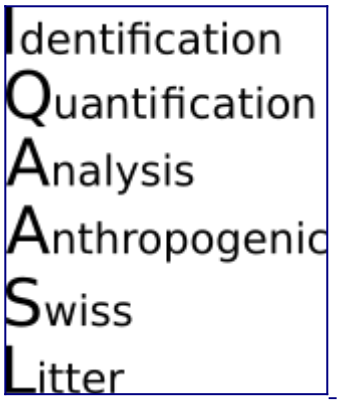
Toggle navigation sidebar
Toggle in-page Table of Contents



- Identification, quantification and analysis of observable anthropogenic litter along Swiss lake systems

Zusammenfassung

- 1. Seen und Fliessgewässer
- 2. Alpen und der Jura
- 3. Aare
- 4. Bielersee

Powered by Jupyter Book

- Binder

- repository
- open issue

- .ipynb
- .pdf

Contents

- 3.1. Erhebungsorte und Landnutzungsprofile
  - 3.1.1. Landnutzungsprofil der Erhebungsorte
  - 3.1.2. Kumulative Gesamtmengen nach Gewässer
  - 3.1.3. Verteilung der Erhebungsergebnisse
  - 3.1.4. Zusammengefasste Daten und Materialarten
- 3.2. Die am häufigsten gefundenen Objekte
  - 3.2.1. Die am häufigsten gefundenen Objekte nach Gewässer
  - 3.2.2. Die am häufigsten gefundenen Objekte im monatlichen Durchschnitt
- 3.3. Erhebungsergebnisse und Landnutzung
- 3.4. Verwendungszweck der gefundenen Objekte
- 3.5. Fliessgewässer
  - 3.5.1. Die an Fliessgewässern am häufigsten gefundenen Objekte
- 3.6. Anhang
  - 3.6.1. Schaumstoffe und Kunststoffe nach Grösse
  - 3.6.2. Die Erhebungsorte

- 3.6.3. Inventar der Objekte

Aare

Contents

- 3.1. Erhebungsorte und Landnutzungsprofile
  - 3.1.1. Landnutzungsprofil der Erhebungsorte
  - 3.1.2. Kumulative Gesamtmengen nach Gewässer
  - 3.1.3. Verteilung der Erhebungsergebnisse
  - 3.1.4. Zusammengefasste Daten und Materialarten
- 3.2. Die am häufigsten gefundenen Objekte
  - 3.2.1. Die am häufigsten gefundenen Objekte nach Gewässer
  - 3.2.2. Die am häufigsten gefundenen Objekte im monatlichen Durchschnitt
- 3.3. Erhebungsergebnisse und Landnutzung
- 3.4. Verwendungszweck der gefundenen Objekte
- 3.5. Fliessgewässer
  - 3.5.1. Die an Fliessgewässern am häufigsten gefundenen Objekte
- 3.6. Anhang
  - 3.6.1. Schaumstoffe und Kunststoffe nach Grösse
  - 3.6.2. Die Erhebungsorte
  - 3.6.3. Inventar der Objekte


```
# -*- coding: utf-8 -*-
# This is a report using the data from IQAASL.
# IQAASL was a project funded by the Swiss Confederation
# It produces a summary of litter survey results for a defined region.
# These charts serve as the models for the development of plagespropres.ch
# The data is gathered by volunteers.
# Please remember all copyrights apply, please give credit when applicable
# The repo is maintained by the community effective January 01, 2022
# There is ample opportunity to contribute, learn and teach
# contact dev@hammerdirt.ch

# Dies ist ein Bericht, der die Daten von IQAASL verwendet.
# IQAASL war ein von der Schweizerischen Eidgenossenschaft finanziertes Projekt.
# Es erstellt eine Zusammenfassung der Ergebnisse der Littering-Umfrage für eine
bestimmte Region.
# Diese Grafiken dienten als Vorlage für die Entwicklung von plagespropres.ch.
# Die Daten werden von Freiwilligen gesammelt.
# Bitte denken Sie daran, dass alle Copyrights gelten, bitte geben Sie den Namen an,
wenn zutreffend.
# Das Repo wird ab dem 01. Januar 2022 von der Community gepflegt.
# Es gibt reichlich Gelegenheit, etwas beizutragen, zu lernen und zu lehren.
# Kontakt dev@hammerdirt.ch

# Il s'agit d'un rapport utilisant les données de IQAASL.
# IQAASL était un projet financé par la Confédération suisse.
# Il produit un résumé des résultats de l'enquête sur les déchets sauvages pour une
région définie.
# Ces tableaux ont servi de modèles pour le développement de plagespropres.ch
```

```
# Les données sont recueillies par des bénévoles.
# N'oubliez pas que tous les droits d'auteur s'appliquent, veuillez indiquer le
crédit lorsque cela est possible.
# Le dépôt est maintenu par la communauté à partir du 1er janvier 2022.
# Il y a de nombreuses possibilités de contribuer, d'apprendre et d'enseigner.
# contact dev@hammerdirt.ch

# sys, file and nav packages:
import datetime as dt
from datetime import date, datetime, time
from babel.dates import format_date, format_datetime, format_time, get_month_names
import locale

# math packages:
import pandas as pd
import numpy as np
from scipy import stats
from statsmodels.distributions.empirical_distribution import ECDF

# charting:
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from matplotlib import ticker
from matplotlib import colors
from matplotlib.colors import LinearSegmentedColormap
from matplotlib.gridspec import GridSpec
import seaborn as sns

# home brew utitilties
import resources.chart_kwargs as ck
import resources.sr_ut as sut

# images and display
from IPython.display import Markdown as md
from myst_nb import glue

# set the locale to the language desired
date_lang =  'de_DE.utf8'
language = "DE"
locale.setlocale(locale.LC_ALL, date_lang)

# the date is in iso standard:
date_format = "%Y-%m-%d"

# it gets changed to german format
german_date_format = "%d.%m.%Y"

# set some parameters:
start_date = "2020-03-01"
end_date ="2021-05-31"
start_end = [start_date, end_date]

# The fail rate is 50%, that is objects that
# are found in more than 50% of the samples
# are considered common or among the most common
```

```
a_fail_rate = 50

# the units of the repor
unit_label = "p/100 m"

# charting and colors
sns.set_style("whitegrid")
table_row = "saddlebrown"

# colors for gradients
cmap2 = ck.cmap2
colors_palette = ck.colors_palette

## !! Begin Note book variables !!

# Changing these variables produces different reports
# Call the map image for the area of interest
bassin_map = "resources/maps/survey_areas/aare_scaled.jpeg"

# the label for the aggregation of all data in the region
top = "Alle Erhebungsgebiete"

# define the feature level and components
# the feature of interest is the Aare (aare) at the river basin (river_bassin) level.
# the label for charting is called 'name'
this_feature = {'slug':'aare', 'name':"Erhebungsgebiet Aare", 'level':'river_bassin'}

# these are the smallest aggregated components
# choices are water_name_slug=lake or river, city or location at the scale of a river
bassin
# water body or lake maybe the most appropriate
this_level = 'water_name_slug'

# identify the lakes of interest for the survey area
lakes_of_interest = ["neuenburgersee", "thunersee", "bielersee", "brienzersee"]

# !! End note book variables !!

# column names for the dimensions table output
dims_table_columns={
    "samples":"Erhebungen",
    "quantity":"Objekte",
    "total_w":"Gesamt-kg",
    "mac_plast_w":"kg Plastik",
    "area":"m²",
    "length":"Meter"
}

# !! explanatory variabales, code definitions, language specific

# Survey location details (GPS, city, land use)
dfBeaches = pd.read_csv("resources/beaches_with_land_use_rates.csv")

# Object code definitions, labels and material type
dfCodes = pd.read_csv("resources/codes_with_group_names_2015.csv")
```

```
# Survey dimensions and weights
dfDims = pd.read_csv("resources/corrected_dims.csv")

# set the index of the beach data to location slug
dfBeaches.set_index("slug", inplace=True)

# set the index of code data to code
dfCodes.set_index("code", inplace=True)

# the surveyor designated the object as aluminum instead of metal
dfCodes.loc["G708", "material"] = "Metal"

# language specific
# importing german code descriptions
de_codes = pd.read_csv("resources/codes_german_Version_1.csv")
de_codes.set_index("code", inplace=True)

# use the german translation of the code descriptions
for x in dfCodes.index:
    dfCodes.loc[x, "description"] = de_codes.loc[x, "german"]

# there are long code descriptions that may need to be shortened for display
codes_to_change = [
    ["G704", "description", "Seilbahnbürste"],
    ["Gfrags", "description", "Fragmentierte Kunststoffstücke"],
    ["G30", "description", "Snack-Verpackungen"],
    ["G124", "description", "Kunststoff-oder Schaumstoffprodukte"],
    ["G87", "description", "Abdeckklebeband / Verpackungsklebeband"],
    ["G3","description","Einkaufstaschen, Shoppingtaschen"],
    ["G33", "description", "Einwegartikel; Tassen/Becher & Deckel"],
    ["G31", "description", "Schleckstengel, Stengel von Lutscher"],
    ["G211", "description", "Sonstiges medizinisches Material"],
    ["G904", "description", "Feuerwerkskörper; Raketenkappen"],
    ["G940", "description", "Schaumstoff EVA (flexibler Kunststoff)"],
    ["G178", "description", "Kronkorken, Lasche von Dose/Ausfreisslachen"],
    ["G74", "description", "Schaumstoffverpackungen/Isolierung"],
    ["G941", "description", "Verpackungsfolien, nicht für Lebensmittel"]
]

# apply changes
for x in codes_to_change:
    dfCodes = sut.shorten_the_value(x, dfCodes)

# translate the material column to german
dfCodes["material"] = dfCodes.material.map(lambda x: sut.mat_ge[x])

# make a map to the code descriptions
code_description_map = dfCodes.description

# make a map to the code materials
code_material_map = dfCodes.material
```

3. Aare#

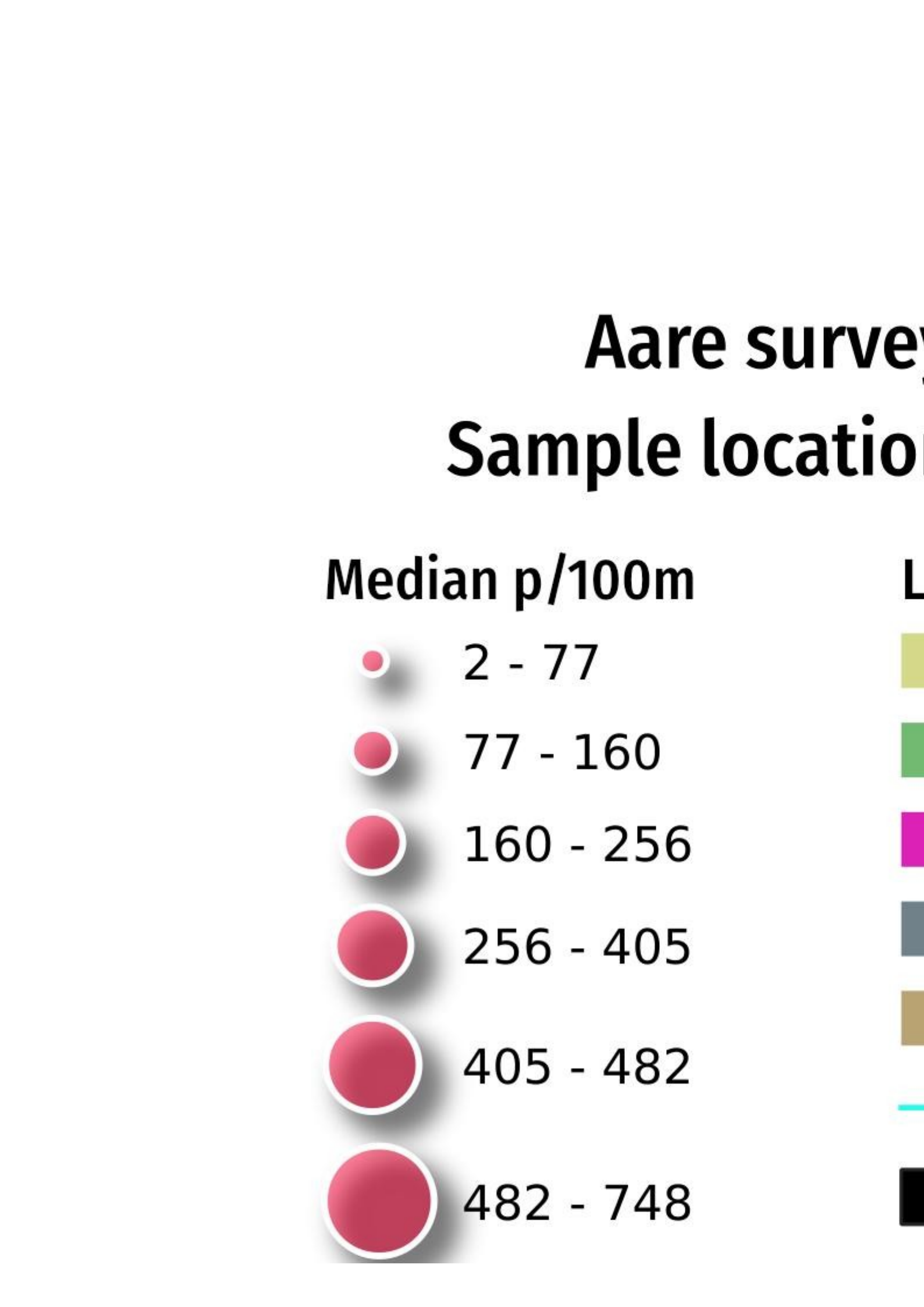

Aare survey
Sample locatio
Median p/100m
2 - 77
77 - 160
160 - 256
256 - 405
405 - 482
482 - 748
L

Abb. 3.1 #

Abbildung 3.1: Karte des Erhebungsgebiets März 2020 bis Mai 2021. Der Durchmesser der Punktsymbole entspricht dem Median der Abfallobjekte pro 100 Meter (p/100 m) am jeweiligen Erhebungsort.

3.1. Erhebungsorte und Landnutzungsprofile#

```
def thereIsData(data=False, atype=(pd.DataFrame, )):
    # checkes that the provided data is a certain type
    if isinstance(data, atype):
        return data
    else:
        raise TypeError(f"There is no data or it is not the right type: 
is_instance({data}, {atype}).")

def loadData(filename):
    # loads data from a .csv
    filename = thereIsData(data=filename, atype=(str,))

    try:
        a = pd.read_csv(filename)
    except OSError:
        print("The file could not be read, is this the right file extension?")
        raise
    return a

def changeColumnNames(data, columns={}):
    # changes the column names of a data frame
    data = thereIsData(data=data, atype=(pd.DataFrame, ))
    cols = thereIsData(data=columns, atype=(dict, ))

    try:
        a = data.rename(columns=columns)
    except ValueError:
        print("The columns did not go with the data")
        raise
    return a

def makeEventIdColumn(data, feature_level, these_features=[], index_name="loc_date", 
index_prefix="location", index_suffix="date", **kwargs):
    # Combines the location and date column into one str: "slug-date"
    # makes it possible ot group records by event
    # converts string dates to timestamps and localizes to UTC
    data = thereIsData(data=data, atype=(pd.DataFrame, ))
    feature_level = thereIsData(data=feature_level, atype=(str, ))
    these_features = thereIsData(data=these_features, atype=(list, np.ndarray))

    try:
        sliced_data = data[data[feature_level].isin(these_features)].copy()
        sliced_data[index_name] = list(zip(sliced_data[index_prefix].values, 
sliced_data[index_suffix].values))
        sliced_data["date"] = pd.to_datetime(sliced_data["date"], 
format=date_format).dt.tz_localize('UTC')
    except RuntimeError:
```

```
        print("The pandas implementation did not function")
        raise

    return sliced_data

def featureData(filename, feature_level, these_features=[], columns=False,
language="EN"):
    # makes the feature data to be explored
    data = loadData(filename)

    if columns != False:
        data = changeColumnNames(data, columns=columns)
    if language == "DE":
        data["groupname"] = data["groupname"].map(lambda x: sut.group_names_de[x])
    a = makeEventIdColumn(data, feature_level, these_features=these_features)

    return a, data

def convert_case(str_camelcase):
    # This function takes in a string in camelCase and converts it to snake_case
    str_camelcase = thereIsData(data=str_camelcase, atype=(str, ))
    str_snake_case = ""
    for ele in list(str_camelcase):
        if ele.islower():
            str_snake_case = str_snake_case + ele
        else:
            str_snake_case = str_snake_case + "_" + ele.lower()
    return str_snake_case


def checkInitiateAttribute(data=False, check=False, atype=(list, np.ndarray),
a_method=None, **kwargs):
    # check the data type of the requested element against the required data type
    # if check the data is returned, else <a_method> will be applied to <data>
    # and checked again.
    if isinstance(check, atype):
        return check
    else:
        try:
            new_data = a_method(data)
            check_again = checkInitiateAttribute(check=new_data, data=new_data,
atype=atype, a_method=a_method, **kwargs)
            return check_again

        except ValueError:
            print("neither the data nor the method worked")
            raise

def uniqueValues(data):
    # method to pass pd.series.unique as a variable
    return data.location.unique()

def dateToYearAndMonth(python_date_object, fmat='wide', lang=""):
    a_date = thereIsData(data=python_date_object, atype=(datetime, ))
    amonth = a_date.month
```

```
    a_year = a_date.year
    amonth_foreign = get_month_names(fmat, locale=lang)[amonth]

    return f'{amonth_foreign} {a_year}'

def thousandsSeparator(aninteger, lang):

    astring = "{:,}".format(aninteger)

    if lang == "DE":
        astring = astring.replace(",", " ")

    return astring


class Beaches:
    """The dimendsional and geo data for each survey location"""
    df_beaches = dfBeaches

class AdministrativeSummary(Beaches):

    col_nunique_qty=["location", "loc_date", "city"]
    col_sum_qty = ["quantity"]
    col_population = ["city", "population"]
    col_sum_pop = ["population"]
    col_nunique_city = ["city"]
    locations_of_interest = None
    lakes_of_interest = None
    rivers_of_interest = None

    def __init__(self, data = None, label=None, **kwargs):

        self.data = data
        self.label = label
        super().__init__()

    def locationsOfInterest(self, **kwargs):
        data = thereIsData(self.data, (pd.DataFrame))
        locations = checkInitiateAttribute(check=self.locations_of_interest,
data=data, atype=(list, np.ndarray), a_method=uniqueValues, **kwargs)

        self.locations_of_interest = locations

    def resultsObject(self, col_nunique=None, col_sum=None, **kwargs):
        data = thereIsData(self.data, (pd.DataFrame))

        if not col_nunique:
            col_nunique=self.col_nunique_qty
        if not col_sum:
            col_sum=self.col_sum_qty

        t = sut.make_table_values(data, col_nunique=col_nunique, col_sum=col_sum)

        return t
```

```
    def populationKeys(self):
        data = thereIsData(self.data, (pd.DataFrame))
        locs = checkInitiateAttribute(check=self.locations_of_interest, data=data,
atype=(list, np.ndarray), a_method=uniqueValues)

        try:
            popmap = self.df_beaches.loc[locs][self.col_population].drop_duplicates()
        except TypeError as e:
            print("that did not work")

        return popmap

    def lakesOfInterest(self):
        data = thereIsData(self.data, (pd.DataFrame))
        locs = checkInitiateAttribute(check=self.locations_of_interest, data=data,
atype=(list, np.ndarray), a_method=uniqueValues)

        if not isinstance(self.lakes_of_interest, list):
            mask = (self.df_beaches.index.isin(locs))&(self.df_beaches.water == "l")
            d = self.df_beaches.loc[mask]["water_name"].unique()
            self.lakes_of_interest = d
            return d
        else:
            return self.lakes_of_interest

    def riversOfInterest(self):
        data = thereIsData(self.data, (pd.DataFrame))
        locs = checkInitiateAttribute(check=self.locations_of_interest, data=data,
atype=(list, np.ndarray), a_method=uniqueValues)

        if not isinstance(self.rivers_of_interest, list):
            mask = (self.df_beaches.index.isin(locs))&(self.df_beaches.water == "r")
            d = self.df_beaches.loc[mask]["water_name"].unique()
            self.rivers_of_interest = d
            return d
        else:
            return self.rivers_of_interest

    def summaryObject(self, **kwargs):
        t = self.resultsObject()
        pop_values = sut.make_table_values(self.populationKeys(),
col_nunique=self.col_nunique_city, col_sum=self.col_sum_pop)
        self.locationsOfInterest()
        t["locations_of_interest"] = self.locations_of_interest

        t.update(pop_values)

        return t

# the survey data
filename = "resources/checked_sdata_eos_2020_21.csv"
columns={"% to agg":"% agg", "% to recreation": "% recreation", "% to woods":"%
woods", "% to buildings":"% buildings", "p/100m":"p/100 m"}
```

```
fd, a_data = featureData(filename, this_feature["level"],
these_features=[this_feature["slug"]], columns=columns, language="DE")


# collects the summarized values for all the data in the region
summary_data = AdministrativeSummary(data=fd, label=this_feature["name"])

# collects the names of the survey location
a_summary = summary_data.summaryObject()

rivers = summary_data.riversOfInterest()
lakes = summary_data.lakesOfInterest()

# string objects for display
obj_string = thousandsSeparator(a_summary["quantity"], language)
surv_string = "{:,}".format(a_summary["loc_date"])
pop_string = thousandsSeparator(int(a_summary["population"]), language)

# make strings
date_quantity_context = F"Im Zeitraum von
{dateToYearAndMonth(datetime.strptime(start_date, date_format), lang=date_lang)}  bis
{dateToYearAndMonth(datetime.strptime(end_date, date_format), lang= date_lang)}
wurden im Rahmen von {surv_string} Datenerhebungen insgesamt {obj_string } Objekte
entfernt und identifiziert."
geo_context = F"Die Ergebnisse des {this_feature['name']} umfassen
{a_summary['location']} Orte, {a_summary['city']} Gemeinden und eine
Gesamtbevölkerung von etwa {pop_string} Einwohnenden."

# lists of landmarks of interest
munis_joined = ", ".join(sorted(summary_data.populationKeys()["city"]))
lakes_joined = ", ".join(sorted(lakes))
rivers_joined = ", ".join(sorted(rivers))

# put that all together:
lake_string = F"""
{date_quantity_context} {geo_context }

*Seen:*\n\n>{lakes_joined}

*Fliessgewässer:*\n\n>{rivers_joined}

*Gemeinden:*\n\n>{munis_joined}
"""
md(lake_string)
```

Im Zeitraum von März 2020 bis Mai 2021 wurden im Rahmen von 140 Datenerhebungen insgesamt 13 847 Objekte entfernt und identifiziert. Die Ergebnisse des Erhebungsgebiet Aare umfassen 51 Orte, 35 Gemeinden und eine Gesamtbevölkerung von etwa 493 799 Einwohnenden.

Seen:

> Bielersee, Brienzersee, Neuenburgersee, Thunersee

Fliessgewässer:

Aare, Aare|Nidau-Büren-Kanal, Emme, La Thièle, Schüss

Gemeinden:

Aarau, Beatenberg, Bern, Biel/Bienne, Boudry, Brienz (BE), Brugg, Brügg, Burgdorf, Bönigen, Cheyres-Châbles, Cudrefin, Erlach, Estavayer, Gals, Gebenstorf, Grandson, Hauterive (NE), Kallnach, Köniz, Le Landeron, Ligerz, Luterbach, Lüscherz, Neuchâtel, Nidau, Port, Rubigen, Solothurn, Spiez, Thun, Unterseen, Vinelz, Walperswil, Yverdon-les-Bains

3.1.1. Landnutzungsprofil der Erhebungsorte#

Das Landnutzungsprofil zeigt, welche Nutzungen innerhalb eines Radius von 1500 m um jeden Erhebungsort dominieren. Flächen werden einer von den folgenden vier Kategorien zugewiesen:

- Fläche, die von Gebäuden eingenommen wird in %
- Fläche, die dem Wald vorbehalten ist in %
- Fläche, die für Aktivitäten im Freien genutzt wird in %
- Fläche, die von der Landwirtschaft genutzt wird in %

Strassen (inkl. Wege) werden als Gesamtzahl der Strassenkilometer innerhalb eines Radius von 1500 m angegeben.

Es wird zudem angegeben, wie viele Flüsse innerhalb eines Radius von 1500 m um den Erhebungsort herum in das Gewässer münden.

Das Verhältnis der gefundenen Abfallobjekte unterscheidet sich je nach Landnutzungsprofil. Das Verhältnis gibt daher einen Hinweis auf die ökologischen und wirtschaftlichen Bedingungen um den Erhebungsort.

Für weitere Informationen siehe *17 Landnutzungsprofil*

```
# land use characteristics
# the ratio of samples with respect to the different land use characteristics for each survey area
# the data to use is the unique combinations of loc_date and the land_use charcteristics of each location
# land use explanatory variables are in the :
# land_use_columns = ["% buildings", "% recreation", "% agg", "% woods", "streets km", "intersects"]

def empiricalCDF(anarray):
    data = thereIsData(anarray, (list, np.ndarray))
    y = np.arange(1, len(data)+1)/float(len(data))
    x = sorted(data)

    return x, y

def ecdfOfaColumn(data, column=""):
    data = thereIsData(data, (pd.DataFrame,))
    col = thereIsData(column, (list, np.ndarray))

    anarray=data[col].values
```

```
    x,y = empiricalCDF(anarray)

    return {"column":col, "x":x, "y":y}


class LandUseProfile:

    def __init__(self, data=None, index_column="loc_date",
aggregation_level=this_feature["level"], feature_of_interest=this_feature["slug"],
land_use_columns = ["% buildings", "% recreation", "% agg", "% woods", "streets km",
"intersects"],  **kwargs):
        self.data = data
        self.land_use_columns = land_use_columns
        self.aggregation_level = aggregation_level
        self.index_column= index_column
        self.feature_of_interest = feature_of_interest
        super().__init__()

    def byIndexColumn(self):
        data = thereIsData(data=self.data, atype=(pd.DataFrame, ))
        columns = [self.index_column, self.aggregation_level, *self.land_use_columns]
        d = data[columns].drop_duplicates()

        return d

    def featureOfInterest(self):
        data = thereIsData(data=self.data, atype=(pd.DataFrame, ))
        d_indexed = self.byIndexColumn()
        d =d_indexed[d_indexed[self.aggregation_level] == self.feature_of_interest]

        return d

land_use_columns = ["% buildings", "% recreation", "% agg", "% woods", "streets km",
"intersects"]
project_profile = LandUseProfile(data=a_data).byIndexColumn()
feature_profile = LandUseProfile(data=fd).featureOfInterest()

fig, axs = plt.subplots(2, 3, figsize=(9,8), sharey="row")
from matplotlib.ticker import MultipleLocator
for i, n in enumerate(land_use_columns):
    r = i%2
    c = i%3
    ax=axs[r,c]

    # the value of landuse feature n for the survey area:
    data=feature_profile[n].values
    xs, ys = empiricalCDF(data)
    sns.lineplot(x=xs, y=ys, ax=ax, label=summary_data.label)

    # the value of the land use feature n for all the data
    testx, testy = empiricalCDF(project_profile[n].values)
    sns.lineplot(x=testx, y=testy, ax=ax, label=top, color="magenta")
```

```
    # get the median from the data
    the_median = np.median(data)

    # plot the median and drop horzontal and vertical lines
    ax.scatter([the_median], 0.5, color="red",s=50, linewidth=2, zorder=100,
label="Median")
    ax.vlines(x=the_median, ymin=0, ymax=0.5, color="red", linewidth=2)
    ax.hlines(xmax=the_median, xmin=0, y=0.5, color="red", linewidth=2)

    if i <= 3:
        if c == 0:
            ax.set_ylabel("Ratio of samples", **ck.xlab_k)
            ax.yaxis.set_major_locator(MultipleLocator(.1))
        ax.xaxis.set_major_formatter(ticker.PercentFormatter(1.0, 0, "%"))
    else:
        pass

    handles, labels = ax.get_legend_handles_labels()
    ax.get_legend().remove()
    ax.set_xlabel(list(sut.luse_ge.values())[i], **ck.xlab_k)

plt.tight_layout()
plt.subplots_adjust(top=.9, hspace=.3)
plt.suptitle("Landnutzung im Umkries von 1 500 m um den Erhebungsort", ha="center",
y=1, fontsize=16)
fig.legend(handles, labels, bbox_to_anchor=(.5,.94), loc="center", ncol=3)

glue("aare_survey_area_landuse", fig, display=False)

plt.close()
```

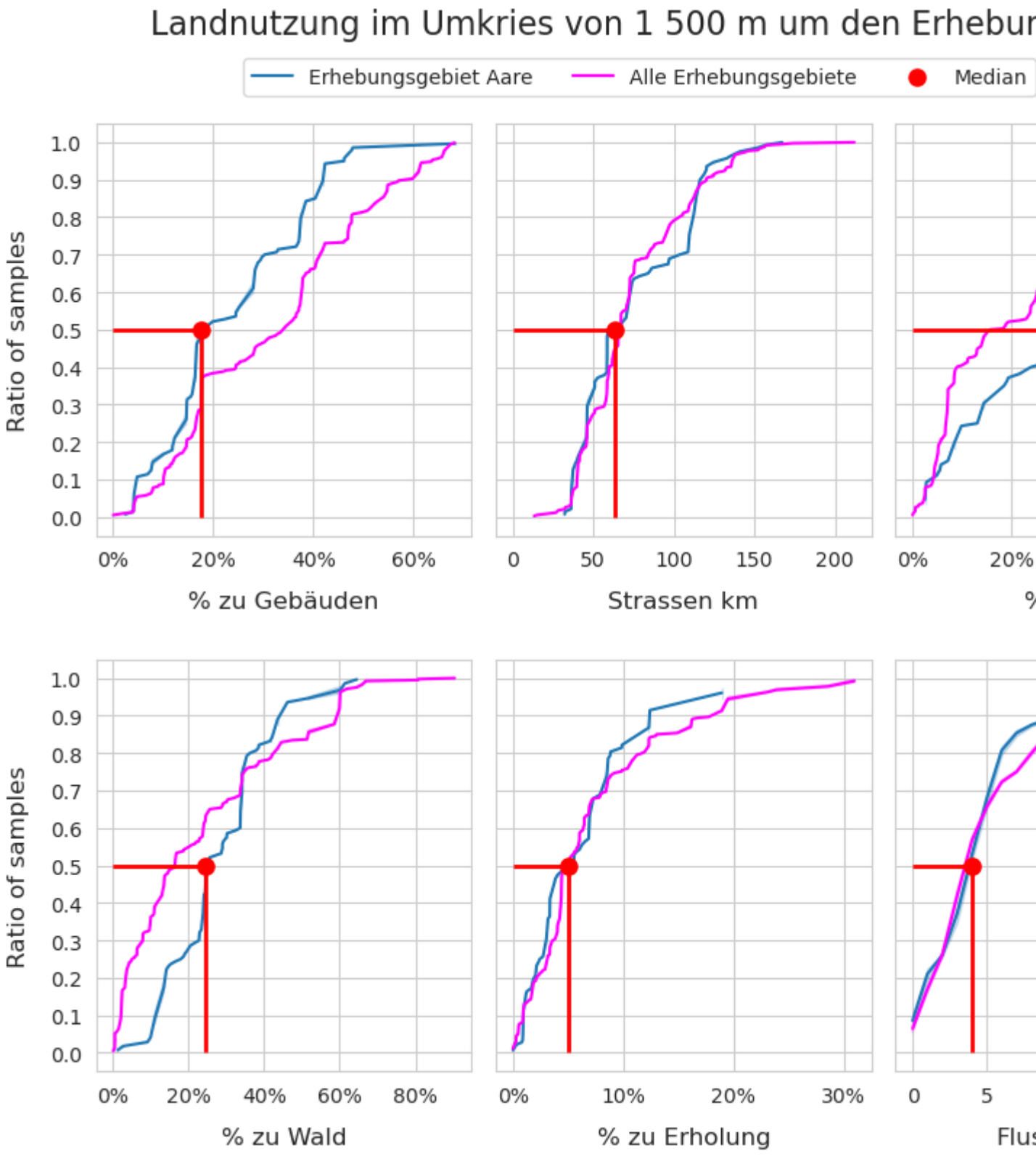


Abb. 3.2 #

Abbildung 3.2: Landnutzungsprofil der Erhebungsorte. Verteilung der Erhebungen in Bezug auf die Landnutzung. LWS = Landwirtschaft

3.1.2. Kumulative Gesamtmengen nach Gewässer#

```
# aggregate the dimensional data
agg_dims = {"total_w":"sum", "mac_plast_w":"sum", "area":"sum", "length":"sum"}

dims_parameters = dict(this_level=this_level,
```

```
                        locations=fd.location.unique(),
                        start_end=start_end,
                        agg_dims=agg_dims)

dims_table = sut.gather_dimensional_data(dfDims, **dims_parameters)

# map the qauntity to the dimensional data
q_map = fd.groupby(this_level).quantity.sum()

# collect the number of samples from the survey total data:
for name in dims_table.index:
    dims_table.loc[name, "samples"] = fd[fd[this_level] == name].loc_date.nunique()
    dims_table.loc[name, "quantity"] = q_map[name]

# map the proper name of the lake to the slug value
wname_wname =
dfBeaches[["water_name_slug","water_name"]].reset_index(drop=True).drop_duplicates().
set_index("water_name_slug")
comp_labels = {x:wname_wname.loc[x][0] for x in fd[this_level].unique()}

# add the proper names for display
dims_table["water_feature"] = dims_table.index.map(lambda x: comp_labels[x])
dims_table.set_index("water_feature", inplace=True)

# get the sum of all survey areas
dims_table.loc[this_feature["name"]]= dims_table.sum(numeric_only=True, axis=0)

# for display
dims_table.sort_values(by=["quantity"], ascending=False, inplace=True)
dims_table.rename(columns=dims_table_columns, inplace=True)

# format kilos and text strings
dims_table["kg Plastik"] = dims_table["kg Plastik"]/1000
dims_table[["m²", "Meter", "Erhebungen", "Objekte"]] = dims_table[["m²", "Meter",
"Erhebungen", "Objekte"]].applymap(lambda x: thousandsSeparator(int(x), language))
dims_table[["kg Plastik", "Gesamt-kg"]] = dims_table[["kg Plastik", "Gesamt-
kg"]].applymap(lambda x: "{:.2f}".format(x))

# make table
data = dims_table.reset_index()
colLabels = data.columns

fig, ax = plt.subplots(figsize=(len(colLabels)*1.8,len(data)*.7))
sut.hide_spines_ticks_grids(ax)

table_one = sut.make_a_table(ax, data.values, colLabels=colLabels, colWidths=[.28,
*[.12]*6], a_color=table_row)
table_one.get_celld()[(0,0)].get_text().set_text(" ")

plt.tight_layout()
glue("aare_survey_area_dimensional_summary", fig, display=False)
plt.close()
```

	Gesamt-kg	kg Plastik	r
Erhebungsgebiet Aare	71.98	31.45	37
Neuenburgersee	17.43	9.45	15
Bielersee	13.11	6.67	6
Thunersee	19.98	5.89	9
Brienzersee	2.27	1.97	1
Aare	13.78	5.86	2
Aare|Nidau-Büren-Kanal	2.15	0.26	9
Schüss	0.36	0.21	3
Emme	2.91	1.13	4
La Thièle	0.00	0.00	1

Abb. 3.3 #

Abbildung 3.3: Die kumulierten Gewichte und Merkmale für das Erhebungsgebiet Aare nach Gewässern.

3.1.3. Verteilung der Erhebungsergebnisse#

```
# the surveys to chart
# common aggregations: sum the pcs/m and quantity per event and location
agg_pcs_quantity = {unit_label:"sum", "quantity":"sum"}

# daily survey totals
dt_all = fd.groupby(["loc_date","location",this_level, "city","date"],
as_index=False).agg(agg_pcs_quantity)
fd_dindex = dt_all.copy()

# Daily totals from all other locations
ots = dict(level_to_exclude=this_feature["level"],
```

```
components_to_exclude=fd[this_feature["level"]].unique())
dts_date = sut.the_other_surveys(a_data, **ots)
dts_date = dts_date.groupby(["loc_date","date"], as_index=False)[unit_label].sum()
dts_date["date"] = pd.to_datetime(dts_date["date"]).dt.tz_localize('UTC')

# scale the chart as needed to accomodate for extreme values
y_lim = 95
y_limit = np.percentile(dts_date[unit_label], y_lim)

# label for the chart that alerts to the scale
not_included = F"Werte grösser als {thousandsSeparator(int(round(y_limit,0)),
language)} {unit_label} werden nicht gezeigt."

chart_notes = f"""
*__Links:__ {this_feature["name"]}, {dateToYearAndMonth(datetime.strptime(start_date,
date_format), lang=date_lang)} bis {dateToYearAndMonth(datetime.strptime(end_date,
date_format), lang=date_lang)}, n = {a_summary["loc_date"]}. {not_included}
__Rechts:__ empirische Verteilungsfunktion im {this_feature["name"]}.*
"""

# months locator, can be confusing
# https://matplotlib.org/stable/api/dates_api.html
months = mdates.MonthLocator(interval=1)
months_fmt = mdates.DateFormatter("%b")
days = mdates.DayLocator(interval=7)

# get the monthly or quarterly results for the feature
rsmp = fd_dindex.set_index("date")
resample_plot, rate = sut.quarterly_or_monthly_values(rsmp, this_feature["name"],
vals=unit_label, quarterly=["ticino"])

fig, axs = plt.subplots(1,2, figsize=(10,5))

ax = axs[0]

# feature surveys
sns.scatterplot(data=dts_date, x=dts_date["date"], y=unit_label, label=top,
color="black", alpha=0.4,  ax=ax)
# all other surveys
sns.scatterplot(data=fd_dindex, x=fd_dindex["date"], y=unit_label,
label=this_feature["name"], color="red", s=34, ec="white", ax=ax)
# monthly or quaterly plot
sns.lineplot(data=resample_plot, x=resample_plot.index, y=resample_plot,
label=F"{this_feature['name']}: monatlicher Medianwert", color="magenta", ax=ax)

ax.set_ylim(0,y_limit )
ax.set_ylabel(unit_label, **ck.xlab_k14)

ax.set_xlabel("")
ax.xaxis.set_minor_locator(days)
ax.xaxis.set_major_formatter(months_fmt)
ax.legend()

# the cumlative distributions:
axtwo = axs[1]
```

```
# the feature of interest
feature_ecd = ECDF(dt_all[unit_label].values)
sns.lineplot(x=feature_ecd.x, y=feature_ecd.y, color="darkblue", ax=axtwo,
label=this_feature["name"])

# the other features
other_features = ECDF(dts_date[unit_label].values)
sns.lineplot(x=other_features.x, y=other_features.y, color="magenta", label=top,
linewidth=1, ax=axtwo)

axtwo.set_xlabel(unit_label, **ck.xlab_k14)
axtwo.set_ylabel("Verhältnis der Erhebungen", **ck.xlab_k14)

plt.tight_layout()

glue('aare_survey_area_sample_totals', fig, display=False)
plt.close()
```

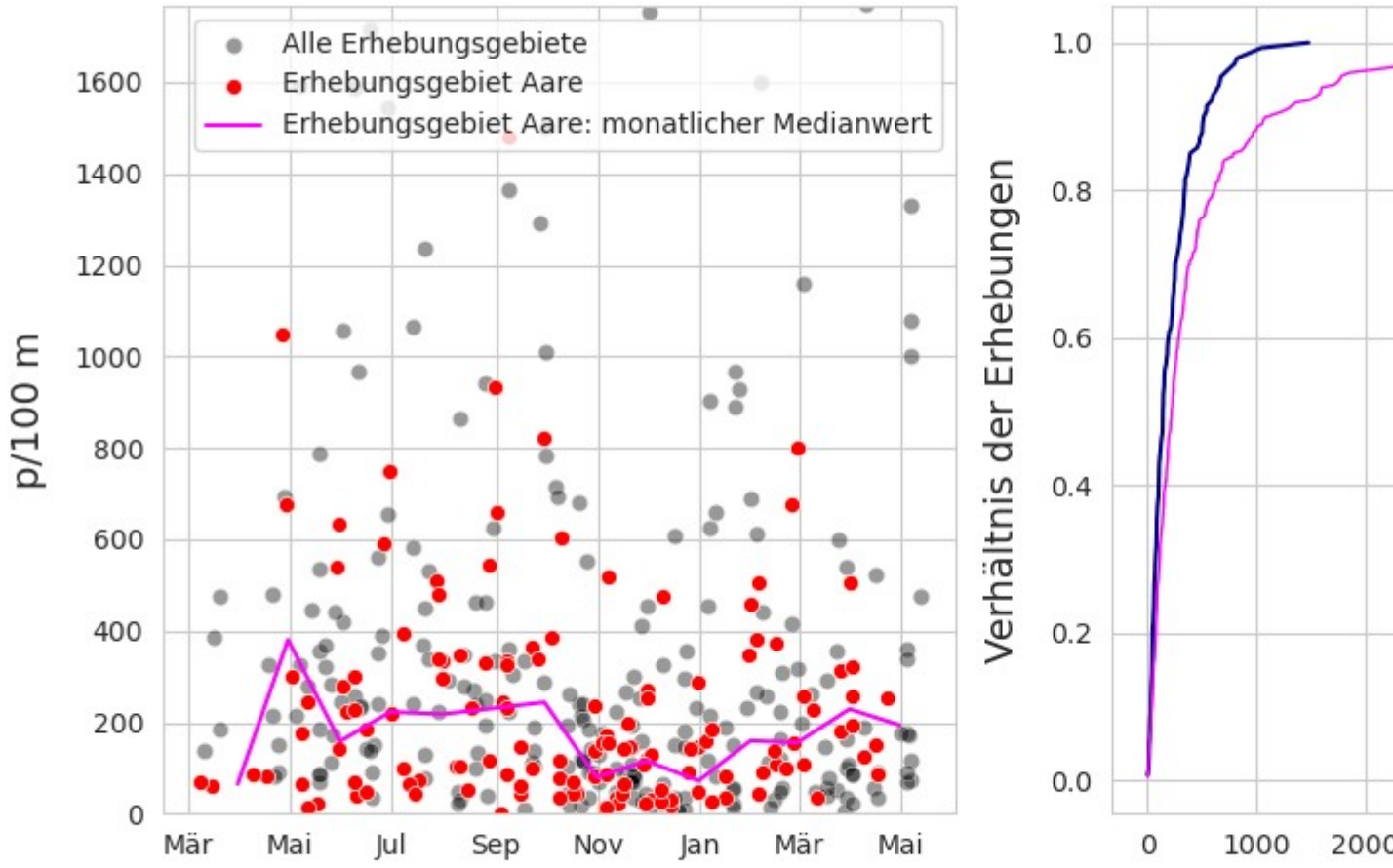


Abb. 3.4 #

Abbildung 3.4: **Links:** Erhebungsgebiet Aare, März 2020 bis Mai 2021, n = 140. Werte grösser als 1 766 p/100 m werden nicht gezeigt. **Rechts:** empirische Verteilungsfunktion im Erhebungsgebiet Aare.

3.1.4. Zusammengefasste Daten und Materialarten#

```
# figure caption
summary_of_survey_totals = f"""
```

```
*__Links:__ Zusammenfassung der Daten aller Erhebungen {this_feature["name"]}.
__Rechts:__ Gefundene Materialarten im {this_feature["name"]} in Stückzahlen und als
prozentuale Anteile (stückzahlbezogen).*
"""

# get the basic statistics from pd.describe
cs = dt_all[unit_label].describe().round(2)

# change the names
csx = sut.change_series_index_labels(cs, sut.create_summary_table_index(unit_label,
lang="DE"))

combined_summary =[(x, thousandsSeparator(int(csx[x]), language)) for x in csx.index]

agg_pcs_median = {unit_label:"median", "quantity":"sum"}

# cumulative statistics for each code
code_totals = sut.the_aggregated_object_values(
    fd,
    agg=agg_pcs_median,
    description_map=code_description_map,
    material_map=code_material_map
)

# the materials table
fd_mat_totals = sut.the_ratio_object_to_total(code_totals)
fd_mat_totals = sut.fmt_pct_of_total(fd_mat_totals)

# applly new column names for printing
cols_to_use = {"material":"Material","quantity":"Gesamt", "% of total":"% Gesamt"}
fd_mat_t = fd_mat_totals[cols_to_use.keys()].values
fd_mat_t = [(x[0], thousandsSeparator(int(x[1]), language), x[2]) for x in fd_mat_t]

# make tables
fig, axs = plt.subplots(1,2, figsize=(8,6))

# summary table
# names for the table columns
a_col = [this_feature["name"], "total"]

axone = axs[0]
sut.hide_spines_ticks_grids(axone)

table_two = sut.make_a_table(axone, combined_summary,  colLabels=a_col,
colWidths=[.5,.25,.25],  bbox=[0,0,1,1], **{"loc":"lower center"})
table_two.get_celld()[(0,0)].get_text().set_text(" ")
table_two.set_fontsize(12)

# material table
axtwo = axs[1]
axtwo.set_xlabel(" ")
sut.hide_spines_ticks_grids(axtwo)

table_three = sut.make_a_table(axtwo, fd_mat_t,
colLabels=list(cols_to_use.values()), colWidths=[.4, .3,.3],  bbox=[0,0,1,1],
```

```
**{"loc":"lower center"})
table_three.get_celld()[(0,0)].get_text().set_text(" ")

plt.tight_layout()
plt.subplots_adjust(wspace=0.2)
glue('aare_survey_area_sample_material_tables', fig, display=False)
plt.close()
```

Abbildung 3.5: **Links**: Zusammenfassung der Daten aller Erhebungen Erhebungsgebiet Aare. **Rechts:** Gefundene Materialarten im Erhebungsgebiet Aare in Stückzahlen und als prozentuale Anteile (stückzahlbezogen).

	total
Anzahl Proben	140
Durchschnitt p/100 m	225
Standardfehler	234
min p/100 m	2
25%	63
50%	143
75%	315
max p/100 m	1 480

	Gesamt	% G
Plastik	11 622	83
Glas	874	6.
Metall	579	4.
Papier	391	2.
Chemikalien	99	0.
Gummi	96	0.
Holz	94	0.
Stoff	92	0.
Unbekannt	0	0.

Abb. 3.5 #

3.2. Die am häufigsten gefundenen Objekte#

Die am häufigsten gefundenen Objekte sind die zehn mengenmässig am meisten vorkommenden Objekte und/oder Objekte, die in mindestens 50 % aller Datenerhebungen identifiziert wurden (Häufigkeitsrate)

```
# the top ten by quantity
most_abundant = code_totals.sort_values(by="quantity", ascending=False)[:10]
```

```
# the most common
most_common = code_totals[code_totals["fail rate"] >= 
a_fail_rate].sort_values(by="quantity", ascending=False)

# merge with most_common and drop duplicates
m_common = pd.concat([most_abundant, most_common]).drop_duplicates()

# get percent of total
m_common_percent_of_total = m_common.quantity.sum()/code_totals.quantity.sum()

rb_string = f"""
*__Unten__: Häufigste Objekte im {this_feature['name']}: d. h. Objekte mit einer 
Häufigkeitsrate von mindestens 50 % und/oder Top Ten nach Anzahl. Zusammengenommen 
machen die häufigsten Objekte {int(m_common_percent_of_total*100)}% aller gefundenen 
Objekte aus. Anmerkung: p/100 m = Medianwert der Erhebung.*
"""
md(rb_string)
```

***Unten**: Häufigste Objekte im Erhebungsgebiet Aare: d. h. Objekte mit einer Häufigkeitsrate von mindestens 50 % und/oder Top Ten nach Anzahl. Zusammengenommen machen die häufigsten Objekte 67% aller gefundenen Objekte aus. Anmerkung: p/100 m = Medianwert der Erhebung.*

```
# format values for table
m_common["item"] = m_common.index.map(lambda x: code_description_map.loc[x])
m_common["% of total"] = m_common["% of total"].map(lambda x: F"{x}%")
m_common["quantity"] = m_common.quantity.map(lambda x:thousandsSeparator(x, 
language))
m_common["fail rate"] = m_common["fail rate"].map(lambda x: F"{x}%")
m_common[unit_label] = m_common[unit_label].map(lambda x: F"{round(x,1)}")

# table header rows
cols_to_use = {"item":"Objekt","quantity":"Gesamt", "% of total":"% Gesamt", "fail 
rate":"fail-rate", unit_label:unit_label}

most_common_table = m_common[cols_to_use.keys()].values

fig, axs = plt.subplots(figsize=(11,len(m_common)*.7))

sut.hide_spines_ticks_grids(axs)

table_four = sut.make_a_table(axs, most_common_table, 
colLabels=list(cols_to_use.values()), colWidths=[.52, .12,.12,.12, .12], 
bbox=[0,0,1,1], **{"loc":"lower center"})
table_four.get_celld()[(0,0)].get_text().set_text(" ")
table_four.set_fontsize(12)
plt.tight_layout()
glue('aare_survey_area_most_common_tables', fig, display=False)
plt.close()
```

	Gesamt
Zigarettenfilter	2 561
Fragmentierte Kunststoffstücke	2 004
Snack-Verpackungen	900
Expandiertes Polystyrol	839
Industriefolie (Kunststoff)	812
Getränke Glasflasche, Stücke	687
Verpackungsfolien, nicht für Lebensmittel	445
Schaumstoffverpackungen/Isolierung	298
Styropor < 5mm	292
Industriepellets (Nurdles)	262
Verpackungen aus Aluminiumfolie	205

Abb. 3.6 #

3.2.1. Die am häufigsten gefundenen Objekte nach Gewässer#

```
# aggregated survey totals for the most common codes for all the water features
m_common_st = fd[fd.code.isin(m_common.index)].groupby([this_level,
"loc_date","code"], as_index=False).agg(agg_pcs_quantity)
m_common_ft = m_common_st.groupby([this_level, "code"], as_index=False)
[unit_label].median()

# proper name of water feature for display
m_common_ft["f_name"] = m_common_ft[this_level].map(lambda x: comp_labels[x])

# map the desctiption to the code
```

```
m_common_ft["item"] = m_common_ft.code.map(lambda x: code_description_map.loc[x])

# pivot that
m_c_p = m_common_ft[["item", unit_label, "f_name"]].pivot(columns="f_name",
index="item")

# quash the hierarchal column index
m_c_p.columns = m_c_p.columns.get_level_values(1)

# the aggregated totals for the survey area
c = sut.aggregate_to_group_name(fd[fd.code.isin(m_common.index)], column="code",
name=this_feature["name"], val="med", unit_label=unit_label)

m_c_p[this_feature["name"]]= sut.change_series_index_labels(c,
{x:code_description_map.loc[x] for x in c.index})

# the aggregated totals of all the data
c = sut.aggregate_to_group_name(a_data[(a_data.code.isin(m_common.index))],
column="code", name=top, val="med", unit_label=unit_label)
m_c_p[top] = sut.change_series_index_labels(c, {x:code_description_map.loc[x] for x
in c.index})

# chart that
fig, ax  = plt.subplots(figsize=(len(m_c_p.columns)*.9,len(m_c_p)*.9))
axone = ax

sns.heatmap(m_c_p, ax=axone, cmap=cmap2, annot=True, annot_kws={"fontsize":12},
fmt=".1f", square=True, cbar=False, linewidth=.1, linecolor="white")
axone.set_xlabel("")
axone.set_ylabel("")
axone.tick_params(labelsize=14, which="both", axis="x")
axone.tick_params(labelsize=12, which="both", axis="y")

plt.setp(axone.get_xticklabels(), rotation=90)

glue('aare_survey_area_most_common_heat_map', fig, display=False)
plt.close()
```

Abbildung 3.7: Median (p/100 m) der häufigsten Objekte im Erhebungsgebiet Aare.

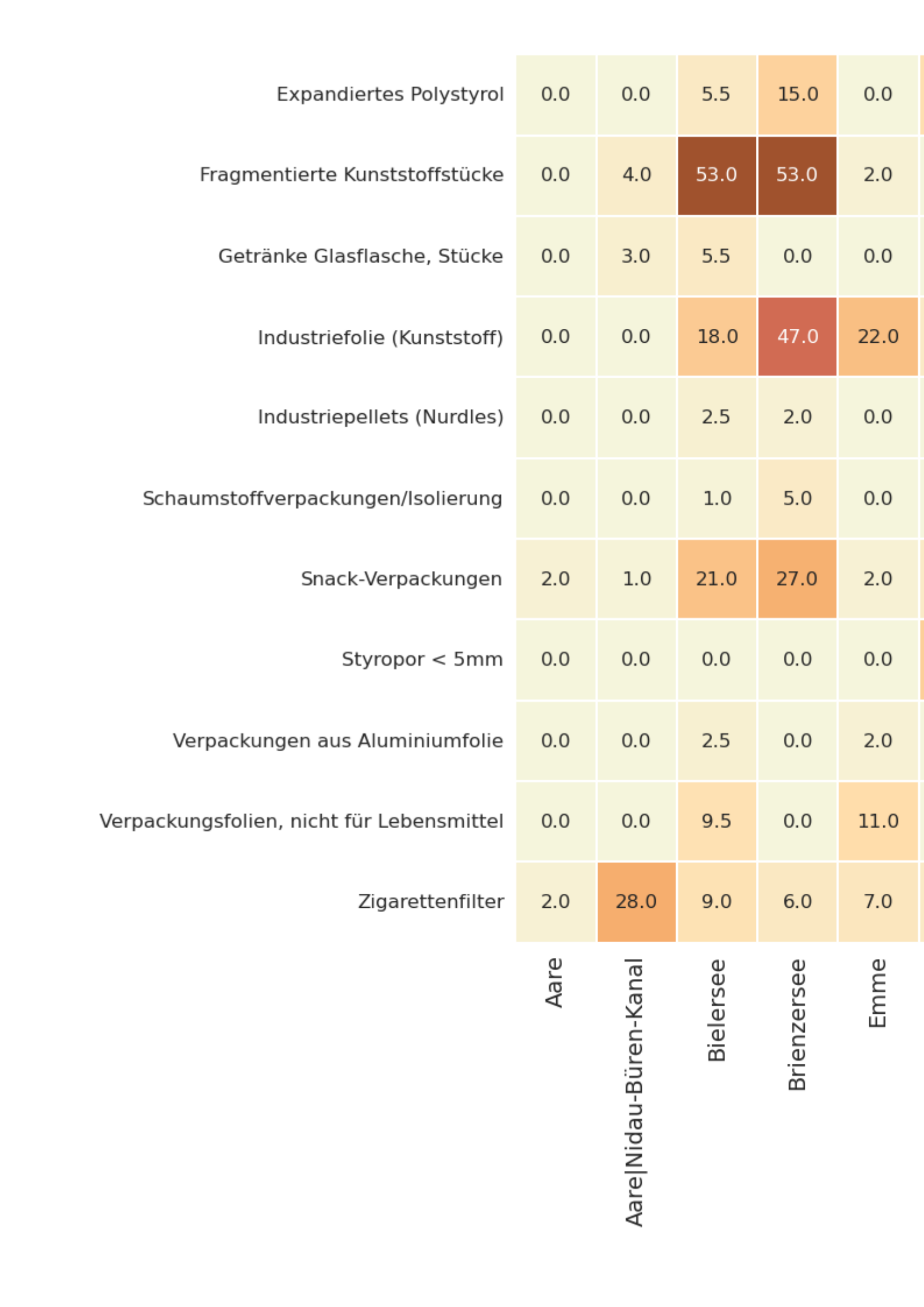

Expandiertes Polystyrol
0.0
0.0
5.5
15.0
0.0
Fragmentierte Kunststoffstücke
0.0
4.0
53.0
53.0
2.0
Getränke Glasflasche, Stücke
0.0
3.0
5.5
0.0
0.0
Industriefolie (Kunststoff)
0.0
0.0
18.0
47.0
22.0
Industriepellets (Nurdles)
0.0
0.0
2.5
2.0
0.0
Schaumstoffverpackungen/Isolierung
0.0
0.0
1.0
5.0
0.0
Snack-Verpackungen
2.0
1.0
21.0
27.0
2.0
Styropor < 5mm
0.0
0.0
0.0
0.0
0.0
Verpackungen aus Aluminiumfolie
0.0
0.0
2.5
0.0
2.0
Verpackungsfolien, nicht für Lebensmittel
0.0
0.0
9.5
0.0
11.0
Zigarettenfilter
2.0
28.0
9.0
6.0
7.0
Aare
Aare|Nidau-Büren-Kanal
Bielersee
Brienzersee
Emme

Abb. 3.7 #

3.2.2. Die am häufigsten gefundenen Objekte im monatlichen Durchschnitt#

```
# collect the survey results of the most common objects
m_common_m = fd[(fd.code.isin(m_common.index))].groupby(["loc_date","date","code",
"groupname"], as_index=False).agg(agg_pcs_quantity)
m_common_m.set_index("date", inplace=True)

# set the order of the chart, group the codes by groupname columns
an_order = m_common_m.groupby(["code","groupname"],
as_index=False).quantity.sum().sort_values(by="groupname")["code"].values

# a manager dict for the monthly results of each code
mgr = {}

# get the monhtly results for each code:
for a_group in an_order:
    # resample by month
    a_plot = m_common_m[(m_common_m.code==a_group)]
[unit_label].resample("M").mean().fillna(0)
    this_group = {a_group:a_plot}
    mgr.update(this_group)

months={
    0:"Jan",
    1:"Feb",
    2:"Mar",
    3:"Apr",
    4:"May",
    5:"Jun",
    6:"Jul",
    7:"Aug",
    8:"Sep",
    9:"Oct",
    10:"Nov",
    11:"Dec"
}

# convenience function to lable x axis
def new_month(x):
    if x <= 11:
        this_month = x
    else:
        this_month=x-12
    return this_month

fig, ax = plt.subplots(figsize=(10,7))

# define a bottom
bottom = [0]*len(mgr["G27"])

# the monhtly survey average for all objects and locations
# makes the backdrop of the barchart
monthly_fd = fd.groupby(["loc_date", "date"], as_index=False).agg(agg_pcs_quantity)
```

```
monthly_fd.set_index("date", inplace=True)
m_fd = monthly_fd[unit_label].resample("M").mean().fillna(0)

# define the xaxis
this_x = [i for i,x in  enumerate(m_fd.index)]

# plot the monthly total survey average
ax.bar(this_x, m_fd.to_numpy(), color=table_row, alpha=0.2, linewidth=1,
edgecolor="teal", width=1, label="Monatsdurschnitt")

# plot the monthly survey average of the most common objects
for i, a_group in enumerate(an_order):

    # define the axis
    this_x = [i for i,x in  enumerate(mgr[a_group].index)]

    # collect the month
    this_month = [x.month for i,x in enumerate(mgr[a_group].index)]

    # if i == 0 laydown the first bars
    if i == 0:
        ax.bar(this_x, mgr[a_group].to_numpy(), label=a_group,
color=colors_palette[a_group], linewidth=1, alpha=0.6 )
    # else use the previous results to define the bottom
    else:
        bottom += mgr[an_order[i-1]].to_numpy()
        ax.bar(this_x, mgr[a_group].to_numpy(), bottom=bottom, label=a_group,
color=colors_palette[a_group], linewidth=1, alpha=0.8)

# collect the handles and labels from the legend
handles, labels = ax.get_legend_handles_labels()

# set the location of the x ticks
ax.xaxis.set_major_locator(ticker.FixedLocator([i for i in np.arange(len(this_x))]))
ax.set_ylabel(unit_label, **ck.xlab_k14)

# label the xticks by month
axisticks = ax.get_xticks()
labelsx = [sut.months_de[new_month(x-1)] for x in  this_month]
plt.xticks(ticks=axisticks, labels=labelsx)

# make the legend
# swap out codes for descriptions
new_labels = [code_description_map.loc[x] for x in labels[1:]]
new_labels = new_labels[::-1]

# insert a label for the monthly average
new_labels.insert(0,"Monatsdurschnitt")
handles = [handles[0], *handles[1:][::-1]]

plt.legend(handles=handles, labels=new_labels, bbox_to_anchor=(.5, -.05), loc="upper
center",  ncol=1, fontsize=14)

glue('aare_survey_area_monthly_results', fig, display=False)
```

```
plt.close()
```

Abbildung 3.8: Erhebungsgebiet Aare, monatliche Durchschnittsergebnisse p/100 m.

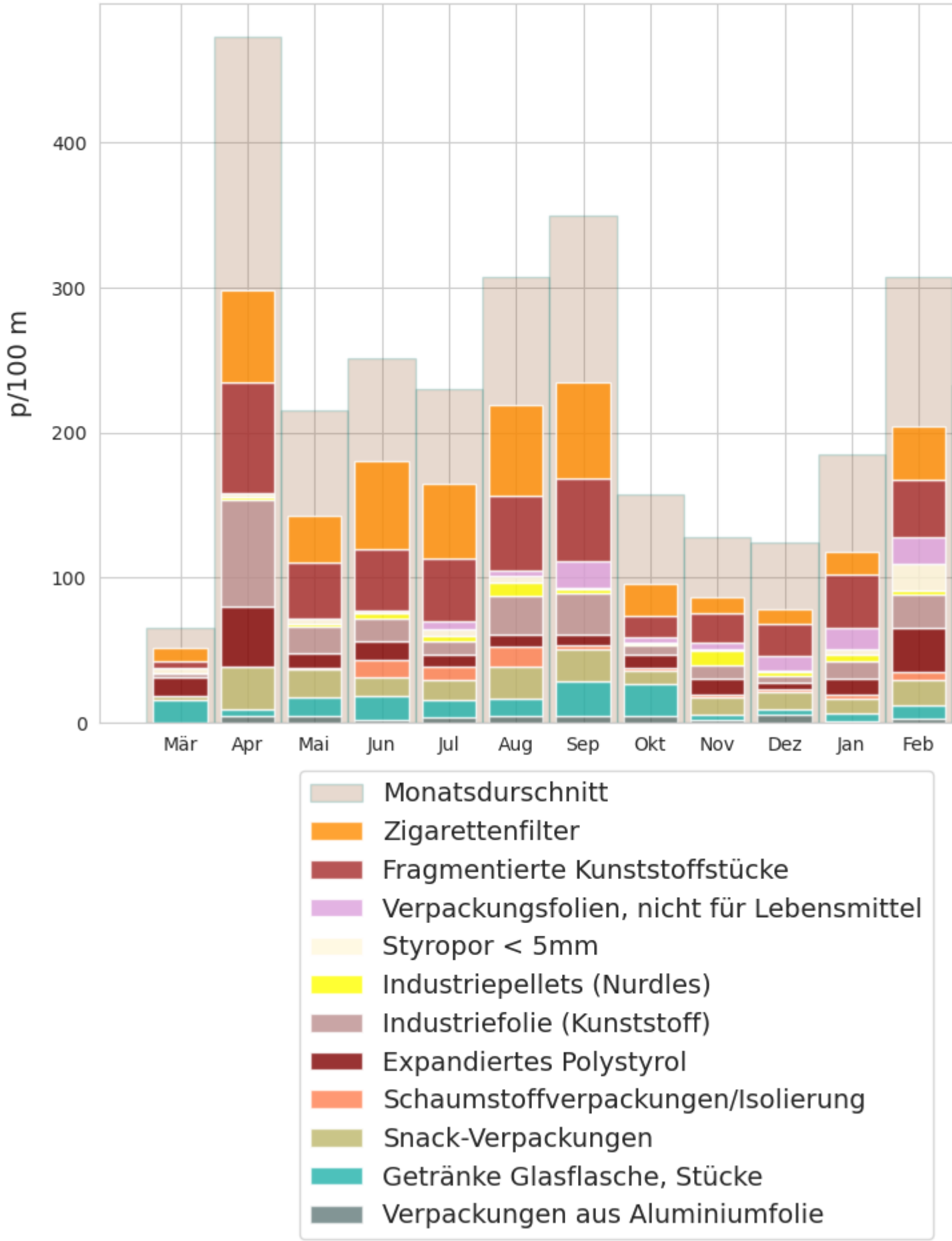


Abb. 3.8 #

3.3. Erhebungsergebnisse und Landnutzung#

Das Landnutzungsprofil ist eine Darstellung der Art und des Umfangs der wirtschaftlichen Aktivität und der Umweltbedingungen rund um den Erhebungsort. Die Schlüsselindikatoren aus den Ergebnissen der Datenerhebungen werden mit dem Landnutzungsprofil für einen Radius von 1500 m um den Erhebungsort verglichen.

Eine Assoziation ist eine Beziehung zwischen den Ergebnissen der Datenerhebungen und dem Landnutzungsprofil, die nicht auf Zufall beruht. Das Ausmass der Beziehung ist weder definiert noch linear.

Die Rangkorrelation ist ein nicht-parametrischer Test, um festzustellen, ob ein statistisch signifikanter Zusammenhang zwischen der Landnutzung und den bei einer Abfallobjekte-Erhebung identifizierten Objekten besteht.

Die verwendete Methode ist der Spearmans Rho oder Spearmans geordneter Korrelationskoeffizient. Die Testergebnisse werden bei $p < 0,05$ für alle gültigen Erhebungen an Seen im Untersuchungsgebiet ausgewertet.

1. Rot/Rosa steht für eine positive Assoziation

2. Gelb steht für eine negative Assoziation

3. Weiss bedeutet, dass keine statistische Grundlage für die Annahme eines Zusammenhangs besteht

```
# chart the results of test for association
fig, axs = plt.subplots(len(m_common.index),len(land_use_columns),
figsize=(len(land_use_columns)+7,len(m_common.index)+1), sharey="row")

# the test is conducted on the survey results for each code
for i,code in enumerate(m_common.index):
    # slice the data
    data = corr_data[corr_data.code == code]

    # run the test on for each land use feature
    for j, n in enumerate(land_use_columns):
        # assign ax and set some parameters
        ax=axs[i, j]
        ax.grid(False)
        ax.tick_params(axis="both",
which="both",bottom=False,top=False,labelbottom=False, labelleft=False, left=False)

        # check the axis and set titles and labels
        if i == 0:
            ax.set_title(F"{n}")
        else:
            pass

        if j == 0:
            ax.set_ylabel(F"{code_description_map[code]}", rotation=0, ha="right",
**ck.xlab_k14)
            ax.set_xlabel(" ")
        else:
```

```
            ax.set_xlabel(" ")
            ax.set_ylabel(" ")
        # run test
        _, corr, a_p = sut.make_plot_with_spearmans(data, ax, n, 
unit_label=unit_label)

        # if siginficant set adjust color to direction
        if a_p < 0.05:
            if corr > 0:
                ax.patch.set_facecolor("salmon")
                ax.patch.set_alpha(0.5)
            else:
                ax.patch.set_facecolor("palegoldenrod")
                ax.patch.set_alpha(0.5)

plt.tight_layout()
plt.subplots_adjust(wspace=0, hspace=0)
glue('aare_survey_area_spearmans', fig, display=False)
plt.close()
```

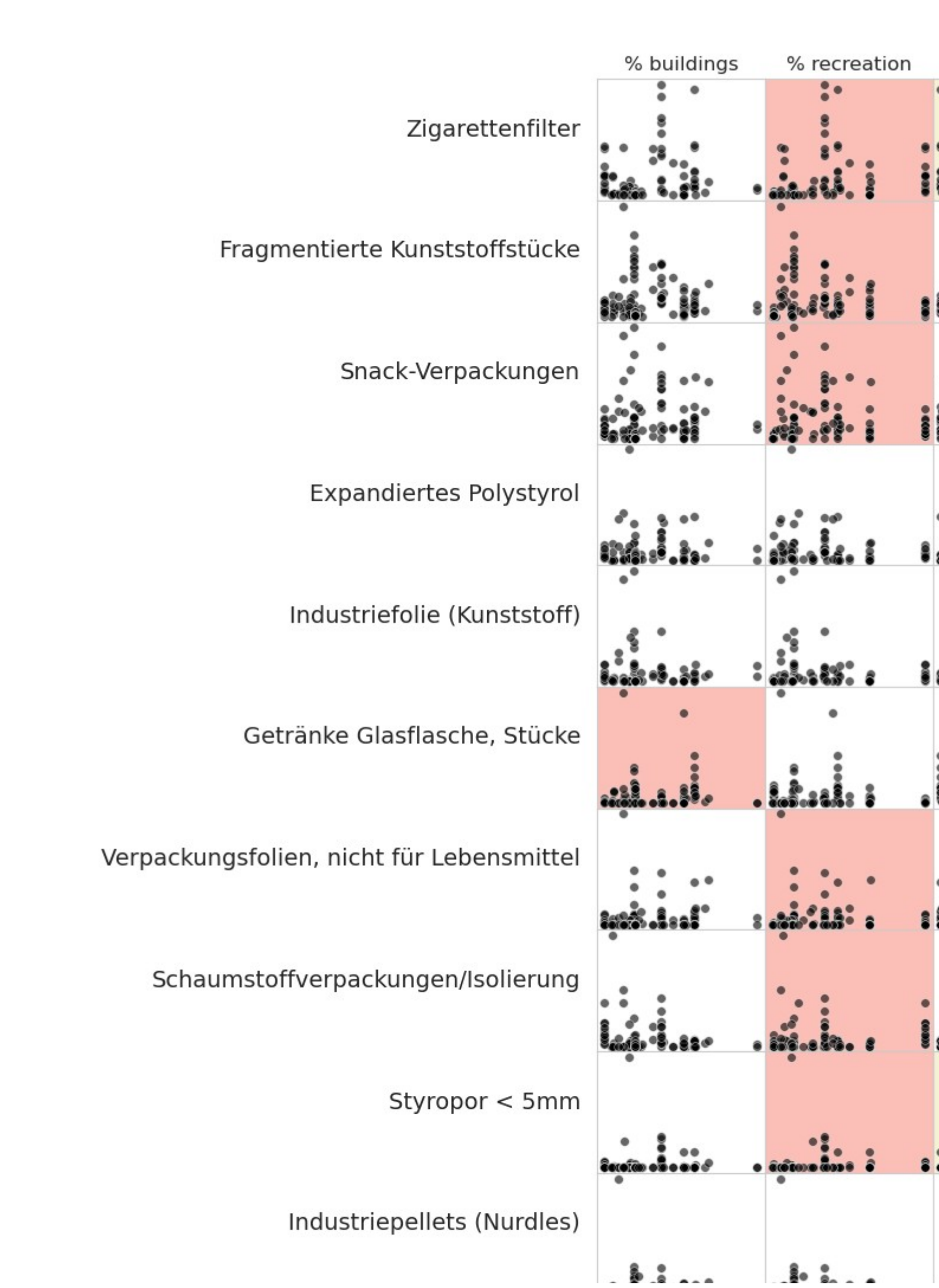

% buildings
% recreation
Zigarettenfilter
Fragmentierte Kunststoffstücke
Snack-Verpackungen
Expandiertes Polystyrol
Industriefolie (Kunststoff)
Getränke Glasflasche, Stücke
Verpackungsfolien, nicht für Lebensmittel
Schaumstoffverpackungen/Isolierung
Styropor < 5mm
Industriepellets (Nurdles)

Abb. 3.9 #

Abbildung 3.9: Ausgewertete Korrelationen der am häufigsten gefundenen Objekte in Bezug auf das Landnutzungsprofil im Erhebungsgebiet Aare. Für alle gültigen Erhebungen an Seen n = 118. ***Legende:*** *wenn p > 0,05 = weiss, wenn p < 0,05 und Rho > 0 = rot, wenn p < 0,05 und Rho < 0 = gelb*

3.4. Verwendungszweck der gefundenen Objekte#

Der Verwendungszweck basiert auf der Verwendung des Objekts, bevor es weggeworfen wurde, oder auf der Artikelbeschreibung, wenn die ursprüngliche Verwendung unbestimmt ist. Identifizierte Objekte werden einer der 260 vordefinierten Kategorien zugeordnet. Die Kategorien werden je nach Verwendung oder Artikelbeschreibung gruppiert.

- Abwasser: Objekte, die aus Kläranlagen freigesetzt werden, sprich Objekte, die wahrscheinlich über die Toilette entsorgt werden
- Mikroplastik (< 5 mm): fragmentierte Kunststoffe und Kunststoffharze aus der Vorproduktion
- Infrastruktur: Artikel im Zusammenhang mit dem Bau und der Instandhaltung von Gebäuden, Strassen und der Wasser-/Stromversorgung
- Essen und Trinken: alle Materialien, die mit dem Konsum von Essen und Trinken in Zusammenhang stehen
- Landwirtschaft: Materialien z. B. für Mulch und Reihenabdeckungen, Gewächshäuser, Bodenbegasung, Ballenverpackungen. Einschliesslich Hartkunststoffe für landwirtschaftliche Zäune, Blumentöpfe usw.
- Tabakwaren: hauptsächlich Zigarettenfilter, einschliesslich aller mit dem Rauchen verbundenen Materialien
- Freizeit und Erholung: Objekte, die mit Sport und Freizeit zu tun haben, z. B. Angeln, Jagen, Wandern usw.
- Verpackungen ausser Lebensmittel und Tabak: Verpackungsmaterial, das nicht lebensmittel- oder tabakbezogen ist
- Plastikfragmente: Plastikteile unbestimmter Herkunft oder Verwendung
- Persönliche Gegenstände: Accessoires, Hygieneartikel und Kleidung

Im Anhang (Kapitel 3.6.3) befindet sich die vollständige Liste der identifizierten Objekte, einschliesslich Beschreibungen und Gruppenklassifizierung. Das Kapitel 16 Codegruppen beschreibt jede Codegruppe im Detail und bietet eine umfassende Liste aller Objekte in einer Gruppe.

```
# code groups results aggregated by survey
groups = ["loc_date","groupname"]
cg_t = fd.groupby([this_level,*groups], as_index=False).agg(agg_pcs_quantity)

# the total per water feature
cg_tq = cg_t.groupby(this_level).quantity.sum()

# get the fail rates for each group per survey
cg_t["fail"]=False
cg_t["fail"] = cg_t.quantity.where(lambda x: x == 0, True)
```

```
# aggregate all that for each municipality
agg_this = {unit_label:"median", "quantity":"sum", "fail":"sum",
"loc_date":"nunique"}
cg_t = cg_t.groupby([this_level, "groupname"], as_index=False).agg(agg_this)

# assign survey area total to each record
for a_feature in cg_tq.index:
    cg_t.loc[cg_t[this_level] == a_feature, "f_total"] = cg_tq.loc[a_feature]

# get the percent of total for each group for each survey area
cg_t["pt"] = (cg_t.quantity/cg_t.f_total).round(2)

# pivot that
data_table = cg_t.pivot(columns=this_level, index="groupname", values="pt")

# repeat for the survey area
data_table[this_feature['name'] ] = sut.aggregate_to_group_name(fd,
unit_label=unit_label, column="groupname", name=this_feature['name'] , val="pt")

# repeat for all the data
data_table[top] = sut.aggregate_to_group_name(a_data, unit_label=unit_label,
column="groupname", name=top, val="pt")

data = data_table
data.rename(columns={x:wname_wname.loc[x][0] for x in data.columns[:-2]},
inplace=True)

fig, ax = plt.subplots(figsize=(10,10))

axone = ax
sns.heatmap(data , ax=axone, cmap=cmap2, annot=True, annot_kws={"fontsize":12},
cbar=False, fmt=".0%", linewidth=.1, square=True, linecolor="white")

axone.set_ylabel("")
axone.set_xlabel("")
axone.tick_params(labelsize=14, which="both", axis="both", labeltop=False,
labelbottom=True)

plt.setp(axone.get_xticklabels(), rotation=90, fontsize=14)
plt.setp(axone.get_yticklabels(), rotation=0, fontsize=14)

glue('aare_survey_area_codegroup_percent', fig, display=False)

plt.close()
```

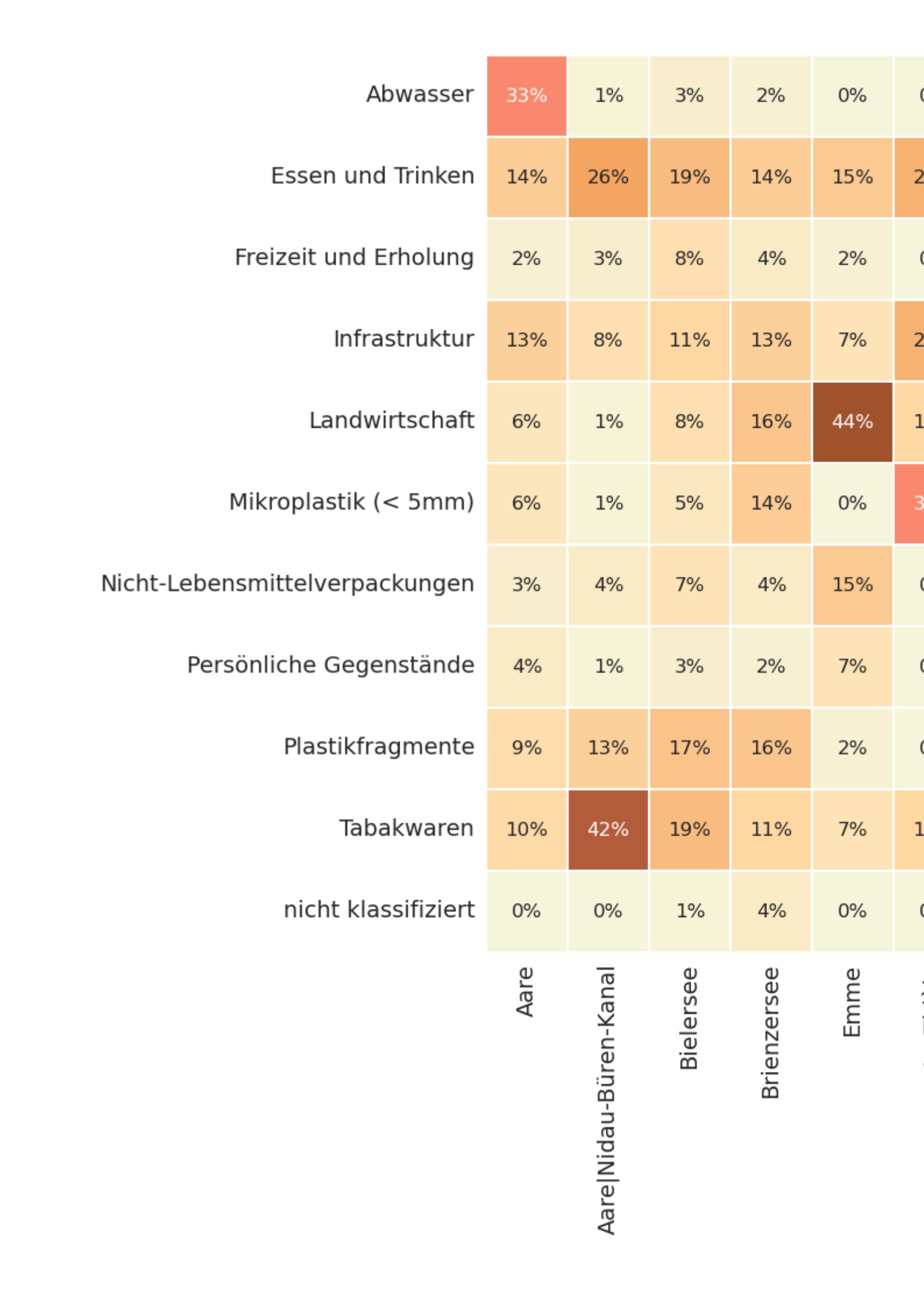

Abwasser
Essen und Trinken
Freizeit und Erholung
Infrastruktur
Landwirtschaft
Mikroplastik (< 5mm)
Nicht-Lebensmittelverpackungen
Persönliche Gegenstände
Plastikfragmente
Tabakwaren
nicht klassifiziert
33% 1% 3% 2% 0%
14% 26% 19% 14% 15%
2% 3% 8% 4% 2%
13% 8% 11% 13% 7%
6% 1% 8% 16% 44%
6% 1% 5% 14% 0%
3% 4% 7% 4% 15%
4% 1% 3% 2% 7%
9% 13% 17% 16% 2%
10% 42% 19% 11% 7%
0% 0% 1% 4% 0%
Aare
Aare|Nidau-Büren-Kanal
Bielersee
Brienzersee
Emme

Abb. 3.10 #

Abbildung 3.10: Verwendungszweck oder Beschreibung der identifizierten Objekte in % der Gesamtzahl nach Gewässer im Erhebungsgebiet Aare. Fragmentierte Objekte, die nicht eindeutig identifiziert werden können, werden weiterhin nach ihrer Grösse klassifiziert.

```
# median p/50m of all the water features
data_table = cg_t.pivot(columns="water_name_slug", index="groupname",
values=unit_label)

# the survey area columns
data_table[this_feature['name'] ] = sut.aggregate_to_group_name(fd,
unit_label=unit_label, column="groupname", name=this_feature['name'] , val="med")

# column for all the surveys
data_table[top] = sut.aggregate_to_group_name(a_data, unit_label=unit_label,
column="groupname", name=top, val="med")

# merge with data_table
data = data_table
data.rename(columns={x:wname_wname.loc[x][0] for x in data.columns[:-2]},
inplace=True)
fig, ax = plt.subplots(figsize=(10,10))

axone = ax
sns.heatmap(data , ax=axone, cmap=cmap2, annot=True, annot_kws={"fontsize":12},
fmt="g", cbar=False, linewidth=.1, square=True, linecolor="white")

axone.set_xlabel("")
axone.set_ylabel("")
axone.tick_params(labelsize=14, which="both", axis="both", labeltop=False,
labelbottom=True)

plt.setp(axone.get_xticklabels(), rotation=90, fontsize=14)
plt.setp(axone.get_yticklabels(), rotation=0, fontsize=14)
glue('aare_survey_area_codegroup_pcsm', fig, display=False)
plt.close()
```

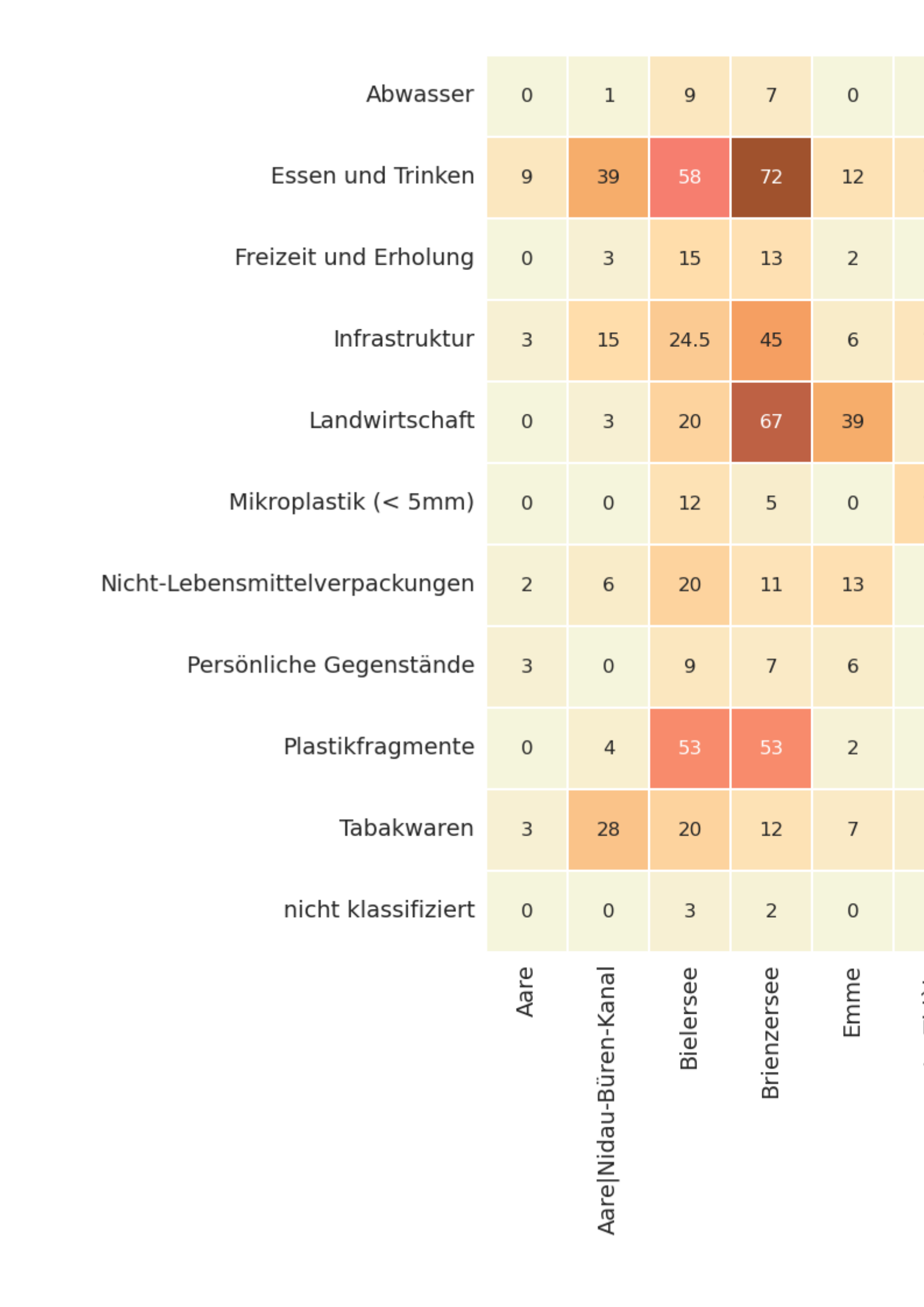

Abwasser
Essen und Trinken
Freizeit und Erholung
Infrastruktur
Landwirtschaft
Mikroplastik (< 5mm)
Nicht-Lebensmittelverpackungen
Persönliche Gegenstände
Plastikfragmente
Tabakwaren
nicht klassifiziert
0 1 9 7 0
9 39 58 72 12
0 3 15 13 2
3 15 24.5 45 6
0 3 20 67 39
0 0 12 5 0
2 6 20 11 13
3 0 9 7 6
0 4 53 53 2
3 28 20 12 7
0 0 3 2 0
Aare
Aare|Nidau-Büren-Kanal
Bielersee
Brienzersee
Emme

Abb. 3.11 #

Abbildung 3.11: Verwendungszweck der gefundenen Objekte Median p/100 m im Erhebungsgebiet Aare. Fragmentierte Objekte, die nicht eindeutig identifiziert werden können, werden weiterhin nach ihrer Grösse klassifiziert.

3.5. Fliessgewässer#

```
cs = r_smps[unit_label].describe().round(2)

# add project totals
cs["total objects"] = r_smps.quantity.sum()

# change the names
csx = sut.change_series_index_labels(cs, sut.create_summary_table_index(unit_label,
lang="DE"))

combined_summary = sut.fmt_combined_summary(csx, nf=[])

# make the charts
fig = plt.figure(figsize=(11,6))

aspec = fig.add_gridspec(ncols=11, nrows=3)

ax = fig.add_subplot(aspec[:, :6])

line_label = F"{rate} median:{top}"

sns.scatterplot(data=l_smps, x="date", y=unit_label, color="black", alpha=0.4,
label="Lake surveys", ax=ax)
sns.scatterplot(data=r_smps, x="date", y=unit_label, color="red", s=34,
ec="white",label="River surveys", ax=ax)

ax.set_ylim(-10,y_limit )

ax.set_xlabel("")
ax.set_ylabel(unit_label, **ck.xlab_k14)

ax.xaxis.set_minor_locator(days)
ax.xaxis.set_major_formatter(months_fmt)

a_col = [this_feature["name"], "total"]

axone = fig.add_subplot(aspec[:, 7:])
sut.hide_spines_ticks_grids(axone)

table_five = sut.make_a_table(axone, combined_summary,  colLabels=a_col,
colWidths=[.75,.25],  bbox=[0,0,1,1], **{"loc":"lower center"})
table_five.get_celld()[(0,0)].get_text().set_text(" ")

glue('aare_survey_area_rivers_summary', fig, display=False)
plt.close()
```

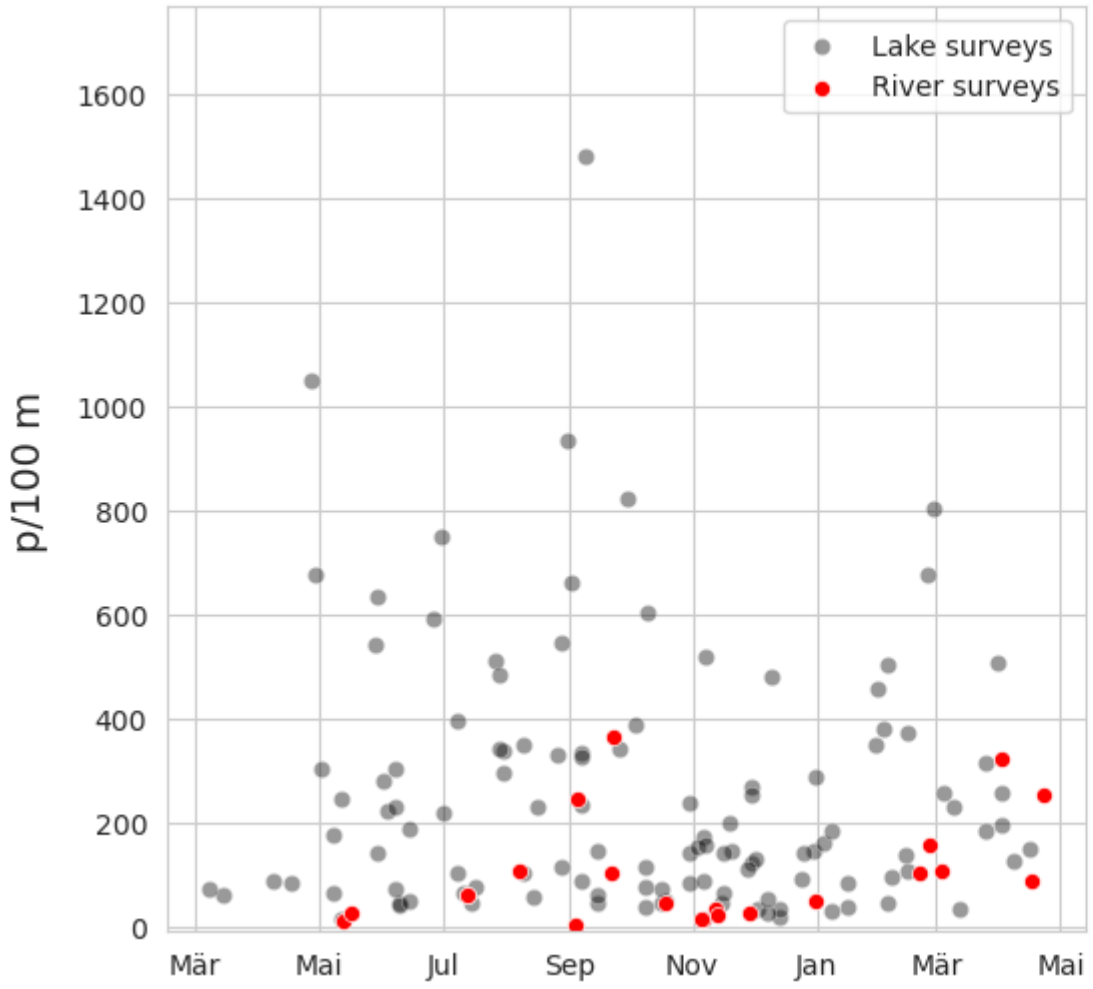


Abb. 3.12 #

Abbildung 3.12: **Links:** Erhebungsgebiet Aare Fliessgewässer, März 2020 bis Mai 2021, n = 22. Werte grösser als 1 766 p/100 m werden nicht gezeigt. **Rechts:** Zusammenfassung der Daten.

3.5.1. Die an Fliessgewässern am häufigsten gefundenen Objekte#

```
# the most common items rivers
r_codes = rivers.groupby("code").agg({"quantity":"sum", "fail":"sum",
unit_label:"median"})
r_codes["Fail rate"] = (r_codes.fail/r_smps.loc_date.nunique()*100).astype("int")

# top ten
r_byq = r_codes.sort_values(by="quantity", ascending=False)[:10].index

# most common
r_byfail = r_codes[r_codes["Fail rate"] > 49.99].index
r_most_common = list(set(r_byq) | set(r_byfail))

# format for display
r_mc= r_codes.loc[r_most_common].copy()
r_mc["item"] = r_mc.index.map(lambda x: code_description_map.loc[x])
r_mc.sort_values(by="quantity", ascending=False, inplace=True)

r_mc["% of total"]=((r_mc.quantity/r_codes.quantity.sum())*100).astype("int")
r_mc["% of total"] = r_mc["% of total"].map(lambda x: F"{x}%")
r_mc["quantity"] = r_mc.quantity.map(lambda x: "{:,}".format(x))
```

```
r_mc["Fail rate"] = r_mc["Fail rate"].map(lambda x: F"{x}%")
r_mc["p/50m"] = r_mc[unit_label].map(lambda x: F"{np.ceil(x)}")
r_mc.rename(columns=cols_to_use, inplace=True)

data=r_mc[["Objekt","Gesamt", "% Gesamt", "Fail rate", unit_label]]

fig, axs = plt.subplots(figsize=(12,len(data)*.8))

sut.hide_spines_ticks_grids(axs)

table_six = sut.make_a_table(axs, data.values,  colLabels=list(data.columns),
colWidths=[.48, .13,.13,.13, .13], **{"loc":"lower center"})
table_six.get_celld()[(0,0)].get_text().set_text(" ")



plt.tight_layout()
glue('aare_survey_area_rivers_most_common', fig, display=False)
plt.close()
```

	Gesamt
Zigarettenfilter	133
Windeln - Tücher	117
Fragmentierte Kunststoffstücke	68
Snack-Verpackungen	41
Expandiertes Polystyrol	37
Industriefolie (Kunststoff)	33
Getränke Glasflasche, Stücke	30
Styropor < 5mm	17
Tampons	17
Schaumstoffverpackungen/Isolierung	16

Abb. 3.13 #

Abbildung 3.13: Häufigste Objekte p/100 m an Fliessgewässern im Erhebungsgebiet Aare: Medianwert der Erhebung.

3.6. Anhang#

3.6.1. Schaumstoffe und Kunststoffe nach Grösse#

Die folgende Tabelle enthält die Komponenten «Gfoam» und «Gfrag», die für die Analyse gruppiert wurden. Objekte, die als Schaumstoffe gekennzeichnet sind, werden als Gfoam gruppiert und umfassen

alle geschäumten Polystyrol-Kunststoffe > 0,5 cm. Kunststoffteile und Objekte aus kombinierten Kunststoff- und Schaumstoffmaterialien > 0,5 cm werden für die Analyse als Gfrags gruppiert.

```
# collect the data before aggregating foams for all locations in the survye area
# group by loc_date and code
# Combine the different sizes of fragmented plastics and styrofoam
# the codes for the foams
before_agg = pd.read_csv("resources/checked_before_agg_sdata_eos_2020_21.csv")
some_foams = ["G81", "G82", "G83", "G74"]
before_agg.rename(columns={"p/100m":unit_label}, inplace=True)

# the codes for the fragmented plastics
some_frag_plas = list(before_agg[before_agg.groupname == "plastic
pieces"].code.unique())

fd_frags_foams = before_agg[(before_agg.code.isin([*some_frag_plas,
*some_foams]))&(before_agg.location.isin(a_summary["locations_of_interest"]))].groupb
y(["loc_date","code"], as_index=False).agg(agg_pcs_quantity)
fd_frags_foams = fd_frags_foams.groupby("code").agg(agg_pcs_median)

# add code description and format for printing
fd_frags_foams["item"] = fd_frags_foams.index.map(lambda x:
code_description_map.loc[x])
fd_frags_foams["% of total"] =
(fd_frags_foams.quantity/fd.quantity.sum()*100).round(2)
fd_frags_foams["% of total"] = fd_frags_foams["% of total"].map(lambda x: F"{x}%")
fd_frags_foams["quantity"] = fd_frags_foams["quantity"].map(lambda x: F"{x:,}")

# table data
data = fd_frags_foams[["item",unit_label, "quantity", "% of total"]]
data.rename(columns={"quantity":"Gesamt", "% of total":"% Gesamt"}, inplace=True)

fig, axs = plt.subplots(figsize=(len(data.columns)*2.4,len(data)*.7))

sut.hide_spines_ticks_grids(axs)

table_seven = sut.make_a_table(axs,data.values,  colLabels=data.columns,
colWidths=[.6, .13, .13, .13], a_color=table_row)
table_seven.get_celld()[(0,0)].get_text().set_text(" ")
table_seven.set_fontsize(12)


plt.tight_layout()
glue('aare_survey_area_fragmented_plastics', fig, display=False)
plt.close()
```

	p/100 m	
Schaumstoffverpackungen/Isolierung	0.0	
Kunststoff-/Polystyrolteile 0,5 - 2,5 cm	0.0	
Kunststoff/Polystyrolschaumstoff 2,5 > < 50	0.0	
Kunststoffstücke 0,5cm - 2,5cm	8.0	
Kunststoffstücke 2,5 cm - 50 cm	7.0	
Kunststoffteile > 50cm	0.0	
Schaumstoffstücke aus Polystyrol 0,5 cm - 2,5 cm	1.0	
Schaumstoffstücke aus Polystyrol 2,5 - 50cm	1.0	
Styroporstücke > 50cm	0.0	

Abb. 3.14 #

Abbildung 3.14: Fragmentierte und geschäumte Kunststoffe nach Grösse im Erhebungsgebiet Aare, Median p/100 m, Anzahl der gefundenen Objekte und Prozent der Gesamtmenge.

3.6.2. Die Erhebungsorte#

```
# display the survey locations
disp_columns = ["latitude", "longitude", "city"]
disp_beaches = dfBeaches.loc[a_summary["locations_of_interest"]][disp_columns]
disp_beaches.reset_index(inplace=True)
disp_beaches.rename(columns={"city":"stat", "slug":"standort"}, inplace=True)
disp_beaches.set_index("standort", inplace=True, drop=True)

disp_beaches
```

	latitude	longitude	stat
standort			
aare-port	47.116170	7.269550	Port
schutzenmatte	47.057666	7.634001	Burgdorf
la-petite-plage	46.785054	6.656877	Yverdon-les-Bains
weissenau-neuhaus	46.676583	7.817528	Unterseen
evole-plage	46.989477	6.923920	Neuchâtel
oberi-chlihochstetten	46.896025	7.532114	Rubigen
plage-de-serriere	46.984850	6.913450	Neuchâtel
mullermatte	47.133339	7.227907	Biel/Bienne
bielersee_vinelz_fankhausers	47.038398	7.108311	Vinelz
erlach-camping-strand	47.047159	7.097854	Erlach
gummligrabbe	46.989290	7.250730	Kallnach
mannewil	46.996382	7.239024	Kallnach
schusspark-strand	47.146500	7.268620	Biel/Bienne
plage-de-cheyres	46.818689	6.782256	Cheyres-Châbles
thun-strandbad	46.739939	7.633520	Thun
oben-am-see	46.744451	8.049921	Brienz (BE)
thunersee_spiez_meierd_1	46.704437	7.657882	Spiez
spackmatt	47.223749	7.576711	Luterbach
luscherz-plage	47.047955	7.151242	Lüscherz
strandboden-biel	47.132510	7.233142	Biel/Bienne
signalpain	46.786200	6.647360	Yverdon-les-Bains
pointe-dareuse	46.946190	6.870970	Boudry
nidau-strand	47.127196	7.232613	Nidau

	latitude	longitude	stat
standort			
camp-des-peches	47.052812	7.074053	Le Landeron
delta-park	46.720078	7.635304	Spiez
pecos-plage	46.803590	6.636650	Grandson
beich	47.048495	7.200528	Walperswil
aare_suhrespitz_badert	47.405669	8.066018	Aarau
sundbach-strand	46.684386	7.794768	Beatenberg
wycheley	46.740370	8.048640	Brienz (BE)
aare_koniz_hoppej	46.934588	7.458170	Köniz
camping-gwatt-strand	46.727140	7.629620	Thun
bern-tiergarten	46.933157	7.452004	Bern
la-thiele_le-mujon_confluence	46.775340	6.630900	Yverdon-les-Bains
nouvelle-plage	46.856646	6.848428	Estavayer
augustmutzenbergstrandweg	46.686640	7.689760	Spiez
aare_bern_gerberm	46.989363	7.452098	Bern
brugg-be-buren-bsg-standort	47.122220	7.281410	Brügg
ruisseau-de-la-croix-plage	46.813920	6.774700	Cheyres-Châbles
ligerz-strand	47.083979	7.135894	Ligerz
hafeli	46.690283	7.898592	Bönigen
aare-solothurn-lido-strand	47.196949	7.521643	Solothurn
bern-fahrstrasse	46.975866	7.444210	Bern
gals-reserve	47.046272	7.085007	Gals
hauterive-petite-plage	47.010797	6.980304	Hauterive (NE)
aare-limmatspitz	47.501060	8.237371	Gebenstorf

	latitude	longitude	stat
standort			
aare_brugg_buchie	47.494855	8.236558	Brugg
luscherz-two	47.047519	7.152829	Lüscherz
impromptu_cudrefin	46.964496	7.027936	Cudrefin
aare_post	47.116665	7.271953	Port
aare_bern_scheurerk	46.970967	7.452586	Bern

3.6.3. Inventar der Objekte#

```
pd.set_option("display.max_rows", None)
complete_inventory = code_totals[code_totals.quantity>0][["item", "groupname", "quantity", "% of total","fail rate"]]
complete_inventory.rename(columns={"item":"Objekte", "groupname":"Gruppenname", "quantity":"Gesamt", "% of total":"% Gesamt", "fail rate":"fail rate" }, inplace=True)
```

```
complete_inventory.sort_values(by="Gesamt", ascending=False)
```

	Objekte	Gruppenname	Gesamt	% Gesamt	fail rate
code					
G27	Zigarettenfilter	Tabakwaren	2561	18.49	84
Gfrags	Fragmentierte Kunststoffstücke	Plastikfragmente	2004	14.47	89
G30	Snack-Verpackungen	Essen und Trinken	900	6.50	82
Gfoam	Expandiertes Polystyrol	Infrastruktur	839	6.06	65
G67	Industriefolie (Kunststoff)	Landwirtschaft	812	5.86	73
G200	Getränke Glasflasche, Stücke	Essen und Trinken	687	4.96	67
G941	Verpackungsfolien, nicht für Lebensmittel	Nicht-Lebensmittelverpackungen	445	3.21	50
G74	Schaumstoffverpackungen/ Isolierung	Infrastruktur	298	2.15	48
G117	Styropor < 5mm	Mikroplastik (< 5mm)	292	2.11	27

	Objekte	Gruppenname	Gesamt	% Gesamt	fail rate
code					
G112	Industriepellets (Nurdles)	Mikroplastik (< 5mm)	262	1.89	32
G98	Windeln - Tücher	Abwasser	258	1.86	28
G89	Kunststoff-Bauabfälle	Infrastruktur	219	1.58	49
G177	Verpackungen aus Aluminiumfolie	Essen und Trinken	205	1.48	50
G95	Wattestäbchen / Tupfer	Abwasser	201	1.45	49
G25	Tabak; Kunststoffverpackungen, Behälter	Tabakwaren	159	1.15	43
G904	Feuerwerkskörper; Raketenkappen	Freizeit und Erholung	155	1.12	41
G156	Papierfragmente	Nicht-Lebensmittelverpackungen	150	1.08	28
G178	Kronkorken, Lasche von Dose/Ausfreisslachen	Essen und Trinken	123	0.89	40
G106	Kunststofffragmente eckig <5mm	Mikroplastik (< 5mm)	104	0.75	27
G940	Schaumstoff EVA (flexibler Kunststoff)	Freizeit und Erholung	102	0.74	11
G21	Getränke-Deckel, Getränkeverschluss	Essen und Trinken	98	0.71	31
G213	Paraffinwachs	Freizeit und Erholung	98	0.71	18
G50	Schnur < 1cm	Freizeit und Erholung	96	0.69	36
G23	unidentifizierte Deckel	Nicht-Lebensmittelverpackungen	93	0.67	30
G33	Einwegartikel; Tassen/Becher & Deckel	Essen und Trinken	87	0.63	35
G922	Etiketten, Strichcodes	Nicht-Lebensmittelverpackungen	87	0.63	32
G35	Strohhalme und Rührstäbchen	Essen und Trinken	82	0.59	35
G186	Industrieschrott	Infrastruktur	82	0.59	20

	Objekte	Gruppenname	Gesamt	% Gesamt	fail rate
code					
G24	Ringe von Plastikflaschen/Behältern	Essen und Trinken	80	0.58	28
G10	Lebensmittelbehälter zum einmaligen Gebrauch a...	Essen und Trinken	74	0.53	28
G211	Sonstiges medizinisches Material	Persönliche Gegenstände	73	0.53	32
G31	Schleckstengel, Stengel von Lutscher	Essen und Trinken	71	0.51	30
G32	Spielzeug und Partyartikel	Freizeit und Erholung	67	0.48	32
G923	Taschentücher, Toilettenpapier, Servietten, Pa...	Persönliche Gegenstände	62	0.45	24
G73	Gegenstände aus Schaumstoff & Teilstücke (nich...	Freizeit und Erholung	58	0.42	19
G66	Umreifungsbänder; Hartplastik für Verpackung f...	Infrastruktur	55	0.40	30
G153	Papierbecher, Lebensmittelverpackungen aus Pap...	Essen und Trinken	54	0.39	17
G203	Geschirr aus Keramik oder Glas, Tassen, Teller...	Essen und Trinken	49	0.35	13
G22	Deckel für Chemikalien, Reinigungsmittel (Ohne...	Infrastruktur	49	0.35	18
G936	Folien für Gewächshäuser	Landwirtschaft	48	0.35	13
G908	Klebeband; elektrisch, isolierend	Infrastruktur	46	0.33	20
G93	Kabelbinder	Infrastruktur	45	0.32	17
G204	Baumaterial; Ziegel, Rohre, Zement	Infrastruktur	45	0.32	10
G152	Papier &Karton;Zigarettenschachteln, Papier/Ka...	Tabakwaren	43	0.31	13

	Objekte	Gruppenname	Gesamt	% Gesamt	fail rate
code					
G3	Einkaufstaschen, Shoppingtaschen	Nicht-Lebensmittelverpackungen	42	0.30	17
G191	Draht und Gitter	Landwirtschaft	41	0.30	15
G96	Hygienebinden/ Höscheneinlagen/Tampons und Appl...	Abwasser	40	0.29	15
G100	Medizin; Behälter/Röhrchen/Verpackungen	Abwasser	40	0.29	22
G87	Abdeckklebeband / Verpackungsklebeband	Infrastruktur	39	0.28	18
G125	Luftballons und Luftballonstäbchen	Freizeit und Erholung	38	0.27	17
G91	Bio-Filtermaterial / Trägermaterial aus Kunst...	Abwasser	37	0.27	16
G905	Haarspangen, Haargummis, persönliche Accessoir...	Persönliche Gegenstände	35	0.25	19
G137	Kleidung, Handtücher und Lappen	Persönliche Gegenstände	35	0.25	10
G944	Pelletmasse aus dem Spritzgussverfahren	nicht klassifiziert	34	0.25	2
G175	Getränkedosen (Dosen, Getränke)	Essen und Trinken	31	0.22	13
G70	Schrotflintenpatronen	Freizeit und Erholung	30	0.22	15
G927	Plastikschnur von Rasentrimmern, die zum Schne...	Infrastruktur	27	0.19	8
G201	Gläser, einschliesslich Stücke	Essen und Trinken	27	0.19	8
G928	Bänder und Schleifen	Persönliche Gegenstände	27	0.19	11
G198	Andere Metallteile < 50 cm	Infrastruktur	26	0.19	14
G208	Glas oder Keramikfragmente >2.5 cm	nicht klassifiziert	26	0.19	7
G165	Glacestengel (Eisstiele),	Essen und Trinken	25	0.18	13

	Objekte	Gruppenname	Gesamt	% Gesamt	fail rate
code					
	Zahnstocher, Essstäb...				
G159	Kork	Essen und Trinken	25	0.18	12
G131	Gummibänder	Persönliche Gegenstände	24	0.17	12
G942	Kunststoffspäne von Drehbänken, CNC-Bearbeitung	nicht klassifiziert	24	0.17	9
G26	Feuerzeug	Tabakwaren	23	0.17	13
G210	Sonstiges Glas/Keramik Materialien	nicht klassifiziert	21	0.15	4
G914	Büroklammern, Wäscheklammern, Gebrauchsgegenst...	Persönliche Gegenstände	20	0.14	8
G90	Blumentöpfe aus Plastik	Landwirtschaft	20	0.14	10
G48	Seile, synthetische	Freizeit und Erholung	19	0.14	11
G4	Kleine Plastikbeutel; Gefrierbeutel, Zippsäckc...	Nicht-Lebensmittelverpackungen	19	0.14	7
G149	Papierverpackungen	Nicht-Lebensmittelverpackungen	19	0.14	6
G28	Stifte, Deckel, Druckbleistifte usw.	Persönliche Gegenstände	19	0.14	9
G34	Besteck, Teller und Tabletts	Essen und Trinken	18	0.13	10
G144	Tampons	Abwasser	18	0.13	1
G7	Getränkeflaschen < = 0,5 l	Essen und Trinken	18	0.13	9
G158	Sonstige Gegenstände aus Papier	Nicht-Lebensmittelverpackungen	17	0.12	5
G148	Kartonkisten und Stücke	Nicht-Lebensmittelverpackungen	16	0.12	6
G134	Sonstiges Gummi	nicht klassifiziert	16	0.12	7
G943	Zäune Landwirtschaft, Kunststoff	Landwirtschaft	15	0.11	2

	Objekte	Gruppenname	Gesamt	% Gesamt	fail rate
code					
G170	Holz (verarbeitet)	Landwirtschaft	15	0.11	7
G194	Kabel, Metalldraht oft in Gummi- oder Kunststo...	Infrastruktur	15	0.11	8
G101	Robidog Hundekot-Säcklein, andere Hundekotsäck...	Persönliche Gegenstände	14	0.10	7
G142	Seil, Schnur oder Netze	Freizeit und Erholung	14	0.10	6
G161	Verarbeitetes Holz	Landwirtschaft	13	0.09	5
G59	Monofile Angelschnur (Angeln)	Freizeit und Erholung	13	0.09	7
G939	Kunststoffblumen, Kunststoffpflanzen	Persönliche Gegenstände	12	0.09	5
G167	Streichhölzer oder Feuerwerke	Freizeit und Erholung	12	0.09	2
G124	Kunststoff-oder Schaumstoffprodukte	nicht klassifiziert	11	0.08	5
G146	Papier, Karton	Nicht-Lebensmittelverpackungen	11	0.08	2
G931	(Absperr)band für Absperrungen, Polizei, Baust...	Infrastruktur	11	0.08	4
G115	Schaumstoff <5mm	Mikroplastik (< 5mm)	10	0.07	5
G919	Nägel, Schrauben, Bolzen usw.	Infrastruktur	10	0.07	4
G901	Medizinische Masken, synthetische	Persönliche Gegenstände	10	0.07	6
G933	Taschen, Etuis für Zubehör, Brillen, Elektroni...	Persönliche Gegenstände	10	0.07	5
G65	Eimer	Landwirtschaft	9	0.06	2
G155	Feuerwerkspapierhülsen und -fragmente	Freizeit und Erholung	9	0.06	6
G2	Säcke, Taschen	Nicht-Lebensmittelverpackungen	9	0.06	3

	Objekte	Gruppenname	Gesamt	% Gesamt	fail rate
code					
G921	Keramikfliesen und Bruchstücke	Infrastruktur	9	0.06	6
G918	Sicherheitsnadeln, Büroklammern, kleine Gebrau...	Persönliche Gegenstände	9	0.06	5
G135	Kleidung, Fussbekleidung, Kopfbedeckung, Hands...	Persönliche Gegenstände	9	0.06	5
G20	Laschen und Deckel	Nicht-Lebensmittelverpackungen	8	0.06	5
G176	Konservendosen (Lebensmitteldosen)	Essen und Trinken	8	0.06	4
G133	Kondome einschließlich Verpackungen	Abwasser	8	0.06	5
G917	Blähton	nicht klassifiziert	8	0.06	4
G68	Fiberglas-Fragmente	Infrastruktur	8	0.06	5
G118	Kleine Industrielle Kügelchen <5mm	Mikroplastik (< 5mm)	7	0.05	2
G122	Kunststofffragmente (>1mm)	Mikroplastik (< 5mm)	7	0.05	0
G925	Pakete: Trockenmittel/Feuchtigkeitsabsorber, m...	Nicht-Lebensmittelverpackungen	7	0.05	3
G929	Elektronik und Teile; Sensoren, Headsets usw.	Persönliche Gegenstände	6	0.04	3
G49	Seile > 1cm	Freizeit und Erholung	6	0.04	2
G182	Angeln; Haken, Gewichte, Köder, Senkblei, usw.	Freizeit und Erholung	6	0.04	3
G38	Abdeckungen; Kunststoffverpackungen, Folien zu...	nicht klassifiziert	6	0.04	3
G938	Zahnstocher, Zahnseide Kunststoff	Essen und Trinken	6	0.04	4
G126	Bälle	Freizeit und Erholung	6	0.04	2
G36	Säcke aus strapazierfähigem	Landwirtschaft	5	0.04	2

	Objekte	Gruppenname	Gesamt	% Gesamt	fail rate
code					
	Kunststoff für 25 ...				
G64	Kotflügel	nicht klassifiziert	5	0.04	2
G141	Teppiche	nicht klassifiziert	5	0.04	2
G29	Kämme, Bürsten und Sonnenbrillen	Persönliche Gegenstände	5	0.04	2
G71	Schuhe Sandalen	Persönliche Gegenstände	4	0.03	2
G104	Kunststofffragmente abgerundet / rundlich <5m...	Mikroplastik (< 5mm)	4	0.03	2
G926	Kaugummi, enthält oft Kunststoffe	Essen und Trinken	4	0.03	2
G103	Kunststofffragmente rund <5mm	Mikroplastik (< 5mm)	4	0.03	2
G8	Getränkeflaschen > 0,5L	Essen und Trinken	4	0.03	2
G913	Schnuller	Persönliche Gegenstände	4	0.03	2
G930	Schaumstoff-Ohrstöpsel	Persönliche Gegenstände	4	0.03	2
G119	Folienartiger Kunststoff (>1mm)	Mikroplastik (< 5mm)	4	0.03	1
G11	Kosmetika für den Strand, z.B. Sonnencreme	Freizeit und Erholung	4	0.03	2
G157	Papier	Nicht-Lebensmittelverpackungen	4	0.03	2
G197	sonstiges Metall	Infrastruktur	4	0.03	2
G37	Netzbeutel, Netztasche, Netzsäcke	Persönliche Gegenstände	4	0.03	2
G145	Andere Textilien	Persönliche Gegenstände	4	0.03	2
G6	Flaschen und Behälter aus Kunststoff, nicht fü...	Nicht-Lebensmittelverpackungen	3	0.02	0
G151	Tetrapack, Kartons	Essen und Trinken	3	0.02	1
G945	Rasierklingen	Persönliche Gegenstände	3	0.02	2
G99	Spritzen - Nadeln	Persönliche Gegenstände	3	0.02	2

	Objekte	Gruppenname	Gesamt	% Gesamt	fail rate
code					
G143	Segel und Segeltuch	Freizeit und Erholung	3	0.02	2
G12	Kosmetika, Behälter für Körperpflegeprodukte, ...	Persönliche Gegenstände	3	0.02	2
G102	Flip-Flops	Persönliche Gegenstände	2	0.01	1
G154	Zeitungen oder Zeitschriften	Persönliche Gegenstände	2	0.01	1
G129	Schläuche und Gummiplatten	nicht klassifiziert	2	0.01	1
G181	Geschirr aus Metall, Tassen, Besteck usw.	Essen und Trinken	2	0.01	1
G188	Andere Kanister/Behälter < 4 L	Infrastruktur	2	0.01	0
G195	Batterien (Haushalt)	Persönliche Gegenstände	2	0.01	1
G202	Glühbirnen	nicht klassifiziert	2	0.01	1
G111	Kugelförmige Pellets < 5mm	Mikroplastik (< 5mm)	2	0.01	1
G174	Aerosolspraydosen	Infrastruktur	2	0.01	1
G136	Schuhe	Persönliche Gegenstände	2	0.01	1
G43	Sicherheitsetiketten, Siegel für Fischerei ode...	Freizeit und Erholung	2	0.01	1
G915	Reflektoren, Mobilitätsartikel aus Kunststoff	Persönliche Gegenstände	2	0.01	0
G916	Bleistifte und Bruchstücke	Persönliche Gegenstände	2	0.01	1
G39	Handschuhe	Persönliche Gegenstände	2	0.01	0
G52	Netze und Teilstücke	Freizeit und Erholung	2	0.01	1
G53	Netze und Teilstücke < 50cm	Freizeit und Erholung	1	0.01	0
G17	Kartuschen von Kartuschen-spritzpistolen	Infrastruktur	1	0.01	0
G902	Medizinische Masken, Stoff	Persönliche Gegenstände	1	0.01	0

	Objekte	Gruppenname	Gesamt	% Gesamt	fail rate
code					
G900	Handschuhe Latex, persönliche Schutzausrüstung	Persönliche Gegenstände	1	0.01	0
G13	Flaschen, Behälter, Fässer zum Transportieren ...	Landwirtschaft	1	0.01	0
G150	Milchkartons, Tetrapack	Essen und Trinken	1	0.01	0
G172	Sonstiges Holz > 50 cm	Landwirtschaft	1	0.01	0
G97	Behälter von Toilettenerfrischer	Abwasser	1	0.01	0
G60	Lichtstab, Knicklicht, Glow-sticks	Freizeit und Erholung	1	0.01	0
G84	CD oder CD-Hülle	Persönliche Gegenstände	1	0.01	0
G61	Sonstiges Angelzubehör	Freizeit und Erholung	1	0.01	0
G171	Sonstiges Holz < 50cm	Landwirtschaft	1	0.01	0
G14	Flaschen für Motoröl (Motorölflaschen)	nicht klassifiziert	1	0.01	0
G94	Tischtuch	Freizeit und Erholung	1	0.01	0
G903	Behälter und Packungen für Handdesinfektionsmi...	Persönliche Gegenstände	1	0.01	0
G179	Einweg Grill	Essen und Trinken	1	0.01	0
G934	Sandsäcke, Kunststoff für Hochwasser- und Eros...	Landwirtschaft	1	0.01	0
G128	Reifen und Antriebsriemen	nicht klassifiziert	1	0.01	0
G138	Schuhe und Sandalen	Persönliche Gegenstände	1	0.01	0
G5	Plastiksäcke/ Plastiktüten	Nicht-Lebensmittelverpackungen	1	0.01	0
G190	Farbtöpfe, Farbbüchsen, (Farbeimer)	Infrastruktur	1	0.01	0
G41	Handschuhe	Landwirtschaft	1	0.01	0

	Objekte	Gruppenname	Gesamt	% Gesamt	fail rate
code					
	Industriell/Professionell				
G107	Zylindrische Pellets <5mm	Mikroplastik (< 5mm)	1	0.01	0
G214	Öl/Teer	Infrastruktur	1	0.01	0
G114	Folien <5mm	Mikroplastik (< 5mm)	1	0.01	0
G40	Handschuhe Haushalt/Garten	Persönliche Gegenstände	1	0.01	0

zurück

2. Alpen und der Jura

weiter

4. Bielersee