6 Quantifying Epistasis

January 31, 2018

1 Table of Contents


```
In [ ]:
```

2 Introduction

In this notebook, we develop the notion of 'genome-wide epistasis'. Genome-wide epistasis is a generalization of the methods used to measure epistasis between genotypes using qPCR. Why genome-wide epistasis can even begin to appear seems a bit mysterious, and we briefly touch on this philosophical aspect at the end of the notebook.

3 Transcriptome-wide epistasis: A definition

Epistasis is defined by Huang and Sternberg (2006) as one allele masking another allele's phenotype. In other words, if an allele X has a phenotype Ph_1, and an allele Y (at a different locus) has a different phenotype Ph_2, we can say that X and Y are epistatic if the double homozygote has a phenotype that is equal to either Ph_1 or Ph_2. Epistasis is also known as non-additivity, and it is the basis of the definition of genetic interactions. Of course, stating that two genes are epistatic to each other is subject to a large number of qualifiers. A particularly important qualifier is that the phenotypes under study must have a reasonable dynamic range—they must not be too strong or too subtle, or non-additivity could occur simply as a result of a compressed range. Another important consideration is that the alleles used to study a genetic interaction must be complete loss of function alleles for the phenotype under consideration. If they are not, trouble can arise from making inferences that are just too strong.

The null hypothesis when observing two mutants of different genes is that they do not interact. Therefore, when the double mutant is made, the result must be that the two phenotypes add. We reasoned that, ideally, this should also be the case for vectorial phenotypes. This enabled us to make a prediction about what a double mutant would look like. Given two alleles X and Y that code for different genes (i.e. that complement), the double mutant X^-Y^- should have expression levels equal to:

$$\beta_{XY,Predicted\ Additive,i} = \beta_{X,i} + \beta_{Y,i},$$

where $\beta_{G,i}$ is the regression coefficient (from Sleuth) for genotype G and isoform i, and $\beta_{XY,Predicted\ Additive,i}$ is the predicted expression of isoform i in a double mutant of X and Y under an additive model. Since we have data for double and single mutants, we reasoned that we should be able to plot the predicted expression, $\beta_{XY,Pred,i}$, against deviations from the predicted expression $\Delta_i = \beta_{XY,i} - \beta_{XY,Pred}$. Given these two numbers (the predicted additive effect and the deviation from predicted), we can generate an epistasis plot, where the X-axis reflects the predicted expression level of the double mutant assuming an additive model, and the Y-axis defines the deviation from predicted. For additive mutants, we expect to see that the genes fall along the line $\Delta_i = 0$ with some noise ϵ_i.

Having defined our null hypothesis, it is now possible to explore what other results could be expected. Suppose that X and Y act along a single, activating pathway of the form $X \rightarrow Y \rightarrow Ph$ or $Y \rightarrow X \rightarrow Ph$. In that case, both genes should: 1. Act on the same phenotype 2. Have the same magnitude of effect.

We can predict the additive effect of an additive interaction when both genes have the same effect on a phenotype, it should be $2\beta_{X,i} = 2\beta_{Y,i} = 2\beta_i$. We can also reason about what the phenotype of the mutant should be. If the two genes are acting along a single pathway, breaking the pathway twice should have the same effect as breaking it once. Therefore, it must be the case that $\beta_{XY,Pred,i} = \beta_i$. Next, we can calculate that the idealized deviation from the additive value should be $\Delta_i = \beta_i - 2\beta_i = -\beta_i$. Putting it all together, we then would expect the coordinates for each isoform to be:

$$(2\beta_i, -\beta_i),$$

which suggests that when two genes interact positively through a single unbranched pathway, an epistasis plot should show points that fall along the line $y = -0.5x$.

What about a model where $X - -|Y$? For this case, I will invoke a limit argument. Suppose that, under "usual laboratory conditions" (whatever those are!), X is `ON` and it is often present in large quantities in the cell. Suppose further, that X is the strongest possible (non-competitive) inhibitor of Y. Then it follows that under usual conditions, Y must be `OFF`. Therefore, a null mutant

of Y should look transcriptomically wild-type or very close to it. The predicted expression of a double mutant should therefore be $\beta_{X,i} + \beta_{Y,i} \sim \beta_{X,i}$. We can reason about the actual expression level of a double mutant as follows: If X inhibits Y, then removing X causes a large increase in the protein levels of Y. However, removing Y from the X^- animal means that protein levels of Y return to wild-type. This is an effect known as suppression. Suppression means that the allele that is downstream of the inhibitor defines what the phenotype will be. Therefore, the expression phenotype of this double mutant, $\beta_{XY,i} = \beta_{Y,i}$. With this number in hand, we can now calculate $\Delta_i = \beta_{Y,i} - \beta_{Y,i} - \beta_{X,i} = -\beta_{X,i}$. From this, it follows that the points will fall near the coordinates,

$$(\beta_{X,i}, -\beta_{X,i}).$$

In other words, the points will fall along the line $y = -x$.

At this point, we have covered most of the notable simple cases. Only two remain. Suppose that for two mutants under study, the double mutant expresses the phenotype of one of the single mutants. This means that $X^-Y^- = X^-$. What slope should we observe? Well, clearly we can predict the additive (x-axis) coordinate: $\beta_{X,i} + \beta_{Y,i}$. What about the deviation from additive? Well, if the double mutant looks like the mutant X^-, then it follows that the expression should also match. In other words, $\beta_{XY,i} = \beta_X$. From this, we can predict the coordinates of each point on the epistasis plot to be:

$$(\beta_{X,i} + \beta_{Y,i}, -\beta_{Y,i}).$$

What does this mean? Well, if $\beta_{X,i}$ was completely uncorrelated from $\beta_{Y,i}$, we might be tempted to say that this should still fall along the line of $y = -x$, perhaps with more noise than the case of suppression. However, this is not the case! $\beta_{X,i}$ and $\beta_{Y,i}$ are covariant! Nothing remains but to make a line of best fit. The closer this α is to -0.5, the closer the two genes are to interacting exclusively in a linear manner; the closer the slope is to -1, the closer these genes are to being in the limit of strong suppression. Anything in between? Well, the in-between is also interpretable.

How can we know that the points will fall on a straight line? Well. Let us consider a branched pathway, where $X \rightarrow Y \rightarrow Ph$, but $X \rightarrow Ph$ is also true (i.e., X acts on Ph in Y-dependent and independent manners). How do we know these will form a line? Well, suppose that the effect of X on Ph is complete. Then this means that $XY = X$. If X interacts with Ph in a simple manner (i.e., suppose X activates a transcription factor that mediates Ph), then we can make the following statement: Y accounts for a fraction f of the interaction of X on Ph.

Given the above statement is true, then it follows that

$$\beta_{Y,i} = f\beta_{X,i}.$$

Then we can now predict the additive effect of the double mutant:

$$\beta_{XY,AddPred,i} = (1 + f)\beta_{X,i}.$$

However, because we know that $XY = X$, we know that the expression of the double mutant will match the expression of X. Therefore, the expected deviation of the double mutant should be

$$\Delta_i = -f\beta_{X,i},$$

and the data will fall along the coordinates $((1 + f)\beta_{X,i}, -f\beta_{X,i})$. Therefore, the points will fall along the line:

$$y = -\frac{f}{1+f}x$$

Notice that f can only range from [0, 1], which restricts the range of slopes from [0, -0.5].

In [1]:
```
# important stuff:
import os
import pandas as pd
import numpy as np
import statsmodels.tools.numdiff as smnd
import scipy

# TEA and morgan
import morgan as morgan
import epistasis as epi
import gvars

# Graphics
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rc
rc('text', usetex=True)
rc('text.latex', preamble=r'\usepackage{cmbright}')
rc('font', **{'family': 'sans-serif', 'sans-serif': ['Helvetica']})

from scipy.stats import gaussian_kde

# Magic function to make matplotlib inline;
%matplotlib inline

# This enables SVG graphics inline.
%config InlineBackend.figure_formats = {'png', 'retina'}

# JB's favorite Seaborn settings for notebooks
rc = {'lines.linewidth': 2,
      'axes.labelsize': 18,
      'axes.titlesize': 18,
      'axes.facecolor': 'DFDFE5'}
sns.set_context('notebook', rc=rc)
sns.set_style("dark")

mpl.rcParams['xtick.labelsize'] = 16
mpl.rcParams['ytick.labelsize'] = 16
mpl.rcParams['legend.fontsize'] = 14
```

In [2]:
```
q=0.1
genvar = gvars.genvars()

# Specify the genotypes to refer to:
single_mutants = ['b', 'c', 'd', 'e', 'g']
double_mutants = {'a' : 'bd', 'f':'bc'}
```

In [3]:
```
tidy_data = pd.read_csv('../output/temp_files/DE_genes.csv')
```

```
tidy_data.sort_values('target_id', inplace=True)
tidy_data.dropna(subset=['ens_gene'], inplace=True)
```

In [4]:
```
tidy_data.head()
```

Out[4]:

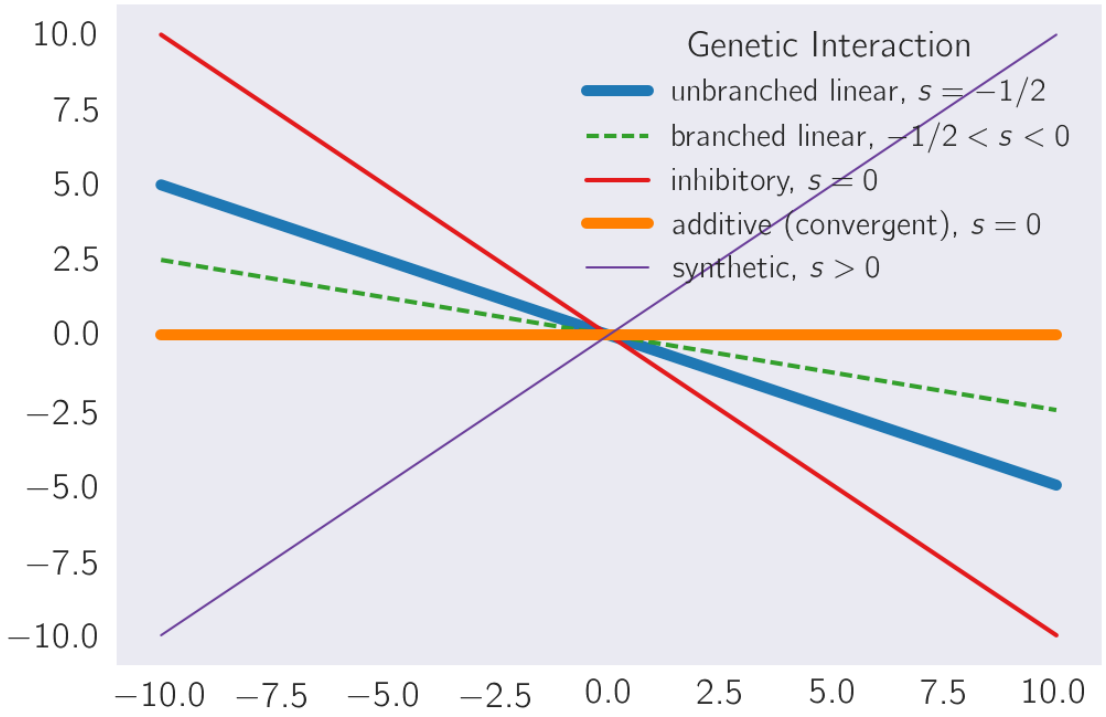
	ens_gene	ext_gene	target_id	b	se_b	qval	genotype	sorter	code
0	WBGene00007064	2RSSE.1	2RSSE.1a	0.150147	0.829418	1.000000	fog-2	5	g
19676	WBGene00007064	2RSSE.1	2RSSE.1a	0.809959	0.586487	0.496563	rhy-1	1	e
118056	WBGene00007064	2RSSE.1	2RSSE.1a	1.121038	0.586487	0.216276	egl-9;vhl-1	6	a
39352	WBGene00007064	2RSSE.1	2RSSE.1a	0.934036	0.586487	0.409735	egl-9	2	b
98380	WBGene00007064	2RSSE.1	2RSSE.1a	0.519789	0.586487	0.791051	hif-1	4	c

Before we begin, let's make a schematic diagram of what the slopes should look like:

In [5]:
```
X = np.linspace(-10, 10)
Y = -1/2*X

plt.plot(X, -1/2*X, ls='-', color= '#1f78b4', lw=5, label='unbranched linear, $s=-1/2$
plt.plot(X, -1/4*X, ls='--', color= '#33a02c', label='branched linear, $-1/2 < s < 0$'
plt.plot(X, -X, ls='-', lw=2, color= '#e31a1c', label='inhibitory, $s = 0$')
plt.plot(X, 0*X, 'k-', lw=5, color= '#ff7f00', label='additive (convergent), $s = 0$')
plt.plot(X, X, '-', lw=1,color= '#6a3d9a',  label='synthetic, $s > 0$')

lgd = plt.legend()
lgd.set_title('Genetic Interaction', prop=(mpl.font_manager.FontProperties(size=16)))
plt.savefig('../output/epistasis_plot_show.svg', bbox_inches='tight')
```



4 Introduction to epistasis plots

Having worked out the theory, we can now make the epistasis plot given our data. Let's plot this for *egl-9* and *vhl-1*.

4.1 *egl-9* is epistatic to *vhl-1*

As a first step, I will define what genotypes I am working with. In this case, we want to work with the *egl-9*, *vhl-1* and *egl-9;vhl-1* genotypes.

```
In [6]: letter1 = 'b'
        letter2 = 'd'
        double = genvar.double_mapping[letter1 + letter2]
```

The procedure to follow now is as follows:

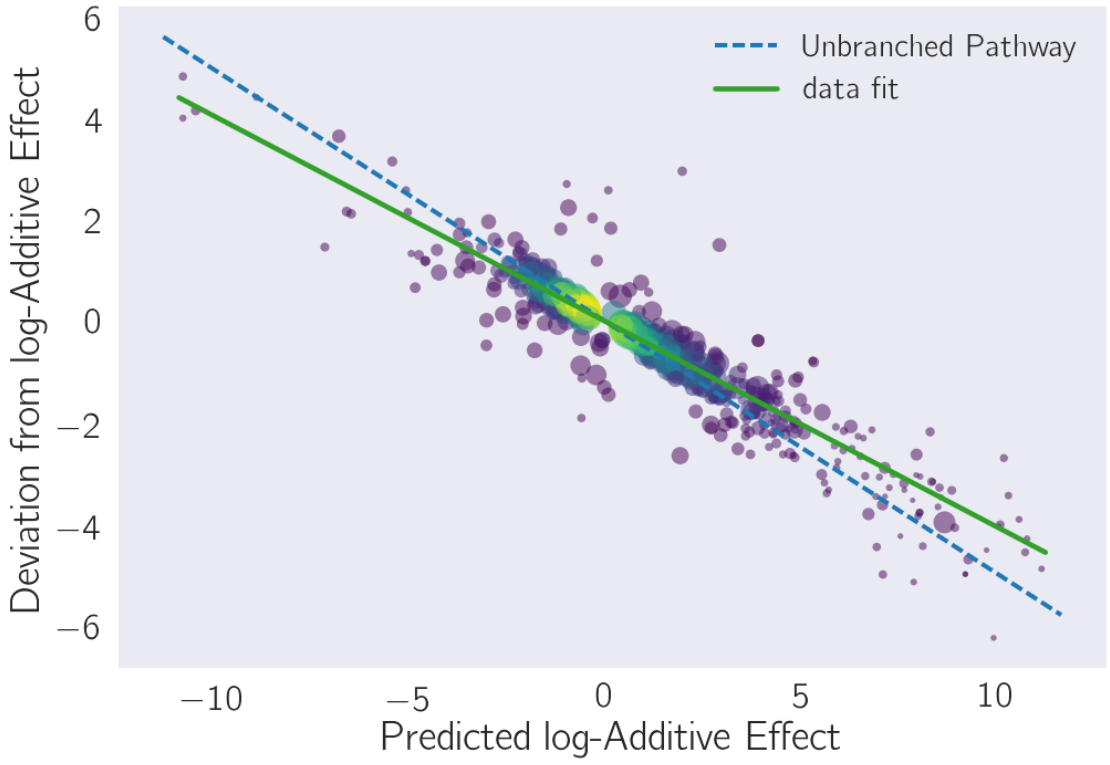
1. Find the set of genes that are differentially expressed (direction agnostic) between the three genotytpes. Call that set D
2. For the set D, make a prediction of what the double mutant looks like by adding the single mutants (additive null model). Calculate the y-axis by taking the difference between the observed coefficient and the expected.
3. Calculate error bars—remember, variances add.
4. Find the line of best fit using Orthogonal Distance Regression with `scipy.odr`.
5. Plot.

I have implemented this procedure in the function `epi.epistasis_plot`, and I call it below. It returns a set of four things: * x - a dataframe containing the identities, beta and q-values of the first letter that was passed to the function (in this case, the *egl-9* genes) * y - same, but for the second genotype (*vhl-1*) * xy - same, but for the double mutant * ax - the plot axis

4.2 Figure 5B

```
In [7]: x = epi.find_overlap([letter1, letter2, double], tidy_data)
```

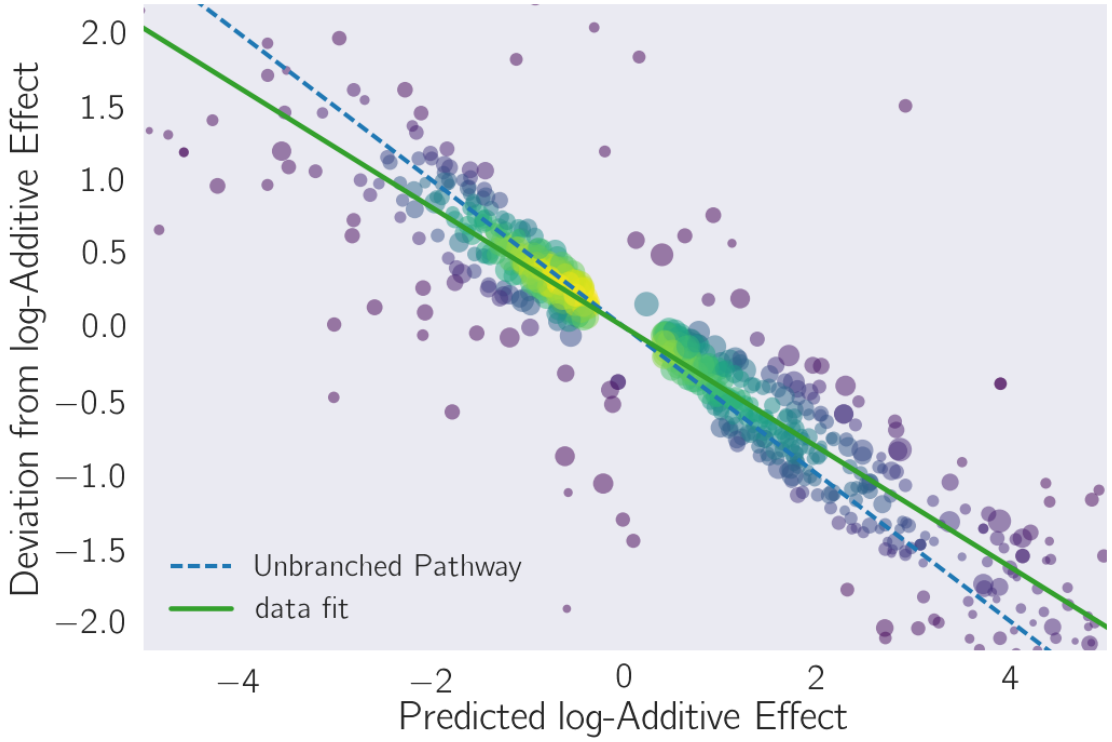
```
In [8]: x, y, xy = epi.find_STP([letter1, letter2], double, tidy_data)
        epi.ODR([x, y], xy, 'actual')
        x, y, xy, ax = epi.make_epiplot([letter1, letter2], double, tidy_data)
        plt.savefig('../output/epistasis{0}{1}.svg'.format(genvar.mapping[letter1], genvar.mapp
                    bbox_inches='tight')
```



The points all fall along a line!!! Yes! We could even look at it in a little more detail, see how the scatter looks like if we zoom in.

```
In [9]: _ = epi.make_epiplot([letter1, letter2], double, tidy_data)
        plt.xlim(-5, 5)
        plt.ylim(-2.2, 2.2)
```

```
Out[9]: (-2.2, 2.2)
```



Let's figure out what the calculated slope of best fit is

```
In [10]: # calculate slope from the data:
         actual = epi.ODR([x,y], xy, epistasis='actual')
         actual.pprint()
```

```
Beta: [-0.40766836]
Beta Std Error: [ 0.00590136]
Beta Covariance: [[  2.84372564e-05]]
Residual Variance: 1.2246619461047352
Inverse Condition #: 1.0
Reason(s) for Halting:
  Sum of squares convergence
```

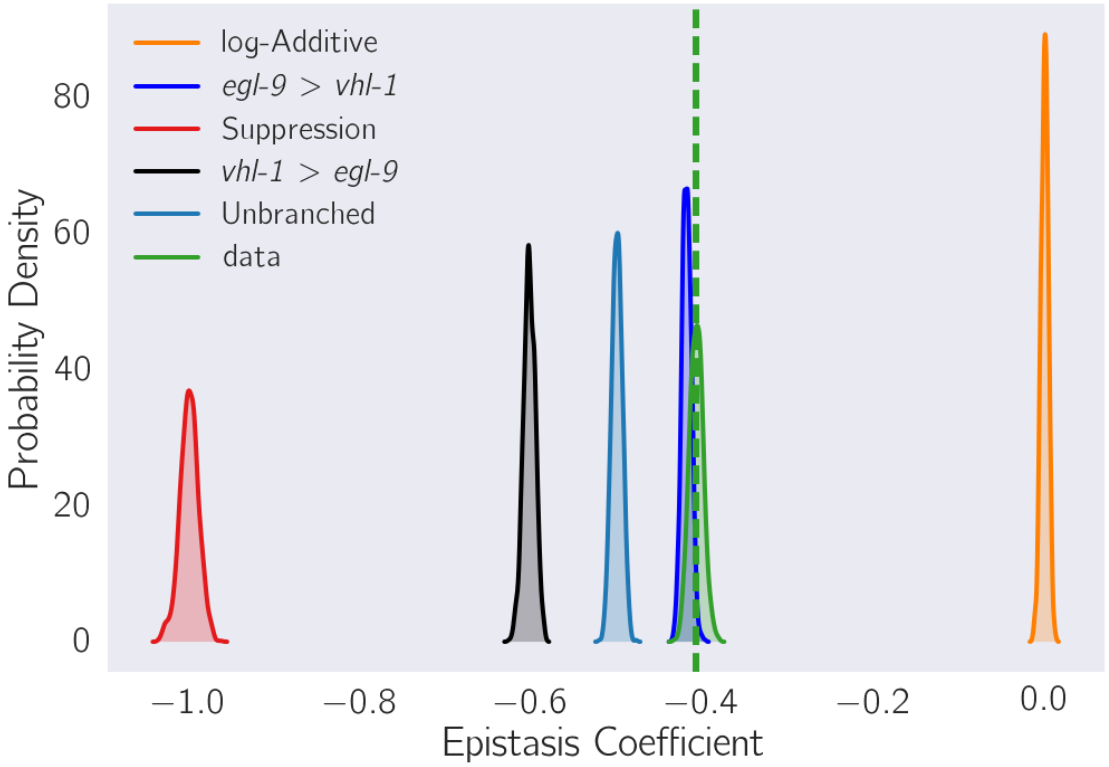
Ok. It's a line, even though it does have scatter. Fortunately, the largest points are pretty close to the line of best fit, which has a slope of -0.41 ± 0.006. Now, what we need to do is perform all of the simulations for the epistasis possibilities. We will also bootstrap the observed distribution just to make sure the slope distribution is what we think it is.

```
In [11]: s = epi.calculate_all_bootstraps(letter1, letter2, double, tidy_data, nsim=1000)
```

4.3 Figure 5C

```
In [12]: ax = epi.plot_bootstraps(letter1, letter2, s, cumulative=False, shade=True)
         plt.xlabel('Epistasis Coefficient')
```

```
plt.ylabel('Probability Density')
plt.savefig('../output/kde-epistasis{0}{1}.svg'.format(genvar.mapping[letter1], genva
            bbox_inches='tight')
```




```
In [13]: np.median(s['actual'])
```

```
Out[13]: -0.40739269084289192
```

Alright! The predicted epistatic curve fits the data perfectly!! Woohoo!!! And the unbranched curve doesn't even overlap with the other ones. We could tentatively say that it looks like we are dealing with a branched pathway of some sort. From part 1 above, we had concluded that the relationship between the slope and the fraction of the effect mediated through the 'main' pathway was:

$$\alpha = \frac{f}{1+f}.$$

We can invert this equation to solve for f, which yields, $f = \alpha/(1-\alpha)$. Plugging in, we find that $f = 0.42/.58 = 0.72$. 72% of the inhibition of HIF-1 by EGL-9 is through the VHL-1-dependent degradation pathway. The other 28% is presumably coming from the SWAN-1-dependent pathway. In order to truly have any confidence in this result, we should have a different way to check. Let's implement a Bayesian Odds Ratio test and see whether we can choose a model this way.

5 Odds ratios

We will perform pairwise comparison between a free model with variable slope and the five theoretical models we tested. First, we need to define the Bayesian function we must optimize. It will be:

$$P(D\,|\alpha, M_1, I) \propto \prod_{(x_i,y_i,w_i)\in D} \exp\left(\frac{(y_i - \alpha\cdot x_i)^2}{w_i}\right)\cdot(1+\beta^2)^{-3/2},$$

where (x_i, y_i) are the coordinates of the point D_i, and w_i is the standard error of y_i. For the theoretic models, we will find the probability,

$$P(D\,|M_i, I) \propto \prod_{(x_i,y_i,w_i)\in D} \exp\left(\frac{(y_i - y_{pred,i})^2}{w_i}\right),$$

where $y_{pred,i}$ is the predicted y-value under model i. Finally, we will approximate the odds ratio by using a Laplace approximation of the functions where the probability is maximized. Briefly, model selection is performed by evaluating the quotient:

$$O_{1i} = \frac{P(M_1\,|I)}{P(M_i\,|I}\frac{P(D\,|M_1, I)}{P(D\,|M_i, I)}$$

The first term in the odds ratio is impossible to evaluate. We cannot know the probability of one model versus another. Qualitatively, we might say that certain models are more likely (for example, tried and true physical models are more likely than brand new recently invented models that come out of nowhere), but we cannot easily assign a number to them. Arbitratrily, we will assign the simpler models slight support, because genetics has been around for a long time. So, we will say the first term is equal to $\exp -2$ in favour of the theoretical models M_j. What is the second term?

Let's remember that the model we specified above is in terms of $P(D\,|M_1, \alpha, I)$. We can get rid of α by marginalizing:

$$P(D\,|M_1) = \int d\alpha\, P(D\,|\alpha, M_1, I).$$

We can use a laplacian approximation on this integral to obtain:

$$P(D\,|M_1) \sim P(D\,|\alpha^*, M_1, I)\cdot P(\alpha^*|M_1, I)\sqrt{2\pi}\sigma_1,$$

where α^* is the *Maximum A Posteriori* (MAP) estimate of α, and σ_1 is the covariance of the Gaussian approximation of the posterior around the point α^*. Therefore, we can now calculate the approximate odds ratio:

$$O_{1i} = \exp(-2)\frac{P(D\,|\alpha^*, M_1, I)\cdot P(\alpha^*|M_1, I)\sqrt{2\pi}\sigma_1}{P(D\,|M_i, I)}.$$

Let's code all of this up!

5.1 Writing the theoretical models

In [14]:
```
# bayes model fitting:
def log_likelihood_fixed(w, x, y, model, alpha=None):
    """Likelihood probability for the theoretical models of epistasis"""
    sigma = w
    epistasis = ['actual', 'xy=x', 'xy=y', 'xy=x=y', 'xy=x+y',
                 'suppress']

    # errors:
    if model not in epistasis:
        raise ValueError('model is not allowed')

    if (model is 'xy=x') or (model is 'xy=y'):
        if alpha is None:
            raise ValueError('alpha cannot be none for epistasis models `xy=x` or `xy=

    # pick your model
    if model == 'xy=x+y':
        y_model = 0

    elif model == 'xy=x=y':
        y_model = -1/2*x

    elif model == 'suppress':
        y_model = -x

    elif (model == 'xy=x') or (model == 'xy=y'):
        y_model = alpha[model]*x

    # return the probability function
    return -0.5 * np.sum(np.log(2 * np.pi * sigma ** 2) + (y - y_model) ** 2 / sigma *

def log_posterior_fixed(w, x, y, model, alpha=None):
    """The posterior probability of the theoretical models"""
    return log_likelihood_fixed(w, x, y, model, alpha)
```

5.2 Writing the free slope model

In [15]:
```
def log_prior(theta):
    """Pareto prior, which makes the lines be evenly sampled between (-1,1) and plus\
    return -1.5 * np.log(1 + theta ** 2)

def log_likelihood(theta, x, y, w):
    """Calculates the weighted chi-square for the free model"""
    sigma = w
    y_model = theta * x
    return -0.5 * np.sum(np.log(2 * np.pi * sigma ** 2) + (y - y_model) ** 2 / sigma *
```

```
def log_posterior(theta, x, y, w):
    """The complete logarithm of the posterior"""
    return log_prior(theta) + log_likelihood(theta, x, y, w)

def neg_log_prob_free(theta, x, y, w):
    """Negative log of the posterior for using scipy.optimize.minimize."""
    return -log_posterior(theta, x, y, w)
```

5.3 Writing the Odds Ratio function

Procedure to follow:

1. Find the MAP for the probability function of the free model. It should agree closely but not exactly with the result from `scipy.ODR` because we are using a slightly different method.
2. Calculate the variance of the logarithm of the posterior as $(dP\ /\ d\alpha)^{-}1$
3. Calculate $P(D\,|M_i, I)$ for each theoretical model M_i
4. Calculate the Odds Ratio and print the results.

```
In [16]: def model_selection(X, Y, wdev, alpha, **kwargs):
             """
             Finds MAP for the free model, then does OR calculation for free model versus theo

             Params:
             -------
             X - The x-coordinates of the points to be used
             Y - y-coordinates
             wdev - the error in the y-coordinates
             alpha - slope for XY=X and XY=Y models. Must be a dictionary of the form {'XY=X':
             guess - starting guess for the MAP approximation (we will use the output from sci

             Outputs:
             Prints the OR for the models.
             """
             guess = kwargs.pop('guess', -0.5)

             # calculate probability of free model:
             res = scipy.optimize.minimize(neg_log_prob_free, guess, args=(X, Y, wdev), method=

             # Compute error bars
             second_derivative = scipy.misc.derivative(log_posterior, res.x, dx=1.0, n=2, args=
             cov_free = -1/second_derivative
             alpha_free = np.float64(res.x)
             log_free = log_posterior(alpha_free, X, Y, wdev)

             # log goodness of fit for fixed models
             eps = ['xy=x', 'xy=y', 'xy=x=y', 'xy=x+y',
                    'suppress']
```

```
good_fits = {}

for epistasis in eps:
    log_MAP = log_posterior_fixed(wdev, X, Y, model=epistasis, alpha=alpha)

    good_fits[epistasis] = log_free - log_MAP


    # occam factor - only the free model has a penalty
    log_occam_factor =(-np.log(2 * np.pi) + np.log(cov_free)
                       - 0) / 2

    # give more standing to simpler models. but just a little bit!
    lg = log_free - log_MAP + log_occam_factor - 2
    print('{0} Odds Ratio: {1:2g}'.format(epistasis, np.exp(lg)))

std = np.float64(np.sqrt(cov_free))
print('the value used for the observed fit was {0:.3g} +/- {1:.3g}'.format(alpha_
```

5.4 Odds ratio for the epistasis between *egl-9* and *vhl-1*

Now that we have written our odds ratio functions, we should test it. Now, one thing to bear in mind is that we have written the theoretical models in such a way that they are extremely conservative. This means that ANY systematic deviation from them will rapidly lead to their rejection in favor of the slightly more complex (but theoretically less pleasing) free-slope model. As a result, we need to be careful how we interpret the Odds ratio. Here are some guidelines:

- Reject theoretical models when there is strong support for them. This means reject when $OR > 10^3$
- When in need of an interpretation, selecting the model with the best support is also valid. Rejecting a model just means we need to keep in mind the epistasis is not exactly what we expected... but when push comes to shove we have to pick a conclusion. Select the conclusion with the most evidence, i.e., the lowest odds ratio.
- Use your gut. If something isn't right, study it more. Let the data speak until you can resolve the controversy.

In [18]:
```
alpha_eglvhl = {'xy=x': s['xy=x'].mean(),
                'xy=y': s['xy=y'].mean()
               }

# Calculate X-coordinates
X = x.b.values + y.b.values
# Calculate Y-coordinates
Y = xy.b.values - X
# Calculate the corrected standard error for the Y-axis
wdev = np.sqrt(x.se_b.values**2 + y.se_b.values**2 + xy.se_b.values**2)
# do the model selection
model_selection(X, Y, wdev, alpha=alpha_eglvhl, guess=actual.beta[0])
```

```
xy=x Odds Ratio: 0.140407
xy=y Odds Ratio: inf
xy=x=y Odds Ratio: 2.994e+80
xy=x+y Odds Ratio: inf
suppress Odds Ratio: inf
the value used for the observed fit was -0.4 +/- 0.00506
```

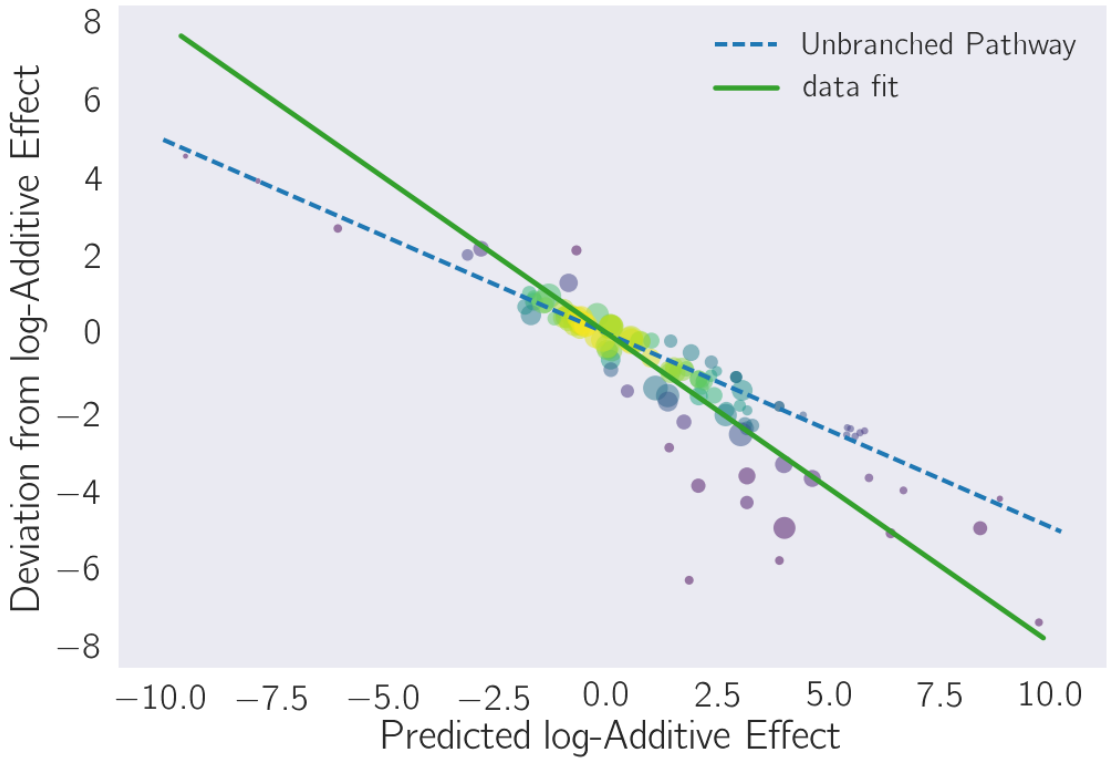
Wow. We can see that all of the models are basically rejected in favor of the free-slope model. Except one. We fail to reject the model $XY = X$. In this case, X is *egl-9*, and Y is *vhl-1*. This means that the parameter-free prediction that *egl-9* is epistatic over *vhl-1* is a preferred model over the free-slope model. Genetics. Works.

6 Measuring suppressive epistasis

6.1 hif-1 suppresses egl-9

This is self explanatory, but let's repeat the analysis above for *egl-9* and *hif-1*.

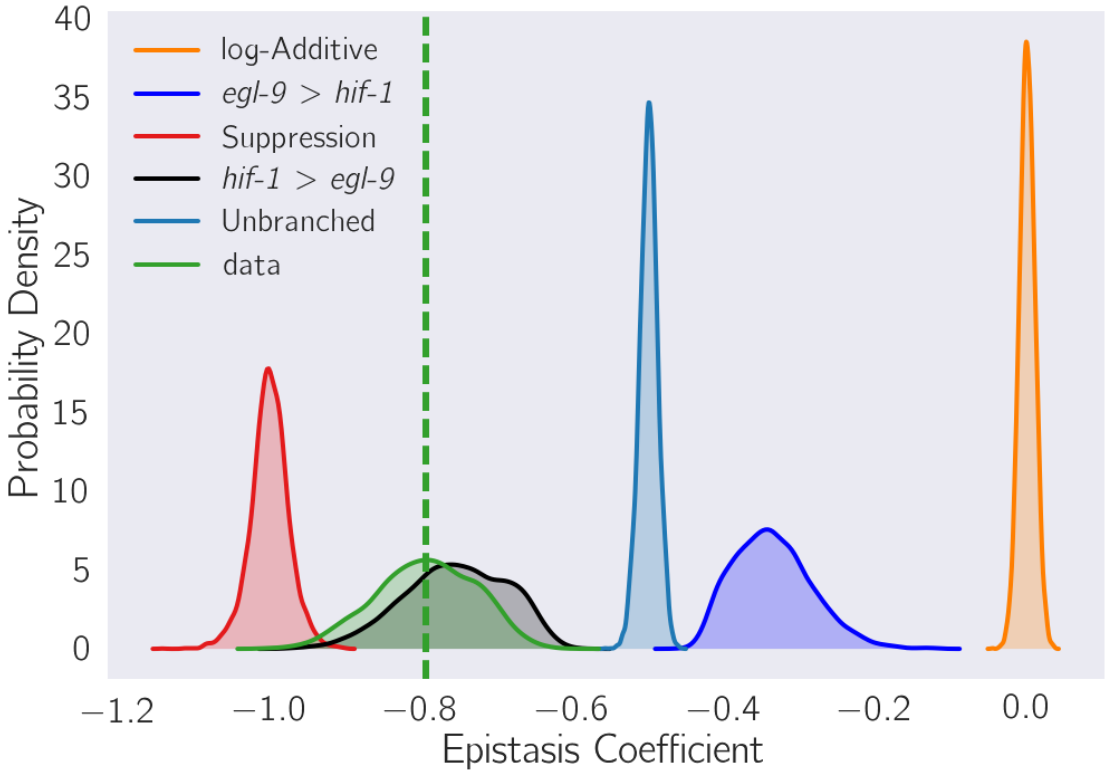
```
In [19]: letter1 = 'b'
         letter2 = 'c'
         double = genvar.double_mapping[letter1 + letter2]
         x, y, xy, ax = epi.make_epiplot([letter1, letter2], double, tidy_data)
         plt.savefig('../output/epistasis{0}{1}.svg'.format(genvar.mapping[letter1], genvar.ma
                     bbox_inches='tight')
```



We can immediately notice a couple of things. First, there are a LOT less points here (around 50) as opposed to the previous plot (around 330). Secondly, they form a line that has a slope $< -\frac{1}{2}$. Both of these are characteristic of a gene that is under strong suppression. Let's see what our models will predict when we simulate them.

```
In [20]: s = epi.calculate_all_bootstraps(letter1, letter2, double, tidy_data, nsim=5000)
```

```
In [22]: ax = epi.plot_bootstraps(letter1, letter2, s, cumulative=False, shade=True)
         plt.xlabel('Epistasis Coefficient')
         plt.ylabel('Probability Density')
         plt.savefig('../output/supp_figures/supplementary_figure_2.svg'.format(genvar.mapping
                     bbox_inches='tight')
```



We notice a couple of things from this graph. First, all of the predictions are considerably wider. This is the result of the considerably smaller number of points in this dataset. The observed fit distribution overlaps significantly with a model where *hif-1* is epistatic over *egl-9* (black curve), but also with the model of complete suppression. Let's take a look at the OR before we can decide what's going on.

```
In [23]: alpha_eglhif = {'xy=x': s['xy=x'].mean(),
                         'xy=y': s['xy=y'].mean()
```

```
                    }

        actual = epi.ODR([x,y], xy, epistasis='actual')
        X = x.b.values + y.b.values
        Y = xy.b.values - X
        wdev = np.sqrt(x.se_b.values**2 + y.se_b.values**2 + xy.se_b.values**2)
        model_selection(X, Y, wdev, alpha=alpha_eglhif, guess=actual.beta[0])
```

```
xy=x Odds Ratio: 5.39871e+254
xy=y Odds Ratio: 0.000370642
xy=x=y Odds Ratio: 4.66488e+93
xy=x+y Odds Ratio: inf
suppress Odds Ratio: 1.86065e+79
the value used for the observed fit was -0.76 +/- 0.0123
```

In this case, we reject all of the models. If we really wanted to select a model, we would say that $XY = Y$ is the one that maximizes the probability of observing the data. The second most likely model is the complete suppression model. Well, this matches intuition. In this case, I am not offended by our inability to select an OR. We had very few data points.

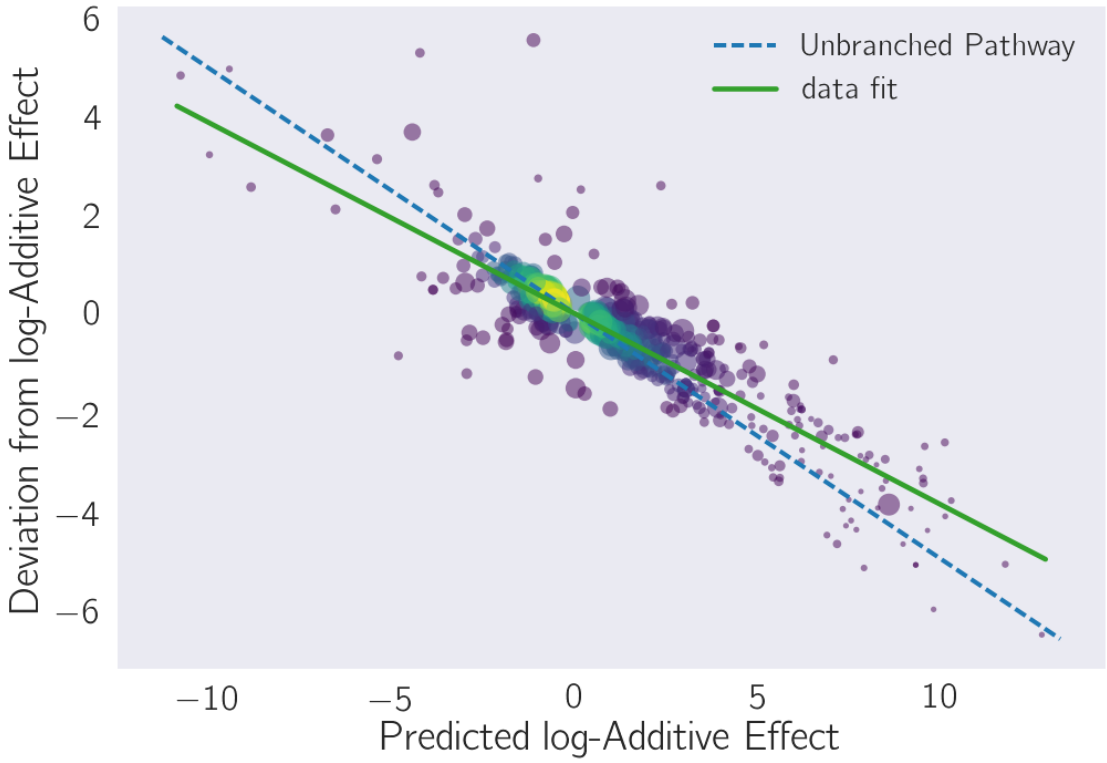
7 Transitivity in transcriptomes

In theory, if two genes are acting through a linear pathway, then both genes should have identical transcriptomes. If they are truly equal, we should be able to substitute one transcriptome for another for any computation we are performing. It follows that we should be able to substitute transcriptomes to predict and/or measure epistasis between two genes if we have a third gene that is related via a linear pathway.

7.1 Predicting epistasis between *egl-9* and *vhl-1* using the *rhy-1* transcriptome

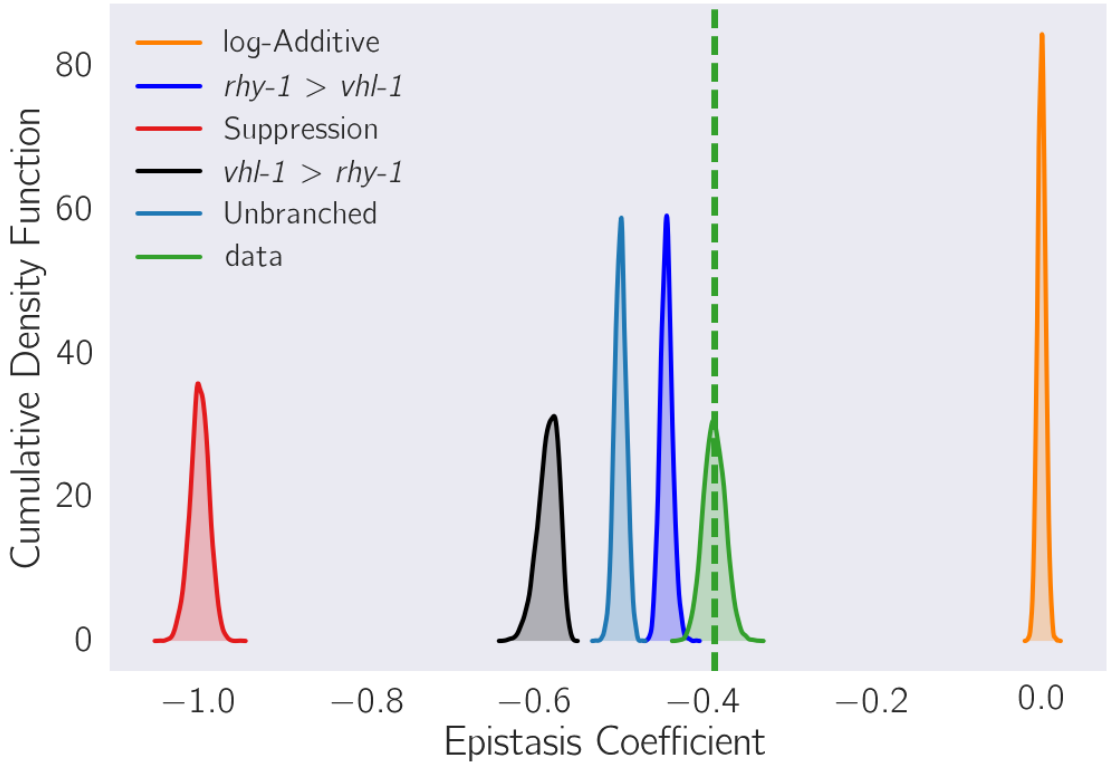
Recall that *rhy-1* genetically activates *egl-9*. If transcriptomes are transitive, then we could use the *rhy-1* transcriptome to predict the epistasis coefficient between *egl-9* and *vhl-1*. We could also use it to "measure" the transcriptome-wide coefficient by substituting *rhy-1* instead of the *egl-9* mutant.

```
In [24]: letter1 = 'e'
         letter2 = 'd'
         double = genvar.double_mapping['b' + letter2]
         x, y, xy, ax = epi.make_epiplot([letter1, letter2], double, tidy_data)
```




```
In [25]: s = epi.calculate_all_bootstraps(letter1, letter2, double, tidy_data, nsim=5000)

In [26]: ax = epi.plot_bootstraps(letter1, letter2, s, cumulative=False, shade=True)
         plt.xlabel('Epistasis Coefficient')
         plt.ylabel('Cumulative Density Function')
         plt.savefig('../output/kde-epistasis{0}{1}.svg'.format(genvar.mapping[letter1], genva
                     bbox_inches='tight')
```




```
In [27]: alpha = {'xy=x': s['xy=x'].mean(),
                  'xy=y': s['xy=y'].mean()
                 }

         actual = epi.ODR([x,y], xy, epistasis='actual')
         print('The observed slope from the ODR regression was {0:.2g}'.format(actual.beta[0]))
         X = x.b.values + y.b.values
         Y = xy.b.values - X
         wdev = np.sqrt(x.se_b.values**2 + y.se_b.values**2 + xy.se_b.values**2)
         model_selection(X, Y, wdev, alpha=alpha, guess=actual.beta[0])
```

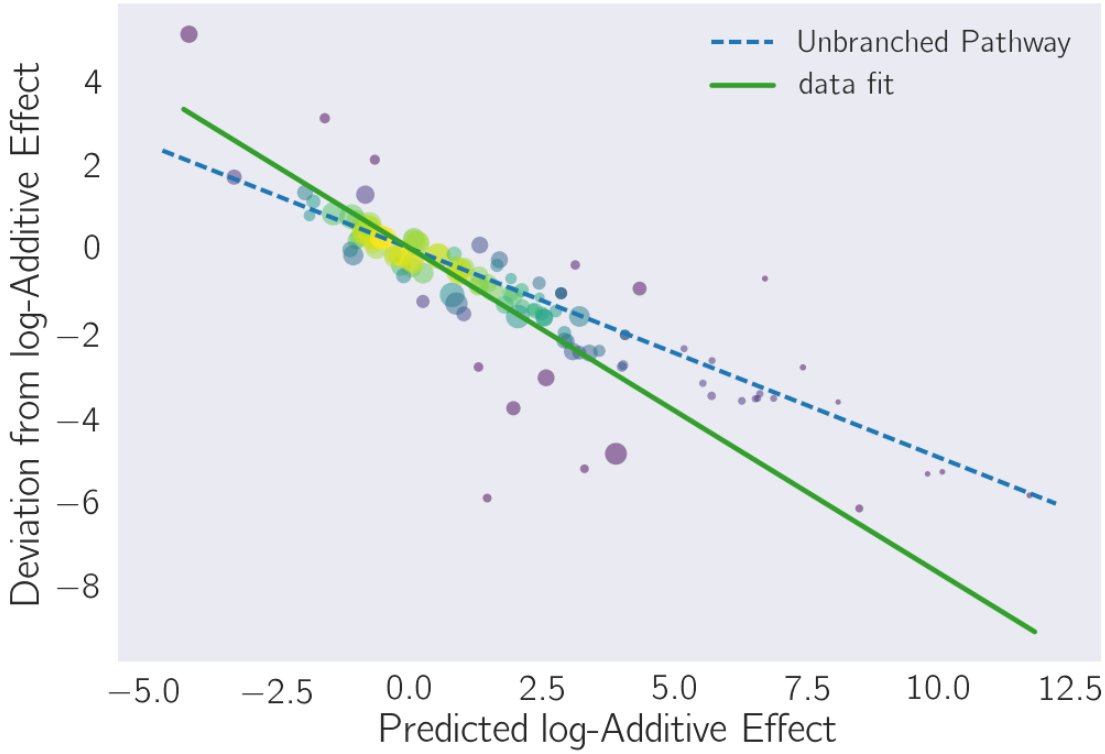
```
The observed slope from the ODR regression was -0.39
xy=x Odds Ratio: 6.04784e+29
xy=y Odds Ratio: 1.51975e+302
xy=x=y Odds Ratio: 5.47657e+102
xy=x+y Odds Ratio: inf
suppress Odds Ratio: inf
the value used for the observed fit was -0.376 +/- 0.00559
```

If we use ODR to predict the epistasis coefficient, we would "measure" an epistasis value of -0.38, which agrees with what we obtained with the *egl-9* mutant. However, unlike with the *egl-9* mutant data, the odds ratio test fails to accept any theoretical model. Clearly, there are deviations

that occur between *rhy-1* and *egl-9*. Maybe knocking out *rhy-1* does not fully inactivate *egl-9*. Indeed, this is a good hypothesis. We see that the epistasis models, $XY = X$ and $XY = Y$, both begin to overlap with the unbranched model. This could suggest that knocking out *rhy-1* inhibits the VHL-1-dependent inhibition of HIF-1 by EGL-9, but not the remaining inhibition.

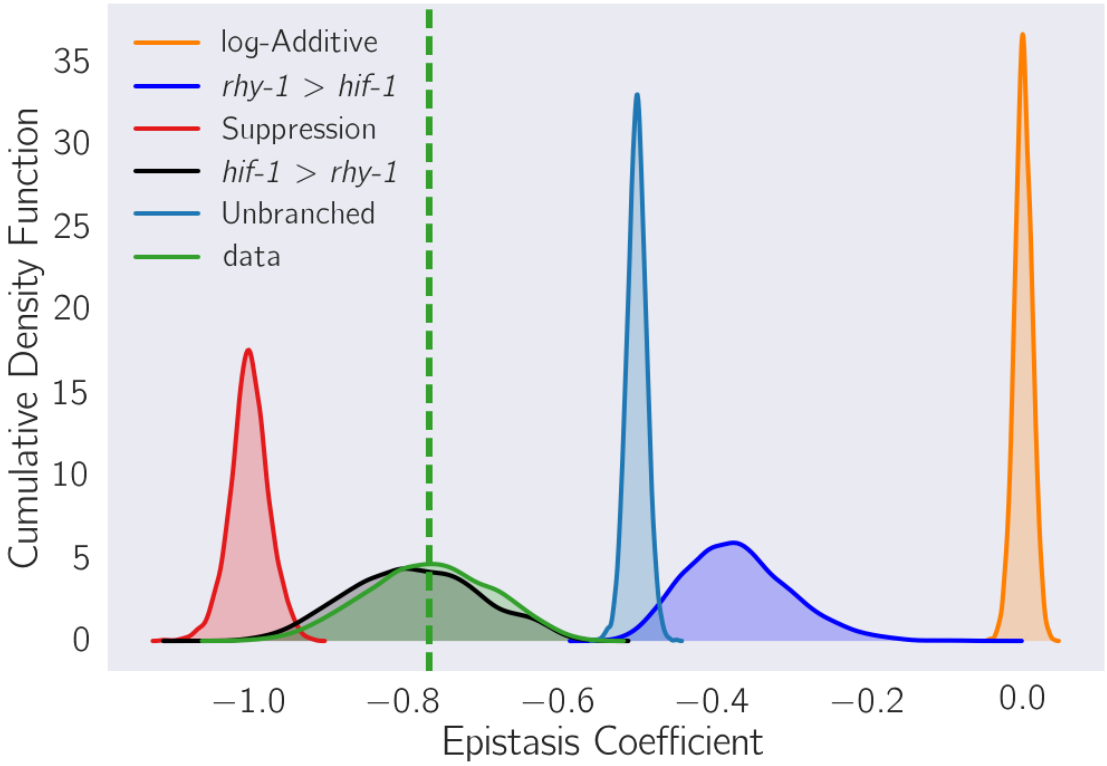
7.2 Predicting epistasis between *egl-9* and *hif-1* using the *rhy-1* transcriptome

```
In [28]: letter1 = 'e'
         letter2 = 'c'
         double = genvar.double_mapping['b' + letter2]
         x, y, xy, ax = epi.make_epiplot([letter1, letter2], double, tidy_data)
```




```
In [29]: s = epi.calculate_all_bootstraps(letter1, letter2, double, tidy_data, nsim=5000)
```

```
In [30]: ax = epi.plot_bootstraps(letter1, letter2, s, cumulative=False, shade=True)
         plt.xlabel('Epistasis Coefficient')
         plt.ylabel('Cumulative Density Function')
         plt.savefig('../output/kde-epistasis{0}{1}.svg'.format(genvar.mapping[letter1], genva
                     bbox_inches='tight')
```




```
In [31]: alpha = {'xy=x': s['xy=x'].mean(),
                  'xy=y': s['xy=y'].mean()
                 }

         actual = epi.ODR([x,y], xy, epistasis='actual')
         X = x.b.values + y.b.values
         Y = xy.b.values - X
         wdev = np.sqrt(x.se_b.values**2 + y.se_b.values**2 + xy.se_b.values**2)
         model_selection(X, Y, wdev, alpha=alpha, guess=actual.beta[0])
```

```
xy=x Odds Ratio: 5.38328e+150
xy=y Odds Ratio: 15.9081
xy=x=y Odds Ratio: 2.11803e+58
xy=x+y Odds Ratio: inf
suppress Odds Ratio: 4.23636e+88
the value used for the observed fit was -0.725 +/- 0.0133
```