Linear regressions on spiracle data, non-parametric and Bayesian

To begin with, we need to import necessary python packages.

In [1]:
```
import numpy as np
import pandas as pd

import statsmodels.api
import scipy.stats

import bokeh.io
import bokeh.plotting

from Bio import Phylo
import io

import cmdstanpy
import arviz as az
import logging

import bebi103

#local .py file for some plotting functions and non-parametric bootstrapping utils
import plotting_utils

bokeh.io.output_notebook()
```

```
BokehJS 1.4.0 successfully loaded.
```

We can now read in the data into a dataframe for analyis.

In [2]:
```
df = pd.read_csv("./20190322_supp_table_2.csv")
```

We take a look at the format for the data.

In [3]:
```
df['species_underscore'] = [spec.replace(" ", "_") for  spec in df['species']]
df.head()
```

Out[3]:

	Unnamed: 0	subfamily	species	sex	mass (g)	spiracle	area (mm^2)	depth (mm)	species_underscore
0	0	Cetoniinae	Goliathus goliathus	M	16.280	6	0.274408	2.512648	Goliathus_goliathus
1	1	Cetoniinae	Goliathus goliathus	F	18.150	6	0.134949	1.606189	Goliathus_goliathus
2	2	Cetoniinae	Coelorrhina hornimani	M	1.130	6	0.212131	0.553833	Coelorrhina_hornimani
3	3	Cetoniinae	Dicronorrhina derbyana	M	2.120	6	0.039532	0.473369	Dicronorrhina_derbyana
4	4	Cetoniinae	Dicronorrhina derbyana	F	2.145	6	0.049701	0.496320	Dicronorrhina_derbyana

For some of this analysis, we will look at the per-species averages for our measurements. To get this, we use a

simple aggregate function on the dataframe and take a look at the results.

In [4]:
```
df_averages = df.groupby(['species', 'species_underscore', 'spiracle'], as_index=False).ag
df_averages['subfamily'] = df.groupby(['species', 'species_underscore', 'spiracle'], as_in
df_averages.head()
```

Out[4]:

	species	species_underscore	spiracle	Unnamed: 0	area (mm^2)	depth (mm)	mass (g)	subfamily
0	Coelorrhina hornimani	Coelorrhina_hornimani	1	87.0	0.135347	0.416717	1.13	Cetoniinae
1	Coelorrhina hornimani	Coelorrhina_hornimani	2	70.0	0.084207	0.451409	1.13	Cetoniinae
2	Coelorrhina hornimani	Coelorrhina_hornimani	3	53.0	0.106693	0.325444	1.13	Cetoniinae
3	Coelorrhina hornimani	Coelorrhina_hornimani	4	36.0	0.115574	0.481558	1.13	Cetoniinae
4	Coelorrhina hornimani	Coelorrhina_hornimani	5	19.0	0.119145	0.506751	1.13	Cetoniinae

Let's take a look at the number of species per subfamily in the data.

In [5]:
```
species_per_subfam=df_averages.groupby(['subfamily', 'spiracle'], as_index=False).count()
species_per_subfam.columns = ('subfamily', 'subfam_count')
species_per_subfam
```

Out[5]:

	subfamily	subfam_count
0	Cetoniinae	6
1	Dynastinae	3
2	Rutelinae	1

In [6]:
```
df_averages = df_averages.merge(species_per_subfam, on='subfamily')
```

For our plots, we will log transform the data. We will add a column to the dataframe with the log transformed data. We will also need some transforms of our data, which we will do here.

In [7]:
```
df_averages['log area (mm^2)'] = np.log10(df_averages['area (mm^2)'])
df_averages['log dist'] = np.log10(df_averages['depth (mm)'])
df_averages['log mass (g)'] = np.log10(df_averages['mass (g)'])
df_averages['log area/dist'] = np.log10(df_averages['area (mm^2)']/df_averages['depth (mm)
df_averages['log area^2/dist'] = np.log10(df_averages['area (mm^2)']**2/df_averages['depth

df_averages.head()
```

Out[7]:

	species	species_underscore	spiracle	Unnamed: 0	area (mm^2)	depth (mm)	mass (g)	subfamily	subfam_count	log ((mm
0	Coelorrhina hornimani	Coelorrhina_hornimani	1	87.0	0.135347	0.416717	1.13	Cetoniinae	6	-0.868
1	Coelorrhina hornimani	Coelorrhina_hornimani	2	70.0	0.084207	0.451409	1.13	Cetoniinae	6	-1.074

	species	species_underscore	spiracle	Unnamed: 0	area (mm^2)	depth (mm)	mass (g)	subfamily	subfam_count	log
2	Coelorrhina hornimani	Coelorrhina_hornimani	3	53.0	0.106693	0.325444	1.13	Cetoniinae	6	-0.97
3	Coelorrhina hornimani	Coelorrhina_hornimani	4	36.0	0.115574	0.481558	1.13	Cetoniinae	6	-0.937
4	Coelorrhina hornimani	Coelorrhina_hornimani	5	19.0	0.119145	0.506751	1.13	Cetoniinae	6	-0.923

In addition to log transforming the species averaged data, we will do the same for the whole data set.

In [8]:
```
df['log area (mm^2)'] = np.log10(df['area (mm^2)'])
df['log dist'] = np.log10(df['depth (mm)'])
df['log mass (g)'] = np.log10(df['mass (g)'])
df['log area/dist'] = np.log10(df['area (mm^2)']/df['depth (mm)'])
df['log area^2/dist'] = np.log10(df['area (mm^2)']**2/df['depth (mm)'])
df.head()
```

Out[8]:

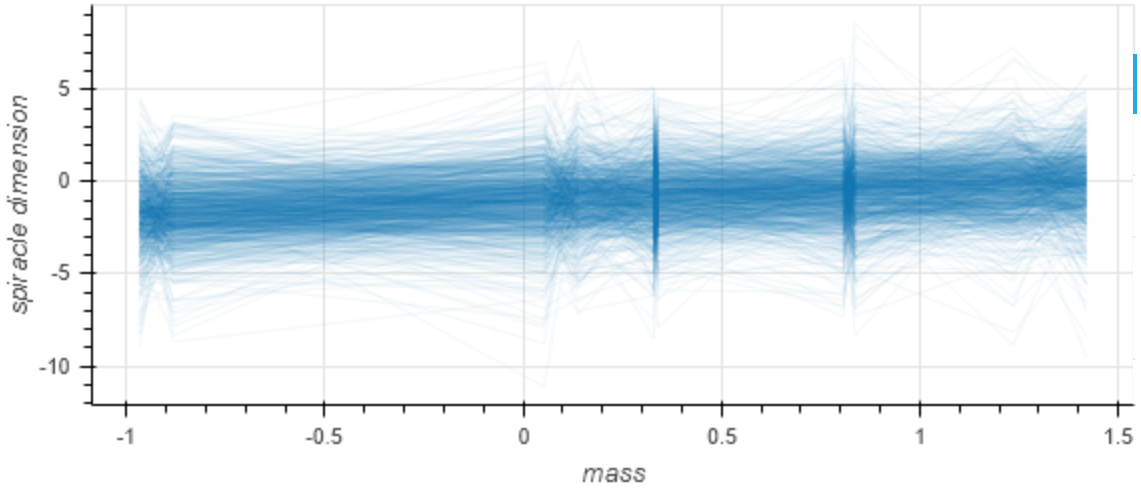
	Unnamed: 0	subfamily	species	sex	mass (g)	spiracle	area (mm^2)	depth (mm)	species_underscore	log area (mm^2)
0	0	Cetoniinae	Goliathus goliathus	M	16.280	6	0.274408	2.512648	Goliathus_goliathus	-0.561603
1	1	Cetoniinae	Goliathus goliathus	F	18.150	6	0.134949	1.606189	Goliathus_goliathus	-0.869831
2	2	Cetoniinae	Coelorrhina hornimani	M	1.130	6	0.212131	0.553833	Coelorrhina_hornimani	-0.673395
3	3	Cetoniinae	Dicronorrhina derbyana	M	2.120	6	0.039532	0.473369	Dicronorrhina_derbyana	-1.403054
4	4	Cetoniinae	Dicronorrhina derbyana	F	2.145	6	0.049701	0.496320	Dicronorrhina_derbyana	-1.303635

With our data in this format, we can build simple model for our regression. To do this, we will write a statistical model for the data. We choose a normal distribution with mean given by a simple linear function (with slope a and intercept b), and homoscedastic variance σ. For our priors, we will take a normal distribution for the slope with mean given by the slope value representing isometric scaling in the log-log space, and standard deviation of 0.3. This encodes our prior expectation that isometry is the most common/likely outcome for morphological features. If they scale uniformly with one another, we would see isometry. The variance term on this prior gives a broad but reasonable range of potential values for the parameter around the isometric value. This prior is hence weakly informative, fairly encoding our prior expectation without constraining the ability of our data to inform the parameter value. For the intercept, we expect a small value for spiracle dimensions of a 1g insect (what the intercept of the log mass x axis represents), so we put this mean at around 1/10th of a square mm for area (mean of -1 in log space), with a standard deviation of 1. Hence ~95% of the probability mass for this prior goes from ~100 times smaller to ~100 times larger than this mean. Once again, this is a pretty broad distribution so shouldn't drag the parameter estimates much, but does encode our prior intuition of what we expect from beetle spiracle dimensions. For the prior on the σ term for the normal distribution, we take a half normal (σ must be positive), with mean at zero and standard deviation of 1, once again this allows for a wide range of values for sigma while also encoding our expectation that morphological measures shouldn't have extreme variation on a log scale. This gives the following model:

$$f_\mu(x_i; a, b) = ax_i + b \tag{1}$$
$$b \sim \text{Norm}(-1, 1) \tag{2}$$
$$a \sim \text{Norm}(\text{isometry}, 0.3) \tag{3}$$
$$[\text{isometry}_{\text{area}} = 2/3,\ \text{isometry}_{\text{depth}} = 1/3, \text{isometry}_{\text{area/depth}} = 1/3,\ \text{isometry}_{\text{area}^2/\text{depth}} = 1] \tag{4}$$
$$\sigma \sim \text{HalfNorm}(0, 1) \tag{5}$$
$$y_i \sim \text{Normal}\left(f_\mu(x_i; a, b), \sigma\right) \tag{6}$$

We will do a quick prior-predictive check of this model to make sure it gives reasonable potential datasets.

In [9]:
```
x = df_averages.sort_values('mass (g)').loc[df_averages['spiracle'] == '1', 'log mass (g)'
size = 1000
lines = []
for a, b, sigma in zip(np.random.normal(0.67,0.3, size=size), np.random.normal(-1,1,size=s
    lines.append(np.random.normal(a*x+b, sigma))

p = bokeh.plotting.figure(plot_height=250, x_axis_label='mass', y_axis_label='spiracle dir
[p.line(x, y, alpha=0.05) for y in lines]
bokeh.io.show(p)
```



This looks good! The proposed model gives a slight upward slope and a nice wide range of values for both the slope and intercept based on the priors. The data should be able to inform the parameter values well based on this model. We are now ready to sample to get parameter estimates! We do this below. In addition to sampling, we also produce some plots of the model outputs given the sampled parameters (i.e. the posterior samples as well as samples from the posterior predictive distribution).

In [36]:
```
logging.getLogger("cmdstanpy").setLevel(logging.WARNING)
all_summaries = {}
all_regressions = {}
N_ppc = 200
for morphs, iso in zip(['log area (mm^2)', 'log dist', 'log area/dist', 'log area^2/dist']

    sigmas = []
    slopes = []
    intercepts = []
    coef_vars = []
    summaries = {}
    plots = []
    for spir in ['S', 'T', '1', '2', '3', '4', '5', '6']:
        x = df_averages.sort_values('mass (g)').loc[df_averages['spiracle'] == spir, 'log
        y = df_averages.sort_values('mass (g)').loc[df_averages['spiracle'] == spir, morph

        x_ppc = np.linspace(x.min(), x.max(), N_ppc)
```

```
        data = {
            "N": len(x),
            "x": x,
            "y": y,
            "priora": iso,
            "x_ppc": x_ppc,
            'N_ppc': N_ppc,
        }

        sm = cmdstanpy.CmdStanModel(stan_file='spiracle_regression.stan')

        samples = sm.sample(data=data, sampling_iters=1000, chains=4)

        samples = az.from_cmdstanpy(posterior=samples, posterior_predictive=["y_ppc"])
        print("_____________________________________________________________________________
              "Checking diagnostics for samples for " + morphs + ' spiracle ' + spir + '\r
              "_____________________________________________________________________________
        bebi103.stan.check_all_diagnostics(samples, var_names=['b', 'a', 'sigma'])
        sigmas.append(samples.posterior['sigma'].values.ravel())
        coef_vars.append(samples.posterior['coef_var'].values.ravel())
        slopes.append(samples.posterior['a'].values.ravel())
        intercepts.append(samples.posterior['b'].values.ravel())
        summaries[spir] = az.summary(samples, var_names=['b', 'a', 'sigma', 'coef_var'])
        summaries[spir]['median'] = [np.median(samples.posterior['b'].values.ravel()),
                                     np.median(samples.posterior['a'].values.ravel()),
                                     np.median(samples.posterior['sigma'].values.ravel()),
                                     np.median(samples.posterior['coef_var'].values.ravel
        plots.append(plotting_utils.plot_regression_comparison(df, df_averages, x, spir, r

    all_regressions[morphs] = plots
    all_summaries[morphs] = summaries
logging.getLogger("cmdstanpy").setLevel(logging.INFO)
```

```
_____________________________________________________________________________
_________________________
Checking diagnostics for samples for log area (mm^2) spiracle S
_____________________________________________________________________________
_________________________

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

_____________________________________________________________________________
_________________________
Checking diagnostics for samples for log area (mm^2) spiracle T
_____________________________________________________________________________
_________________________

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

_____________________________________________________________________________
_________________________
Checking diagnostics for samples for log area (mm^2) spiracle 1
_____________________________________________________________________________
_________________________

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
```

0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area (mm^2) spiracle 2

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area (mm^2) spiracle 3

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area (mm^2) spiracle 4

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area (mm^2) spiracle 5

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area (mm^2) spiracle 6

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log dist spiracle S

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.

0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log dist spiracle T

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log dist spiracle 1

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log dist spiracle 2

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log dist spiracle 3

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log dist spiracle 4

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log dist spiracle 5

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.

0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log dist spiracle 6

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area/dist spiracle S

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area/dist spiracle T

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area/dist spiracle 1

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area/dist spiracle 2

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area/dist spiracle 3

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.

0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area/dist spiracle 4

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area/dist spiracle 5

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area/dist spiracle 6

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area^2/dist spiracle S

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area^2/dist spiracle T

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.

Checking diagnostics for samples for log area^2/dist spiracle 1

Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.

```
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.
```

```
Checking diagnostics for samples for log area^2/dist spiracle 2
```

```
Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.
```

```
Checking diagnostics for samples for log area^2/dist spiracle 3
```

```
Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.
```

```
Checking diagnostics for samples for log area^2/dist spiracle 4
```

```
Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.
```

```
Checking diagnostics for samples for log area^2/dist spiracle 5
```

```
Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.
```

```
Checking diagnostics for samples for log area^2/dist spiracle 6
```

```
Effective sample size looks reasonable for all parameters.
Rhat looks reasonable for all parameters.
0 of 4000 (0.0%) iterations ended with a divergence.
0 of 4000 (0.0%) iterations saturated the maximum tree depth of 10.
E-BFMI indicated no pathological behavior.
```

Nice! the sampling looks to have gone well, with no major issues based on diagnostics. With samples for all the parameters and for all the spiracles/morphological features of interest in hand, we can now generate a plot for some summary stats about the data. We will also compute similar estimates for regression features of interest using a non-parametric bootstrapping approach for comparison.

In [11]:

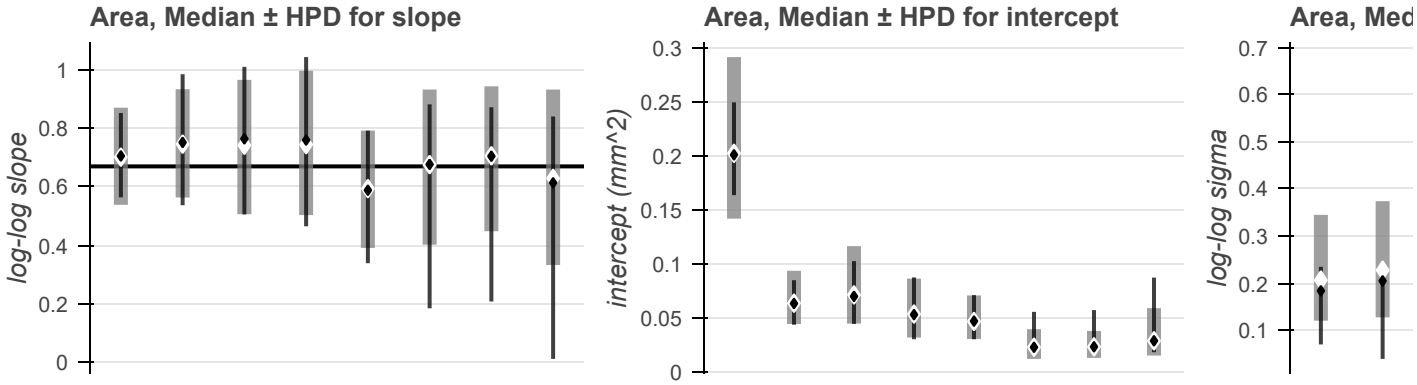
```
plots = []
```

```
for morphs, iso, heading in zip(['log area (mm^2)', 'log dist', 'log area/dist', 'log area
    #print(morphs)
    summaries = all_summaries[morphs]
    for var, yax, var_title in zip(['a', 'b', 'sigma', 'coef_var'], ['log-log slope', 'int
        p = bokeh.plotting.figure(plot_height=200, plot_width=300, x_range=['S', 'T', '1',
        #p = bokeh.plotting.figure(plot_height=200, plot_width=300, x_range=['S', 'T', '1
        p.xaxis.visible = False
        p.outline_line_color = None
        p.yaxis.minor_tick_line_color = None
        p.xaxis.minor_tick_line_color = None
        if var == 'a':
                p.line([-10, 100], [iso, iso], color='black', line_width=2, line_alpha=1)
        for i, spir in enumerate(['S', 'T', '1', '2', '3', '4', '5', '6']):
            CI_a, CI_b, CI_σ, CI_co_σ, slope, intercept, σ, co_σ = plotting_utils.make_CIs

            if var == 'b':
                c = 1
                if morphs == 'log area/dist':
                    c = 0.178*404*(1/10)
                elif morphs == 'log area^2/dist':
                    c = (1/(3.1415*8*1.86*(10**(-8))))*(0.1**3)
                p.line([spir, spir], c*(10**np.array([summaries[spir].loc[var]['hpd_3%'],
                p.line([spir, spir], c*(10**CI_b), color='black', line_width=2, alpha=0.75
                p.diamond([spir,],   c*(10**summaries[spir].loc[var]['median']), color='wh
                p.diamond([spir,],   c*(10**intercept), color='black', size=5)
            elif var == 'a':
                p.line([spir, spir], [summaries[spir].loc[var]['hpd_3%'], summaries[spir].
                p.line([spir, spir], CI_a, color='black', line_width=2, alpha=0.75)
                p.diamond([spir,], summaries[spir].loc[var]['median'], color='white', size
                p.diamond([spir,], slope, color='black', size=5)
            elif var == 'sigma':
                p.line([spir, spir], [summaries[spir].loc[var]['hpd_3%'], summaries[spir].
                p.line([spir, spir], CI_σ, color='black', line_width=2, alpha=0.75)
                p.diamond([spir,], summaries[spir].loc[var]['median'], color='white', size
                p.diamond([spir,], σ, color='black', size=5)

            elif var == 'coef_var':
                p.line([spir, spir], [summaries[spir].loc[var]['hpd_3%'], summaries[spir].
                p.line([spir, spir], CI_co_σ, color='black', line_width=2, alpha=0.75)
                p.diamond([spir,], summaries[spir].loc[var]['median'], color='white', size
                p.diamond([spir,], co_σ, color='black', size=5)

        p.xgrid.grid_line_color = None
        #p.ygrid.grid_line_color = None
        p.output_backend = 'svg'
        plots.append(p)
        #bokeh.io.show(p)

print("FOR ALL PLOTS: White diamond is median Bayesian parameter sample, grey box is 3%-97
print("FOR ALL PLOTS: Black diamond is median summary statistic for non-parametric bootstr
print('Area regressions (log-log slope, intercept (mm^2), log-log σ)')
bokeh.io.show(bokeh.layouts.gridplot(plots[0:4], ncols=4))
print('Depth regressions (log-log slope, intercept (mm), log-log σ)')
bokeh.io.show(bokeh.layouts.gridplot(plots[4:8], ncols=4))
print('Area/Depth regressions (log-log slope, intercept (G_diff), log-log σ)')
bokeh.io.show(bokeh.layouts.gridplot(plots[8:12], ncols=4))
print('Area^2/Depth regressions (log-log slope, intercept (G_adv), log-log σ)')
bokeh.io.show(bokeh.layouts.gridplot(plots[12:16], ncols=4))

#print('Area regressions (log-log slope, intercept (mm^2), log-log σ)')
#bokeh.io.show(bokeh.layouts.gridplot(plots[0:3], ncols=3))
#print('Depth regressions (log-log slope, intercept (mm), log-log σ)')
#bokeh.io.show(bokeh.layouts.gridplot(plots[3:6], ncols=3))
#print('Area/Depth regressions (log-log slope, intercept (G_diff), log-log σ)')
#bokeh.io.show(bokeh.layouts.gridplot(plots[6:9], ncols=3))
```

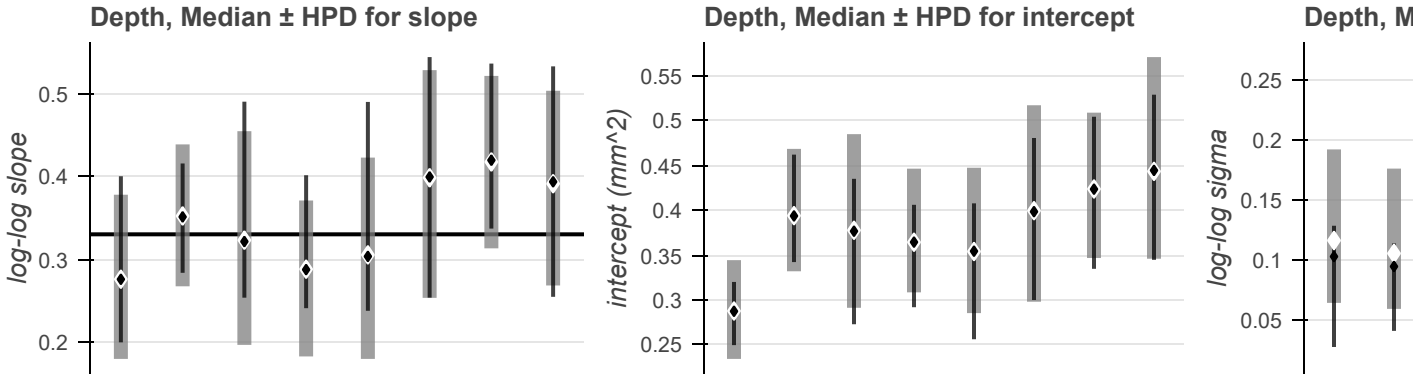
```
#print('Area^2/Depth regressions (log-log slope, intercept (G_adv), log-log σ)')
#bokeh.io.show(bokeh.layouts.gridplot(plots[9:12], ncols=3))
```

```
FOR ALL PLOTS: White diamond is median Bayesian parameter sample, grey box is 3%-97% highe
st posterior density interval for parameter samples
FOR ALL PLOTS: Black diamond is median summary statistic for non-parametric bootstrapping
on OLS, black line is 2.5%-97.5% percentile for bootstrapped summary stat
```

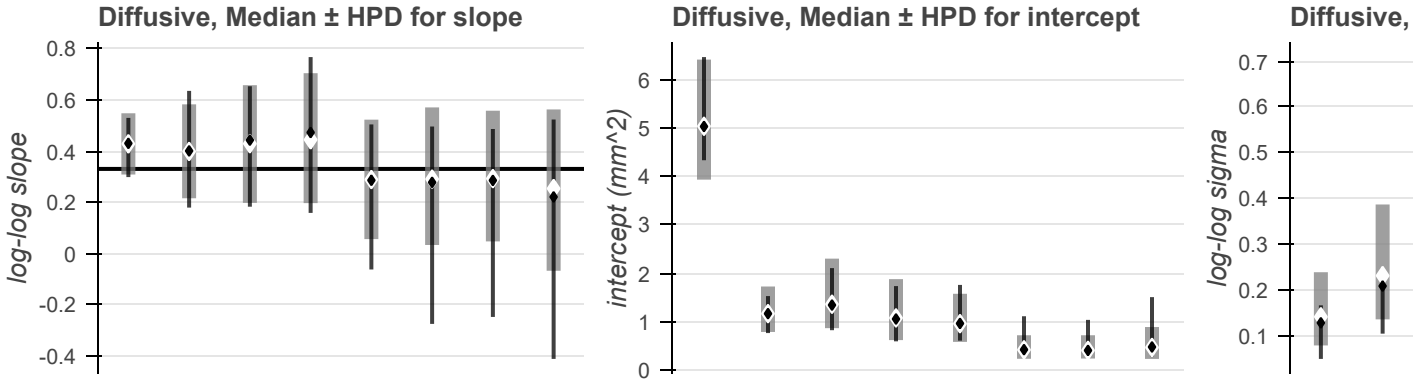
```
Area regressions (log-log slope, intercept (mm^2), log-log σ)
```



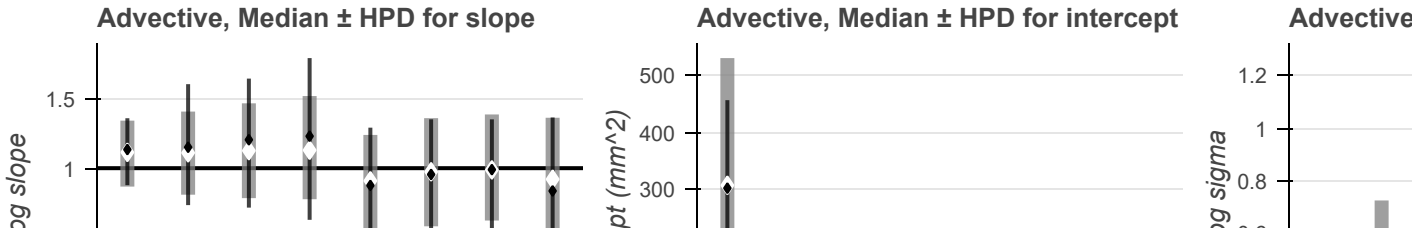

```
Depth regressions (log-log slope, intercept (mm), log-log σ)
```




```
Area/Depth regressions (log-log slope, intercept (G_diff), log-log σ)
```




```
Area^2/Depth regressions (log-log slope, intercept (G_adv), log-log σ)
```



In [12]:
```
bokeh.io.export_svgs(plots[7], filename='C:/Users/jwagne/Downloads/coef_var_depth.svg')
```

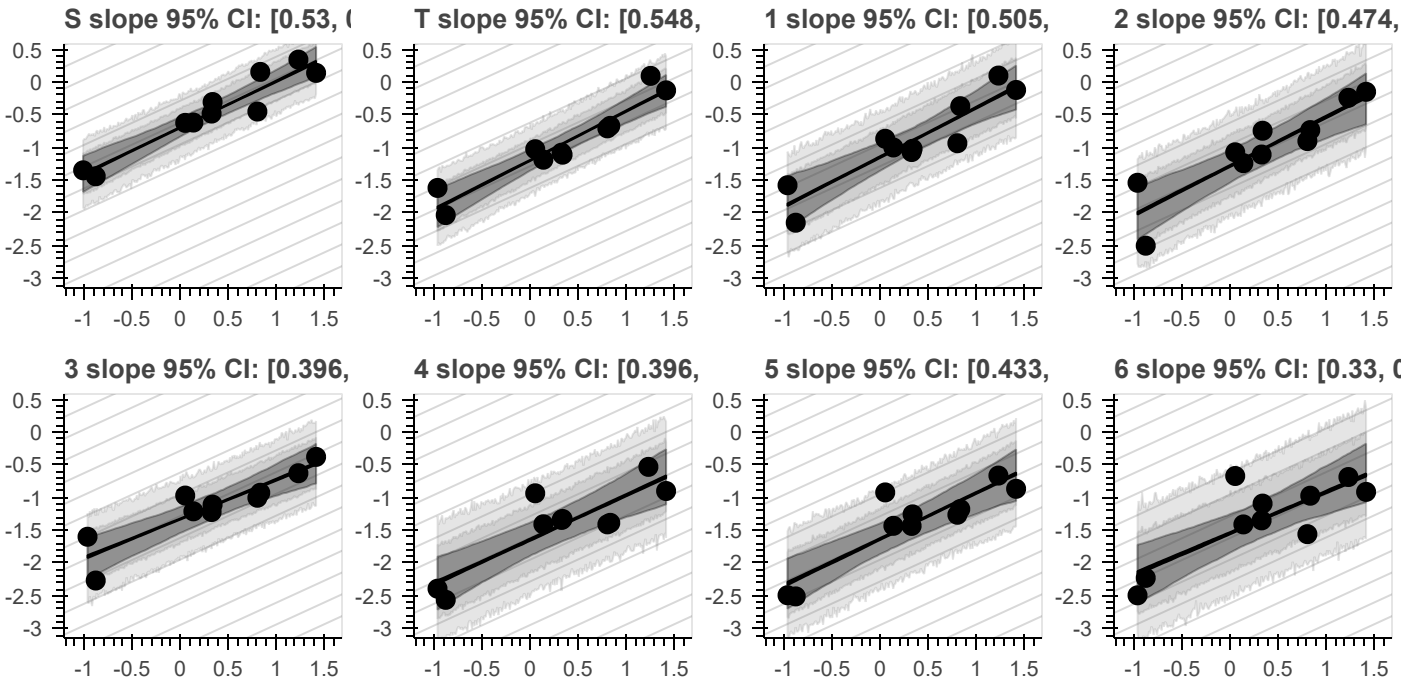
Out[12]:
```
['C:/Users/jwagne/Downloads/coef_var_depth.svg']
```

Great! With these summary stats in hand, we can also do some plots of all of these regressions themselves. Note that for all of these plots the blue ranges show the range of regression output values (either from sampling or bootstrapping) at the 2.5th-97.5th percentiles. Central line is the 45th-55th percentile range for Bayesian regression values, or the OLS line with non-resampled data for the the bootstrap regression plots. The two grey regions for the Bayesian regressions are the 95th and 80th percentile ranges for the posterior predictive distribution for the model.

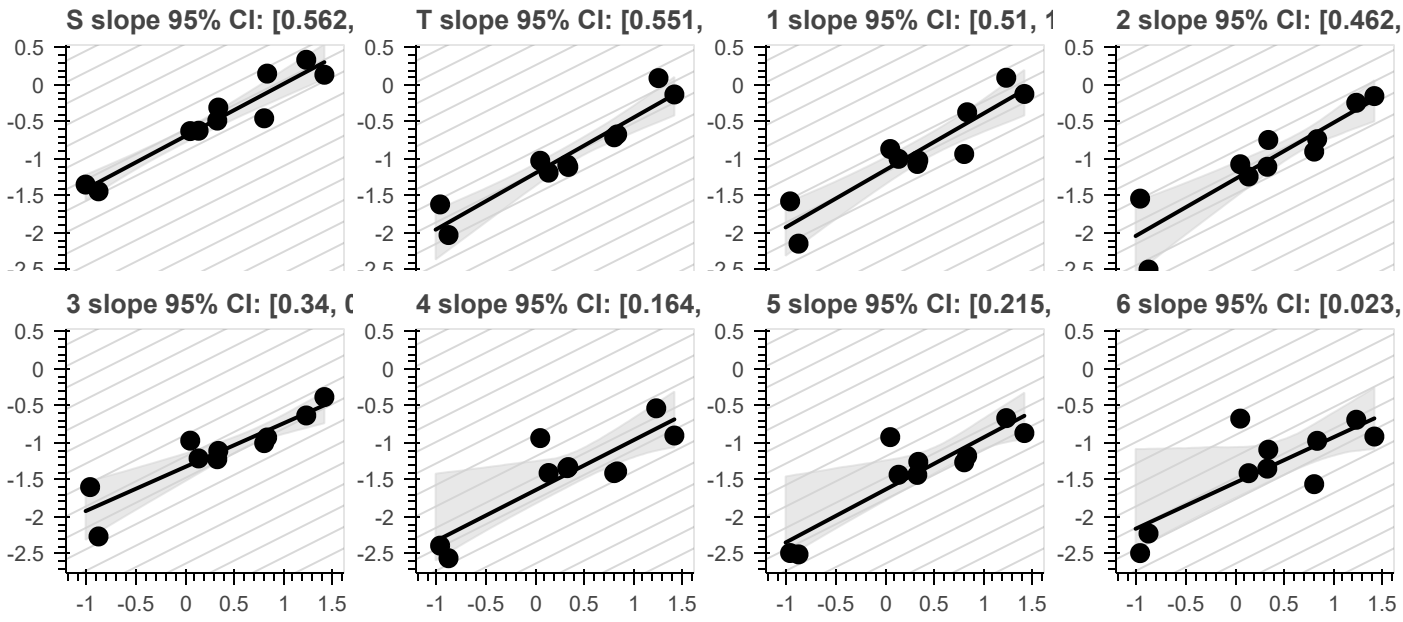
Plot for species averaged mass vs species averaged spiracle area (log transformed)

In [37]:
```
print("Bayesian regression:")
bokeh.io.show(bokeh.layouts.gridplot(all_regressions['log area (mm^2)'], ncols=4))
print("Non-parametric bootstrapping:")
b_plots = plotting_utils.make_plot(df_averages, 'log area (mm^2)', 2/3, width=175, height=
```

```
Bayesian regression:
```




```
Non-parametric bootstrapping:
```



In [ ]:

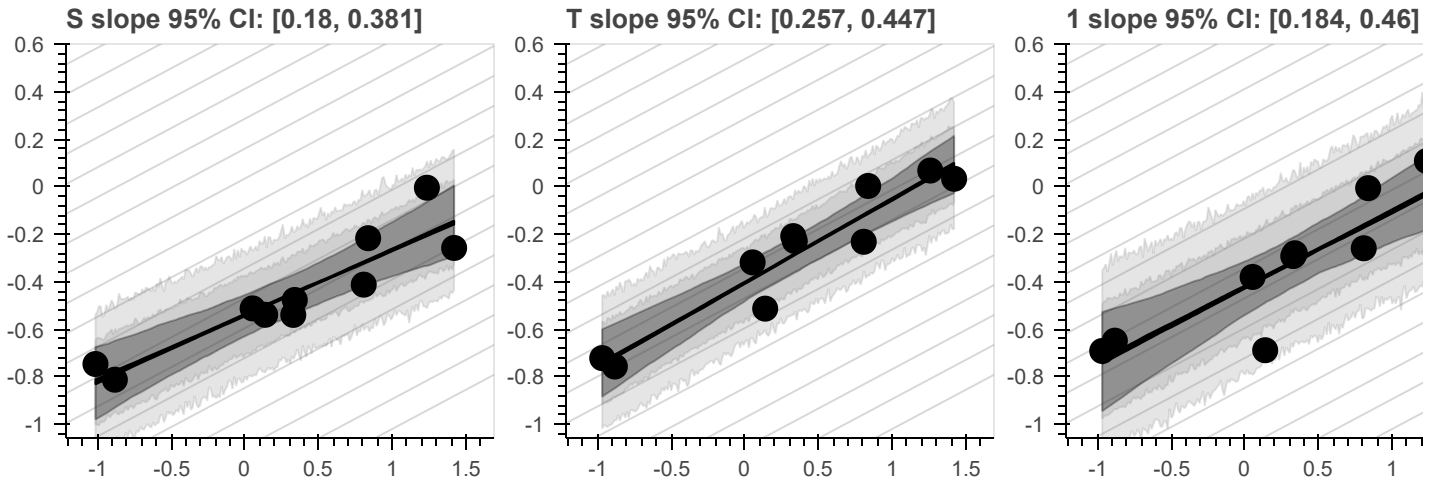
```
p = b_plots[0]
p.outline_line_color = None
p.yaxis.minor_tick_line_color = None
p.xaxis.minor_tick_line_color = None
bokeh.io.export_svgs(p, filename='C:/Users/jwagne/Downloads/S_area_nonparametric.svg')
```

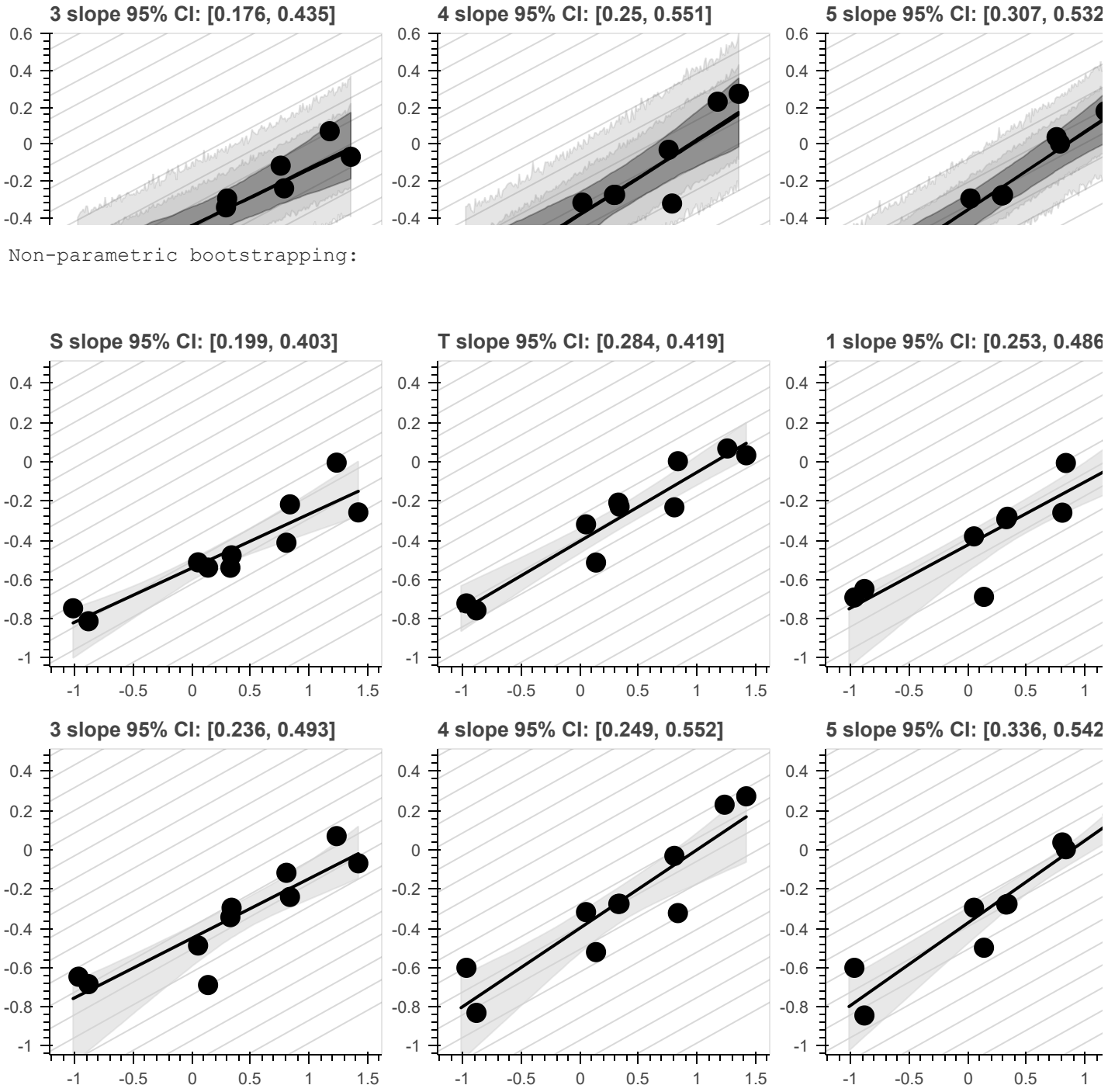
Species averaged mass vs species averaged spiracle depth (log transformed)

In [31]:

```
print("Bayesian regression:")
bokeh.io.show(bokeh.layouts.gridplot(all_regressions['log dist'], ncols=4))
print("Non-parametric bootstrapping:")
b_plots = plotting_utils.make_plot(df_averages, 'log dist', 1/3, width=250, height=250)
```

```
Bayesian regression:
```





In [16]:

```
p = b_plots[-1]
p.outline_line_color = None
p.yaxis.minor_tick_line_color = None
p.xaxis.minor_tick_line_color = None
bokeh.io.export_svgs(p, filename='C:/Users/jwagne/Downloads/6_depth_nonparametric.svg')
```

Out[16]:

```
['C:/Users/jwagne/Downloads/6_depth_nonparametric.svg']
```

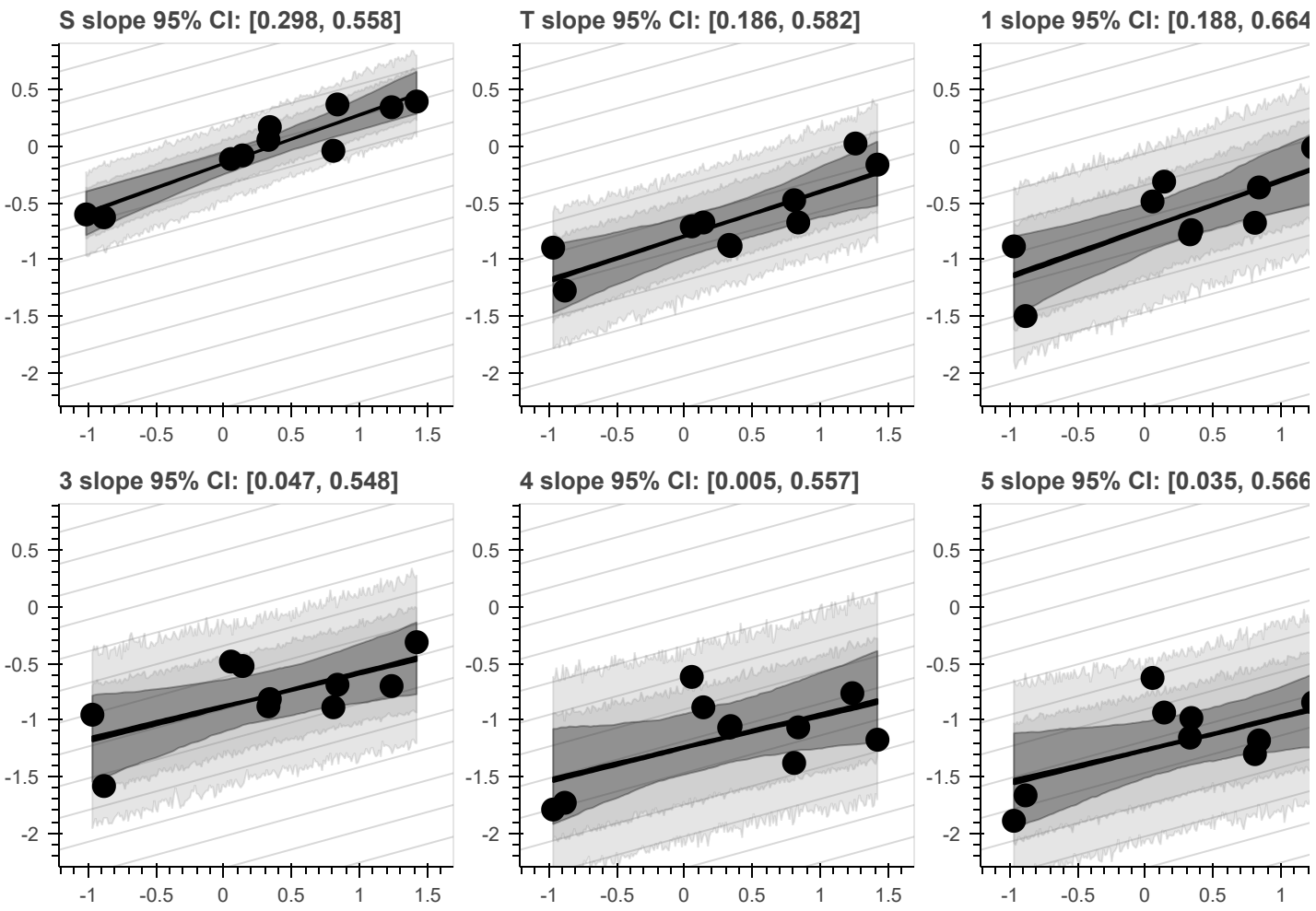
Species averaged mass vs $\frac{\text{species averaged area}}{\text{species averaged depth}}$ (log transformed)

In [29]:

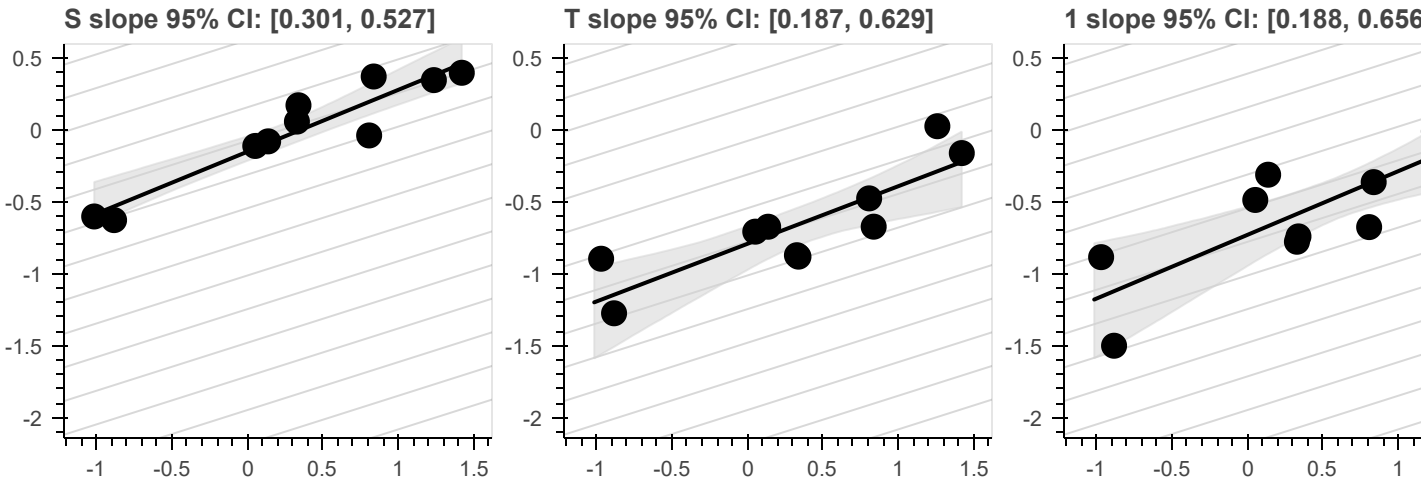
```
print("Bayesian regression:")
bokeh.io.show(bokeh.layouts.gridplot(all_regressions['log area/dist'], ncols=4))
```

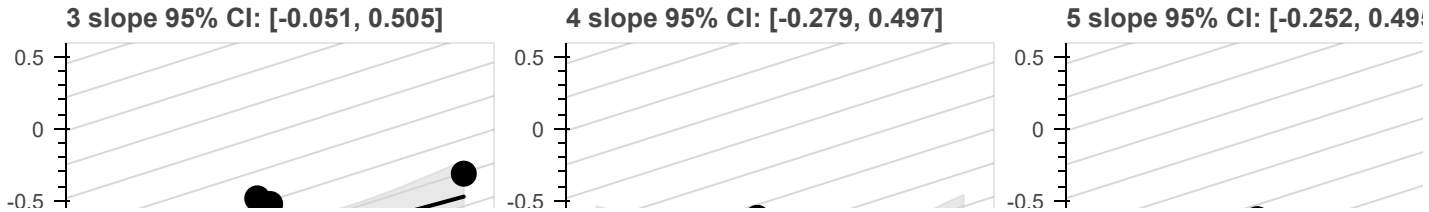
```
print("Non-parametric bootstrapping:")
plotting_utils.make_plot(df_averages, 'log area/dist', 1/3, width=250, height=250)
```

```
Bayesian regression:
```




```
Non-parametric bootstrapping:
```



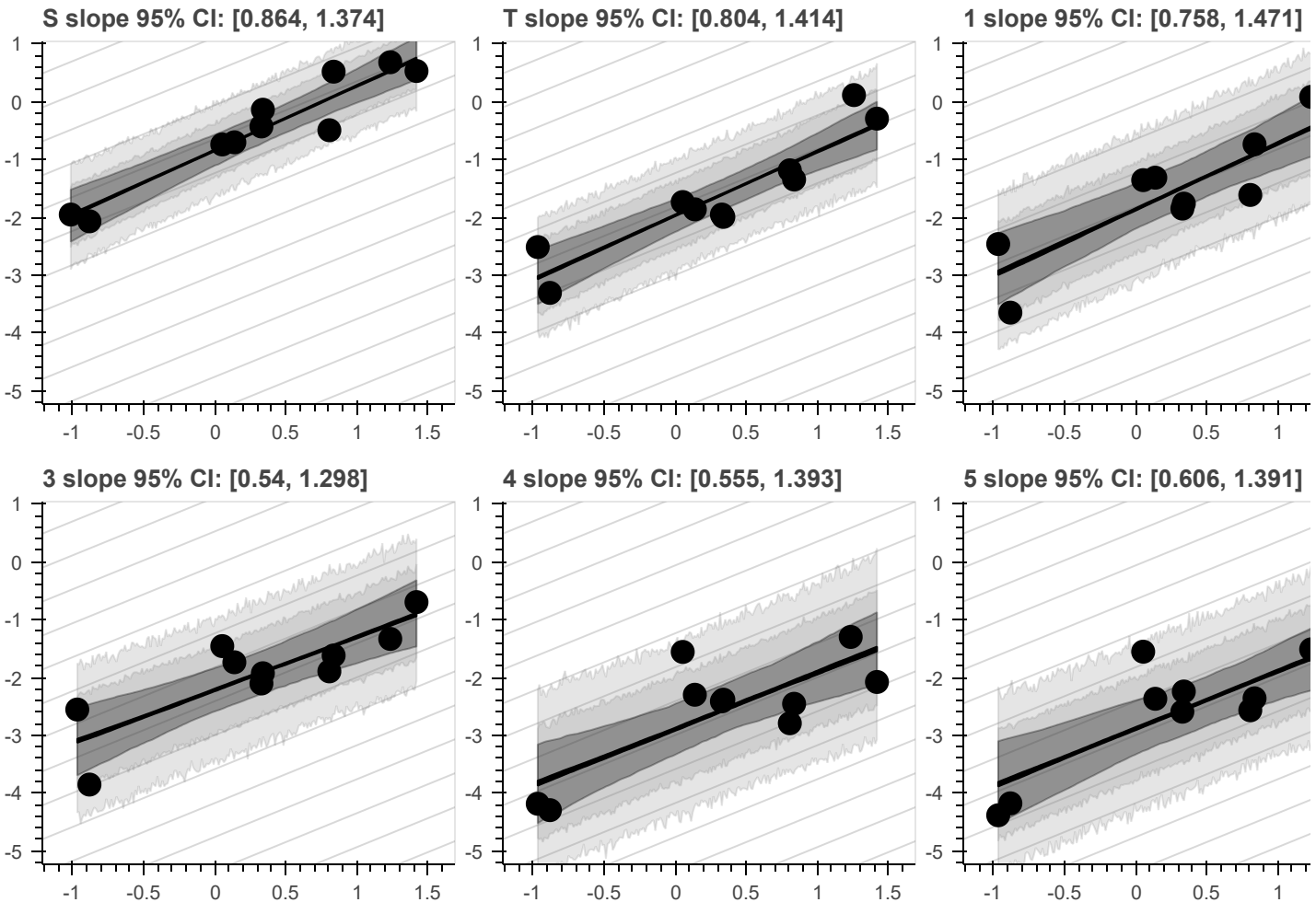


Out[29]:
```
[Figure(id='286658', ...),
 Figure(id='286787', ...),
 Figure(id='286916', ...),
 Figure(id='287045', ...),
 Figure(id='287174', ...),
 Figure(id='287303', ...),
 Figure(id='287432', ...),
 Figure(id='287561', ...)]
```

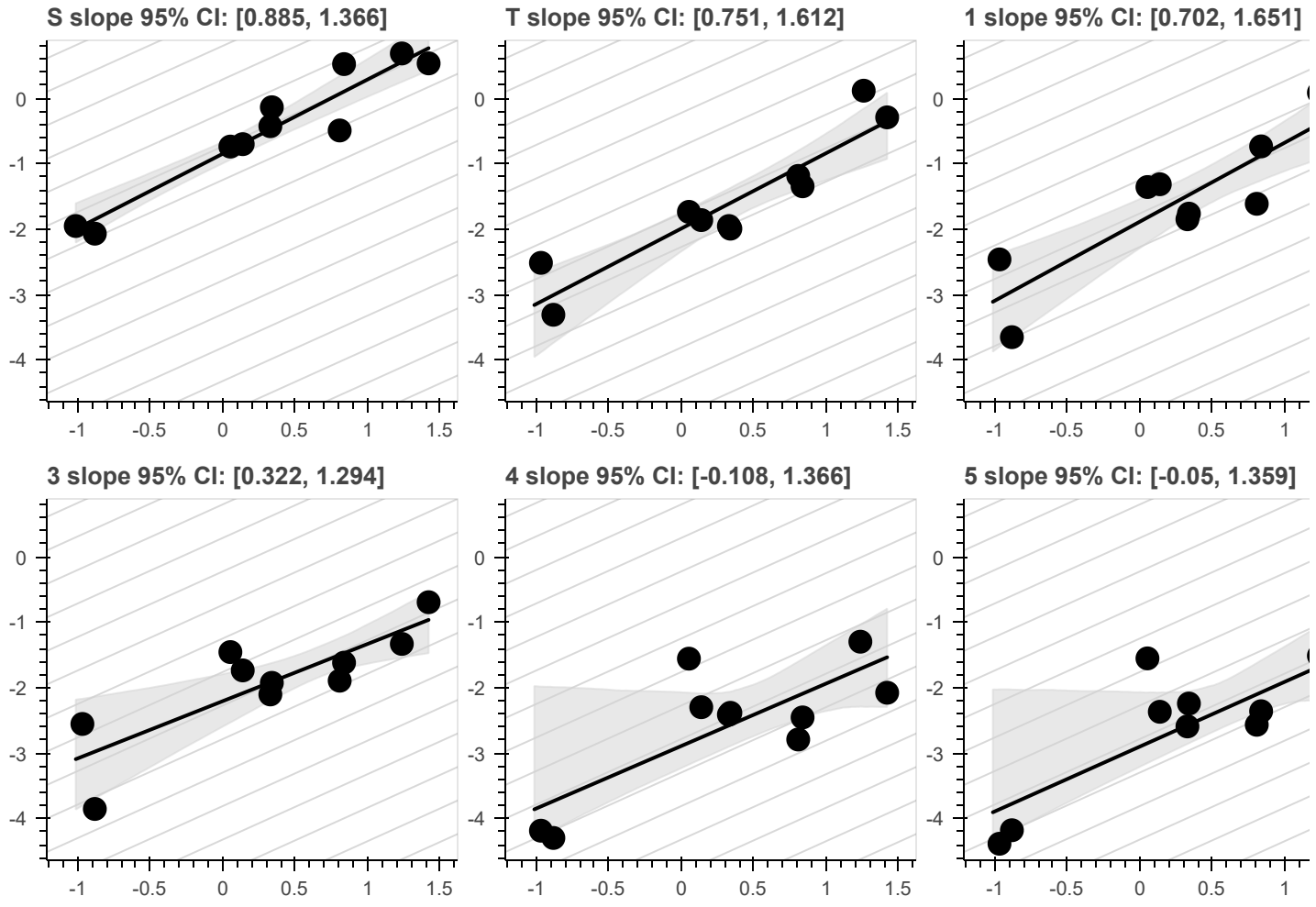
Species averaged mass vs $\frac{(\text{species averaged area})^2}{\text{species averaged depth}}$ (log transformed)

In [30]:
```
print("Bayesian regression:")
bokeh.io.show(bokeh.layouts.gridplot(all_regressions['log area^2/dist'], ncols=4))
print("Non-parametric bootstrapping:")
plotting_utils.make_plot(df_averages, 'log area^2/dist', 1, width=250, height=250)
```

```
Bayesian regression:
```



Non-parametric bootstrapping:



Out[30]:
```
[Figure(id='317368', ...),
 Figure(id='317502', ...),
 Figure(id='317636', ...),
 Figure(id='317770', ...),
 Figure(id='317904', ...),
 Figure(id='318038', ...),
 Figure(id='318172', ...),
 Figure(id='318306', ...)]
```

In [20]:
```
%reload_ext watermark
%watermark -p bokeh,cmdstanpy
```

```
bokeh 1.4.0
cmdstanpy 0.8.0
```

In [ ]: