|  |
| --- |
| **Deep Time Biology Lab**Department of Earth SciencesMontana State UniversityBozeman, Montana, 59717organ@montana.edu | <https://chrisorgan.github.io> |

**QA 1: Research**

#

# Best Practices

Quality assurance measures improve efficiency and reproducibility using administrative and procedural processes. By adhering to these processes, we aim to achieve predictable and standardized results, satisfy the requirements of funding agencies, and exemplify best practices for doing science.

This guide will help ensure that no paper, presentation, or grant leaves our lab that isn’t the highest quality! Ask whether your project meets the standards of our kitchen? Can we make this better in any way?

**Cultivating the Mind of an Interdisciplinary Scientist**

These are traits researched in *Range* by David Epstein. They can be nutured by culivating a broad range of interests and reading more, and more broadly.

* Tolerance for ambiguity
* System-level thinkers
* Knowledge & technical skills from other domains
* Repurpose old ideas and techniques from outside domains, think metaphorically
* Connect disparate pieces of information in new ways
* Synthesize information from different sources
* Able to dance across ideas (don't get stuck)

**Developing a Scientific Project**

**Brainstorm:** “Logic will get you from A to B. Imagination will take you everywhere.” ~Einstein. Try to ask questions other people aren’t thinking about; crazy questions are good.

1. Don’t be skeptical or critical (that part comes later).
2. Empty the *Bucket of the Obvious* so you can get to the good stuff. Unless the initial idea is really novel, reject the first three ideas (that is, low-hanging fruit likely not to be all that creative or novel). Once you do that, you can draw from a deeper source and get to unusual/unique ideas.
3. “The absence of limitations is the enemy of art.” ~Orson Welles
	* Think of limitations, constraints, problems, caveats, exceptions.
	* Why are they there? Can they be overcome?
	* If a problem is insoluble, you are probably asking the wrong question!
4. Use analogies to guide to creative areas.
	* Start project ideas by making tons of analogies to other fields, as far away as possible.

**Frame questions by the context of larger problems/issues**

1. What gap/problem does this address?
2. State hypotheses.
3. Are there complementary lines of evidence that can test the hypothesis?
4. Identify weaknesses and address them, then repeat.
5. Re-spin ideas from the perspective of people from different fields. How would they address the question and interpret the findings?
6. Expand outward on the project’s significance to ever-larger fields of science (but don’t over interpret).

**Refine**

Good science always looks for other explanations and sound thinking. If you are not an expert in an area – collaborate! Strengthen your ideas now before writing the grant:

1. Have you considered alternate hypotheses? You need to
2. Is the quality of data high? How could it be improved & verified
3. Are there biases/gaps in data?
4. Could blind methods be used?
5. What are the controls (+/-) for each objective?
6. Are there other tests or ways to verify results?
7. Are there other ways to interpret results?
8. Are there follow-up tests to deepen the study and extend its impact?

**More!**

1. Rennovate: Innovators refactor, reframe, and extend problems MUCH more than convential thinkers. Go back, question assumptions, expand implications, and rebuild the question and consiquences.

### Organization

**Naming folders & files:** File/folder names should convey information about their contents. Here are the rules we use:

1. Owner.Target.Descriptive\_Name.Version/Revision. E.g.:
* Organ.NSF.Hominoid\_Evolution.Project\_Description.2016.v2.docx
* Wilson.etal.PlosOne.MetaplasticNeuralSpinesTheropods.v1.docx
1. We use camel-case and underscores to connect words. Dot notation adds new information.
2. If you revise someone else’s file, amend your initials at the end. E.g.:
* Wilson.etal.PlosOne.MetaplasticNeuralSpinesTheropods.v1.7.CO.docx

**README.md/README.txt:** This is your lab notebook and analysis log – *it is an essential document for every project*. As such, ensure that other people (including your future self) can easily understand what you did and why. Create a simple text file called “README.md” (using Notepad++ or Atom) in the root project folder. Your readme.md should document:

* Project description and goals
* Date started & edited
* Author, collaborators, their contributions, and contact info
* Question(s), hypotheses, predictions, assumptions
* Methods and workflows
	1. Origin of all data in the project directory
	2. Where and how the data was obtained
	3. Record data version information, how it was changed, and why
	4. Versions of software and scripts used.
* Activities, logged by date
* Other important details

# Data Best Practices

The quality of science is as only as good as the quality of the data on which is it is based. Our data need to be thoroughly & routinely vetted to find inaccuracies, errors, and missing data.

**All data:**

* Know where it came from (is it original or an aggregation of an aggregation?)
* Never alter the original data set; create a working version
* Keep your data as flat files (simple Unicode text files) if possible
* Collect all details, even those you don’t think you’ll need – you often can’t go back
* Make versioned backups of your data regularly
* Data should be “tidy” (Wickham, Tidy data, 2014):
	1. Columns represent variables & rows represent observations
	2. Each row represents individual observations
	3. Observational units form tables
* Use headers that are self-explanatory and contain units (E.g., “BodyMass\_kg”)
* Results saved in easily identifiable flat files or spreadsheets
* Entries must *exactly* match for data spread across files (E.g., species names must be *exactly* the same across different files)

**Empirical data:** Data that you collect/measure

* Note all details about who, what, where, when, and why about the data & collection

**Databases:** Data downloaded from on- or off-line data repositories

* Note when and how the data were accessed
* Any details regarding the version of the database, its structure

**Literature data:** Data harvested from the literature

* Collect data only from primary sources (avoid already aggregated data)
* Collect only high-quality data (don’t take data from non-peer-reviewed sources)
* Retain original data IDs (E.g., GenBank accession numbers)

**Cleaning data:** After data sets are collected, check that:

* Headers are informative
* Consistent columns (E.g., look for mixed data types, spelling or formatting errors)
* Outliers & quartiles with boxplots
* State frequencies & class imbalance for categorical data
* Missing data (impute, drop, or leave in?)
* Duplicate rows (this may not be an issue with a specific data set)

**Blinding data:** If possible, data should be anonymized before analysis to reduce bias. This can significantly impact results – do not underestimate our ability to fool ourselves. We are all subject to confirmation bias.

* **De-identification:** One way to do this is to have another lab member randomly assign labels so that data can be analyzed blindly. For example, species names could be replaced with random alphanumeric labels. Before publication, data are re-identified.

# Code

Complex code is hard to understand, maintain, and reproduce. Here is a summary of coding best practices for coding :

* Use comments to describe functions and variables
* Correct errors as they occur
* Keep code simple!
* Maintain naming conventions uniform throughout

**Versioning:** Use major-minor versions for names. E.g., *FileName.v1.1*

**Commenting:** All code should have in-line documentation using language-specific tags. This is essential for writing code that is easily understandable by other people. Comment often to:

1. Explain what and why you wrote that part of the code
2. Include a to-do item, such as *refactoring* or planning next steps
3. Reviewer comments
4. Attribution if code is borrowed or re-used from another source

**Scripts:** scripts and smaller bits of code should contain header information. For example:

# Script name: ???

# Purpose: ???

# Author(s): ???

# Email: ???

# Date Created: ???

# Notes: ???

**Testing and validation:** Write unit tests *for all code* to ensure thatyour code is doing what it was intended to do each time you start and finish your work using something like pytest (E.g., <https://docs.pytest.org/en/6.2.x/getting-started.html>). If using GitHub, these tests should be automated upon syncing.

# Analysis & Model Development

**Setup**

* Each time an analysis is run, the output should be recorded in an amended or new Data/Results file
* All of the settings and software version numbers used should be noted in README
* If an analysis produces output files, these should be saved and labeled so that the file/folder name conveys information about the analysis performed – which parameters were constrained, variables excluded, data transformed, etc: For example:
	+ *Forelimb.RatesNodes.ANCOVA.Theraopods.txt*

**Data (before analysis)**

* Ensure quality data appropriate for addressing the scientific question
* **Sample size:** rough rule is to have a minimum of 10 times the parameter estimates (e.g., normal regression estimates slope and intercept, so min data = 20 data points)
* **Exploratory data analysis (EDA)**
	+ Continuous data: examine data distributions visually to inspect for normality, typos, or skew (values at the thin end of a skew have more influence in the model)
	+ Discrete/categorical data: Factors (and dummy-coded variables) should not have rare states (90% 0’s and 10% 1’s) because, like above, rare states exert undue influence on the model. Interaction states of several factors should also not be rare and each interaction state should occur multiple times in the data.
* **Centering & transformations**
	+ Center/transform after other transforms (because transformations will make the data more normal).
	+ *Linear transform: Centering* (subtracting each variable by the mean)

X – mean

* + - changes values but not the scale
		- Use when an intercept of 0 doesn’t make sense (e.g., body mass)
		- The intercept is now the mean, but the regression coefficient for each variable will not change.
	+ *Linear transform: Standardizing (z-scores):* centers data with mean of zero and scales data with a standard deviation of one. An extension of centering:

(X – mean) / sd

* + - Use to compare coefficients for predictors measured on different scales.
		- Coefficients represent the average change in response per unit change in standard deviation of the predictor.
		- When interactions are in the model, the coefficients derived for the main effects indicate their effect when interactions are 0.
	+ *Logit transforms* (for percentile data)
	+ *Logarithmic transform* (common for skewed data in biology, base doesn’t matter). Good for Brownian motion models because values for traits like body size can be negative. The primary reason to log transform is to get a multiplicative interpretation of the regression coefficient. log(y) ~ a + b \* log(x) regression is a ...% change in x corresponds with a ...% change in y.

**Model building**

* Consider whether the model you are building makes sense and addresses the underlying scientific question.
* Use model selection approach or null hypothesis significance testing to find which predictors to include, but not both (they are incompatible). Use good scientific reasoning for the inclusion of:
	+ Predictors
	+ Factors (dummy)
	+ Interactions (component terms must be in model too)
	+ Non-linear terms (usually squared; component terms must be in model too)

**Collinearity**

* + Intererpeting betas becomes problematic
	+ Detect using the variance inflation factor (VIF; use a predictor as the response and calculate R2, repeat for each predictor, VIF = 1/(1-R2); values >=10 are problematic).
	+ GVIF (GVIF, Fox and Monette 1992) for different data types
	+ Solutions:
		- Remove offending predictors
		- Identify *proxy sets* of multicollinear variables using clustering
		- Replace collinear predictors with PCA scores

**Interpreting Regression parameters**

*Only the dependent (y) variable is log-transformed:* the coefficient should be exponentiated to derive the multiplication factor for each unitary increase in the independent (x) variable. Consider data transformed by nl(x) and a coefficient of 0.14. By calculating exp(0.14), we get 1.15, which means that for every unitary increase in the independent variable, the dependent variable increases by approximately 1.15 times (15%). You can calculate the exact percent change in Y as (exp(b\*ΔX) - 1)\*100%. Using this formula with the above example, the exact percent change for a 10 unit change would be (exp(0.14\*10) - 1)\*100% = 306%.

*Only the independent (x) variable(s) is log-transformed:* the coefficient is divided by 100 to express how a 1% increase in the independent variable alters the dependent (y) variable by (coefficient/100) units. A coefficient of 0.14 is divided by 100 to get 0.0014, indicating that a 1% increase in the independent variable leads to an approximately 0.001 unit increase in the dependent variable. For an z percent increase, the coefficient is multiplied by log(1.z). For instance, a 10% increase in the independent variable results in an approximate 0.013 unit increase in our dependent variable, calculated by 0.14 \* log(1.10).

*Both response (y) and predictor (x) variable(s) have been log-transformed:* the coefficient can be understood as the percentage increase in the dependent variable for every 1% increase in the independent variable. Suppose the coefficient is 0.14. With every 1% increase in the independent variable, our dependent variable increases by 0.14%. For an z percent increase, we calculate 1.z raised to the power of the coefficient, subtract 1, and multiply by 100. For example, a 10% increase in the independent variable corresponds to an approximate 1.34% increase in the dependent variable [(1.10^0.14 - 1) \* 100].

**Multiple comparisons:** Benjamini and Hochberg procedure tells you which *p* values to consider statistically significant:

* + Perform your statistical tests and get the *p*-value for each. Make a list and sort it in ascending order.
	+ Choose a false-discovery rate and call it *q (0.1 or 0.2)*.
	+ The number of statistical tests is *m*.
	+ Find the largest *p*-value such that *p* ≤ *iq*/*m*, where *i* is the *p* value’s place in the sorted list.
	+ Call that *p*-value and all smaller than it statistically significant (< 0.01)

**Model checks**

* + Residuals
		- Check to see to normally distributed and any outliers (eyeball)
		- Homoskedastic (no pattern along the axis of residuals). Heteroskedastic trends might reveal that the data need to be transformed.
		- Solutions if there is a pattern
			* Transformation (e.g. log)
			* Polynomial regression
			* You may have missed a factor that interacts with your fixed effect. Add that factor and an interaction term
	+ Convergence tests (MCMC)