

I
dentification
Q
uantification
A
nalysis
A
nthropogenic
S
wiss
L
itter

- [Identification, quantification and analysis of observable anthropogenic litter along Swiss lake systems](#)

Zusammenfassung

Powered by [Jupyter Book](#)

- [repository](#)
- [open issue](#)

Contents

Zusammenfassung

- [1. Seen und Fliessgewässer](#)
- [2. Alpen und der Jura](#)
- [3. Aare](#)

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

Contents

Zusammenfassung

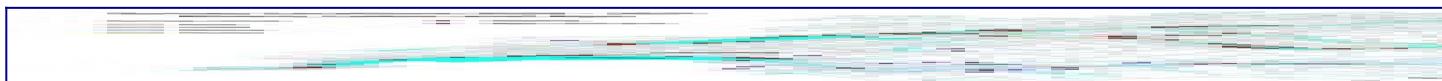
- [1. Seen und Fliessgewässer](#)
- [2. Alpen und der Jura](#)
- [3. Aare](#)

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

Edition 1 ver=1

English

Karte der Erhebungsorte März 2020 - August 2021



Biel/Bienne 31/12/2021

Im Auftrag des Bundesamtes für Umwelt (BAFU)
Sur mandat de l'Office fédéral de l'environnement (OFEV)
Su mandato dell'Ufficio federale dell'ambiente (UFAM)
Commissioned by the Federal Office for the Environment (FOEN)

Impressum#

Auftraggeber: Bundesamt für Umwelt (BAFU), Abt. Abfall und Rohstoffe, CH-3003 Bern Das BAFU ist ein Amt des Eidg. Departements für Umwelt, Verkehr, Energie und Kommunikation (UVEK).

Auftragnehmer: Hammerdirt Brüggstrasse 39 2503 Biel/Bienne Switzerland +41 76 699 06 16
info@hammerdirt.ch

Die Autoren: Roger Erismann, Shannon Erismann

Begleitung BAFU: Rita Barros, Amanda Finger, Samuel Anrig

Hinweis: Diese Bericht wurde im Auftrag des Bundesamtes für Umwelt (BAFU) verfasst. Für den Inhalt ist allein der Auftragnehmer verantwortlich.

Projektteam

Projektverantwortliche: Helen Kurukulasuriya, Martin Benvasser, Débora Camaro, Rachel Aronoff, Thor Erismann, Bettina Siegenthaler, Théo Gürsoy, Adrien Bonny, Gaetan Buser, Louise Schreyers, Andreas Gauer, Shannon Erismann, Roger Erismann, EPFL *Studierende der Fachrichtung Abfallwirtschaft*

Teilnehmende Organisationen: Association pour la Sauvegarde du Léman, Precious Plastic Léman, Geneva International School, Students of Solid Waste Engineering: EPFL, Stiftung Summit, Hackuarium, hammerdirt

Unterstützungsgruppe: Christian Ludwig (EPFL/PSI), Montserrat Filella (UNIGE), Romain Tramoy (LEESU), Gary Hare (HDCA), Taoufik Nouri (FHNW)

Übersetzung: Helen Kurukulasuriya

Besonderer Dank: Hubert Heldner, Kurt Chanton, Marcel Regamey, Bhavish Patel, Olivier Kressmann,

```
# -*- 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
# for date and month formats in french or german
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 as mpl
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.sr_ut as sut
import resources.chart_kwarg as ck

from IPython.display import Markdown as md
from myst_nb import glue

# set the locale to the language desired
# the locale is set back to to original at the end of the script
# set the locale to the language desired
# the locale is set back to to original at the end of the script
loc = locale.getlocale()
lang = "de_CH.utf8"
locale.setlocale(locale.LC_ALL, lang)

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

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

# set some parameters:
start_date = "2020-03-01"
end_date ="2021-05-31"
ge_start = "01.03.2020"
ge_end ="31.05.2021"
start_end = [start_date, end_date]
a_fail_rate = 50
unit_label = "p/100 m"
reporting_unit = 100
a_color = "saddlebrown"

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

# colors for the individual survey areas
bassin_palette = {"rhone":"dimgray", "aare":"salmon", "linth":"tan",
"ticino":"steelblue", "reuss":"purple"}

# set the maps
bassin_map = "resources/maps/lakes_rivers_map.jpeg"

# define the feature level and components
this_level = "river_bassin"
comps = ["linth", "rhone", "aare", "ticino"]

# proper labels for charts and tables
comp_labels = {"linth":"Linth", "rhone":"Rhône", "aare":"Aare", "ticino":"Ticino",
"reuss":"Reuss"}
top_name = ["Alle Erhebungsgebiete"]

```

```

# explanatory variables:
luse_exp = list(sut.luse_ge.values())

# common aggregations
agg_pcs_quantity = {"unit_label": "sum", "quantity": "sum"}
agg_pcs_median = {"unit_label": "median", "quantity": "sum"}

# aggregation of dimensional data
agg_dims = {"total_w": "sum", "mac_plast_w": "sum", "area": "sum", "length": "sum"}

# columns needed
use_these_cols = ["loc_date",
                   "% buildings",
                   "% to trans",
                   "% recreation",
                   "% ag",
                   "% woods",
                   "population",
                   this_level,
                   "water_name_slug",
                   "streets km",
                   "intersects",
                   "length",
                   "groupname",
                   "code"
                  ]
]

# get the data:
dfBeaches = pd.read_csv("resources/beaches_with_land_use_rates.csv")
dfCodes = pd.read_csv("resources/codes_with_group_names_2015.csv")
dfDims = pd.read_csv("resources/corrected_dims.csv")

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

# make any adjustments to code definitions here:
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)

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)
```

```
def thousandsSeparator(aninteger, lang):  
  
    astring = "{:,}").format(aninteger)  
  
    if lang == "DE":  
        astring = astring.replace(", ", " ")  
  
    return astring
```

```
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
```

1. Seen und Fließgewässer#

Karte der Erhebungsorte IQAASL März 2020 - Mai 2021.

All survey areas

Sample locations

Median p/100m

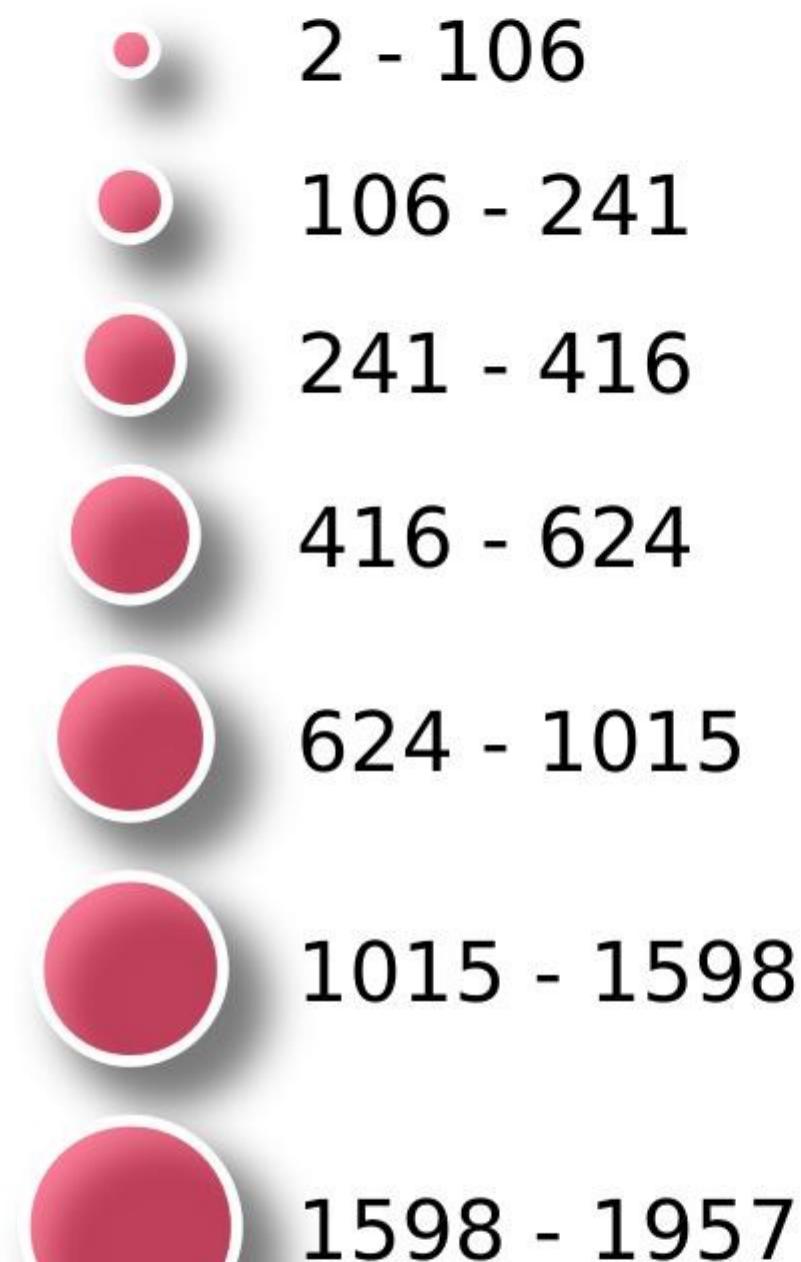


Abb. 1.1 #

Abbildung 1.1: Die Erhebungsorte sind für die Analyse nach Erhebungsgebiet gruppiert. Die Grösse der Markierung gibt einen Hinweis auf die Anzahl Abfallobjekte, die gefunden wurden.

1.1. Landnutzungsprofil#

Die Erhebungsgebiete sind nach Flusseinzugsgebieten gruppiert. In diesem Bericht werden mehrere Einzugsgebiete zusammengefasst, um regionale Trends zu analysieren:

- Aare: Emme, Aare
- Linth: Reuss, Linth, Limmat
- Rhône: Rhône
- Tessin/Ceresio: Tessin, Lugarnersee, Lago Maggiore

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.

Weitere Informationen gibt es im Kapitel *Landnutzungsprofil*

Verteilung der Landnutzungsmerkmale

```
# this is the data before the expanded fragmented plastics and foams are aggregated  
to Gfrags and Gfoams  
before_agg = pd.read_csv("resources/checked_before_agg_sdata_eos_2020_21.csv")  
before_agg["loc_date"] = list(zip(before_agg.location, before_agg["date"]))  
before_agg.rename(columns={"p/100m":unit_label}, inplace=True)  
  
# this is the aggregated survey data that is being used  
fd = pd.read_csv(F"resources/checked_sdata_eos_2020_21.csv")  
  
fd["loc_date"] = list(zip(fd.location, fd["date"]))  
  
# apply local date configuration  
fd["date"] = pd.to_datetime(fd.date, format=d)  
  
fd = fd[(fd["date"] >= start_date)&(fd["date"] <= end_date)].copy()  
  
fd["groupname"] = fd["groupname"].map(lambda x: sut.group_names_de[x])
```

```

fd.rename(columns={"p/100m":unit_label}, inplace=True)
fd.rename(columns=sut.luse_ge, inplace=True)

# 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)

dt_nw = fd.groupby(["loc_date", "river_bassin", *luse_exp],
as_index=False).agg({unit_label:"sum"})

sns.set_style("whitegrid")

fig, axs = plt.subplots(2, 3, figsize=(9,9), sharey="row")

for i, n in enumerate(luse_exp):
    # get the dist for each survey area here
    r = i%2
    c = i%3
    ax=axs[r,c]
    for element in comps:
        the_data = ECDF(dt_nw[dt_nw[this_level] == element][n].values)
        sns.lineplot(x=the_data.x, y=the_data.y, ax=ax, label=comp_labels[element],
color=bassin_palette[element])

    # get the dist for all here
    a_all_surveys = ECDF(dt_nw[n].values)
    sns.lineplot(x=a_all_surveys.x, y=a_all_surveys.y, ax=ax, label="Alle
Erhebungsgebiete", color="magenta", linewidth=2.5)

    # get the median from the data
    the_median = np.median(dt_nw[n].values)

    # plot the median and drop horizontal 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)

    # format the % of total on the xaxis:
    if i <= 3:
        if c == 0:
            ax.set_ylabel("% der Erhebungen", **ck.xlab_k)
            ax.xaxis.set_major_formatter(ticker.PercentFormatter(1.0, 0, "%"))
        else:
            pass

    ax.set_xlabel(n, **ck.xlab_k)
    handles, labels = ax.get_legend_handles_labels()
    ax.get_legend().remove()
    ax.set_title(F"Median: {(round(the_median, 2))}", fontsize=12, loc="left")

plt.tight_layout()
plt.subplots_adjust(top=.88, hspace=.35)
plt.suptitle("Landnutzung im Umkreis von 1500 m um den Erhebungsort", ha="center",
y=1, fontsize=16)

```

```

fig.legend(handles, labels, bbox_to_anchor=( .53, .94), loc="center", ncol=6)
glue("eosluse_de", fig, display=False)
plt.close()

```

Landnutzung im Umkreis von 1500 m um den Erhebung

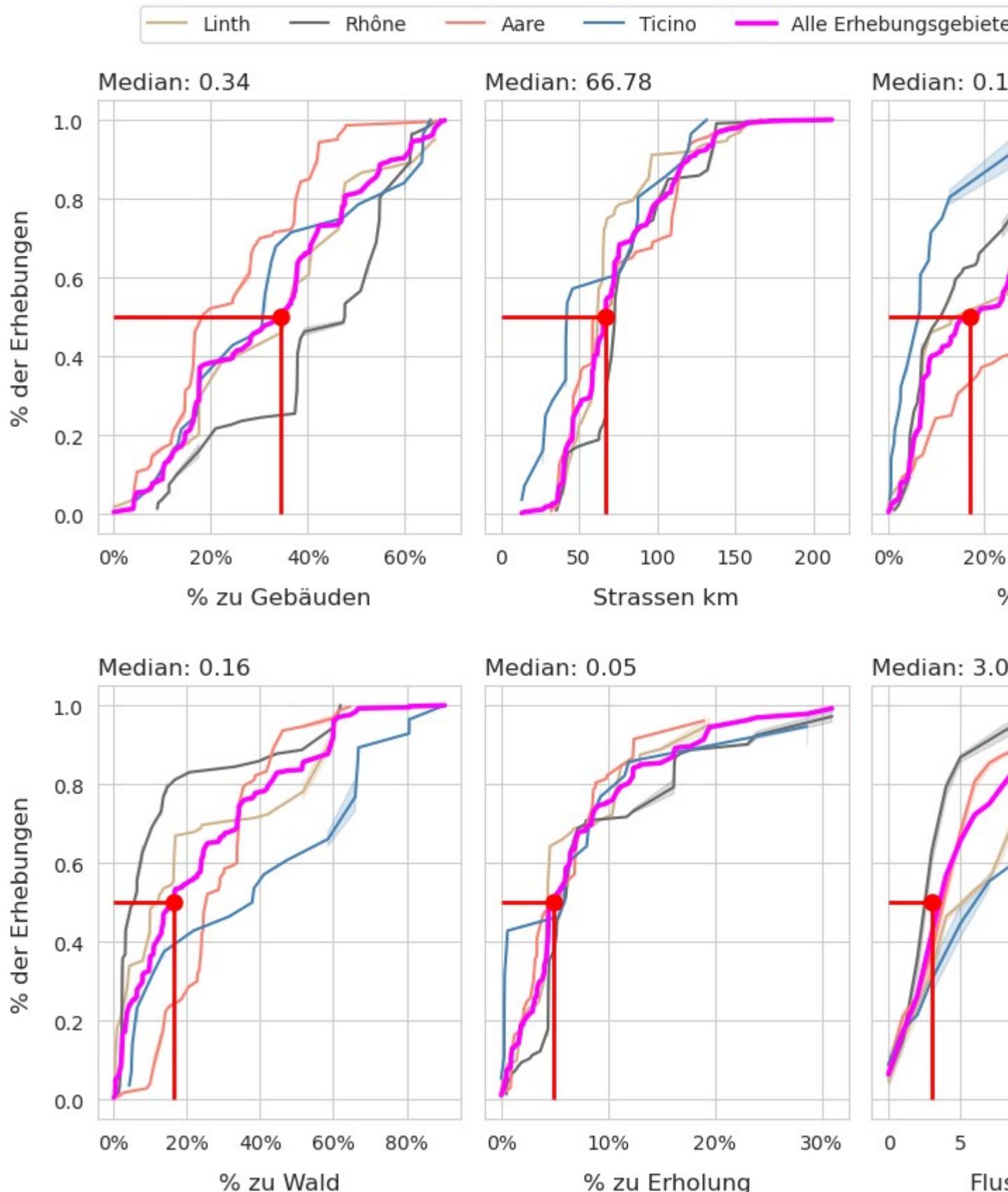


Abb. 1.2 #

Abbildung 1.2: Die Erhebungen in den Gebieten Rhône und Linth wiesen mit 47 % bzw. 40 % im Median den grössten Anteil an bebauter Fläche (Gebäude) und mit 5 % bzw. 8 % den geringsten Anteil an Wald auf. Im Erhebungsgebiet Aare war der Medianwert der bebauten Fläche (Gebäude) mit 16 % am niedrigsten und der Anteil der landwirtschaftlich genutzten Fläche mit 30 % am höchsten. Bei den Flächen, die den Aktivitäten im Freien zugeordnet werden, handelt es sich um Sportplätze, öffentliche Strände und andere öffentliche Versammlungsorte.

1.1.1. Gesamtergebnisse nach Erhebungsgebiet#

```
# aggregate the dimensional data
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)

dt_all = fd.groupby(["loc_date", "location", this_level, "city", "date"],
                    as_index=False).agg({unit_label:"sum", "quantity":"sum"})

for name in dims_table.index:
    dims_table.loc[name, "samples"] = fd[fd[this_level] == name].loc_date.nunique()
    dims_table.loc[name, "quantity"] = fd[fd[this_level] == name].quantity.sum()

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

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

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

dims_table.rename(columns={"samples": "Erhebungen", "quantity": "Objekte",
                           "total_w": "Gesamt-kg", "mac_plast_w": "kg Plastik", "area": "m\u00b2", "length": "Meter"}, inplace=True)

# format kilos and text strings
dims_table["kg Plastik"] = dims_table["kg Plastik"]/1000
dims_table[[ "m\u00b2", "Meter", "Erhebungen", "Objekte"]] = dims_table[[ "m\u00b2", "Meter",
                           "Erhebungen", "Objekte]].applymap(lambda x: thousandsSeparator(int(x), "DE"))
dims_table[[ "kg Plastik", "Gesamt-kg"]] = dims_table[[ "kg Plastik", "Gesamt-kg]].applymap(lambda x: "{:.2f}".format(x))

data = dims_table.reset_index()

fig, axs = plt.subplots(figsize=(len(data.columns)*1.8, len(data)*.8))
sut.hide_spines_ticks_grids(axs)

table_one = sut.make_a_table(axs, data.values, colLabels=data.columns,
                            colWidths=[.22, *[.13]*6], a_color=a_color)
table_one.get_celld()[(0,0)].get_text().set_text(" ")
table_one.set_fontsize(12)

plt.tight_layout()
```

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

	Gesamt-kg	kg Plastik	m ²
Alle Erhebungsgebiete	305.51	94.21	96 616
Rhône	151.31	45.97	25 986
Aare	71.98	31.45	37 017
Linth	35.96	12.77	25 637
Ticino	46.26	4.03	7 974

Abb. 1.3 #

Abbildung 1.3: Summen der Ergebnisse für alle Erhebungsgebiete. Im Gebiet Aare ist die Anzahl Erhebungen am grössten. Gleichzeitig ist dort die Anzahl gesammelter Objekte pro Fläche am geringsten.

1.2. Erhebungsergebnisse für alle Objekte#

Verteilung der Erhebungsergebnisse. Die Werte werden als Anzahl der identifizierten Abfallobjekte pro 100 Meter (p/100 m) dargestellt.

```
# the surveys to chart
dt_all["date"] = pd.to_datetime(dt_all["date"], format=g)
fd_n_samps = dt_all.loc_date.nunique()
fd_dindex = dt_all.set_index("date")

# monthly median survey total
monthly_plot = fd_dindex[unit_label].resample("M").median()

# scale the chart as needed to accomodate for extreme values
y_lim = 97
y_limit = np.percentile(dt_all[unit_label], y_lim)

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

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

rate = "Monthly"
```

```

line_label = "monatlicher Medianwert"

ax = axs[0]
sns.scatterplot(data=fd_dindex, x="date", y=unit_label, hue=this_level,
palette=bassin_palette, alpha=0.8, ax=ax)

sns.lineplot(data=monthly_plot, x=monthly_plot.index, y=monthly_plot,
color="magenta", label=line_label, ax=ax)

ax.set_ylim(0,y_limit)

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

ax.xaxis.set_major_formatter(months_fmt)
ax.get_legend().remove()

axtwo = axs[1]
lake_dts = fd.groupby([this_level, "loc_date"], as_index=False)[unit_label].sum()

for i, a_name in enumerate(fd[this_level].unique()):
    label=comp_labels[a_name]

    a_ecdf = ECDF(lake_dts[lake_dts[this_level] == a_name][unit_label].values)

    sns.lineplot(x=a_ecdf.x, y=a_ecdf.y, color=bassin_palette[a_name], ax=axtwo,
label=label)

r_bassin = ECDF(dt_all[unit_label].values)

sns.lineplot(x=r_bassin.x, y=r_bassin.y, color="magenta", label=top_name[0],
linewidth=2, ax=axtwo)

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

plt.tight_layout()

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

plt.close()

```

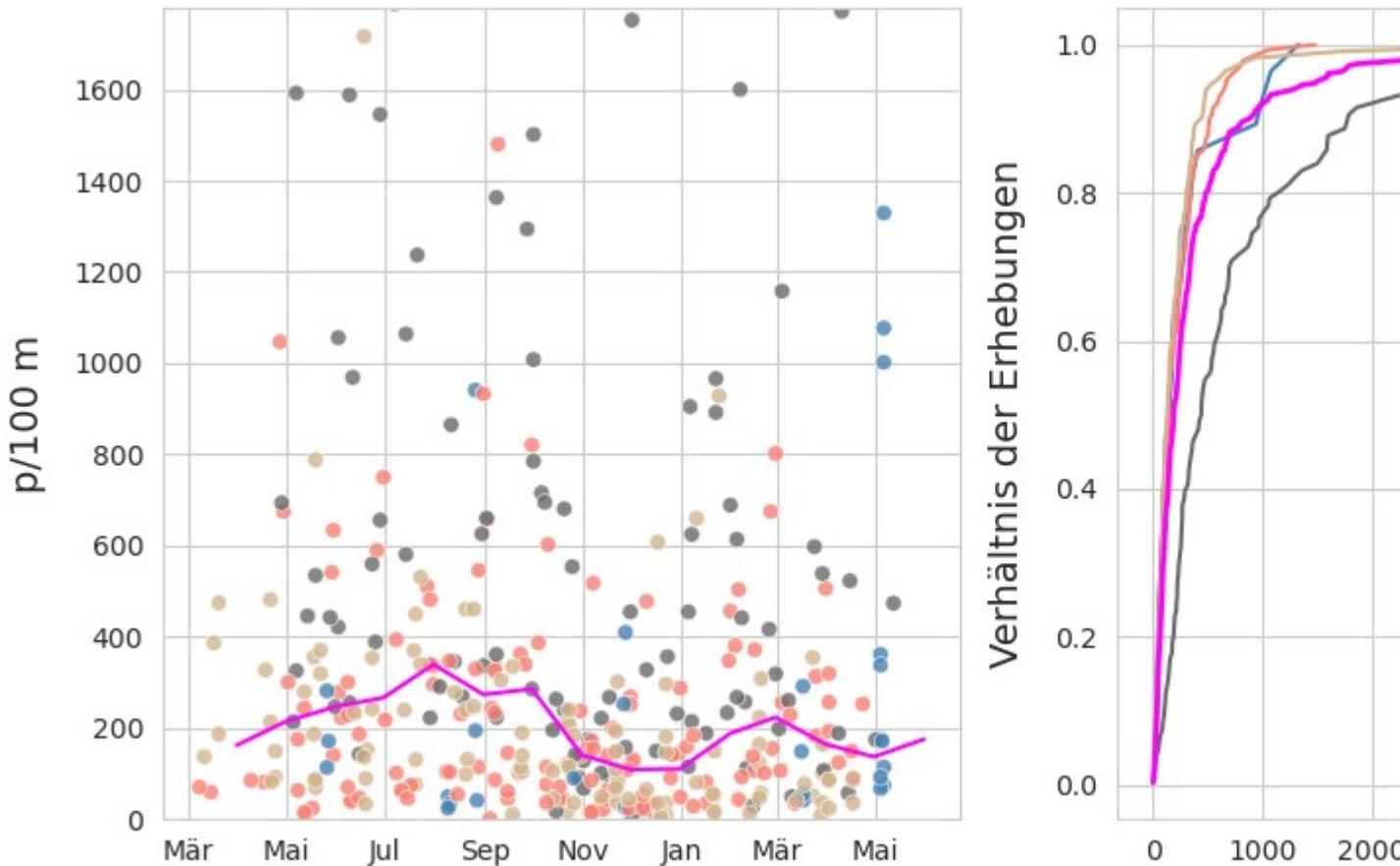


Abb. 1.4 #

Abbildung 1.4: Links: Alle Erhebungen zwischen März 2020 bis Mai 2021, gruppiert nach Erhebungsgebiet und aggregiert zum monatlichen Median. Werte über 1778 p/100 m sind nicht dargestellt. Rechts: Die empirische Verteilungsfunktion der Gesamtwerte der Erhebungen.

Abfall nach Materialarten im Überblick

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

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

# 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])), "DE") for x in csx.index]

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

# fd_mat_totals = sut.the_ratio_object_to_total(code_totals)
# fd_mat_totals = sut.fmt_pct_of_total(fd_mat_totals)
# fd_mat_totals = sut.make_string_format(fd_mat_totals)
# 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)
```

```

# apply 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])), "DE"), x[2]) for x in fd_mat_t]

# # apply 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

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

# summary table
# names for the table columns
a_col = ["All survey areas", "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(" ")
table_three.set_fontsize(12)

plt.tight_layout()
plt.subplots_adjust(wspace=0.2)
glue("summarymaterial_de", fig, display=False)
plt.close()

```

	total	Gesamt	% G
Anzahl Proben	386	Plastik	47 143
Durchschnitt p/100 m	395	Glas	2 919
Standardfehler	706	Metall	1 874
min p/100 m	2	Papier	1 527
25%	82	Holz	406
50%	189	Gummi	390
75%	387	Stoff	343
max p/100 m	6 617	Chemikalien	140
Abfallobjekte total	54 744	Unbekannt	2

Abb. 1.5 #

Abbildung 1.5: **Links:** Zusammenfassung der Resultate für alle Erhebungsgebiete. **Rechts:** Materialarten in Stückzahlen und als Prozentsatz an der Gesamtmenge (Stückzahl) für alle Erhebungsgebiete.

1.3. Die am häufigsten gefundenen Gegenstände#

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()
```

```

# 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(int(x),
"DE"))
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"{np.ceil(x)}")

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

# final table data
all_survey_areas = m_common[cols_to_use.keys()].values

fig, axs = plt.subplots(figsize=(9, len(m_common)*.6))

sut.hide_spines_ticks_grids(axs)

table_four = sut.make_a_table(axs, all_survey_areas,
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("mcommoneos_de", fig, display=False)
plt.close()

```

	Gesamt	% Gesamt	fail-
Zigarettenfilter	8 485	15.5%	87%
Fragmentierte Kunststoffstücke	7 400	13.52%	80%
Expandiertes Polystyrol	5 563	10.16%	68%
Snack-Verpackungen	3 325	6.07%	85%
Industriefolie (Kunststoff)	2 534	4.63%	69%
Getränke Glasflasche, Stücke	2 136	3.9%	65%
Industriepellets (Nurdles)	1 968	3.59%	30%
Schaumstoffverpackungen/Isolierung	1 702	3.11%	53%
Wattestäbchen / Tupfer	1 406	2.57%	50%
Styropor < 5mm	1 209	2.21%	25%
Kunststoff-Bauabfälle	992	1.81%	52%
Kronkorken, Lasche von Dose/Ausfreisslachen	700	1.28%	52%

Abb. 1.6 #

Abbildung 1.6: Die häufigsten Objekte für alle Erhebungsgebiete. Die Häufigkeitsrate gibt an, wie oft ein Objekt in den Erhebungen im Verhältnis zu allen Erhebungen identifiziert wurde: Zigarettenfilter beispielsweise wurden in 87 % der Erhebungen gefunden. Zusammengenommen machen die häufigsten Objekte 68 % aller gefundenen Objekte aus.

Häufigste Objekte im Median p/100 m nach Erhebungsgebiet

```
# aggregated survey totals for the most common codes for all the survey areas
# aggregate the code totals by survey and then get the median value per survey
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 description 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 hierachal column index
m_c_p.columns = m_c_p.columns.get_level_values(1)

# the aggregated totals of all the data
a_s_a = fd(fd.code.isin(m_common.index)].groupby(["water_name_slug", "loc_date",
"code"], as_index=False).agg(agg_pcs_quantity)
a_s_a_cols = sut.aggregate_to_code(a_s_a, code_description_map, name=top_name[0],
unit_label=unit_label)

ad_t_ten = pd.concat([m_c_p, a_s_a_cols], axis=1).sort_values(by=top_name[0],
ascending=False)

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

sns.heatmap(ad_t_ten, 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="both")

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

glue("mcommonpcs_de", fig, display=False)
plt.close()

```

	Zigarettenfilter	11.0	23.0	41.5	15.0
	Fragmentierte Kunststoffstücke	18.5	10.5	48.0	9.5
	Snack-Verpackungen	8.0	6.0	19.0	4.0
	Expandiertes Polystyrol	4.0	3.0	17.5	6.0
	Industriefolie (Kunststoff)	5.0	2.0	9.0	3.5
	Getränke Glasflasche, Stücke	3.0	4.0	2.0	8.0
Kronkorken, Lasche von Dose/Ausfreisslachen	0.0	1.0	3.0	1.0	0.0
Kunststoff-Bauabfälle	0.0	0.0	5.5	2.5	0.0
Schaumstoffverpackungen/Isolierung	0.0	0.0	7.0	3.0	0.0
Wattestäbchen / Tupfer	0.0	0.0	10.5	0.0	0.0
Industriepellets (Nurdles)	0.0	0.0	0.0	0.0	0.0
Styropor < 5mm	0.0	0.0	0.0	0.0	0.0
	Aare	Linth	Rhône	Ticino	

Abb. 1.7 #

Abbildung 1.7: Der Median p/100 m der häufigsten Objekte für alle Erhebungsgebiete. Die Werte für die jeweiligen Erhebungsgebiete unterscheiden sich deutlich.

Häufigste Objekte im Monatsdurchschnitt

```
# 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["date"] = pd.to_datetime(m_common_m["date"], format="%Y-%m-%d")
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=sut.months_de
# 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=(9,7))

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

# the monhtly survey average for all objects and locations
monthly_fd = dt_all.copy()
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=a_color, alpha=0.2, linewidth=1,
edgecolor="teal", width=1, label="Monthly survey average")

# 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 = [months[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,"Monatsdurchschnitt")
handles = [handles[0], *handles[1:][::-1]]

plt.legend(handles=handles, labels=new_labels, bbox_to_anchor=(.5, -.05), loc="upper
center", ncol=2, fontsize=14)
glue("monthlyeos_de", fig, display=False)
plt.close()

```

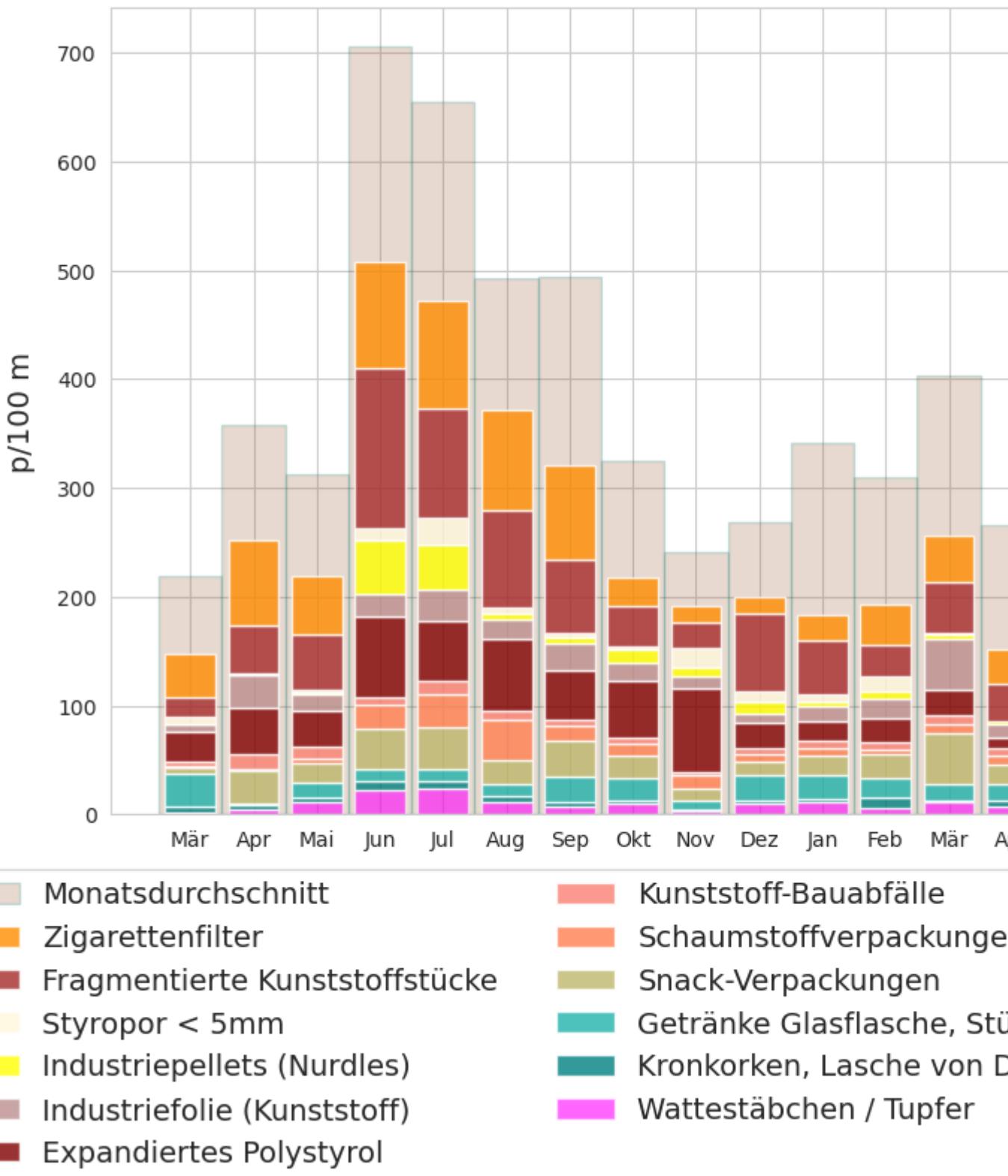


Abb. 1.8 #

Abbildung 1.8: Monatliches Durchschnittsergebnis der Erhebungen als p/100 m der häufigsten Objekte für alle Erhebungsgebiete.

1.4. Erhebungsergebnisse und Landnutzung#

Die Erhebungsergebnisse in Bezug auf die Landnutzung auf nationaler Ebene werden in den Kapiteln Landnutzungsprofil und Geteilte Verantwortung ausführlich erörtert.

1.5. 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 1.8.3) befindet sich die vollständige Liste der identifizierten Objekte, einschliesslich Beschreibungen und Gruppenklassifizierung. Das Kapitel Codegruppen beschreibt jede Codegruppe im Detail und bietet eine umfassende Liste aller Objekte in einer Gruppe.

Der Nutzungszweck oder die Beschreibung der identifizierten Objekte in % der Gesamtfläche der Erhebung.

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

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

# quantity per water survey area
cg_tq = cg_t.groupby(this_level).quantity.sum()

# 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")

data_table[top_name[0]] = sut.aggregate_to_group_name(fd, unit_label=unit_label,
column="groupname", name=top_name[0], val="pt")
data_table.rename(columns={x:comp_labels[x] for x in data_table.columns[:-1]}, inplace=True)

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

axone = ax
sns.heatmap(data_table, ax=axone, cmap=cmap2, annot=True, annot_kws={"fontsize":12},
cbar=False, fmt=".0%", 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("utility_de", fig, display=False)

plt.close()
```

	Abwasser	2%	6%	2%	5%
Essen und Trinken	19%	22%	18%	21%	19%
Freizeit und Erholung	5%	5%	4%	2%	4%
Infrastruktur	13%	15%	22%	16%	18%
Landwirtschaft	7%	6%	5%	4%	6%
Mikroplastik (< 5mm)	5%	3%	10%	10%	8%
Nicht-Lebensmittelverpackungen	7%	7%	4%	6%	5%
Persönliche Gegenstände	3%	4%	2%	3%	3%
Plastikfragmente	14%	10%	15%	9%	14%
Tabakwaren	20%	24%	12%	26%	17%
nicht klassifiziert	1%	1%	1%	0%	1%
	Aare	Linth	Rhône	Ticino	Alle Erhebungsgebiete

Abb. 1.9 #

Abbildung 1.9: Prozentuale Verteilung der identifizierten Objekte nach Verwendungszweck.

```
# median p/50m solve cg_t for unit_label
data_table = cg_t.pivot(columns=this_level, index="groupname", values=unit_label)

# column for all the data
a = fd.groupby(["groupname", "loc_date"], as_index=False)[unit_label].sum()
a[top_name[0]] = a[unit_label]
a = a.groupby("groupname")[top_name[0]].median()

# stick that together
data = pd.concat([data_table, a], axis=1)

# format for display
data.rename(columns={x:comp_labels[x] for x in data.columns[:-1]}, 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("utility2_de", fig, display=False)

plt.close()
```

	Abwasser	0.5	19	2	3
Essen und Trinken	27	30.5	76	29	37
Freizeit und Erholung	6	4	16.5	3.5	6
Infrastruktur	15	12.5	55.5	22.5	20
Landwirtschaft	6	3	14	6	7
Mikroplastik (< 5mm)	1	0	11.5	0	1
Nicht-Lebensmittelverpackungen	7.5	10	13	5.5	9
Persönliche Gegenstände	4	4	10	6.5	6
Plastikfragmente	18.5	10.5	48	9.5	18
Tabakwaren	15	26.5	50	18	25
nicht klassifiziert	0	0	2	0	0
	Aare	Linth	Rhône	Ticino	Alle Erhebungsgebiete

Abb. 1.10 #

Abbildung 1.10: Das Erhebungsgebiet Rhône weist die höchsten Medianwerte für die jeweiligen Verwendungszwecke auf. Allerdings ist der prozentuale Anteil von Objekten, die mit Tabakwaren, Essen und Trinken zu tun haben, geringer als der von Objekten, die mit der Infrastruktur zu tun haben.

1.6. Fleisgewässer#

```
rivers = fd[fd.w_t == "r"].copy()
r_smps = rivers.groupby(["loc_date", "date", "location", "water_name_slug"],
as_index=False).agg(agg_pcs_quantity)
l_smps = fd[fd.w_t == "l"].groupby(["loc_date", "date", "location", "water_name_slug"],
as_index=False).agg(agg_pcs_quantity)

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 =[ (x, thousandsSeparator(int(csx[x]), "DE")) for x in csx.index]

# 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_name[0]}"

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

ax.set_xlim(-10,y_limit)

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

ax.xaxis.set_major_locator(months_loc)
ax.xaxis.set_major_formatter(months_fmt)

a_col = [top_name[0], "total"]

axone = fig.add_subplot(aspec[:, 6:])
sut.hide_spines_ticks_grids(axone)

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

```

table_five.set_fontsize(12)

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

plt.close()

```

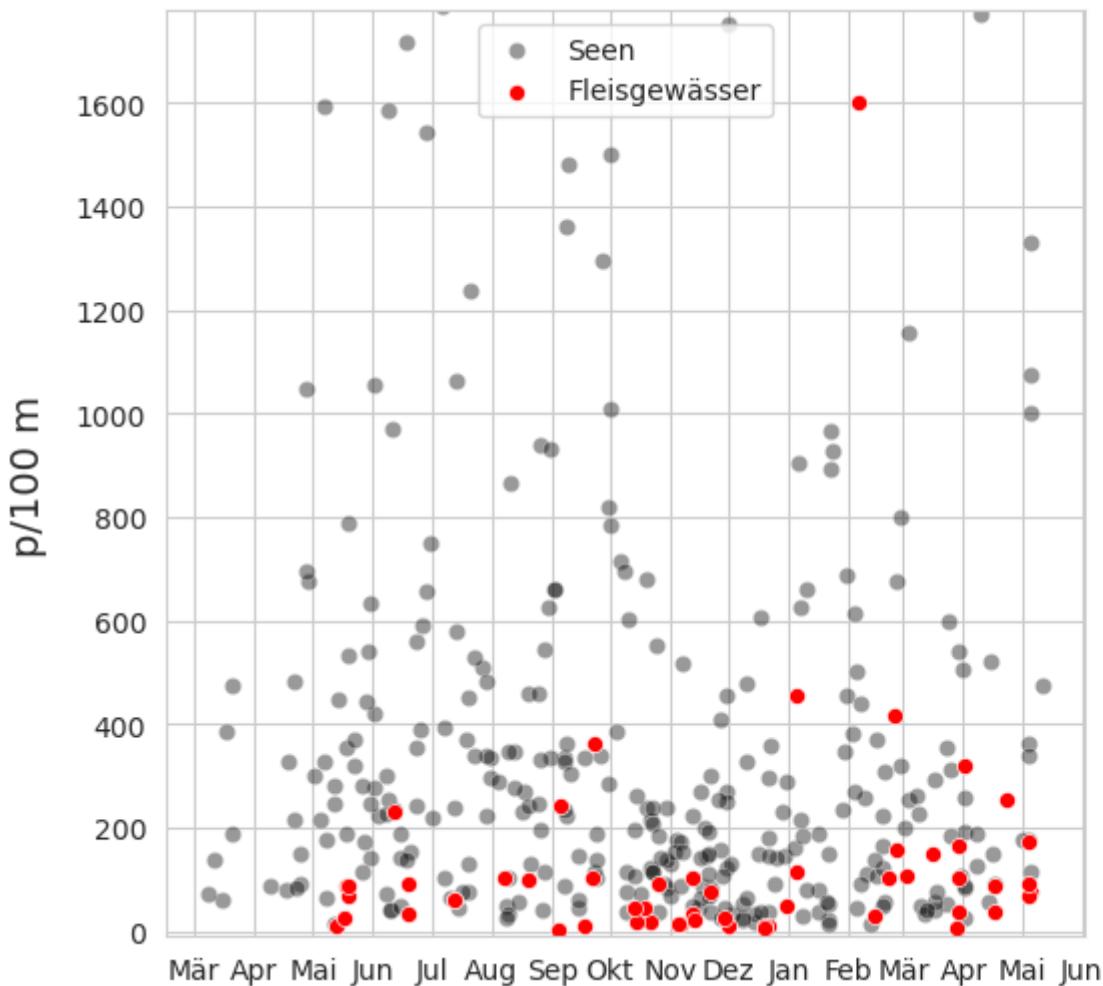


Abb. 1.11 #

Abbildung 1.11: Links: Gesamtergebnisse der Erhebungen an Fliessgewässern für alle Erhebungsgebiete von März 2020 bis Mai 2021, n = 55. Werte über 1779 p/100 m sind nicht dargestellt. Rechts: Zusammenfassende Daten zu Fliessgewässern.

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_smpls.loc_date.nunique()*100).round(2)

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

# most common
r_byfail = r_codes[r_codes["fail rate"] >= a_fail_rate]
r_common = list(set(r_byq) | set(r_byfail))

r_mc = pd.concat([r_byfail, r_byq]).drop_duplicates()

```

```

# format for display
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).round(2)
r_mc["% of total"] = r_mc["% of total"].map(lambda x: F"{x}%")
r_mc["quantity"] = r_mc.quantity.map(lambda x: thousandsSeparator(int(x), "DE"))
r_mc["fail rate"] = r_mc["fail rate"].map(lambda x: F"{x}%")
r_mc[unit_label] = r_mc[unit_label].map(lambda x: F"{np.ceil(x)}")

cols_to_use = {"item":"Objekt", "quantity":"Gesamt", "% of total": "% Gesamt", "fail rate": "fail-rate", unit_label:unit_label}
r_mc.rename(columns=cols_to_use, inplace=True)

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

fig, axs = plt.subplots(figsize=(11, len(data)*.6))

sut.hide_spines_ticks(axs)

table_six = sut.make_a_table(axs, data.values, colLabels=data.columns,
colWidths=[.52, .12, .12, .12, .12], bbox=[0,0,1,1], **{"loc": "lower center"})
table_six.get_celld()[(0,0)].get_text().set_text(" ")
table_six.set_fontsize(12)

plt.tight_layout()
glue("rivers2_de", fig, display=False)
plt.close()

```

	Gesamt	
Windeln - Tücher	305	
Zigarettenfilter	300	
Getränke Glasflasche, Stücke	181	
Industriefolie (Kunststoff)	176	
Snack-Verpackungen	130	
Fragmentierte Kunststoffstücke	109	
Verpackungsfolien, nicht für Lebensmittel	87	
Kronkorken, Lasche von Dose/Ausfreisslachen	80	
Einkaufstaschen, Shoppingtaschen	60	
Expandiertes Polystyrol	53	

Abb. 1.12 #

Abbildung 1.12: Die häufigsten Objekte aus Erhebungen an Fliessgewässern. Windeln, Feuchttücher und Taschen/Tüten aus Kunststoff resp. Fragmente davon gehören nur entlang von Fliessgewässern zu den häufigsten Objekten, nicht jedoch an Ufern von Seen.

1.7. Annex#

1.7.1. 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.

Alle Erhebungsgebiete fragmentierte Kunststoffe und geschäumte Kunststoffe nach Grösse, Median p/100 m, Anzahl der Stücke und Prozent der Gesamtmenge.

```
# collect the data before aggregating foams for all locations in the survey area
# group by loc_date and code
# Combine the different sizes of fragmented plastics and styrofoam
# the codes for the foams
some_foams = ["G81", "G82", "G83", "G74"]
```

```

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

fd_frags_foams = before_agg[(before_agg.code.isin([*some_frag_plas,
*some_foams]))&(before_agg.location.isin(fd.location.unique()))].groupby(["loc_date",
"code"], as_index=False).agg(agg_pcs_quantity)
fd_frags_foams = fd_frags_foams.groupby("code").agg({unit_label:"median",
"quantity":"sum"})

# 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:thousandsSeparator(int(x), "DE"))

# 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=(9,len(data)*.6))
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=a_color)
table_seven.get_celld()[(0,0)].get_text().set_text(" ")
table_seven.set_fontsize(12)

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

```

	p/100 m	Gesa
Schaumstoffverpackungen/Isolierung	0.0	203
Kunststoff-/Polystyrolteile 0,5 - 2,5 cm	0.0	495
Kunststoff/Polystyrolschaumstoff 2,5 > < 50	0.0	111
Kunststoffstücke 0,5cm - 2,5cm	7.0	3 66
Kunststoffstücke 2,5 cm - 50 cm	7.0	3 13
Kunststoffteile > 50cm	0.0	2
Schaumstoffstücke aus Polystyrol 0,5 cm - 2,5 cm	2.0	3 71
Schaumstoffstücke aus Polystyrol 2,5 - 50cm	1.0	1 84
Styroporstücke > 50cm	0.0	7

Abb. 1.13 #

Abbildung 1.13: Fragmentierte Kunststoffe und geschäumte Kunststoffe verschiedener Größenordnungen nach Median p/100 m, Anzahl der gefundenen Objekte und prozentualer Anteil an der Gesamtmenge.

1.7.2. Organisationen:#

1. Precious Plastic Léman
2. Association pour la Sauvegarde du Léman
3. Geneva international School
4. Teilnehmende am Kurs zu Abfalltechnik (Solid waste engineering), Eidgenössische Technische Hochschule Lausanne
5. Summit Foundation
6. Ostschweizer Fachhochschule OST
7. Hackuarium
8. hammerdirt

1.7.3. Gemeinden, Seen und Fliessgewässer mit Erhebungen#

```
lakes = dfBeaches.loc[(dfBeaches.index.isin(fd.location.unique()))&(dfBeaches.water == "l")]["water_name"].unique()
rivers = dfBeaches.loc[(dfBeaches.index.isin(fd.location.unique()))&(dfBeaches.water == "r")]["water_name"].unique()

# gather the municipalities and the population:
fd_pop_map = dfBeaches.loc[fd.location.unique()][["city", "population"]].copy()
fd_pop_map.drop_duplicates(inplace=True)
fd_pop_map.set_index("city", drop=True, inplace=True)
munis_joined = ", ".join(sorted(fd_pop_map.index))

muni_string = F"""**Gemeinden:**\n\n>{munis_joined}"""
"""
md(muni_string)
```

Gemeinden:

Aarau, Allaman, Ascona, Beatenberg, Bellinzona, Bern, Biel/Bienne, Boudry, Bourg-en-Lavaux, Brienz (BE), Brissago, Brugg, Brügg, Burgdorf, Bönigen, Cheyres-Châbles, Cudrefin, Dietikon, Erlach, Estavayer, Freienbach, Gals, Gambarogno, Gebenstorf, Genève, Gland, Glarus Nord, Grandson, Hauterive (NE), Hünenberg, Kallnach, König, Küsnacht (ZH), La Tour-de-Peilz, Lausanne, Lavey-Morcles, Le Landeron, Leuk, Ligerz, Locarno, Lugano, Luterbach, Lüscherz, Merenschwand, Minusio, Montreux, Neuchâtel, Nidau, Port, Préverenges, Quartet, Rapperswil-Jona, Richterswil, Riddes, Rubigen, Saint-Gingolph, Saint-Sulpice (VD), Salgesch, Schmerikon, Sion, Solothurn, Spiez, Stäfa, Thun, Tolochenaz, Unterengstringen, Unterseen, Versoix, Vevey, Vinelz, Walenstadt, Walperswil, Weesen, Weggis, Yverdon-les-Bains, Zug, Zürich

```
lakes_joined = ", ".join(sorted(lakes))

lake_string = F"""**Seen:**\n\n>{lakes_joined}"""
"""
md(lake_string)
```

Seen:

Bielersee, Brienzersee, Lac Léman, Lago Maggiore, Lago di Lugano, Neuenburgersee, Quatre Cantons, Thunersee, Walensee, Zugersee, Zurichsee

Fliessgewässer:

Aare, Aare|Nidau-Büren-Kanal, Cassarate, Dorfbach, Emme, Escherkanal, Jona, La Thièle, Limmat, Linthkanal, Maggia, Reuss, Rhône, Schüss, Seez, Sihl, Ticino

```
# summary statistics:
# number of samples
fd_n_samps = len(dt_all)
# quantity
fd_n_obj = fd.quantity.sum()
# number of locations
fd_n_locs = fd.location.nunique()
# number of cities
fd_n_munis = len(fd_pop_map.index)
```

```

# populations
fd_effected_population = fd_pop_map.sum()
# list of locations
fd_locs = fd.location.unique()
# list of survey keys
fd_samps = fd.loc_date.unique()
(lambda x: f"{locale.format_string('%d', x, grouping=True)}")
```

obj_string = thousandsSeparator(int(fd_n_obj), "DE")
surv_string = locale.format_string('%d', int(fd_n_samps), grouping=True)
pop_string = thousandsSeparator(int(fd_effected_population), "DE")

date_quantity_context = F"Für den Zeitraum zwischen März 2020 und Mai 2021, wurden im Rahmen von {surv_string} Erhebungen insgesamt {obj_string} Objekte entfernt und identifiziert."
geo_context = F"Die Erhebungen wurden an {fd_n_locs} verschiedenen Orten durchgeführt. In den {fd_n_munis} betroffenen Gemeinden leben ca. {pop_string} Menschen."

md(F"{date_quantity_context} {geo_context }")

Für den Zeitraum zwischen März 2020 und Mai 2021, wurden im Rahmen von 386 Erhebungen insgesamt 54 744 Objekte entfernt und identifiziert. Die Erhebungen wurden an 143 verschiedenen Orten durchgeführt. In den 77 betroffenen Gemeinden leben ca. 1 735 991 Menschen.

1.7.4. Die Erhebungsorte#

```
# display the survey locations
```

pd.set_option("display.max_rows", None)
disp_columns = ["latitude", "longitude", "city"]
disp_beaches = dfBeaches.loc[fd_locs][disp_columns]
new_names = {"slug": "Standort", "city": "Stadt"}
disp_beaches.reset_index(inplace=True)
disp_beaches.rename(columns=new_names, inplace=True)
disp_beaches.set_index("Standort", inplace=True, drop=True)

disp_beaches

	latitude	longitude	Stadt
Standort			
maladaire	46.446296	6.876960	La Tour-de-Peilz
preverenges	46.512690	6.527657	Préverenges
caprino	45.987963	8.986241	Lugano
focce-del-cassarate	46.002411	8.961477	Lugano
lido	46.002004	8.962156	Lugano
lugano-centro	46.002627	8.950724	Lugano

	latitude	longitude	Stadt
Standort			
spiaggia-parco-ciani	46.002510	8.960820	Lugano
via-brunari-spiaggia	46.202350	9.016910	Bellinzona
golene-gudo	46.170655	8.946657	Bellinzona
isole-di-brissago-porto	46.132760	8.735030	Brissago
magadino-lido	46.149349	8.855663	Gambarogno
rivapiana	46.172079	8.813255	Minusio
via-mirasole-spiaggia	46.199530	9.014930	Bellinzona
aare-port	47.116170	7.269550	Port
limmat_dietikon_keiserp	47.407989	8.409540	Dietikon
schutzenmatte	47.057666	7.634001	Burgdorf
zugerseecholler_cham_blarerm	47.178216	8.480013	Zug
la-petite-plage	46.785054	6.656877	Yverdon-les-Bains
vidy-ruines	46.516221	6.596279	Lausanne
baby-plage-geneva	46.208558	6.162923	Genève
grand-clos	46.387746	6.843686	Saint-Gingolph
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
vierwaldstattersee_weggis_schoberls_1	47.044532	8.418569	Weggis
vierwaldstattersee_weggis_schoberls_2	47.043953	8.417308	Weggis
vierwaldstattersee_weggis_schoberls_3	47.029957	8.410801	Weggis

	latitude	longitude	Stadt
Standort			
mullermatte	47.133339	7.227907	Biel/Bienne
limmat_unterengstringen_oggierbuhrer	47.409400	8.446933	Unterengstringen
limmat_zuerich_wagnerc	47.403496	8.477770	Zürich
limmat_zurich_mortensen_a_meiera	47.400252	8.485411	Zürich
quai-maria-belgia	46.460156	6.836718	Vevey
sihl_zuerich_eggerskohlingera	47.381898	8.538328	Zürich
bielersee_vinelz_fankhausers	47.038398	7.108311	Vinelz
erlach-camping-strand	47.047159	7.097854	Erlach
anarchy-beach	46.447216	6.859612	La Tour-de-Peilz
gasi-strand	47.128442	9.110831	Weesen
rastplatz-stampf	47.215177	8.844286	Rapperswil-Jona
zuerichsee_richterswil_benkoem_2	47.217646	8.698713	Richterswil
sentiero-giro-del-golf-spiaggia	46.145770	8.790430	Ascona
vira-gambarogno	46.143343	8.838759	Gambarogno
canale-saleggi	46.200625	9.015853	Bellinzona
gummligrabbe	46.989290	7.250730	Kallnach
mannewil	46.996382	7.239024	Kallnach
lavey-les-bains-2	46.207726	7.011685	Lavey-Morcles
schusspark-strand	47.146500	7.268620	Biel/Bienne
walensee_walenstadt_wysse	47.121828	9.299552	Walenstadt
pfafikon-bad	47.206766	8.774182	Freienbach
zurichsee_kusnachterhorn_thirkell-whitej	47.317685	8.576860	Küschnacht (ZH)
plage-de-cheyres	46.818689	6.782256	Cheyres-Châbles

	latitude	longitude	Stadt
Standort			
leuk-mattenstrasse	46.314754	7.622521	Leuk
thun-strandbad	46.739939	7.633520	Thun
zurichsee_wollishofen_langendorf	47.345890	8.536155	Zürich
pont-sous-terre	46.202960	6.131577	Genève
oben-am-see	46.744451	8.049921	Brienz (BE)
thunersee_spiez_meierd_1	46.704437	7.657882	Spiez
cully-plage	46.488887	6.741396	Bourg-en-Lavaux
preverenges-le-sout	46.508905	6.534526	Préverenges
la-pecherie	46.463919	6.385732	Allaman
villa-barton	46.222350	6.152500	Genève
untertenzen	47.115260	9.254780	Quarten
zurcher-strand	47.364200	8.537420	Zürich
oyonne	46.456682	6.852262	La Tour-de-Peilz
lavey-la-source	46.200804	7.021866	Lavey-Morcles
lavey-les-bains	46.205159	7.012722	Lavey-Morcles
spackmatt	47.223749	7.576711	Luterbach
luscherz-plage	47.047955	7.151242	Lüscherz
strandboden-biel	47.132510	7.233142	Biel/Bienne
jona-river-rastplatz	47.216100	8.844430	Rapperswil-Jona
reuss_hunenberg_eberhardy	47.232976	8.401012	Hünenberg
reuss_ottenbach_schoenenbergerl	47.278354	8.394224	Merenschwand
zuerichsee_staefa_hennm	47.234643	8.769881	Stäfa
signalpain	46.786200	6.647360	Yverdon-les-Bains

		latitude	longitude	Stadt
Standort				
baby-plage-ii-geneve		46.208940	6.164330	Genève
pointe-dareuse		46.946190	6.870970	Boudry
les-glariers		46.176736	7.228925	Riddes
nidau-strand		47.127196	7.232613	Nidau
tiger-duck-beach		46.518256	6.582546	Saint-Sulpice (VD)
linthdammweg		47.131790	9.091910	Weesen
mols-rocks		47.114343	9.288174	Quarten
seeflechsen		47.130223	9.103400	Glarus Nord
seemuhlestrasse-strand		47.128640	9.295100	Walenstadt
muhlehorn-dorf		47.118448	9.172124	Glarus Nord
zuerichsee_waedenswil_colomboc_1		47.219547	8.691961	Richterswil
camp-des-peches		47.052812	7.074053	Le Landeron
delta-park		46.720078	7.635304	Spiez
le-pierrier		46.439727	6.888968	Montreux
pecos-plage		46.803590	6.636650	Grandson
beich		47.048495	7.200528	Walperswil
aare_suhrespitz_badert		47.405669	8.066018	Aarau
tschilljus		46.304399	7.580262	Salgesch
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

		latitude	longitude	Stadt
	Standort			
	boiron	46.491030	6.480162	Tolochenaz
	rocky-plage	46.209737	6.164952	Genève
	lacleman_gland_lecoanets	46.402811	6.281959	Gland
	route-13	46.162521	8.782579	Locarno
	murg-bad	47.115307	9.215691	Quarten
	schmerikon-bahnhof-strand	47.224780	8.944180	Schmerikon
	aabach	47.220989	8.940365	Schmerikon
	la-thiele_le-mujon_confluence	46.775340	6.630900	Yverdon-les-Bains
	nouvelle-plage	46.856646	6.848428	Estavayer
	baye-de-montreux-g	46.430834	6.908778	Montreux
	les-vieux-ronquoz	46.222049	7.361664	Sion
	sihl_zuerich_eichenbergerd	47.339588	8.516697	Zürich
	versoix	46.289194	6.170569	Versoix
	augustmutzenbergstrandweg	46.686640	7.689760	Spiez
	parc-des-pierrettes	46.515215	6.575531	Saint-Sulpice (VD)
	plage-de-st-sulpice	46.513265	6.570977	Saint-Sulpice (VD)
	aare_bern_gerberm	46.989363	7.452098	Bern
	brugg-be-buren-bsg-standort	47.122220	7.281410	Brügg
	kusnachter-dorfbach-tobelweg-1-4	47.317770	8.588258	Küschnacht (ZH)
	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

		latitude	longitude	Stadt
Standort				
bern-fahrstrasse		46.975866	7.444210	Bern
gals-reserve		47.046272	7.085007	Gals
ascona-traghetto-spiaggia		46.153882	8.768480	Ascona
pacha-mama-gerra		46.123473	8.783508	Gambarogno
gerra-gambarogno		46.123366	8.785786	Gambarogno
san-nazzaro-gambarogno		46.130443	8.798657	Gambarogno
hauterive-petite-plage		47.010797	6.980304	Hauterive (NE)
flibach-river-right-bank		47.133742	9.105461	Weesen
aare-limmatspitz		47.501060	8.237371	Gebenstorf
aare_brugg_buchie		47.494855	8.236558	Brugg
linth_route9brucke		47.125730	9.117340	Glarus Nord
seez_spennwiesenbrucke		47.114948	9.310123	Walenstadt
bain-des-dames		46.450507	6.858092	La Tour-de-Peilz
luscherz-two		47.047519	7.152829	Lüscherz
impromptu_cudrefin		46.964496	7.027936	Cudrefin
vira-spiaggia-left-of-river		46.142900	8.838407	Gambarogno
linth_mollis		47.131370	9.094768	Glarus Nord
seez		47.119830	9.300251	Walenstadt
tolochenaz		46.497509	6.482875	Tolochenaz
limmat_zuerich_suterdglaurserp		47.394539	8.513443	Zürich
aare_post		47.116665	7.271953	Port
aare_bern_scheurerk		46.970967	7.452586	Bern
strandbeiz		47.215862	8.843098	Rapperswil-Jona

1.7.5. Inventar der Objekte#

```
complete_inventory = code_totals[code_totals.quantity>0][["item", "groupname",
"quantity", "% of total","fail rate"]]

new_names = {"item":"Objekte", "groupname":"Gruppenname", "quantity":"Menge", "fail
rate":"Fail-Pass", "% of total": "% Gesamt", }

complete_inventory.rename(columns=new_names, inplace=True)
complete_inventory.sort_values(by="Menge", ascending=False)
```

	Objekte	Gruppenname	Menge	% Gesamt	Fail-Pass
code					
G27	Zigarettenfilter	Tabakwaren	8485	15.50	87
Gfrags	Fragmentierte Kunststoffstücke	Plastikfragmente	7400	13.52	86
Gfoam	Expandiertes Polystyrol	Infrastruktur	5563	10.16	68
G30	Snack-Verpackungen	Essen und Trinken	3325	6.07	85
G67	Industriefolie (Kunststoff)	Landwirtschaft	2534	4.63	69
G200	Getränke Glasflasche, Stücke	Essen und Trinken	2136	3.90	65
G112	Industriepellets (Nurdles)	Mikroplastik (< 5mm)	1968	3.59	30
G74	Schaumstoffverpackungen/ Isolierung	Infrastruktur	1702	3.11	53
G95	Wattestäbchen / Tupfer	Abwasser	1406	2.57	50
G117	Styropor < 5mm	Mikroplastik (< 5mm)	1209	2.21	25
G89	Kunststoff-Bauabfälle	Infrastruktur	992	1.81	52
G941	Verpackungsfolien, nicht für Lebensmittel	Nicht- Lebensmittelverpackungen	894	1.63	38
G178	Kronkorken, Lasche von Dose/Ausfreisslachen	Essen und Trinken	700	1.28	52
G25	Tabak; Kunststoffverpackungen, Behälter	Tabakwaren	649	1.19	46
G21	Getränke-Deckel, Getränkeverschluss	Essen und Trinken	623	1.14	37
G106	Kunststofffragmente eckig	Mikroplastik (< 5mm)	592	1.08	20

	Objekte	Gruppenname	Menge	% Gesamt	Fail-Pass
code					
	<5mm				
G98	Windeln - Tücher	Abwasser	588	1.07	26
G24	Ringe von Plastikflaschen/Behältern	Essen und Trinken	541	0.99	42
G177	Verpackungen aus Aluminiumfolie	Essen und Trinken	521	0.95	47
G10	Lebensmittelbehälter zum einmaligen Gebrauch a...	Essen und Trinken	448	0.82	31
G23	unidentifizierte Deckel	Nicht-Lebensmittelverpackungen	423	0.77	35
G156	Papierfragmente	Nicht-Lebensmittelverpackungen	420	0.77	32
G35	Strohhalme und Rührstäbchen	Essen und Trinken	419	0.77	42
G70	Schrotflintenpatronen	Freizeit und Erholung	391	0.71	21
G31	Schleckstengel, Stengel von Lutscher	Essen und Trinken	381	0.70	37
G73	Gegenstände aus Schaumstoff & Teilstücke (nich...	Freizeit und Erholung	335	0.61	23
G33	Einwegartikel; Tassen/Becher & Deckel	Essen und Trinken	328	0.60	34
G32	Spielzeug und Partyartikel	Freizeit und Erholung	320	0.58	38
G904	Feuerwerkskörper; Raketenkappen	Freizeit und Erholung	301	0.55	28
G3	Einkaufstaschen, Shoppingtaschen	Nicht-Lebensmittelverpackungen	285	0.52	18
G22	Deckel für Chemikalien, Reinigungsmittel (Ohne...	Infrastruktur	257	0.47	19
G211	Sonstiges medizinisches Material	Persönliche Gegenstände	256	0.47	31
G922	Etiketten, Strichcodes	Nicht-	252	0.46	27

	Objekte	Gruppenname	Menge	% Gesamt	Fail-Pass
code					
		Lebensmittelverpackungen			
G66	Umreifungsbänder; Hartplastik für Verpackung f...	Infrastruktur	248	0.45	32
G100	Medizin; Behälter/Röhrchen/Verpackungen	Abwasser	244	0.45	28
G923	Taschentücher, Toilettenpapier, Servietten, Pa...	Persönliche Gegenstände	241	0.44	25
G103	Kunststofffragmente rund <5mm	Mikroplastik (< 5mm)	238	0.43	3
G921	Keramikfliesen und Bruchstücke	Infrastruktur	231	0.42	13
G96	Hygienebinden/ Höscheneinlagen/Tampons und Appl...	Abwasser	215	0.39	18
G50	Schnur < 1cm	Freizeit und Erholung	205	0.37	29
G153	Papierbecher, Lebensmittelverpackungen aus Pap...	Essen und Trinken	201	0.37	19
G125	Luftballons und Luftballonstäbchen	Freizeit und Erholung	170	0.31	20
G91	Bio-Filtermaterial / Trägermaterial aus Kunst...	Abwasser	162	0.30	19
G159	Kork	Essen und Trinken	162	0.30	21
G38	Abdeckungen; Kunststoffverpackungen, Folien zu...	nicht klassifiziert	160	0.29	4
G940	Schaumstoff EVA (flexibler Kunststoff)	Freizeit und Erholung	158	0.29	7
G208	Glas oder Keramikfragmente >2.5 cm	nicht klassifiziert	145	0.26	10
G186	Industrieschrott	Infrastruktur	141	0.26	15

	Objekte	Gruppenname	Menge	% Gesamt	Fail-Pass
code					
G213	Paraffinwachs	Freizeit und Erholung	138	0.25	13
G165	Glacestengel (Eisstiele), Zahnstocher, Essstäb...	Essen und Trinken	126	0.23	15
G204	Baumaterial; Ziegel, Rohre, Zement	Infrastruktur	125	0.23	9
G155	Feuerwerkspapierhülsen und -fragmente	Freizeit und Erholung	124	0.23	6
G93	Kabelbinder	Infrastruktur	121	0.22	16
G149	Papierverpackungen	Nicht-Lebensmittelverpackungen	121	0.22	8
G152	Papier &Karton;Zigarettenschachteln, Papier/Ka...	Tabakwaren	120	0.22	13
G90	Blumentöpfe aus Plastik	Landwirtschaft	119	0.22	14
G137	Kleidung, Handtücher und Lappen	Persönliche Gegenstände	118	0.22	13
G908	Klebeband; elektrisch, isolierend	Infrastruktur	117	0.21	15
G203	Geschirr aus Keramik oder Glas, Tassen, Teller...	Essen und Trinken	115	0.21	11
G936	Folien für Gewächshäuser	Landwirtschaft	110	0.20	12
G34	Besteck, Teller und Tabletts	Essen und Trinken	109	0.20	16
G905	Haarspangen, Haargummis, persönliche Accessoires...	Persönliche Gegenstände	105	0.19	18
G198	Andere Metallteile < 50 cm	Infrastruktur	104	0.19	16
G914	Büroklammern, Wäscheclammern, Gebrauchsgegenstände...	Persönliche Gegenstände	99	0.18	12
G131	Gummibänder	Persönliche Gegenstände	94	0.17	16
G191	Draht und Gitter	Landwirtschaft	90	0.16	12

	Objekte	Gruppenname	Menge	% Gesamt	Fail-Pass
code					
G28	Stifte, Deckel, Druckbleistifte usw.	Persönliche Gegenstände	88	0.16	13
G115	Schaumstoff <5mm	Mikroplastik (< 5mm)	88	0.16	4
G124	Kunststoff-oder Schaumstoffprodukte	nicht klassifiziert	87	0.16	8
G87	Abdeckklebeband / Verpackungsklebeband	Infrastruktur	87	0.16	14
G148	Kartonkisten und Stücke	Nicht-Lebensmittelverpackungen	85	0.16	8
G928	Bänder und Schleifen	Persönliche Gegenstände	84	0.15	8
G26	Feuerzeug	Tabakwaren	83	0.15	14
G146	Papier, Karton	Nicht-Lebensmittelverpackungen	82	0.15	6
G175	Getränkedosen (Dosen, Getränke)	Essen und Trinken	76	0.14	11
G157	Papier	Nicht-Lebensmittelverpackungen	66	0.12	6
G7	Getränkeflaschen < = 0,5 l	Essen und Trinken	65	0.12	9
G4	Kleine Plastikbeutel; Gefrierbeutel, Zipsäckc...	Nicht-Lebensmittelverpackungen	64	0.12	8
G142	Seil, Schnur oder Netze	Freizeit und Erholung	62	0.11	10
G135	Kleidung, Fussbekleidung, Kopfbedeckung, Hands...	Persönliche Gegenstände	61	0.11	9
G105	Kunststofffragmente subangulär <5mm	Mikroplastik (< 5mm)	59	0.11	2
G927	Plastikschnur von Rasentrimmern, die zum Schne...	Infrastruktur	57	0.10	8
G201	Gläser, einschliesslich Stücke	Essen und Trinken	55	0.10	7
G942	Kunststoffspäne von	nicht klassifiziert	53	0.10	8

	Objekte	Gruppenname	Menge	% Gesamt	Fail-Pass
code					
	Drehbänken, CNC-Bearbeitung				
G194	Kabel, Metalldraht oft in Gummi- oder Kunststo...	Infrastruktur	52	0.09	10
G939	Kunststoffblumen, Kunststoffpflanzen	Persönliche Gegenstände	49	0.09	6
G20	Laschen und Deckel	Nicht-Lebensmittelverpackungen	48	0.09	6
G65	Eimer	Landwirtschaft	48	0.09	6
G943	Zäune Landwirtschaft, Kunststoff	Landwirtschaft	47	0.09	5
G144	Tampons	Abwasser	46	0.08	3
G48	Seile, synthetische	Freizeit und Erholung	46	0.08	8
G134	Sonstiges Gummi	nicht klassifiziert	45	0.08	8
G210	Sonstiges Glas/Keramik Materialien	nicht klassifiziert	43	0.08	3
G170	Holz (verarbeitet)	Landwirtschaft	43	0.08	6
G937	Pheromonköder für Weinberge	Landwirtschaft	38	0.07	4
G101	Robidog Hundekot-Säcklein, andere Hundekotsäck...	Persönliche Gegenstände	37	0.07	8
G917	Blähton	nicht klassifiziert	36	0.07	4
G59	Monofile Angelschnur (Angeln)	Freizeit und Erholung	35	0.06	6
G944	Pelletmasse aus dem Spritzgussverfahren	nicht klassifiziert	34	0.06	0
G2	Säcke, Taschen	Nicht-Lebensmittelverpackungen	32	0.06	4
G12	Kosmetika, Behälter für Körperpflegeprodukte, ...	Persönliche Gegenstände	31	0.06	5
G207	Tonkrüge zum Fangen von	nicht klassifiziert	30	0.05	0

	Objekte	Gruppenname	Menge	% Gesamt	Fail-Pass
code					
	Kraken				
G126	Bälle	Freizeit und Erholung	30	0.05	5
G901	Medizinische Masken, synthetische	Persönliche Gegenstände	29	0.05	6
G918	Sicherheitsnadeln, Büroklammern, kleine Gebrau...	Persönliche Gegenstände	28	0.05	5
G931	(Absperr)band für Absperrungen, Polizei, Baust...	Infrastruktur	28	0.05	3
G99	Spritzen - Nadeln	Persönliche Gegenstände	28	0.05	6
G919	Nägel, Schrauben, Bolzen usw.	Infrastruktur	27	0.05	3
G158	Sonstige Gegenstände aus Papier	Nicht-Lebensmittelverpackungen	27	0.05	2
G167	Streichhölzer oder Feuerwerke	Freizeit und Erholung	26	0.05	2
G933	Taschen, Etuis für Zubehör, Brillen, Elektroni...	Persönliche Gegenstände	26	0.05	4
G6	Flaschen und Behälter aus Kunststoff, nicht fü...	Nicht-Lebensmittelverpackungen	25	0.05	2
G154	Zeitungen oder Zeitschriften	Persönliche Gegenstände	25	0.05	2
G161	Verarbeitetes Holz	Landwirtschaft	24	0.04	4
G182	Angeln; Haken, Gewichte, Köder, Senklei, usw.	Freizeit und Erholung	23	0.04	3
G8	Getränkeflaschen > 0,5L	Essen und Trinken	23	0.04	3
G145	Andere Textilien	Persönliche Gegenstände	22	0.04	4
G114	Folien <5mm	Mikroplastik (< 5mm)	22	0.04	1
G43	Sicherheitsetiketten, Siegel für Fischerei ode...	Freizeit und Erholung	21	0.04	2
G930	Schaumstoff-Ohrstöpsel	Persönliche Gegenstände	20	0.04	4

	Objekte	Gruppenname	Menge	% Gesamt	Fail-Pass
code					
G133	Kondome einschließlich Verpackungen	Abwasser	20	0.04	4
G68	Fiberglas-Fragmente	Infrastruktur	20	0.04	4
G929	Elektronik und Teile; Sensoren, Headsets usw.	Persönliche Gegenstände	20	0.04	4
G176	Konservendosen (Lebensmitteldosen)	Essen und Trinken	19	0.03	3
G925	Pakete: Trockenmittel/Feuchtigkeitsabsorber, m...	Nicht-Lebensmittelverpackungen	19	0.03	4
G938	Zahnstocher, Zahnpasta Kunststoff	Essen und Trinken	18	0.03	4
G926	Kaugummi, enthält oft Kunststoffe	Essen und Trinken	18	0.03	3
G36	Säcke aus strapazierfähigem Kunststoff für 25 ...	Landwirtschaft	18	0.03	2
G118	Kleine Industrielle Kugelchen <5mm	Mikroplastik (< 5mm)	17	0.03	2
G123	Polyurethan-Granulat < 5mm	Mikroplastik (< 5mm)	17	0.03	0
G11	Kosmetika für den Strand, z.B. Sonnencreme	Freizeit und Erholung	17	0.03	3
G64	Kotflügel	nicht klassifiziert	16	0.03	1
G119	Folienartiger Kunststoff (>1mm)	Mikroplastik (< 5mm)	16	0.03	1
G49	Seile > 1cm	Freizeit und Erholung	14	0.03	2
G113	Fäden <5mm	Mikroplastik (< 5mm)	14	0.03	1
G197	sonstiges Metall	Infrastruktur	13	0.02	3
G199	Andere Metallteile > 50 cm	Infrastruktur	13	0.02	1
G29	Kämme, Bürsten und Sonnenbrillen	Persönliche Gegenstände	13	0.02	3

	Objekte	Gruppenname	Menge	% Gesamt	Fail-Pass
code					
G195	Batterien (Haushalt)	Persönliche Gegenstände	12	0.02	2
G128	Reifen und Antriebsriemen	nicht klassifiziert	12	0.02	2
G916	Bleistifte und Bruchstücke	Persönliche Gegenstände	12	0.02	2
G136	Schuhe	Persönliche Gegenstände	11	0.02	2
G37	Netzbeutel, Netztasche, Netzsäcke	Persönliche Gegenstände	11	0.02	2
G906	Kaffeekapseln Aluminium	Essen und Trinken	10	0.02	1
G900	Handschuhe Latex, persönliche Schutzausrüstung	Persönliche Gegenstände	10	0.02	2
G61	Sonstiges Angelzubehör	Freizeit und Erholung	9	0.02	2
G913	Schnuller	Persönliche Gegenstände	8	0.01	2
G104	Kunststofffragmente abgerundet / rundlich <5m...	Mikroplastik (< 5mm)	7	0.01	1
G40	Handschuhe Haushalt/Garten	Persönliche Gegenstände	7	0.01	1
G122	Kunststofffragmente (>1mm)	Mikroplastik (< 5mm)	7	0.01	0
G19	Autoteile	nicht klassifiziert	7	0.01	1
G129	Schlüche und Gummiplatten	nicht klassifiziert	7	0.01	1
G17	Kartuschen von Kartuschen-spritzpistolen	Infrastruktur	7	0.01	1
G141	Teppiche	nicht klassifiziert	6	0.01	1
G5	Plastiksäcke/ Plastiktüten	Nicht-Lebensmittelverpackungen	6	0.01	1
G181	Geschirr aus Metall, Tassen, Besteck usw.	Essen und Trinken	6	0.01	1
G171	Sonstiges Holz < 50cm	Landwirtschaft	6	0.01	1
G97	Behälter von Toilettenerfrischer	Abwasser	6	0.01	1

	Objekte	Gruppenname	Menge	% Gesamt	Fail-Pass
code					
G915	Reflektoren, Mobilitätsartikel aus Kunststoff	Persönliche Gegenstände	6	0.01	1
G151	Tetrapack, Kartons	Essen und Trinken	6	0.01	1
G147	Papiertragetaschen, (Papiertüten)	Nicht-Lebensmittelverpackungen	6	0.01	1
G107	Zylindrische Pellets <5mm	Mikroplastik (< 5mm)	6	0.01	1
G903	Behälter und Packungen für Handdesinfektionsmi...	Persönliche Gegenstände	5	0.01	1
G92	Körperbehälter	Freizeit und Erholung	5	0.01	1
G71	Schuhe Sandalen	Persönliche Gegenstände	5	0.01	1
G138	Schuhe und Sandalen	Persönliche Gegenstände	5	0.01	1
G108	Scheibenförmige Pellets <5mm	Mikroplastik (< 5mm)	5	0.01	1
G172	Sonstiges Holz > 50 cm	Landwirtschaft	4	0.01	1
G945	Rasierklingen	Persönliche Gegenstände	4	0.01	1
G53	Netze und Teilstücke < 50cm	Freizeit und Erholung	4	0.01	0
G52	Netze und Teilstücke	Freizeit und Erholung	4	0.01	1
G907	Kaffeekapseln aus Kunststoff	Essen und Trinken	4	0.01	1
G39	Handschuhe	Persönliche Gegenstände	4	0.01	0
G111	Kugelförmige Pellets < 5mm	Mikroplastik (< 5mm)	4	0.01	1
G62	Schwimmer für Netze	nicht klassifiziert	4	0.01	0
G13	Flaschen, Behälter, Fässer zum Transportieren ...	Landwirtschaft	4	0.01	0
G202	Glühbirnen	nicht klassifiziert	3	0.01	0
G174	Aerosolspraydosen	Infrastruktur	3	0.01	0
G41	Handschuhe Industriell/Professionell	Landwirtschaft	3	0.01	0

	Objekte	Gruppenname	Menge	% Gesamt	Fail-Pass
code					
G934	Sandsäcke, Kunststoff für Hochwasser- und Eros...	Landwirtschaft	3	0.01	0
G143	Segel und Segeltuch	Freizeit und Erholung	3	0.01	0
G63	Bojen	Freizeit und Erholung	3	0.01	0
G140	Sack oder Beutel (Tragetasche), Jute oder Hanf	Landwirtschaft	3	0.01	0
G179	Einweg Grill	Essen und Trinken	3	0.01	0
G116	Granulat <5mm	Mikroplastik (< 5mm)	3	0.01	0
G109	Flache Pellets <5mm	Mikroplastik (< 5mm)	3	0.01	0
G14	Flaschen für Motoröl (Motorölflaschen)	nicht klassifiziert	3	0.01	0
G139	Rucksäcke	Persönliche Gegenstände	3	0.01	0
G902	Medizinische Masken, Stoff	Persönliche Gegenstände	3	0.01	0
G102	Flip-Flops	Persönliche Gegenstände	3	0.01	0
G150	Milchkartons, Tetrapack	Essen und Trinken	3	0.01	0
G999	Keine Gegenstände bei dieser Erhebung/Studie g...	nicht klassifiziert	2	0.00	0
G55	Angelschnur (verwickelt)	Freizeit und Erholung	2	0.00	0
G935	Gummi-Puffer für Geh- und Wanderstöcke und Tei...	Persönliche Gegenstände	2	0.00	0
G932	Bio-Beads, Mikroplastik für die Abwasserbehand...	Abwasser	2	0.00	0
G173	Sonstiges	nicht klassifiziert	2	0.00	0
G185	Behälter mittlerer Grösse	nicht klassifiziert	2	0.00	0
G188	Andere Kanister/Behälter < 4 L	Infrastruktur	2	0.00	0
G190	Farbtöpfe, Farbbüchsen, (Farbeimer)	Infrastruktur	2	0.00	0

	Objekte	Gruppenname	Menge	% Gesamt	Fail-Pass
code					
G214	Öl/Teer	Infrastruktur	2	0.00	0
G132	Schwimmer (Angeln)	Freizeit und Erholung	2	0.00	0
G166	Farbpinsel	Infrastruktur	1	0.00	0
G94	Tischtuch	Freizeit und Erholung	1	0.00	0
G84	CD oder CD-Hülle	Persönliche Gegenstände	1	0.00	0
G196	Grosse metallische Gegenstände	nicht klassifiziert	1	0.00	0
G9	Flaschen und Behälter für Reinigungsmittel und...	Infrastruktur	1	0.00	0
G183	Teile von Angelhaken	Freizeit und Erholung	1	0.00	0
G60	Lichtstab, Knicklicht, Glowsticks	Freizeit und Erholung	1	0.00	0
G51	Fischernetz	Freizeit und Erholung	1	0.00	0
G193	Teile von Autos und Autobatterien	nicht klassifiziert	1	0.00	0

```

# -*- coding: utf8 -*-

# 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 PIL import Image as PILImage
from IPython.display import Markdown as md
from myst_nb import glue
# from IPython.display import display

# 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 ="2021-09-30"
start_end = [start_date, end_date]
a_fail_rate = 50
reporting_unit = 100
unit_label = "p/100 m"

# colors for gradients, tables and charts
cmap2 = ck.cmap2
colors_palette = ck.colors_palette
a_color = "dodgerblue"

# the search term for the survey area
bassin_name = "les-alpes"

# the names for the survey area and the cumulative data
level_names = ["Die Alpen", "Alle Erhebungsgebiete"]

# common aggregations
agg_pcs_quantity = {unit_label:"sum", "quantity":"sum"}
agg_pcs_median = {unit_label:"median", "quantity":"sum"}

# alpes data:
alldata = pd.read_csv("resources/checked_alpes_survey_data.csv")

# survey data lakes and rives
sdata = pd.read_csv("resources/checked_sdata_eos_2020_21.csv")

# location and object data
dfBeaches = pd.read_csv("resources/beaches_with_land_use_rates.csv")
dfCodes = pd.read_csv("resources/codes_with_group_names_2015.csv")

# the dimensional data from each survey
dfDims = pd.read_csv("resources/alpes_dims.csv")

# remove the prefix from the beach names
alpb = dfBeaches[dfBeaches.river_bassin == "les-alpes"].copy()

# removing the prefix from the location names in the location data
alpb["slug"] = alpb.slug.apply(lambda x: x.replace("clean-up-tour-", ""))

# put that back together
dfBeaches = pd.concat([alpb, dfBeaches[dfBeaches.river_bassin != "les-alpes"]])

dfBeaches.set_index("slug", inplace=True)

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

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

```

```

de_codes.set_index("code", inplace=True)

# apply the new language to the codes and materials:
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
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"]
]
for x in codes_to_change:
    dfCodes = sut.shorten_the_value(x, dfCodes)

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: {isinstance({data}, {atype})}.")

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}'

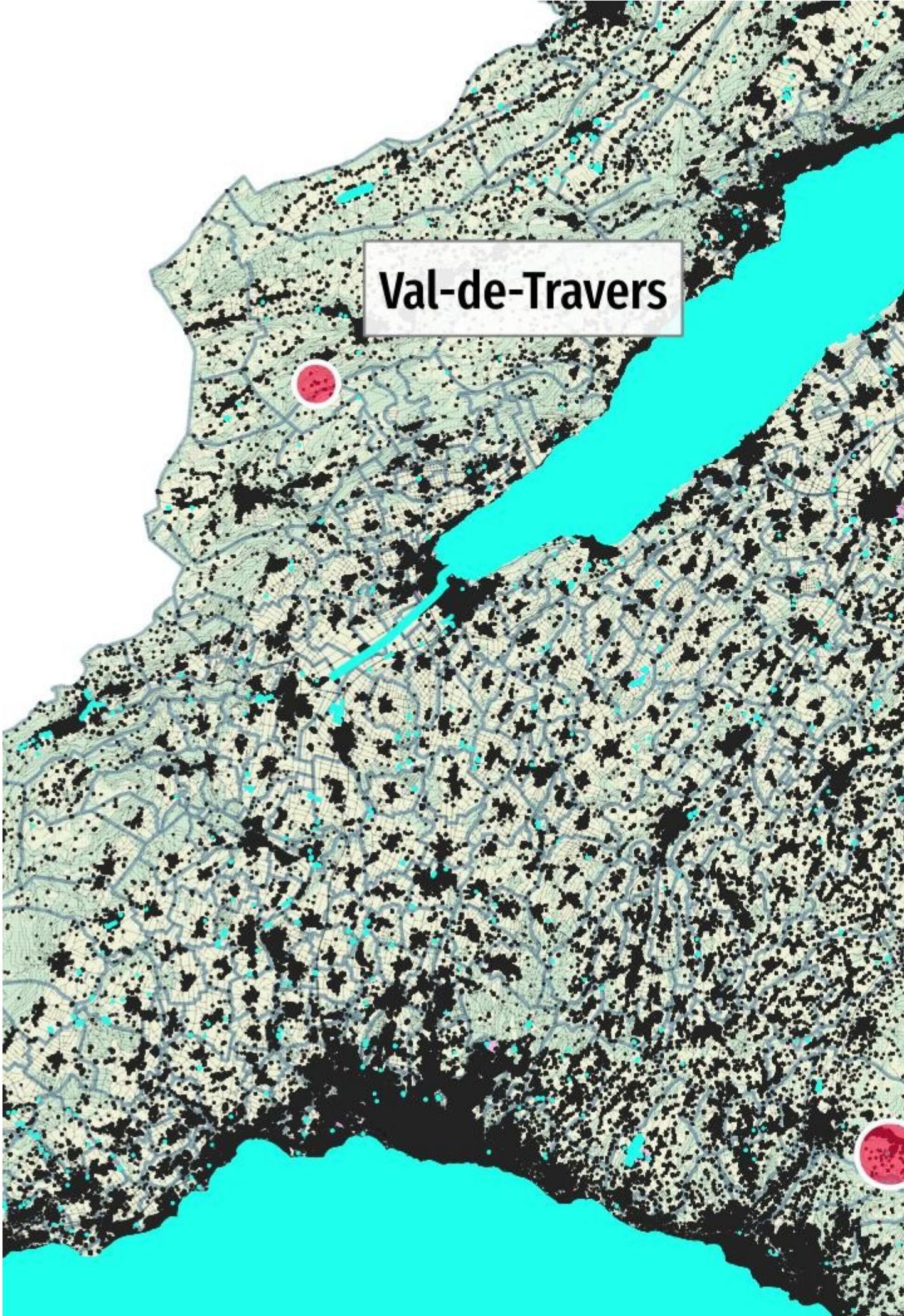
def thousandsSeparator(aninteger, lang):
    astring = "{:,}".format(aninteger)
    if lang == "DE":
        astring = astring.replace(",", " ")
    return astring

# the material was mislabeled by the surveyor
dfCodes.loc["G708", "material"] = "Metal"

```

```
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
```

2. Alpen und der Jura#



Val-de-Travers

Abb. 2.1 #

Abbildung 2.1: Karte des Erhebungsgebiets Alpen und Jura, Clean-ups 2021.

Die Verantwortung für die Erhebungen in den Alpen und im Jura lag bei der Summit Foundation. Die *Summit Foundation* führt seit vielen Jahren Clean-ups (Gruppenevents, bei denen Müll aus dem Gelände beseitigt wird) in Schweizer Bergregionen durch. Zu den Clean-ups im Jahr 2021 gehörten auch eine Reihe von Erhebungen zu Abfallobjekten, die parallel zu den regelmässig stattfindenden Clean-ups durchgeführt wurden. Die Summit Foundation hatte zwei Fragen in Bezug auf IQAASL:

1. Wie kann die Datenerfassung in das aktuelle Geschäftsmodell integriert werden?
2. Was ergeben die Erhebungen auf den Bergpfaden im Vergleich zu denen am Wasser?

Der Zweck von Aufräumaktionen ist es, so viele Abfallobjekte wie möglich aus einem bestimmten Gebiet zu entfernen. Wie viel entfernt werden kann, hängt von den zur Verfügung stehenden Ressourcen ab. Eine Erhebung über Abfallobjekte dient der Identifizierung und Zählung der Objekte in einem bestimmten Gebiet. In diesem Sinne ist eine Aufräumaktion eine Annäherung an das Abfallproblem aus der Perspektive der Abschwächung oder Milderung, und Erhebungen liefern die notwendigen Daten zur Verbesserung der Prävention.

2.1. Erhebungsmethoden#

Insgesamt wurden zwanzig Erhebungen zu Abfallobjekten von der Summit Foundation durchgeführt. Ursprünglich wurden zwei Methoden ausgewählt:

1. Erhebungen entlang bestimmter Streckenabschnitte mit einer definierten Länge und Breite (inbesondere Wanderwege)
2. Erhebungen im Umfeld der Liftinfrastruktur (insbesondere Wartebereiche Schneesport)

Erhebungen im Umfeld der Liftinfrastruktur (insbesondere Wartebereiche Schneesport):

1. Ein Abschnitt des Weges oder der Fläche wird gemessen
2. Alle sichtbaren Verunreinigungen werden entfernt, gezählt und klassifiziert
3. Die Ergebnisse und Abmessungen werden aufgezeichnet

Der Unterschied zwischen den beiden Methoden liegt in der Art und Weise, wie die Grenzen des Vermessungsgebiets festgelegt werden. Wenn ein Weg benutzt wird, werden die Grenzen des Vermessungsgebiets durch den Weg selbst festgelegt, nicht durch die Person, die die Erhebung ausführt. Im Sommer sind die Barrieren und Schilder, welche Pisten etc. markieren, alle entfernt worden, so dass es für die Person, die die Erhebung ausführt, schwierig ist, die korrekten Grenzen genau zu bestimmen.

2.2. Kumulierte Gesamtzahlen für das Erhebungsgebiet#

```
# define the final survey data set here:  
# combined the alps and the lakes and rivers data  
predata = pd.concat([alldata, sdata])  
  
# remove the prefix from the location names  
predata["location"] = predata["location"].map(lambda x: sut.get_rid_of_ix(x,  
prefix="clean-up-tour-"))  
  
# language specific
```

```

predata.rename(columns=sut.luse_ge, inplace=True)
predata["groupname"] = predata.groupname.map(lambda x: sut.group_names_de[x])

# assign loc_date and make date stamp
predata["loc_date"] = list(zip(predata.location, predata["date"]))
predata["date"] = pd.to_datetime(predata["date"])
predata["date"] = predata["date"].dt.strftime(date_format)
predata["date"] = pd.to_datetime(predata["date"], format=date_format)
predata.rename(columns={"p/100m": unit_label}, inplace=True)

# remove prefixes from the alps location names in the surveys
fd = predata[predata.river_bassin == "les-alpes"].copy()
sd = predata[predata.river_bassin != "les-alpes"]

# merge it back in with the rest
a_data = pd.concat([fd, sd])

# scale the streets to kilometers
a_data["streets"] = a_data.streets.astype("int")

# the totals for each survey and the locations in the feature data
fd_dt=fd.groupby(["loc_date", "date", "month", "location"],
as_index=False).agg(agg_pcs_quantity)

# survey totals
dt_all = a_data.groupby(["loc_date", "location", "river_bassin", "date"],
as_index=False).agg(agg_pcs_quantity)

# the unique locations and samples
t = {"samples":fd.loc_date.nunique(),
      "locations":fd.location.unique(),
      "nlocations":fd.location.nunique(),
      "fdtotalq": fd.quantity.sum()
     }

# gather the municipalities and the population:
fd_pop_map = dfBeaches.loc[fd.location.unique()][["city", "population"]].copy()
fd_pop_map.drop_duplicates(inplace=True)
fd_pop_map.set_index("city", drop=True, inplace=True)

t.update({"nmunis":len(fd_pop_map.index)})

obj_string = thousandsSeparator( int(t["fdtotalq"]), language)
surv_string = locale.format_string('%d', int(t["samples"]), grouping=True)
pop_string = thousandsSeparator( int(fd_pop_map.sum()[0]), language)

date_quantity_context = f"Zwischen {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} Erhebungen in
{t['samples']}"

geo_context = f"verschiedenen Orten in {t['nmunis']} Gemeinden und einer
Gesamtbevölkerung von {pop_string}. Einwohnern insgesamt
{locale.format_string('d',t['fdtotalq'])} Objekte entfernt und identifiziert."
munis_joined = ", ".join(sorted(fd_pop_map.index))

```

```
# put that all together:
lake_string = F"""
Zwischen März 2020 und September 2021 wurden im Rahmen von 20 Erhebungen in 18
Gemeinden mit einer Gesamtbevölkerung von 70 606 Personen insgesamt 7 776 Objekte
entfernt und identifiziert.
```

```
\n\n >{munis_joined}"""
```

Zwischen März 2020 und September 2021 wurden im Rahmen von 20 Erhebungen in 18 Gemeinden mit einer Gesamtbevölkerung von 70 606 Personen insgesamt 7 776 Objekte entfernt und identifiziert.

Airolo, Andermatt, Calanca, Châtel-Saint-Denis, Grindelwald, La Roche, Lens, Mesocco, Nendaz, Ollon, Ormont-Dessus, Riddes, Rovio, Troistorrents, Val de Bagnes, Val-d'Illiez, Val-de-Charmey, Val-de-Travers

2.2.1. Gesamtzahlen der Erhebungen#

```
# gather the dimensional data for the time frame from dfDims
# match records to survey data
# the dimensional data, remove prefix and reassemble
fd_dims = dfDims[(dfDims.river_bassin == "les-alpes")&(dfDims.date >=
start_date)&(dfDims.date <= end_date)].copy()
fd_dims["location"] = fd_dims.location.apply(lambda x: x.replace("clean-up-tour-", ""))
fd_dims["loc_date"] = list(zip(fd_dims.location, fd_dims.date))

# map the daily totals to the survey dimensions
q_map = fd_dt[["loc_date", "quantity"]].set_index("loc_date").quantity
fd_dims["quantity"] = fd_dims.loc_date.map(lambda x: q_map.loc[[x]].values[0])

# get the pieces per meter
fd_dims[unit_label] = ((fd_dims.quantity/fd_dims["length"])*reporting_unit).round(2)

# make a loc_date column
fd_dims["loc_date"] = list(zip(fd_dims.location, fd_dims.date))

# get the cumulative values for each location:
agg_for_table = {
    "quantity":"sum",
    unit_label:"mean",
    "total_w":"sum",
    "mac_plast_w":"sum",
    "mic_plas_w":"sum",
    "area":"sum",
    "length":"sum",
    "num_parts_other":"sum",
    "num_parts_staff":"sum",
    "time_minutes":"sum"
}
}

# the table of cumulative values
dims_table = fd_dims.groupby(["location"]).agg(agg_for_table )

# collect the number of samples from the survey total data:
```

```

for name in dims_table.index:
    dims_table.loc[name, "samples"] = fd_dt[fd_dt.location == name].loc_date.nunique()

# get the sum of all the samples
dims_table.loc[level_names[0]] = dims_table.sum(numeric_only=True, axis=0)

# take the median pcs_m of all the samples
dims_table.loc[level_names[0], unit_label] = fd_dt.groupby(["location"])
[unit_label].sum().median()

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

# german column names for the dimensional data table
new_col_names={
    "samples": "Erhebungen",
    "quantity": "Objekte",
    "total_w": "Gesamt-e kg",
    "mac_plast_w": "kg Plastik",
    "mic_plas_w": "Gesamt-s kg",
    "area": "m2",
    "length": "Länge",
    "num_parts_other": "Teilnehmer",
    "num_parts_staff": "Mitarbeiter",
    "time_minutes": "Std"
}
}

# order the columns
dims_table.rename(columns=new_col_names, inplace=True)

# kilos, these weights are recorded in grams
kilos = ["kg Plastik", "Gesamt-s kg"]
dims_table[kilos] = (dims_table[kilos]/1000).round(2)

# numerical type and rounding
tints = ["Erhebungen", "Gesamt-e kg", "m2", "Länge", "Objekte", "Mitarbeiter",
"Teilnehmer"]
twodec = [unit_label]

# apply formatting
dims_table[tints] = dims_table[tints].astype("int")
dims_table[twodec] = dims_table[twodec].round(2)
dims_table["Std"] = (dims_table["Std"]/60).round(1)

# apply string formatting
dims_table.reset_index(inplace=True)
table_one = dims_table[["location", "Erhebungen", "Objekte", unit_label, "kg Plastik",
"Gesamt-s kg", "m2", "Länge"]].copy()
commas = ["Erhebungen", "Objekte", "m2", "Länge", unit_label]
table_one.loc[:, commas[:-1]] = table_one.loc[:, commas].applymap(lambda
x:thousandsSeparator(int(x), language))

# make table
fig, ax = plt.subplots(figsize=(14, 13))

```

```
sut.hide_spines_ticks_grids(ax)
a_table = sut.make_a_table(ax,table_one.values , colLabels=table_one.columns,
colWidths=[.23, *[.11]*7], bbox=[0, 0, 1, 1], bottom_row=True)
a_table.get_celld()[(0,0)].get_text().set_text(" ")
a_table.set_fontsize(12)
glue('alpes_survey_area_dimensional_summary', fig, display=False)
plt.close()
```

	Erhebungen	Objekte	p/100 m	kg Plastik
Die Alpen	20	7 776	203.0	6.35
veysonnaz	1	4 981	41508.33	0.12
les-crosets	1	518	551.06	0.68
san-bernardino	1	431	8620.0	0.8
airolo	1	224	589.47	0.14
verbier	1	205	410.0	0.16
nendaz	1	182	284.38	0.35
charmey	1	174	580.0	0.91
grindelwald	1	169	277.05	0.15
cabanes-des-diablerets	1	165	1375.0	0.02
morgins	1	123	164.0	0.23
les-paccots	1	118	245.83	0.09
villars	1	105	95.45	0.3
la-tzoumaz	1	88	110.0	0.74
val-calanca	1	78	97.5	0.06
la-robella	1	47	58.75	0.28
crans-montana	1	47	109.3	0.36
la-berra	1	34	56.67	0.01
monte-generoso	1	32	41.56	0.1
les-diablerets	1	31	26.27	0.75
andermatt	1	24	24.0	0.1

Abb. 2.2 #

Abbildung 2.2: Die aggregierten Ergebnisse der Abfallerhebungen. Ein Teil der Daten befindet sich aus Platzgründen in einer zweiten Tabelle darunter.

2.2.2. Gesamtzahlen in Bezug auf die Clean-upsGesamtzahlen in Bezug auf die Clean-ups#

```
# table two event totals
table_two = dims_table[["location", "Gesamt-e kg", "Teilnehmer",
"Mitarbeiter", "Std"]].copy()
table_two["Gesamt kg"] = table_two["Gesamt-e kg"].map(lambda
x:thousandsSeparator(int(x), language))
table_two = table_two[["location", "Gesamt kg", "Teilnehmer", "Mitarbeiter", "Std"]]

# make a table
fig, axs = plt.subplots(figsize=(10,13))
sut.hide_spines_ticks_grids(axs)

a_table = sut.make_a_table(axs, table_two.values, colLabels=table_two.columns,
colWidths=[.44, *[.14]*4])
a_table.get_celld()[(0,0)].get_text().set_text(" ")
a_table.set_fontsize(12)

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

		Gesamt kg	Teilnehmer
	Die Alpen	3 120	865
	veysonnaz	48	12
	les-crosets	205	50
	san-bernardino	9	12
	airolo	1	0
	verbier	76	27
	nendaz	155	70
	charmey	235	72
	grindelwald	26	21
	cabanes-des-diablerets	731	20
	morgins	129	60
	les-paccots	160	90
	villars	256	125
	la-tzoumaz	200	50
	val-calanca	228	32
	la-robella	37	23
	crans-montana	185	35

Abb. 2.3 #

Abbildung 2.3: Die Gesamtmenge des gesammelten Mülls in Kilogramm, die Anzahl der Teilnehmenden und des Personals sowie die Zeit, die für die Durchführung der Erhebung benötigt wurde.

2.2.3. 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*

```
# explanatory variables:  
luse_exp = list(sut.luse_ge.values())  
  
# columns needed  
use_these_cols = ["loc_date", *luse_exp, "groupname", "code"]  
  
# the land use data from all other locations  
datax = sd.groupby(use_these_cols[:-2], as_index=False).agg(agg_pcs_quantity)  
  
# work off the copy  
data = fd.groupby(use_these_cols[:-2], as_index=False).agg(agg_pcs_quantity)  
  
sns.set_style("whitegrid")  
  
fig, axs = plt.subplots(2, 3, figsize=(9,8), sharey="row")  
  
for i, n in enumerate(luse_exp):  
    r = i%2  
    c = i%3  
    ax=axs[r,c]  
    # get the empirical distribution of the independent variable  
    all_surveys = ECDF(datax[n].values)  
    les_alpes = ECDF(data[n].values)  
  
    # plot that  
    sns.lineplot(x=all_surveys.x, y=all_surveys.y, ax=ax, color=a_color,
```

```

label=level_names[1])
    # plot that
    sns.lineplot(x=les_alpes.x, y=les_alpes.y, ax=ax, color="magenta",
label=level_names[0])
    the_median = data[n].median()

    # get its position reference the surrounding survey area
    a = (stats.percentileofscore(les_alpes.x, the_median))/100

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

    # save the handles and labels but remove them from the ax
    handles, labels = ax.get_legend_handles_labels()
    ax.get_legend().remove()

    # format the % of total on the xaxis:
    if i <= 3:
        if c == 0:
            ax.set_ylabel("Prozent der Standorte", **ck.xlab_k)
            ax.xaxis.set_major_formatter(ticker.PercentFormatter(1.0, 0, "%"))
        else:
            pass
    ax.set_xlabel(n, **ck.xlab_k)

plt.tight_layout()
plt.subplots_adjust(top=.9, hspace=.3)
plt.suptitle("Landnutzung im Umkreis 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('alpes_survey_area_landuse', fig, display=False)
plt.close()

```

Landnutzung im Umkreis von 1 500 m um den Erhebungsort

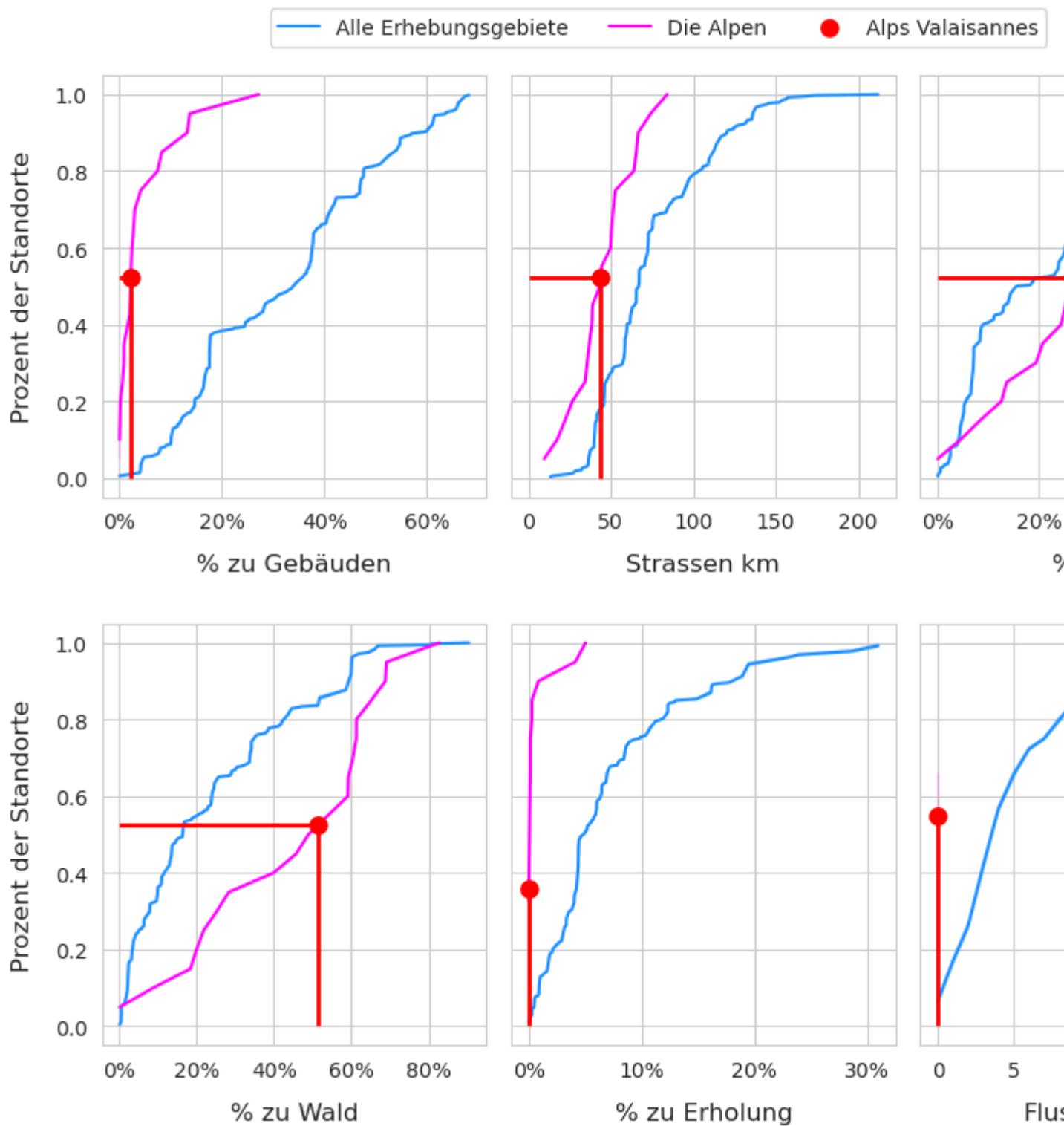


Abb. 2.4 #

Abbildung 2.4: Die Erhebungsorte in den Alpen und im Jura wiesen im Vergleich zu den Ergebnisorten IQAASL einen höheren Prozentsatz an forst- und landwirtschaftlichen Flächen und einen geringeren Prozentsatz an bebauter Fläche (Gebäude) und an Fläche, die für Aktivitäten im Freien genutzt werden, auf.

Die aggregierten Ergebnisse zeigen den Unterschied zwischen den beiden Erhebungsmethoden. Die drei Erhebungsorte mit dem höchsten p/100 m haben auch die kürzeste Länge. Im Fall von Cabanes-des-Diablerets entspricht die Fläche (in m²) der Länge (in m), was darauf hindeutet, dass ein kleiner Bereich um eine Struktur oder ein Gebäude herum vermessen wurde. In Veysonnaz befindet sich die Talstation

der Seilbahn nach Thyon (Wintersportgebiet Veysonnaz / 4 Vallées).

Der Unterschied in den Methoden führt zu abweichenden Ergebnissen. Ausserdem wurden diese beiden Orte aufgrund der früheren Erfahrungen der Person, die die Erhebung ausführt, speziell für die Bestandsaufnahme ausgewählt. Wegen der unterschiedlichen Dimensionen und Methoden werden die Erhebungsergebnisse aus Veysonnaz, San-Beranardino und Cabanes-des-Diablerets in der weiteren Analyse nicht berücksichtigt.

2.3. Verteilung der Erhebungsergebnisse¶#

```
remove = ["veysonnaz", "cabanes-des-diablerets", "san-bernardino"]

# the feature data without the three locations
wt_data = fd[~fd.location.isin(remove)].copy()

# all the data without the three locations
a_data = a_data[~a_data.location.isin(remove)]

wt_dt = fd_dt[~fd_dt.location.isin(remove)].copy()

nvsn = wt_dt.location.unique()

# make a df of survey totals with date as index
# the daily survey totals of all the data for the survey period
a_dt = a_data.groupby(["loc_date", "date", "location"],
as_index=False).agg(agg_pcs_quantity)

# only the surveys from all other survey areas
dts_date = a_dt[(~a_dt.location.isin([*nvsn, *remove]))].copy()

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

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

ax = axs[0]

# there is a big value in here, that should be seen.
sns.scatterplot(data=dts_date, x="date", y=unit_label, color="black", alpha=0.4,
label=label_names[1], ax=ax)
sns.scatterplot(data=wt_dt, x="date", y=unit_label, color="red", s=34,
ec="white", label=label_names[0], ax=ax)

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

# ax.tick_params(axis="x", which="both", bottom=False)
ax.xaxis.set_major_formatter(months_fmt)

axtwo = axs[1]

box_props = {
    "boxprops": {"facecolor": "none", "edgecolor": "magenta"},
    "medianprops": {"color": "magenta"},
```

```

    "whiskerprops":{"color":"magenta"},
    "capprops":{"color":"magenta"}
}
sns.boxplot(data=dts_date, y=unit_label, color="black", ax=axtwo, showfliers=False,
**box_props, zorder=1)
sns.stripplot(data=dts_date[dts_date[unit_label] <= 1000], s=10, y=unit_label,
color="black", ax=axtwo, alpha=0.5, jitter=0.3, zorder=0)
sns.stripplot(data=wt_dt, y=unit_label, color="red", s=10, ec="white", linewidth=1,
ax=axtwo, jitter=0.3, zorder=2)

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

axtwo.tick_params(which="both", axis="x", bottom=False)

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

```

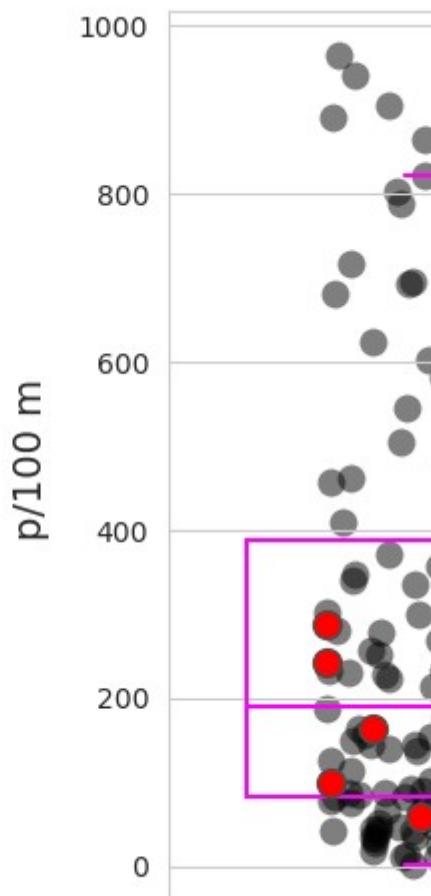
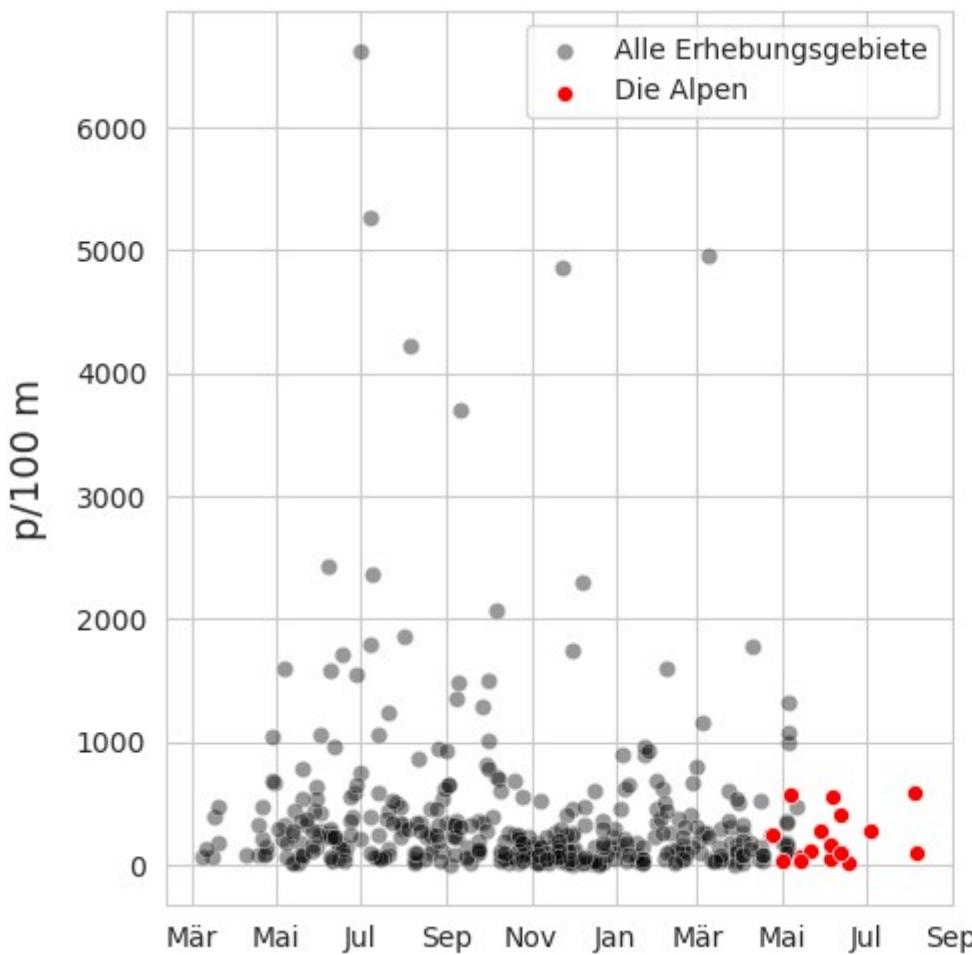


Abb. 2.5 #

Abbildung 2.5: Links: Zusammenfassung der Daten aller Erhebungen Erhebungsgebiet Alpes März 2020 bis September 2021, n = 20. Rechts: Gefundene Materialarten im Erhebungsgebiet Alpes in Stückzahlen und als prozentuale Anteile (stückzahlbezogen).

2.3.1. Zusammenfassende Daten und Materialtypen#

```
# get the basic statistics from pd.describe
```

```

cs = wt_dt[unit_label].describe().round(2)

# add project totals
cs["total objects"] = fd[~fd.location.isin(remove)].quantity.sum()

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

# format the text
combined_summary = sut.fmt_combined_summary(csx, nf=[])

# the material totals
code_totals = sut.the_aggregated_object_values(wt_data, agg=agg_pcs_median,
description_map=code_description_map, material_map=code_material_map)
code_totals.sort_values(by="quantity", ascending=False)

fd_mat_totals = sut.the_ratio_object_to_total(code_totals)

fd_mat_totals = sut.fmt_pct_of_total(fd_mat_totals)
fd_mat_totals = sut.make_string_format(fd_mat_totals)

# apply new column names for printing
cols_to_use = {"material":"Material", "quantity":"Quantity", "% of total": "% of
total"}
fd_mat_t = fd_mat_totals[cols_to_use.keys()].values

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

# summary table
# names for the table columns
a_col = [level_names[0], "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)

# column names for display
cols_to_use = {"material":"Material", "quantity":"Gesamt", "% of total": "% Gesamt"}

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(" ")
table_three.set_fontsize(12)

```

```

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

```

	total
Anzahl Proben	17
Durchschnitt p/100 m	219
Standardfehler	200
min p/100 m	24
25%	59
50%	110
75%	287
max p/100 m	595
Abfallobjekte total	2.199

	Gesamt	% G
Plastik	1.757	79
Metall	220	10
Glas	108	4
Papier	63	2
Stoff	25	1
Holz	16	0
Gummi	10	0
Chemikalien	0	0
Unbekannt	0	0

Abb. 2.6 #

Abbildung 2.6: Links: Zusammenfassung der erhebungen entlang der Wanderwege. Rechts: Aufschlüsselung nach Materialart (Stückzahlen und prozentuale Verteilung).

2.4. 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 Erhebungen identifiziert wurden (Häufigkeitsrate).

```

# code totals for les Alps not including veysnnaz
code_totals.rename(columns={"groupname":"utility"}, inplace=True)

# objects with a fail rate of > 50% in the survey area
most_common = code_totals[code_totals["fail rate"] >= 50].sort_values(by="quantity",
ascending=False)

```

```

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

# 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()

# 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: "{:,}").format(x)
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"{np.ceil(x)}")

# final table wt_data
cols_to_use = {"item":"Objekt", "quantity":"Gesamt", "% of total": "% Gesamt", "fail rate": "Ausfallsrate", unit_label:unit_label}
walking_trails = m_common[cols_to_use.keys()].values

# figure caption
rb_string = F"""
*__Unten:__ Häufigste Objekte auf Wanderwegen: Fail-Pass Rate >/= 50% und/oder Top
Ten nach Anzahl. Zusammengenommen stellen die zehn häufigsten
Objekte {int(m_common_percent_of_total*100)}% aller gefundenen Objekte der,
{unit_label}: Medianwert der Erhebung.*
"""
fig, axs = plt.subplots(figsize=(12, len(m_common)*.7))

sut.hide_spines_ticks_grids(axs)

table_three = sut.make_a_table(axs, walking_trails,
colLabels=list(cols_to_use.values()), colWidths=[.48, .13, .13, .13, .13],
bbox=[0,0,1,1], **{"loc":"lower center"})
table_three.get_celld()[(0,0)].get_text().set_text(" ")
table_three.set_fontsize(12)

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

```

	Gesamt
Seilbahnbürste	461
Zigarettenfilter	270
Fragmentierte Kunststoffstücke	222
Snack-Verpackungen	187
Schrauben und Bolzen	81
Kabelbinder	73
Getränke Glasflasche, Stücke	67
Verpackungen aus Aluminiumfolie	67
Kunststoff-oder Schaumstoffprodukte	60
Abdeckklebeband / Verpackungsklebeband	48

Abb. 2.7 #

Abbildung 2.7: Häufigste Objekte auf Wanderwegen: Objekte mit einer Häufigkeitsrate von mindestens 50 % und/oder die 10 häufigsten Objekte. Zusammengenommen stellen die zehn häufigsten Objekte 69 % aller gefundenen Objekte dar. p/100 m: Medianwert der Erhebung.

2.4.1. Die am häufigsten gefundenen Gegenstände nach Erhebungsort#

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

# 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", "location"]].pivot(columns="location",
index="item")

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

# the aggregated totals for the locations
m_c_p[level_names[0]] =
sut.aggregate_to_code(wt_data[wt_data.code.isin(m_common.index)],
code_description_map, name=level_names[0], unit_label=unit_label)

m_c_p[level_names[1]] =
sut.aggregate_to_code(a_data[a_data.code.isin(m_common.index)],
code_description_map, name=level_names[1], unit_label=unit_label)

# chart that
fig, ax = plt.subplots(figsize=(len(m_c_p.columns)*.8, len(m_c_p)*.9))

axone = ax
sns.heatmap(m_c_p, ax=axone, annot=True, vmax=300, annot_kws={"fontsize":12},
cmap=cmap2, fmt=".1f", square=True, cbar=False, linewidth=.1, linecolor="white")

axone.set_xlabel("")
axone.set_ylabel("")

axone.tick_params(labelsize=12, which="both", axis="both", labeltop=True,
labelbottom=False)
plt.setp(axone.get_xticklabels(), rotation=90)

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

```

	airolo	andermatt	charmey	crans-montana	grindelwald	la-berra
Abdeckklebeband / Verpackungsklebeband	21.0	1.0	17.0	5.0	0.0	0.0
Fragmentierte Kunststoffstücke	69.0	3.0	147.0	7.0	12.0	12.0
Getränke Glasflasche, Stücke	34.0	2.0	0.0	0.0	0.0	3.0
Kabelbinder	8.0	0.0	37.0	2.0	2.0	0.0
Kunststoff-oder Schaumstoffprodukte	0.0	0.0	0.0	0.0	0.0	0.0
Schrauben und Bolzen	3.0	0.0	7.0	9.0	3.0	0.0
Seilbahnbürste	0.0	0.0	0.0	0.0	0.0	0.0
Snack-Verpackungen	84.0	1.0	63.0	9.0	72.0	15.0
Verpackungen aus Aluminiumfolie	74.0	0.0	27.0	0.0	15.0	3.0
Zigarettenfilter	66.0	3.0	50.0	16.0	125.0	2.0

Abb. 2.8 #

Abbildung 2.8: Häufigste Objekte an Wanderwegen: Median p/100 m.

2.4.2. Seilbahnbürsten#



Abb. 2.9 #

Abbildung 2.9: Seilbahnbürsten, die verwendet werden, um Eis und Schnee von Skiliften zu entfernen, können sich von der Anlage lösen und Tausende von schweren Kunststofffäden erzeugen.*.

2.5. 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.

Unten: *Wanderwege Nutzen der gefundenen Objekte: % der Gesamtzahl nach Wassermerkmal. Fragmentierte Objekte, die nicht eindeutig identifiziert werden können, bleiben nach Grösse klassifiziert.*

```
# the median survey total and % of total for codegroups
cg_t = wt_data.groupby(["loc_date", "location", "groupname"],
as_index=False).agg(agg_pcs_quantity)

# aggregate all that for each water feature
cg_t = cg_t.groupby(["location", "groupname"],
as_index=False).agg({unit_label:"median", "quantity":"sum"})

# quantity per water feature
cg_tq = cg_t.groupby("location").quantity.sum()

# assign the water feature total to each record
for a_feature in cg_tq.index:
    cg_t.loc[cg_t.location == a_feature, "f_total"] = cg_tq.loc[a_feature]

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

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

# make a column for the survey area and all data
data_table[level_names[0]] = sut.aggregate_to_group_name(wt_data, name=level_names[0])
data_table[level_names[1]] = sut.aggregate_to_group_name(sd, name=level_names[1])

# chart that
fig, ax = plt.subplots(figsize=(len(data_table.columns)*.8, len(data_table)*.9))

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

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

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

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

	airolo	andermatt	charmey	crans-montana	grindelwald	la-berra
Abwasser	0%	0%	1%	0%	1%	0%
Essen und Trinken	50%	29%	16%	23%	34%	41%
Freizeit und Erholung	4%	0%	3%	11%	2%	9%
Infrastruktur	7%	21%	24%	21%	2%	9%
Landwirtschaft	0%	4%	1%	2%	0%	0%
Mikroplastik (< 5mm)	4%	0%	0%	9%	8%	0%
Nicht-Lebensmittelverpackungen	5%	4%	7%	2%	0%	0%
Persönliche Gegenstände	4%	8%	8%	6%	1%	18%
Plastikfragmente	12%	12%	25%	6%	4%	21%
Tabakwaren	12%	12%	11%	17%	47%	3%
nicht klassifiziert	0%	8%	5%	2%	1%	0%

Abb. 2.10 #

Abbildung 2.10: Verwendungszweck der gefundenen Objekte an Wanderwegen: prozentualer Anteil an der Gesamtzahl nach Verwendungszweck. Fragmentierte Objekte, die nicht eindeutig identifiziert werden können, bleiben nach Grösse klassifiziert.

```
# median p/50m solve cg_t for unit_label
data_table = cg_t.pivot(columns="location", index="groupname", values=unit_label)

# survey area median
data_table[level_names[0]] = sut.aggregate_to_group_name(fd, name=level_names[0],
val="med", unit_label=unit_label)

# all survey area median
data_table[level_names[1]] = sut.aggregate_to_group_name(sd, unit_label=unit_label,
name=level_names[1], val="med" )
```

Abbildung 2.11: Unten: Verwendungszweck der gefundenen Objekte an Wanderwegen: Median p/100 m.

	airolo	andermatt	charmey	crans-montana	grindelwald	la-berra
Abwasser	3	0	3	0	4	0
Essen und Trinken	296	7	93	25	95	23
Freizeit und Erholung	24	0	20	11	6	6
Infrastruktur	42	5	137	23	7	5
Landwirtschaft	3	1	3	2	0	0
Mikroplastik (< 5mm)	26	0	0	9	21	0
Nicht-Lebensmittelverpackungen	30	1	40	2	0	0
Persönliche Gegenstände	25	2	47	7	4	10
Plastikfragmente	69	3	147	7	12	12
Tabakwaren	74	3	63	18	130	2
nicht klassifiziert	3	2	27	2	2	0

Abb. 2.11 #

2.5.1. Perzentil-Rangfolge der Erhebungsergebnisse in Bezug auf die Landnutzung#

get the percentile ranking under each condition:

```

# define land use ranges based on the sample data
l_f = ["% zu LWS", "% zu Wald", "population"]

this_range = (fd[l_f[0]].min(), fd[l_f[0]].max())
this_range_w = (fd[l_f[1]].min(), fd[l_f[1]].max())
this_range_p = (fd[l_f[2]].min(), fd[l_f[2]].max())

# apply them to all the data
# one test for agg -- a dominant land use feature of the sample data
some_data = a_data[(a_data[l_f[0]] >= this_range[0])&(a_data[l_f[0]] <=
this_range[1])].copy()

# one test for woods -- the dominant land use feature of the sample data
some_data_w = a_data[(a_data[l_f[1]] >= this_range_w[0])&(a_data[l_f[1]] <=
this_range_w[1])].copy()

# one test for population --
some_data_p = a_data[(a_data[l_f[2]] >= this_range_p[0])&(a_data[l_f[2]] <=
this_range_p[1])].copy()

# remove Alps valaisannes
some_data = some_data[~some_data.location.isin(fd.location.unique())].copy()
some_data_w = some_data_w[~some_data_w.location.isin(fd.location.unique())].copy()
some_data_p = some_data_p[~some_data_p.location.isin(fd.location.unique())].copy()

# the number of samples and locations that have similar land use profiles as AV:
# agg to loc_date for each criteria
# data for charting and comparing
data=some_data.groupby(["loc_date","location",l_f[0]], as_index=False)
[unit_label].sum()
data_w =some_data_w.groupby(["loc_date","location",l_f[1]], as_index=False)
[unit_label].sum()
data_p = some_data_p.groupby(["loc_date","location",*l_f[1:]], as_index=False)
[unit_label].sum()

# get the percentile ranking for each location under each condition:
table_data = {}
for i,x in enumerate(nvsn):
    this_num = wt_dt.loc[wt_dt.location == x, unit_label].values[0]
    a = (stats.percentilefscore(data[unit_label].to_numpy(), this_num))
    b = (stats.percentilefscore(data_p[unit_label].to_numpy(), this_num))
    c = (stats.percentilefscore(data_w[unit_label].to_numpy(), this_num))
    table_data.update({x:{'Landwirtschaftlich':a, 'Wald':b, "population":c}})

# make df and format
t_data = pd.DataFrame(table_data)
t_data = t_data.astype("int")
t_data.reset_index(inplace=True)
t_data.rename(columns={"index":"variable"}, inplace=True)
t_data.set_index("variable", inplace=True, drop=True)

fig, ax = plt.subplots(figsize=(len(nvsn)*.8,5))

axone = ax
sns.heatmap(t_data , ax=axone, cmap=cmap2, vmax=300, annot=True,

```

```

annot_kws={"fontsize":12}, fmt="g", cbar=False, linewidth=.1, square=True,
linecolor="white")

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

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

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

plt.close()

```

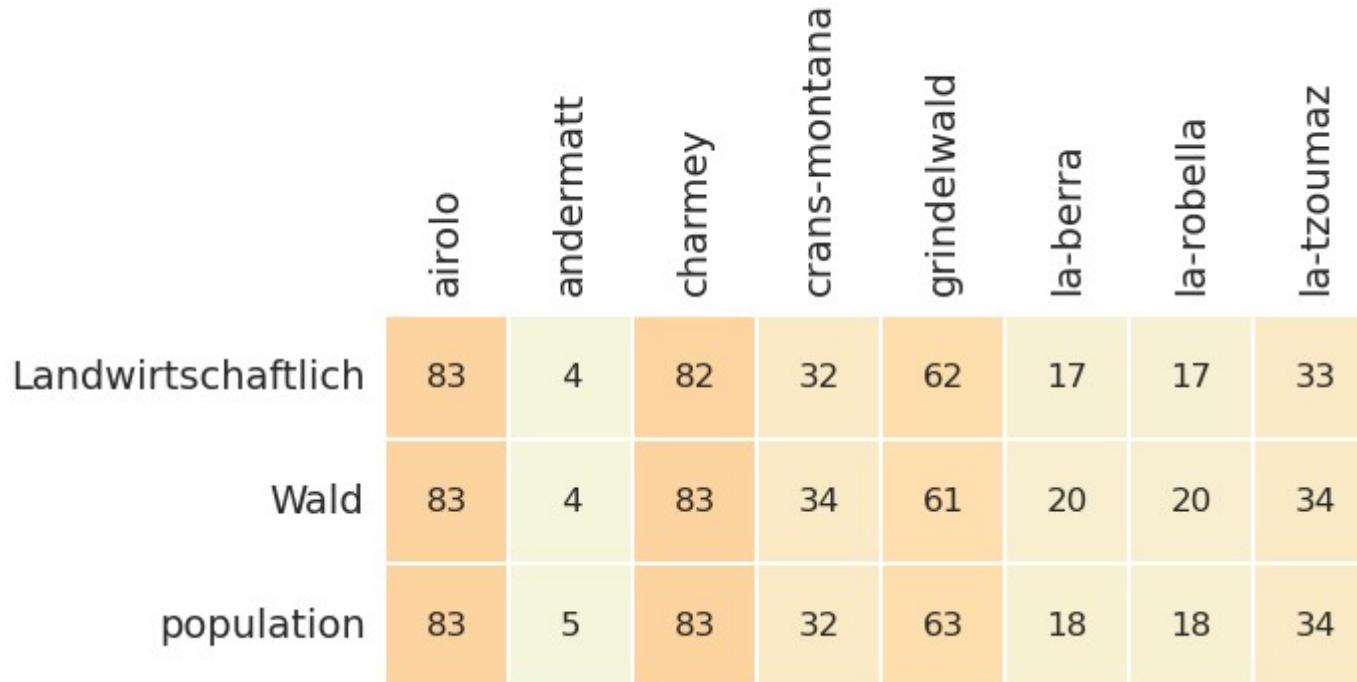


Abb. 2.12 #

Abbildung 2.12: Die Rangfolge der Erhebungsorte in den Alpen und im Jura in Bezug auf die Landnutzung. Die Erhebungsergebnisse in Airolo waren zum Beispiel höher als 83 % aller Erhebungen (Seen, Fliessgewässer, Alpen und Jura). In Andermatt liegen die Erhebungsergebnisse unter 95 % aller Erhebungen mit einem vergleichbaren Landnutzungsprofil.

2.6. Diskussion#

2.6.1. Vergleich der Ergebnisse: Alpen und Jura versus Seen und Fliessgewässer#

Der Medianwert beträgt 110 p/100 m für die 17 Erhebungsorte, die die Kriterien für Länge und Breite im Erhebungsgebiet Alpen und Jura erfüllen, und liegt damit unter dem Medianwert aller anderen Erhebungsgebiete (189 p/100 m). Objekte, die mit dem Konsum von Nahrungsmitteln, Getränken und Tabakwaren in Verbindung stehen, machten einen geringeren Prozentsatz der Gesamtzahl aus und wiesen eine niedrigere p/100 m-Rate auf als Erhebungsorte entlang von Wassersystemen. Dieser Unterschied könnte zum Teil auf die geringe Verstädterung zurückzuführen sein, die das Erhebungsgebiet Alpen und Jura im Vergleich zu allen anderen Erhebungsgebieten kennzeichnet.

Der Anteil von Objekten, die mit der Infrastruktur zusammenhängen, ist mit 36 % doppelt so hoch wie in

allen Untersuchungsgebieten zusammen. Dies ist grösstenteils auf die Fäden von Seilbahnbursten zurückzuführen, die in Les-Crosets in grossen Mengen gefunden wurden. Seilbahnbursten werden verwendet, um den Schnee von der Oberseite der abgedeckten Seilbahnkabinen zu entfernen, wenn diese sich dem Einstiegsort nähern. Ähnlich wie Industriepellets oder Schaumstoffkügelchen in der aquatischen Umwelt werden Teile von Seilbahnbursten wahrscheinlich immer wieder in gelegentlich grossen Mengen an ganz bestimmten Orten gefunden.

Das Verhältnis von Objekten, die mit der Infrastruktur zusammenhängen zu solchen, die mit dem Konsum von Lebensmitteln und Tabakwaren in Verbindung stehen, ist fast 1:1. Solche Ergebnisse sind typisch für Umgebungen mit einer besser entwickelten Infrastruktur, siehe [Gemeinsame Verantwortung] (Verkehr). Fragmentierte Kunststoffe werden in ähnlichem Umfang wie in den anderen Untersuchungsgebieten gefunden. Neu auf der Liste der häufigsten Objekte sind Kabelbinder und Abdeckband. Beide Objekte werden auch an Seen und Fliessgewässern gefunden, allerdings mit einer Häufigkeit von unter 50 %.

Es sei daran erinnert, dass diese Erhebungen entlang von Wintersportinfrastrukturen (Pisten, Wartebereiche bei den Liften etc.) oder eines Wanderweges in einem Skigebiet durchgeführt wurden. Auch wenn die Nutzung im Winter erhöht sein mag, sind viele Gebiete auch im Sommer hervorragende Wandergebiete, so dass eine ganzjährige Nutzung dieser Regionen möglich ist.

2.6.1.1. Die am häufigsten gefundenen Objekte#

Die am häufigsten vorkommenden Objekte machen 74 % sämtlicher gefundener Objekte aus. Die Zigarettenstummel lagen im Erhebungsgebiet Alpen und Jura nicht über dem nationalen Median, allerdings wurden in Verbier, Grindelwald und Airolo signifikante Werte festgestellt. Zudem sind spezifische Objekte aus der Gruppe der Infrastruktur vertreten, wie z. B.:

1. Schrauben und Bolzen
2. Kabelbinder
3. Abdeckband
4. Seilbahnburste

Das Fehlen von expandiertem oder extrudiertem Polystyrol in der Liste der am häufigsten vorkommenden Objekte im Erhebungsgebiet Alpen und Jura steht in scharfem Kontrast zu den anderen Erhebungsgebieten, in denen expandiertes oder extrudiertes Polystyrol etwa 13 % der Gesamtmenge ausmacht, siehe [Seen und Flüsse](#).

2.6.2. Implementierung von Abfallerhebungen in das bestehende Geschäftsmodell#

Im Vergleich zu einer Abfallerhebung an Seen und Fliessgewässern deckt eine Aufräumaktion ein relativ grosses geographisches Gebiet ab. Freiwillige, die an einem solchen Clean-up teilnehmen, werden von der Möglichkeit angezogen, sich um die Umwelt zu kümmern und sich in der Gesellschaft anderer (in den Bergen) zu bewegen. Erhebungen zu Abfallobjekten an Seen und Fliessgewässern bieten nicht dasselbe Aktivitätsniveau und sind möglicherweise nicht für alle Freiwilligen von Interesse.

Wer Abfallerhebungen durchführt, gibt Freiwilligen die Möglichkeit, vor Ort Erfahrungen zu sammeln, muss aber auch intern die Ressourcen bereitstellen, um sicherzustellen, dass die Untersuchung gemäss dem Protokoll durchgeführt wird. Dazu gehören das Identifizieren, Zählen und Eingeben von Daten. Die Summit Foundation war in der Lage, dies zu tun, indem sie dafür sorgte, dass bei jedem Clean-up eine Person anwesend war, die die Erhebung durchführen konnte.

Die Personen, die die Erhebung ausführten, zogen es vor, die Proben entlang der Wintersportinfrastrukturen zu nehmen und bei den Bergstationen zu beginnen. Die auf diese Weise entnommenen Proben folgen dem Verlauf des Clean-ups: bergab und in den Bereichen mit hohem

Verkehrsaufkommen.

Proben, die in der Nähe von Gebäuden oder anderen Einrichtungen genommen wurden, ergaben höhere Erhebungsergebnisse. Damit bestätigte sich, was die Mitglieder der Summit Foundation in den vergangenen Jahren festgestellt hatten. Aus diesen Erfahrungen erklärte der Projektleiter, Téo Gürsoy:

Die Personen, die Erhebung ausführen, konzentrieren sich nämlich hauptsächlich auf die Abschnitte unter den Sesselliften, Gondeln oder bei der Abfahrt und Ankunft dieser Anlagen, die stark frequentierte Orte sind.

In einigen Fällen ist die Dichte der Objekte so gross, dass sich die Person, welche die Erhebung ausführt, gezwungen sah, sich auf einen Bereich zu konzentrieren. Téo Gürsoy beschrieb, was passierte, als eine Person, welche die Erhebung ausführt, auf einen Ort stiess, der grosse Mengen von Skiliftbürsten enthielt:

Die Person, welche die Erhebung ausführt, begann den Streckenabschnitt [...] an der Ankunftsstation der Gondel. Die Skiliftbürsten erregten schnell die Aufmerksamkeit der Person, die beschloss, sich nur auf den betroffenen Bereich zu konzentrieren, um herauszufinden, wie viele von ihnen zu finden waren.

Die Erhebungsergebnisse rund um Infrastruktur oder Gebäude sind kein Indikator für den Zustand der Umwelt im gesamten Gebiet. Erhebungen in der Umgebung dieser Strukturen weisen tendenziell höhere Werte auf, machen aber nur einen kleinen Teil der gesamten Landnutzung aus.

Es mussten Anpassungen an der Software und dem Berichtsschema vorgenommen werden, um die verschiedenen Arten von Daten zu verarbeiten, die bei Aufräumarbeiten anfallen. Dazu gehörte auch die Schaffung neuer Identifikationscodes für bestimmte Objekte, die im Untersuchungsgebiet Alpen und Jura gefunden werden. Ausserdem stellte die Summit Foundation die Ressourcen zur Verfügung, damit ein Mitarbeiter der Stiftung in der Anwendung des Projektprotokolls und der Software geschult werden konnte.

2.6.2.1. Schlussfolgerungen#

Die Erhebungen, die entlang der Wege und Wintersportinfrastrukturen im Untersuchungsgebiet Alpen und Jura durchgeführt wurden, ergaben Daten, die den Daten der Erhebungen entlang von Seen und Fliessgewässern sehr ähnlich waren. Wenn sich die Personen, die die Erhebungen durchgeführt haben, jedoch auf bestimmte Infrastruktureinrichtungen konzentrierten, wurden extreme Werte ermittelt. Die Erhebungen Seen und Fliessgewässern würden zu den gleichen Ergebnissen führen, wenn die Erhebungen nur an Orten durchgeführt würden, an denen einen hohen Anteil an Abfallobjekten wahrscheinlicher sind.

Objekte aus dem Bereich Essen und Trinken machen nur 11 % der insgesamt gefundenen Objekte aus, verglichen mit 36 % in den anderen Untersuchungsgebieten. Der Anteil an Abfallobjekten aus dem Bereich Infrastruktur beträgt in den Alpen und im Jura jedoch 75 % gegenüber 18 % in allen anderen Untersuchungsgebieten. Dies ist zum Teil auf den Unterschied in der menschlichen Präsenz im Vergleich zu Orten in niedrigeren Höhenlagen zurückzuführen, wo die menschliche Präsenz das ganze Jahr über konstant ist, so dass der Druck durch Nahrungs- und Genussmittel im Gegensatz zur Infrastruktur grösser ist.

Dieses erste Projekt hat auch gezeigt, dass es möglich ist, die Überwachung mit Clean-ups zu kombinieren. In Vorbereitung auf die Überwachung tauschten die Mitglieder beider Teams Ideen aus und sortierten gemeinsam Proben. Dies ermöglichte es beiden Organisationen, sich gegenseitig besser zu verstehen und Basisleistungen zu bestimmen, die bei der Datenerfassung für einen nationalen Bericht erbracht werden konnten:

1. Unterstützung bei der Erfassung und Identifizierung von Abfallobjekten
2. Unterstützung bei der Dateneingabe

3. Erstellung von Diagrammen, Grafiken und Daten, die von den teilnehmenden Organisationen verwendet werden können

Eine Erhebung von Abfällen an Seen und Fliessgewässern dauert 2–4 Stunden, je nachdem, wie viele verschiedene Objekte es gibt. Diese Ressourcen waren im Betriebsbudget der beiden Organisationen nicht vorgesehen. Daher stellte die Summit Foundation die Koordination und Infrastruktur zur Verfügung und Hammerdirt eine zusätzliche Person, die die Erhebung ausführt, sowie IT-Unterstützung.

Die zur Verfügung gestellten Daten ermöglichen direkte Vergleiche zwischen den Orten, vorausgesetzt, es wird die gleiche Erhebungsmethode verwendet. Eine grosse Anzahl von Abfallobjekten mit Infrastrukturbzug im Vergleich zu Objekten aus dem Bereich Lebensmittel und Tabakwaren ist typisch für ländliche Gebiete. Wie gut die Daten aus dem Erhebungsgebiet Alpen und Jura mit jenen an Seen und Fliessgewässern vergleichbar sind, muss noch weiter untersucht werden. Zigarettenstummel, Glasscherben, Plastiksplitter und Snack-Verpackungen gehören jedoch zu den häufigsten Objekten, die in Wassernähe gefunden werden.

Wir danken allen Mitgliedern der Summit Foundation für ihre Hilfe, insbesondere Olivier Kressmann und Téo Gürsoy.

2.7. Anhang#

2.7.1. Schaumstoffe und Kunststoffe nach Grösse#

Die folgende Tabelle enthält die Komponenten “Gfoam” und “Gfrags”, 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 survey area
h=pd.read_csv("resources/checked_alpes_survey_data_be.csv")

# remove prefix
h["location"] = h["location"].map(lambda x: sut.get_rid_of_ix(x, prefix="clean-up-tour-"))

# remove prefixes from survey data
lAlps = h[h.river_bassin == bassin_name]
[[["loc_date", "code", "location", "river_bassin", "groupname", "quantity",
"pcs_m"]].copy()

# convert to reporting unit
lAlps[unit_label] = (lAlps.pcs_m*100).round(2)

# the fragmented plastics and foams
some_foams = ["G81", "G82", "G83", "G74"]
some_frag_plas = list(lAlps[lAlps.groupname == "plastic pieces"].code.unique())

# get just the foams and plastics and aggregate to code
conditions = ((lAlps.code.isin([*some_frag_plas,
*some_foams]))&(~lAlps.location.isin(remove)))
fd_frags_foams = lAlps[conditions].groupby(["loc_date", "code"],
as_index=False).agg(agg_pcs_quantity)
fd_frags_foams = fd_frags_foams.groupby("code", as_index=False).agg(agg_pcs_median)

# add code description and format for printing
fd_frags_foams["item"] = fd_frags_foams.code.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"]].copy()
data.rename(columns={"quantity": "Gesamt", "% of total": "% Gesamt"}, inplace=True)

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

this_table = sut.make_a_table(axs, data.values, colLabels=data.columns,
colWidths=[.6, .13, .13, .13], bbox=[0, 0, 1, 1])
this_table.get_celld()[(0,0)].get_text().set_text(" ")
this_table.set_fontsize(12)

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



Abb. 2.13 #

Abbildung 2.13: fragmentierte Schaumstoffe und Kunststoffe nach Grössengruppen. Median p/100 m, Anzahl Objekte, Prozent der Gesamtmenge.

2.7.2. Landnutzungsprofil der Erhebungsorte#

```
# get the land use profile of AV
lu_prof = fd[["location", "% zu Gebäuden", "% zu Erholung", "% zu Wald", "% zu LWS",
"population"]].drop_duplicates()

# format for printing
lu_prof.loc[:, lu_prof.columns[1:-1]] = lu_prof.loc[:, lu_prof.columns[1:-
1]].applymap(lambda x: F"{int((x*100))}%")
lu_prof.loc[:, lu_prof.columns[5:]] = lu_prof.loc[:, lu_prof.columns[5:]].applymap(lambda x: F"{int(x)}:,}")
```

```
# put that to a table
data=lu_prof.copy()

fig, axs = plt.subplots(figsize=(13, len(table_one)*.6))
sut.hide_spines_ticks_grids(axs)

this_table = sut.make_a_table(axs, data.values, colLabels=data.columns,
colWidths=[.22, *[.13]*6], bbox=[0, 0, 1, 1])
this_table.get_celld()[(0,0)].get_text().set_text(" ")
this_table.set_fontsize(12)

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

	% zu Gebäuden	% zu Erholung
cabanes-des-diablerets	0%	0%
val-calanca	2%	0%
san-bernardino	2%	0%
airolo	2%	0%
nendaz	1%	0%
veysonnaz	2%	0%
andermatt	0%	0%
verbier	27%	4%
crans-montana	0%	0%
villars	0%	5%
les-crosets	1%	0%
morgins	8%	0%
la-berra	0%	0%
grindelwald	0%	0%
la-tzoumaz	13%	0%
la-robella	2%	0%
les-diablerets	13%	0%

Abb. 2.14 #

Abbildung 2.14: Landnutzungsprofil der ErhebungsorteGesamtmenge. LWS = Landwirtschaft

2.7.3. Alpen und Jura in Bezug auf die Landnutzung#

Die Ergebnisse aus den Alpen und dem Jura werden mit den anderen Erhebungsergebnissen verglichen, die entweder % bis Wald oder % bis Landwirtschaft (LWS) innerhalb der gleichen Spanne wie AV liegen. Die Bereiche für AV sind:

- % zu LWS: 0 to 66%
- % zu Wald: 0 to 83%
- population: 199 to 10,668

```
# define land use ranges based on the sample data
this_range = (fd[l_f[0]].min(), fd[l_f[0]].max())
this_range_w = (fd[l_f[1]].min(), fd[l_f[1]].max())
this_range_p = (fd[l_f[2]].min(), fd[l_f[2]].max())

# apply them to all the data
# one test for agg -- a dominant land use feature of the sample data
some_data = a_data[(a_data[l_f[0]] >= this_range[0])&(a_data[l_f[0]] <=
this_range[1])].copy()

# one test for woods -- the dominant land use feature of the sample data
some_data_w = a_data[(a_data[l_f[1]] >= this_range_w[0])&(a_data[l_f[1]] <=
this_range_w[1])].copy()

# one test for population --
some_data_p = a_data[(a_data[l_f[2]] >= this_range_p[0])&(a_data[l_f[2]] <=
this_range_p[1])].copy()

# remove Alps valaisannes
some_data = some_data[~some_data.location.isin(fd.location.unique())].copy()
some_data_w = some_data_w[~some_data_w.location.isin(fd.location.unique())].copy()
some_data_p = some_data_p[~some_data_p.location.isin(fd.location.unique())].copy()

# the number of samples and locations that have similar land use profiles as AV:
# agg to loc_date for each criteria
# data for charting and comparing
data=some_data.groupby(["loc_date","location",l_f[0]], as_index=False)
[unit_label].sum()
data_w =some_data_w.groupby(["loc_date","location", l_f[1]], as_index=False)
[unit_label].sum()
data_p = some_data_p.groupby(["loc_date","location",l_f[1], "population"],
as_index=False)[unit_label].sum()
regional = fd.groupby(["loc_date","location", *l_f], as_index=False)
[unit_label].sum()

# locations that share the characteristics
commonsamps = set(data.loc_date.unique()) & set(data_w.loc_date.unique())&
set(data_p.loc_date.unique())
commonlocs = set(data.location.unique()) &
set(data_w.location.unique())&set(data_p.location.unique())
```

```

# print these out to get the comparison

# print("agg")
# print(this_range)
# print(len(data.location.unique()))
# print(data.loc_date.nunique())
# print("woods")
# print(this_range_w)
# print(len(data_w.location.unique()))
# print(data_w.loc_date.nunique())
# print("p")
# print(this_range_p)
# print(len(data_p.location.unique()))
# print(data_p.loc_date.nunique())
# print(len(commonamps))
# print(commonlocs)

# make a categorical df for mapping
mat_agg = dfBeaches.loc[data.location.unique()]
mat_agg["match"] = "agg"
mat_w = dfBeaches.loc[data_w.location.unique()]
mat_w["match"] = "woods"
mat_p = dfBeaches.loc[data_p.location.unique()]
mat_p["match"] = "pop"

# merge all that and export to .csv
landusemap = pd.concat([mat_agg, mat_w, mat_p], axis=0)
# landusemap.to_csv("output/Alps-valaisannes/lu_comparison.csv", index=False)

```

Oben links: Gesamtsumme der Erhebung in Bezug auf den %-Anteil an Agg, Bereich=(0%, 66%). **Oben rechts:** Gesamtzahl der Erhebungen in Bezug auf den Waldanteil, Bereich=(0%, 65%). **Unten links:** Gesamtzahl der Erhebungen in Bezug auf die Bevölkerung, Bereich=(199, 10.668)

```

fig, axs = plt.subplots(2,2, figsize=(10,8), sharey=True)

axone=axs[0,0]
axtwo=axs[0,1]
axthree=axs[1,0]
axfour=axs[1,1]

# plot the samples from all the data that meet the x criteria
sns.scatterplot(data=data, x=l_f[0], y=unit_label, color="black", alpha=1,
linewidth=1, label="All surveys", ax=axone, zorder=1)

# point estimates of the percentile ranking based off the edcf of all surveys
# place to store the rankings
rankings = {}

# plot the values for AV
for x in regional.location.unique():
    this_y = regional[regional.location == x][unit_label]
    this_x = regional[regional.location == x][l_f[0]]
    axone.scatter(this_x, this_y, color="red", s=60, zorder=2)

```

```

# handle extreme values
axone.set_ylim(0, max(data[unit_label].to_numpy()))

# set labels
axone.set_ylabel(unit_label, **ck.xlab_k14)
axone.set_xlabel(l_f[0], **ck.xlab_k14)

# gather up legend handles
axone.get_legend().remove()

# start axtwo
# plot the samples from all the data that meet the x criteria
sns.scatterplot(data=data_w, x=l_f[1], y=unit_label, color="black", alpha=1,
linewidth=1, label="All surveys", ax=axtwo, zorder=1)

# plot the values from AV
for x in regional.location.unique():
    this_y = regional[regional.location == x][unit_label]
    this_x = regional[regional.location == x][l_f[1]]
    rankings.update({x:(this_x, this_y)})
    axtwo.scatter(this_x, this_y, color="red", s=60, zorder=2)

# handle extreme values
axtwo.set_ylim(0, max(data[unit_label].to_numpy()))

# set labels
axtwo.set_ylabel(unit_label, **ck.xlab_k14)
axtwo.set_xlabel(l_f[1], **ck.xlab_k14)
axtwo.get_legend().remove()

# start axthree
# plot the samples from all the data that meet the x criteria
sns.scatterplot(data=data_p, x=l_f[2], y=unit_label, color="black", alpha=1,
linewidth=1, label=level_names[1], ax=axthree, zorder=1)

# plot the values from AV
for x in regional.location.unique():
    this_y = regional[regional.location == x][unit_label]
    this_x = regional[regional.location == x][l_f[2]]
    rankings.update({x:(this_x, this_y)})
    axthree.scatter(this_x, this_y, color="red", s=60, label=level_names[0],
zorder=2)

# handle extreme values
axthree.set_ylim(-100, max(data[unit_label].to_numpy()))

# start axfour,# clear axfour
sut.hide_spines_ticks_grids(axfour)

# set labels
axthree.set_ylabel(unit_label, **ck.xlab_k14)
axthree.set_xlabel(l_f[2], **ck.xlab_k14)
handles, labels = axthree.get_legend_handles_labels()
axthree.get_legend().remove()

```

```

fig.legend(handles[:2], labels[:2], bbox_to_anchor=(.75,.25), loc="lower center",
fontsize=12)
plt.tight_layout()
glue('alpes_survey_area_compare_luse', fig, display=False)
plt.close()

```

Abbildung 2.15: **Oben links:** Gesamtsumme der Erhebung in Bezug auf den prozentualen Anteil an landwirtschaftlich genutzter Fläche, Bereich = (0 %, 66 %). **Oben rechts:** Gesamtzahl der Erhebungen in Bezug auf den Waldanteil, Bereich = (0 %, 65 %). **Unten links:** Gesamtzahl der Erhebungen in Bezug auf die Bevölkerung, Bereich = (199, 10 668)

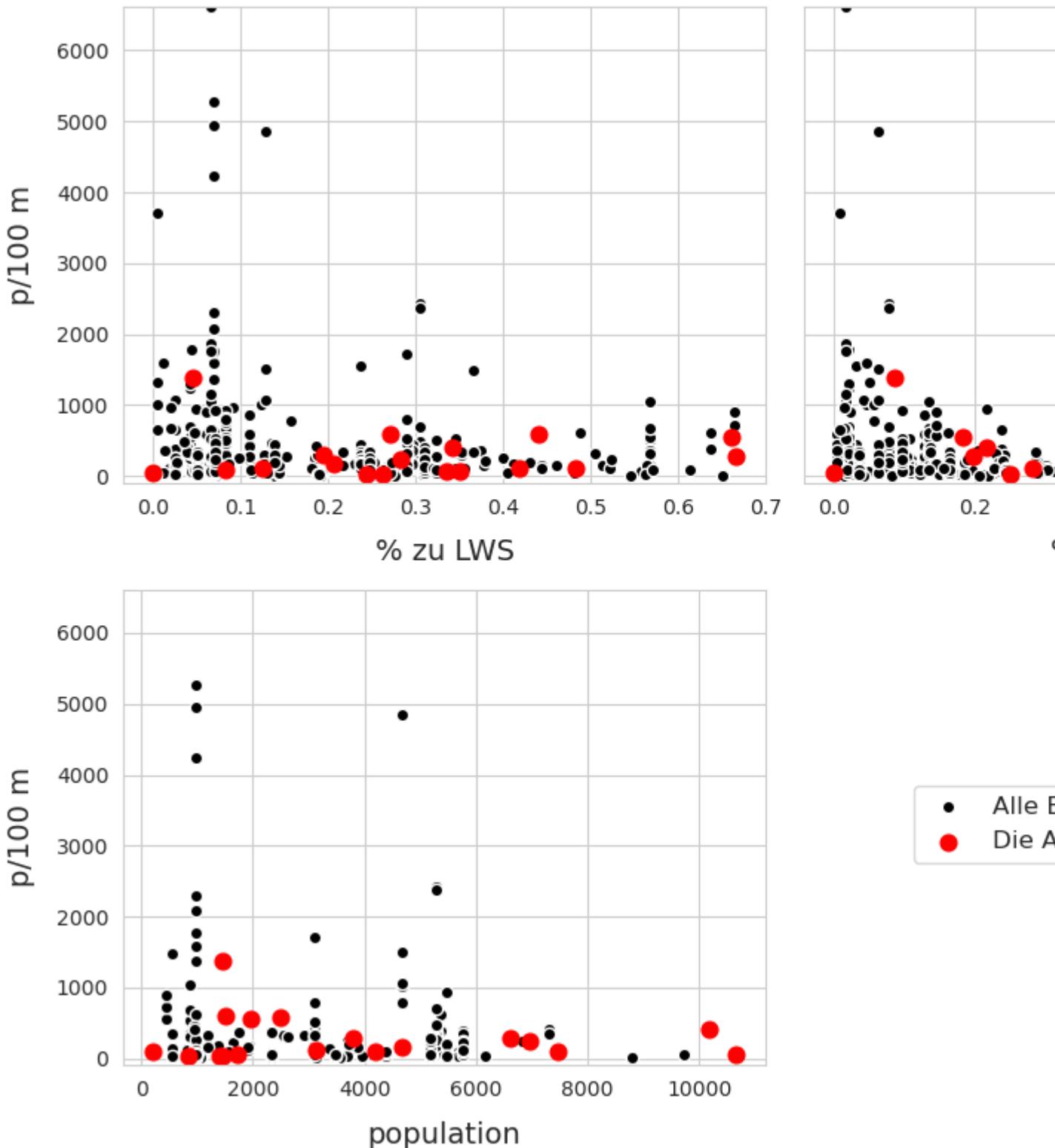


Abb. 2.15 #

2.7.4. Organisation und Durchführung#

Summit foundation: Téo Gürsoy

Hammerdirt: Bettina Siegenthaler

2.7.5. Die Erhebungsorte#

```
# display the survey locations
pd.set_option("display.max_rows", None)

disp_columns = ["latitude", "longitude", "city"]
disp_beaches = dfBeaches.loc[t["locations"]][disp_columns]
new_names = {"slug": "Standort", "city": "Stadt"}
disp_beaches.reset_index(inplace=True)
disp_beaches.rename(columns=new_names, inplace=True)
disp_beaches.set_index("Standort", inplace=True, drop=True)

disp_beaches
```

	latitude	longitude	Stadt
Standort			
cabanes-des-diablerets	46.338604	7.215525	Ormont-Dessus
val-calanca	46.330718	9.120103	Calanca
san-bernardino	46.445947	9.191344	Mesocco
airolo	46.514257	8.607884	Airolo
nendaz	46.162099	7.283486	Nendaz
veysonnaz	46.180334	7.366005	Nendaz
andermatt	46.618327	8.598803	Andermatt
verbier	46.103981	7.224760	Val de Bagnes
crans-montana	46.334214	7.479386	Lens
villars	46.321937	7.073841	Ollon
les-crosets	46.183591	6.832916	Val-d'Illiez
morgins	46.236681	6.858132	Troistorrents
la-berra	46.677399	7.178738	La Roche
grindelwald	46.658523	8.065323	Grindelwald
la-tzoumaz	46.144287	7.232897	Riddes

	latitude	longitude	Stadt
Standort			
la-robella	46.875905	6.548577	Val-de-Travers
les-diablerets	46.346141	7.151228	Ormont-Dessus
charmey	46.622641	7.189341	Val-de-Charmey
monte-generoso	45.929625	9.019921	Rovio
les-paccots	46.512592	6.949421	Châtel-Saint-Denis

2.7.6. Inventar der Objekte#

```
pd.set_option("display.max_rows", None)
complete_inventory = code_totals[code_totals.quantity>0][["item", "utility",
"quantity", "% of total","fail rate"]]

new_names = {"item":"Objekte", "groupname":"Gruppenname", "quantity":"Menge", "fail
rate":"Fail-Pass", "% of total": "% Gesamt", }

complete_inventory.rename(columns=new_names, inplace=True)
complete_inventory.sort_values(by="Menge", ascending=False)
```

	Objekte	utility	Menge	% Gesamt	Fail-Pass
code					
G704	Seilbahnbürste	Infrastruktur	461	20.96	5
G27	Zigarettenfilter	Tabakwaren	270	12.28	94
Gfrags	Fragmentierte Kunststoffstücke	Plastikfragmente	222	10.10	100
G30	Snack-Verpackungen	Essen und Trinken	187	8.50	100
G705	Schrauben und Bolzen	Infrastruktur	81	3.68	64
G93	Kabelbinder	Infrastruktur	73	3.32	64
G200	Getränke Glasflasche, Stücke	Essen und Trinken	67	3.05	41
G177	Verpackungen aus Aluminiumfolie	Essen und Trinken	67	3.05	64
G124	Kunststoff-oder Schaumstoffprodukte	nicht klassifiziert	60	2.73	5

	Objekte	utility	Menge	% Gesamt	Fail-Pass
code					
G87	Abdeckklebeband / Verpackungsklebeband	Infrastruktur	48	2.18	64
G941	Verpackungsfolien, nicht für Lebensmittel	Nicht-Lebensmittelverpackungen	37	1.68	23
G12	Kosmetika, Behälter für Körperpflegeprodukte, ...	Persönliche Gegenstände	34	1.55	5
G33	Einwegartikel; Tassen/Becher & Deckel	Essen und Trinken	31	1.41	35
G89	Kunststoff-Bauabfälle	Infrastruktur	31	1.41	11
G122	Kunststofffragmente (>1mm)	Mikroplastik (< 5mm)	30	1.36	29
G922	Etiketten, Strichcodes	Nicht-Lebensmittelverpackungen	27	1.23	47
G923	Taschentücher, Toilettenpapier, Servietten, Pa...	Persönliche Gegenstände	26	1.18	47
G204	Baumaterial; Ziegel, Rohre, Zement	Infrastruktur	22	1.00	35
G919	Nägel, Schrauben, Bolzen usw.	Infrastruktur	20	0.91	11
G178	Kronkorken, Lasche von Dose/Ausfreisslachen	Essen und Trinken	17	0.77	41
G934	Sandsäcke, Kunststoff für Hochwasser- und Eros...	Landwirtschaft	17	0.77	17
G926	Kaugummi, enthält oft Kunststoffe	Essen und Trinken	17	0.77	29
G34	Besteck, Teller und Tabletts	Essen und Trinken	16	0.73	23
G710	Ski / Snowboards (Ski, Befestigungen und ander...	Freizeit und Erholung	15	0.68	35
Gfoam	Expandiertes Polystyrol	Infrastruktur	14	0.64	17
G24	Ringe von Plastikflaschen/Behältern	Essen und Trinken	13	0.59	29
G901	Medizinische Masken,	Persönliche Gegenstände	13	0.59	23

	Objekte	utility	Menge	% Gesamt	Fail-Pass
code					
	synthetische				
G905	Haarspangen, Haargummis, persönliche Accessoires...	Persönliche Gegenstände	13	0.59	41
G198	Andere Metallteile < 50 cm	Infrastruktur	12	0.55	47
G707	Skiausrüstungsetikett	Freizeit und Erholung	11	0.50	23
G142	Seil, Schnur oder Netze	Freizeit und Erholung	11	0.50	41
G145	Andere Textilien	Persönliche Gegenstände	11	0.50	23
G146	Papier, Karton	Nicht-Lebensmittelverpackungen	10	0.45	23
G152	Papier &Karton;ZigarettenSchachteln, Papier/Ka...	Tabakwaren	10	0.45	23
G7	Getränkeflaschen < = 0,5 l	Essen und Trinken	10	0.45	41
G208	Glas oder Keramikfragmente >2.5 cm	nicht klassifiziert	10	0.45	23
G702	Pistenmarkierungsposten (Holz)	Infrastruktur	9	0.41	23
G709	Teller von Skistock (runder Teil aus Plastik u...)	Freizeit und Erholung	9	0.41	29
G66	Umreifungsbänder; Hartplastik für Verpackung f...	Infrastruktur	8	0.36	5
G31	Schleckstengel, Stengel von Lutscher	Essen und Trinken	8	0.36	23
G129	Schlüche und Gummiplatten	nicht klassifiziert	8	0.36	5
G921	Keramikfliesen und Bruchstücke	Infrastruktur	7	0.32	11
G123	Polyurethan-Granulat < 5mm	Mikroplastik (< 5mm)	6	0.27	5
G100	Medizin; Behälter/Röhrchen/Verpackungen	Abwasser	6	0.27	29

	Objekte	utility	Menge	% Gesamt	Fail-Pass
code					
G918	Sicherheitsnadeln, Büroklammern, kleine Gebrau...	Persönliche Gegenstände	6	0.27	17
G156	Papierfragmente	Nicht-Lebensmittelverpackungen	6	0.27	29
G25	Tabak; Kunststoffverpackungen, Behälter	Tabakwaren	5	0.23	17
G149	Papierverpackungen	Nicht-Lebensmittelverpackungen	5	0.23	11
G48	Seile, synthetische	Freizeit und Erholung	5	0.23	17
G20	Laschen und Deckel	Nicht-Lebensmittelverpackungen	5	0.23	17
G706	Skiabonnement	Freizeit und Erholung	5	0.23	29
G191	Draht und Gitter	Landwirtschaft	4	0.18	17
G943	Zäune Landwirtschaft, Kunststoff	Landwirtschaft	4	0.18	11
G915	Reflektoren, Mobilitätsartikel aus Kunststoff	Persönliche Gegenstände	4	0.18	5
G175	Getränkedosen (Dosen, Getränke)	Essen und Trinken	4	0.18	17
G73	Gegenstände aus Schaumstoff & Teilstücke (nich...	Freizeit und Erholung	4	0.18	5
G211	Sonstiges medizinisches Material	Persönliche Gegenstände	4	0.18	17
G194	Kabel, Metalldraht oft in Gummi- oder Kunststo...	Infrastruktur	3	0.14	11
G712	Skihandschuhe	Freizeit und Erholung	3	0.14	11
G23	unidentifizierte Deckel	Nicht-Lebensmittelverpackungen	3	0.14	5
G3	Einkaufstaschen,	Nicht-	3	0.14	11

	Objekte	utility	Menge	% Gesamt	Fail-Pass
code					
	Shoppingtaschen	Lebensmittelverpackungen			
G703	Pistenmarkierungsposten (Plastik)	Infrastruktur	3	0.14	5
G158	Sonstige Gegenstände aus Papier	Nicht-Lebensmittelverpackungen	3	0.14	17
G35	Strohhalme und Rührstäbchen	Essen und Trinken	3	0.14	17
G10	Lebensmittelbehälter zum einmaligen Gebrauch a...	Essen und Trinken	3	0.14	11
G171	Sonstiges Holz < 50cm	Landwirtschaft	3	0.14	17
G28	Stifte, Deckel, Druckbleistifte usw.	Persönliche Gegenstände	2	0.09	11
G931	(Absperr)band für Absperrungen, Polizei, Baust...	Infrastruktur	2	0.09	11
G181	Geschirr aus Metall, Tassen, Besteck usw.	Essen und Trinken	2	0.09	5
G713	Skiförderband	Infrastruktur	2	0.09	5
G155	Feuerwerkspapierhülsen und -fragmente	Freizeit und Erholung	2	0.09	5
G210	Sonstiges Glas/Keramik Materialien	nicht klassifiziert	2	0.09	5
G90	Blumentöpfe aus Plastik	Landwirtschaft	2	0.09	11
G26	Feuerzeug	Tabakwaren	2	0.09	11
G167	Streichhölzer oder Feuerwerke	Freizeit und Erholung	2	0.09	5
G8	Getränkeflaschen > 0,5L	Essen und Trinken	1	0.05	5
G101	Robidog Hundekot-Säcklein, andere Hundekotsäck...	Persönliche Gegenstände	1	0.05	5
G96	Hygienebinden/ Höscheneinlagen/Tampons und Appl...	Abwasser	1	0.05	5

	Objekte	utility	Menge	% Gesamt	Fail-Pass
code					
G195	Batterien (Haushalt)	Persönliche Gegenstände	1	0.05	5
G936	Folien für Gewächshäuser	Landwirtschaft	1	0.05	5
G935	Gummi-Puffer für Geh- und Wanderstöcke und Tei...	Persönliche Gegenstände	1	0.05	5
G128	Reifen und Antriebsriemen	nicht klassifiziert	1	0.05	5
G74	Schaumstoffverpackungen/ Isolierung	Infrastruktur	1	0.05	5
G929	Elektronik und Teile; Sensoren, Headsets usw.	Persönliche Gegenstände	1	0.05	5
G65	Eimer	Landwirtschaft	1	0.05	5
G131	Gummibänder	Persönliche Gegenstände	1	0.05	5
G148	Kartonkisten und Stücke	Nicht-Lebensmittelverpackungen	1	0.05	5
G708	Skistöcke	Freizeit und Erholung	1	0.05	5
G159	Kork	Essen und Trinken	1	0.05	5
G21	Getränke-Deckel, Getränkeverschluss	Essen und Trinken	1	0.05	5
G916	Bleistifte und Bruchstücke	Persönliche Gegenstände	1	0.05	5
G199	Andere Metallteile > 50 cm	Infrastruktur	1	0.05	5
G711	Handwärmern	Freizeit und Erholung	1	0.05	5

```

# -*- 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 report
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 variables, 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 location

Median p/100m



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:
isinstance({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, Brienzsee, Neuenburgersee, Thunersee

Fliessgewässer:

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 horizontal 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 Umkreis 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()
```



```

                start_end=start_end,
                agg_dims=agg_dims)

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

# map the quantity 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	
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	28
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_yticks([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()

```

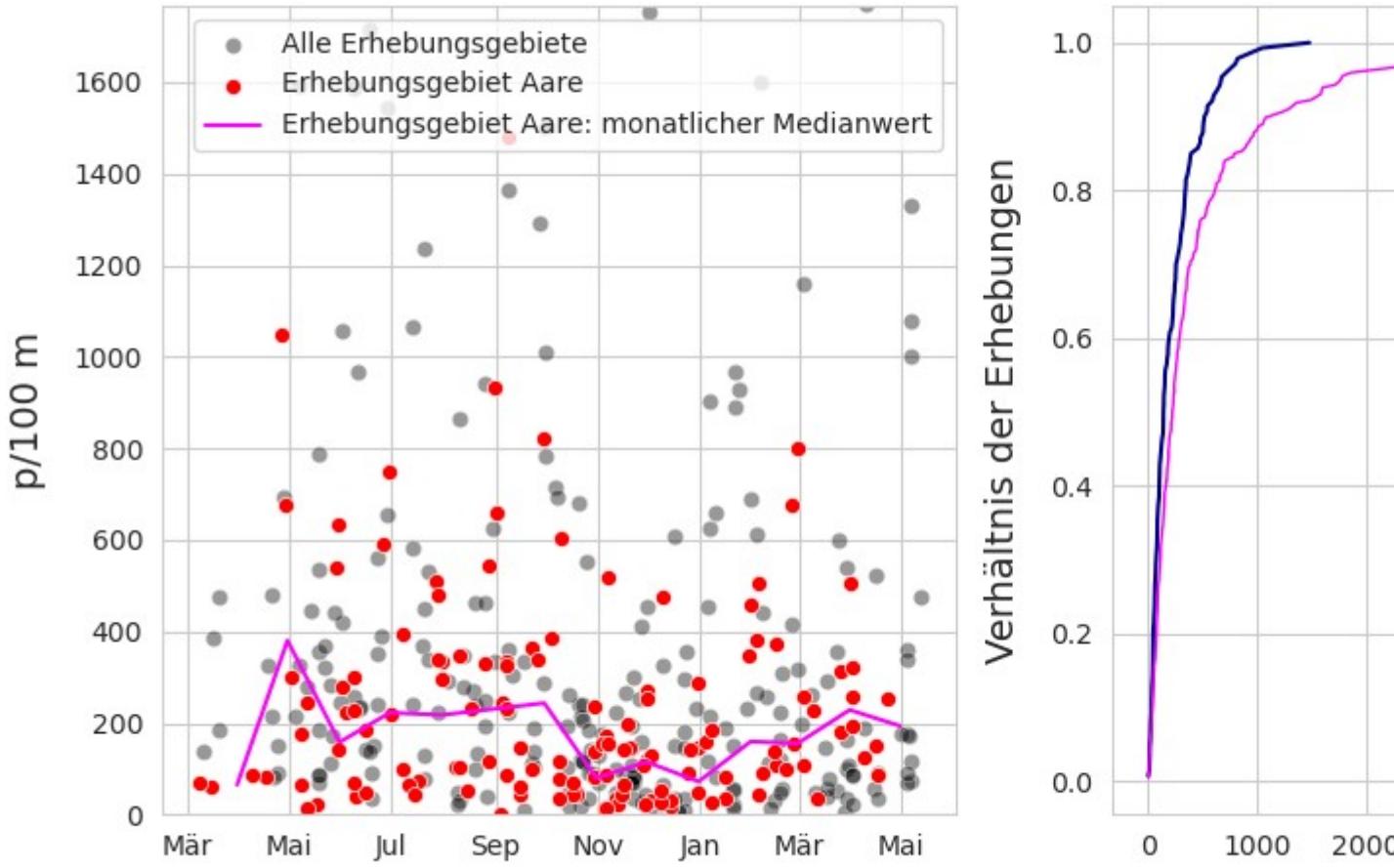


Abb. 3.4 #

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))

# apply 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_cellld()[(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	83.0
Glas	874	6.2
Metall	579	4.1
Papier	391	2.7
Chemikalien	99	0.7
Gummi	96	0.7
Holz	94	0.7
Stoff	92	0.7
Unbekannt	0	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(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 hierachal 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.

	Aare	Aare Nidau-Büren-Kanal	Bielersee	Brienzsee	Emme
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

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 monthly 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 label 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 monthly 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.

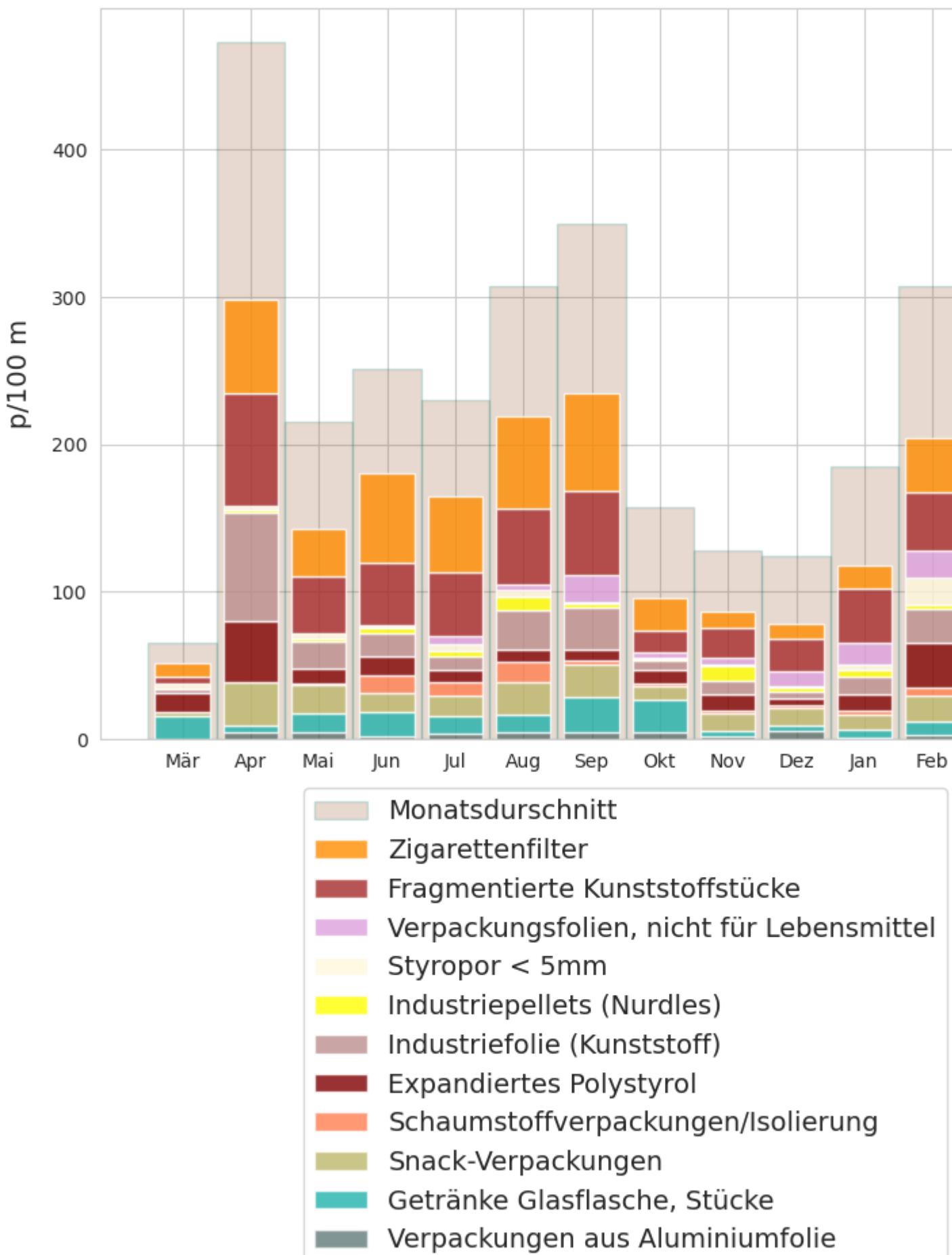


Abb. 3.8 #

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 significant 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()
```

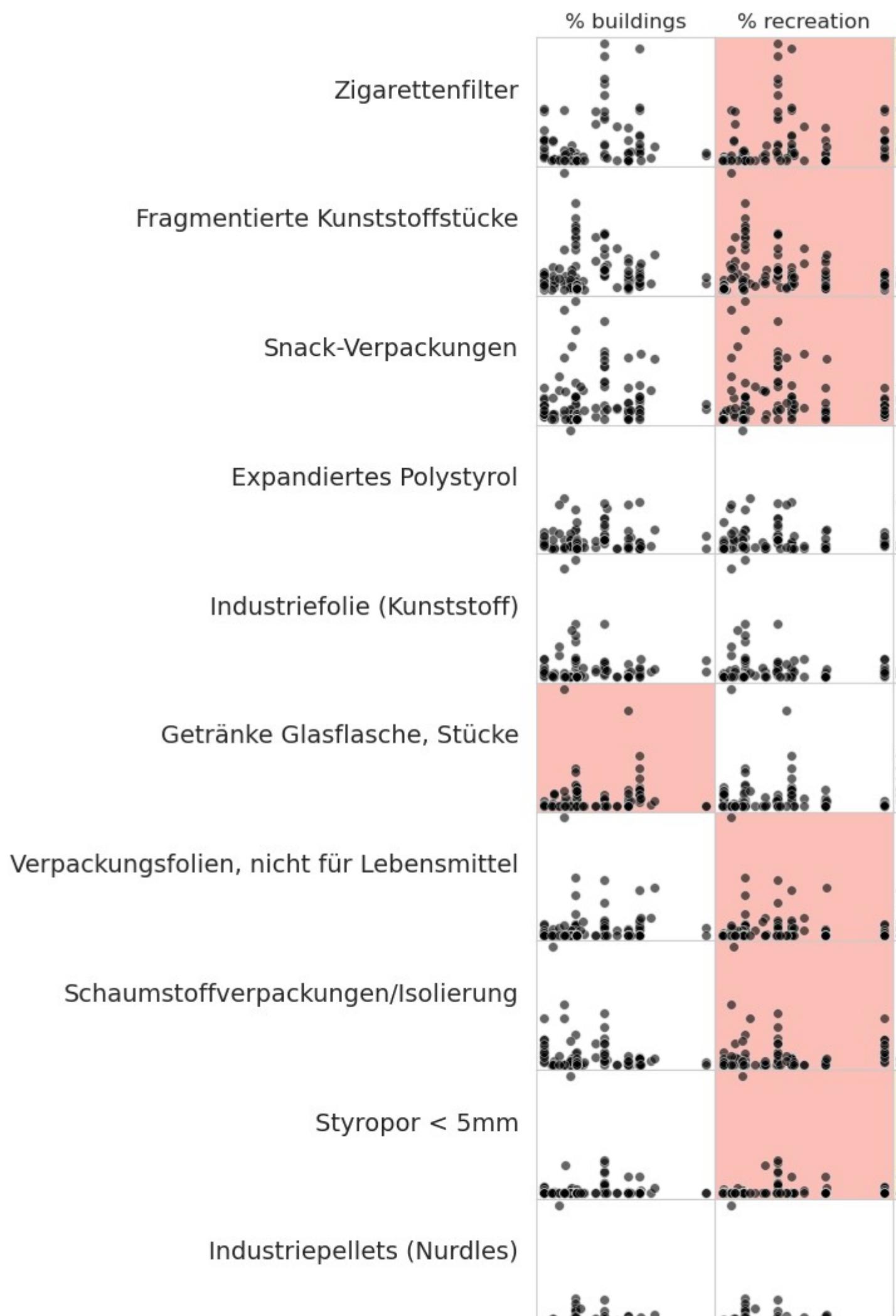


Abb. 3.9 #

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()

```

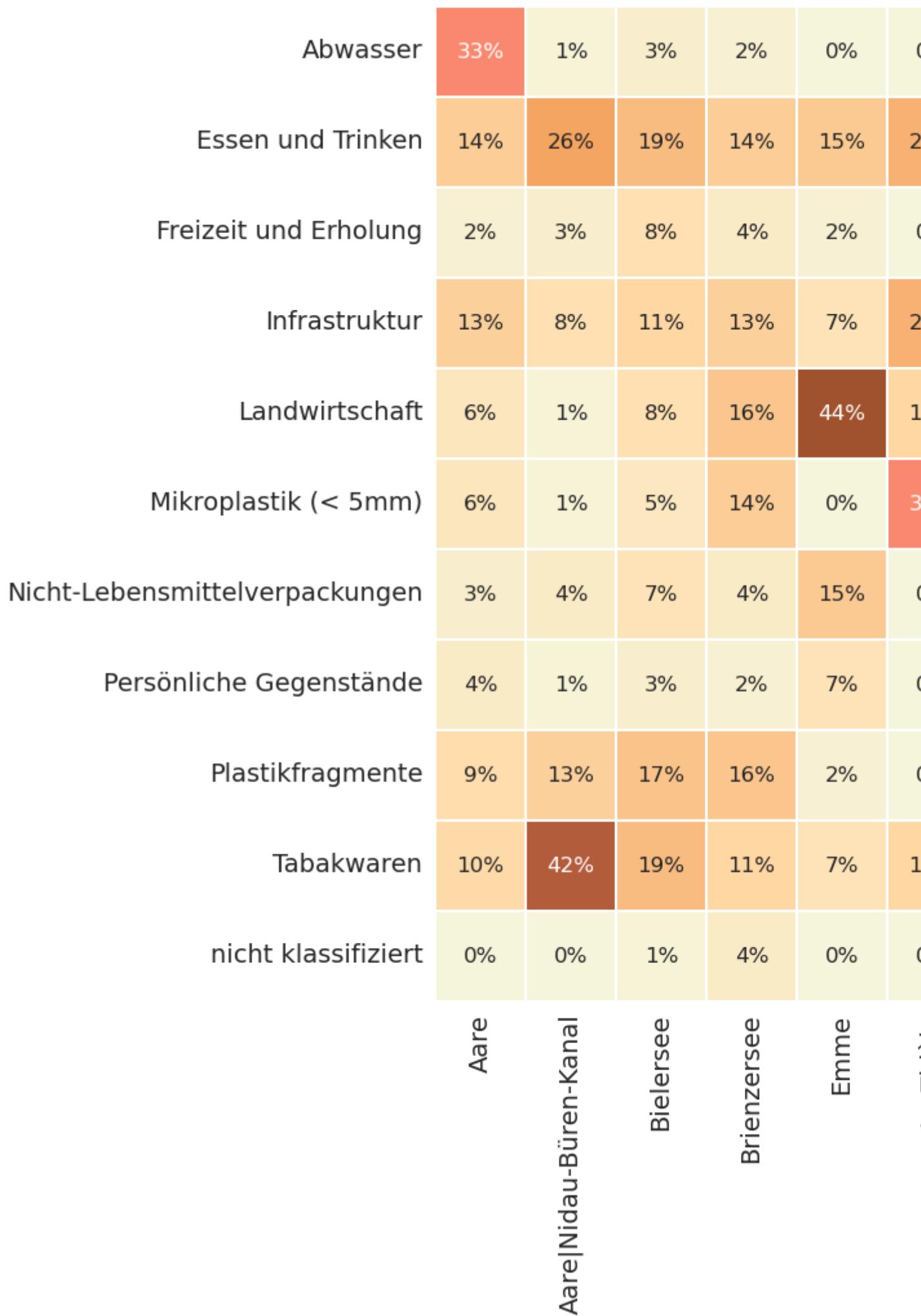


Abb. 3.10 #

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

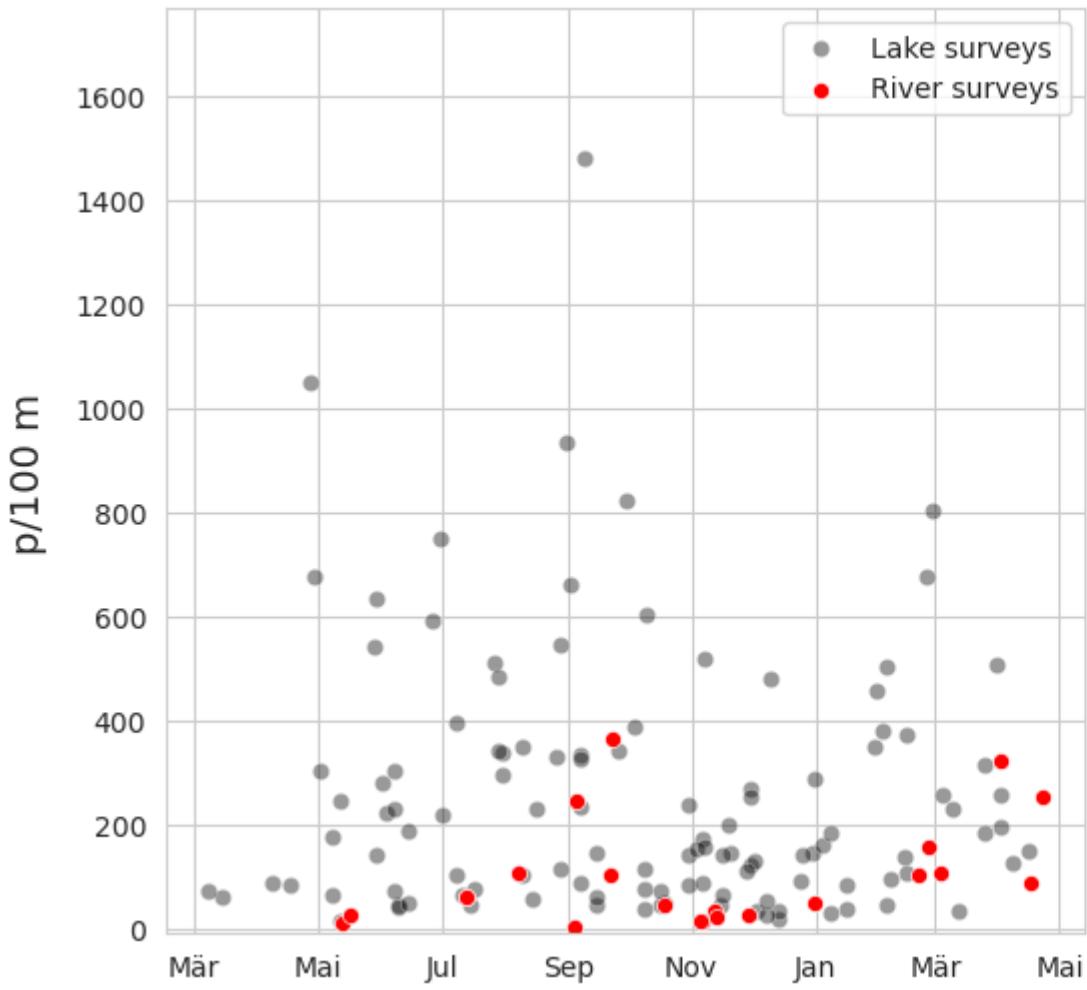


Abb. 3.12 #

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_smpls.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 survey 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.groupby(["loc_date", "code"], as_index=False).agg(agg_pcs_quantity)].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"]))].groupby(["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_frags_foams.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

	latitude	longitude	stat
standort			
schutzenmatte	47.057666	7.634001	Burgdorf
la-petite-plage	46.785054	6.656877	Yverdon-les-Bains
weissenau-neuhaus	46.676583	7.817528	Unterseen
evoile-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
G112	Industriepellets (Nurdles)	Mikroplastik (< 5mm)	262	1.89	32

	Objekte	Gruppenname	Gesamt	% Gesamt	fail rate
code					
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/ Höscheninlagen/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 Accessoires...	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, Zipsäckc...	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 Kugelchen <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, Senklei, usw.	Freizeit und Erholung	6	0.04	3
G38	Abdeckungen; Kunststoffverpackungen, Folien zu...	nicht klassifiziert	6	0.04	3
G938	Zahnstocher, Zahndeide 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

Im Auftrag des Bundesamtes für Umwelt (BAFU), © 2021