Data_sorting.py

```
import pandas as pd
import numpy as np
import time
pd.set_option("display.max_rows", None, "display.max_columns", None)
desired_width = 320
pd.set_option('display.width', desired_width)

startTime = time.time()
date = "2023-02-07"     # Here the desired date is entered as "YYYY-MM-DD"
 file_type = 'f0000'     # The 9th of november has file 0001, all others have 0000
 print("1/5: Date which has been loaded: " + date)

 def df_maker(filename, n=None):
    """This function creates a dataframe from a .csv-file or .txt-file, with tabs as delimiter"""
    with open(filename) as file:
        df = pd.DataFrame(pd.read_csv(file, delimiter='\t', skiprows=n))
    return(df)

 def df_maker2(filename, n=None):
    """This function creates a dataframe from a .csv-file or .txt-file, with comma as delimiter"""
    with open(filename) as file:
        df = pd.DataFrame(pd.read_csv(file, delimiter=',', skiprows=n))
    return(df)

 def df_maker3(filename, n=None):
    """This function creates a dataframe from a .csv-file or .txt-file, with semi-colon as delimiter"""
    with open(filename) as file:
        df = pd.DataFrame(pd.read_csv(file, delimiter=';', skiprows=n))
    return(df)

 '''Following is an example on how to read Excel files'''
 path_to_excel_folder = "../Speciale/Metadata_files/"
 excel_name = "Time_for_python.xlsx"
 excel_file = path_to_excel_folder+excel_name
 time_df = pd.read_excel(excel_file)
 time_df = time_df.dropna()
 list_of_columns = list(time_df.columns)
 list_of_columns.remove('Date')

 for c in list_of_columns:   # This for-loop makes the time-dataframe nice to read and makes computing easier
    time_df[c] = pd.to_datetime(time_df['Date'].astype(str) + " " + time_df[c].astype(str), format='%Y-%m-%d %H:%M:%S')
 time_df['Date'] = time_df['Date'].dt.date

 print("2/5: Time frames have been loaded")

 '''Predefined values'''
 volume_of_glasses_l = 2
 area_of_soil_cylinder_cm2 = (7/2)**2 * np.pi
 height_of_cylinder_cm = 12
```

```
area_of_total_cylinder_cm2 = (7.5/2)**2 * np.pi
volume_of_total_cylinder = area_of_total_cylinder_cm2 * height_of_cylinder_cm
volume_of_tubes_to_machine_l = ((np.pi * 180 * 0.635 * 0.635)/1000)*2
volume_of_glass_minus_cylinder_l = ((2*1000)-volume_of_total_cylinder)/1000 +
volume_of_tubes_to_machine_l
temp_of_glasses_K = 7+273.15
pressure_atm = 1
molar_weight_of_N2O_g_mol = 44.01   # Molar weight of N2O
ideal_gas_constant = 0.0821  # L*atm*K-1*mol-1
n_mol =
(pressure_atm*volume_of_glass_minus_cylinder_l)/(ideal_gas_constant*temp_of_glasses_K)  # Number
of moles within the glass bottle
standard_molar_volume_L_mol = volume_of_glass_minus_cylinder_l/n_mol    # The standard molar
volume
print("3/5: Fixed values have been defined")

def filter_dataframe(file, name, lines_to_ignore, keep_columns, renames):
    """This function takes in the raw files from the machine and
    returns a new file only with important columns and saves this in the working data folder"""
    df = df_maker2(file, n=lines_to_ignore)     # This loads the file to a dataframe and the n is the
number of lines we wish to ignore
    df = df[keep_columns]   # Here we filter out the columns we do not wish to keep to reduce
computing time
    df = df.rename(columns = renames)   # We rename the columns to give nicer names
    df['Time'] = pd.to_datetime(df['Time'], errors= 'coerce')   # We remove lines where there is no date,
sometimes there are strange lines at the bottom of the file
    df = df.dropna(subset='Time', axis = 0)
    df.to_csv('../Speciale/First_WD_columns_removed/' + name, index=False)
    return (df)

name_of_file = date
columns_to_keep = ['                Time', '    [NN15O]_ppm', '    [N15NO]_ppm',
                   '    [NNO18]_ppm', '         d15NA', '         d15NB',
                   '         d15N', '     [N2O]_ppm', '           SP']
new_column_names = {'                Time': 'Time', '    [NN15O]_ppm': 'NN15O_ppm',
                    '    [N15NO]_ppm': 'N15NO_ppm', '    [NNO18]_ppm': 'NNO18_ppm', '         d15NA':
'd15NA',
                    '         d15NB': 'd15NB', '         d15N': 'd15N', '     [N2O]_ppm': 'N2O_ppm', '           SP':
'SP'}
sorting = filter_dataframe('../Speciale/Raw_data_files/'+date+'/N2Oiso_'+date+'_' + file_type + '.txt',
name_of_file, 1, columns_to_keep, new_column_names)

print("4/5: Unecessary columns have been removed")

def apply_treatment(file, seconds):
    """This function sorts the total data file into which bottle was measured at which time"""
    df = df_maker2('../Speciale/First_WD_columns_removed/'+file)    # We load
    df['Time'] = pd.to_datetime(df['Time'])     # We convert the time column to datetime, making sorting
easier
    time_column = df[['Time']].copy()   # We copy the time column
    time_column['Time'] = pd.to_datetime(time_column['Time'])
    time_df['Date'] = time_df['Date'].astype(str)
    time_frames = time_df[time_df['Date'] == file]  # We load the time frame file and select the date we
are interested in
    time_frames = time_frames.drop(columns='Date')  # We drop the date column due to it making
computing easier
```

```
    list_to_use = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54]  # This was necessary to make the labelling easier
    new_df = pd.DataFrame()    # This is the dateframe we will fill
    for c in list_to_use:
        bottle = c/2 + 1
        start = time_frames.iloc[0][list_of_columns[c]]    # The start time is loaded
        end = time_frames.iloc[0][list_of_columns[c+1]]    # The end time is loaded
        working_df = df[(df['Time'] > start) & (df['Time'] <= end)].copy()  # This selects all the lines between the start and the end point for each measurement
         working_df = working_df.iloc[seconds:]  # Here we remove the first 200 seconds of the file
         working_df = working_df.reset_index(drop=True)  # Here we reset the index
         working_df['Bottle'] = int(bottle)  # We label the time series with the bottle measured
         new_df = pd.concat([new_df, working_df])    # We add it to the new dataframe
    new_df = new_df.sort_values('Time')
    new_df['N2O_umol_L-1'] = (new_df['N2O_ppm'] * 10 ** (-6))/standard_molar_volume_L_mol * 1000000  # Here we estimate the amount of N2O in the air in terms of micromol, pr liter
    new_df['N2O_umol'] = new_df['N2O_umol_L-1'] * volume_of_glass_minus_cylinder_l  # Here we estimate the amount of N2O in the air in terms of micromol by multiplying with the volume
    new_df.to_csv('../Speciale/Second_WD_sorted/' + file, index=False)
    return(new_df)

sorting2 = apply_treatment(date, 200)

print("5/5: File has been sorted and given propper bottle label")
executionTime = (time.time() - startTime)
print('Execution time in seconds: ' + str(round(executionTime, 2)))

```

Regression.py

```
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import time
import warnings
warnings.filterwarnings(action='ignore', category=RuntimeWarning)
pd.set_option("display.max_rows", None, "display.max_columns", None)
desired_width = 320
 pd.set_option('display.width', desired_width)
 startTime = time.time()

 def df_maker(filename, n=None):
    """This function creates a dataframe from a .csv-file or .txt-file, with tabs as delimiter"""
    with open(filename) as file:
        df = pd.DataFrame(pd.read_csv(file, delimiter='\t', skiprows=n))
    return(df)

 def df_maker2(filename, n=None):
    """This function creates a dataframe from a .csv-file or .txt-file, with comma as delimiter"""
    with open(filename) as file:
        df = pd.DataFrame(pd.read_csv(file, delimiter=',', skiprows=n))
    return(df)

 def df_maker3(filename, n=None):
    """This function creates a dataframe from a .csv-file or .txt-file, with semi-colon as delimiter"""
    with open(filename) as file:
        df = pd.DataFrame(pd.read_csv(file, delimiter=';', skiprows=n))
    return(df)

 def polynomial_func(x, *params):
    return np.polyval(params, x)

 def regression(x, y, degree):
    params = np.polyfit(x, y, degree)
    p_value = stats.linregress(x, y).pvalue
    return params, p_value

 date = '2023-02-07'   # Put in the date to be used in YYYY-MM-DD format
 print("1/6: Date which has been loaded: " + date)

 "'Predefined values"'
 volume_of_glasses_l = 2
 area_of_soil_cylinder_cm2 = (7/2)**2 * np.pi
 height_of_cylinder_cm = 12
 area_of_total_cylinder_cm2 = (7.5/2)**2 * np.pi
 volume_of_total_cylinder = area_of_total_cylinder_cm2 * height_of_cylinder_cm
 volume_of_tubes_to_machine_l = ((np.pi * 180 * 0.635 * 0.635)/1000)*2
 volume_of_glass_minus_cylinder_l = ((2*1000)-volume_of_total_cylinder)/1000 +
volume_of_tubes_to_machine_l
 temp_of_glasses_K = 7+273.15
 pressure_atm = 1
 molar_weight_of_N2O_g_mol = 44.01
```

```
ideal_gas_constant = 0.0821  # L*atm*K-1*mol-1
n_mol = (pressure_atm*volume_of_glass_minus_cylinder_l)/(ideal_gas_constant*temp_of_glasses_K)
standard_molar_volume_L_mol = volume_of_glass_minus_cylinder_l/n_mol

print("2/6: Fixed values have been defined")

def rate_calculator(file):
    """This function calculates the rate for each measurement and returns one line
    with measurement number, rates of change and p-value. The function returns one dataframe
    with one rate pr replicate"""
    df = df_maker2(file)    # We open the file as a dataframe
    df['Time'] = pd.to_datetime(df['Time'])     # We convert the time column to datetime format
    list_of_dicts = []  # We create a list where we will fill in dictionaries, these are quick to compute and requires less computer power
    list_of_bottles = sorted(list(df.Bottle.unique()))  # We create a list of all the bottles used
    for i in list_of_bottles:
        working_df = df[df['Bottle'] == i].copy()   # We create a data frame only with the time for each bottle
        working_df['Time_step_s'] = (working_df['Time'] - working_df.iloc[0]['Time']).astype('timedelta64[s]')  # We create a column with the amount of time since the first line. This line has unit seconds
        x = working_df['Time_step_s'].to_numpy()    # We decide upon an x-column to perform linear regression on, and convert this to a numpy array
        ppm = working_df['N2O_ppm'].to_numpy()      # We convert the N2O ppm column to a numpy array, for easier regression
        umol = working_df['N2O_umol'].to_numpy()    # We convert the N2O umol column to a numpy array, for easier regression
        ppm_slope, ppm_intercept, ppm_r_value, ppm_p_value, ppm_std_err = stats.linregress(x, ppm) # This in the linear regression
        umol_slope, umol_intercept, umol_r_value, umol_p_value, umol_std_err = stats.linregress(x, umol)  # This in the linear regression
        params, p_value = regression(x, umol, 2)
        coefficients = np.polyder(params)
        poly_slope = np.polyval(coefficients, 2)
        x_fit = np.linspace(x.min(), x.max(), 100)
        y_fit = polynomial_func(x_fit, *params)
        # r_value, p_value, _, _, _ = stats.linregress(x, y_fit)

        # y_fit_linear = np.polyval([umol_slope, umol_intercept], x_fit)  #Uncomment this section to see fits
        # plt.scatter(x, umol, marker="x", color="black", label="Data points")
        # plt.plot(x_fit, y_fit, label="Polynomial fit")
        # plt.plot(x_fit, y_fit_linear, label="Linear fit")
        # plt.legend()
        # plt.show()

        if ppm_p_value < 0.05:  # This loop tests significance. Will be used to remove rates if they are not significant
            p = 'significant'
        else:
            p = 'not significant'
        if poly_slope < 0:
            flux = poly_slope/area_of_soil_cylinder_cm2
        else:
            flux = poly_slope/area_of_soil_cylinder_cm2
```

```
        dictionary = {'Bottle': i, 'Date': date, 'PPM/s': ppm_slope, 'umol/s': umol_slope, 'umol_cm2_s': flux, 'p': ppm_p_value, 'significance': p}     # We create a dictionary only of important numbers
        list_of_dicts.append(dictionary)
    flux_df = pd.DataFrame(list_of_dicts)
    flux_df.to_csv('../Speciale/Third_WD_rate_pr_bottle/'+date, index=False)
    return(flux_df)

calculate_flux = rate_calculator('../Speciale/Second_WD_sorted/'+date)
print("3/6: Rates of change have been calculated")

def rate_combiner(file, date):
    df = df_maker2(file)    # We load the file into a dataframe
    list_of_dicts = []  # We create a list where we will fill in dictionaries, these are quick to compute and requires less computer power
    list_of_treatments = [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18], [19, 20, 21], [22, 23, 24], [25, 26, 27], [28]]  # This is a list, where each element is a list of bottles with the same treatment
    for i in list_of_treatments:
        working_df = df[df['Bottle'].isin(i)]   # We select all rows which have bottle-number in the same treatment
        avg_ppm = working_df['PPM/s'].mean()    # We average the ppm/s
        std_ppm = working_df['PPM/s'].std()     # We estimate std of ppm/s
        avg_umol = working_df['umol/s'].mean()
        std_umol = working_df['umol/s'].std()
        avg_flux = working_df['umol_cm2_s'].mean()  # We estimate the average flux for each treatment
        std_flux = working_df['umol_cm2_s'].std()   # We estimate the standard deviation for the average flux for each treatment
        if 1 in i:              # All below is only labelling the treatments
            treatment = "HW"
        elif 4 in i:
            treatment = "HD"
        elif 7 in i:
            treatment = "HA"
        elif 10 in i:
            treatment = "LW"
        elif 13 in i:
            treatment = "LD"
        elif 16 in i:
            treatment = "AA"
        elif 19 in i:
            treatment = "AW"
        elif 22 in i:
            treatment = "AD"
        elif 25 in i:
            treatment = "LA"
        else:
            treatment = "W"
        dictionary = {"Date": date, 'Treatment': treatment, "avg_PPM/s": avg_ppm, "std_PPM/s": std_ppm, "avg_umol/s": avg_umol, "std_umol/s": std_umol,
                      "avg_flux_umol_cm2_s": avg_flux, "std_flux": std_flux}
        list_of_dicts.append(dictionary)
    combined_flux = pd.DataFrame(list_of_dicts)
    combined_flux.to_csv("../Speciale/Fourth_WD_flux_pr_day/" + date, index=False)
    return(combined_flux)

```

```
combine = rate_combiner('../Speciale/Third_WD_rate_pr_bottle/'+date, date)

print("4/6: Mean flux for each treatment have been calculated")

def combine_all_rates(list_of_files):
    final_df = pd.DataFrame()
    for f in list_of_files:
        df = df_maker2("../Speciale/Fourth_WD_flux_pr_day/" + f[0])
        df['24h'] = f[1]
        final_df = pd.concat([final_df, df])
    final_df.to_csv("../Speciale/Output_files/Combined", index=False)
    return(final_df)

list_of_dates = [['2022-11-09', 1], ['2022-11-10', 1], ['2022-11-11', 1], ['2022-11-12', 1], ['2022-11-13',
1], ['2022-11-14', 1],
                 ['2022-11-15', 1], ['2022-11-16', 1], ['2022-11-17', 1], ['2022-11-18', 1], ['2022-11-19', 1],
['2022-11-20', 1],
                 ['2022-11-21', 1], ['2022-11-22', 1], ['2022-11-23', 1], ['2022-11-24', 1], ['2022-11-25', 1.5],
['2022-11-27', 2],
                 ['2022-11-29', 1.5], ['2022-11-30', 2], ['2022-12-03', 3], ['2022-12-07', 3], ['2022-12-10', 3],
['2022-12-13', 3],
                 ['2022-12-16', 3], ['2022-12-19', 3], ['2022-12-22', 3.5], ['2022-12-26', 5.5], ['2023-01-02',
7], ['2023-01-09', 7],
                 ['2023-01-16', 6.5], ['2023-01-22', 6.5], ['2023-01-22', 11], ['2023-02-07', 9.5]]
combine_all = combine_all_rates(list_of_dates)

print("5/6: All average fluxes have been combined to one file")

def combine_bottles(list_of_files):
    final_df = pd.DataFrame()
    for l in list_of_files:
        df = df_maker2("../Speciale/Third_WD_rate_pr_bottle/" + l[0])
        df['24h'] = l[1]
        final_df = pd.concat([final_df, df])
    final_df = final_df[['Date', 'Bottle', 'PPM/s', 'umol/s', 'umol_cm2_s', 'p', 'significance', '24h']]
    final_df.to_csv("../Speciale/Output_files/Combined_bottles", index=False)
    return(final_df)

bottles = combine_bottles(list_of_dates)

print("6/6: All fluxes for each bottle have been combined to one file")
executionTime = (time.time() - startTime)
print('Execution time in seconds: ' + str(round(executionTime, 2)))

```

Calculation_bottles.py

```
import pandas as pd
import warnings
import time
warnings.filterwarnings(action='ignore', category=RuntimeWarning)
pd.set_option("display.max_rows", None, "display.max_columns", None)
desired_width = 320
pd.set_option('display.width', desired_width)
startTime = time.time()
def df_maker(filename, n=None):
    """This function creates a dataframe from a .csv-file or .txt-file, with tabs as delimiter"""
    with open(filename) as file:
        df = pd.DataFrame(pd.read_csv(file, delimiter='\t', skiprows=n))
    return(df)

def df_maker2(filename, n=None):
    """This function creates a dataframe from a .csv-file or .txt-file, with comma as delimiter"""
    with open(filename) as file:
        df = pd.DataFrame(pd.read_csv(file, delimiter=',', skiprows=n))
    return(df)

def df_maker3(filename, n=None):
    """This function creates a dataframe from a .csv-file or .txt-file, with semi-colon as delimiter"""
    with open(filename) as file:
        df = pd.DataFrame(pd.read_csv(file, delimiter=';', skiprows=n))
    return(df)

bottle_df = df_maker2('../Speciale/Output_files/Computed_bottles.csv')
print(bottle_df.columns)
list_of_dates = list(bottle_df.Date.unique())
list_of_bottle_pairs = [[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15],
                        [16, 17, 18], [19, 20, 21], [22, 23, 24], [25, 26, 27]]
list_of_dicts = []

for d in list_of_dates:
    working_df = bottle_df[bottle_df['Date'] == d]
    for i in list_of_bottle_pairs:
        working_df_2 = working_df[working_df['Bottle'].isin(i)]
        X24h = working_df_2.iloc[0]['X24h']
        avg_umol_N2O_cm2_s = working_df_2['umol_cm2_s'].mean()
        std_umol_N2O_cm2_s = working_df_2['umol_cm2_s'].std()
        avg_mg_N2O_cm2_h = working_df_2['mg_N2O_cm2_h'].mean()
        std_mg_N2O_cm2_h = working_df_2['mg_N2O_cm2_h'].std()
        avg_g_N2O_m2_d = working_df_2['g_N2O_m2_d'].mean()
        std_g_N2O_m2_d = working_df_2['g_N2O_m2_d'].std()
        avg_mg_N2O_m2_d = working_df_2['mg_N2O_m2_d'].mean()
        std_mg_N2O_m2_d = working_df_2['mg_N2O_m2_d'].std()
        avg_kg_N2O_ha_d = working_df_2['kg_N2O_ha_d'].mean()
        std_kg_N2O_ha_d = working_df_2['kg_N2O_ha_d'].std()
        avg_mg_N2O_N_cm2_h = working_df_2['mg_N2O_N_cm2_h'].mean()
        std_mg_N2O_N_cm2_h = working_df_2['mg_N2O_N_cm2_h'].std()
        avg_mg_N2O_N_m2_d = working_df_2['mg_N2O_N_m2_d'].mean()
        std_mg_N2O_N_m2_d = working_df_2['mg_N2O_N_m2_d'].std()
        avg_g_N2O_N_m2_d = working_df_2['g_N2O_N_m2_d'].mean()
```

```
        std_g_N2O_N_m2_d = working_df_2['g_N2O_N_m2_d'].std()
        avg_kg_N2O_N_ha_d = working_df_2['kg_N2O_N_ha_d'].mean()
        std_kg_N2O_N_ha_d = working_df_2['kg_N2O_N_ha_d'].std()
        avg_mg_N2O_N_d = working_df_2['mg_N2O_N_d'].mean()
        std_mg_N2O_N_d = working_df_2['mg_N2O_N_d'].std()
        avg_cumulative_mg_N2O_N = working_df_2['cumulative'].mean()
        std_cumulative_mg_N2O_N = working_df_2['cumulative'].std()
        avg_cumulative_mg_N2O_N_m2 = working_df_2['cumulative_m2'].mean()
        std_cumulative_mg_N2O_N_m2 = working_df_2['cumulative_m2'].std()
        if 1 in i:              # All below is only labelling the treatments
            treatment = "HW"
        elif 4 in i:
            treatment = "HD"
        elif 7 in i:
            treatment = "HA"
        elif 10 in i:
            treatment = "LW"
        elif 13 in i:
            treatment = "LD"
        elif 16 in i:
            treatment = "AA"
        elif 19 in i:
            treatment = "AW"
        elif 22 in i:
            treatment = "AD"
        elif 25 in i:
            treatment = "LA"
        else:
            treatment = "W"
        dictionary = {'Date': d, 'Treatment': treatment, 'X24h': X24h,
                      'avg_umol_N2O_cm2_s': avg_umol_N2O_cm2_s, 'std_umol_N2O_cm2_s':
std_umol_N2O_cm2_s,
                      'avg_mg_N2O_cm2_s': avg_mg_N2O_cm2_h, 'std_mg_N2O_cm2_s':
std_mg_N2O_cm2_h,
                      'avg_g_N2O_m2_d': avg_g_N2O_m2_d, 'std_g_N2O_m2_d': std_g_N2O_m2_d,
                      'avg_mg_N2O_m2_d': avg_mg_N2O_m2_d, 'std_mg_N2O_m2_d':
std_mg_N2O_m2_d,
                      'avg_kg_N2O_ha_d': avg_kg_N2O_ha_d, 'std_kg_N2O_ha_d': std_kg_N2O_ha_d,
                      'avg_mg_N2O_N_cm2_h': avg_mg_N2O_N_cm2_h, 'std_mg_N2O_N_cm2_h':
std_mg_N2O_N_cm2_h,
                      'avg_mg_N2O_N_m2_d': avg_mg_N2O_N_m2_d, 'std_mg_N2O_N_m2_d':
std_mg_N2O_N_m2_d,
                      'avg_g_N2O_N_m2_d': avg_g_N2O_N_m2_d, 'std_g_N2O_N_m2_d':
std_g_N2O_N_m2_d,
                      'avg_kg_N2O_N_ha_d': avg_kg_N2O_N_ha_d, 'std_kg_N2O_N_ha_d':
std_kg_N2O_N_ha_d,
                      'avg_mg_N2O_N_d': avg_mg_N2O_N_d, 'std_mg_N2O_N_d': std_mg_N2O_N_d,
                      'avg_cumulative_mg_N2O_N': avg_cumulative_mg_N2O_N,
'std_cumulative_mg_N2O_N': std_cumulative_mg_N2O_N,
                      'avg_cumulative_mg_N2O_N_m2': avg_cumulative_mg_N2O_N_m2,
'std_cumulative_mg_N2O_N_m2': std_cumulative_mg_N2O_N_m2}
        list_of_dicts.append(dictionary)

    combined_final = pd.DataFrame(list_of_dicts)
    combined_final.to_csv("../Speciale/Output_files/Combined_final", index=False)

```

```
print("1/1: Fluxes for each bottle have been estimated and statistics are now possible")
executionTime = (time.time() - startTime)
print('Execution time in seconds: ' + str(round(executionTime, 2)))

```

Plotting_lab.py

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
pd.set_option("display.max_rows", None, "display.max_columns", None)
desired_width = 320
pd.set_option('display.width', desired_width)

def df_maker(filename, n=None):
    """This function creates a dataframe from a .csv-file or .txt-file, with tabs as delimiter"""
    with open(filename) as file:
        df = pd.DataFrame(pd.read_csv(file, delimiter='\t', skiprows=n))
    return(df)

def df_maker2(filename, n=None):
    """This function creates a dataframe from a .csv-file or .txt-file, with comma as delimiter"""
    with open(filename) as file:
        df = pd.DataFrame(pd.read_csv(file, delimiter=',', skiprows=n))
    return(df)

def df_maker3(filename, n=None):
    """This function creates a dataframe from a .csv-file or .txt-file, with semi-colon as delimiter"""
    with open(filename) as file:
        df = pd.DataFrame(pd.read_csv(file, delimiter=';', skiprows=n))
    return(df)

list_of_dates = sorted(os.listdir('../Speciale/Second_WD_sorted/'))
for d in list_of_dates:
    date = d
    df = df_maker2('../Speciale/Second_WD_sorted/' + date, 0)
    list_of_bottles = sorted(list(df['Bottle'].unique()))
    df_1 = df[df['Bottle'] == 1]
    df_2 = df[df['Bottle'] == 2]
    df_3 = df[df['Bottle'] == 3]
    df_4 = df[df['Bottle'] == 4]
    df_5 = df[df['Bottle'] == 5]
    df_6 = df[df['Bottle'] == 6]
    df_7 = df[df['Bottle'] == 7]
    df_8 = df[df['Bottle'] == 8]
    df_9 = df[df['Bottle'] == 9]
    df_10 = df[df['Bottle'] == 10]
    df_11 = df[df['Bottle'] == 11]
    df_12 = df[df['Bottle'] == 12]
    df_13 = df[df['Bottle'] == 13]
    df_14 = df[df['Bottle'] == 14]
    df_15 = df[df['Bottle'] == 15]
    df_16 = df[df['Bottle'] == 16]
    df_17 = df[df['Bottle'] == 17]
    df_18 = df[df['Bottle'] == 18]
    df_19 = df[df['Bottle'] == 19]
    df_20 = df[df['Bottle'] == 20]
    df_21 = df[df['Bottle'] == 21]
    df_22 = df[df['Bottle'] == 22]
```

```
    df_23 = df[df['Bottle'] == 23]
    df_24 = df[df['Bottle'] == 24]
    df_25 = df[df['Bottle'] == 25]
    df_26 = df[df['Bottle'] == 26]
    df_27 = df[df['Bottle'] == 27]

    x1 = np.array(df_1.index.values.tolist())
    y1 = np.array(df_1['N2O_ppm'])
    x2 = np.array(df_2.index.values.tolist())
    y2 = np.array(df_2['N2O_ppm'])
    x3 = np.array(df_3.index.values.tolist())
    y3 = np.array(df_3['N2O_ppm'])
    x4 = np.array(df_4.index.values.tolist())
    y4 = np.array(df_4['N2O_ppm'])
    x5 = np.array(df_5.index.values.tolist())
    y5 = np.array(df_5['N2O_ppm'])
    x6 = np.array(df_6.index.values.tolist())
    y6 = np.array(df_6['N2O_ppm'])
    x7 = np.array(df_7.index.values.tolist())
    y7 = np.array(df_7['N2O_ppm'])
    x8 = np.array(df_8.index.values.tolist())
    y8 = np.array(df_8['N2O_ppm'])
    x9 = np.array(df_9.index.values.tolist())
    y9 = np.array(df_9['N2O_ppm'])
    x10 = np.array(df_10.index.values.tolist())
    y10 = np.array(df_10['N2O_ppm'])
    x11 = np.array(df_11.index.values.tolist())
    y11 = np.array(df_11['N2O_ppm'])
    x12 = np.array(df_12.index.values.tolist())
    y12 = np.array(df_12['N2O_ppm'])
    x13 = np.array(df_13.index.values.tolist())
    y13 = np.array(df_13['N2O_ppm'])
    x14 = np.array(df_14.index.values.tolist())
    y14 = np.array(df_14['N2O_ppm'])
    x15 = np.array(df_15.index.values.tolist())
    y15 = np.array(df_15['N2O_ppm'])
    x16 = np.array(df_16.index.values.tolist())
    y16 = np.array(df_16['N2O_ppm'])
    x17 = np.array(df_17.index.values.tolist())
    y17 = np.array(df_17['N2O_ppm'])
    x18 = np.array(df_18.index.values.tolist())
    y18 = np.array(df_18['N2O_ppm'])
    x19 = np.array(df_19.index.values.tolist())
    y19 = np.array(df_19['N2O_ppm'])
    x20 = np.array(df_20.index.values.tolist())
    y20 = np.array(df_20['N2O_ppm'])
     x21 = np.array(df_21.index.values.tolist())
     y21 = np.array(df_21['N2O_ppm'])
     x22 = np.array(df_22.index.values.tolist())
     y22 = np.array(df_22['N2O_ppm'])
     x23 = np.array(df_23.index.values.tolist())
     y23 = np.array(df_23['N2O_ppm'])
     x24 = np.array(df_24.index.values.tolist())
     y24 = np.array(df_24['N2O_ppm'])
     x25 = np.array(df_25.index.values.tolist())
     y25 = np.array(df_25['N2O_ppm'])
```

```
    x26 = np.array(df_26.index.values.tolist())
    y26 = np.array(df_26['N2O_ppm'])
    x27 = np.array(df_27.index.values.tolist())
    y27 = np.array(df_27['N2O_ppm'])

    fig, axs = plt.subplots(3, 9, figsize=(30, 7))
    fig.tight_layout(pad=5.0)
    axs[0, 0].plot(x1, y1)
    axs[0, 0].title.set_text('df_1')
    axs[1, 0].plot(x2, y2)
    axs[1, 0].title.set_text('df_2')
    axs[2, 0].plot(x3, y3)
    axs[2, 0].title.set_text('df_3')
    axs[0, 1].plot(x4, y4)
    axs[0, 1].title.set_text('df_4')
    axs[1, 1].plot(x5, y5)
    axs[1, 1].title.set_text('df_5')
    axs[2, 1].plot(x6, y6)
    axs[2, 1].title.set_text('df_6')
    axs[0, 2].plot(x7, y7)
    axs[0, 2].title.set_text('df_7')
    axs[1, 2].plot(x8, y8)
    axs[1, 2].title.set_text('df_8')
    axs[2, 2].plot(x9, y9)
    axs[2, 2].title.set_text('df_9')
    axs[0, 3].plot(x10, y10)
    axs[0, 3].title.set_text('df_10')
    axs[1, 3].plot(x11, y11)
    axs[1, 3].title.set_text('df_11')
    axs[2, 3].plot(x12, y12)
    axs[2, 3].title.set_text('df_12')
    axs[0, 4].plot(x13, y13)
    axs[0, 4].title.set_text('df_13')
    axs[1, 4].plot(x14, y14)
    axs[1, 4].title.set_text('df_14')
    axs[2, 4].plot(x15, y15)
    axs[2, 4].title.set_text('df_15')
    axs[0, 5].plot(x16, y16)
    axs[0, 5].title.set_text('df_16')
    axs[1, 5].plot(x17, y17)
    axs[1, 5].title.set_text('df_17')
    axs[2, 5].plot(x18, y18)
    axs[2, 5].title.set_text('df_18')
    axs[0, 6].plot(x19, y19)
    axs[0, 6].title.set_text('df_19')
    axs[1, 6].plot(x20, y20)
    axs[1, 6].title.set_text('df_20')
    axs[2, 6].plot(x21, y21)
    axs[2, 6].title.set_text('df_21')
    axs[0, 7].plot(x22, y22)
    axs[0, 7].title.set_text('df_22')
    axs[1, 7].plot(x23, y23)
    axs[1, 7].title.set_text('df_23')
    axs[2, 7].plot(x24, y24)
    axs[2, 7].title.set_text('df_24')
    axs[0, 8].plot(x25, y25)
```

```
    axs[0, 8].title.set_text('df_25')
    axs[1, 8].plot(x26, y26)
    axs[1, 8].title.set_text('df_26')
    axs[2, 8].plot(x27, y27)
    axs[2, 8].title.set_text('df_27')
    fig.suptitle(str(date))
    plt.show()

```

Bottles.R

```
install.packages("dplyr")
install.packages("tidyr")
install.packages("ggplot2")
install.packages("readxl")
install.packages("tidypaleo")
install.packages("tidyverse")
install.packages("gasfluxes")
install.packages("VGAM")
install.packages("pracma")
 install.packages("egg")
 install.packages("ggpubr")
 install.packages('reshape2')
 install.packages("patchwork")
 install.packages("cowplot")
 install.packages('Hmisc')
 install.packages("mmtable2")
 install.packages("knitr")

 library(gridExtra)
 library(dplyr)
 library(tidyr)
 library(ggplot2)
 library(readxl)
 library(gasfluxes)
 library(tidyverse)
 library(multcompView)
 library(multcomp)
 library(multcompLetters)
 library(tidypaleo)
 library(pracma)
 library(egg)
 library(ggpubr)
 library(reshape2)
 library(data.table)
 library(patchwork)
 library(cowplot)
 library(emmeans)
 library(extrafont)
 my_theme <- theme_article() +
   theme(text = element_text(family = "EB Garamond"))

 # This is for loading the data and converting columns to the right format
 calculation_string <-"~/Documents/MacBook 2020/Geografi
KU/Speciale/Data/Python/Speciale/Output_files/"
 calculation_name <- "Combined_bottles"
 calculation_file_location <- paste(calculation_string, calculation_name, sep="")
 calculation_data_from_python <- read.csv(calculation_file_location, header=TRUE)
 calculation_data_from_python <- na.omit(calculation_data_from_python)
 calculation_data_from_python$Date <- as.Date(calculation_data_from_python$Date, format = "%Y-
%m-%d")

 # Below is the conversion to the right units
```

```r
calculation_data_from_python$umol_cm2_s  <- ifelse(calculation_data_from_python$umol_cm2_s == 0, 3.391683e-11, calculation_data_from_python$umol_cm2_s)
calculation_data_from_python['X24h_2'] <- calculation_data_from_python['X24h']
calculation_data_from_python$X24h_2[calculation_data_from_python$Date == '2023-01-09'] <- 3.5
calculation_data_from_python['mg_N2O_cm2_h'] <- (((calculation_data_from_python[['umol_cm2_s']] * 3600)/1000000) * 44.013) * 1000
calculation_data_from_python['g_N2O_m2_d'] <- (calculation_data_from_python['mg_N2O_cm2_h']/1000) * 10000 * 24
calculation_data_from_python['mg_N2O_m2_d'] <- calculation_data_from_python['g_N2O_m2_d'] * 1000
calculation_data_from_python['kg_N2O_ha_d'] <- (calculation_data_from_python['g_N2O_m2_d']/1000) * 10000
calculation_data_from_python['mg_N2O_N_cm2_h'] <- calculation_data_from_python['mg_N2O_cm2_h'] * (28.014/44.013)
calculation_data_from_python['g_N2O_N_m2_d'] <- calculation_data_from_python['g_N2O_m2_d'] * (28.014/44.013)
calculation_data_from_python['mg_N2O_N_m2_d'] <- calculation_data_from_python['g_N2O_N_m2_d'] * 1000
calculation_data_from_python['kg_N2O_N_ha_d'] <- calculation_data_from_python['kg_N2O_ha_d'] * (28.014/44.013)
calculation_data_from_python['mg_N2O_N_d'] <- calculation_data_from_python['mg_N2O_N_cm2_h'] * 24 * calculation_data_from_python['X24h'] * (7/2)**2 * pi
calculation_data_from_python['mg_N2O_N_m2_d_total'] <- calculation_data_from_python['mg_N2O_N_cm2_h'] * 24 * calculation_data_from_python['X24h_2'] * 10000

# Below is the calculation of the cumulated data
calculation_data_from_python <- calculation_data_from_python %>%
  group_by(Bottle) %>%
  mutate(cumulative = cumsum(mg_N2O_N_d))
calculation_data_from_python <- calculation_data_from_python %>%
  group_by(Bottle) %>%
  mutate(cumulative_m2 = cumsum(mg_N2O_N_m2_d_total))

# This saves the entire data file to a nice .csv file which can be used for many things
write.csv(calculation_data_from_python, "~/Documents/MacBook 2020/GeografiKU/Speciale/Data/Python/Speciale/Output_files/Computed_bottles.csv", row.names=FALSE)

# The proper treatment names are added
calculation_data_from_python['Treatment'] = ""
calculation_data_from_python <- within(calculation_data_from_python, Treatment[Bottle == 1 | Bottle == 2 | Bottle == 3] <- 'HW')
calculation_data_from_python <- within(calculation_data_from_python, Treatment[Bottle == 4 | Bottle == 5 | Bottle == 6] <- 'HD')
calculation_data_from_python <- within(calculation_data_from_python, Treatment[Bottle == 7 | Bottle == 8 | Bottle == 9] <- 'HA')
calculation_data_from_python <- within(calculation_data_from_python, Treatment[Bottle == 10 | Bottle == 11 | Bottle == 12] <- 'LW')
calculation_data_from_python <- within(calculation_data_from_python, Treatment[Bottle == 13 | Bottle == 14 | Bottle == 15] <- 'LD')
calculation_data_from_python <- within(calculation_data_from_python, Treatment[Bottle == 16 | Bottle == 17 | Bottle == 18] <- 'AA')
calculation_data_from_python <- within(calculation_data_from_python, Treatment[Bottle == 19 | Bottle == 20 | Bottle == 21] <- 'AW')
```

```r
calculation_data_from_python <- within(calculation_data_from_python, Treatment[Bottle == 22 | Bottle == 23 | Bottle == 24] <- 'AD')
calculation_data_from_python <- within(calculation_data_from_python, Treatment[Bottle == 25 | Bottle == 26 | Bottle == 27] <- 'LA')
calculation_data_from_python <- calculation_data_from_python[!(calculation_data_from_python$Bottle == 28), ]
date_to_examine <- '2022-11-16'
stat_df <- calculation_data_from_python[calculation_data_from_python$Date == date_to_examine, ]
stat_df <- select(stat_df, Treatment, Bottle, Date, cumulative)

# We summarise the cumulative emissions
group_by(stat_df, Treatment) %>%
  summarise(
    count = n(),
    mean = mean(cumulative, na.rm = TRUE),
    sd = sd(cumulative, na.rm = TRUE)
  )

# Here I pairwise use t-test to evaluate the difference between the emission factors of each treatment
list_of_treatments <- list('AD', 'AA', 'AW', 'HD', 'HA', 'HW', 'LD', 'LA', 'LW')
p_list <- data.frame(Treatment1 = c(), Treatment2 = c(), p = c())
for (t in list_of_treatments){
  working_df0 <- calculation_data_from_python[calculation_data_from_python$Date == date_to_examine,]
  working_df1 <- working_df0[working_df0$Treatment == t,]
  for (j in list_of_treatments){
    working_df2 <- working_df0[working_df0$Treatment == j,]
    if (t == j){
    } else {
      t1_cumulative <- working_df1[, "cumulative_m2"]
      t2_cumulative <- working_df2[, "cumulative_m2"]
      results <- t.test(t1_cumulative, t2_cumulative, var.equal = FALSE)
      p_value <- results$p.value
      tuple <- c(t, j, p_value)
      p_list <- rbind(p_list, tuple)
    }
  }
}

t1 <- c(132, 438, 126)
t2 <- c(0.32, 0.97, 3.79)

t.test(t1, t2, var.equal = FALSE)

#Examine specific period, not used
folder_name <- "~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Week_results/"
period_start <- "2022-11-09"
period_end <- "2023-01-09"  # Enter last date to get entire period
period_df <- subset(calculation_data_from_python, Date >= period_start & Date <= period_end)
length_period <- round(sum(period_df[period_df$Bottle == 1,]$X24h_2))
period_df['mg_N2O_N_lost_m2'] <- period_df['cumulative_m2']
keeps <- c("Date", "Bottle", "X24h", "mg_N2O_N_m2_d", "mg_N2O_N_lost_m2", "Treatment")
period_df <- period_df[keeps]
max_df <- subset(period_df, Date == period_end)

```

```
mean_loss <- aggregate(max_df['mg_N2O_N_lost_m2'], list(max_df$Treatment), mean)
mean_loss <- mean_loss %>%
  rename("mg_N2O_N_lost_mean_m2" = "mg_N2O_N_lost_m2")
std_loss <- aggregate(max_df['mg_N2O_N_lost_m2'], list(max_df$Treatment), std)
std_loss <- std_loss %>%
  rename("mg_N2O_N_lost_std_m2" = "mg_N2O_N_lost_m2")
obs_loss <- aggregate(max_df['mg_N2O_N_lost_m2'], list(max_df$Treatment), length)
obs_loss <- obs_loss %>%
  rename("mg_N2O_N_lost_obs_m2" = "mg_N2O_N_lost_m2")

period_summary <- merge(mean_loss, std_loss, by = "Group.1")
period_summary <- merge(period_summary, obs_loss, by = "Group.1")
period_summary <- period_summary %>%
  rename("Treatment" = "Group.1")

period_summary2 <- period_df %>% group_by(Treatment) %>%
  summarise(mg_N2O_N_m2_d_mean = weighted.mean(mg_N2O_N_m2_d, X24h),
            mg_N2O_N_m2_d_std = sqrt(Hmisc::wtd.var(mg_N2O_N_m2_d, X24h)),
            mg_N2O_N_m2_d_obs = length(mg_N2O_N_m2_d))

period_summary <- merge(period_summary, period_summary2, by = "Treatment")
period_summary$Treatment <- factor(period_summary$Treatment, levels = c("LD", "LA", "LW",
"AD", "AA", "AW", "HD", "HA", "HW"))
period_summary$input[period_summary$Treatment=='AA'] <- 19.2
period_summary$input[period_summary$Treatment=='AD'] <- 19.2
period_summary$input[period_summary$Treatment=='AW'] <- 19.2
period_summary$input[period_summary$Treatment=='HA'] <- 57.7
period_summary$input[period_summary$Treatment=='HD'] <- 57.7
period_summary$input[period_summary$Treatment=='HW'] <- 57.7
period_summary$input[period_summary$Treatment=='LA'] <- 3.8
period_summary$input[period_summary$Treatment=='LD'] <- 3.8
period_summary$input[period_summary$Treatment=='LW'] <- 3.8
period_summary$percent_loss <-
round(period_summary$mg_N2O_N_lost_mean/period_summary$input * 100, 2)

period_summary = subset(period_summary, select = -c(mg_N2O_N_lost_obs,
mg_N2O_N_m2_d_obs))

max_loss <- summarise(period_summary, max(mg_N2O_N_lost_mean))
max_std <- summarise(period_summary, max(mg_N2O_N_lost_std))
position <- as.numeric(max_loss+max_std)
upper_limit <- ceiling(position)

#plots of total loss and mean flux over the examined period
period_loss <- ggplot(aes(x = Treatment, y = mg_N2O_N_lost_mean), data = period_summary) +
  geom_bar(stat = 'identity') + theme_article() +
  geom_errorbar(aes(ymin = ifelse(mg_N2O_N_lost_mean-
mg_N2O_N_lost_std<0,0,mg_N2O_N_lost_mean-mg_N2O_N_lost_std), ymax =
mg_N2O_N_lost_mean+mg_N2O_N_lost_std), width = 0.2)+
  geom_text(aes(label = paste(percent_loss, "%"), y = ifelse(mg_N2O_N_lost_mean +
(mg_N2O_N_lost_std/1) > 5,  mg_N2O_N_lost_mean - (mg_N2O_N_lost_std/0.75),
mg_N2O_N_lost_mean + (mg_N2O_N_lost_std/1)), vjust=-1))+
  ylab(~Total ~N ~released ~as ~N[2]*O ~(mg))+
  annotate("text", x = 2, y = upper_limit, label = paste("From",period_start, "to", period_end,": ",
period_length, "days"))+
  ylim(0, upper_limit)
```

```
184
185  max_flux <- ceiling(as.numeric(summarise(period_summary,
max(mg_N2O_N_m2_d_mean+mg_N2O_N_m2_d_std))))
186  period_flux <- ggplot(data = period_summary, aes(x = Treatment, y = mg_N2O_N_m2_d_mean)) +
187    geom_point(shape = 15, size = 2) + theme_article() +
188    geom_errorbar(aes(ymin = ifelse(mg_N2O_N_m2_d_mean-
mg_N2O_N_m2_d_std<0,0,mg_N2O_N_m2_d_mean-mg_N2O_N_m2_d_std), ymax =
mg_N2O_N_m2_d_mean+mg_N2O_N_m2_d_std), width = 0.2)+
189    ylab(~Mean ~flux ~over ~the ~duration ~(mg ~N[2]*O ~-N ~m^{-2} ~d^{-1}))+
190    annotate("text", x = 2, y = max_flux, label = paste("From",period_start, "to", period_end,": ",
period_length, "days"))
191
192  #Creating tables for the examined period
193  period_summary_table <- period_summary
194  period_summary_table <- period_summary_table %>%
195    rename("input (mg N)" = "input", "loss (%)" = "percent_loss", "mg N2O-N lost
(avg)"="mg_N2O_N_lost_mean",
196           "mg N2O-N lost (std)"="mg_N2O_N_lost_std", "mg N2O-N lost (obs)"="mg_N2O_N_lost_obs",
197           "mg N2O-N m2 d1 (avg)" = "mg_N2O_N_m2_d_mean", "mg N2O-N m2 d1 (std)" =
"mg_N2O_N_m2_d_std", "mg N2O-N m2 d1 (obs)" = "mg_N2O_N_m2_d_obs")
198  period_summary_table <- as.data.frame(t(period_summary_table))
199  colnames(period_summary_table) <- period_summary$Treatment
200  period_summary_table <- subset(period_summary_table,!(AA=="AA"))
201  period_summary_table <- data.frame(period_summary_table)
202  period_summary_table <- mutate_all(period_summary_table, function(x)
as.numeric(as.character(x)))
203  period_summary_table <- period_summary_table %>%
204    mutate_if(is.numeric, round, digits = 4)
205  period_summary_table
206
207  period_table <- tableGrob(period_summary_table)
208
209  # Cumulative
210  df <- calculation_data_from_python[calculation_data_from_python$Date == '2023-01-02',]
211  flux_copy <- calculation_data_from_python
212  flux_copy['max_cumulative'] <- ''
213  flux_copy <- within(flux_copy, max_cumulative[Bottle == 1] <- 1749.104)
214  flux_copy <- within(flux_copy, max_cumulative[Bottle == 2] <- 2139.130)
215  flux_copy <- within(flux_copy, max_cumulative[Bottle == 3] <- 808.6125)
216  flux_copy <- within(flux_copy, max_cumulative[Bottle == 4] <- 2.952519)
217  flux_copy <- within(flux_copy, max_cumulative[Bottle == 5] <- 6.466581)
218  flux_copy <- within(flux_copy, max_cumulative[Bottle == 6] <- 5.427279)
219  flux_copy <- within(flux_copy, max_cumulative[Bottle == 7] <- 52.48218)
220  flux_copy <- within(flux_copy, max_cumulative[Bottle == 8] <- 7.363181)
221  flux_copy <- within(flux_copy, max_cumulative[Bottle == 9] <- 100.0195)
222  flux_copy <- within(flux_copy, max_cumulative[Bottle == 10] <- 45.40896)
223  flux_copy <- within(flux_copy, max_cumulative[Bottle == 11] <- 171.9291)
224  flux_copy <- within(flux_copy, max_cumulative[Bottle == 12] <- 66.04378)
225  flux_copy <- within(flux_copy, max_cumulative[Bottle == 13] <- 4.387771)
226  flux_copy <- within(flux_copy, max_cumulative[Bottle == 14] <- 3.566830)
227  flux_copy <- within(flux_copy, max_cumulative[Bottle == 15] <- 1.973524)
228  flux_copy <- within(flux_copy, max_cumulative[Bottle == 16] <- 35.34258)
229  flux_copy <- within(flux_copy, max_cumulative[Bottle == 17] <- 3.934895)
230  flux_copy <- within(flux_copy, max_cumulative[Bottle == 18] <- 68.49925)
231  flux_copy <- within(flux_copy, max_cumulative[Bottle == 19] <- 654.5374)
232  flux_copy <- within(flux_copy, max_cumulative[Bottle == 20] <- 730.0350)
```

```r
flux_copy <- within(flux_copy, max_cumulative[Bottle == 21] <- 399.0687)
flux_copy <- within(flux_copy, max_cumulative[Bottle == 22] <- 2.187777)
flux_copy <- within(flux_copy, max_cumulative[Bottle == 23] <- 1.952786)
flux_copy <- within(flux_copy, max_cumulative[Bottle == 24] <- 2.215709)
flux_copy <- within(flux_copy, max_cumulative[Bottle == 25] <- 2.318660)
flux_copy <- within(flux_copy, max_cumulative[Bottle == 26] <- 0.3907410)
flux_copy <- within(flux_copy, max_cumulative[Bottle == 27] <- 3.187841)

flux_copy$max_cumulative <- as.numeric(flux_copy$max_cumulative)
flux_copy['percent'] <- flux_copy['cumulative_m2']/flux_copy['max_cumulative'] * 100
flux_copy <- flux_copy[flux_copy$Date <= as.Date('2023-01-04'),]

flux_copy_date <- flux_copy[flux_copy$Date == as.Date('2022-11-22'),]

comb_flux <<- flux_copy %>%
  group_by(Date, Treatment) %>%
  summarise(avg_percent = mean(percent),
            std_percent = sd(percent))


comb_flux['N_amount'] <- ""
comb_flux <- within(comb_flux, N_amount[Treatment == 'LD'|Treatment == 'LA'|Treatment == 'LW'] <
- 'low N')
comb_flux <- within(comb_flux, N_amount[Treatment == 'AD'|Treatment == 'AA'|Treatment == 'AW']
<- 'common N')
comb_flux <- within(comb_flux, N_amount[Treatment == 'HD'|Treatment == 'HA'|Treatment == 'HW']
<- 'high N')
comb_flux['Wetness'] <- ""
comb_flux <- within(comb_flux, Wetness[Treatment == 'LD'|Treatment == 'AD'|Treatment == 'HD'] <-
'Drained,')
comb_flux <- within(comb_flux, Wetness[Treatment == 'LA'|Treatment == 'AA'|Treatment == 'HA'] <-
'Field conditions,')
comb_flux <- within(comb_flux, Wetness[Treatment == 'LW'|Treatment == 'AW'|Treatment == 'HW']
<- 'Saturated,')

ggplot(aes(x = Date, y = avg_percent), data = comb_flux) +
  geom_line() + geom_point(shape = 1) + my_theme + facet_grid(N_amount ~ Wetness)

unique(comb_flux$Date)

examining_date <- comb_flux[comb_flux$Date == as.Date("2022-11-22"),]

list_of_treatments <- c("HW", "HD", "HA", "LW", "LD", "LA", "AW", "AD", "AA")
p_df <- data.frame(t1 = character(), t2 = character, p_val = numeric(), stringsAsFactors = FALSE)

for (t in list_of_treatments) {
  working_df <- flux_copy_date[flux_copy_date$Treatment == t,]
  for (j in list_of_treatments) {
    if (t == j) {
      next  # Skip to the next iteration if t is equal to j
    } else {
      working_df2 <- flux_copy_date[flux_copy_date$Treatment == j,]
      p <- t.test(working_df$percent, working_df2$percent, var.equal = FALSE)$p.value
      p_df <- rbind(p_df, data.frame(t1 = t, t2 = j, p_val = p))
    }
  }
```

```
}

p_df <- p_df[!duplicated(p_df$p_val), ]
letters <- multcomp::cld(p_df$p_val, lower = TRUE, upper = TRUE, Letters = letters)


flux_copy_date <- subset(flux_copy_date, select = c(Treatment, percent))

```

R_plotting.R

```r
install.packages("dplyr")
install.packages("tidyr")
install.packages("ggplot2")
install.packages("readxl")
install.packages("tidypaleo")
install.packages("tidyverse")
install.packages("gasfluxes")
install.packages("VGAM")
install.packages("pracma")
 install.packages("egg")
 install.packages("ggpubr")
 install.packages('reshape2')
 install.packages("patchwork")
 install.packages("cowplot")
 install.packages('extrafont')
 install.packages('RColorBrewer')

 library(dplyr)
 library(tidyr)
 library(ggplot2)
 library(readxl)
 library(gasfluxes)
 library(tidyverse)
 library(tidypaleo)
 library(pracma)
 library(egg)
 library(ggpubr)
 library(reshape2)
 library(data.table)
 library(patchwork)
 library(cowplot)
 library(extrafont)
 library(RColorBrewer)

 my_theme <- theme_article() +
   theme(text = element_text(family = "EB Garamond"))

 # This is for loading the data and converting columns to the right format
 python_string <-"~/Documents/MacBook 2020/Geografi
KU/Speciale/Data/Python/Speciale/Output_files/"
 name <- "Combined_final"
 file_location <- paste(python_string, name, sep="")
 data_from_python <- read.csv(file_location, header=TRUE)
 data_from_python <- na.omit(data_from_python)
 data_from_python$Date <- as.Date(data_from_python$Date, format = "%Y-%m-%d")
 ref_date <- as.Date('2022-11-09')
 data_from_python$Day_number <- difftime(data_from_python$Date, ref_date, units = 'days')
 data_from_python$Day_number <- as.integer(data_from_python$Day_number)
 pd <- position_dodge(width = 0.4)

 # Here I add the proper labels, such that legends will have proper layout
 data_from_python['N_amount'] <- ""
```

```r
data_from_python <- within(data_from_python, N_amount[Treatment == 'LD'|Treatment == 'LA'|Treatment == 'LW'] <- 'low N')
data_from_python <- within(data_from_python, N_amount[Treatment == 'AD'|Treatment == 'AA'|Treatment == 'AW'] <- 'common N')
data_from_python <- within(data_from_python, N_amount[Treatment == 'HD'|Treatment == 'HA'|Treatment == 'HW'] <- 'high N')
data_from_python['Wetness'] <- ""
data_from_python <- within(data_from_python, Wetness[Treatment == 'LD'|Treatment == 'AD'|Treatment == 'HD'] <- 'Drained,')
data_from_python <- within(data_from_python, Wetness[Treatment == 'LA'|Treatment == 'AA'|Treatment == 'HA'] <- 'Field conditions,')
data_from_python <- within(data_from_python, Wetness[Treatment == 'LW'|Treatment == 'AW'|Treatment == 'HW'] <- 'Saturated,')
data_from_python['True_treatment'] <- paste(data_from_python$Wetness, data_from_python$N_amount)

legend_order <- c("Drained, low N", "Drained, common N", "Drained, high N",
                  "Field conditions, low N", "Field conditions, common N", "Field conditions, high N",
                  "Saturated, low N", "Saturated, common N", "Saturated, high N")
data_from_python$True_treatment <- factor(data_from_python$True_treatment, levels = legend_order)

# Below are all the plots used in the thesis/paper
end_date <- as.Date('2022-11-16')    # These dates are to filter out, so that we get the dataframe that ends in a specific date
#end_date <- as.Date('2022-11-23')
#end_date <- as.Date('2023-01-04')

# Here I create new data frames so that I can inspect each water content more easily, also used for plotting
wet_df <- data_from_python[!(data_from_python$Treatment == "HA" | data_from_python$Treatment == "HD" | data_from_python$Treatment == "AA" | data_from_python$Treatment == "AD" | data_from_python$Treatment == "LA" | data_from_python$Treatment == "LD"), ]
field_df <- data_from_python[!(data_from_python$Treatment == "HW" | data_from_python$Treatment == "HD" | data_from_python$Treatment == "AW" | data_from_python$Treatment == "AD" | data_from_python$Treatment == "LW" | data_from_python$Treatment == "LD"), ]
drained_df <- data_from_python[!(data_from_python$Treatment == "HA" | data_from_python$Treatment == "HW" | data_from_python$Treatment == "AA" | data_from_python$Treatment == "AW" | data_from_python$Treatment == "LA" | data_from_python$Treatment == "LW"), ]
wet_df <- wet_df[wet_df$Date <= end_date, ]
field_df <- field_df[field_df$Date <= end_date, ]
drained_df <-  drained_df[drained_df$Date <= end_date, ]

# These are the plots used in the thesis/paper, fluxes and cumulated values
#summary_df <- aggregate(data_from_python$mg_cm2_h, by = list(data_from_python$Treatment), FUN = "mean")
#summary_df['kg_n2o_ha'] <- summary_df['x'] * 100000000 / 1000
#summary_df['kg_n2o-n_ha'] <- summary_df['kg_n2o_ha'] * 28/44

# Figures, combined
p1 <- ggplot(data = wet_df, aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group = True_treatment, color = True_treatment)) +
  geom_errorbar(aes(ymin = ifelse(avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d<0, 0, avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d),
                    ymax = avg_mg_N2O_N_m2_d + std_mg_N2O_N_m2_d), position = pd,
```

```
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  ylab(~ mg ~N[2]*O-N ~m^{-2} ~d^{-1}) +
  scale_color_manual(values = c("black", "red", "blue")) +
  my_theme +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="right",
        axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank(),
legend.title=element_blank(),
        axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black")) +
  annotate('text', x = -Inf, y = Inf, label = "a)", size = 3, hjust = -1, vjust = 2, family = 'EB Garamond') +
  theme(legend.justification = "left", legend.position = "right")

 p2 <- ggplot(data = field_df, aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group =
True_treatment, color = True_treatment)) +
  geom_errorbar(aes(ymin = ifelse(avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d<0, 0,
avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d),
                    ymax = avg_mg_N2O_N_m2_d + std_mg_N2O_N_m2_d), position = pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  ylab(~ mg ~N[2]*O-N ~m^{-2} ~d^{-1}) +
  scale_color_manual(values = c("black", "red", "blue")) +
  my_theme +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="right",
        axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank(),
legend.title=element_blank(),
        axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black")) +
  annotate('text', x = -Inf, y = Inf, label = "b)", size = 3, hjust = -1, vjust = 2, family = 'EB Garamond') +
  theme(legend.justification = "left", legend.position = "right")

 p3 <- ggplot(data = drained_df, aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group =
True_treatment, color = True_treatment)) +
  geom_errorbar(aes(ymin = ifelse(avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d<0, 0,
avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d),
                    ymax = avg_mg_N2O_N_m2_d + std_mg_N2O_N_m2_d), position = pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  ylab(~ mg ~N[2]*O-N ~m^{-2} ~d^{-1}) + xlab(" Time since start (days)") +
  scale_color_manual(values = c("black", "red", "blue")) +
  my_theme +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="right",
legend.title=element_blank(),
        axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black")) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  annotate('text', x = -Inf, y = Inf, label = "c)", size = 3, hjust = -1, vjust = 2, family = 'EB Garamond') +
  theme(legend.justification = "left", legend.position = "right")

 p4 <- ggplot(data = wet_df, aes(x = Day_number, y = avg_cumulative_mg_N2O_N_m2, group =
True_treatment, color = True_treatment)) +
  geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2<0, 0, avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2),
                    ymax = avg_cumulative_mg_N2O_N_m2 + std_cumulative_mg_N2O_N_m2), position =
pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  ylab(~ mg ~N[2]*O-N ~m^{-2} ~d^{-1}) + xlab(" Time since start (days)") +
```

```
  scale_color_manual(values = c("black", "red", "blue")) +
  my_theme +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="right",
        axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank(),
legend.title=element_blank(),
        axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black")) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  annotate('text', x = -Inf, y = Inf, label = "a)", size = 3, hjust = -1, vjust = 2, family = 'EB Garamond') +
  theme(legend.justification = "left", legend.position = "right")

p5 <- ggplot(data = field_df, aes(x = Day_number, y = avg_cumulative_mg_N2O_N_m2, group =
True_treatment, color = True_treatment)) +
  geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2<0, 0, avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2),
                    ymax = avg_cumulative_mg_N2O_N_m2 + std_cumulative_mg_N2O_N_m2), position =
pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  ylab(~ mg ~N[2]*O-N ~m^{-2} ~d^{-1}) +
  scale_color_manual(values = c("black", "red", "blue")) +
  my_theme +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="right",
        axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank(),
legend.title=element_blank(),
        axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black")) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  annotate('text', x = -Inf, y = Inf, label = "b)", size = 3, hjust = -1, vjust = 2, family = 'EB Garamond') +
  theme(legend.justification = "left", legend.position = "right")

p6 <- ggplot(data = drained_df, aes(x = Day_number, y = avg_cumulative_mg_N2O_N_m2, group =
True_treatment, color = True_treatment)) +
  geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2<0, 0, avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2),
                    ymax = avg_cumulative_mg_N2O_N_m2 + std_cumulative_mg_N2O_N_m2), position =
pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  ylab(~ mg ~N[2]*O-N ~m^{-2} ~d^{-1}) +xlab("Time since start (days)") +
  scale_color_manual(values = c("black", "red", "blue")) +
  my_theme +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="right",
legend.title=element_blank(),
        axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black")) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  annotate('text', x = -Inf, y = Inf, label = "c)", size = 3, hjust = -1, vjust = 2, family = 'EB Garamond') +
  theme(legend.justification = "left", legend.position = "right")

plot_grid(p1, p2, p3, ncol = 1, align = 'v', rel_heights =c(1, 1, 1.15))
ggsave("~/Documents/MacBook 2020/Geografi
KU/Speciale/Data/R/Figures_for_thesis/lab_fluxes_one_week.png", width = 12, height = 12, units = "cm",
dpi = 300)

plot_grid(p4, p5, p6, ncol = 1, align = 'v', rel_heights =c(1, 1, 1.15))
```

```
175  ggsave("~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Figures_for_thesis/lab_cumulated_one_week.png", width = 12, height = 12, units = "cm", dpi = 300)
176
177
178  wet_1_df <- data_from_python[!(data_from_python$Treatment == "HA" | data_from_python$Treatment == "HD" | data_from_python$Treatment == "AA" | data_from_python$Treatment == "AD" | data_from_python$Treatment == "LA" | data_from_python$Treatment == "LD") & data_from_python$Date <= as.Date('2022-11-16'), ]
179  wet_2_df <- data_from_python[!(data_from_python$Treatment == "HA" | data_from_python$Treatment == "HD" | data_from_python$Treatment == "AA" | data_from_python$Treatment == "AD" | data_from_python$Treatment == "LA" | data_from_python$Treatment == "LD") & data_from_python$Date <= as.Date('2022-11-23'), ]
180  wet_8_df <- data_from_python[!(data_from_python$Treatment == "HA" | data_from_python$Treatment == "HD" | data_from_python$Treatment == "AA" | data_from_python$Treatment == "AD" | data_from_python$Treatment == "LA" | data_from_python$Treatment == "LD") & data_from_python$Date <= as.Date('2023-01-04'), ]
181
182  field_1_df <- data_from_python[!(data_from_python$Treatment == "HW" | data_from_python$Treatment == "HD" | data_from_python$Treatment == "AW" | data_from_python$Treatment == "AD" | data_from_python$Treatment == "LW" | data_from_python$Treatment == "LD") & data_from_python$Date <= as.Date('2022-11-16'), ]
183  field_2_df <- data_from_python[!(data_from_python$Treatment == "HW" | data_from_python$Treatment == "HD" | data_from_python$Treatment == "AW" | data_from_python$Treatment == "AD" | data_from_python$Treatment == "LW" | data_from_python$Treatment == "LD") & data_from_python$Date <= as.Date('2022-11-23'), ]
184  field_8_df <- data_from_python[!(data_from_python$Treatment == "HW" | data_from_python$Treatment == "HD" | data_from_python$Treatment == "AW" | data_from_python$Treatment == "AD" | data_from_python$Treatment == "LW" | data_from_python$Treatment == "LD") & data_from_python$Date <= as.Date('2023-01-04'), ]
185
186  dry_1_df <- data_from_python[!(data_from_python$Treatment == "HA" | data_from_python$Treatment == "HW" | data_from_python$Treatment == "AA" | data_from_python$Treatment == "AW" | data_from_python$Treatment == "LA" | data_from_python$Treatment == "LW") & data_from_python$Date <= as.Date('2022-11-16'), ]
187  dry_2_df <- data_from_python[!(data_from_python$Treatment == "HA" | data_from_python$Treatment == "HW" | data_from_python$Treatment == "AA" | data_from_python$Treatment == "AW" | data_from_python$Treatment == "LA" | data_from_python$Treatment == "LW") & data_from_python$Date <= as.Date('2022-11-23'), ]
188  dry_8_df <- data_from_python[!(data_from_python$Treatment == "HA" | data_from_python$Treatment == "HW" | data_from_python$Treatment == "AA" | data_from_python$Treatment == "AW" | data_from_python$Treatment == "LA" | data_from_python$Treatment == "LW") & data_from_python$Date <= as.Date('2023-01-04'), ]
189
190  # Create the flux plots
191  a <- ggplot(aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group = True_treatment, color = True_treatment), data = dry_1_df) + my_theme +
192    geom_errorbar(aes(ymin = ifelse(avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d<0, 0, avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d),
193                      ymax = avg_mg_N2O_N_m2_d + std_mg_N2O_N_m2_d), position = pd,
194                  width = .2, linetype = 1, color = "gray", alpha = 0.5) +
195    geom_line(position = pd, linewidth = 0.4) +
196    xlab("") + ylab(~ mg ~N[2]*O-N ~m^{-2} ~d^{-1}) +
197    annotate('text', x = -Inf, y = Inf, label = "a) drained", size = 3, hjust = -0.2, vjust = 2, family = 'EB Garamond') +
```

```
198    theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none",
legend.title=element_blank(),
199          axis.title.x=element_blank(),
200          axis.text.x=element_blank(),
201          axis.ticks.x=element_blank(), axis.text=element_text(size=8, color="black"),
axis.title=element_text(size=8, color="black")) +
202    scale_color_manual(values = c("black", "red", "blue"))
203
204  b <- ggplot(aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group = True_treatment, color =
True_treatment), data = field_1_df) + my_theme +
205    geom_errorbar(aes(ymin = ifelse(avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d<0, 0,
avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d),
206                      ymax = avg_mg_N2O_N_m2_d + std_mg_N2O_N_m2_d), position = pd,
207                  width = .2, linetype = 1, color = "gray", alpha = 0.5) +
208    geom_line(position = pd, linewidth = 0.4) +
209    xlab("") + ylab(~ mg ~N[2]*O-N ~m^{-2} ~d^{-1}) +
210    annotate('text', x = -Inf, y = Inf, label = "b) field conditions", size = 3, hjust = -0.1, vjust = 2, family =
'EB Garamond') +
211    theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none",
legend.title=element_blank(),
212          axis.title.x=element_blank(),
213          axis.text.x=element_blank(),
214          axis.ticks.x=element_blank(), axis.text=element_text(size=8, color="black"),
axis.title=element_text(size=8, color="black")) +
215    scale_color_manual(values = c("black", "red", "blue"))
216
217  c <- ggplot(aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group = True_treatment, color =
True_treatment), data = wet_1_df) + my_theme +
218    geom_errorbar(aes(ymin = ifelse(avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d<0, 0,
avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d),
219                      ymax = avg_mg_N2O_N_m2_d + std_mg_N2O_N_m2_d), position = pd,
220                  width = .2, linetype = 1, color = "gray", alpha = 0.5) +
221    geom_line(position = pd, linewidth = 0.4) +
222    xlab("Time since start (days)") + ylab(~ mg ~N[2]*O-N ~m^{-2} ~d^{-1}) +
223    annotate('text', x = -Inf, y = Inf, label = "c) saturated", size = 3, hjust = -0.2, vjust = 2, family = 'EB
Garamond') +
224    theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none",
legend.title=element_blank(),
225          axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black")) +
226    scale_color_manual(values = c("black", "red", "blue"))+
227    scale_x_continuous(breaks = scales::pretty_breaks(n = 10))
228
229  d <- ggplot(aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group = True_treatment, color =
True_treatment), data = dry_2_df) + my_theme +
230    geom_errorbar(aes(ymin = ifelse(avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d<0, 0,
avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d),
231                      ymax = avg_mg_N2O_N_m2_d + std_mg_N2O_N_m2_d), position = pd,
232                  width = .2, linetype = 1, color = "gray", alpha = 0.5) +
233    geom_line(position = pd, linewidth = 0.4) +
234    xlab("") +
235    annotate('text', x = -Inf, y = Inf, label = "d) drained", size = 3, hjust = -0.2, vjust = 2, family = 'EB
Garamond') +
236    theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none",
legend.title=element_blank(),
237          axis.title.x=element_blank(),
238          axis.text.x=element_blank(),
```

```
        axis.ticks.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.title.y = element_blank()) +
  scale_color_manual(values = c("black", "red", "blue"))

e <- ggplot(aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group = True_treatment, color = True_treatment), data = field_2_df) + my_theme +
  geom_errorbar(aes(ymin = ifelse(avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d<0, 0, avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d),
                    ymax = avg_mg_N2O_N_m2_d + std_mg_N2O_N_m2_d), position = pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  xlab("") +
  annotate('text', x = -Inf, y = Inf, label = "e) field conditions", size = 3, hjust = -0.1, vjust = 2, family = 'EB Garamond') +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none", legend.title=element_blank(),
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.title.y = element_blank()) +
  scale_color_manual(values = c("black", "red", "blue"))

f <- ggplot(aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group = True_treatment, color = True_treatment), data = wet_2_df) + my_theme +
  geom_errorbar(aes(ymin = ifelse(avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d<0, 0, avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d),
                    ymax = avg_mg_N2O_N_m2_d + std_mg_N2O_N_m2_d), position = pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  xlab("Time since start (days)") +
  annotate('text', x = -Inf, y = Inf, label = "f) saturated", size = 3, hjust = -0.2, vjust = 2, family = 'EB Garamond') +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none", legend.title=element_blank(),
        axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black"), axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.title.y = element_blank()) +
  scale_color_manual(values = c("black", "red", "blue")) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 7))

g <- ggplot(aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group = True_treatment, color = True_treatment), data = dry_8_df) + my_theme +
  geom_errorbar(aes(ymin = ifelse(avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d<0, 0, avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d),
                    ymax = avg_mg_N2O_N_m2_d + std_mg_N2O_N_m2_d), position = pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  xlab("") +
  annotate('text', x = -Inf, y = Inf, label = "g) drained", size = 3, hjust = -0.2, vjust = 2, family = 'EB Garamond') +
```

```
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none", legend.title=element_blank(),
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.title.y = element_blank()) +
  scale_color_manual(values = c("black", "red", "blue"))

 h <- ggplot(aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group = True_treatment, color = True_treatment), data = field_8_df) + my_theme +
  geom_errorbar(aes(ymin = ifelse(avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d<0, 0, avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d),
                    ymax = avg_mg_N2O_N_m2_d + std_mg_N2O_N_m2_d), position = pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  xlab("") +
  annotate('text', x = -Inf, y = Inf, label = "h) field conditions", size = 3, hjust = -0.1, vjust = 2, family = 'EB Garamond') +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none", legend.title=element_blank(),
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.title.y = element_blank()) +
  scale_color_manual(values = c("black", "red", "blue"))

 i <- ggplot(aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group = True_treatment, color = True_treatment), data = wet_8_df) + my_theme +
  geom_errorbar(aes(ymin = ifelse(avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d<0, 0, avg_mg_N2O_N_m2_d - std_mg_N2O_N_m2_d),
                    ymax = avg_mg_N2O_N_m2_d + std_mg_N2O_N_m2_d), position = pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  xlab("Time since start (days)") +
  annotate('text', x = -Inf, y = Inf, label = "i) saturated", size = 3, hjust = -0.2, vjust = 2, family = 'EB Garamond') +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none", legend.title=element_blank(),
        axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black"),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.title.y = element_blank()) +
  scale_color_manual(values = c("black", "red", "blue")) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 7))

 # I create a copy only to get the legend
 wet_copy <- wet_1_df
 wet_copy$True_treatment <- gsub("Saturated, high N", "High N    ", wet_copy$True_treatment)
 wet_copy$True_treatment <- gsub("Saturated, low N", "Low N    ", wet_copy$True_treatment)
 wet_copy$True_treatment <- gsub("Saturated, common N", "Common N    ", wet_copy$True_treatment)
```

```
i_copy <- ggplot(aes(x = Day_number, y = avg_mg_N2O_N_m2_d, group = True_treatment, color = True_treatment), data = wet_copy) + my_theme +
  geom_line(position = pd, linewidth = 0.4) + scale_color_manual(values = c("black", "red", "blue"), labels = c("Low N    ", "Common N    ", "High N    ")) +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="bottom", legend.title=element_blank(),
        axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black"),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())

# Combination of flux-plots
legend_t <- get_legend(i_copy)
grid <- plot_grid(a, d, g,
          b, e, h,
          c, f, i,
          ncol = 3,
          rel_widths = c(0.9, 1, 1.2),
          rel_heights = c(1, 1, 1.15),
          align="v",
          axis = "lbt")
grid2 <- plot_grid(grid, legend_t, ncol = 1,
          rel_heights = c(1, 0.1))
grid3 <- grid2 + theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position = 'bottom')
ggsave("~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Figures_for_thesis/lab_fluxes_all.png", width = 15, height = 12, units = "cm", dpi = 300)

# Cumulated fluxes
ac <- ggplot(aes(x = Day_number, y = avg_cumulative_mg_N2O_N_m2, group = True_treatment, color = True_treatment), data = dry_1_df) + my_theme +
  geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N_m2 - std_cumulative_mg_N2O_N_m2<0, 0, avg_cumulative_mg_N2O_N_m2 - std_cumulative_mg_N2O_N_m2),
                    ymax = avg_cumulative_mg_N2O_N_m2 + std_cumulative_mg_N2O_N_m2), position = pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  xlab("") + ylab(~ mg ~N[2]*O-N ~m^{-2} ~d^{-1}) +
  annotate('text', x = -Inf, y = Inf, label = "a) drained", size = 3, hjust = -0.2, vjust = 2, family = 'EB Garamond') +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none", legend.title=element_blank(),
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(), axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black")) +
  scale_color_manual(values = c("black", "red", "blue"))

bc <- ggplot(aes(x = Day_number, y = avg_cumulative_mg_N2O_N_m2, group = True_treatment, color = True_treatment), data = field_1_df) + my_theme +
  geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N_m2 - std_cumulative_mg_N2O_N_m2<0, 0, avg_cumulative_mg_N2O_N_m2 - std_cumulative_mg_N2O_N_m2),
                    ymax = avg_cumulative_mg_N2O_N_m2 + std_cumulative_mg_N2O_N_m2), position = pd,
```

```
366                 width = .2, linetype = 1, color = "gray", alpha = 0.5) +
367   geom_line(position = pd, linewidth = 0.4) +
368   xlab("") + ylab(~ mg ~N[2]*O-N ~m^{-2} ~d^{-1}) +
369   annotate('text', x = -Inf, y = Inf, label = "b) field conditions", size = 3, hjust = -0.1, vjust = 2, family =
'EB Garamond') +
370   theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none",
legend.title=element_blank(),
371         axis.title.x=element_blank(),
372         axis.text.x=element_blank(),
373         axis.ticks.x=element_blank(), axis.text=element_text(size=8, color="black"),
axis.title=element_text(size=8, color="black")) +
374   scale_color_manual(values = c("black", "red", "blue"))
375
376  cc <- ggplot(aes(x = Day_number, y = avg_cumulative_mg_N2O_N_m2, group = True_treatment,
color = True_treatment), data = wet_1_df) + my_theme +
377   geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2<0, 0, avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2),
378                     ymax = avg_cumulative_mg_N2O_N_m2 + std_cumulative_mg_N2O_N_m2), position =
pd,
379                 width = .2, linetype = 1, color = "gray", alpha = 0.5) +
380   geom_line(position = pd, linewidth = 0.4) +
381   xlab("Time since start (days)") + ylab(~ mg ~N[2]*O-N ~m^{-2} ~d^{-1}) +
382   annotate('text', x = -Inf, y = Inf, label = "c) saturated", size = 3, hjust = -0.2, vjust = 2, family = 'EB
Garamond') +
383   theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none",
legend.title=element_blank(),
384         axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black")) +
385   scale_color_manual(values = c("black", "red", "blue"))+
386   scale_x_continuous(breaks = scales::pretty_breaks(n = 10))
387
388  dc <- ggplot(aes(x = Day_number, y = avg_cumulative_mg_N2O_N_m2, group = True_treatment,
color = True_treatment), data = dry_2_df) + my_theme +
389   geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2<0, 0, avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2),
390                     ymax = avg_cumulative_mg_N2O_N_m2 + std_cumulative_mg_N2O_N_m2), position =
pd,
391                 width = .2, linetype = 1, color = "gray", alpha = 0.5) +
392   geom_line(position = pd, linewidth = 0.4) +
393   xlab("") +
394   annotate('text', x = -Inf, y = Inf, label = "d) drained", size = 3, hjust = -0.2, vjust = 2, family = 'EB
Garamond') +
395   theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none",
legend.title=element_blank(),
396         axis.title.x=element_blank(),
397         axis.text.x=element_blank(),
398         axis.ticks.x=element_blank(),
399         axis.text.y=element_blank(),
400         axis.ticks.y=element_blank(),
401         axis.title.y = element_blank()) +
402   scale_color_manual(values = c("black", "red", "blue"))
403
404  ec <- ggplot(aes(x = Day_number, y = avg_cumulative_mg_N2O_N_m2, group = True_treatment,
color = True_treatment), data = field_2_df) + my_theme +
```

```
405    geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2<0, 0, avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2),
406                      ymax = avg_cumulative_mg_N2O_N_m2 + std_cumulative_mg_N2O_N_m2), position =
pd,
407                  width = .2, linetype = 1, color = "gray", alpha = 0.5) +
408    geom_line(position = pd, linewidth = 0.4) +
409    xlab("") +
410    annotate('text', x = -Inf, y = Inf, label = "e) field conditions", size = 3, hjust = -0.1, vjust = 2, family =
'EB Garamond') +
411    theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none",
legend.title=element_blank(),
412          axis.title.x=element_blank(),
413          axis.text.x=element_blank(),
414          axis.ticks.x=element_blank(),
415          axis.text.y=element_blank(),
416          axis.ticks.y=element_blank(),
417          axis.title.y = element_blank()) +
418    scale_color_manual(values = c("black", "red", "blue"))
419
420  fc <- ggplot(aes(x = Day_number, y = avg_cumulative_mg_N2O_N_m2, group = True_treatment,
color = True_treatment), data = wet_2_df) + my_theme +
421    geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2<0, 0, avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2),
422                      ymax = avg_cumulative_mg_N2O_N_m2 + std_cumulative_mg_N2O_N_m2), position =
pd,
423                  width = .2, linetype = 1, color = "gray", alpha = 0.5) +
424    geom_line(position = pd, linewidth = 0.4) +
425    xlab("Time since start (days)") +
426    annotate('text', x = -Inf, y = Inf, label = "f) saturated", size = 3, hjust = -0.2, vjust = 2, family = 'EB
Garamond') +
427    theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none",
legend.title=element_blank(),
428          axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black"),
axis.text.y=element_blank(),
429          axis.ticks.y=element_blank(),
430          axis.title.y = element_blank()) +
431    scale_color_manual(values = c("black", "red", "blue")) +
432    scale_x_continuous(breaks = scales::pretty_breaks(n = 7))
433
434  gc <- ggplot(aes(x = Day_number, y = avg_cumulative_mg_N2O_N_m2, group = True_treatment,
color = True_treatment), data = dry_8_df) + my_theme +
435    geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2<0, 0, avg_cumulative_mg_N2O_N_m2 -
std_cumulative_mg_N2O_N_m2),
436                      ymax = avg_cumulative_mg_N2O_N_m2 + std_cumulative_mg_N2O_N_m2), position =
pd,
437                  width = .2, linetype = 1, color = "gray", alpha = 0.5) +
438    geom_line(position = pd, linewidth = 0.4) +
439    xlab("") +
440    annotate('text', x = -Inf, y = Inf, label = "g) drained", size = 3, hjust = -0.2, vjust = 2, family = 'EB
Garamond') +
441    theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none",
legend.title=element_blank(),
442          axis.title.x=element_blank(),
```

```
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.title.y = element_blank()) +
  scale_color_manual(values = c("black", "red", "blue"))

hc <- ggplot(aes(x = Day_number, y = avg_cumulative_mg_N2O_N_m2, group = True_treatment, color = True_treatment), data = field_8_df) + my_theme +
  geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N_m2 - std_cumulative_mg_N2O_N_m2<0, 0, avg_cumulative_mg_N2O_N_m2 - std_cumulative_mg_N2O_N_m2),
                    ymax = avg_cumulative_mg_N2O_N_m2 + std_cumulative_mg_N2O_N_m2), position = pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  xlab("") +
  annotate('text', x = -Inf, y = Inf, label = "h) field conditions", size = 3, hjust = -0.1, vjust = 2, family = 'EB Garamond') +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none", legend.title=element_blank(),
        axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.title.y = element_blank()) +
  scale_color_manual(values = c("black", "red", "blue"))

ic <- ggplot(aes(x = Day_number, y = avg_cumulative_mg_N2O_N_m2, group = True_treatment, color = True_treatment), data = wet_8_df) + my_theme +
  geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N_m2 - std_cumulative_mg_N2O_N_m2<0, 0, avg_cumulative_mg_N2O_N_m2 - std_cumulative_mg_N2O_N_m2),
                    ymax = avg_cumulative_mg_N2O_N_m2 + std_cumulative_mg_N2O_N_m2), position = pd,
                width = .2, linetype = 1, color = "gray", alpha = 0.5) +
  geom_line(position = pd, linewidth = 0.4) +
  xlab("Time since start (days)") +
  annotate('text', x = -Inf, y = Inf, label = "i) saturated", size = 3, hjust = -0.2, vjust = 2, family = 'EB Garamond') +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="none", legend.title=element_blank(),
        axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black"),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank(),
        axis.title.y = element_blank()) +
  scale_color_manual(values = c("black", "red", "blue")) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 7))

gridc <- plot_grid(ac, dc, gc,
                   bc, ec, hc,
                   cc, fc, ic,
                   ncol = 3,
                   rel_widths = c(0.9, 1, 1.2),
                   rel_heights = c(1, 1, 1.15),
```

```
                  align="v",
                  axis = "lbt")
grid2c <- plot_grid(gridc, legend_t, ncol = 1,
                    rel_heights = c(1, 0.1))
grid3c <- grid2c + theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position = 'bottom')
ggsave("~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Figures_for_thesis/lab_cumulated_fluxes_all.png", width = 15, height = 12, units = "cm", dpi = 300)

# Here I inspect the total lost of N2O-N, below is just to make calculation easier
last_date <- '2022-11-29'
total_loss <- data_from_python[data_from_python$Date == last_date,]
total_loss <- select(total_loss, Treatment, Date, avg_cumulative_mg_N2O_N, std_cumulative_mg_N2O_N)
total_loss$input[total_loss$Treatment=='AA'] <- 19.2
total_loss$input[total_loss$Treatment=='AD'] <- 19.2
total_loss$input[total_loss$Treatment=='AW'] <- 19.2
total_loss$input[total_loss$Treatment=='HA'] <- 57.7
total_loss$input[total_loss$Treatment=='HD'] <- 57.7
total_loss$input[total_loss$Treatment=='HW'] <- 57.7
total_loss$input[total_loss$Treatment=='LA'] <- 3.8
total_loss$input[total_loss$Treatment=='LD'] <- 3.8
total_loss$input[total_loss$Treatment=='LW'] <- 3.8
total_loss$samples <- 3
total_loss$percent_loss <- total_loss$avg_cumulative_mg_N2O_N/total_loss$input * 100  # Here we calculate the emission factor for N2O
total_loss$Treatment <- factor(total_loss$Treatment, levels = c("LD", "LA", "LW", "AD", "AA", "AW", "HD", "HA", "HW"))
total_loss$percent_loss <- round(total_loss$percent_loss ,digit=2)

start <- as.Date(min(data_from_python$Date))
end <- as.Date(last_date)
duration <- length(seq(from =start, to = end, by = "day"))

ggplot(total_loss, aes(y = avg_cumulative_mg_N2O_N, x = Treatment, label=percent_loss)) + theme_article() +
  geom_point(shape = 15, size = 2) +
  geom_text(aes(label = paste(percent_loss, "%"), y = ifelse(avg_cumulative_mg_N2O_N + (std_cumulative_mg_N2O_N/1) > 5,  avg_cumulative_mg_N2O_N - (std_cumulative_mg_N2O_N/0.75), avg_cumulative_mg_N2O_N + (std_cumulative_mg_N2O_N/1)), vjust=-1))+
  geom_errorbar(aes(ymin = ifelse(avg_cumulative_mg_N2O_N - std_cumulative_mg_N2O_N<0, 0, avg_cumulative_mg_N2O_N - std_cumulative_mg_N2O_N), ymax = avg_cumulative_mg_N2O_N + std_cumulative_mg_N2O_N), color = 'black', width = 0.2)+
  annotate("text", x = 2.5, y = 5, label = paste("From",start, "to", end))+annotate("text", x = 1.42, y = 4.5, label = paste("Duration =", duration))+
  ylab(~Total ~N ~released ~as ~N[2]*O ~(mg))
ggsave("~/Documents/MacBook 2020/Geografi KU/Speciale/Presentations/total_loss.png", width =15, height = 10.5, units = "cm", dpi = 300)


# Plotting til Bo
plotting_data <- data_from_python
plotting_data <- plotting_data[plotting_data$Date <= as.Date('2023-01-04'),]
plotting_data['avg_ug_N2O_N_m2_h'] <- (plotting_data['avg_g_N2O_N_m2_d'] * 1000 * 1000)/24
plotting_data['std_ug_N2O_N_m2_h'] <- (plotting_data['std_g_N2O_N_m2_d'] * 1000 * 1000)/24
```

```r
plotting_data['se_ug_N2O_N_m2_h'] <- plotting_data['std_ug_N2O_N_m2_h']/sqrt(3)
keeps <- c('Date', 'Treatment', 'avg_ug_N2O_N_m2_h', 'se_ug_N2O_N_m2_h')
plotting_data <- plotting_data[keeps]
ref_date <- as.Date('2022-11-09')
plotting_data$Day_number <- difftime(plotting_data$Date, ref_date, units = 'days')
plotting_data$Day_number <- as.integer(plotting_data$Day_number)

wet_ug <- plotting_data[!(plotting_data$Treatment == "HA" | plotting_data$Treatment == "HD" | plotting_data$Treatment == "AA" | plotting_data$Treatment == "AD" | plotting_data$Treatment == "LA" | plotting_data$Treatment == "LD"), ]
wet_ug$Treatment <- factor(wet_ug$Treatment, levels = c('LW', 'AW', 'HW'))
not_wet_ug <- plotting_data[!(plotting_data$Treatment == "HW" | plotting_data$Treatment == "AW" | plotting_data$Treatment == "LW"), ]
not_wet_ug$Treatment <- factor(not_wet_ug$Treatment, levels = c('LD', 'LA', 'AD', 'AA', 'HD', 'HA'))
drained_ug <- plotting_data[!(plotting_data$Treatment == "HW" | plotting_data$Treatment == "AW" | plotting_data$Treatment == "LW" | plotting_data$Treatment == "HA" |plotting_data$Treatment == "AA" | plotting_data$Treatment == "LA"), ]
drained_ug$Treatment <- factor(drained_ug$Treatment, levels = c('LD', 'AD', 'HD'))
ambient_ug <- plotting_data[!(plotting_data$Treatment == "HW" | plotting_data$Treatment == "AW" | plotting_data$Treatment == "LW" | plotting_data$Treatment == "HD" |plotting_data$Treatment == "AD" | plotting_data$Treatment == "LD"), ]
ambient_ug$Treatment <- factor(ambient_ug$Treatment, levels = c('LA', 'AA', 'HA'))

wet <- ggplot(aes(x = Day_number, y = avg_ug_N2O_N_m2_h, group = Treatment, color = Treatment), data = wet_ug) +
  geom_line(position = pd) + theme_article() +
  geom_errorbar(aes(ymin = ifelse(avg_ug_N2O_N_m2_h - se_ug_N2O_N_m2_h<0, 0, avg_ug_N2O_N_m2_h - se_ug_N2O_N_m2_h), ymax = avg_ug_N2O_N_m2_h + se_ug_N2O_N_m2_h), position = pd, width = .2, color = 'grey', alpha = 0.5) +
  xlab(expression('Time since start (days)')) + ylab(~ mu~g ~N[2]*O-N ~m^{-2} ~h^{-1})+
  theme(legend.position = 'bottom') + theme(legend.title = element_blank())+
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 5)) +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'))
ggsave("~/Documents/MacBook 2020/Geografi KU/Speciale/Presentations/wet_mug.png", width = 20, height = 10.5, units = "cm", dpi = 300)

not_wet <- ggplot(aes(x = Day_number, y = avg_ug_N2O_N_m2_h, group = Treatment, color = Treatment), data = not_wet_ug) +
  geom_line(position = pd) + theme_article() +
  geom_errorbar(aes(ymin = ifelse(avg_ug_N2O_N_m2_h - se_ug_N2O_N_m2_h<0, 0, avg_ug_N2O_N_m2_h - se_ug_N2O_N_m2_h), ymax = avg_ug_N2O_N_m2_h + se_ug_N2O_N_m2_h), position = pd, width = .2, color = 'grey', alpha = 0.5) +
  xlab(expression('Time since start (days)')) + ylab(~ mu~g ~N[2]*O-N ~m^{-2} ~h^{-1})+
  theme(legend.position = 'bottom') + theme(legend.title = element_blank())+
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 5)) +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'))
ggsave("~/Documents/MacBook 2020/Geografi KU/Speciale/Presentations/not_wet_mug.png", width = 20, height = 10.5, units = "cm", dpi = 300)

drained <- ggplot(aes(x = Day_number, y = avg_ug_N2O_N_m2_h, group = Treatment, color = Treatment), data = drained_ug) +
  geom_line(position = pd) + theme_article() +
  geom_errorbar(aes(ymin = ifelse(avg_ug_N2O_N_m2_h - se_ug_N2O_N_m2_h<0, 0, avg_ug_N2O_N_m2_h - se_ug_N2O_N_m2_h), ymax = avg_ug_N2O_N_m2_h +
```

```
se_ug_N2O_N_m2_h), position = pd, width = .2, color = 'grey', alpha = 0.5) +
  xlab(expression('Time since start (days)')) + ylab(~ mu~g ~N[2]*O-N ~m^{-2} ~h^{-1})+
  theme(legend.position = 'bottom') + theme(legend.title = element_blank())+
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 5)) +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'))
ggsave("~/Documents/MacBook 2020/Geografi KU/Speciale/Presentations/drained_mug.png", width = 20, height = 10.5, units = "cm", dpi = 300)

ambient <- ggplot(aes(x = Day_number, y = avg_ug_N2O_N_m2_h, group = Treatment, color = Treatment), data = ambient_ug) +
  geom_line(position = pd) + theme_article() +
  geom_errorbar(aes(ymin = ifelse(avg_ug_N2O_N_m2_h - se_ug_N2O_N_m2_h<0, 0, avg_ug_N2O_N_m2_h - se_ug_N2O_N_m2_h), ymax = avg_ug_N2O_N_m2_h + se_ug_N2O_N_m2_h), position = pd, width = .2, color = 'grey', alpha = 0.5) +
  xlab(expression('Time since start (days)')) + ylab(~ mu~g ~N[2]*O-N ~m^{-2} ~h^{-1})+
  theme(legend.position = 'bottom') + theme(legend.title = element_blank())+
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 5)) +
  theme(plot.background = element_rect(fill = 'white', colour = 'white'))
ggsave("~/Documents/MacBook 2020/Geografi KU/Speciale/Presentations/ambient_mug.png", width = 20, height = 10.5, units = "cm", dpi = 300)

```

R_lmer.R

```r
install.packages("dplyr")
install.packages("tidyr")
install.packages("ggplot2")
install.packages("tidyverse")
install.packages("VGAM")
install.packages("pracma")
install.packages("egg")
install.packages("ggpubr")
install.packages('lme4')
 install.packages('broom')
 install.packages('multcomp')
 install.packages('multcompView')
 install.packages('emmeans')

 library(dplyr)
 library(tidyr)
 library(ggplot2)
 library(tidyverse)
 library(VGAM)
 library(pracma)
 library(egg)
 library(ggpubr)
 library(lme4)
 library(broom)
 library(multcomp)
 library(multcompView)
 library(emmeans)

 # This is for loading the data and converting columns to the right format
 loc_string <-"~/Documents/MacBook 2020/Geografi KU/Speciale/Data/Python/Speciale/Output_files/"
 loc_name <- "Computed_bottles.csv"
 file_location <- paste(loc_string, loc_name, sep="")
 flux_data <- read.csv(file_location, header=TRUE)
 flux_data <- na.omit(flux_data)
 flux_data$Date <- as.Date(flux_data$Date, format = "%Y-%m-%d")
 keeps <- c('Date', 'Bottle', 'X24h_2', 'mg_N2O_N_m2_d', 'cumulative_m2')
 flux_data <- flux_data[keeps]
 flux_data <- subset(flux_data, Date <= '2023-01-09')

 # Here I add the proper labels, such that legends will have proper layout
 flux_data['Treatment'] <- ''
 flux_data <- within(flux_data, Treatment[Bottle == 1 | Bottle == 2 | Bottle == 3] <- 'HW')
 flux_data <- within(flux_data, Treatment[Bottle == 4 | Bottle == 5 | Bottle == 6] <- 'HD')
 flux_data <- within(flux_data, Treatment[Bottle == 7 | Bottle == 8 | Bottle == 9] <- 'HA')
 flux_data <- within(flux_data, Treatment[Bottle == 10 | Bottle == 11 | Bottle == 12] <- 'LW')
 flux_data <- within(flux_data, Treatment[Bottle == 13 | Bottle == 14 | Bottle == 15] <- 'LD')
 flux_data <- within(flux_data, Treatment[Bottle == 16 | Bottle == 17 | Bottle == 18] <- 'AA')
 flux_data <- within(flux_data, Treatment[Bottle == 19 | Bottle == 20 | Bottle == 21] <- 'AW')
 flux_data <- within(flux_data, Treatment[Bottle == 22 | Bottle == 23 | Bottle == 24] <- 'AD')
 flux_data <- within(flux_data, Treatment[Bottle == 25 | Bottle == 26 | Bottle == 27] <- 'LA')
 flux_data <- flux_data[!(flux_data$Bottle == 28), ]
 flux_data <- flux_data %>%
   group_by(Bottle) %>%
```

```
  mutate(days = cumsum(X24h_2))
flux_data['Day_fac'] <- factor(flux_data$days)

log_flux <- flux_data[c('Date', 'Bottle', 'Treatment', 'Day_fac', 'mg_N2O_N_m2_d', 'cumulative_m2')]
log_flux['Input_mg_m2'] <- ''
log_flux <- within(log_flux, Input_mg_m2[Treatment == 'HW'|Treatment == 'HA'|Treatment == 'HD'] <- 15000)
log_flux <- within(log_flux, Input_mg_m2[Treatment == 'AW'|Treatment == 'AA'|Treatment == 'AD'] <- 5000)
log_flux <- within(log_flux, Input_mg_m2[Treatment == 'LW'|Treatment == 'LA'|Treatment == 'LD'] <- 1000)
log_flux$Input_mg_m2 <- as.integer(log_flux$Input_mg_m2)
log_flux['Emission_factor'] <- log_flux['cumulative_m2']/log_flux['Input_mg_m2']*100

test <- log_flux#[log_flux$Treatment == 'AD' | log_flux$Treatment == 'AW' | log_flux$Treatment == 'AA',]
test <- test[test$Day_fac == 57,]
summm <- test %>%
  group_by(Treatment) %>%
  summarise(mean_ef = mean(Emission_factor),
            sd_ef = sd(Emission_factor),
            mean_loss = mean(cumulative_m2))
summm

test$cumulative_m2

# The functions below are very slow, only run if necessary
# The functions below are the linear mixed models, which is used for repeated measures analysis.

# This line of code says that I should evaluate the difference in cumulated emissions between treatments by each day, and use the bottles as the random parameter
mixed <- lmer(log(cumulative_m2 + 1) ~ Treatment*Day_fac + (1|Bottle), data = log_flux)
emm <- emmeans(mixed, pairwise ~ Treatment|Day_fac, adjust = 'tukey')   # This takes in the mixed model and performs the test, using tukey test
cld_pairs <- cld(object = emm, Letters = letters, adjust = 'tukey', alpha = 0.05)  # This adds significance groups, which makes significant differences much easier to read
sink("~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Week_results/cumulated_signi_groups.txt")
print(cld_pairs)
sink()

emm2 <- emmeans(mixed, pairwise ~ Treatment|Day_fac, adust = 'tukey')
sink("~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Week_results/cumulated.txt")
summary(emm2)
sink()

emission_model <- lmer(log(Emission_factor + 1) ~ Treatment*Day_fac + (1|Bottle), data = log_flux)
emm3 <- emmeans(emission_model, pairwise ~ Treatment|Day_fac, adjust = 'tukey')
cld_pairs <- cld(object = emm3, Letters = letters, adjust = 'tukey', alpha = 0.05)
sink("~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Week_results/emission_groups.txt")
print(cld_pairs)
sink()

emm4 <- emmeans(emission_model, pairwise ~ Treatment|Day_fac, adust = 'tukey')
 sink("~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Week_results/emission.txt")
```

```r
summary(emm4)
sink()

flux_model <- lmer(log(mg_N2O_N_m2_d + 1) ~ Treatment*Day_fac + (1|Bottle), data = log_flux)
emm5 <- emmeans(flux_model, pairwise ~ Treatment|Day_fac, adjust = 'tukey')
cld_pairs <- cld(object = emm5, Letters = letters, adjust = 'tukey', alpha = 0.05)
sink("~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Week_results/flux_groups.txt")
print(cld_pairs)
sink()

emm6 <- emmeans(flux_model, pairwise ~ Treatment|Day_fac, adust = 'tukey')
sink("~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Week_results/flux.txt")
summary(emm6)
sink()

```

R_1_2_8_weeks.R

```r
install.packages("dplyr")
install.packages("tidyr")
install.packages("ggplot2")
install.packages("readxl")
install.packages("tidypaleo")
install.packages("tidyverse")
install.packages("gasfluxes")
install.packages("VGAM")
install.packages("pracma")
 install.packages("egg")
 install.packages("ggpubr")
 install.packages('reshape2')
 install.packages("patchwork")
 install.packages("cowplot")
 install.packages('Hmisc')
 install.packages("mmtable2")
 install.packages("gridExtra")

 library(gridExtra)
 library(dplyr)
 library(tidyr)
 library(ggplot2)
 library(readxl)
 library(gasfluxes)
 library(tidyverse)
 library(tidypaleo)
 library(pracma)
 library(egg)
 library(ggpubr)
 library(reshape2)
 library(data.table)
 library(patchwork)
 library(cowplot)
 library(Hmisc)
 library(gridExtra)
 library(grid)
 library(forcats)

 # This is for loading the data and converting columns to the right format
 loc_string <-"~/Documents/MacBook 2020/Geografi KU/Speciale/Data/Python/Speciale/Output_files/"
 loc_name <- "Computed_bottles.csv"
 file_location <- paste(loc_string, loc_name, sep="")
 flux_data <- read.csv(file_location, header=TRUE)
 flux_data <- na.omit(flux_data)
 flux_data$Date <- as.Date(flux_data$Date, format = "%Y-%m-%d")

 # Here I add the proper labels, such that legends will have proper layout
 flux_data['Treatment'] <- ''
 flux_data <- within(flux_data, Treatment[Bottle == 1 | Bottle == 2 | Bottle == 3] <- 'HW')
 flux_data <- within(flux_data, Treatment[Bottle == 4 | Bottle == 5 | Bottle == 6] <- 'HD')
 flux_data <- within(flux_data, Treatment[Bottle == 7 | Bottle == 8 | Bottle == 9] <- 'HA')
 flux_data <- within(flux_data, Treatment[Bottle == 10 | Bottle == 11 | Bottle == 12] <- 'LW')
 flux_data <- within(flux_data, Treatment[Bottle == 13 | Bottle == 14 | Bottle == 15] <- 'LD')
```

```r
flux_data <- within(flux_data, Treatment[Bottle == 16 | Bottle == 17 | Bottle == 18] <- 'LA')
flux_data <- within(flux_data, Treatment[Bottle == 19 | Bottle == 20 | Bottle == 21] <- 'AW')
flux_data <- within(flux_data, Treatment[Bottle == 22 | Bottle == 23 | Bottle == 24] <- 'AD')
flux_data <- within(flux_data, Treatment[Bottle == 25 | Bottle == 26 | Bottle == 27] <- 'AA')
flux_data <- flux_data[!(flux_data$Bottle == 28), ]
flux_data['Input_N_mg'] <- ''
flux_data <- within(flux_data, Input_N_mg[Treatment == 'HW' | Treatment == 'HD' | Treatment == 'HA'] <- 15000)
flux_data <- within(flux_data, Input_N_mg[Treatment == 'LW' | Treatment == 'LD' | Treatment == 'LA'] <- 1000)
flux_data <- within(flux_data, Input_N_mg[Treatment == 'AW' | Treatment == 'AD' | Treatment == 'AA'] <- 5000)
flux_data$Input_N_mg <- as.numeric(flux_data$Input_N_mg)

# This is for making the dataframe nicer to work with
flux_data['date_weight'] <- flux_data['X24h']
flux_data['ug_N2O_N_m2_h'] <- (flux_data['g_N2O_N_m2_d'] * 1000*1000)/24
flux_data['cumulative_ug_N2O_N_m2'] <- flux_data['cumulative_m2'] * 1000
keeps <- c('Date', 'Bottle', 'Treatment', 'date_weight', 'ug_N2O_N_m2_h', 'cumulative_ug_N2O_N_m2', 'Input_N_mg')
flux_data <- flux_data[keeps]

# I make subsets which are good to work with, so I can create summary tables
one_week <- subset(flux_data, Date <= '2022-11-16')
two_weeks <- subset(flux_data, Date <= '2022-11-23')
eight_weeks <- subset(flux_data, Date <= '2023-01-04')
eight_weeks$date_weight[eight_weeks$Date == '2023-01-02'] <- 6

# Below I create summary tables for each period. This is used for the large table i the results section,
# all lines until next comment is the same but for different parameters
one_week_results <- one_week %>%
  group_by(Treatment) %>%
  summarise(avg_flux = weighted.mean(ug_N2O_N_m2_h, date_weight),
            weighted_std = sqrt(wtd.var(ug_N2O_N_m2_h, date_weight)),
            count = n())
one_week_results['Duration'] <- 'One week'

two_weeks_results <- two_weeks %>%
  group_by(Treatment) %>%
  summarise(avg_flux = weighted.mean(ug_N2O_N_m2_h, date_weight),
            weighted_std = sqrt(wtd.var(ug_N2O_N_m2_h, date_weight)),
            count = n())
two_weeks_results['Duration'] <- 'Two weeks'

eight_weeks_results <- eight_weeks %>%
  group_by(Treatment) %>%
  summarise(avg_flux = weighted.mean(ug_N2O_N_m2_h, date_weight),
            weighted_std = sqrt(wtd.var(ug_N2O_N_m2_h, date_weight)),
            count = n())
eight_weeks_results['Duration'] <- 'Eight weeks'

results <- rbind(one_week_results, two_weeks_results, eight_weeks_results)
write.csv(results, "~/Documents/MacBook 2020/GeografiKU/Speciale/Data/R/Week_results/1_2_8_week_results.csv", row.names=FALSE)

```

```r
one_week_max <- one_week %>%
  group_by(Treatment) %>%
  summarise(max_flux = max(ug_N2O_N_m2_h))
one_week_max['Duration'] <- 'One week'
two_weeks_max <- two_weeks %>%
  group_by(Treatment) %>%
  summarise(max_flux = max(ug_N2O_N_m2_h))
two_weeks_max['Duration'] <- 'Two weeks'
eight_weeks_max <- eight_weeks %>%
  group_by(Treatment) %>%
  summarise(max_flux = max(ug_N2O_N_m2_h))
eight_weeks_max['Duration'] <- 'Eight weeks'
max_results <- rbind(one_week_max, two_weeks_max, eight_weeks_max)
write.csv(max_results, "~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Week_results/1_2_8_week_results_max.csv", row.names=FALSE)

one_week_min <- one_week %>%
  group_by(Treatment) %>%
  summarise(min_flux = min(ug_N2O_N_m2_h))
one_week_min['Duration'] <- 'One week'
two_weeks_min <- two_weeks %>%
  group_by(Treatment) %>%
  summarise(min_flux = min(ug_N2O_N_m2_h))
two_weeks_min['Duration'] <- 'Two weeks'
eight_weeks_min <- eight_weeks %>%
  group_by(Treatment) %>%
  summarise(min_flux = min(ug_N2O_N_m2_h))
eight_weeks_min['Duration'] <- 'Eight weeks'
min_results <- rbind(one_week_min, two_weeks_min, eight_weeks_min)
write.csv(min_results, "~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Week_results/1_2_8_week_results_min.csv", row.names=FALSE)

eight_weeks <- eight_weeks[eight_weeks$Date == '2023-01-02',]
eight_weeks['cumulative_mg_N2O_N_m2'] <- eight_weeks['cumulative_ug_N2O_N_m2']/1000
eight_weeks['percent_loss'] <- eight_weeks['cumulative_mg_N2O_N_m2']/eight_weeks['Input_N_mg'] * 100
two_weeks <- two_weeks[two_weeks$Date == '2022-11-23',]
two_weeks['cumulative_mg_N2O_N_m2'] <- two_weeks['cumulative_ug_N2O_N_m2']/1000
two_weeks['percent_loss'] <- two_weeks['cumulative_mg_N2O_N_m2']/two_weeks['Input_N_mg'] * 100
one_week <- one_week[one_week$Date == '2022-11-16',]
one_week['cumulative_mg_N2O_N_m2'] <- one_week['cumulative_ug_N2O_N_m2']/1000
one_week['percent_loss'] <- one_week['cumulative_mg_N2O_N_m2']/one_week['Input_N_mg'] * 100

eight_weeks_loss <- eight_weeks %>%
  group_by(Treatment) %>%
  summarise(avg_loss = mean(percent_loss),
            std_loss = std(percent_loss))
eight_weeks_loss['Duration'] <- 'Eight weeks'

two_weeks_loss <- two_weeks %>%
  group_by(Treatment) %>%
  summarise(avg_loss = mean(percent_loss),
            std_loss = std(percent_loss))
two_weeks_loss['Duration'] <- 'Two weeks'

```

```r
one_week_loss <- one_week %>%
  group_by(Treatment) %>%
  summarise(avg_loss = mean(percent_loss),
            std_loss = std(percent_loss))
one_week_loss['Duration'] <- 'One week'

loss_results <- rbind(one_week_loss, two_weeks_loss, eight_weeks_loss)
write.csv(loss_results, "~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Week_results/1_2_8_week_loss.csv", row.names=FALSE)

eight_weeks_loss_mg <- eight_weeks %>%
  group_by(Treatment) %>%
  summarise(avg_loss = mean(cumulative_mg_N2O_N_m2),
            std_loss = std(cumulative_mg_N2O_N_m2))
eight_weeks_loss_mg['Duration'] <- 'Eight weeks'

two_weeks_loss_mg <- two_weeks %>%
  group_by(Treatment) %>%
  summarise(avg_loss = mean(cumulative_mg_N2O_N_m2),
            std_loss = std(cumulative_mg_N2O_N_m2))
two_weeks_loss_mg['Duration'] <- 'Two weeks'

one_week_loss_mg <- one_week %>%
  group_by(Treatment) %>%
  summarise(avg_loss = mean(cumulative_mg_N2O_N_m2),
            std_loss = std(cumulative_mg_N2O_N_m2))
one_week_loss_mg['Duration'] <- 'One week'
loss_mg <- rbind(one_week_loss_mg, two_weeks_loss_mg, eight_weeks_loss_mg)
write.csv(loss_mg, "~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Week_results/1_2_8_week_loss_mg.csv", row.names=FALSE)

total_df <- merge(results, max_results, by = c('Treatment', 'Duration'))
total_df <- merge(total_df, min_results, by = c('Treatment', 'Duration'))
total_df <- merge(total_df, loss_mg, by = c('Treatment', 'Duration'))
total_df <- merge(total_df, loss_results, by = c('Treatment', 'Duration'))

write.csv(total_df, "~/Documents/MacBook 2020/Geografi KU/Speciale/Data/R/Week_results/1_2_8_all.csv", row.names=FALSE)

#plots, only emission factors, this is not used in the paper, but the values are reported in a table

#This is to add significance groups to each treatment
one_week_signi <- data.frame(
  Treatment = c('LD', 'LA', 'LW', 'AD', 'AA', 'AW', 'HD', 'HA', 'HW'),
  signi = c('a', 'ab', 'b', 'a', 'a', 'ab', 'a', 'a', 'ab'))
two_weeks_signi <- data.frame(
  Treatment = c('LD', 'LA', 'LW', 'AD', 'AA', 'AW', 'HD', 'HA', 'HW'),
  signi = c('ab', 'bc', 'c', 'a', 'a', 'c', 'a', 'ab', 'c'))
eight_weeks_signi <- data.frame(
  Treatment = c('LD', 'LA', 'LW', 'AD', 'AA', 'AW', 'HD', 'HA', 'HW'),
  signi = c('ab', 'bc', 'cd', 'a', 'a', 'd', 'a', 'ab', 'd'))

# Here I merge with the summary table calculated above
one_week_signi <- merge(one_week_signi, one_week_loss, by = 'Treatment')
two_weeks_signi <- merge(two_weeks_signi, two_weeks_loss, by = 'Treatment')
```

```r
eight_weeks_signi <- merge(eight_weeks_signi, eight_weeks_loss, by = 'Treatment')

# Add proper labels
one_week_signi['N_amount'] <- ""
one_week_signi <- within(one_week_signi, N_amount[Treatment == 'LD'|Treatment == 'LA'|Treatment == 'LW'] <- 'Low N')
one_week_signi <- within(one_week_signi, N_amount[Treatment == 'AD'|Treatment == 'AA'|Treatment == 'AW'] <- 'Normal N')
one_week_signi <- within(one_week_signi, N_amount[Treatment == 'HD'|Treatment == 'HA'|Treatment == 'HW'] <- 'High N')
one_week_signi['Wetness'] <- ""
one_week_signi <- within(one_week_signi, Wetness[Treatment == 'LD'|Treatment == 'AD'|Treatment == 'HD'] <- 'Drained')
one_week_signi <- within(one_week_signi, Wetness[Treatment == 'LA'|Treatment == 'AA'|Treatment == 'HA'] <- 'Field conditions')
one_week_signi <- within(one_week_signi, Wetness[Treatment == 'LW'|Treatment == 'AW'|Treatment == 'HW'] <- 'Saturated')
one_week_signi <- one_week_signi %>%
  arrange(factor(N_amount, levels = c("Low N", "Normal N", "High N")),
          Wetness) %>%
  mutate(N_amount = fct_relevel(N_amount, "Low N", "Normal N", "High N"))

two_weeks_signi['N_amount'] <- ""
two_weeks_signi <- within(two_weeks_signi, N_amount[Treatment == 'LD'|Treatment == 'LA'|Treatment == 'LW'] <- 'Low N')
two_weeks_signi <- within(two_weeks_signi, N_amount[Treatment == 'AD'|Treatment == 'AA'|Treatment == 'AW'] <- 'Normal N')
two_weeks_signi <- within(two_weeks_signi, N_amount[Treatment == 'HD'|Treatment == 'HA'|Treatment == 'HW'] <- 'High N')
two_weeks_signi['Wetness'] <- ""
two_weeks_signi <- within(two_weeks_signi, Wetness[Treatment == 'LD'|Treatment == 'AD'|Treatment == 'HD'] <- 'Drained')
two_weeks_signi <- within(two_weeks_signi, Wetness[Treatment == 'LA'|Treatment == 'AA'|Treatment == 'HA'] <- 'Field conditions')
two_weeks_signi <- within(two_weeks_signi, Wetness[Treatment == 'LW'|Treatment == 'AW'|Treatment == 'HW'] <- 'Saturated')
two_weeks_signi <- two_weeks_signi %>%
  arrange(factor(N_amount, levels = c("Low N", "Normal N", "High N")),
          Wetness) %>%
  mutate(N_amount = fct_relevel(N_amount, "Low N", "Normal N", "High N"))

eight_weeks_signi['N_amount'] <- ""
eight_weeks_signi <- within(eight_weeks_signi, N_amount[Treatment == 'LD'|Treatment == 'LA'|Treatment == 'LW'] <- 'Low N')
eight_weeks_signi <- within(eight_weeks_signi, N_amount[Treatment == 'AD'|Treatment == 'AA'|Treatment == 'AW'] <- 'Normal N')
eight_weeks_signi <- within(eight_weeks_signi, N_amount[Treatment == 'HD'|Treatment == 'HA'|Treatment == 'HW'] <- 'High N')
eight_weeks_signi['Wetness'] <- ""
eight_weeks_signi <- within(eight_weeks_signi, Wetness[Treatment == 'LD'|Treatment == 'AD'|Treatment == 'HD'] <- 'Drained')
eight_weeks_signi <- within(eight_weeks_signi, Wetness[Treatment == 'LA'|Treatment == 'AA'|Treatment == 'HA'] <- 'Field conditions')
eight_weeks_signi <- within(eight_weeks_signi, Wetness[Treatment == 'LW'|Treatment == 'AW'|Treatment == 'HW'] <- 'Saturated')
eight_weeks_signi <- eight_weeks_signi %>%
```

```r
  arrange(factor(N_amount, levels = c("Low N", "Normal N", "High N")),
          Wetness) %>%
  mutate(N_amount = fct_relevel(N_amount, "Low N", "Normal N", "High N"))

# Plots
ggplot(one_week_signi, aes(x = N_amount, y = avg_loss)) +
  geom_point() +
  geom_errorbar(aes(ymin = avg_loss - std_loss, ymax = avg_loss + std_loss), width = 0.1) +
  geom_text(aes(label = signi, y = avg_loss + std_loss), vjust = -0.5) +
  facet_grid(. ~ Wetness, scales = "free_y") +
  theme_article() +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        strip.background = element_rect(colour = "white", fill = "white"),
        axis.text.x = element_text(margin = margin(t = 10)),
        panel.border = element_rect(colour = "NA"),
        axis.line = element_line(colour = "black"),
        axis.title = element_text(size = 10),
        strip.text = element_text(size = 10),
        plot.background = element_rect(fill = 'white', colour = 'white')) +
  labs(x = "", y = "Emission factor (%)") +
  geom_text(data = subset(one_week_signi, Wetness == "Drained"), x = -Inf , y = Inf, label = "a)",
size = 5, hjust = -1, vjust =2)+
  ylim(0, 9)
ggsave('~/Documents/MacBook 2020/Geografi
KU/Speciale/Data/R/Figures_for_thesis/emission_1.png', width = 15, height = 10, units = "cm", dpi = 300)

ggplot(two_weeks_signi, aes(x = N_amount, y = avg_loss)) +
  geom_point() +
  geom_errorbar(aes(ymin = avg_loss - std_loss, ymax = avg_loss + std_loss), width = 0.1) +
  geom_text(aes(label = signi, y = avg_loss + std_loss), vjust = -0.5) +
  facet_grid(. ~ Wetness, scales = "free_y") +
  theme_article() +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        strip.background = element_rect(colour = "white", fill = "white"),
        axis.text.x = element_text(margin = margin(t = 10)),
        panel.border = element_rect(colour = "NA"),
        axis.line = element_line(colour = "black"),
        axis.title = element_text(size = 10),
        strip.text = element_text(size = 10),
        plot.background = element_rect(fill = 'white', colour = 'white')) +
  labs(x = "", y = "Emission factor (%)") +
  geom_text(data = subset(two_weeks_signi, Wetness == "Drained"), x = -Inf , y = Inf, label = "b)",
size = 5, hjust = -1, vjust =2)+
  ylim(0, 15)
ggsave('~/Documents/MacBook 2020/Geografi
KU/Speciale/Data/R/Figures_for_thesis/emission_2.png', width = 15, height = 10, units = "cm", dpi = 300)

ggplot(eight_weeks_signi, aes(x = N_amount, y = avg_loss)) +
  geom_point() +
  geom_errorbar(aes(ymin = avg_loss - std_loss, ymax = avg_loss + std_loss), width = 0.1) +
  geom_text(aes(label = signi, y = avg_loss + std_loss), vjust = -0.5) +
  facet_grid(. ~ Wetness, scales = "free_y") +
  theme_article() +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        strip.background = element_rect(colour = NA, fill = NA),
        axis.text.x = element_text(margin = margin(t = 10)),
```

```
298         panel.border = element_rect(colour = "NA"),
299         axis.line = element_line(colour = "black"),
300         axis.title = element_text(size = 10),
301         strip.text = element_text(size = 10),
302         plot.background = element_rect(fill = 'white', colour = 'white')) +
303   labs(x = "", y = "Emission factor (%)") +
304   geom_text(data = subset(eight_weeks_signi, Wetness == "Drained"), x = -Inf , y = Inf, label = "c)",
size = 5, hjust = -1, vjust =2)+
305   ylim(0, 18)
306  ggsave('~/Documents/MacBook 2020/Geografi
KU/Speciale/Data/R/Figures_for_thesis/emission_8.png', width = 15, height = 10, units = "cm", dpi = 300)
307
```

Climate.R

```r
install.packages("dplyr")
install.packages("tidyr")
install.packages("ggplot2")
install.packages("readxl")
install.packages("tidypaleo")
install.packages("tidyverse")
install.packages("gasfluxes")
install.packages("VGAM")
install.packages("pracma")
 install.packages("egg")
 install.packages("ggpubr")

 library(gridExtra)
 library(dplyr)
 library(tidyr)
 library(ggplot2)
 library(readxl)
 library(gasfluxes)
 library(tidyverse)
 library(tidypaleo)
 library(pracma)
 library(egg)
 library(ggpubr)
 library(reshape2)
 library(data.table)
 library(patchwork)
 library(cowplot)
 library(Hmisc)
 library(gridExtra)
 library(grid)
 library(forcats)
 library(splines)
 my_theme <- theme_article() +
   theme(text = element_text(family = "EB Garamond"))

 # This is for loading the data and converting columns to the right format
 loc_string <-"~/Documents/MacBook 2020/Geografi KU/Speciale/Data/Excel/"
 loc_name <- "climate_soro.csv"
 file_location <- paste(loc_string, loc_name, sep="")
 climate_data <- read.csv(file_location, header=TRUE)
 climate_data$label <- factor(climate_data$label, levels = unique(climate_data$label), ordered =
TRUE)

 # Here I add the proper labels, such that legends will have proper layout
 climate_data['Season'] <- ""
 climate_data <- within(climate_data, Season[label == 'Jan'|label == 'Feb'|label == 'Dec'] <- 'Winter')
 climate_data <- within(climate_data, Season[label == 'Mar'|label == 'Apr'|label == 'May'] <- 'Spring')
 climate_data <- within(climate_data, Season[label == 'Jun'|label == 'Jul'|label == 'Aug'] <- 'Summer')
```

```r
climate_data <- within(climate_data, Season[label == 'Sep'|label == 'Oct'|label == 'Nov'] <- 'Autumn')

# I make a summary of the most important data, this is used for plotting
summary_climate <- climate_data %>%
  group_by(Season) %>%
  summarise(Mean_temp = mean(avg_t),
            Std_temp = std(avg_t),
            Sum_p = sum(avg_p))

climate_data %>%
  summarise(mean_t = mean(avg_t),
            std_t = std(avg_t),
            sum_p = sum(avg_p))

climate_data[order(climate_data$avg_t),]

# Below is the hydrotherm for the area
f = 4
ggplot(aes(x = label, y = avg_p), data = climate_data) + my_theme +
  geom_bar(stat = 'identity') +
  geom_line(aes(x = label, y = avg_t * f, group = 1)) +
  scale_y_continuous(name = "Monthly precipitation (mm)",
                     sec.axis = sec_axis(~ . / f, name = "Daily temperature (ºC)"))+
  theme(plot.background = element_rect(fill = 'white', colour = 'white'),
        axis.text=element_text(size=9, color="black"), axis.title=element_text(size=9, color="black"),
        axis.text.x = element_text(angle = 45, hjust=1), axis.title.x=element_blank())

ggsave('~/Documents/MacBook 2020/Geografi
KU/Speciale/Data/R/Figures_for_thesis/hydrotherm.png', width = 9, height = 7, units = "cm", dpi = 300)

theme(plot.background = element_rect(fill = 'white', colour = 'white'), legend.position="right",
      axis.title.x=element_blank(), axis.text.x=element_blank(), axis.ticks.x=element_blank(),
legend.title=element_blank(),
      axis.text=element_text(size=8, color="black"), axis.title=element_text(size=8, color="black"))

```