Multinomial Regression

Regarding your data, what you have is a response with multiple categories, and anytime you are trying to model a response which is categorical you are right to try and use some type of generalized linear model (GLM). In your case you have additional information which you must take into account regarding your response and that is that your response levels have a natural ordering good ¿ middle ¿ bad, notice how this is different from trying to model a response such as what color balloon someone is likely to buy (red/blue/green), these values have no natural ordering. When doing this type of model with an ordered response you may want to consider using a proportional odds model.

I haven't used it myself, but the polr() function in the MASS package is likely to be of some use, alternatively I have used the lrm() function in the rms package to do similar types of analysis, and have found it quite useful. If you load these packages just use ?polr or ?lrm for the function information.

Alright enough background, on to your questions:
This should be covered above, check out these packages/functions and read up on ordinal logistic regression and proportional odds models
Any time you have a covariate which is categorical (Race/Sex/Hair color) you want to treat these as 'factors' in your R coding in order to model them appropriately. It's important to know what a factor is and how they are treated, but essentially you treat each category as a separate level and then model them in an appropriate way. Just read up on factors in models and you should be able to tease out whats going on. Keep in mind that treating categorical variables as factors is not unique to glm models or proportional odds models, but is typically

Missing values can sometimes be a hassle to deal with but if you're doing a fairly basic analysis its probably safe to just remove data rows which contain missing values (this isn't always true, but based on your current experience level I'm guessing you need not be concerned with the specifics of when and how to deal with missing values). In fact this is pretty much what R does. If you have a data which you are using to model, if you are missing information in a row for your response or any covariate in the model R is just going to exclude this data (this is the warning your seeing).

Obviously if you're excluding a large proportion of your data due to missingness, your results could be biased and its probably good to try and get some more info on why there are so many missing values, but if you're missing 162 observations in 10,000 rows of data I wouldn't sweat it too much. You can google up on methods for handling missing data if you're interested in some more specifics.

Almost all R model objects (lm, glm, lrm,...) will have an associated predict() function which will allow you to calculate the predicted values for your current modeling dataset and additionally for another dataset which you wish to predict an outcome for. Just search ?predict.glm or ?predict.lm to try and get some more info for whatever model type you want to work with. This is a very typical thing people wish to do with models so rest assured that there are some built in functions and methods that should make doing this relatively straightforward.

Ordered logistic regression `R`

Below we use the `polr` command from the MASS package to estimate an ordered logistic regression model. The command name comes from proportional odds logistic regression, highlighting the proportional odds assumption in our model.

Ordered logistic regression `R`

`polr` uses the standard formula interface in R for specifying a regression model with outcome followed by predictors. We also specify `Hess=TRUE` to have the model return the observed information matrix from optimization (called the Hessian) which is used to get standard errors.

Ordered logistic regression R

```
## fit ordered logit model and store results
m <- polr(apply ~ pared +
          public + gpa, data = dat, Hess=TRU

## view a summary of the model
summary(m)
## Call:
## polr(formula = apply ~ pared +
              public + gpa, data = dat, Hess
```

Ordered logistic regression R

```
## Coefficients:
##              Value Std. Error t value
## pared    1.0477      0.266    3.942
## public -0.0588       0.298   -0.197
## gpa      0.6159      0.261    2.363
##
## Intercepts:
##                                 Value   Std. E
## unlikely|somewhat likely         2.204   0.780
## somewhat likely|very likely      4.299   0.804
##
## Residual Deviance: 717.02
```

Ordered logistic regression R

In the output above, we see
Call, this is R reminding us what type of model we ran, what options we specified, etc. Next we see the usual regression output coefficient table including the value of each coefficient, standard errors, and t value, which is simply the ratio of the coefficient to its standard error. There is no significance test by default.

Ordered logistic regression R

Next we see the estimates for the two intercepts, which are sometimes called cutpoints. The intercepts indicate where the latent variable is cut to make the three groups that we observe in our data. Note that this latent variable is continuous. In general, these are not used in the interpretation of the results. The cutpoints are closely related to thresholds, which are reported by other statistical packages.

Ordered logistic regression R

- Finally, we see the residual deviance, -2 * Log Likelihood of the model as well as the AIC. Both the deviance and AIC are useful for model comparison.
- Some people are not satisfied without a p value. One way to calculate a p-value in this case is by comparing the t-value against the standard normal distribution, like a z test.

Ordered logistic regression R

Of course this is only true with infinite degrees of freedom, but is reasonably approximated by large samples, becoming increasingly biased as sample size decreases. This approach is used in other software packages such as Stata and is trivial to do. First we store the coefficient table, then calculate the pvalues and combine back with the table.

Ordered logistic regression R

```
## store table
(ctable <- coef(summary(m)))
##                                        Value Std. Error t
## pared                                1.04769     0.2658  3
## public                              -0.05879     0.2979 -0
## gpa                                  0.61594     0.2606  2
## unlikely|somewhat likely             2.20391     0.7795  2
## somewhat likely|very likely          4.29936     0.8043  5
## calculate and store p values
p <- pnorm(abs(ctable[, "t value"]), lower.tail = FAL
```

Ordered logistic regression R

```
## combined table
(ctable <- cbind(ctable, "p value" = p))
##                                        Value Std. Error t
## pared                                1.04769     0.2658  3
## public                              -0.05879     0.2979 -0
## gpa                                  0.61594     0.2606  2
## unlikely|somewhat likely             2.20391     0.7795  2
## somewhat likely|very likely          4.29936     0.8043  5
```

We can also get confidence intervals for the parameter estimates. These can be obtained either by profiling the likelihood function or by using the standard errors and assuming a normal distribution. Note that profiled CIs are not symmetric (although they are usually close to symmetric). If the 95

```
(ci <- confint(m))
# default method gives profiled CIs
## Waiting for profiling to be done...
##           2.5 % 97.5 %
## pared    0.5282 1.5722
## public -0.6522 0.5191
## gpa      0.1076 1.1309
confint.default(m) # CIs assuming normality
##           2.5 % 97.5 %
## pared    0.5268  1.569
## public -0.6426  0.525
## gpa      0.1051  1.127
```

The CIs for both pared and gpa do not include 0; public does. The estimates in the output are given in units of ordered logits, or ordered log odds. So for pared, we would say that for a one unit increase in pared (i.e., going from 0 to 1), we expect a 1.05 increase in the expect value of apply on the log odds scale, given all of the other variables in the model are held constant.

For gpa, we would say that for a one unit increase in gpa, we would expect a 0.62 increase in the expected value of apply in the log odds scale, given that all of the other variables in the model are held constant.

The coefficients from the model can be somewhat difficult to interpret because they are scaled in terms of logs. Another way to interpret logistic regression models is to convert the coefficients into odds ratios. To get the OR and confidence intervals, we just exponentiate the estimates and confidence intervals.

```
## odds ratios
exp(coef(m))
##  pared public    gpa
## 2.8511 0.9429 1.8514
## OR and CI
exp(cbind(OR = coef(m), ci))
##              OR  2.5 % 97.5 %
## pared  2.8511 1.6958  4.817
## public 0.9429 0.5209  1.681
## gpa    1.8514 1.1136  3.098
```

These coefficients are called proportional odds ratios and we would interpret these pretty much as we would odds ratios from a binary logistic regression. For pared, we would say that for a one unit increase in parental education, i.e., going from 0 (Low) to 1 (High), the odds of "very likely" applying versus "somewhat likely" or "unlikely" applying combined are 2.85 greater, given that all of the other variables in the model are held constant.

Likewise, the odds "very likely" or "somewhat likely" applying versus "unlikely" applying is 2.85 times greater, given that all of the other variables in the model are held constant. For gpa (and other continuous variables), the interpretation is that when a student's gpa moves 1 unit, the odds of moving from "unlikely" applying to "somewhat likely" or "very likley" applying (or from the lower and middle categories to the high category) are multiplied by 1.85.

Ordered logistic regression R

Turning our attention to the predictions with public as a predictor variable, we see that when public is set to "no" the difference in predictions for apply greater than or equal to two, versus apply greater than or equal to three is about 2.14 (-0.204 - -2.345 = 2.141). When public is set to "yes" the difference between the coefficients is about 1.37 (-0.175 - -1.547 = 1.372).

Ordered logistic regression R

The differences in the distance between the two sets of coefficients (2.14 vs. 1.37) may suggest that the parallel slopes assumption does not hold for the predictor public. That would indicate that the effect of attending a public versus private school is different for the transition from "unlikely" to "somewhat likely" and "somewhat likely" to "very likely."

Ordered logistic regression R

The plot command below tells R that the object we wish to plot is s. The command which=1:3 is a list of values indicating levels of y should be included in the plot. If your dependent variable had more than three levels you would need to change the 3 to the number of categories (e.g. 4 for a four category variable, even if it is numbered 0, 1, 2, 3). The command pch=1:3 selects the markers to use, and is optional, as are xlab='logit' which labels the x-axis, and main=' ' which sets the main label for the graph to blank.

Ordered logistic regression R

If the proportional odds assumption holds, for each predictor variable, distance between the symbols for each set of categories of the dependent variable, should remain similar. To help demonstrate this, we normalized all the first set of coefficients to be zero so there is a common reference point.

Ordered logistic regression R

Looking at the coefficients for the variable pared we see that the distance between the two sets of coefficients is similar. In contrast, the distances between the estimates for public are different (i.e. the markers are much further apart on the second line than on the first), suggesting that the proportional odds assumption may not hold.

Ordered logistic regression R

Ordered logistic regression R

```
plot(s, which=1:3, pch=1:3, xlab='logit', ma
```

Ordered logistic regression R

Plot viewing proportional odds assumption
Once we are done assessing whether the assumptions of our model hold, we can obtain predicted probabilities, which are usually easier to understand than either the coefficients or the odds ratios. For example, we can vary gpa for each level of pared and public and calculate the probability of being in each category of apply. We do this by creating a new dataset of all the values to use for prediction.

Ordered logistic regression R

```
newdat <- data.frame(
  pared = rep(0:1, 200),
  public = rep(0:1, each = 200),
  gpa = rep(seq(from = 1.9, to = 4, length.o

newdat <- cbind(newdat, predict(m, newdat, t
```

Ordered logistic regression R

```
##show first few rows
head(newdat)
##   pared public   gpa unlikely somewhat li
## 1     0      0 1.900   0.7376          0.
## 2     1      0 1.921   0.4932          0.
## 3     0      0 1.942   0.7325          0.
## 4     1      0 1.964   0.4867          0.
## 5     0      0 1.985   0.7274          0.
## 6     1      0 2.006   0.4802          0.
```

Ordered logistic regression R

Now we can reshape the data long with the reshape2 package and plot all of the predicted probabilities for the different conditions. We plot the predicted probilities, connected with a line, coloured by level of the outcome, apply, and facetted by level of pared and public. We also use a custom label function, to add clearer labels showing what each column and row of the plot represent.

Ordered logistic regression R

```
lnewdat <- melt(newdat, id.vars = c("pared",
  variable.name = "Level", value.name="Proba
%## view first few rows
%head(lnewdat)
%##   pared public   gpa     Level Probabilit
%## 1     0      0 1.900 unlikely       0.737
%## 2     1      0 1.921 unlikely       0.493
%## 3     0      0 1.942 unlikely       0.732
%## 4     1      0 1.964 unlikely       0.486
%## 5     0      0 1.985 unlikely       0.727
%## 6     1      0 2.006 unlikely       0.480
```

Ordered logistic regression R

```
ggplot(lnewdat, aes(x = gpa, y = Probability
  geom_line() +
  facet_grid(pared ~ public, scales="free",
    labeller=function(x, y) sprintf("%s = %d
```

Ordered logistic regression R

Plot of predicted probabilities for each group Things to consider Perfect prediction: Perfect prediction means that one value of a predictor variable is associated with only one value of the response variable. If this happens, Stata will usually issue a note at the top of the output and will drop the cases so that the model can run.

Ordered logistic regression R

Sample size: Both ordered logistic and ordered probit, using maximum likelihood estimates, require sufficient sample size. How big is big is a topic of some debate, but they almost always require more cases than OLS regression. Empty cells or small cells: You should check for empty or small cells by doing a crosstab between categorical predictors and the outcome variable. If a cell has very few cases, the model may become unstable or it might not run at all.

Ordered logistic regression R

Pseudo-R-squared: There is no exact analog of the R-squared found in OLS. There are many versions of pseudo-R-squares. Please see Long and Freese 2005 for more details and explanations of various pseudo-R-squares. **Diagnostics:** Doing diagnostics for non-linear models is difficult, and ordered logit/probit models are even more difficult than binary models.

Negative Binomial Regression with R

Negative binomial regression is for modeling count variables, usually for over-dispersed count outcome variables.

This page uses the following packages. Make sure that you can load them before trying to run the examples on this page. If you do not have a package installed, run: install.packages("packagename"), or if you see the version is out of date, run: update.packages().

```
require(foreign)
require(ggplot2)
require(MASS)
```

Negative Binomial Regression with R

Please note: The purpose of this page is to show how to use various data analysis commands. It does not cover all aspects of the research process which researchers are expected to do. In particular, it does not cover data cleaning and checking, verification of assumptions, model diagnostics or potential follow-up analyses.

Negative Binomial Regression with R

Examples of negative binomial regression Example 1. School administrators study the attendance behavior of high school juniors at two schools. Predictors of the number of days of absence include the type of program in which the student is enrolled and a standardized test in math.
Example 2. A health-related researcher is studying the number of hospital visits in past 12 months by senior citizens in a community based on the characteristics of the individuals and the types of health plans under which each one is covered.

Negative Binomial Regression with R

Description of the data Let's pursue Example 1 from above.
We have attendance data on 314 high school juniors from two urban high schools in the file **nb_data**. The response variable of interest is days absent, daysabs. The variable math gives the standardized math score for each student. The variable prog is a three-level nominal variable indicating the type of instructional program in which the student is enrolled.

Negative Binomial Regression with R

Let's look at the data. It is always a good idea to start with descriptive statistics and plots.

```
dat <- read.dta("http://www.ats.ucla.edu/sta
dat <- within(dat, {
prog <- factor(prog, levels = 1:3, labels =
id <- factor(id)
})
```

Negative Binomial Regression with R

```
summary(dat)
##          id          gender          math
##  1001    :  1   female:160   Min.   : 1.0
##  1002    :  1   male  :154   1st Qu.:28.0
##  1003    :  1                Median :48.0
##  1004    :  1                Mean   :48.3
##  1005    :  1                3rd Qu.:70.0
##  1006    :  1                Max.   :99.0
##  (Other):308
##          prog
##  General    : 40
##  Academic   :167
```

Negative Binomial Regression with R

```
ggplot(dat, aes(daysabs, fill = prog)) + geom_
., margins = TRUE, scales = "free")
```

Histogram plots showing distribution of the data
Each variable has 314 valid observations and their distributions seem quite reasonable. The unconditional mean of our outcome variable is much lower than its variance.

Negative Binomial Regression with R

```
newdata1 <- data.frame(math = mean(dat$math),
labels = levels(dat$prog)))
newdata1$phat <- predict(m1, newdata1, type
newdata1
##    math        prog   phat
## 1 48.27     General 10.237
## 2 48.27   Academic  6.588
## 3 48.27 Vocational  2.850
```

Negative Binomial Regression with R

In the output above, we see that the predicted number of events (e.g., days absent) for a general program is about 10.24, holding math at its mean. The predicted number of events for an academic program is lower at 6.59, and the predicted number of events for a vocational program is about 2.85.

Negative Binomial Regression with R

Below we will obtain the mean predicted number of events for values of math across its entire range for each level of prog and graph these.

Negative Binomial Regression with R

```
newdata2 <- data.frame(
math = rep(seq(from = min(dat$math), to = ma
prog = factor(rep(1:3, each = 100), levels =
levels(dat$prog)))
```

Negative Binomial Regression with R

```
newdata2 <- cbind(newdata2, predict(m1, newdat
newdata2 <- within(newdata2, {
DaysAbsent <- exp(fit)
LL <- exp(fit - 1.96 * se.fit)
UL <- exp(fit + 1.96 * se.fit)
})
```

Negative Binomial Regression with R

Negative Binomial Regression with R

```
ggplot(newdata2, aes(math, DaysAbsent)) +
geom_ribbon(aes(ymin = LL, ymax = UL, fill =
geom_line(aes(colour = prog), size = 2) +
labs(x = "Math Score", y = "Predicted Days A
```

Negative Binomial Regression with R

Negative Binomial Regression with R

Plot of the model predicted days absent with confidence intervals The graph shows the expected count across the range of math scores, for each type of program along with 95 percent confidence intervals. Note that the lines are not straight because this is a log linear model, and what is plotted are the expected values, not the log of the expected values.

Negative Binomial Regression with R

Things to consider It is not recommended that negative binomial models be applied to small samples. One common cause of over-dispersion is excess zeros by an additional data generating process. In this situation, zero-inflated model should be considered.

Negative Binomial Regression with R

If the data generating process does not allow for any 0s (such as the number of days spent in the hospital), then a zero-truncated model may be more appropriate.

Negative Binomial Regression with R

Count data often have an exposure variable, which indicates the number of times the event could have happened. This variable should be incorporated into your negative binomial regression model with the use of the offset option. See the glm documentation for details.

Negative Binomial Regression with R

The outcome variable in a negative binomial regression cannot have negative numbers. You will need to use the `m1$resid` command to obtain the residuals from our model to check other assumptions of the negative binomial model (see Cameron and Trivedi (1998) and Dupont (2002) for more information).