Constraining multi-variate controls on Diatom growth and physiology

In these experiments, four environmental variables have been manipulated: **Temperature**, **CO2**, **Light** and **Fe**. These are our *independent variables*. Each *independent variable* has two states, yielding a total of $2^4 = 16$ experimental conditions.

In each experimental condition, six *dependent variables* have been measured: **Cell Size**, **Chlorophyll A**, **C/N ratio**, **FvFm**, **Sigma** (Chlorophyll antenna size) and **Growth Rate**.

We seek to determine whether there are relationships between our *independent* and *dependent* variables.

The small number of replicates in the study render the use of common frequentist statistical approaches (e.g. ANOVA) innappropriate. Instead, we adopt a 'Model Selection' approach similar to that of Boyd et al (2016) to identify patterns in the data.

Boyd et al (2016) explore the relative importance of independent variables in their data by comparing five arbitrary models which include or exclude independent variables. The relative skill of these models to predict patterns in their data is used as a measure of the importance of each independent variable in their analysis.

We have modified this approach such that instead of evaluating five arbitrary models, we compare the skill of all possible linear models to explain the data. We calculate the relative importance of parameters based on *all* this model ensemble, weighted by the skill of the model. With four independent variables, this yields 112 candidate models, which encompass all permutations of linear and interactive terms.

Model Selection

In the most general case, a multivariate relationship may be approximated by a first order polynomial:

$$f(x, y) = a_{11}xy + a_{10}x + a_{01}y + a_{00}$$

Where x and y are independent variables, a is a model parameter, and each additive term in the equation is termed a *covariate*. Given this form, we evalutae the *skill* of all possible combinations and permutations of model covariates against each of our dependent variables.

A model with high *skill* is one which explains the most variance in the dependent variable with the fewest covariates - i.e. minimising over-fitting.

Metrics such as R^2 and χ^2 to gauge the *goodness-of-fit* of a model, but neglect the number of cofactors included in the model so do not allow us to discern model is *skill*.

Information Criteria, such as the Aikake Information Criterion (**AIC**) or Bayesian Information Criterion (**BIC**) take the goodness-of-fit, the number of covariates, and sample size (BIC only) into consideration to give a measure of the *quality* of a model. These information criteria are

useful, but are ultimately approximations of more difficult-to-calculate metrics from information theory ('information loss' and 'Bayes factor' in the cases of AIC and BIC, respectively).

In our evaluation, we employ a method to directly calculate the Bayes Factor of a model compared to a null-model ($y = c$) based on the model R^2, the number of covariates (= degress of freedom), and sample size (Rouder & Morey, 2013).

The Bayes Factor (Kass & Raftery, 1995) is defined as:

$$B_{01} = \frac{p(\mathbf{D}|H_1)}{p(\mathbf{D}|H_0)}$$

That is, the probablity (p) of the data ($\mathbf{D}$) given your hypothesis (H_1) relative to the probability of the data given a null hypothesis. In other words, if $B_{01} = 5$, the observed data are 5 times more probable if H_1 is correct, rather than H_0.

A convenient feature of Bayes Factors is that the relative probablity of the data given any two models may be calculated from the Bayes Factors comparing each of them to the same null model, i.e.:

$$B_{21} = \frac{B_{20}}{B_{10}}$$

Because the method of Rouder and Morey (2013) calculates the bayes factor of your model compared to the same null model ($y = c$), this means that the relative probabilities of all models may be calculated.

Following this, we calculate the probability of all models relative to the 'best' model (that with highest *skill*). We then use these Bayes factors as weights when calculating an ensemble 'best fit' prediction, and in assessing the relative importance of independent variables.

Normalisation

Before analysis, we normalise all data such that it has a mean of zero and a standard deviation of one. This allows interpretation of the relative importance of best-fit model parameters.

In [2]:
```
import brutefit as bf  # this is a module I wrote that's available here: http
from tqdm import tqdm_notebook as tqdm
import pandas as pd
import numpy as np
idx = pd.IndexSlice

import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
dat = pd.read_excel('multi_species_model.xlsx', header=[0,1], index_col=0, er
```

In [4]:
```
brutes = {}
i = 0
for species in tqdm(['P. antarctica', 'C. flexuosus']):
    X = dat.loc[species, idx['controlled', ['temperature','CO2','light','Fe']

    brutes[species] = {}
```

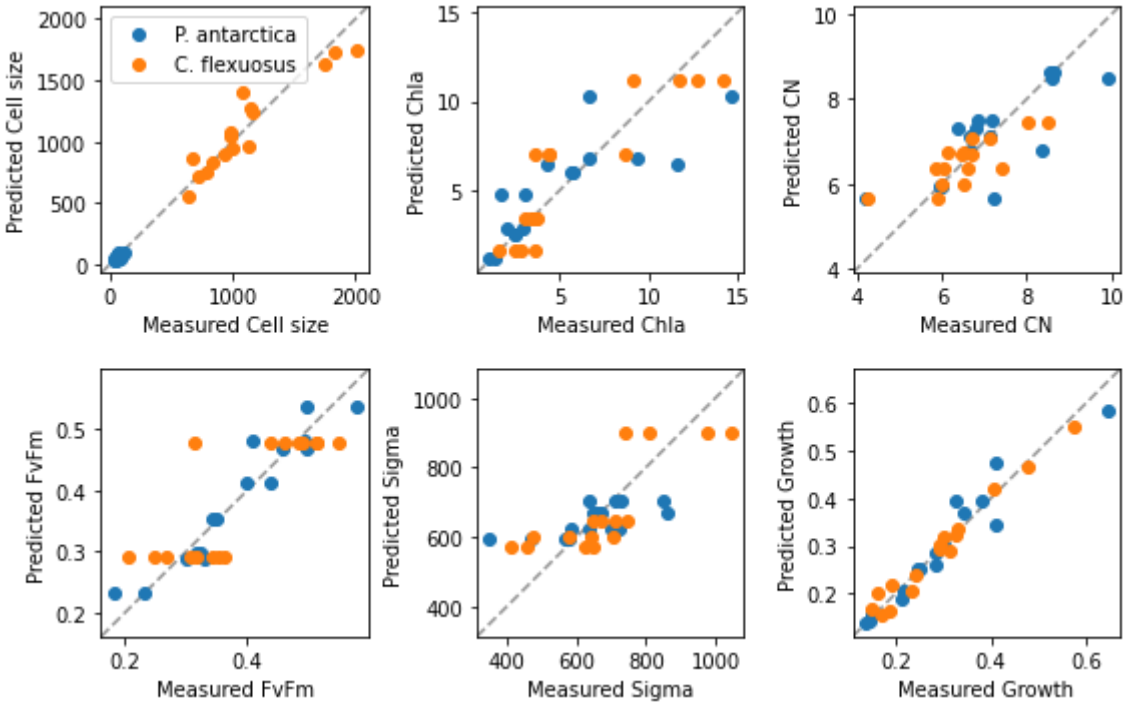
```
    for yvar in tqdm(['Cell size', 'Chla', 'CN', 'FvFm', 'Sigma', 'Growth'],
        y = dat.loc[species, (yvar, 'mean')].values.reshape(-1, 1)
        w = 1 / dat.loc[species, (yvar, 'std')]**2

        brute = bf.Brute(X, y, w, max_interaction_order=1, include_bias=False
        brute.evaluate_polynomials()
        brute.predict()

        brutes[species][yvar] = brute
```

'Best' Model

Below we show the measured data compared to the prediction of the 'best' (highest Bayes factor) model. This demonstrates that a linear model is able to explain patterns in the data.

In [5]:

```
fig, axs = plt.subplots(2, 3, figsize=[8, 5.2])

axf = axs.flatten()

xvars = ['temperature','CO2','light','Fe']

for species, mods in brutes.items():

    i = 0
    for yvar, brute in mods.items():

        axf[i].scatter(brute.y_orig, brute.pred_all[0],
                       label=species)
        axf[i].set_xlabel('Measured {}'.format(yvar))
        axf[i].set_ylabel('Predicted {}'.format(yvar))
        axf[i].set_aspect(1)
        i += 1

for ax in axf:
    lim = (min(ax.get_xlim()[0], ax.get_ylim()[0]),
           max(ax.get_xlim()[1], ax.get_ylim()[1]))
    ax.set_xlim(lim)
    ax.set_ylim(lim)
    ax.plot(lim, lim, zorder=-1, c=(0,0,0,0.4), ls='dashed')

axf[0].legend()

fig.tight_layout()

fig.savefig('meas_vs_model.pdf')
```



However, as there is no *a-priori* reason to assume that patterns in the data follow a particular functional form, it does not follow to base inferences solely on this single 'best' model. Given the uncertainty in functional form linking independent and dependent variables, a more realistic picture of the relationships in our dataset can be gained from including considering *all* possible linear models.

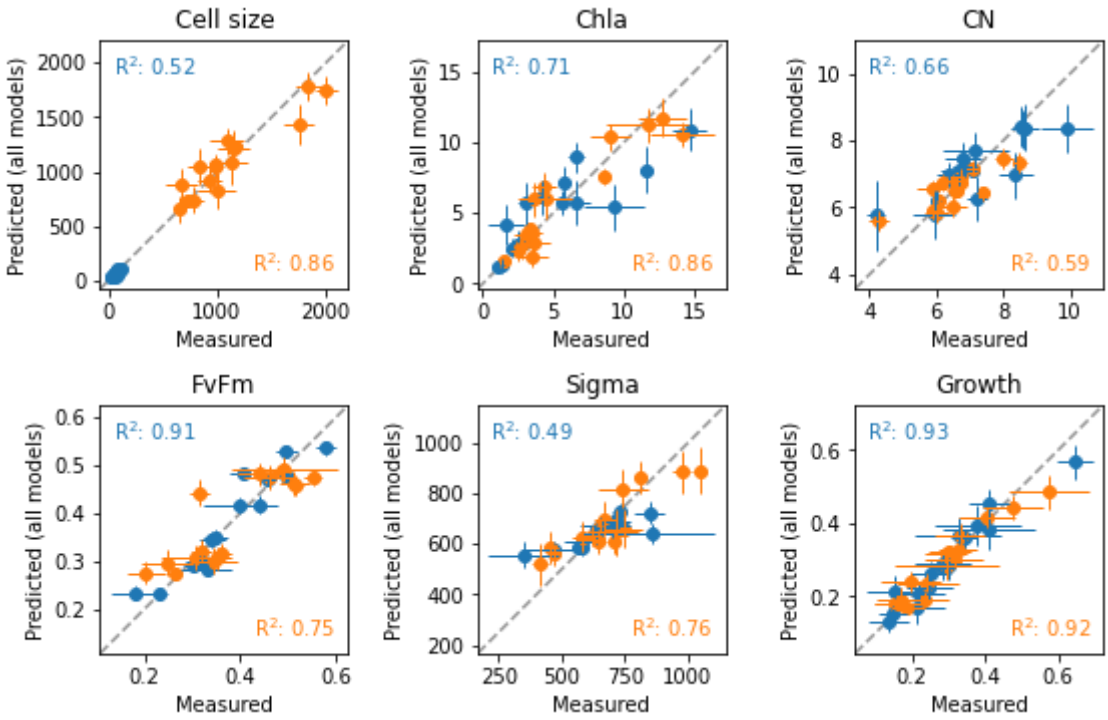
Ensemble Prediction from *all* models.

To accomplish this, we calculated the predicted values of *all* possible models, and calculate our 'best fit' values from the weighted average of all models using the Bayes Factors as weights. This identifies relationships within the data *without* prescribing a functional form to the relationship.

In [6]:

```
fig, axs = plt.subplots(2, 3, figsize=[8, 5.2])

xvars = ['temperature','CO2','light','Fe']
allfits = {}
means = {}
stds = {}

for c, species in zip(['C0', 'C1'], ['P. antarctica', 'C. flexuosus']):
    for ax, yvar in zip(axs.flat, ['Cell size', 'Chla', 'CN', 'FvFm', 'Sigma'
        brute = brutes[species][yvar]
        brute.plot_obs_vs_pred(ax=ax)

        r2 = bf.stats.calc_R2(brute.y_orig[:,0], brute.pred_means)

        if c == 'C0':
            ax.text(0.05, 0.95, f'$R^2$: {r2:.2f}', color=c, ha='left', va='t
        else:
            ax.text(0.95, 0.05, f'$R^2$: {r2:.2f}', color=c, ha='right', va='

        ax.set_title(yvar)

for ax in axs.flat:
    lim = (min(ax.get_xlim()[0], ax.get_ylim()[0]),
           max(ax.get_xlim()[1], ax.get_ylim()[1]))
```

```
    ax.set_xlim(lim)
    ax.set_ylim(lim)
    ax.plot(lim, lim, zorder=-1, c=(0,0,0,0.4), ls='dashed')

fig.tight_layout()

fig.savefig('meas_vs_model_ensemble.pdf')
```



In [8]:

```
obs_v_pred = pd.DataFrame(columns=pd.MultiIndex.from_product([[''],['Controll

for species in ['P. antarctica', 'C. flexuosus']:
    for variable in ['Cell size', 'Chla', 'CN', 'FvFm', 'Sigma', 'Growth']:
        brute = brutes[species][variable]
        obs_v_pred.loc[:, (species, 'Measured', variable)] = brute.y_orig
        obs_v_pred.loc[:, (species, 'Measured', variable + '_std')] = (1 / br

        obs_v_pred.loc[:, (species, 'Predicted', variable)] = brute.pred_mear
        obs_v_pred.loc[:, (species, 'Predicted', variable + '_std')] = brute.

obs_v_pred.sort_index(1, inplace=True)

obs_v_pred.to_csv('export_ObsVPred.csv')
```

Effect Sizes based on *all* Models

Extending this weighted-ensemble approach, we may estimate the relative importance of each model parameter using a kernel-density estimator where the contribution of each each model to the overall distribution is weighted by its Bayes Factor. This provides an estimate of the relative importance of each covariate in the model.

Interpreting distributions - what is 'significant'?

In this analysis we purposefully steer clear of frequentist 'p' values and 'significance' terminology, as the small sample sizes limit the applicability of standard statistical methods, and render the interpretation of 'p' values to be, at best, precarious. Rather, we appraise the relative importance of our dependent variables graphically, in the plot below.

In this plot, each distribution represents the importance of a model covariate derived from our ensemble of 112 models, weighted by their Bayes Factors. Parameter values from models with low Bayes Factors, and hence low 'skill' are down-weighted in these distributions.

If a distribution is sharp, the value of this covariate was relatively consistent across all high-skill models - i.e. its value is relatively constant, despite the inclusion or exclusion of other covariates in the model. If the distribution is broader, importance of this covariate in high-skill varies with the addition or removal of other covariates.

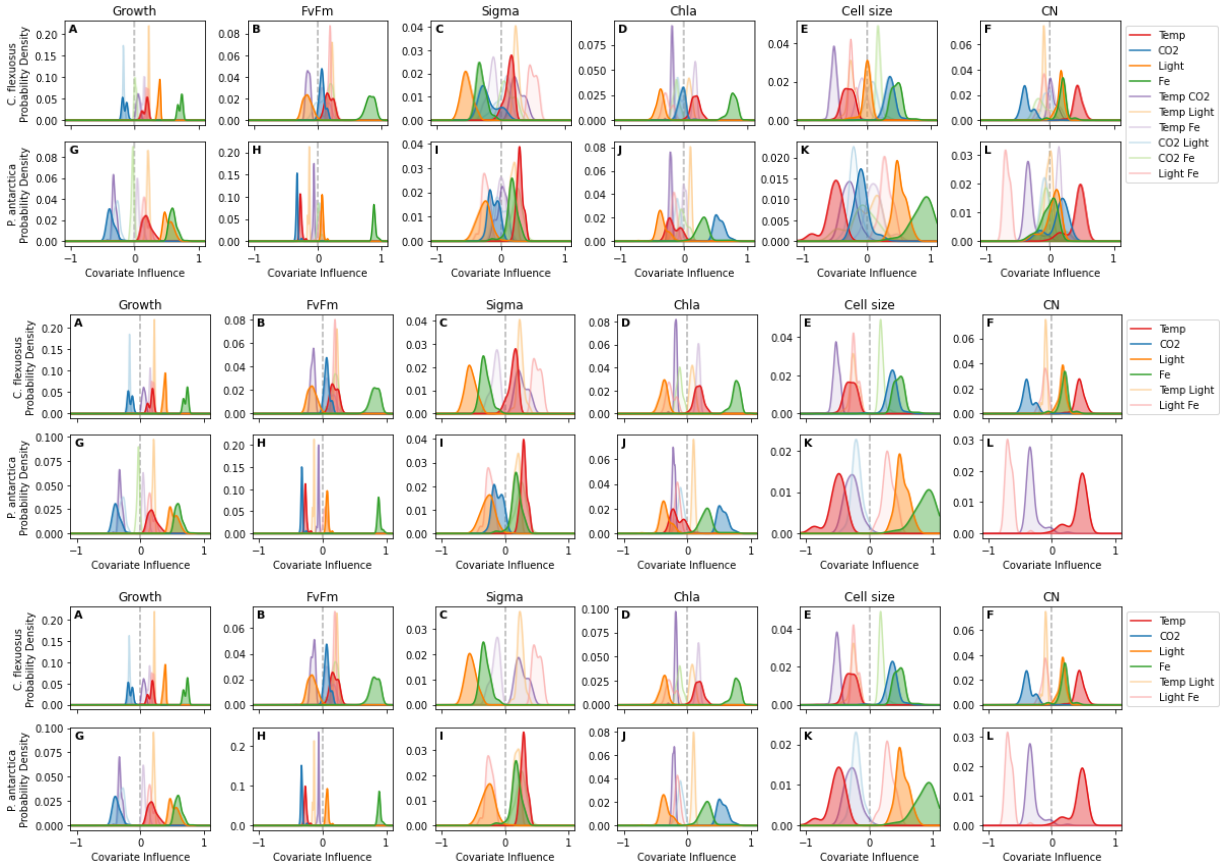
The further the centre of the distribution lies from the zero line, the more stronger the relationship between that covariate and the dependent variable. Values above zero indicating a positive relationship and vice versa. If the distribution overlaps with zero, this may be interpreted as having no (or little) effect on the dependent variable - or being 'non significant' in frequentist terminology.

In summary, these plots give an estimate of the relative, individual influence of each model covariate on each dependent variable. Distributions further from the zero line indicate a stronger effect, and we can be more confident in them. I a distribution does not overlap with zero, it can be confidently described as a 'real' influence on the dependent variable. Distributions closer to zero are weaker, and if they overlap with the zero line they cannot be conclusively described as having an influence on the dependent variable.

In [9]:
```
c = {
    'Temp': '#e31a1c',
    'Light': '#ff7f00',
    'CO2': '#1f78b4',
    'Fe': '#33a02c',
    'Temp CO2': '#6a3d9a',
    'Temp Light': '#fdbf6f',
    'Temp Fe': '#cab2d6',
    'CO2 Light': '#a6cee3',
    'CO2 Fe': '#b2df8a',
    'Light Fe': '#fb9a99'
}

labels = 'ABCDEFGHIJKL'

for s, sub in brutes.items():
    for v, brute in sub.items():
        brute.set_covariate_colors(c)
```

In [10]:
```
for zero_overlap in [1, 0.1, 0.05]:

    fig, axs = plt.subplots(2, 6, sharex=True, sharey=False, figsize=[17, 4])

    mn = -1
    mx = 1
    rn = mx - mn
    pad = 0.05
    vals = np.linspace(mn - rn * pad, mx + rn * pad, 500)

    bin_width = 0.5
    i = 0

    ymax = []
    for row, species in zip(axs, ['C. flexuosus', 'P. antarctica']):
        for ax, variable in zip(row, ['Growth', 'FvFm', 'Sigma', 'Chla', 'Cel
```

```
            brute = brutes[species][variable]

            brute.plot_param_dists(vals, bw_method=bin_width, ax=ax, filter_z

            if ax.is_first_row():
                ax.set_title(variable)
                ax.set_xlabel(None)
            else:
                ax.set_xlabel('Covariate Influence')
            if ax.is_first_col():
                ax.set_ylabel('{}\nProbability Density'.format(species))
            else:
                ax.set_ylabel(None)

            ax.text(0.03, 0.97, labels[i], weight='bold', size=11, va='top',

            i += 1

            ymax.append(ax.get_ylim()[1])

    for ax in axs.flat:
    #     ax.set_ylim(0, max(ymax))
        ax.set_xlim(vals[0], vals[-1])

    fig.tight_layout()

    fig.subplots_adjust(right=0.9)
    axs[0,-1].legend(bbox_to_anchor=(1, 1))

    fig.savefig(f'parameter_contributions_{zero_overlap:.2f}.pdf')
```



In [11]:

```
plot_pdfs = []
for species in ['P. antarctica', 'C. flexuosus']:
    for variable in ['Cell size', 'Chla', 'CN', 'FvFm', 'Sigma', 'Growth']:
        brute = brutes[species][variable]
        pdfs = bf.stats.calc_param_distributions(brute, bw_method=bin_width,
```

```
        pdfs.columns = pd.MultiIndex.from_tuples([(species, variable, brute.v
        plot_pdfs.append(pdfs)
plot_pdfs = pd.concat(plot_pdfs, 1)

plot_pdfs.to_csv('export_pdfs.csv')
```

Modern vs. Future Ocean

In [12]:

```
def condition_plot(sub, brute, idx_future, idx_now, ind, ax):
    ax.errorbar(idx_future[ind], sub.loc[ind, (yvar, 'mean')], sub.loc[ind, (
                lw=0, elinewidth=1, marker='o', color=c)
    ax.plot(idx_future[ind], brute.pred_means[ind], color=c)
    ax.fill_between(idx_future[ind],
                    brute.pred_means[ind] + brute.pred_stds[ind],
                    brute.pred_means[ind] - brute.pred_stds[ind],
                    alpha=0.2, color=c)

yvars = ['Growth', 'FvFm', 'Sigma', 'Chla', 'Cell size', 'CN']
labels = 'ABCDEFGHIJKL'
colors = ['darkgrey', 'C1', 'C0', 'C2']

# panel size
pw = 2.5
ph = 2
fig, axs = plt.subplots(dat.index.unique().size, len(yvars), figsize=[len(yva

i = 0
for row, s in zip(axs, dat.index.unique()):
    sub = dat.loc[s]
    idx_now = (sub.controlled.temperature == 3) & (sub.controlled.CO2 == 350)
    idx_future = (sub.controlled.temperature == 5) & (sub.controlled.CO2 == 1
    idx_relevant = idx_now | idx_future

    idx_Fe_lo = sub.controlled.Fe == 4
    idx_Fe_hi = ~idx_Fe_lo
    idx_light_lo = sub.controlled.light == 25
    idx_light_hi = ~idx_light_lo

    for ax, yvar in zip(row, yvars):
        brute = brutes[s][yvar]

        # low light, low iron
        c = colors[0]
        ind = idx_Fe_lo & idx_light_lo & idx_relevant
        condition_plot(sub, brute, idx_future, idx_now, ind, ax)

        # high light, low iron
        c = colors[1]
        ind = idx_Fe_lo & idx_light_hi & idx_relevant
        condition_plot(sub, brute, idx_future, idx_now, ind, ax)

        # low light, high iron
        c = colors[2]
        ind = idx_Fe_hi & idx_light_lo & idx_relevant
        condition_plot(sub, brute, idx_future, idx_now, ind, ax)

        # high light, high iron
        c = colors[3]
        ind = idx_Fe_hi & idx_light_hi & idx_relevant
        condition_plot(sub, brute, idx_future, idx_now, ind, ax)

        if ax.is_first_row():
```

```
            ax.set_title(yvar)
        else:
            ax.set_xticks([0, 1])
            ax.set_xticklabels(['Now', 'Future'])

        if ax.is_first_col():
            ax.set_ylabel(s)

        ax.text(0.03, 0.97, labels[i], weight='bold', size=11, va='top', ha='

        i += 1

fig.tight_layout()

# legend
right = .85
fig.subplots_adjust(right=right)
upper = axs[0, -1].get_position()
lower = axs[1, -1].get_position()
legax = fig.add_axes([right + 0.01, lower.y0, 1 - right - 0.01 - 0.05, upper.

conditions = ['- Light, - Fe', '+ Light, - Fe', '- Light, + Fe', '+ Light, +
for var, c in zip(conditions, colors):
    legax.errorbar([],[],[],lw=0, elinewidth=1, marker='o', color=c, label=va
legax.plot([],[],c=(.5,.5,.5), label='Model')
legax.fill_between([],[],[],color=(.5,.5,.5), alpha=0.2, label='Model $1\sigm

legax.legend(loc='upper left')
legax.axis('off')

fig.savefig('Now_vs_Future.pdf')
```