[str, RGB_CameraSensitivities]

Sync, parse, convert and return the *Zhao et al. (2009) Spectral Sensitivity Database* dataset content.

Returns
Zhao et al. (2009) Spectral Sensitivity Database dataset content.

Return type
dict

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> dataset = DatasetLoader_Zhao2009()
>>> with suppress_stdout():
...     dataset.load()
...
>>> len(dataset.content.keys())
12
```

build_Zhao2009([load])	Singleton factory that builds the *Zhao et al. (2009) Spectral Sensitivity Database* dataset loader.

colour_datasets.loaders.build_Zhao2009

colour_datasets.loaders.**build_Zhao2009**(*load: bool = True*) → *DatasetLoader_Zhao2009*

Singleton factory that builds the *Zhao et al. (2009) Spectral Sensitivity Database* dataset loader.

Parameters
load (bool) – Whether to load the dataset upon instantiation.

Returns
Singleton instance of the *Zhao et al. (2009) Spectral Sensitivity Database* dataset loader.

Return type
colour_datasets.loaders.DatasetLoader_Zhao2009

References

[ZKTI09]

Spectral Sensitivity Database - Zhao et al. (2009)

DatasetLoader_Zhao2009()	Define the *Zhao et al. (2009) Spectral Sensitivity Database* dataset loader.

build_Zhao2009([load])	Singleton factory that builds the *Zhao et al. (2009) Spectral Sensitivity Database* dataset loader.

Zenodo Records & Community

Configuration

colour_datasets

Configuration([configuration])	*Colour - Datasets* configuration factory based on colour.utilities.Structure class and allowing to access key values using dot syntax.

colour_datasets.Configuration

class colour_datasets.**Configuration**(*configuration: Dict | None = None*)

Bases: Structure

Colour - Datasets configuration factory based on colour.utilities.Structure class and allowing to access key values using dot syntax.

Parameters

configuration (Dict | None) – Configuration to use instead of the default one.

sandbox([api_url, community])	A context manager and decorator temporarily setting the configuration to the *Zenodo* sandbox.

colour_datasets.sandbox

class colour_datasets.**sandbox**(*api_url: str = 'https://sandbox.zenodo.org/api'*, *community: str = 'colour-science-datasets'*)

A context manager and decorator temporarily setting the configuration to the *Zenodo* sandbox.

Parameters

- **api_url** (str) – *Zenodo* sandbox url.
- **community** (str) – *Zenodo* community.

__init__(*api_url: str = 'https://sandbox.zenodo.org/api'*, *community: str = 'colour-science-datasets'*) → None

Parameters

- **api_url** (str) –
- **community** (str) –

Return type
None

Methods

__init__([api_url, community])	

colour_datasets.records

use_sandbox([state, api_url, community])	Modify the *Colour - Datasets* configuration to use *Zenodo* sandbox.

colour_datasets.records.use_sandbox

colour_datasets.records.**use_sandbox**(*state: bool = True*, *api_url: str = 'https://sandbox.zenodo.org/api'*, *community: str = 'colour-science-datasets'*)

Modify the *Colour - Datasets* configuration to use *Zenodo* sandbox.

Parameters

- **state** (bool) – Whether to use *Zenodo* sandbox.
- **api_url** (str) – *Zenodo* sandbox url.
- **community** (str) – *Zenodo* community.

Record

colour_datasets

Record(data[, configuration])	Define an object storing a *Zenodo* record data and providing methods to sync it in a local repository.

colour_datasets.Record

class colour_datasets.**Record**(*data: dict*, *configuration: Configuration | None = None*)

Bases: object

Define an object storing a *Zenodo* record data and providing methods to sync it in a local repository.

Parameters

- **data** (dict) – *Zenodo* record data.
- **configuration** (Configuration | None) – *Colour - Datasets* configuration.

Attributes

- colour_datasets.Record.data
- colour_datasets.Record.configuration
- colour_datasets.Record.repository
- colour_datasets.Record.id
- colour_datasets.Record.title

Methods

- colour_datasets.Record.__init__()
- colour_datasets.Record.__str__()
- colour_datasets.Record.__repr__()
- colour_datasets.Record.from_id()
- colour_datasets.Record.synced()
- colour_datasets.Record.pull()
- colour_datasets.Record.remove()

Examples

```
>>> record = Record(json_open("https://zenodo.org/api/records/3245883"))
>>> record.id
'3245883'
>>> record.title
'Camera Spectral Sensitivity Database - Jiang et al. (2013)'
```

property data: dict

Getter property for the *Zenodo* record data.

Returns
Zenodo record data.

Return type
dict

property configuration: Configuration

Getter property for the *Colour - Datasets* configuration.

Returns
Colour - Datasets configuration.

Return type
colour_datasets.Configuration

property repository: str

Getter property for the *Zenodo* record local repository.

Returns
Zenodo record local repository.

Return type
str

property id: str

Getter property for the *Zenodo* record id.

Returns
Zenodo record id.

Return type
str

property title: str

Getter property for the *Zenodo* record title.

Returns
Zenodo record title.

Return type
str

static from_id(*id_: str*, *configuration: Configuration | None = None*, *retries: int = 3*) → *Record*

`colour_datasets.Record` class factory that builds an instance using given *Zenodo* record id.

Parameters

- **id** – *Zenodo* record id.
- **configuration** (Configuration | None) –

  configuration
  Colour - Datasets configuration.
- **retries** (int) – Number of retries in case where a networking error occurs.
- **id_** (str) –

Returns
Zenodo record data.

Return type
colour_datasets.Record

Examples

```
>>> Record.from_id("3245883").title
'Camera Spectral Sensitivity Database - Jiang et al. (2013)'
```

synced() → bool

Return whether the *Zenodo* record data is synced to the local repository.

Returns
Whether the *Zenodo* record data is synced to the local repository.

Return type
bool

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> record = Record.from_id("3245883")
>>> with suppress_stdout():
...     record.pull()
...
>>> record.synced()
True
>>> record.remove()
>>> record.synced()
False
```

pull(*use_urls_txt_file: bool = True*, *retries: int = 3*)

Pull the *Zenodo* record data to the local repository.

Parameters

- **use_urls_txt_file** (bool) – Whether to use the *urls.txt* file: if such a file is present in the *Zenodo* record data, the urls it defines take precedence over the record data files. The later will be used in the eventuality where the urls are not available.
- **retries** (int) – Number of retries in case where a networking error occurs or the *MD5* hash is not matching.

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> record = Record.from_id("3245883")
>>> record.remove()
>>> with suppress_stdout():
...     record.pull()
...
>>> record.synced()
True
```

remove()

Remove the *Zenodo* record data local repository.

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> record = Record.from_id("3245883")
>>> with suppress_stdout():
...     record.pull()
...
>>> record.remove()
>>> record.synced()
False
```

Community

colour_datasets

Community(data[, configuration])	Define an object storing a *Zenodo* community data.

colour_datasets.Community

class colour_datasets.**Community**(*data: Dict*, *configuration:* Configuration | *None* = *None*)

Bases: Mapping

Define an object storing a *Zenodo* community data.

Parameters

- **data** (Dict) – *Zenodo* community data.
- **configuration** (Configuration | None) – *Colour - Datasets* configuration.

Attributes

- colour_datasets.Community.data
- colour_datasets.Community.configuration
- colour_datasets.Community.repository
- colour_datasets.Community.records

Methods

- colour_datasets.Community.__init__()
- colour_datasets.Community.__str__()
- colour_datasets.Community.__repr__()
- colour_datasets.Community.__getitem__()
- colour_datasets.Community.__iter__()
- colour_datasets.Community.__len__()
- colour_datasets.Community.from_id()
- colour_datasets.Community.synced()
- colour_datasets.Community.pull()
- colour_datasets.Community.remove()

Examples

```
>>> community_data = json_open(
...     "https://zenodo.org/api/communities/colour-science-datasets"
... )
>>> records_data = json_open(community_data["links"]["records"])
>>> community = Community(
...     {
...         "community": community_data,
...         "records": records_data,
...     }
... )
>>> community["3245883"].title
'Camera Spectral Sensitivity Database - Jiang et al. (2013)'
```

property data: Dict

Getter property for the *Zenodo* community data.

Returns
Zenodo community data.

Return type
dict

property configuration: Configuration

Getter property for the *Colour - Datasets* configuration.

Returns
Colour - Datasets configuration.

Return type
colour_datasets.Configuration

property repository: str

Getter property for the *Zenodo* community local repository.

Returns
Zenodo community local repository.

Return type
str

property records: Dict

Getter property for the *Zenodo* community records.

Returns
Zenodo community records.

Return type
dict

static from_id(*id_: str*, *configuration:* Configuration | *None = None*, *retries: int = 3*) → *Community*

colour_datasets.Community class factory that builds an instance using given *Zenodo* community id.

Parameters

- **id** – *Zenodo* community id.
- **configuration** (Configuration | None) –

  configuration :
  Colour - Datasets configuration.

- **retries** (int) – Number of retries in case where a networking error occurs.
- **id_** (str) –

Returns
Zenodo community data.

Return type
colour_datasets.Community

Examples

```
>>> community = Community.from_id("colour-science-datasets-tests")
>>> community["3245883"].title
'Camera Spectral Sensitivity Database - Jiang et al. (2013)'
```

synced() → bool

Return whether the *Zenodo* community data is synced to the local repository.

Returns
Whether the *Zenodo* community data is synced to the local repository.

Return type
bool

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> community = Community.from_id("colour-science-datasets-tests")
>>> with suppress_stdout():
...     community.pull()
...
>>> community.synced()
True
>>> community.remove()
>>> community.synced()
False
```

pull(*use_urls_txt_file: bool = True*, *retries: int = 3*)

Pull the *Zenodo* community data to the local repository.

Parameters

- **use_urls_txt_file** (bool) – Whether to use the *urls.txt* file: if such a file is present in a *Zenodo* record data, the urls it defines take precedence over the record data files. The later will be used in the eventuality where the urls are not available.
- **retries** (int) – Number of retries in case where a networking error occurs or the *MD5* hash is not matching.

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> community = Community.from_id("colour-science-datasets-tests")
>>> community.remove()
>>> with suppress_stdout():
...     community.pull()
...
>>> community.synced()
True
```

remove()

Remove the *Zenodo* community data local repository.

Examples

```
>>> from colour_datasets.utilities import suppress_stdout
>>> community = Community.from_id("colour-science-datasets-tests")
>>> with suppress_stdout():
...     community.pull()
...
>>> community.remove()
>>> community.synced()
False
```

datasets()	Singleton factory that returns *Zenodo* community that holds the datasets information.

colour_datasets.datasets

colour_datasets.**datasets**() → *Community*

Singleton factory that returns *Zenodo* community that holds the datasets information.

Returns

Singleton instance of the *Zenodo* community.

Return type

colour_datasets.Community

Examples

```
>>> datasets()["3245883"].title
'Camera Spectral Sensitivity Database - Jiang et al. (2013)'
```

Utilities

Common

`colour_datasets.utilities`

`json_open`(url[, retries])	Open given url and return its content as *JSON*.
`hash_md5`(filename[, chunk_size])	Compute the *Message Digest 5 (MD5)* hash of given file.
`suppress_stdout`()	A context manager and decorator temporarily suppressing standard output.
`url_download`(url, filename[, md5, retries])	Download given url and saves its content at given file.

colour_datasets.utilities.json_open

`colour_datasets.utilities.`**`json_open`**(*url: str*, *retries: int = 3*) → Dict

Open given url and return its content as *JSON*.

Parameters

- **`url`** (`str`) – Url to open.
- **`retries`** (`int`) – Number of retries in case where a networking error occurs.

Returns
JSON data.

Return type
`dict`

Raises
`urllib.error.URLError`, `ValueError` – If the url cannot be opened or parsed as *JSON*.

Notes

- The definition caches the request *JSON* output for 5 minutes.

Examples

```
>>> json_open("https://zenodo.org/api/records/3245883")
...
'{"conceptdoi":"10.5281/zenodo.3245882"'
```

colour_datasets.utilities.hash_md5

`colour_datasets.utilities.`**`hash_md5`**(*filename: str*, *chunk_size: int = 2**16*) → str

Compute the *Message Digest 5 (MD5)* hash of given file.

Parameters

- **`filename`** (`str`) – File to compute the *MD5* hash of.
- **`chunk_size`** (`int`) – Chunk size to read from the file.

Returns
MD5 hash of given file.

Return type
str

colour_datasets.utilities.suppress_stdout

class colour_datasets.utilities.**suppress_stdout**
A context manager and decorator temporarily suppressing standard output.

__init__()

Methods

__init__()

colour_datasets.utilities.url_download

colour_datasets.utilities.**url_download**(*url: str*, *filename: str*, *md5: str | None = None*, *retries: int = 3*)
Download given url and saves its content at given file.

Parameters

- **url** (str) – Url to download.
- **filename** (str) – File to save the url content at.
- **md5** (str | None) – *Message Digest 5 (MD5)* hash of the content at given url. If provided the saved content at given file will be hashed and compared to md5.
- **retries** (int) – Number of retries in case where a networking error occurs or the *MD5* hash is not matching.

Examples

```
>>> import os
>>> url_download(
...     "https://github.com/colour-science/colour-datasets", os.devnull
... )
```

Spreadsheet

colour_datasets.utilities

row_to_index(row)	Return the 0-based index of given row name.
index_to_row(index)	Return the row name of given 0-based index.
column_to_index(column)	Return the 0-based index of given column letters.
index_to_column(index)	Return the column letters of given 0-based index.
cell_range_values(sheet, cell_range)	Return given workbook sheet cell range values, i.e. the values of the rows and columns for given cell range.

colour_datasets.utilities.row_to_index

colour_datasets.utilities.**row_to_index**(*row: int | str*) → int

Return the 0-based index of given row name.

Parameters
row (int | str) – Row name.

Returns
0-based row index.

Return type
class`int` or str

Examples

```
>>> row_to_index("1")
0
```

colour_datasets.utilities.index_to_row

colour_datasets.utilities.**index_to_row**(*index: int*) → str

Return the row name of given 0-based index.

Parameters
index (int) – 0-based row index.

Returns
Row name.

Return type
str

Examples

```
>>> index_to_row(0)
'1'
```

colour_datasets.utilities.column_to_index

colour_datasets.utilities.**column_to_index**(*column: str*) → int

Return the 0-based index of given column letters.

Parameters
column (str) – Column letters

Returns
0-based column index.

Return type
int

Examples

```
>>> column_to_index("A")
0
```

colour_datasets.utilities.index_to_column

colour_datasets.utilities.**index_to_column**(*index: int*) → str

Return the column letters of given 0-based index.

Parameters
index (int) – 0-based column index.

Returns
Column letters

Return type
str

Examples

```
>>> index_to_column(0)
'A'
```

colour_datasets.utilities.cell_range_values

colour_datasets.utilities.**cell_range_values**(*sheet: Sheet*, *cell_range: str*) → List[str]

Return given workbook sheet cell range values, i.e. the values of the rows and columns for given cell range.

Parameters

- **sheet** (Sheet) – Workbook sheet.
- **cell_range** (str) – Cell range values, e.g. “A1:C3”.

Returns
List of row values.

Return type
list

3.1.2 Indices and tables

- genindex
- search

CHAPTER

FOUR

1.4 CODE OF CONDUCT

The *Code of Conduct*, adapted from the Contributor Covenant 1.4, is available on the Code of Conduct page.

CHAPTER

FIVE

1.5 CONTACT & SOCIAL

The *Colour Developers* can be reached via different means:

- Email
- Facebook
- Github Discussions
- Gitter
- Twitter

CHAPTER

SIX

1.6 ABOUT

BIBLIOGRAPHY

[Asa15] Yuta Asano. *Individual Colorimetric Observers for Personalized Color Imaging*. PhD thesis, R.I.T., 2015.

[Bre20] Harald Brendel. Measured Commercial LED Spectra. April 2020.

[Bre87] Edwin J. Breneman. Corresponding chromaticities for different states of adaptation to complex visual fields. *Journal of the Optical Society of America A*, 4(6):1115, June 1987. doi:10.1364/JOSAA.4.001115.

[DFI+17] Scott Dyer, Alexander Forsythe, Jonathon Irons, Thomas Mansencal, and Miaoqi Zhu. RAW to ACES Utility Data. 2017.

[EF98] Fritz Ebner and Mark D. Fairchild. Finding constant hue surfaces in color space. In Giordano B. Beretta and Reiner Eschbach, editors, *Proc. SPIE 3300, Color Imaging: Device-Independent Color, Color Hardcopy, and Graphic Arts III, (2 January 1998)*, 107–117. January 1998. doi:10.1117/12.298269.

[HUniversityoKuopioa] Jouni Haanpalo and University of Kuopio. Munsell Colors Glossy (Spectrofotometer Measured). doi:10.5281/zenodo.3269916.

[HUniversityoKuopiob] Jouni Haanpalo and University of Kuopio. Paper Spectra. doi:10.5281/zenodo.3269922.

[HUniversityoKuopioc] Jouni Hiltunen and University of Kuopio. Lumber Spectra. doi:10.5281/zenodo.3269924.

[HB95] Po-Chieh Hung and Roy S. Berns. Determination of constant Hue Loci for a CRT gamut and their predictions using color appearance spaces. *Color Research & Application*, 20(5):285–295, October 1995. doi:10.1002/col.5080200506.

[JH19] Wenzel Jakob and Johannes Hanika. A Low-Dimensional Function Space for Efficient Spectral Upsampling. *Computer Graphics Forum*, 38(2):147–155, May 2019. doi:10.1111/cgf.13626.

[JLGS13] Jun Jiang, Dengyu Liu, Jinwei Gu, and Sabine Susstrunk. What is the space of spectral sensitivity functions for digital color cameras? In *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, 168–179. IEEE, January 2013. doi:10.1109/WACV.2013.6475015.

[KFE15] Andreas Karge, Jan Froehlich, and Bernhard Eberhardt. A Spectral Database of Commonly Used Cine Lighting. *Color and Imaging Conference*, October 2015.

[LF20] Anders Langlands and Luca Fascione. PhysLight: An End-to-End Pipeline for Scene-Referred Lighting. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Talks*, 1–2. Virtual Event USA, August 2020. ACM. doi:10.1145/3388767.3407368.

[LR97] M Ronnier Luo and Peter A. Rhodes. Using the LUTCHI Colour Appearance Data. https://web.archive.org/web/20040212195937/http://colour.derby.ac.uk:80/colour/info/lutchi/, 1997.

[LCR+91a] M. Ronnier Luo, Anthony A. Clarke, Peter A. Rhodes, André Schappo, Stephen A. R. Scrivener, and Chris J. Tait. Quantifying colour appearance. Part I. Lutchi

colour appearance data. *Color Research & Application*, 16(3):166–180, June 1991. doi:10.1002/col.5080160307.

[LCR+91b] M. Ronnier Luo, Anthony A. Clarke, Peter A. Rhodes, André Schappo, Stephen A.R. Scrivener, and Chris J. Tait. Quantifying colour appearance. Part II. Testing colour models performance using lutchi colour appearance data. *Color Research & Application*, 16(3):181–197, 1991. doi:10.1002/col.5080160308.

[LGR+93] M. Ronnier Luo, X. Wang Gao, Peter A. Rhodes, H. John Xin, Anthony A. Clarke, and Stephen A.R. Scrivener. Quantifying colour appearance. part III. Supplementary LUTCHI colour appearance data. *Color Research & Application*, 18(2):98–113, 1993. doi:10.1002/col.5080180207.

[LR99] M. Ronnier Luo and Peter A. Rhodes. Corresponding-colour datasets. *Color Research & Application*, 24(4):295–296, August 1999. doi:10.1002/(SICI)1520-6378(199908)24:4{$<$}295::AID-COL10{$>$}3.0.CO;2-K.

[MUniversityoKuopio] Elzbieta Marszalec and University of Kuopio. Agfa IT8.7/2 Set. doi:10.5281/zenodo.3269926.

[MMT76] John J. McCann, Suzanne P. McKee, and Thomas H Taylor. Quantitative studies in retinex theory a comparison between theoretical predictions and observer responses to the "color mondrian" experiments. *Vision Research*, 16(5):445–IN3, January 1976. doi:10.1016/0042-6989(76)90020-1.

[OUniversityoKuopio] Joni Orava and University of Kuopio. Munsell Colors Glossy (All) (Spectrofotometer Measured). doi:10.5281/zenodo.3269918.

[SUniversityoKuopio] Raimo Silvennoinen and University of Kuopio. Forest Colors. doi:10.5281/zenodo.3269920.

[SA23] Grigory Solomatov and Derya Akkaynak. Spectral sensitivity estimation without a camera. In *IEEE International Conference on Computational Photography (ICCP)*. July 2023.

[WTWetaDigital22] Erik Winquist, Kimball Thurston, and Weta Digital. Physlight - Camera Spectral Sensitivity Curves. 2022.

[ZKTI09] Hongxun Zhao, Rei Kawakami, Robby T Tan, and Katsushi Ikeuchi. Estimating basis functions for spectral sensitivity of digital cameras. 2009.

[HautaKasariUniversityoKuopioa] Markku Hauta-Kasari and University of Kuopio. Munsell Colors Matt (AOTF Measured). doi:10.5281/zenodo.3269914.

[HautaKasariUniversityoKuopiob] Markku Hauta-Kasari and University of Kuopio. Munsell Colors Matt (Spectrofotometer Measured). doi:10.5281/zenodo.3269912.

[Labsphere19] Labsphere. Labsphere SRS-99-020. 2019. doi:10.5281/zenodo.3245875.

[OpenpyxlDevelopers19] Openpyxl Developers. Openpyxl. 2019.

[XRite16] X-Rite. New color specifications for ColorChecker SG and Classic Charts. http://xritephoto.com/ph_product_overview.aspx?ID=938&Action=Support&SupportID=5884#, 2016.

INDEX

Symbols

A

B

C

D

F

H

I

J

L

M

P

R

S

T

U