## Bijlage 2 - Code chlorofyl A en chemicaliën

Deze bijlage bevat alle code, uitgezonderd die van de tijdreeks, die gerelateerd is aan het deel van het onderzoek over chlorofyl A.

### Databewerking

In [ ]:

```
#Libraries
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import geopandas as gpd
from shapely.geometry import Point
import datetime as dt


%matplotlib inline
```

In [ ]:

```
#Read datafile
df = pd.read_csv('../data/data_clean.csv', error_bad_lines=False)
```

In [ ]:

```
#List of parameters to be removed - kept in seperate file for convenience
remove_pars = pd.read_csv('../data/remove_cols.csv', header=None)
remove_pars_list = remove_pars[0].tolist()
```

In [ ]:

```
#Overwrite datafile with filtered version, then fix PAR column for our pur
poses
df = df[~df.PAR.isin(remove_pars_list)]
df['PAR'] = df['PAR'] + ' ' + df['EHD'] + ' ' + df['HDH']
df.drop(columns=['Unnamed: 0'], inplace=True)
#Repair broken datetime field
df.DATETIME = pd.to_datetime(df.DATETIME)
```

### Ontwikkeling Chlorofyl A in Nederland

In [ ]:

```
#Create the dataframe that only contains chlf-A, and remove one location t
hat has too much influence on the trend
chlf_ts = df[(df.PAR == 'CHLFa ug/l NVT') & (df.BGC != '<') & (df.LOC != 'ST
EILBK')][['DATETIME', 'WAARDE']]
chlf_ts['DATETIME'] = pd.to_datetime(chlf_ts['DATETIME'])
chlf_ts.index = chlf_ts.DATETIME
chlf_ts.drop(columns=['DATETIME'], inplace=True)
```

```
#Resample with monthly means - we need equal spacing for time series, after
all.
chlf_ts = chlf_ts.resample('M').mean()
```

In [ ]:

```
#Quick overview
fig, ax=plt.subplots(figsize=(12,6))
chlf_ts.plot(ax=ax, legend=False)
ax.set_xlabel("Jaartal", fontweight='bold')
ax.set_ylabel("Gem. hoeveelheid chlorofyl A", fontweight='bold')
ax.set_title("Figuur 1 - Gemiddelde hoeveelheid chlorofyl A", fontsize=20)
ax.yaxis.grid(color='lightgray', linestyle='-', linewidth=0.5 )
ax.set_axisbelow(True)
#plt.savefig('../img/mean_chlfa_unfiltered.png')


None
```

In [ ]:

```
#Store the CSV somewhere to load in R for time series
chlf_ts.to_csv('../data/chlf_ts.csv')
```

**Correlaties Chlf-A**

In [ ]:

```
#Create pivot table with multi-index
df_pivot_pars = df.pivot_table(index=['DATETIME', 'LOC'], values='WAARDE',
columns='PAR')
```

In [ ]:

```
#Calculate correlation matrix
correlated_pars = df_pivot_pars.corr()
```

In [ ]:

```
#Filter relevant correlations
chlf_a_corrs = correlated_pars['CHLFa ug/l NVT']
```

In [ ]:

```
#Only keep the strongest correlations
strong_neg_corrs = chlf_a_corrs.sort_values()[0:15]
strong_pos_corrs = chlf_a_corrs.sort_values(ascending=False)[1:15]
```

In [ ]:

```
#How often do the chemicals with strong correlations to chlf-A appear in th
e data?
pos_corr_counts = [df_pivot_pars[x].notnull().sum() for x in
strong_pos_corrs.index]
neg_corr_counts =[df_pivot_pars[x].notnull().sum() for x in
strong_neg_corrs.index]
```

In [ ]:

```
#Create new dataframes with strong/weak corrs and their counts

#Positive
strong_positive_df = pd.DataFrame({"Chemical": strong_pos_corrs.index,
                                   "Correlation": strong_pos_corrs.va
ues,
                                   "No of Observations": pos_corr_co
ts})
strong_positive_df.set_index('Chemical', inplace=True)

#Negative
strong_negative_df = pd.DataFrame({"Chemical": strong_neg_corrs.index,
                                   "Correlation": strong_neg_corrs.va
ues,
                                   "No of Observations": neg_corr_co
ts})
strong_negative_df.set_index('Chemical', inplace=True)
```

In [ ]:

```
#Carbuforan has been found while it really shouldn be - are the concentrati
ons relevant?
df[df.PAR == 'cbfrn ug/l NVT'].BGC.value_counts()
```

In [ ]:

```
#Dataframe manipulation for plotting purposes
strong_positive_df.reset_index(inplace=True)

strong_negative_df.reset_index(inplace=True)

full_corrs_df = pd.concat((strong_positive_df.iloc[:10,:],
strong_negative_df.iloc[3:12,:]))

full_corrs_df.sort_values(by='Correlation', ascending=False, inplace=True)

full_corrs_df['Chemical'] = full_corrs_df['Chemical'].apply(lambda x:
x.split()[0])

full_corrs_df.reset_index(drop=True, inplace=True)
```

In [ ]:

```
#Bars corr
fig, ax = plt.subplots(figsize=(16,12))
g = sns.barplot(x="Correlation", y="Chemical", data=full_corrs_df,
                ax=ax, palette=("GnBu_d"))

# Set background color
sns.set_style("whitegrid")

#Fix lim
ax.set_xlim(-1,1)
# Set title
plt.title('Chlorofyl A: Positieve en Negatieve correlaties', fontsize=20)


# Set x-axis label
plt.xlabel('Correlaties', fontweight='bold')
```

```
# Set y-axis label
plt.ylabel('Stof', fontweight='bold')

# Text on the top of each barplot
for index, row in full_corrs_df[:10].iterrows():
        g.text(0.1,
           index,
           'Corr: ' + str(round(row['Correlation'],3)) + '; Observaties: ' +
str(row['No of Observations']),
           color='white', fontsize=14, fontweight='bold', ha="left", va='cen
ter')

for index, row in full_corrs_df[10:].iterrows():
    g.text(-0.05,
           index,
           'Corr: ' + str(round(row['Correlation'],3)) + '; Observaties: ' +
str(row['No of Observations']),
           color='white', fontsize=14, fontweight='bold', ha="right", va='ce
nter')

plt.savefig('../img/corr_chlfa_bars.png')
```

**PCDD's, PCB's en PCDF's: Dioxines in Nederland**

In [ ]:

```
#Subset of all data that contains dioxins
df_dioxins = df[(df.PAR.str.contains('PCB')) | (df.PAR.str.contains('PCDF')
) | (df.PAR.str.contains('PCDD'))]
```

In [ ]:

```
#Filter out all measurements of dioxins that are below the reporting
treshold
df_dioxins = df_dioxins[df_dioxins.BGC != '<']
```

In [ ]:

```
#Repair datetime field
df_dioxins['DATETIME'] = pd.to_datetime(df_dioxins['DATETIME'])
```

In [ ]:

```
#Categorise values to make EHD irrelevant
quantiles_dioxins = pd.qcut(df_dioxins[df_dioxins.EHD == 'ug/kg'].WAARDE, 5
, labels=False)
quantiles_dioxins = quantiles_dioxins.append(pd.qcut(df_dioxins[df_dioxins.
EHD == 'ug/l'].WAARDE, 5, labels=False))
quantiles_dioxins = quantiles_dioxins.append(pd.qcut(df_dioxins[df_dioxins.
EHD == 'ng/kg'].WAARDE, 5, labels=False))
df_dioxins = pd.concat([df_dioxins, quantiles_dioxins], axis=1)
df_dioxins['QUANTILE'] = df_dioxins.iloc[:,-1]
df_dioxins.drop(df_dioxins.columns[-2], axis=1, inplace=True)
```

In [ ]:

```
#Convert df_dioxins to a GeoDataFrame for visualization
```

```
dioxins_geometry = [Point(xy) for xy in zip(df_dioxins.X_RD, df_dioxins.Y_RD)]
dioxins_geo_df = gpd.GeoDataFrame(df_dioxins, geometry=dioxins_geometry)
```

In [ ]:

```
#Read NL map
nl_map = gpd.read_file('../data/shapefiles/2018-Imergis_provinciegrenzen_kustlijn.shp')
nl_rivers = gpd.read_file('../data/shapefiles/NL-water-simpel.shp')
```

In [ ]:

```
def plot_this_year(year):
    '''Takes a year (that is within the Geodataframe, of course) and uses it to visualize locations
    where the highest concentrations of dioxins were measured. Annotations will only be placed on locations
    with the highest quantile of concentrations.'''
    #Annotation box props
    bbox_properties = dict(boxstyle="round,pad=0.2", fc="white", ec="k", lw=1.5)

    #DF manipulation
    geo_df_slice = dioxins_geo_df[dioxins_geo_df.DATETIME.dt.year == year]
    quantile_values = pd.DataFrame(geo_df_slice.groupby('LOC').QUANTILE.agg('mean'))
    geo_df_slice = geo_df_slice.merge(quantile_values, left_on='LOC', right_index=True)
    geo_df_slice['new_quantiles'] = pd.qcut(geo_df_slice.QUANTILE_y, 5, labels=False)

    #Plotting
    fig, ax = plt.subplots(figsize=(14,14), subplot_kw={'aspect':'equal'})
    ax.set_xlim(0,300000)

    #Annoying fig-num condition
    if year == 2016:
        ax.set_title("Figuur 4 - Dioxine-vervuiling in kwantielen, " + str(year), size=20)
    else:
        ax.set_title("Dioxine-vervuiling in kwantielen, " + str(year), size=20)

    #Plotting continued
    ax.set_ylim(300000,650000)
    ax.set_axis_off()
    nl_map.plot(ax=ax, color='#629fca', edgecolor='darkgrey', linewidth=0.3)
    nl_rivers.plot(ax=ax, alpha=0.9, color='white')
    geo_df_slice.plot(ax=ax, cmap='RdYlGn_r',alpha=0.8, column='new_quantiles', scheme='quantiles', markersize=100)

    #Annotation
    annotated_locs = []
    for i, txt in enumerate(geo_df_slice.LOCOMS.tolist()):
        if geo_df_slice.iloc[i, -1] > 3:
            if geo_df_slice.iloc[i, 1] not in annotated_locs:
                if (txt == "Beerkanaal midden"):
                    ax.text(s=txt,
```

```
                            x=geo_df_slice.iloc[i,-8] - 15000,
                            y=geo_df_slice.iloc[i,-7] - 5500,
                           size=12,
                           bbox=bbox_properties)
                    annotated_locs.append(geo_df_slice.iloc[i, 1])
                elif (txt != 'Nieuwegein') & (txt != "Westzaan (kilometer 13
)"):
                    ax.text(s=txt,
                            x=geo_df_slice.iloc[i,-8],
                            y=geo_df_slice.iloc[i,-7] - 4500,
                           size=12,
                           bbox=bbox_properties)
                    annotated_locs.append(geo_df_slice.iloc[i, 1])
                else:
                    ax.text(s=txt,
                            x=geo_df_slice.iloc[i,-8],
                            y=geo_df_slice.iloc[i,-7] + 3500,
                           size=12,
                           bbox=bbox_properties)
                    annotated_locs.append(geo_df_slice.iloc[i, 1])


    return geo_df_slice
```

In [ ]:

```
#Manual list of interesting locations for later use
locs_of_interest = ['Nieuwegein', 'Gouda voorhaven', 'Nederweert', 'Schaar
van Ouden Doel', 'Sas van Gent',
                    'Haringvlietsluis','Brienenoord (kilometer 996.5)', 'Nie
wersluis', 'Puttershoek', 'Keizersveer',
                    'Belfeld boven']
```

In [ ]:

```
#Creates a dict with the top "polluted" locations per year.
#Also creates a new dataframe with locations for visualization for later.
annotated_locations_yearly = {}
quantiled_dioxins_df = pd.DataFrame()

#Plot each year, save the figure, and build a special quantiled dataframe w
hile we're at it.
for year in dioxins_geo_df.DATETIME.dt.year.unique():
    x = plot_this_year(year)
    #Commented out savefig to avoid unnecessary repetition
    #plt.savefig('../img/dioxins_concentration_' + str(year) + '.png')
    annotated_locations_yearly.update({year: x[x.new_quantiles > 3].LOCOMS.u
nique()})
    quantiled_dioxins_df = pd.concat([quantiled_dioxins_df, x])
    #Suppress output for now
    plt.close()
```

In [ ]:

```
#Populate a list, where each entry is the quantile for a location in a
year
quantile_change_locs = []
```

```
for year in dioxins_geo_df.DATETIME.dt.year.unique():
    for loc in locs_of_interest:

quantile_change_locs.append([loc,quantiled_dioxins_df[(quantiled_dioxins_df
.DATETIME.dt.year == year) &
                            (quantiled_dioxins_df.LOCOMS ==
loc)].new_quantiles.mean(), year])
```

In [ ]:

```
#Convert said list to dataframe and pivot for plotting purposes.
quantile_change_locs = pd.DataFrame(quantile_change_locs).fillna(0)
quantile_change_plottable = quantile_change_locs.pivot_table(index=0, value
s=1, columns=[2])
quantile_change_plottable.index.name=None
quantile_change_plottable.columns.name=None
```

In [ ]:

```
#Sort and visualize
quantile_change_plottable.sort_values(by=[2016, 2015, 2014], ascending=Fals
e, inplace=True)
quantile_change_plottable.style.background_gradient(cmap='RdYlGn_r')
```

**Lanthanides**

In [ ]:

```
#Manual list of lanthanides found earlier
lanthanide_list = ['Gd mg/kg dg', 'Yb mg/kg dg', 'Er mg/kg dg', 'Tm mg/kg d
g', 'Ho mg/kg dg', 'Dy mg/kg dg']
```

In [ ]:

```
#Create DF with just lanthanides
df_lanthanides = df[df.PAR.isin(lanthanide_list)]
df_lanthanides['DATETIME'] = pd.to_datetime(df_lanthanides['DATETIME'])

#Instantiate geo-df
lanthanides_geometry = [Point(xy) for xy in zip(df_lanthanides.X_RD,
df_lanthanides.Y_RD)]
lanthanides_geo_df = gpd.GeoDataFrame(df_lanthanides,
geometry=lanthanides_geometry)
```

In [ ]:

```
#Filter DF and set textbox props
lanthanides_2016 = lanthanides_geo_df[lanthanides_geo_df.DATETIME.dt.year =
= 2016]
bbox_properties = dict(boxstyle="round,pad=0.3", fc="white", ec="k", lw=1.5
)

#Plot
fig, ax = plt.subplots(figsize=(14,14), subplot_kw={'aspect':'equal'})
ax.set_xlim(0,300000)
ax.set_ylim(300000,650000)
ax.set_axis_off()
ax.set_title("Figuur 6 - Hoogste aantal metingen lanthanides in Nederland,
2016", size=20)
nl_map.plot(ax=ax, color='#629fca', edgecolor='darkgrey', linewidth=0.3)
```

```
nl_map.plot(ax=ax, color='#6291ca', edgecolor='darkgrey', linewidth=0.3)
nl_rivers.plot(ax=ax, alpha=0.9, color='white')
lanthanides_2016.plot(ax=ax, cmap='RdYlGn_r',alpha=0.8, column='WAARDE', ma
rkersize=100)

#Annotate
annotated_locs = []
for i, txt in enumerate(lanthanides_2016.LOCOMS.tolist()):
    if lanthanides_2016.iloc[i, 1] not in annotated_locs:
        if lanthanides_2016.LOCOMS.value_counts()[txt] > 100:
            ax.text(s=txt + ', ' + str(lanthanides_2016.LOCOMS.value_counts(
)[txt]) + 'x',
                    x=lanthanides_2016.iloc[i,-5],
                    y=lanthanides_2016.iloc[i,-4] + 4500,
                   size=12,
                   bbox=bbox_properties)
            annotated_locs.append(lanthanides_2016.iloc[i, 1])

plt.savefig('../img/lanthanides.png')
```