<| Main Contents (Index.ipynb) |>

Appendix: The computing Miniproject

Contents

In [53]:

```
# Some imports to explore the datasets in Python
import pandas as pd
import scipy as sc
import matplotlib.pylab as pl
import seaborn as sns # You might need to install this (e.g., sudo pip install
seaborn)
```

We have talked a lot about workflows and confronting models with data. It's time to do something concrete with

all the techniques you have been learning.

The CMEE Miniproject gives you an opportunity to try the "whole nine yards" of developing and implementing a workflow and delivering a "finished product" — where you ask and answer a scientific question in biology (potentially involving multiple sub-questions/hypotheses). It will give you an opportunity to perform a "dry run" of executing your actual dissertation project, and you may use it to trial some of the techniques and/or explore some of the data/theory you might use in your Dissertation project.

Objectives

The general question you will address is: *What mathematical model best fits an empirical dataset?*

You may think of this as testing a set of alternative hypotheses — every alternative hypothesis is nothing but an alternative model to describe an observed phenomenon, as you will have learned in the lectures on model fitting.

The Project

You will choose a dataset and set of alternative models from the options provided to you.

Please read the papers in the Readings section — these will help orient you in the right direction for tackling your miniproject.

The Miniproject must satisfy the following criteria (and follow the accompanying guidelines):

1. It should employ all the biological computing tools you have learned so far: shell (bash) scripting, git, LaTeX, R, and Python. Using these tools, you will build a workflow that starts with the data and ends with a written report (in LaTeX). How you choose the different tools (e.g., Python vs R) is your choice. This is part of the assessment.
2. *At least* two different models (hypotheses) must be fitted to the data. The models should be fitted and selected using an appropriate method. Specifically, irrespective of the problem/dataset you choose (see below), use Nonlinear Least Squares (NLLS) to fit ≥ 2 alternative models to data, followed by model selection using AIC and BIC (read the Johnson and Omland 2005 paper). You may choose additional means for model comparison/selection beyond these.*
3. The project should be fully reproducible. Write a script that "glues" the workflow together and runs it, from data processing to model fitting to plotting (e.g., in R) to compilation of the LaTeX written report (*More detailed instructions on report below*). Look back at the TheMulQuaBio to see how you would run the different components. For example, we have covered how to run R and compile $LaTeX$ using the `subprocess` module in Python. The assessor should be able to run just this script to get everything to work without errors. Use Python or to write this main script. If using bash, call it `run_MiniProject.sh` and if using Python, called it `run_MiniProject.py` .

You will be given lectures and practicals on model fitting before you start on your Miniproject.

The Report

The report should,

- be written in LaTeX using the article document class, in 11pt (any font will do, within reason!).
- be double-spaced, with *continuous* line numbers.
- have a title, author name with affiliation and wordcount (next point) on a separate title page.
- have an introduction with objectives of the study, and appropriate additional sections such as methods, data, results, discussion, etc.

- should contain in the *Methods* a sub-section called "Computing tools" which states briefly how each of the three scripting language (bash, R, Python) and what packages within them were used and a justification of why.
- must contain $\leq$3500 words *excluding the contents of the title page, references, and Figure or Table captions+legends*; there should be a word count at the beginning of the document (typically using the `texcount` package).
- have references properly cited in text and formatted in a list using bibtex.

For the writeup, you probably should read the *general* (*not* word count, formatting etc.) dissertation writing guidelines given in the Silwood Masters Student Guidebook.

Submission

Add, commit and push all your work to your bitbucket repository using a directory called `MiniProject` at the same level as the Week1, Week2 etc. directories, by the Miniproject deadline given in your course guidebook.

At this stage you are not going to be told you how to organize your project — that's part of the marking criteria (see next section).

Marking criteria

Equal weightage will be given to the code+workflow and writeup components — each component will be marked to a max of 100 pts and then rescaled to a single mark / 100 using equal weightage

The assessor will be looking for the following while assessing your submission:

- A well-organized project where code, results, data, etc., are easy to locate, inspect, and use. In the project's README also include:
  - Any dependencies or special packages the user/marker should be aware of
  - What each package you used is for
  - Version of each language used
- A project that runs smoothly, without any errors once the appropriate script (i.e., `run_MiniProject.py` or `run_MiniProject.sh`) is called.
- A report that contains all the components indicated above in "The Report" subsection, with some original thought and synthesis in the **Introduction** and **Discussion** sections.
- Quality of the presentation of the graphics and tables in your report, as well as any plots showing model fits to the data.
- The marking criteria you may refer to is the summative marking criteria (./MARKING_CRITERIA.pdf).

The Model Fitting Problems

You can pick from one of the following three options.

Thermal Performance Curves

The Question

How well do different mathematical models, e.g., based upon biochemical (mechanistic) principles vs. phenomenological ones, fit to the thermal responses of metabolic traits?

This is currently a "hot" (no pun intended!) topic in biology. On the *ecological side*, because the temperature-dependence of metabolic rate sets the rate of intrinsic $r_{\max}$ (papers by Savage et al., Brown et al.) as well as interactions between species, it has a strong effect on population dynamics. In this context, note that 99.9% of life on earth is ectothermic! On the *evolutionary side*, the temperature-dependence of fitness and species interactions also means that warmer environments may have stronger rates of evolution. This may be compounded by the fact that mutation rates may also increase with temperature (papers by Gillooly et al.).

The Data

The dataset is called `ThermRespData.csv`. It contains a subset of the full "BioTraits" database. This subset contains hundreds of "thermal responses" for growth, respiration and photosynthesis rates in plants and bacteria (both aquatic and terrestrial). These data were collected through lab experiments across the world, and compiled by various people over the years. The field names are defined in a file called `BiotraitsTemplateDescription.pdf`, also in the `data` directory. The two main fields of interest are `OriginalTraitValue` (the trait values responding to temperature), and `ConTemp` (the temperature). Individual thermal response curves can be identified by `ID` values --- each `ID` corresponds to one thermal performance curve.

Let's have a look at the data:

In [54]:

```
data = pd.read_csv("../data/ThermRespData.csv")
print("Loaded {} columns.".format(len(data.columns.values)))
```

```
Loaded 78 columns.

/usr/local/lib/python3.5/dist-packages/IPython/core/interactiveshell.
py:2728: DtypeWarning: Columns (8,9,37,41,44,55,58,59,63,71) have mix
ed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

In [55]:

```
data.head()
```

Out[55]:

	Input	ID	OrignalTraitName	OriginalTraitDef	StandardisedTraitName	StandardisedTraitDef	OriginalTraitVa
0	Richard	1	photosynthetic co2 assimilation	NaN	net photosynthesis rate	NaN	9.876
1	Richard	1	photosynthetic co2 assimilation	NaN	net photosynthesis rate	NaN	11.743

In [56]:

```
print(data.columns.values)
```

```
['Input' 'ID' 'OrignalTraitName' 'OriginalTraitDef' 'StandardisedTrai
tName'
 'StandardisedTraitDef' 'OriginalTraitValue' 'OriginalTraitUnit'
 'OriginalErrorPos' 'OriginalErrorNeg' 'OriginalErrorUnit'
 'StandardisedTraitValue' 'StandardisedTraitUnit' 'StandardisedErrorP
os'
 'StandardisedErrorNeg' 'StandardisedErrorUnit' 'Replicates' 'Consume
r'
 'ConCommon' 'Habitat' 'Location' 'LocationType' 'LocationDate'
 'CoordinateType' 'Latitude' 'Longitude' 'ConThermy' 'ConStage' 'ConS
ize'
 'ConSizeUnit' 'ConSizeType' 'ConDenValue' 'ConDenUnit' 'ConTemp'
 'ConTempUnit' 'ConTempMethod' 'Labfield' 'Labtemp' 'Labtempunit' 'La
btime'
 'Labtimeunit' 'ArenaValue' 'ArenaUnit' 'AmbientTemp' 'AmbientTempUni
t'
 'AmbientLight' 'AmbientLightUnit' 'Resource' 'ResCommon' 'ResStage'
 'ResThermy' 'ResTemp' 'ResTempMethod' 'ResSize' 'ResSizeUnit'
 'ResDenValue' 'ResDenUnit' 'ResRepValue' 'ResRepUnit' 'ObsTimeValue'
 'ObsTimeUnit' 'EquilibTimeValue' 'EquilibTimeUnit' 'AcclimFixTemp'
 'AcclimFixTempUnit' 'AcclimFixTempDur' 'AcclimFixTempDurUnit'
 'AcclimVarTemp' 'AcclimVarTempUnit' 'AcclimVarTempDur'
 'AcclimVarTempDurUnit' 'LabGrowthTemp' 'LabGrowthTempUnit' 'LabGrowt
hDur'
 'LabGrowthDurUnit' 'Citation' 'FigureTable' 'Notes']
```

In [57]:

```
print(data.OriginalTraitUnit.unique()) #units of the response variable
```

```
['micromol m^-2 s^-1' 'mg o2 (10 minutes^-1)' 'micromol o2 dm^-2 min^
-1'
 'mg co2 dm^-2 h^-1' 'micromol co2 mg(Chl)^-1 h^-1' 'micro (day^-1)'
 '% of maximum' 'mg co2 g(FW)^-1 h^-1' 'ng co2 cm^-2 s^-1' 'gh^-1 m^-
2'
 'nmol g^-1 s^-1' 'mg co2 g^-1 h^-1' 'nmol co2 cm^-2 s^-1'
 'micromol co2 m^-2 s^-1' 'micromol o2 mg(Chl)^-1 h^-1'
 'micromol co2 kg^-1 s^-1' 'mg co2 g(DW)^-1 h^-1'
 'micromol o2 g(DW)^-1 min^-1' 'mg C g(DW)^-1 h^-1' 'mg o2 g(DW)^-2 h
^-1'
 'microg co2 m^-2 s^-1' 'micromol o2 g^-1 h^-1'
 'microg co2 g(DW)^-1 min^-1' 'microl o2 g(DW)^-1 (10 minutes)^-1'
 'mg co2 m^-2 s^-1' 'micromol o2 mg(Chl)^-1 min^-1'
 'mg o2 10^-9 cells hour^-1' 'mmol e^- [mg chl a]^-1 h^-1'
 'microg o2 mg(Chl)^-1 min^-1' 'dpm 14C per 10^3 cells' 'mmol kg^-1 h
^-1'
 'microg C cm^-2 h^-1' 'g co2 m^-2 h^-1' 'microl co2 m^-2 s^-1'
 'nmol cm^-2 s^-1' 'micromol co2 g(FW)^-1 h^-1' '?'
 'micromol mg(Chl a)^-1 h^-1' 'micromol g(DW)^-1 h^-1' 'g o2 cm^-2 h^
-1'
 'microg o2 cm^-2 h^-1' '%' 'micromol o2 mg(Chl a)^-1 h^-1'
 'micromol co2 g^-1 min^-1' 'cpm (microg protein)^-1'
 'microg o2 g(DW)^-1 min^-1' 'microl o2 mg(DW)^-1 h^-1'
 'fmol C cell^-1 min^-1' 'microl o2 mg(Chl)^-1 h^-1'
 'microl o2 g(DW)^-1 h^-1' 'mm^3 o2 cm^-2 h^-1' 'nmol o2 g(DW)^-1 s^-
1'
 'micromol o2 g(DW)^-1 h^-1' 'microg co2 dm^-2 (both sides) min^-1'
 'mg o2 g^-1 h^-1' 'mg c g(DW)^-1' 'microl o2 g(FW)^-1 h^-1' 'microW
mg^-1'
 'nmol g(DW)^-1 s^-1' 'nmol co2 g(DW)^-1 s^-1' 'micromol g^-1 h^-1'
 'microl o2 g(FW)^-1 min^-1' 'nmol co2 g^-1 s^-1' 'micromol g^-1 s^-
1'
 'pgC cell^-1 h^-1' 'pg C pg(chl a)^-1 h^-1' 'micromol O2 g^-1 h^-1'
 'micromol o2 g(FW)^-1 h^-1' nan 'mg o2 g(DW)^-1 h^-1'
 'nmol CO2 cm^-2 s^-1' 'micromol o2 m^-2 s^-1' '% of max'
 'micromol dm^-2 min^-1' 'microg C g^-1 h^-1' 'mg o2 dm^-2 h^-1'
 'mg co2 h^-1 g^-1' 'g C g(Chl a)^-1 h^-1']
```

In [58]:

```
print(data.ConTempUnit.unique()) #units of the independent variable
```

```
['celsius' 'Celsius']
```

In [59]:

```
print(data.ID.unique()) #units of the independent variable
```

```
 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 28
7 288
 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 30
5 306
 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 32
3 324
 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 34
1 342
 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 35
9 360
 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 37
7 378
 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 39
5 396
 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 41
3 414
 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 43
1 432
 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 44
```

In [60]:

```
data_subset = data[data['ID']==110]
data_subset.head()
```

Richard	110	net photosynthesis	NaN	net photosynthesis rate	NaN	4.380516
Richard	110	net photosynthesis	NaN	net photosynthesis rate	NaN	4.891680
Richard	110	net photosynthesis	NaN	net photosynthesis rate	NaN	4.735478

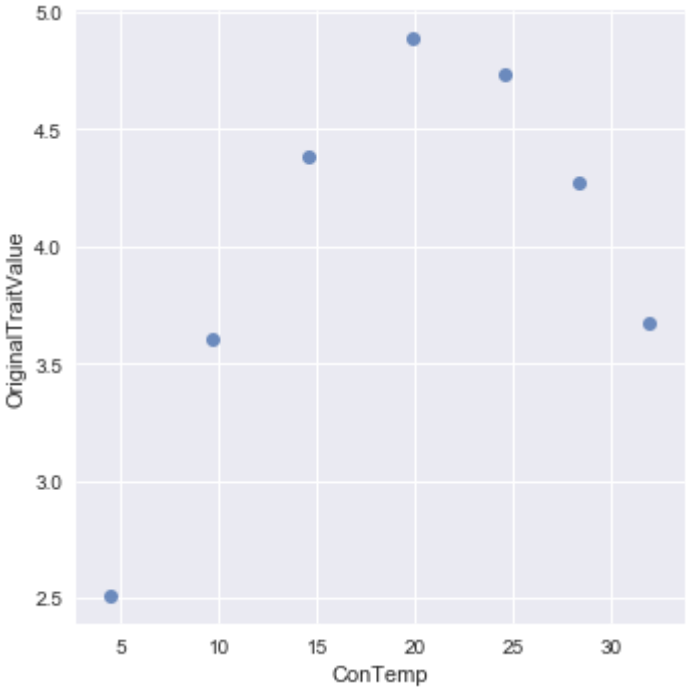
In [61]:

```
sns.lmplot("ConTemp", "OriginalTraitValue", data=data_subset, fit_reg=False) # you
may ignore the warning taht appears below
```

```
/usr/local/lib/python3.5/dist-packages/matplotlib/__init__.py:830: Ma
tplotlibDeprecationWarning: axes.color_cycle is deprecated and replac
ed with axes.prop_cycle; please use the latter.
  mplDeprecation)
```

Out[61]:

```
<seaborn.axisgrid.FacetGrid at 0x7fb3782cc1d0>
```



The Models

All the following parameters and variables are in SI units.

There are multiple models that might best describe these data. The simplest are the general quadratic and cubic polynomial models:

$$B = B_0 + B_1 x + B_2 x^2 \tag{1}$$

$$B = B_0 + B_1 x + B_2 x^2 + B_3 x^3 \tag{2}$$

These are phenomenological models, with the parameters B_0, B_1, B_2 and B_3 lacking any mechanistic interpretation. x is the independent variable (in this case Temperature, T)

Another phenomenological model option is the Briere model:

$$B = B_0 T (T - T_0) \sqrt{T_m - T} \tag{3}$$

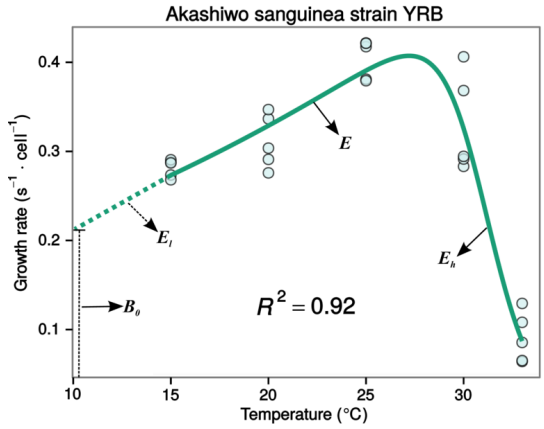
Where T is temperature, T_0 and T_m are the minimum and maximum feasible temperatures for the trait (below or above which the traits goes to zero), and B_0 is a normalization constant.

In contrast, the Schoolfield model (Schoolfield et al 1981) is a mechanistic option that is based upon thermodynamics and enzyme kinetics:

$$B = \frac{B_0 e^{\frac{-E}{k}(\frac{1}{T} - \frac{1}{283.15})}}{1 + e^{\frac{E_l}{k}(\frac{1}{T_l} - \frac{1}{T})} + e^{\frac{E_h}{k}(\frac{1}{T_h} - \frac{1}{T})}} \tag{4}$$

Please also have a look at the Delong et al 2017 paper, which lists this and other mechanistic TPC models (see Readings). You may choose additional models listed in that paper for comparison, if you want.

Here, k is the Boltzmann constant (8.617×10^{-5} eV $\cdot$ K^{-1}), B the value of the trait at a given temperature T (K) (K = ˚C + 273.15), while B_0 is the trait value at 283.15 K (10˚C) which stands for the value of the growth rate at low temperature and controls the vertical offset of the curve. E_l is the enzyme's low-temperature de-activation energy (eV) which controls the behavior of the enzyme (and the curve) at very low temperatures, and T_l is the at which the enzyme is 50% low-temperature deactivated. E_h is the enzyme's high-temperature de-activation energy (eV) which controls the behavior of the enzyme (and the curve) at very high temperatures, and T_h is the at which the enzyme is 50% high-temperature deactivated. E is the activation energy (eV) which controls the rise of the curve up to the peak in the "normal operating range" for the enzyme (below the peak of the curve and above T_h).



Example of the Sharpe-Schoolfield eqn, that is, 1 and 2, . 4) fitted to the thermal response curve of a (replaceiological trait. < /figcaptio x with resource abundance.>

In many cases, a simplified Schoolfield model would be more appropriate for thermal response data, because low temperature inactivation is weak, or is undetectable in the data because low-temperature measurements were not made.

$$B = \frac{B_0 e^{\frac{-E}{k}(\frac{1}{T} - \frac{1}{283.15})}}{1 + e^{\frac{E_h}{k}(\frac{1}{T_h} - \frac{1}{T})}} \tag{5}$$

In other cases, a different simplified Schoolfield model would be more appropriate, because high temperature inactivation was not detectable in the data because measurements were not made at sufficiently high temperatures:

$$B = \frac{B_0 e^{\frac{-E}{k}(\frac{1}{T} - \frac{1}{283.15})}}{1 + e^{\frac{E_l}{k}(\frac{1}{T_l} - \frac{1}{T})}} \quad (6)$$

Note that the cubic model (Equation 2) has the same number of parameters as the the reduced Schoolfield models (eq. 5 & 6). Also, the temperature parameter (T) of the cubic model (Equation 2) is in ˚C, whereas the Temperature parameter in the Schoolfield model is in K.

Functional Responses

The Question

How well do different mathematical models, e.g., based upon foraging theory (mechanistic) principles vs. phenomenological ones, fit to functional responses data across species?

In ecological parlance, a functional response is the relationship between a consumer's (e.g., predator) biomass consumption rate and abundance of the target resource (e.g., prey). Functional responses arise from fundamental biological and physical constraints on consumer-resource interactions (e.g., Holling 1959, Pawar et al, 2012), and determine the rate of biomass flow between species in ecosystems across the full scale of sizes, from the smallest (e.g., microbes) to the largest (e.g., blue whales). Functional responses also play a key sole in determining the stability (responses to perturbations) of the food webs that underpin ecosystems.

The Data

The dataset is called `CRat.csv`. It contains measurements of rates of consumption of a single resource (e.g., prey, plants) species' by a consumer species (e.g., predators, grazers). These data were collected through lab and field experiments across the world. The field names are defined in a file called `BiotraitsTemplateDescription.pdf`, also in the `data` directory. The two main fields of interest are `N_TraitValue` (The number of resources consumed per consumer per unit time), and `ResDensity` (the resource abundance). Individual functional response curves can be identified by `ID` values --- each `ID` corresponds to one curve. Or you can reconstruct them as unique combinations of `Citation` (where the functional response dataset came from), `ConTaxa` (consumer species ID), `ResTaxa` (resource species ID).

Let's have a look at the data:

In [11]:

```
data = pd.read_csv("../data/CRat.csv")
print("Loaded {} columns.".format(len(data.columns.values)))
```

```
Loaded 68 columns.
```

In [12]:

```
data.head()
```

Out[12]:

	ID	DataType	ORIGINAL_TraitID	ORIGINAL_TraitDefinition	TraitValue	TraitUnit	N_TraitVal
0	39835	new	Resource Consumption Rate	The number of resource consumed per number of ...	8.67993	Individual/(Individual*120 mins)	0.0012
1	39835	new	Resource Consumption Rate	The number of resource consumed per number of ...	7.66727	Individual/(Individual*120 mins)	0.0010
2	39835	new	Resource Consumption Rate	The number of resource consumed per number of ...	8.67993	Individual/(Individual*120 mins)	0.0012

In [13]:

```
print(data.columns.values)
```

```
['ID' 'DataType' 'ORIGINAL_TraitID' 'ORIGINAL_TraitDefinition' 'Trait
Value'
 'TraitUnit' 'N_TraitValue' 'ConTaxa' 'ResTaxa' 'ConTaxaStage'
 'ResTaxaStage' 'Con_ForagingMovement' 'Con_RESDetectionDimensionalit
y'
 'Res_ForagingMovement' 'Res_CONDetectionDimensionality' 'CON_MASS_va
lue'
 'RES_MASS_value' 'ResArenaSize_SI_UNIT' 'ResDensity_SI_VALUE'
 'ResDensityUnit' 'ResDensity' 'ResArenaSize_SI_VALUE' 'ConTemp' 'Res
Temp'
 'ResReplaceRate' 'ResReplaceUnit' 'ResReplace' 'TraitSIValue'
 'TraitSIUnit' 'N_TraitID' 'N_TraitConversion' 'N_CONVERTED' 'N_Trait
Unit'
 'Original_ErrorValue' 'Original_ErrorValueUnit' 'Replicates' 'ConTax
on'
 'ConStage' 'ConCommon' 'Con_MovementDimensionality' 'Con_Thermy'
 'Res_MovementDimensionality' 'ResTaxon' 'ResStage' 'ResCommon'
 'Res_Thermy' 'Habitat' 'LabField' 'ObservationtimeSI'
 'ObservationtimeSIUnits' 'ConStarvationTimeSI' 'ConStarvationTimeSIU
nits'
 'ConArenaSiz_VALUE' 'ConArenaSize_UNIT' 'ConDensity' 'ConDensityUni
t'
 'ConDensity_SI_VALUE' 'ConDensityConst' 'ConSizType' 'CON_ORIGINAL_v
alue'
 'CON_ORIGINAL_unit' 'Con_Siz_reference' 'ResSizType' 'RES_ORIGINAL_v
alue'
 'RES_ORIGINAL_unit' 'Res_Siz_reference' 'Citation' 'FigureTable']
```

In [14]:

```
print(data.TraitUnit.unique()) #units of the response variable
```

```
['Individual/(Individual*120 mins)' 'Individual/(Individual*90 mins)'
 'individual / (individual * second)'
 'individual / (5 individual * 24 hrs)'
 'individual / (1 individual * 3 hrs)'
 'individual / (1 individual * 5 hrs)'
 'individual / (1 individual * 8 hrs)'
 'individual / (1 individual * 6 hrs)'
 'individual / (1 individual * 24 hrs)'
 'individual / (1 individual *12 hrs)'
 'individual / (1 individual *24 hrs)' 'individual / (1 individual *2
hrs)'
 'milligram (body mass - dry) / (sqrt (gram (body mass - wet)) * 1 mi
nute)'
 'individual / (1 individual *3 hrs)'
 'cubic micrometer (yeast cell wet volume) / (gram (body mass - wet)
* 1 hour)'
 'individual / (1 individual *30 mins)' 'individual / (1 individual *
day)'
 'individual / (1 individual * hr)'
 'gram (biomass - dry) / (individual * 1 minute)'
 'Individual/(Individual*1 mins)' 'Individual/(Individual*1 hr)'
 'Individual/(Individual*1 day)'
 'gram (biomass - wet) / (individual * 3 hrs)'
 'Individual/(Individual * 1 hr)' 'Individual/(Individual*7 days)'
 'gram (biomass - wet) / (individual * 22.5 minutes)'
 'Individual/(Individual * 8 hr)' 'Individual/(Individual * day)'
 'Individual/(Individual * 106 days)'
 'milligram (biomass - dry) / (individual * 1 hr)'
 'Individual/(Individual * 12 hrs)' 'Individual/(Individual * 1 day)'
 'Individual/(Individual * 2 day)' 'Individual/(Individual * 3 day)'
 'Individual/(Individual * 15 mins)'
 'microgram (Carbon) / (individual * 1 hr)'
 'cubic micrometer (cell wet biovolume) / (gram (body mass - wet) * 1
hour)'
 'Individual/(Individual * 3 mins)' 'Individual/(Individual*1 d)'
 'nanogram (chlorophyll a weight) / (milligram (body mass - dry) * 1
hour) '
 'Individuals / (milligram (body mass - dry) * 1 hour) '
 'individual / (1 individual * 1 day)'
 'individual / (1 individual * 1 hour)'
 'individual / (1 individual * 15 minutes)'
 'individual / (1 individual * 4 hour)'
 'nanogram (Carbon) / (individual * 1 hr)'
 'individual / (1 individual * 1 minute)'
 'individual / (1 individual * 1 second)' 'Individual/(Individual*20
mins)'
 'Individual/(Individual * second)' 'Individual/(Individual*1 hour)'
 'Individual/(Individual*111 day)' 'Individual/(Individual*15 min)'
 'Individual/(Individual*1 min)' 'Individual/(Individual*12 hour)'
 'Individual/(Individual*168 hour)' 'Individual/(Individual*12 day)'
 'individual / (1 individual * 24 hour)'
 'individual / (1 individual * 6 hour)'
 'individual / (1 individual * 3 hour)'
 'individual / (1 individual * 1 hr)'
 'individual / (1 individual * 15 hrs)'
 'individual / (1 individual *5 mins)']
```

In [15]:

```
print(data.ResDensityUnit.unique()) #units of the independent variable
```

```
['Individuals per arena' 'g (body - dry) per square meter'
 'cubic micrometer (yeast wet cell biovolume) per milliliter'
 'Individuals per liter' 'Individuals per cubic centimeter'
 'kg (body - dry) per hectare' 'Individuals per square meter'
 'Individuals per microliter' 'g (body - wet) per square meter'
 'individuals per arena' 'individuals per square km'
 'mg (body - dry) per square cm' 'individuals per Liter'
 'mg (Carbon) per liter'
 'cubic micrometer (wet cell biovolume) per milliliter'
 'individuals per .15 ha' 'microgram (Carbon) per liter'
 'Individuals per ml' 'microgram (chlorophyll a weight) per liter'
 'Individuals per milliliter' 'Individuals per 1 ha'
 'Individuals per cubic meter' 'Individuals per 100 ha'
 'individual per milliliter (1)' 'individual per arena (1)'
 'individual per liter (1)']
```

In [16]:

```
print(data.ID.unique()) #units of the independent variable
```

```
 39871 39873 39874 39875 39876 39877 39878 39879 39880 39881 39882
39883
 39884 39885 39886 39887 39888 39889 39890 39891 39892 39893 39894
39895
 39896 39897 39898 39899 39900 39901 39902 39903 39904 39905 39906
39907
 39908 39909 39910 39911 39912 39913 39914 39915 39916 39917 39918
39919
 39920 39921 39922 39923 39924 39925 39926 39927 39928 39929 39931
39932
 39933 39934 39935 39936 39937 39938 39939 39940 39941 39942 39943
39944
 39945 39946 39947 39948 39949 39950 39951 39952 39953 39954 39955
39956
 39957 39958 39959 39960 39961 39962 39963 39964 39965 39966 39967
39968
 39969 39970 39971 39972 39973 39974 39975 39976 39977 39978 39979
39980
 39981 39982 39983 39984 39985 39986 39987 39988 39989 39990 39991
```

In [17]:

```
data_subset = data[data['ID']==39982]
data_subset.head()
```

Out[17]:

	ID	DataType	ORIGINAL_TraitID	ORIGINAL_TraitDefinition	TraitValue	TraitUnit	N_TraitValu
1552	39982	new	Resource Consumption Rate	The number of resource consumed per number of ...	15.3020	Individual/(Individual * 1 hr)	0.00425
1553	39982	new	Resource Consumption Rate	The number of resource consumed per number of ...	17.6631	Individual/(Individual * 1 hr)	0.00490

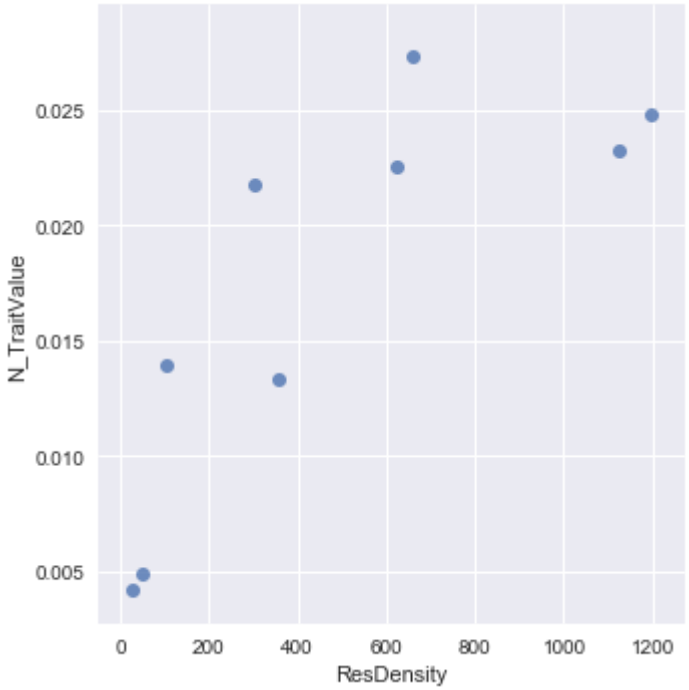
In [18]:

```
sns.lmplot("ResDensity", "N_TraitValue", data=data_subset, fit_reg=False)
```

```
/usr/local/lib/python3.5/dist-packages/matplotlib/__init__.py:830: Ma
tplotlibDeprecationWarning: axes.color_cycle is deprecated and replac
ed with axes.prop_cycle; please use the latter.
  mplDeprecation)
```

Out[18]:

```
<seaborn.axisgrid.FacetGrid at 0x7fb37906a710>
```



The Models

All the following parameters and variables are in SI units.

The fundamental measure of interest (the response variable) is consumption rate (c). This is expressed in terms of biomass quantity or number of individuals of resource consumed *per unit time per unit consumer* (so units of Mass (or Individuals) / Time).

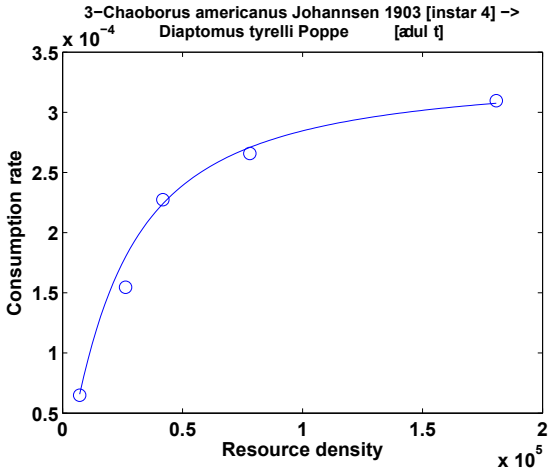
Again, the simplest mathematical models you can use are the phenomenological quadratic and cubic polynomial models, that is eqns. 1 and 2 (replace x with resource abundance).

Then, there is the more mechanistic Holling Type II model (Holling, 1959):

$$c = \frac{ax_R}{1 + hax_R} \tag{7}$$

Here, x_R is resource density (Mass / Area or Volume), a is consumer's search rate (Area or Volume / Time), and h is handling time of the consumer for that resource (time taken to overpower and ingest it).

Below is an example FR curve from the dataset you have been given with the Type II model fitted to it.

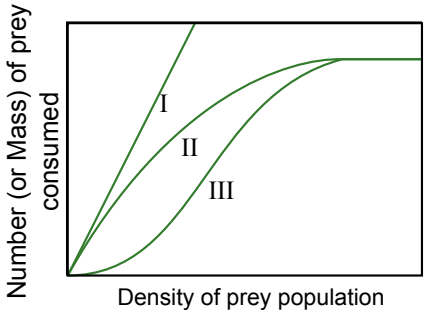


Example of the a Type II model (eqn. 7) fitted to a functional response of a consumer on a resource.

There is also the less-mechanistic "generalized" functional response model:

$$c = \frac{ax_R^{q+1}}{1 + hax_R^{q+1}} \tag{8}$$

Where everything is same as 7, but the additional parameter q (dimensionless) is a shape parameter that allows the shape of the response to be more flexible/variable, from "Type I" to "Type III". This model is less mechanistic because it includes a phenomenological parameter q which does not have a formal biological meaning. Note that if $q = 0$, eqn (8 becomes same as the Type II model (eqn. 7)).



The range of functional responses captured by the generalized functional response model (eqn. 8).

There are other models for functional responses as well (some more mechanistic), that define parameters of the functional response in terms of body size of predator and prey (Pawar et al 2012).

Population Growth

The Question

How well do different mathematical models, e.g., based upon population growth (mechanistic) theory vs. phenomenological ones, fit to functional responses data across species?

Fluctuations in the abundance (density) of single populations may play a crucial role in ecosystem dynamics and emergent functional characteristics, such as rates of carbon fixation or disease transmission. A population grows exponentially while its abundance is low and resources are not limiting (the Malthusian principle). This growth then slows and eventually stops as resources become limiting. There may also be a time lag before the population growth really takes off at the start. We will focus on microbial (specifically, bacterial) growth rates. Bacterial growth in batch culture follows a distinct set of phases; lag phase, exponential phase and stationary phase. During the lag phase a suite of transcriptional machinery is activated, including genes involved in nutrient uptake and metabolic changes, as bacteria prepare for growth. During the exponential growth phase, bacteria divide at a constant rate, the population doubling with each generation. When the carrying capacity of the media is reached, growth slows and the number of cells in the culture stabilises, beginning the stationary phase. Traditionally, microbial growth rates were measured by plotting cell numbers or culture density against time on a semi-log graph and fitting a straight line through the exponential growth phase – the slope of the line gives the maximum growth rate (r_{max}). Models have since been developed which we can use to describe the whole sigmoidal bacterial growth curve.

The Data

The dataset is called `LogisticGrowthData.csv`. It contains measurements of change in biomass or number of cells of microbes over time. These data were collected through lab experiments across the world. The field names are defined in a file called `LogisticGrowthMetaData.csv`, also in the `data` directory. The two main fields of interest are `PopBio` (abundance), and `Time`. Single population growth rate curves can be identified by as unique temperature-species-medium-citation-replicate combinations (concatenate them to get a new string variable that identifies unique growth curves).

Let's have a look at the data:

In [19]:

```
data = pd.read_csv("../data/LogisticGrowthData.csv")
print("Loaded {} columns.".format(len(data.columns.values)))
```

```
Loaded 10 columns.
```

In [22]:

```
print(data.columns.values)
```

```
['X' 'Time' 'PopBio' 'Temp' 'Time_units' 'PopBio_units' 'Species' 'Me
dium'
 'Rep' 'Citation']
```

In [20]:

```
pd.read_csv("../data/LogisticGrowthMetaData.csv")
```

Out[20]:

	Time	Time at which measurement was taken.
0	PopBio	Population or biomass measurement.
1	Temp	Temperature at which the microbe was grown (de...
2	Time_units	Units time is measured in.
3	PopBio_units	Units population or biomass are measured in.
4	Species	Species or strain used.
5	Medium	Medium the microbe was grown in.
6	Rep	Replicate within the experiment.
7	Citation	Citation for the paper in which the study was...

In [21]:

```
data.head()
```

Out[21]:

	X	Time	PopBio	Temp	Time_units	PopBio_units	Species	Mec
0	1	669.879518	0.283276	5	Hours	OD_595	Chryseobacterium.balustinum	
1	2	646.987952	0.283342	5	Hours	OD_595	Chryseobacterium.balustinum	
2	3	622.891566	0.285151	5	Hours	OD_595	Chryseobacterium.balustinum	
3	4	597.590361	0.281746	5	Hours	OD_595	Chryseobacterium.balustinum	
4	5	574.698795	0.273117	5	Hours	OD_595	Chryseobacterium.balustinum	

In [23]:

```
print(data.PopBio_units.unique()) #units of the response variable
```

```
['OD_595' 'N' 'CFU' 'DryWeight']
```

In [24]:

```
print(data.Time_units.unique()) #units of the independent variable
```

```
['Hours']
```

Unlike the previous two datasets there are no ID coulmns, so you will have to infer single growth curves by combining `Species`, `Medium`, `Temp` and `Citation` columns (each species-medium-citation combination is unique):

In [44]:

```
data.insert(0, "ID", data.Species + "_" + data.Temp.map(str) + "_" + data.Medium +
"_" + data.Citation)
```

Note that the `map()` method coverts temperature values to string (`str`) for concatenation.

In [46]:

```
print(data.ID.unique()) #units of the independent variable
```

```
[ 'Chryseobacterium.balustinum_5_TSB_Bae, Y.M., Zheng, L., Hyun, J.
E., Jung, K.S., Heu, S. and Lee, S.Y., 2014. Growth characteristics
and biofilm formation of various spoilage bacteria isolated from fr
esh produce. Journal of food science, 79(10), pp.M2072-M2080.'
 'Enterobacter.sp._5_TSB_Bae, Y.M., Zheng, L., Hyun, J.E., Jung, K.
S., Heu, S. and Lee, S.Y., 2014. Growth characteristics and biofilm
formation of various spoilage bacteria isolated from fresh produce.
Journal of food science, 79(10), pp.M2072-M2080.'
 'Pantoea.agglomerans.1_5_TSB_Bae, Y.M., Zheng, L., Hyun, J.E., Jun
g, K.S., Heu, S. and Lee, S.Y., 2014. Growth characteristics and bi
ofilm formation of various spoilage bacteria isolated from fresh pr
oduce. Journal of food science, 79(10), pp.M2072-M2080.'
 'Pantoea.agglomerans.2_5_TSB_Bae, Y.M., Zheng, L., Hyun, J.E., Jun
g, K.S., Heu, S. and Lee, S.Y., 2014. Growth characteristics and bi
ofilm formation of various spoilage bacteria isolated from fresh pr
oduce. Journal of food science, 79(10), pp.M2072-M2080.'
 'Bacillus.pumilus_5_TSB_Bae, Y.M., Zheng, L., Hyun, J.E., Jung, K.
S., Heu, S. and Lee, S.Y., 2014. Growth characteristics and biofilm
formation of various spoilage bacteria isolated from fresh produce.
Journal of food science, 79(10), pp.M2072-M2080.'
```

These are rather ungainly IDs, so you might want to replace them with numbers!

In [47]:

```
data_subset = data[data['ID']=='Chryseobacterium.balustinum_5_TSB_Bae, Y.M., Zheng, L., Hyun, J.E., Jung, K.S., Heu, S. and Lee, S.Y., 2014. Growth characteristics and biofilm formation of various spoilage bacteria isolated from fresh produce. Journal of food science, 79(10), pp.M2072-M2080.']
data_subset.head()
```

Out[47]:

	ID	X	Time	PopBio	Temp	Time_units	PopBio_units	
0	Chryseobacterium.balustinum_5_TSB_Bae, Y.M., Z...	1	669.879518	0.283276	5	Hours	OD_595	Chryseo
1	Chryseobacterium.balustinum_5_TSB_Bae, Y.M., Z...	2	646.987952	0.283342	5	Hours	OD_595	Chryseo

In [49]:

```
sns.lmplot("Time", "PopBio", data = data_subset, fit_reg = False) # will give warning - you can ignore it
```

The Models

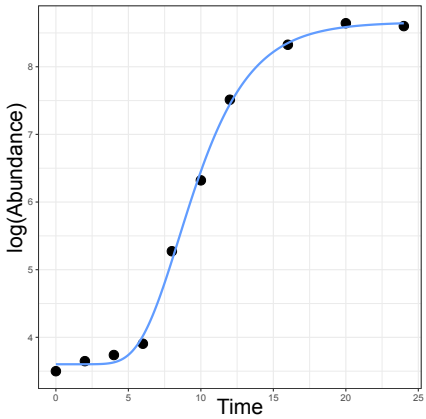
All the following parameters and variables are in SI units.

Yet again, the simplest mathematical models you can use are the phenomenological quadratic and cubic polynomial models, that is eqns. 1 and 2 (replace x with Time). A Polynomial model may be able to capture decline in population size after some maximum value (the carrying capacity) has been reached (the "death phase" of population growth).

A classical somewhat mechanistic model (recall the Modeling Lecture), is the logistic equation:

$$N_t = \frac{N_0 N_{max} e^{rt}}{N_{max} + N_0(e^{rt} - 1)} \tag{9}$$

Here N_t is population size at time t, N_0 is initial population size, r is maximum growth rate (AKA $r_{\max}$), and N_{max} is carrying capacity (commonly denoted by K in the ecological literature).



An example population growth curve dataset to which the modified Gompertz model (Zwietering et. al., 1990) has been fitted.

Other popular models are the modified Gompertz model (Zwietering et. al., 1990), the Baranyi model (Baranyi, 1993), and the Buchanan model (or three-phase logistic model; Buchanan, 1997). The Buchanan model in particular is capable of capturing the lag phase before the population starts growing exponentially, often seen in microbial population growth. An example of fitting these models is available in the model fitting appendix (./Appendix-ModelFitting.ipynb#Population-growth-rate-example).

The modified Gompertz model has been used frequently in the literature to model bacterial growth:

$$N_t = Ae^{-e^{\frac{r_{max}e(t_{lag}-t)}{A}+1}} \tag{10}$$

Here maximum growth rate (r_{max}) is the tangent to the inflection point, t_{lag} is the x-axis intercept to this tangent (duration of the delay before the population starts growing exponentially) and A is the asymptote ($A = \ln\left(\frac{N_{max}}{N_0}\right)$), N_0 is initial cell culture (Population) density, N_{max} is maximum population density (aka "carrying capacity"), and .

The Baranyi model (eq. 11) introduces a new dimensionless parameter h_0 which represents the initial physiological state of the cells. The length of the lag phase is determined by the value of h_0 at inoculation and the post-inoculation environment. Thus the definition of lag is independent from the shape of the growth curve, and the effect of the previous environment is separated from the effects of the present environment. This allows modeling growth without a lag period following inoculation from media favorable to growth to new media also favorable to growth. One formulation given is:

$$N_t = N_0 + r_{max} A_t - \ln\left(1 + \frac{e^{r_{max} A_t} - 1}{e^{N_{max} - N_0}}\right), \tag{11}$$

where:

$$A_t = t + \frac{1}{r_{max}} \cdot \ln\left(\frac{e^{-r_{max}t} + h_0}{1 + h_0}\right). \tag{12}$$

In this model, r_{max} and h_0 can be related to obtain the lag time, t_{lag}:

$$t_{lag} = \frac{\ln\left(1 + \frac{1}{h_0}\right)}{r_{max}}, \tag{13}$$

taking us back to the same four parameters as the Gompertz model.

The Buchanan model, or "three-phase logistic model" is relatively simple:

$$N(t) = \begin{cases} N_0 & \text{if } t \leq t_{lag} \\ N_{max} + r_{max} \cdot (t - t_{lag}) & \text{if } t_{lag} < t < t_{max} \\ N_{max} & \text{if } t \geq t_{max} \end{cases} \tag{14}$$

Here, t_{max} is the time at which N_{max} is reached. This model makes three assumptions:

1. growth rate during lag phase is zero,
2. growth rate during exponential phase is constant,
3. growth rate during stationary phase is zero.

This is not a good description of the shape of the curve (no curvature), but the argument is that it captures the growth parameters well without the need for a more complicated model.

Additional models and questions you can tackle

In all three options above, you may try to tackle fitting to additional models you find in the literature. Some readings have been provided for each of the three data types. In addtion, you may wish to tackle some other hypotheses or explore patterns by considering additional covariates. For example,

Do different models fit different types of thermal performance curves (e.g., Photosynthesis vs Respiration)?

Do different taxa show different functional responses?

Does temperature or taxon identity affect which population growth rate model fits best?

You may also want to revisit the results of another paper that has done comparisons of the models you have chosen with your new dataset.

Suggested Workflow

You will build a workflow that starts with the data and ends with a report written in LaTeX. I suggest the following components and sequence in your workflow (you may choose to do it differently):

Data preparation script

First, a script that imports the data and prepares it for NLLS fitting. This may be in Python or R, and will typically have the following features:

- Creates unique ids so that you can identify unique datasets (e.g., single thermal responses or functional responses). *This may not always be necessary because your data might already contain a field that delineates single curves (e.g., an `ID` field/column)*
- Filters out datasets with less than x data points, where x is the minimum number of data points needed to fit the models. Note that this step is not necessary because in any case, the model fitting (or estimation of goodness of fit statistics) will fail for datasets with small sample sizes anyway, and you can then filter these datasets *after* the NLLS fitting script (see below) has finished running and you are in the analysis phase.
- Deals with missing, and other problematic data values.
- Saves the modified data to one or more csv file(s).

NLLS fitting script

A separate script that does the NLLS fitting. For example, it may have the following features:

- Opens the (new, modified) dataset from previous step.
- Calculates [starting values](more on this below).
- Does the NLLS fitting.
  - If you choose Python for this use `lmfit` (look up submodules `minimize`, `Parameters`, `Parameter`, and `report_fit`. *Have a look through* http://lmfit.github.io/lmfit-py (http://lmfit.github.io/lmfit-py), especially http://lmfit.github.io/lmfit-py/fitting.html#minimize (http://lmfit.github.io/lmfit-py/fitting.html#minimize). You will have to install `lmfit` using `pip` or `easy_install` (use sudo mode). Lots if examples of using lmfit online.
  - If you choose `R`, examples are here (Appendix-ModelFitting.ipynb).
- Uses the `try` construct because not all runs will converge: for Python, see this (https://docs.python.org/3.6/tutorial/errors.html); for R, recall this (07-R.ipynb#Errors-and-Debugging). *The more data curves you are able to fit, the better — that is part of the challenge*
- Calculates AIC, BIC, R^2, and other statistical measures of model fit (you decide what you want to include)
- Exports the results to a csv that the final plotting script can read.

Obtaining starting values

The main challenge for NLLS fitting is finding starting values. Ideally, you should determine starting values specific to each dataset (e.g., single thermal performance, functional response, or population growth rate curve) that you are trying to fit a model to. To do so, understanding how each parameter in the model corresponds to features of the actual data is key. For example, in the Gompertz population growth rate model(eq. 10), your starting values generator would essentially be an algorithm which, for each dataset,

- Calculates a starting value for r_{max} by searching for the steepest slope of the growth curve using the first few data points (fitting a straight line using OLS)
- Calculates a starting value of t_{lag} by intersecting the fitted line with the x (time)-axis
- Calculates a starting value for the asymptote A as the highest data (abundance) value in the dataset.

In general, a good strategy to optimize fits (and maximize how many dataseta are successfully fitted to a non-linear model) is to not sample starting values from a distribution. For example, you can choose a gaussian (high confidence in mean of parameter) or a uniform distribution (low confidence in mean, high confidence in the range of values that the parameter can take) with the mean being the value you inferred from the data.

We suggest you write a separate script/module/function that calculates starting values for the model parameters.

Final plotting and analysis script

Next, you can import the results from the previous step and plot every curve with the two (or more) models (or none, if nothing converges) overlaid. Doing this will help you identify poor fits visually and help you decide whether the previous, NLLS fitting script can be further optimized (e.g., by improving the starting values generator). All plots should be saved in a single separate sub-directory. This script will also perform any analyses of the results of the Model fitting, for example to summarize which model(s) fit(s) best, and address any biological questions involving co-variates.

Report compiling script

Then comes the $LaTeX$ source code and a (typically, Bash) script that compiles it.

A single script to run them all

Finally, write a Python or Bash script called `run_MiniProject.py` or `run_MiniProject.sh` respectively, which runs the whole project, right down to compilation of the LaTeX document.

Getting started

Doing all this may seem a bit scary at the start. However, if you approach the problem systematically and methodically, you will soon be on your way.

Here are some suggested first steps to get started:

- Explore the data in R or Python (e.g., using Jupyter).
- Write a preliminary version of the plotting script without the fitted models overlaid. That will also give you a feel for the data and allow you to see (literally) what shapes the curves can take.
- Explore the models you will be fitting. Basically, be able to plot them. Write them as functions in your Python/R script (you can then re-use these functions in your NLLS fitting script as well). Then do some plotting of the functions (you can suppress or sandbox those code lines for exploratory plotting of the functions in the final product).
- Figure out, using a minimal example (say, with one, "nice-looking" thermal performance, functional response, or population growth curve/dataset) to see how the NLLS fitting package and its commands work. This is your minimal example
- next, write a loop over all unique datasets (data curves) using the `try` to catch errors in case the fitting doesn't converge.

One thing to note is that you may need to do the NLLS fitting on the logarithm of the function (and therefore, the data) to facilitate convergence.

Readings

Many of these papers are in pdf format in the Readings directory on TheMulQuaBio repository.

General

- Levins, R. (1966) The strategy of model building in population biology. Am. Sci. 54, 421–431.
- Johnson, J. B. & Omland, K. S. (2004) Model selection in ecology and evolution. Trends Ecol. Evol. 19, 101–108.
- Motulsky, H. & Christopoulos A. (2004) Fitting models to biological data using linear and nonlinear regression: a practical guide to curve fitting. Oxford University Press, USA.

- Bolker, B. M. et al. (2013) Strategies for fitting nonlinear ecological models in R, AD Model Builder, and BUGS. Methods Ecol. Evol. 4, 501–512.

Thermal Performance Curves

- Schoolfield, R. M., P. J H Sharpe, and C. E. Magnuson. 1981. Non-Linear Regression of Biological Temperature-Dependent Rate Models Based on Absolute Reaction-Rate Theory. Journal of Theoretical Biology 88 (4): 719–31. https://doi.org/10.1016/0022-5193(81)90246-0 (https://doi.org/10.1016/0022-5193(81)90246-0).
- Zwietering, M. H., J. T de Koos, B. E. Hasenack, J. C. de Witt, and K. van't Riet. 1991. Modeling of bacterial growth as a function of temperature. Appl. Environ. Microbiol. 57, 1094–101.
- Dell, A. I., S. Pawar, and V. M. Savage. 2011. Systematic Variation in the Temperature Dependence of Physiological and Ecological Traits. Proceedings of the National Academy of Sciences of the United States of America 108 (26): 10591–10596. https://doi.org/doi (https://doi.org/doi): 10.1073/pnas.1015178108.
- DeLong, J. P., J. P. Gibert, T. M. Luhring, G. Bachman, B. Reed, A. Neyer, and K. L. Montooth. 2017. The Combined Effects of Reactant Kinetics and Enzyme Stability Explain the Temperature Dependence of Metabolic Rates. Ecology and Evolution 7 (11): 3940–50. https://doi.org/10.1002/ece3.2955 (https://doi.org/10.1002/ece3.2955).

Functional responses

- Holling, C. S. 1959. Some Characteristics of Simple Types of Predation and Parasitism. The Canadian Entomologist 91 (7): 385–98. https://doi.org/10.4039/Ent91385-7 (https://doi.org/10.4039/Ent91385-7).
- Holling, C S. 1966. The Functional Response of Invertebrate Predators to Prey Density. Mem. Entomol. Soc. Canada 48 (48): 1–86.
- Aljetlawi, A. A., E. Sparrevik, and K. Leonardsson. 2004. Prey-predator size-dependent functional response: derivation and rescaling to the real world. J. Anim. Ecol. 73, 239–252.
- Jeschke, J. M., M. Kopp & R. Tollrian. 2002. Predator functional responses: Discriminating between handling and digesting prey. Ecol. Monogr. 72, 95–112.
- Pawar, S., A. I. Dell, and V. M. Savage. 2012. Dimensionality of Consumer Search Space Drives Trophic Interaction Strengths. Nature 486 (7404): 485–89. https://doi.org/10.1038/nature11131 (https://doi.org/10.1038/nature11131).
- Pritchard, D. W., R. A. Paterson, H. C. Bovy, and D. Barrios-O'Neill. 2017. frair: an R package for fitting and comparing consumer functional responses. Methods Ecol. Evol. 8, 1528–1534.

Population Growth

- Zwietering, M. H., I. Jongenburger, F. M. Rombouts, and K. Van't Riet. 1990. Modeling of the Bacterial Growth Curve. Applied and Environmental Microbiology 56 (6): 1875–81.
- Buchanan, R. L., R. C. Whiting, and W. C. Damert. 1997. When Is Simple Good Enough: A Comparison of the Gompertz, Baranyi, and Three-Phase Linear Models for Fitting Bacterial Growth Curves. Food Microbiology 14 (4): 313–26. https://doi.org/10.1006/fmic.1997.0125 (https://doi.org/10.1006/fmic.1997.0125).
- Grijspeerdt, K. and P. Vanrolleghem. 1999. Estimating the parameters of the Baranyi model for bacterial growth. Food Microbiol. 16, 593–605.
- Micha, P., and M. G. Corradini. 2011. Microbial Growth Curves: What the Models Tell Us and What They Cannot. Critical Reviews in Food Science and Nutrition. https://doi.org/10.1080/10408398.2011.570463 (https://doi.org/10.1080/10408398.2011.570463).