# Goose_test_notebook

March 19, 2020

## 1 Goose Test Page

This is the test page that is supposed to analyse the goose model data. First let us define the location of the data to analyse.

[2]:
```
data_dir = "~/CLionProjects/GooseTests/run-directory/"
source_dir = "~/CLionProjects/ALMaSS_all"
```

[3]:
```
import pandas as pd
import datetime as dt
import numpy as np
import time
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.dates as mdates

# this is definition whether we take into account timed values or not
# Is there a reason to use not timed values
is_timed = True

species_names = ["barnacle", "greylag", "pinkfoot"]
geese_foods = ['grain', 'grass', 'maize']
if is_timed==True:
    is_timed_str = '_timed'
else:
    is_timed_str = ''


# Let us define a mask that allows for all the fields to pass the filter if␣
 ↪asterisk is used
# Here we assume that our data does not have asterisks, otherwise another␣
 ↪symbol should be used
def ac_mask(df, key, value):
      if  value == '*':
          return df
      else:
          return df[df[key] == value]
```

```
# The same mask as previously, but allows to merge the values together, uses pd.Series
# also faster than the previous one
def ac_mask_mult(df, key, value):
      if  value == '*':
          return df
      else:
          if not(isinstance(value, list)):
              value = [value]
          return df[pd.Series(df[key]).isin(value)]

# We will use this mask instead of a standard one
pd.DataFrame.mask = ac_mask_mult
```

[ ]:
```
simulation_start_date = dt.date(2009, 1, 1)# we should check again that this is a right date, probably should be read from somewhere
simulation_start_date_ordinal=dt.date.toordinal(simulation_start_date)

# Forage data first: load data , while stripping the spaces
#forage_data=pd.read_csv(data_dir+"GooseFieldForageData.txt", sep='\t', header=0, parse_dates=['day'], dtype={'day':'str'},date_parser=my_dateparser)
forage_data=pd.read_csv(data_dir+"GooseFieldForageData.txt", sep='\t', header=0, dtype={'day': np.int16}, converters={'last_sown_veg': str.strip, 'veg_type_chr': str.strip, 'previous_crop': str.strip})
# The field dayordinal has the current day counting from 1/1/0001
forage_data['dayordinal']=forage_data['day']+simulation_start_date_ordinal
# Useful function that parses the data
my_dateparser=(lambda x: pd.to_datetime(x,unit='D', origin=simulation_start_date))
# The field 'daydate includes the date of the day for the data'
forage_data['daydate']=my_dateparser(forage_data['day'])
forage_data['habitat'] = 'Unknown'
```

The next cell defines the dictionary that ”translates” from field type into predefined forage habitats

[ ]:
```
# A dictionary that will allow to map vegetation to habitat
# The value is a list of tuples of last_sown_veg, veg_phase, veg_type_chr, previous_crop
# asterisk means don't care
# The key is habitat (In current R file there different columns for each
# species, but the resulting values are the same, so why?--> less code, less values is better)
veg_to_habitat_filt_keys=('last_sown_veg', 'veg_phase', 'veg_type_chr', 'previous_crop')

veg_to_habitat = {
```

```
    # Grasses: check that none is missing
    'Grass': [('PermanentGrassGrazed', [3,2,0], '*', '*'),
    ('PermanentGrassTussocky', [0,2], '*', '*'),
    ('CloverGrassGrazed1', [2,3], '*', '*'),
    ('CloverGrassGrazed2', 2, '*', '*'),
    ('OWinterWheatUndersown', 2, '*', '*'),
    ('OSeedGrass1', [3,2,0], '*', '*'),
    ('OSeedGrass2', [3,2,0], '*', '*'),
    ('SeedGrass2', [3,2,0], '*', '*'),
    ('SeedGrass1', [3,2,0], '*', '*'),
    ('OCloverGrassGrazed2', [0,2], '*', '*'),
    ('OCloverGrassGrazed1', [0,2,3], '*', '*'),
    ('CloverGrassGrazed1', 0, '*', '*'),
    ('CloverGrassGrazed2', 0, '*', '*'),
    ('NaturalGrass', '*', '*', '*')],
    'Rape': [('WinterRape', [0,2,3], '*', '*')],
    'WinterCereal':[('SpringBarley', [0,1,2], '*', '*'),
                    ('WinterWheat', [0,1,2], '*', '*'),
                    ('OWinterWheat', [0,1,2], '*', '*'),
                    ('SprBarleyCloverGrass', [0,1,2], '*', '*'),
                    ('WinterRye', [0,1,2], '*','*'),
                    ('OBarleyPeaCloverGrass', [2,1,0], '*', '*'),
                    ('OWinterRye', [2,1,0], '*', '*'),
                    ('OSBarleySilage', [0,1,2], '*', '*'),
                    ('OCarrots', 1, '*', '*'),
                    ('OSpringBarley', [2,1,0], '*', '*'),
                    ('OTriticale', [0,1,2], '*', '*'),
                    ('WinterBarley', [0,1,2], '*', '*'),
                    ('Triticale', [0,1,2], '*', '*'),
                    ('SpringBarleySilage', [0,1,2], '*', '*'),
                    ('WinterRape', 1, '*', '*'),
                    ('OWinterWheatUndersown', [0,1], '*', '*'),
                    ('Undefined', 0, '*', '*'),
                    ('OOats', [0,1,2], '*', '*'),
                    ('Oats', [0,1,2], '*', '*'),
                    ('OFieldPeas', 0, '*', '*')],
    'Stubble': [(['WinterBarley', 'OBarleyPeaCloverGrass',␣
↪'SprBarleyCloverGrass',
                  'SpringBarleySilage', 'OSBarleySilage', 'OWinterRye',␣
↪'WinterRye',
                  'OSpringBarley', 'SpringBarley', 'WinterWheat',␣
↪'OWinterWheat', 'OOats',
                  'Oats', 'OTriticale', 'OWinterWheatUndersown', 'Triticale',␣
↪'OFieldPeas'], 3, '*', '*'),
                ('CloverGrassGrazed1','SeedGrass1'], '*', '*',␣
↪'SprBarleyCloverGrass'),
```

```
                ('CloverGrassGrazed1','SeedGrass1'], '*',␣
↪'SprBarleyCloverGrass', '*'),
                ('CloverGrassGrazed1', '*', '*', 'SpringBarley')
        ],
    'Maize': [(['MaizeSilage','OMaizeSilage'], [0,1,2,3], '*', '*')
              ],
    'Undefined':[('Undefined', [2,3], '*', '*')]
    }
```

We will use this dictionary to filter and put the result in the column called habitat (Is it ever used?)

[ ]:
```
# let us combine the fields for testing
t = time.time()
for habitat_i in veg_to_habitat.keys():
    forage_data_idxx = [forage_data.mask(veg_to_habitat_filt_keys[0],t[0])
                        .mask(veg_to_habitat_filt_keys[1],t[1]).
↪mask(veg_to_habitat_filt_keys[2],t[2])
                        .mask(veg_to_habitat_filt_keys[3],t[3]).index.values␣
↪for t in veg_to_habitat[habitat_i]]
    idxs = np.concatenate(forage_data_idxx).ravel().tolist()

    forage_data.iloc[idxs, forage_data.columns.get_loc("habitat")] = habitat_i
elapsed = time.time() - t
print('Calculating new foraging habitats--> Elapsed: %s' % (elapsed))
```

[ ]:
```
# now let us filter the table where the geese exist and the months are between␣
↪august and march
t = time.time()
forage_data_months_filtered =␣
↪forage_data[forage_data['geese'+is_timed_str]&((forage_data['daydate'].dt.
↪month>7) | (forage_data['daydate'].dt.month<4))]

species_habs_to_plot = {new_list: {new_list1:pd.DataFrame() for new_list1 in␣
↪geese_foods} for new_list in species_names}
for i in range(len(species_names)):
    for j in range(len(geese_foods)):
        # temporary view that holds the areas/times when species are bigger␣
↪than 0
        temp = forage_data_months_filtered[species_names[i]+is_timed_str]>0
        if geese_foods[j] == 'grass':
            species_habs_to_plot[species_names[i]][geese_foods[j]] = pd.
↪DataFrame({'Date': forage_data_months_filtered[temp]['daydate'],
                                                                        ␣
↪   'FlockSize':␣
↪forage_data_months_filtered[temp][species_names[i]+is_timed_str],
                                                                        ␣
↪   ' kJ/min': forage_data_months_filtered[temp]['grass_'+species_names[i]]})
```

```
        else:
            if geese_foods[j] == 'grain':
                title1 = ' gr/m^2'
            else:
                title1 = ' kJ/m^2'

            species_habs_to_plot[species_names[i]][geese_foods[j]] = pd.
 ↪DataFrame({'Date': forage_data_months_filtered[temp]['daydate'],
                                                                          ␣
 ↪    'FlockSize':␣
 ↪forage_data_months_filtered[temp][species_names[i]+is_timed_str],
                                                                          ␣
 ↪    title1: forage_data_months_filtered[temp][geese_foods[j]]})
elapsed = time.time() - t
print('Filtering foraging data--> Elapsed: %s' % (elapsed))
```

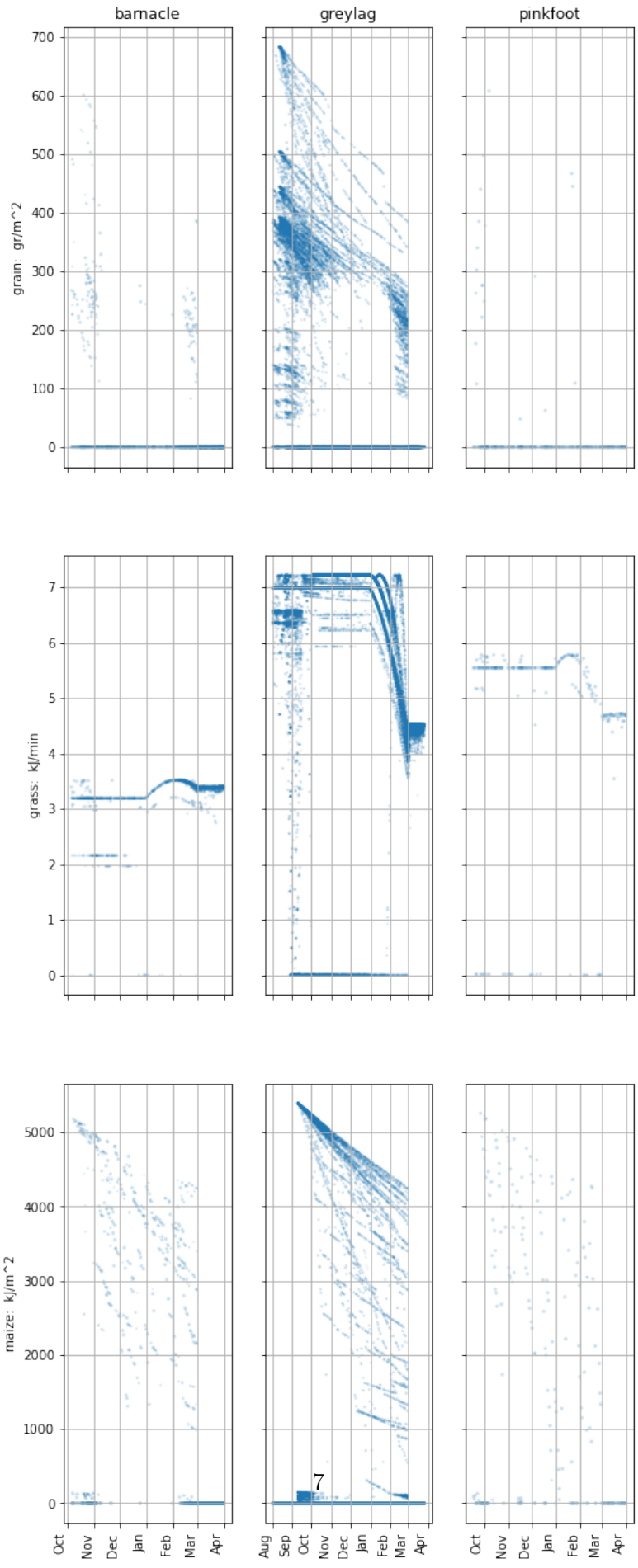
and plot the results

[9]:
```
fig, ax = plt.subplots(3, 3, sharex='col', sharey='row', figsize=mpl.figure.
 ↪figaspect(3.)*2)
# ax=plt.gca()
months = mdates.MonthLocator()
myFmt = mdates.DateFormatter('%b')
# plt.sca()
fig.autofmt_xdate(rotation='vertical')

for i in range(3):
    for j in range(3):
        ax[i,j].xaxis.set_major_formatter(myFmt)
        # ax[2,0].subplot(3, 3, 0)
        ax[i,j].grid()

        # for axx in fig.axes:
        #     plt.sca(axx)
        #     plt.xticks(rotation='vertical')
        ax[i,j].xaxis.set_minor_locator(months)
        ax[i,j].xaxis_date()
        # ax[2,0].xticks(rotation='vertical')
        # ax[2,0].ylabel(species_habs_to_plot['barnacle']['grass'].columns.
 ↪values[2])
        ax[i,j].
 ↪scatter(x=species_habs_to_plot[species_names[j]][geese_foods[i]].iloc[:,0],
                    y=species_habs_to_plot[species_names[j]][geese_foods[i]].
 ↪iloc[:,2], alpha=0.15,
                    s=np.
 ↪log10(species_habs_to_plot[species_names[j]][geese_foods[i]].iloc[:,1]))
        if j == 0:
```

```
            ax[i,j].set_ylabel(geese_foods[i]+': '
↪'+species_habs_to_plot[species_names[j]][geese_foods[i]].columns.values[2])
        if i == 0:
            ax[i,j].set_title(species_names[j])
```


barnacle
greylag
pinkfoot
grain: gr/m^2
grass: kJ/min
maize: kJ/m^2
Oct
Nov
Dec
Jan
Feb
Mar
Apr
Aug
Sep

Note that the scale is completely different because for each grain we compare different values.
Available amount of maize