IPM tutorial

Tom Miller

2022-07-13

A bit about Markdown and tidyverse

This is an RMarkdown document, which just means that it intersperses plain text with "chunks" of R code. Here, for example, is the first chunk, which sets some global options and loads R packages we will need. You can interact with RMarkdown documents either by compiling the whole thing to pdf (click the "knit" button in RStudio) or by running the code chunk by chunk (hit the green triangle in the top corner of each chunk to run the chunk).

I use the tidyverse package for data manipulation. I am not sure if you have experience with this but tidyverse is pretty great. You will see me use "pipes" (the %>% symbol) and if you want to know what this is doing let's talk more or check out https://r4ds.had.co.nz/.

Getting started

The starting point for data-driven models such as IPM is, of course, data. I will use my cactus data, which I have been collecting from the Sevilleta LTER since 2003. The data frame is set up such that each row is an individual plant tracked over a single transition year, from t to $t+1$. You should have a data set in this general format in order to effectively use this script as a workable template. (In your case the transition period may more than one year but the same principles apply.) I have pared down these data to the bare essentials.

Here are the variables:

```
str(cactus)
```

```
## tibble [8,186 x 13] (S3: tbl_df/tbl/data.frame)
##  $ Plot            : chr [1:8186] "T1" "T1" "T1" "T1" ...
##  $ TagID           : chr [1:8186] "AD10" "AD11" "AD14" "AD15" ...
##  $ Year_t          : num [1:8186] 2004 2004 2004 2004 2004 ...
##  $ Height_t        : num [1:8186] 89 78 99 127 93 79 111 94 120 110 ...
##  $ Width_t         : num [1:8186] 72 42 126 54 115 69 61 67 60 100 ...
##  $ Perp_t          : num [1:8186] 72 42 126 54 115 69 61 67 60 100 ...
##  $ TotFlowerbuds_t : num [1:8186] 1 1 9 9 10 11 6 21 27 4 ...
##  $ Newplant        : num [1:8186] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Survival_t1     : num [1:8186] 1 1 1 1 1 1 1 1 1 1 ...
##  $ Height_t1       : num [1:8186] 102 93 100 105 68 95 107 98 125 117 ...
##  $ Width_t1        : num [1:8186] 63 34 117 49 115 62 53 61 81 119 ...
##  $ Perp_t1         : num [1:8186] 36 29 45 27 38 49 43 42 34 56 ...
##  $ TotFlowerbuds_t1: num [1:8186] 3 3 13 2 4 18 11 11 7 6 ...
```

Most of this should be pretty self explanatory. We have eight plots and each plant in each plot has a unique TagID. There are three size measurements: max height, max width, and width perpendicular to the

maximum. We also count the total number of flowerbuds as a measure of reproduction. The start and end of the transition year are indicated with `t` and `t1`. We record survival at the end of the transition year. When we find new plants, these are indicated with `Newplant==1` and they first appear in year `t1`.

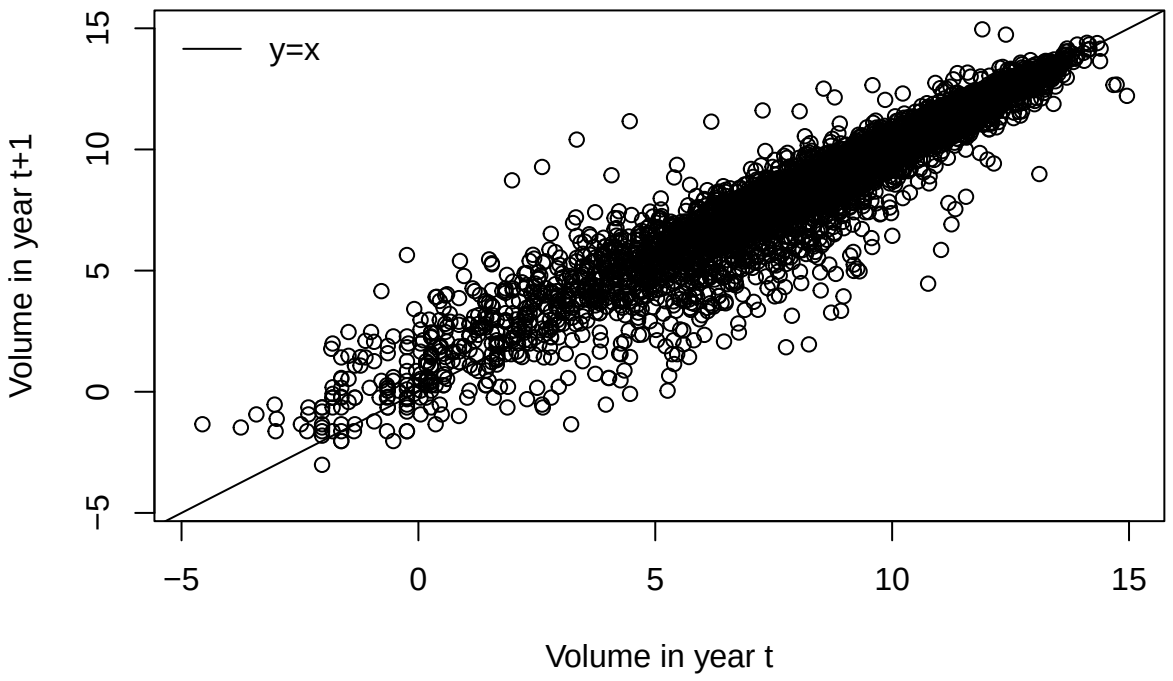
We need one size variable for the IPM, but we have three. We can use the size measurements (in cm) to estimate plant size based on the volume of a cone. First we'll define a function to calculate volume (cm^3) based on the three measurements, then we will use the function to create two new variables, for volume at the start of the transition year and volume at the end of the transition year.

```
volume <- function(h, w, p){
  (1/3)*pi*h*(((w + p)/2)/2)^2
}
cactus <- cactus %>%
  mutate(vol_t = volume(h = Height_t, w = Width_t, p = Perp_t),
         vol_t1 = volume(h = Height_t1, w = Width_t1, p = Perp_t1))
```

General linear model for growth

Now we would like to fit our first model: a linear model for size at the end of the census interval (year) as a function of size at the start. This will also end up being our growth sub-model for the IPM kernel. For most organisms, including this one, the natural log of size is better-behaved, statistically, than raw size. So we will be working with log(volume). Here is what the one-year change in size looks like:

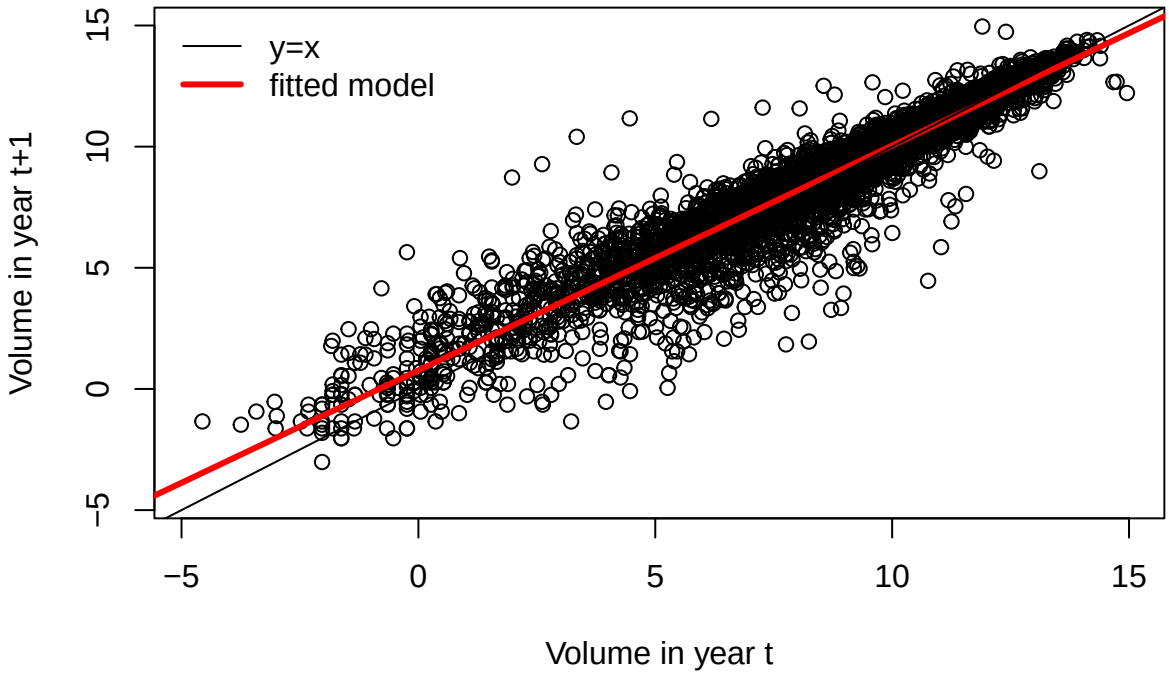
```
plot(log(cactus$vol_t1) ~ log(cactus$vol_t), xlab = "Volume in year t", ylab = "Volume in year t+1")
abline(0,1)
legend("topleft",bty="n",lty=1,legend=c("y=x"))
```



Here is the model statement, a simple linear regression for future size as a function of current size, and a plot of the fit.

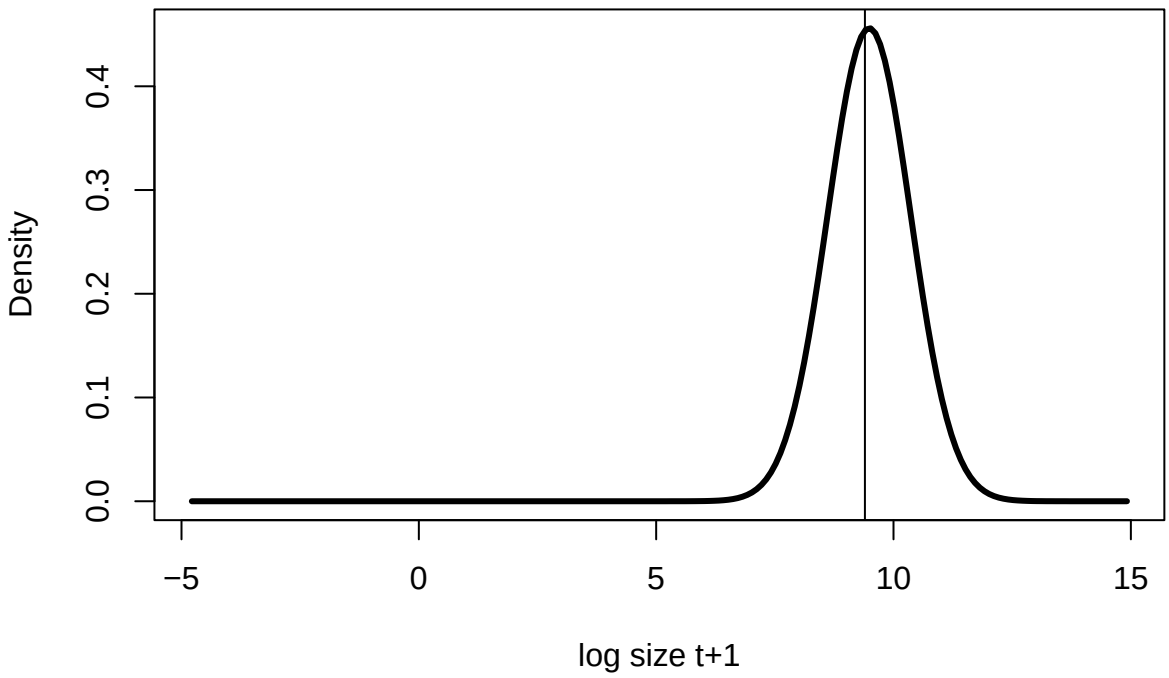
```
growth_model <- lm(log(vol_t1) ~ log(vol_t), data = cactus)

plot(log(cactus$vol_t1) ~ log(cactus$vol_t), xlab = "Volume in year t", ylab = "Volume in year t+1")
abline(coef(growth_model)[1],coef(growth_model)[2],col="red",lwd=3)
abline(0,1)
legend("topleft",bty="n",lty=1,col=c("black","red"),lwd=c(1,3),legend=c("y=x","fitted model"))
```



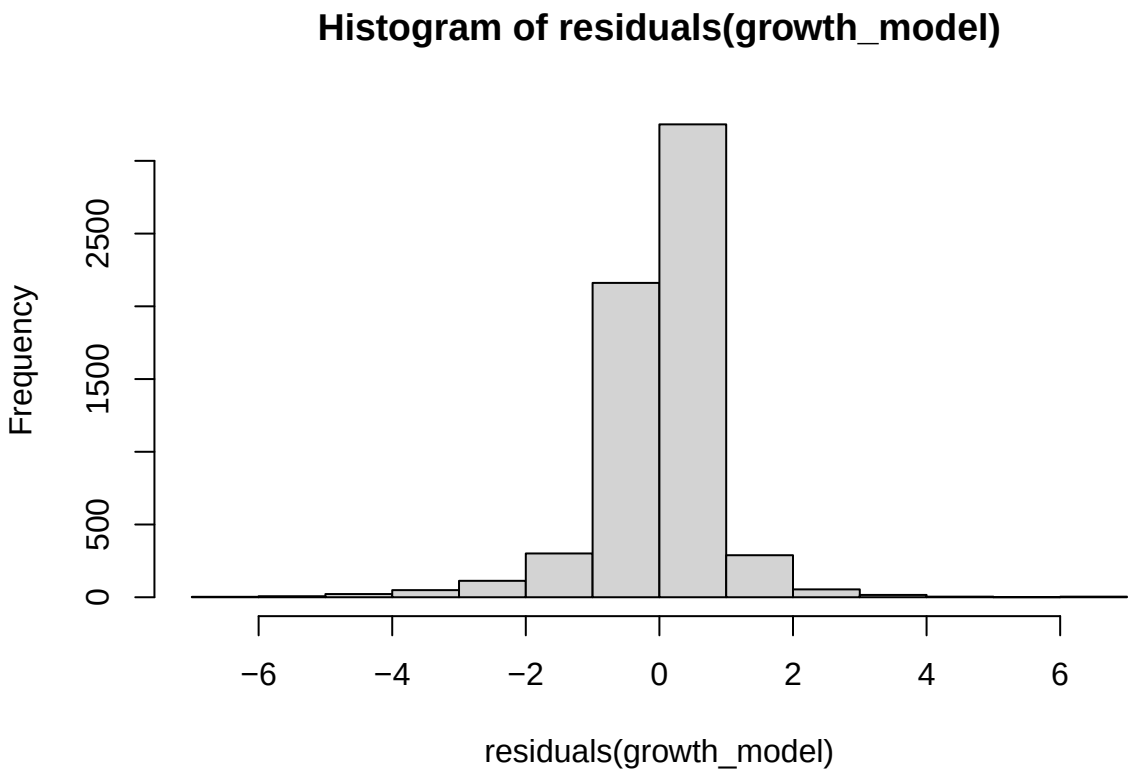
Remember that the plot above is just showing *mean* expected size - that is only half of the story! The model also provides an estimate for the distribution of future size given current size. For example, for a plant of median size in time t, here it the distribution of it's expected size at time t+1:

```
## dummy variable for size to make smooth plots
x_dummy <- seq(min(log(cactus$vol_t),na.rm=T),max(log(cactus$vol_t),na.rm=T),0.1)
plot(x_dummy,
dnorm(x = x_dummy,
      mean = coef(growth_model)[1] + coef(growth_model)[2] * median(log(cactus$vol_t),na.rm=T),
      sd = summary(growth_model)$sigma),
type="l",lwd=3,xlab="log size t+1",ylab="Density")
abline(v=median(log(cactus$vol_t),na.rm=T))
```



Getting the mean and variance of growth correct is really important because this will govern size transitions within the IPM kernel. So we should always be skeptical of our fitted models. Here I will have a look at whether the residuals are normally distributed.

```
##are the residuals normally distributed
hist(residuals(growth_model))
```

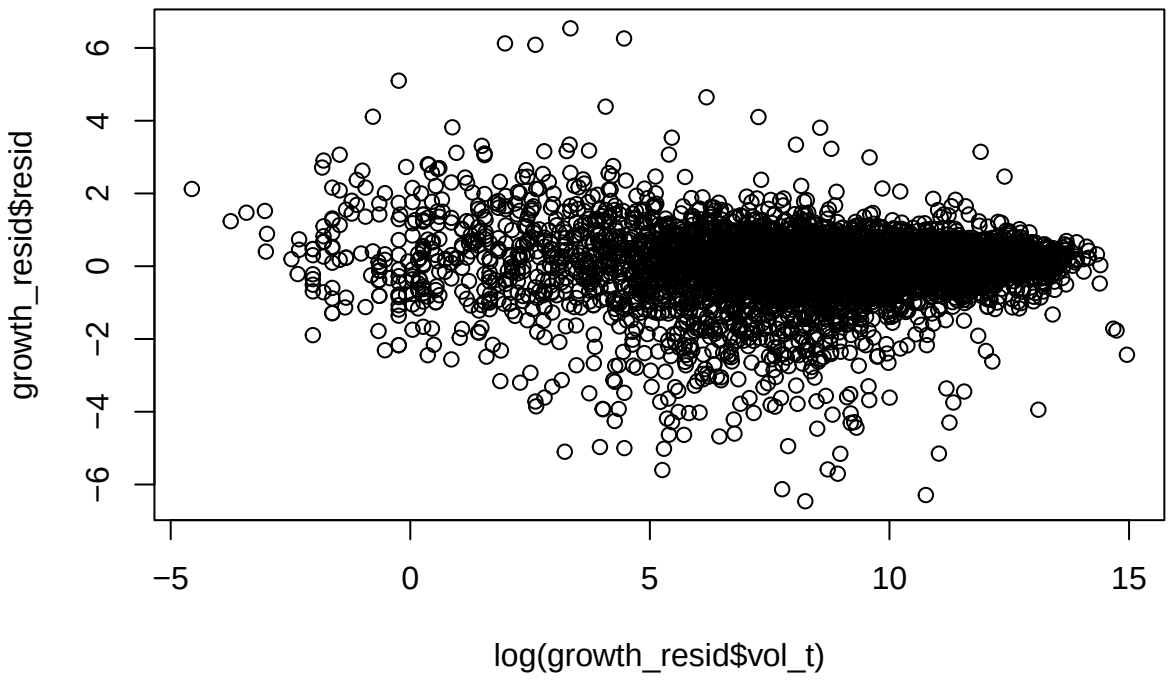



```
shapiro.test(residuals(growth_model)[1:5000])
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(growth_model)[1:5000]
## W = 0.86437, p-value < 2.2e-16
```

No, actually these residuals do not pass a normality test. We can also look at whether the variance is constant across all sizes, another assumption of our linear model.

```
##is the residual variation related to size?
growth_resid <- cactus %>%
  filter(!is.na(vol_t),!is.na(vol_t1)) %>%
  mutate(resid = growth_model$residuals)
plot(log(growth_resid$vol_t),growth_resid$resid)
```



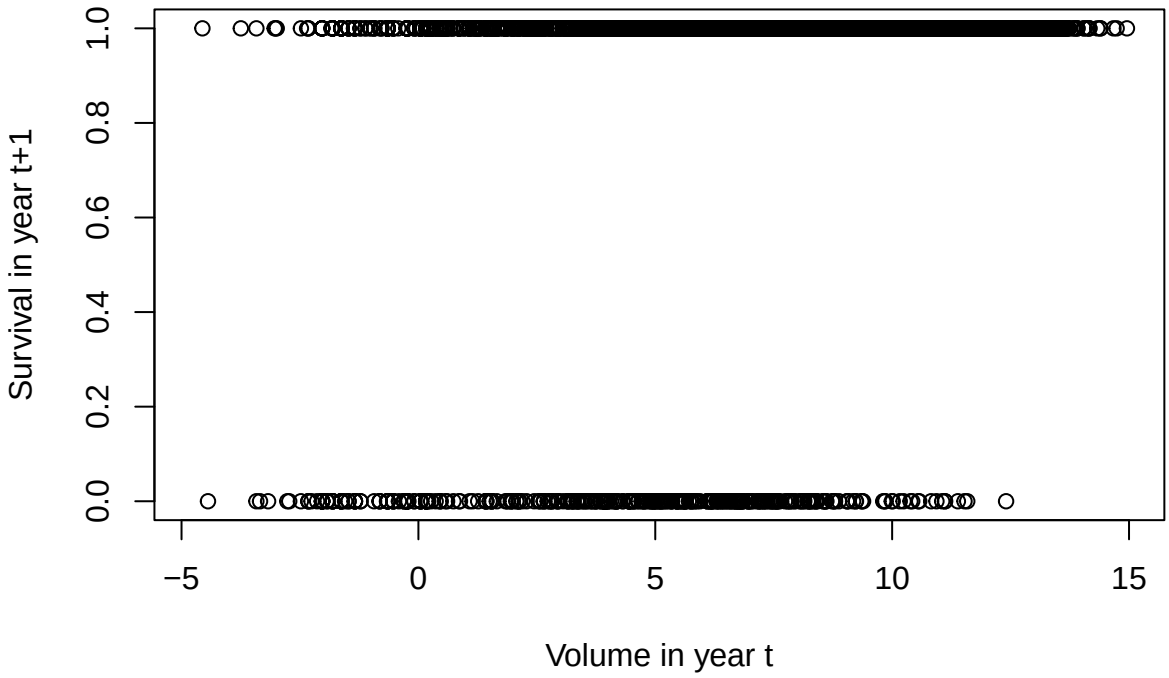
There appears to be greater variance at smaller sizes, and you can also see this in the `size_t1` vs `size_t` plot above. So the bottom line here is that the growth model looks pretty good if we are just worried about running a line through points, but if we dig deeper the simple Gaussian model does not describe the real size transitions very well. From here, I would want to explore more complex growth functions that may do a better job, but whether that will be necessary for you depend entirely on your data. So for now I'll proceed to other vital rates, but we should have a close look at your growth data to decide the best approach.

Generalized linear model for survival

Survival is certainly not Gaussian, since survival can only take one of two values: alive or dead. This is consistent with a binomial distribution (the coin-flipping distribution), so we will fit a generalized linear model assuming a binomial response distribution. This will tell us the probability of surviving to year $t+1$ as a function of size in year t.

Let's have a look at the survival data with respect to size. The relevant response variable is `Survival_t1`:

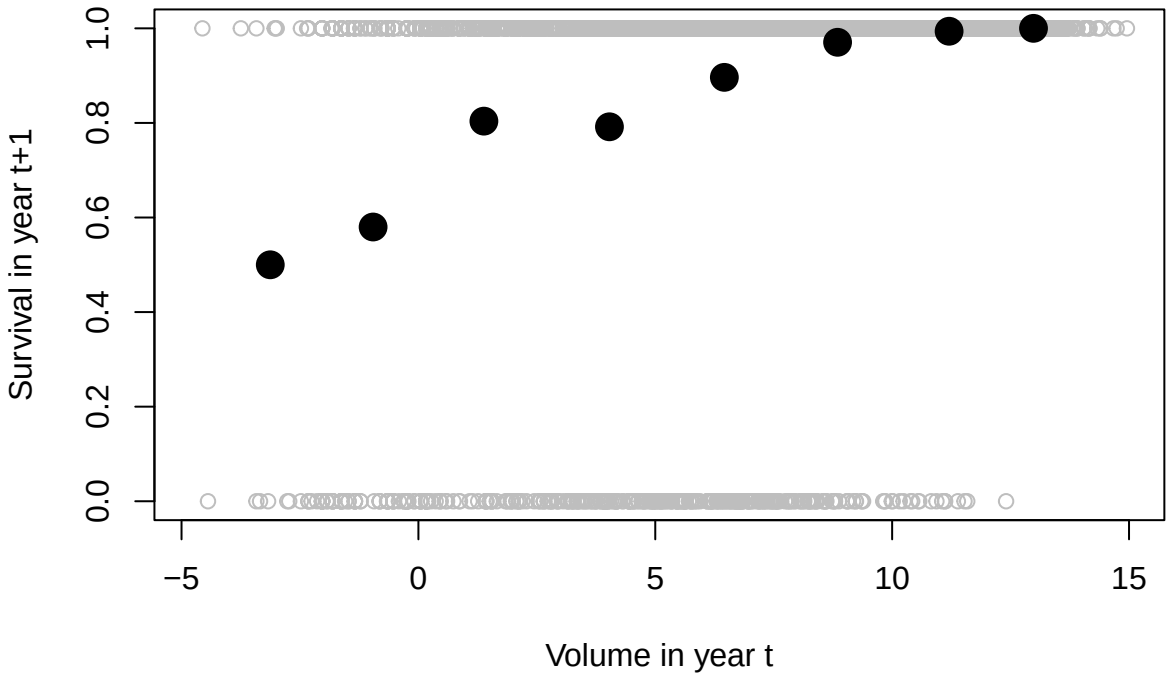
```
plot(cactus$Survival_t1 ~ log(cactus$vol_t), xlab = "Volume in year t", ylab = "Survival in year t+1")
```



You'll notice that it is hard to make too much of the raw data, because they are just 0's and 1's. Binomial data such as these are often best visualized by taking binned means over the x-axis. This is discretization, but just for visualization. The model we fit will treat size continuously.

```
surv_bin <- cactus %>%
  mutate(size_bin = cut_interval(log(vol_t), n=8)) %>%
  group_by(size_bin) %>%
  summarise(mean_size = mean(log(vol_t),na.rm=T),
            mean_surv = mean(Survival_t1,na.rm=T))

plot(cactus$Survival_t1 ~ log(cactus$vol_t), xlab = "Volume in year t", ylab = "Survival in year t+1",
points(surv_bin$mean_size, surv_bin$mean_surv, pch=16, cex=2)
```



Now here is the model and output.

```
survival_model <- glm(Survival_t1 ~ log(vol_t), family = "binomial", data=cactus)
summary(survival_model)
```

```
##
## Call:
## glm(formula = Survival_t1 ~ log(vol_t), family = "binomial",
##     data = cactus)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0403   0.1645   0.2134   0.3142   1.6621
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.44093    0.08886   4.962 6.98e-07 ***
## log(vol_t)   0.33623    0.01390  24.182  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3111.1  on 6688  degrees of freedom
## Residual deviance: 2455.8  on 6687  degrees of freedom
##   (1497 observations deleted due to missingness)
```

```
## AIC: 2459.8
##
## Number of Fisher Scoring iterations: 6
```

As always, it is important to gain confidence that the model fits the data adequately, because you will pass this model to your IPM for projecting population dynamics. There are several approaches to diagnostics of GLMs - this blog post from Ben Bolker is a useful starting point https://ms.mcmaster.ca/~bolker/R/misc/modelDiag.html.

One common problem with binomial or Poisson models is overdispersion, where the data are more variable than expected under the assumptions of the model. Here is one quick and dirty way to test for it.

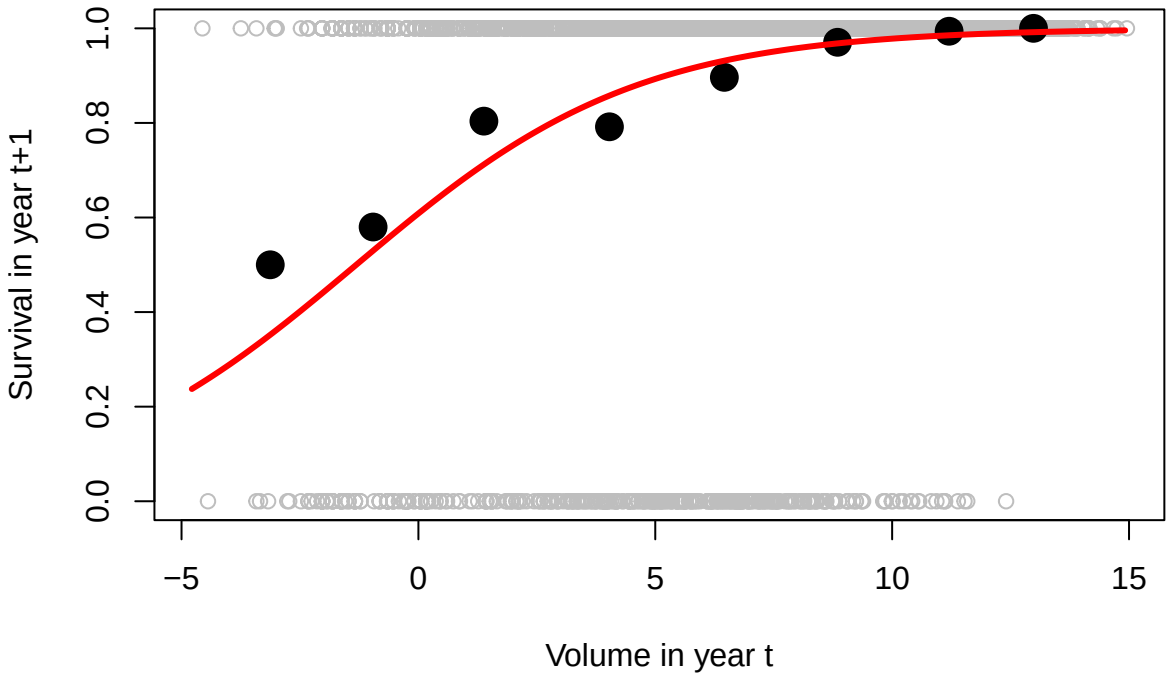
```
resid.ssq <- sum(residuals(survival_model,type="pearson")^2)
resid.df <- nrow(subset(cactus,!is.na(vol_t) & !is.na(Survival_t1)))-length(coef(survival_model))
resid.ssq/resid.df
```

```
## [1] 0.7778849
```

As a rule of thumb, this ratio of ss:df should be close to 1 (if it's much over 1, that would indicate overdispersion). The result here suggests that we don't have an overdispersion problem, so I am going to keep moving forward with the analysis. Let's visualize the predictions of the model. You'll notice that I define and use a function called `invlogit`. This is the inverse link function which takes the predictions of the linear model (any real number between `-Inf` and `+Inf`) and transforms them to probabilities.

```
invlogit <- function(x){exp(x)/(1+exp(x))}

plot(cactus$Survival_t1 ~ log(cactus$vol_t), xlab = "Volume in year t", ylab = "Survival in year t+1", c
points(surv_bin$mean_size, surv_bin$mean_surv, pch=16, cex=2)
lines(x_dummy,invlogit(coef(survival_model)[1]+coef(survival_model)[2]*x_dummy),col="red",lwd=3)
```



This looks great, but remember the red line is not fit to the black points (the binned means), it is fit to the gray dots.

Generalized linear model for flowering

We will fit a model for the probability of flowering that is basically the same model as survival, with the exception that we'll fit flowering in year t as a function of size in year t. Biologically, this makes the most sense for this system but it could be done differently.

```
flowering_model <- glm(TotFlowerbuds_t>0 ~ log(vol_t), family = "binomial", data=cactus)
summary(flowering_model)
```

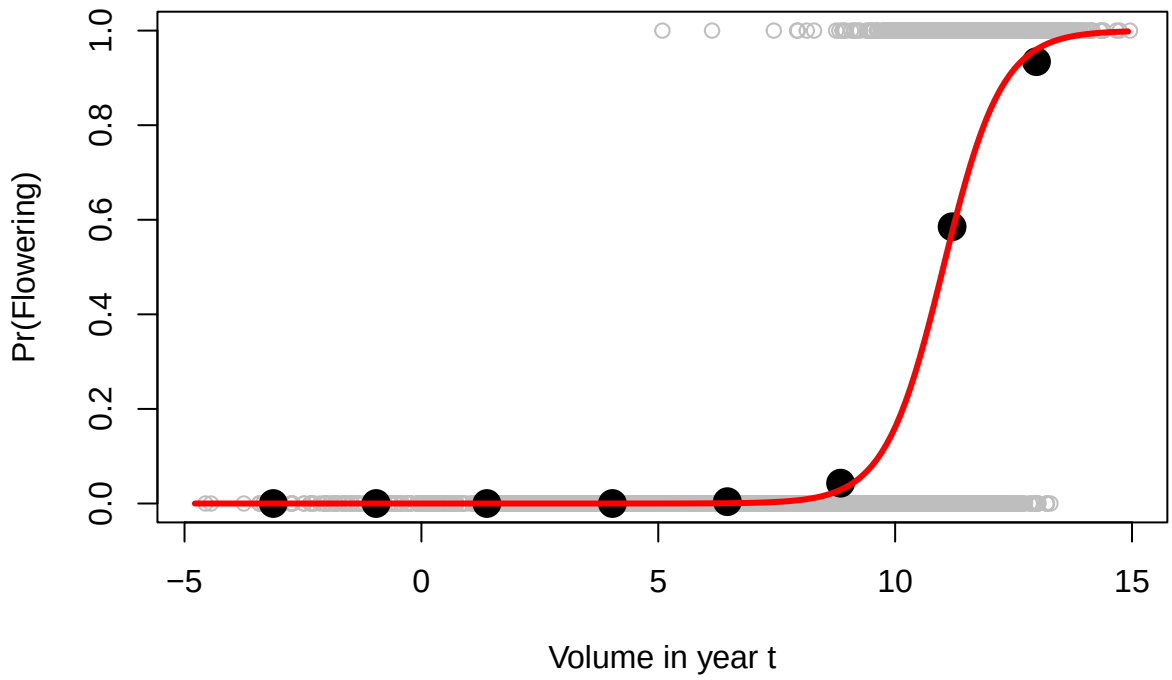
```
##
## Call:
## glm(formula = TotFlowerbuds_t > 0 ~ log(vol_t), family = "binomial",
##     data = cactus)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7160  -0.3020  -0.0206   0.3571   4.3843
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -17.85655    0.58454  -30.55   <2e-16 ***
## log(vol_t)    1.62073    0.05288   30.65   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 6461.4  on 5294  degrees of freedom
## Residual deviance: 2878.7  on 5293  degrees of freedom
##   (2891 observations deleted due to missingness)
## AIC: 2882.7
##
## Number of Fisher Scoring iterations: 8
```

We can visualize the fit just as we did for survival:

```
flow_bin <- cactus %>%
  mutate(size_bin = cut_interval(log(vol_t), n=8)) %>%
  group_by(size_bin) %>%
  summarise(mean_size = mean(log(vol_t),na.rm=T),
            mean_flow = mean(TotFlowerbuds_t>0,na.rm=T))

plot(cactus$TotFlowerbuds_t>0 ~ log(cactus$vol_t), xlab = "Volume in year t", ylab = "Pr(Flowering)", c
points(flow_bin$mean_size, flow_bin$mean_flow, pch=16, cex=2)
lines(x_dummy,invlogit(coef(flowering_model)[1]+coef(flowering_model)[2]*x_dummy),col="red",lwd=3)
```



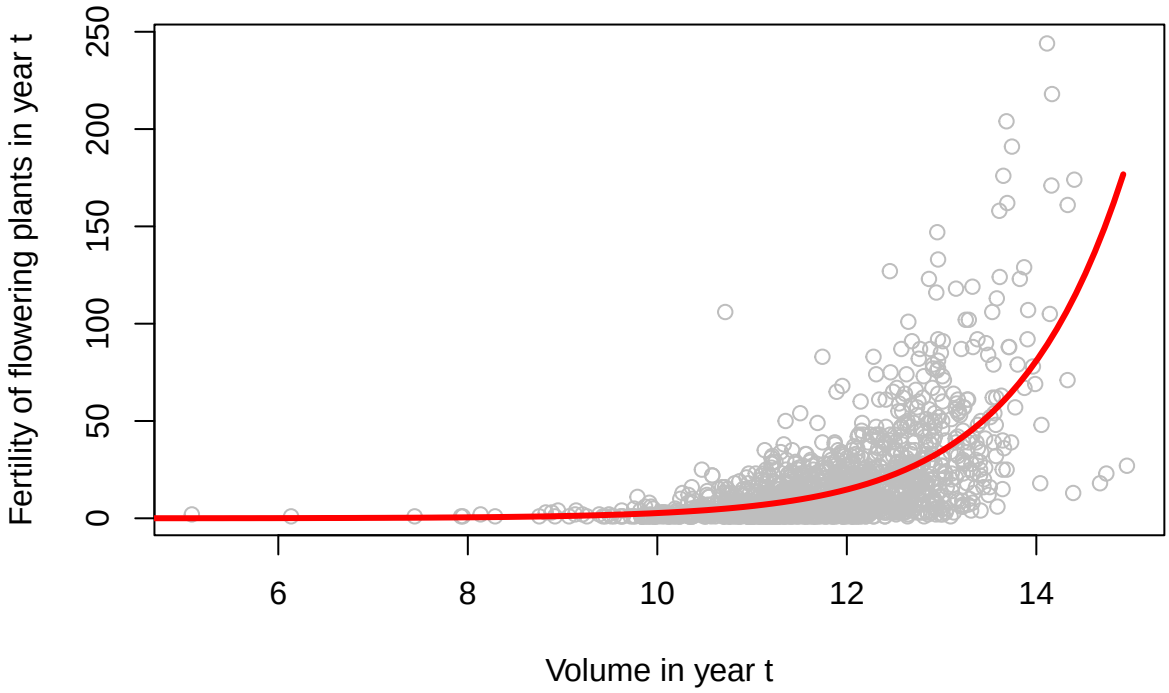
Generalized linear model for fertility

The last vital rate model we will fit is the fertility (# flowerbuds) of flowering plants. Some people combine flowering and fertility into one step and use a zero-inflated model to recognize the difference between flowering and not flowering. I'm in the habit of splitting this into two analyses, but it could be done differently.

The number of flowerbuds is a count, so the Poisson distribution is a natural starting point.

```
fert_model <- glm(TotFlowerbuds_t ~ log(vol_t), family = "poisson", data=subset(cactus,TotFlowerbuds_t>

plot(cactus$TotFlowerbuds_t[cactus$TotFlowerbuds_t>0] ~ log(cactus$vol_t[cactus$TotFlowerbuds_t>0]), xl
lines(x_dummy,exp(coef(fert_model)[1]+coef(fert_model)[2]*x_dummy),col="red",lwd=3)
```




```
resid.ssq <- sum(residuals(fert_model,type="pearson")^2)
resid.df <- nrow(subset(cactus,!is.na(vol_t) & !is.na(TotFlowerbuds_t)))-length(coef(fert_model))
resid.ssq/resid.df
```

```
## [1] 4.178323
```

The fit looks good visually, although the ratio of ssq to df suggests that the data are over-dispersed relative to expectations for a Poisson. The negative binomial distribution has a more flexible mean-variance relationship and that would be a next step to find a better-fitting distribution. But we don't use the variance of this function in the same way we use the variance of growth, so I am less worried about getting the variance exactly right as long as the mean is right.

Other IPM elements

There are lots of other pieces of information that we'll need for a complete IPM, and these details depends on the life cycle of the study species. In my case, I need to know how many seeds come from each flowerbud, the probability that a seed germinates to become a seedling, and the size distribution of seedlings. If there is any seed banking or dormancy, I would want to incorporate that too. We will need to think about those details of the life cycle. But for most IPM applications the vital rates that we fit above represent the bulk of the work.

A note about mixed models

The models I fit above are very simple general or generalized linear models. Many data sets have additional sources of variance or non-independence that need to be accounted for. For demographic data, the most common such sources are plots (or any type of spatial replication) and years (or any type of temporal replication). You'll notice that I have both plot and year information in my data, but I did not use it in the models I fit. If I wanted to use it I would need to fit *mixed* models that include both fixed and random effects. As an example, I will do that here with the survival model.

Let's transition from a survival model with only the fixed effect of size to a model incorporating the heterogeneity in the data that can be explained by plot, a random effect. Here, we will be using the lme4 package.

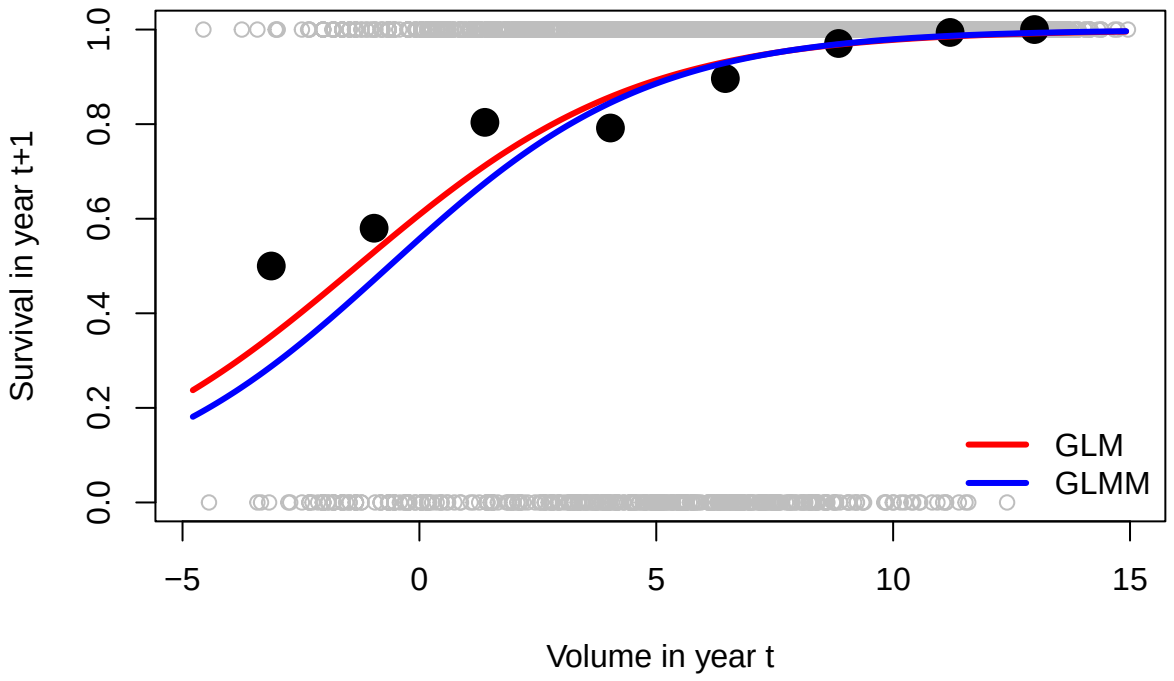
```
surv_glmm <- glmer(Survival_t1~log(vol_t)+(log(vol_t)|Plot),family="binomial",data=cactus)
summary(surv_glmm)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: Survival_t1 ~ log(vol_t) + (log(vol_t) | Plot)
##    Data: cactus
##
##      AIC      BIC   logLik deviance df.resid
##   2452.5   2486.6  -1221.3   2442.5     6684
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -11.0905   0.1125   0.1502   0.2240   1.9555
##
## Random effects:
##  Groups Name        Variance Std.Dev. Corr
##  Plot   (Intercept) 0.319785 0.56550
##         log(vol_t)  0.004842 0.06958  -1.00
## Number of obs: 6689, groups:  Plot, 11
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.23069    0.24725   0.933    0.351
## log(vol_t)   0.36363    0.03268  11.128   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##            (Intr)
```

```
## log(vol_t) -0.961
## optimizer (Nelder_Mead) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

How different is this mixed model to the predictions of the fixed effect model that we started with? Pretty different. Which one should we trust? All else equal, mixed models are preferable. By assigning more sources of variation, often with 'nuisance' variables like plot, random effects help improve resolution of the fixed effect(s) of interest.

```
plot(cactus$Survival_t1 ~ log(cactus$vol_t), xlab = "Volume in year t", ylab = "Survival in year t+1", c
points(surv_bin$mean_size, surv_bin$mean_surv, pch=16, cex=2)
lines(x_dummy,invlogit(coef(survival_model)[1]+coef(survival_model)[2]*x_dummy),col="red",lwd=3)
lines(x_dummy,invlogit(fixef(surv_glmm)[1]+fixef(surv_glmm)[2]*x_dummy),col="blue",lwd=3)
legend("bottomright",bty="n",lty=1,col=c("red","blue"),lwd=3,legend=c("GLM","GLMM"))
```



Whether or not you should be working with mixed models depends entirely on your data, so this is something to discuss.

Have fun! And let me know what questions arise. -Tom