**Best things learned from Murray’s course:**

* Factorizing a binary 0/1 variable will screw up emmeans backtransformations later on! Needs to fit as numeric with glm
* Binomial model coefficients can be used to calculate the LD50 for survival data: LD50 = -intercept/slope
* Interactions between continuous variables **REQUIRE** the covariates to be centered first, otherwise they WILL be collinear
* Can keep in the transformation of predictors (e.g., scale(…, scale=FALSE), log(…)) and then the original variables can be used as inputs in emmeans, etc.
* emmeans package requires TRUE and FALSEs to be spelled out to work properly
* Cook's distance is not needed if you only have categorical predictors
* Use emmeans on link scale %>% regrid %>% pairs %>% confint to get absolute differences (X is # units higher than Y) rather than fractional differences
* *nlme* was the first way of doing mixed-effect models, it uses quasi-likelihood by re-fitting the models different ways, so isn’t as modern as *lme4* or *glmer*, but these are limited in the families that it supports, including zero-inflated models. Thus *glmmTMB* to fit the broad range of models used in ecology
* 'group' works similar to 'by', just that group is more versatile when it comes to mapping, etc. because it will automatically group by polygon, while by needs a named list as input. Otherwise, they work very similarly.
* Allowing a random slope and intercept model is ok with minimum evidence of non-parallel lines, as the model will tell us if this is unnecessary.
* (1|random) is interpreted as ‘do the intercept [‘1’] by [‘|’] the random factor’
* Can fit random slopes using both covariates and factor variables!
* Different random effect structures include correlated vs. non-correlated:
  + Correlated random intercept and slope:   
    x+(x|g) = > 1+x+(1+x|g)
  + Uncorrelated random intercept and slope:   
    x+(x||g) => 1+x+(1|g)+(0+x|g)
* Sometimes models fail to converge using one of the other optimizers, even if they are supposedly better…
* `scales::pseudo\_log\_trans()` is good for dealing with zero-inflated values and still log it!
* When fitting a negative binomial in glmmTMB, use family = **negbinom2**(link = “log”), which is the quadratic parameterization that is more relatable to the Poisson distribution!
* When fitting a random slope model, we should try to keep the random effects slope the same as the fixed effects term. So if the fixed effects was a polynomial, we should keep the random effects term a polynomial too.

GAMs

* The general rule of thumb for wiggliness k.check is that if you need more knot dfs, we should double it.
* edf = 1 suggests just a significant linear term (not curvilinear), edf = 0 suggests no such trend

Bayesian

Priors for sigma:

* *cauchy*(0,5) was a really good prior for sigma prior to the widespread adoption of no U-turn samplers (NUTS); however, now *gamma*(2,1) tends to work the best. *Exp*(1) also is used
* Only baysian stats can say something about where the true mean lies between with their 95% credible intervals. The frequentist cannot, as they assume a single value of the true mean, whereas Bayesian assumes that the true mean is a distribution.
* Evidence ratio = How much evidence is there that the probability is >50 divided by how much evidence is against that
* Priors must be on the link scale – so be careful with their backtransformation!
* Leave one out is the best model.
* Instead of random effects, call them varying effects
* Instead of fixed effects, call them population effects
* **Fitted** values are always on the **response** scale and *predicted* values are always on the *link* scale
* emtrends for polynomial/trends, emmeans for marginal means

Questions:

How do you know when to make smaller subset models

Extra things to make:

\*\*Package for extracting formula, determining model differences, and filling these in for the AIC table\*\*

It is not efficient to loop over and overwrite the object with itself when it comes to larger tasks. It is much more efficient to plan on the size of the memory, then iterate to fill that memory.

e.g.:   
p<-c(); for(i in 1:10){p <- c(p,i)} # bad

p <- rep(NA\_numeric\_, times=10); for(i in 1:length(p){p[i] <- i}