R / Bioconductor: A Short Course

James H. Bullard
Sandrine Dudoit
Division of Biostatistics, UC Berkeley
www.stat.berkeley.edu/~bullard
www.stat.berkeley.edu/~sandrine

Cuernevaca, Mexico
January 21-25, 2008

Statistical Data Analysis with R

Probability Models

- After exploratory data analysis we often want to investigate the applicability of various “probability” models for the data. Classic examples include statistical tests for equal means (t-tests and z-tests), normal linear regression models, generalized linear regression models, and χ^2 tests.
- R provides a wealth of tools for statistical modeling

Example

We want to compute pairwise t-tests for equal means between each drug category in our viral load data set.

- What are the assumptions of this test? How can we test them?
- How do the results compare with R's builtin t-test?
- Is this a "smart" thing to do?

$$\frac{\bar{X}_{AZT} - \bar{X}_{ABC}}{\sqrt{S^2_{AZT}/N_{AZT} + S^2_{ABC}/N_{ABC}}} \tag{1}$$

Testing Normality

- Clearly the "raw" viral load data is not normally distributed.
- What are some plots we can make to assess normality of the transformed data.

```
> plot(density(vl <- log(viralLoad$viral.load)))
> rng <- range(vl)
> points(s <- seq(rng[1], rng[2],
+     length = 1000), dnorm(s, mean = mean(vl),
+     sd = sd(vl)), col = "red",
+     pch = ".")
> qqnorm(vl)
> qqline(vl, col = "red")
```

Testing Normality

Example

Using R test the assumption of normality using both the Kolmogorov-Smirnov test as well as the Shapiro-Wilks test.

1. What are the functions for doing this in R?
2. Do these tests give different results?
3. Why?

We want to investigate these tests using simulation. Simulate 1000 data sets from a normal distribution with 50 observations and test each data set for normality using both tests. What does the distribution of p-values look like? Does this make sense? Which test is more sensitive? Comment on type I error, how do we get to type II error?

Tests of Associations

- Very frequently we are presented with categorical data. A good example of this is genotype data. In this setting, we have a large number of genotypes for each person in our study along with a set of phenotypes which were are interested in studying.
- A test which is often used in this setting to assess the association between genotype and phenotype is a χ^2 test of independence.

	AA	AB	BB
0	192	437	181
1	8	68	114

χ^2 Tests

$$\chi^2_{K-1} = \sum_k^K (O_k - E_k)^2 / E_k \tag{2}$$

Here E_k is the expected number of observations under the independence model, and O_k is the actual number of observations in category k.

Example

We would like to perform a χ^2 test on our genotype data. This will be a test for independence.

1. Implement a function which takes a `table` and returns a list containing both a χ^2 statistic for the test of independence as well as a p-value for this statistic.
2. Compare this result to the builtin `R` function, are they the same? What does this tell us about our data?

The Linear Regression Model

- The linear regression model is one of the most common, if not, the most common way of modeling data.
- In many cases the model is not “correct” but is often very reasonable.

$$Y = X\beta + \epsilon \tag{3}$$

The linear regression model is composed of an $n \times p$ design matrix (X), an $n \times 1$ vector of outcomes (Y), a $p \times 1$ vector of parameters which we wish to estimate (generally denoted $\hat{\beta}$). Linear regression finds the estimate $\hat{\beta}$ which minimizes the L_2 loss (equation: (4)).

$$L_2(\beta) = \sum_{i=1}^{n} (Y - X\beta)^2 \tag{4}$$

The Linear Regression Model

Under the following assumptions linear regression is the best linear unbiased estimator of β.

i. X and Y satisfy equation (3).
ii. The disturbance terms ϵ_i are i.i.d with mean 0 and variance σ^2.
iii. X and ϵ are independent.

Linear Models

- In the Introduction we fit a linear model using linear algebra. While this is a good exercise `R` has a large set of functions to facilitate fitting statistical models.
- We want to be wary of using linear algebra to fit a model, as it is often the case that the builtin methods of `R` will be much faster and more numerically stable.
- Statistical models in R have a special syntax (the formula syntax):

```
Y ~ X
```

- This says that the variable Y is related to X. The formula specification is used in a variety of functions as input and depending on that function different relationships between the predictor variables (X) and the outcome variables (Y) are assumed.

Formulas Continued

The simplest data set to begin to play with the formula functions in R can be generated as follows:

```
> N <- 100
> X <- runif(N, 20, 40)
> Y <- 3 + X * 2 + rnorm(N, mean = 0,
+      sd = 5)
```

Now suppose we would like to fit a linear model to the data. In R this is as simple as:

```
> lm.1 <- lm(Y ~ X)
> lm.1.int <- lm(Y ~ 1 + X)
```

1. What is the class of `lm.1` and `lm.1.int`?
2. How can we extract the estimates $\hat{\beta}$?
3. What are the functions which are specialized for this class (hint methods)?

Formulas

As the above model is not that interesting we might be inclined to to have a look at some more interesting data sets. Let's have another look at our viral load data set.

```
> vL <- read.table("../../data/viral-load.dta")
> lm.vL <- lm(viral.load ~ age +
+     meds + infected, data = vL)
```

1. Is this a sensible thing to do?
2. What are the estimates of the coefficients?
3. What happened with meds?
4. How do we transform the data to get on safer ground? (hint: try to put the log directly in the formula), try to square the age covariate.

Residual Analysis

- As you might expect the `lm` object has plot and summary methods. To get help on these directly it is usually possible to do `?plot.lm`

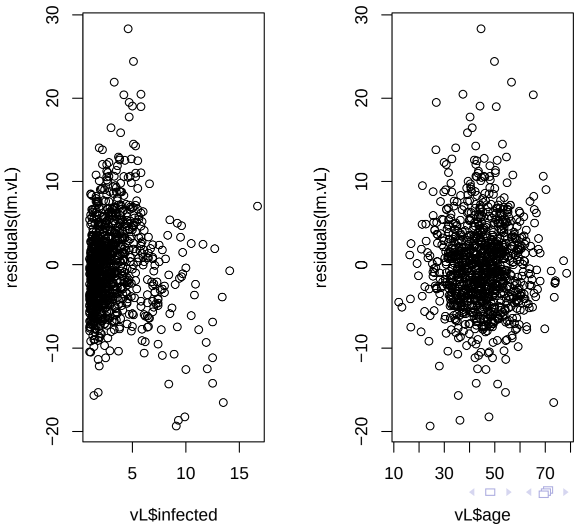
```
> plot(lm.vL)
```

We want to plot the residuals against some of the covariates, remember that residuals are an estimate of the noise term ϵ. We assumed that ϵ was vector of I.I.D. random disturbances and independent of X, what will an age versus residual plot potentially show us?
Is there any trend to this data? What tools from our EDA work can we use here to make any trend stand out?

Residual Analysis

```
> lm.vL <- lm(log(viral.load) ~ age +
+     meds + infected, data = vL)
> par(mfrow = c(1, 2))
> plot(vL$infected, residuals(lm.vL))
> plot(vL$age, residuals(lm.vL))
```

Residual Analysis



Interaction

Suppose based on your EDA work you think that there should be an interaction term in the model for meds and infected. The formula syntax provides a very handy way of modeling this.

```
> lm.vL.interaction <- lm(log(viral.load) ~
+      age + meds * infected, data = vL)
> anv <- anova(lm.vL, lm.vL.interaction)
```

If we look at the coefficients of the model we see what happens, the '*' notation in the formula essentially produces both the main effects terms as well as the interaction terms. Had we just used a ':', then only the interaction terms would have been in the model.

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	994	29254.90				
2	991	29067.44	3	187.46	2.13	0.0948

xtable: a handy little tool

- Often we want to produce tables which are suitable for presentation, such as the ANOVA table produced above.
- xtable - in package xtable can be used to produce really nice latex tables which can be immediately embeded in a .Rnw document.
- xtable - can also produce HTML tables.

```
> library(xtable)
> dta <- data.frame(rbinom(1100,
+     prob = 0.2, size = 10), rbinom(1100,
+     prob = 0.2, size = 10))
> print(xtable(table(dta)), type = "html")
```

Formulas: The last word

- A comprehensive reference of how to specify different formulae in R can be found at: formulas

```
Y ~ M
    Y is modeled as M.
M_1 + M_2
    Include M_1 and M_2.
M_1 - M_2
    Include M_1 leaving out terms of M_2.
M_1 : M_2
    The tensor product of M_1 and M_2. If both terms
    the subclasses factor.
M_1 %in% M_2
    Similar to M_1:M_2, but with a different coding.
```

Formulas: The last word

```
M_1 * M_2
    M_1 + M_2 + M_1:M_2.
M_1 / M_2
    M_1 + M_2 %in% M_1.
M^n
    All terms in M together with interactions
    up to order n
I(M)
    Insulate M. Inside M all operators have their
    normal arithmetic meaning, and that term
    appears in the model matrix.
```

This was lifted right from that page!

Least Squares and Maximum Likelihood

Up to now we have been pretty loose about what exactly we are modeling when we do linear regression. All we have really said is that we are going to choose a $\hat{\beta}$ such that it satisfies equation (5).

$$\hat{\beta}_{LS} = \arg\min_{\beta} \sum (Y - X\beta)^2 \tag{5}$$

If we make the following additional assumption about the distribution of the error term ϵ:

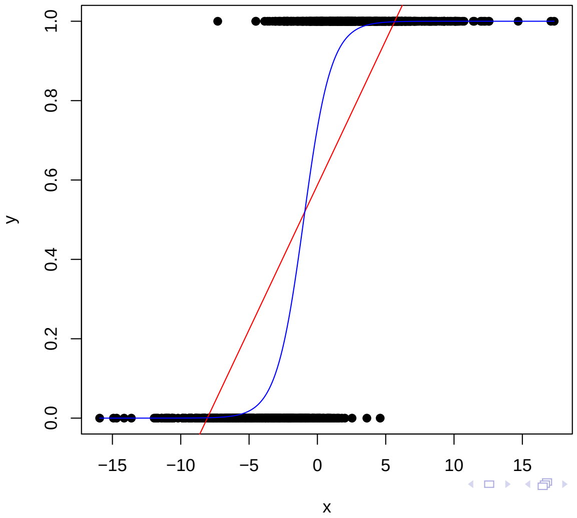
$$\epsilon_i \sim N(0, \sigma^2) \tag{6}$$

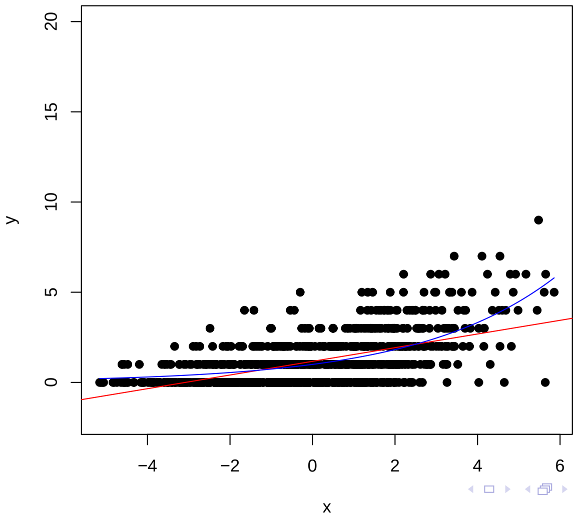
It turns out that minimizing equation (5) gives us the maximum likelihood solution for β as well. This is good news as maximum likelihood solutions have a number of nice properties.

Troublesome Data

In the normal linear regression model we model Y as a linear function of β. This is appropriate in many contexts, however one thing to note is that our $X\hat{\beta}$, or fitted values are unconstrained. Given the following two graphs what problems might arise if we use the linear regression model to model the outcome y as a linear function: $a + xb$?

Troublesome Data




y
20
15
10
5
0
−4
−2
0
2
4
6
x

Generalized Linear Models

- If we assume that the ϵ from equation (3) are $N(0, \sigma^2)$ then the conditional distribution of Y is: $Y|X \sim N(X\beta, \sigma^2)$. That is, Y is conditionally normally distributed with mean $X\beta$ given X. We found estimates of β using least squares, however under the assumption of conditional normality of Y we could also have found $\hat{\beta}$ by maximizing the likelihood. In this case the answers will be the same!

$$\hat{\beta}_{ML} = \arg\max_{\beta \in \beta} \prod_{i=1}^{N} \phi(y_i; \beta) \tag{7}$$

Here β is defined as: $\beta = \{\mu, \sigma\}$ and ϕ is the density function for a normal distribution (for simplicity I have suppressed the dependence on x_i, but this is constant

Generalized Linear Models

during maximization). Again, maximum likelihood says: "Choose values for the parameters so that I maximize the probability of seeing the data."

In the case of count data we might think to model the conditional distribution of Y as Poisson random variable with some mean λ.

$$P(Y = k|X) = \frac{e^{-\lambda}\lambda^{k}}{k!} \tag{8}$$

The key insight is to model λ as a linear function of our data and some parameter vector β, i.e.:

$$\lambda ? \triangleq ? X\beta \tag{9}$$

This doesn't work however because we have the same problem, namely λ must be greater than 0. This is okay, we just replace

Generalized Linear Models

$X\beta$ with $e^{X\beta}$ and we guarantee that λ is always positive. This leads us to the typical way of expressing a "Poisson regression"

$$log(E[Y|X]) = X\beta \tag{10}$$

We recall that the mean of a Poisson random variable with parameter λ is just λ itself, therefore the above equation makes sense.
Again, back to linear regression:

$$E[Y|X] = X\beta \tag{11}$$

And in general we have something like:

$$E[Y|X] = f(X\beta) \tag{12}$$

Generalized Linear Models

$f(\cdot)$ is called the response function. Once we have these pieces in place we can understand how to completely specify our generalized linear model in R. In summary there are three components in a glm:[1]

1. The data X enters the model linearly via: $\eta \triangleq X\beta$
2. The conditional mean $E[Y|X]$ is represented as a function $f(X\beta)$.
3. The response variable y can be characterized by an exponential family distribution.

[1] These are taken from Michael I. Jordan's book in progress.

Logistic and Poisson Regression

- So now that we understand the basic structure of the generalized linear model we have to understand a little more about what is going on in two specific, but very important examples. Below are the two response functions for Logistic and Poisson Regression.
- Logistic Regression

$$f_{lr}(X\beta) = \frac{1}{(1 + e^{-X\beta})} \tag{13}$$

- Poisson Regression

$$f_{pr}(X\beta) = \exp(X\beta) \tag{14}$$

Logistic and Poisson Regression

We assume that the predictor variables form a linear combination with the parameters we wish to estimate and then that linear combination gets pumped through f to produce a mean, this is the mean parameter of our response variable Y.

- It is important to summarize here. We have data which is certainly not normally distributed, and worse the range of the data is not $\mathbb{R}$. Therefore we want to "model" the data in a different way. In the specific case of logistic regression we have a $0, 1$ outcome and thus a sensible model would be one that assigned a probability to each subject of being a 0 or a 1 based on the values of their covariates. GLMs essentially make the constraint that the parameters: β must enter the model linearly, i.e. $\eta = X\beta$. When we transform η using the response function we will have done

Logistic and Poisson Regression

so according to our modeling choice, i.e. the type of Y (real numbers, counts, binary outcomes, rates). The parameter estimates $\hat{\beta}$ have a different interpretation in terms of the "units" of Y than they did for least squares, however one can say (quite loosely) that a large positive $\hat{\beta}_k$ for the kth covariate indicates that larger values of x_k will mean larger values of Y on the appropriate scale.

- Things will be clearer when we see GLMs in practice.

GLM: generating data

Example

In order to play a little with GLM and understand it a little more deeply we want to generate both a binary data set and count data set. I would like the linear component of the model to be specified as below. Make some plots of your generated data.

$$\eta = -20 + .1age + .2weight + cd4 \tag{15}$$

We are going to specify the covariates as follows:

$$\begin{aligned}
\text{age} &\sim \text{Uniform}(20, 40) && (16)\\
\text{weight} &\sim \text{Norm}(65, 10) && (17)\\
\text{cd4} &\sim \text{exponential}(1/4) && (18)\\
& && (19)
\end{aligned}$$

GLM: fitting

- Now that we have generated the data and understood the model we can "fit" the model - Again we are going to be estimating some parameters β - What these parameters mean in this context is a little bit more complicated to sort out, however we'll interpret the parameters once we get them.
- In order to fit the model we do:

```
> glm.binary <- glm(Y.binary ~ age +
+      weight + cd4, family = binomial())
> glm.count <- glm(Y.count ~ age +
+      weight + cd4, family = poisson())
```

The key thing here is that we have specified the "family" - that is the conditional distribution of Y given X.

GLM: Fitting

One thing to keep in mind is that there is not, in general, a closed form solution for the maximum likelihood estimates of the parameters β. Most of the time the algorithms we use an iteratively reweighted least squares to do the fitting (glm uses this).

```
> glm.binary <- glm(Y.binary ~ age +
+     weight + cd4, family = binomial(),
+     control = glm.control(maxit = 100,
+         epsilon = 0.1, trace = TRUE))
```

1. What class does glm return?
2. How can we make a plot of the fitted values verus the residuals? Is this a good plot?

ALL data set

Example

The ALL data package is a Bioconductor data package containing gene expression measures (RMA) for 12,625 genes for 128 patients; 95 of the patients have been diagnosed with B-cell leukemia, and 33 patients have been diagnosed with T-cell leukemia. We are interested in finding genes which are predictive of leukemia type. In addition, we also have a number of classic covariates from the patients which we might be interested in examining.

We will need to do the following:

1. Download and install the data package from Bioconductor

ALL data set

2. Determine which patients have which type of leukemia, in addition understand what other covariates might be important to have a look at.
3. Do a small EDA to determine whether we might think about including the phenotypic covariates present (use some of the nice smoothing techniques we learned last time).
4. Make density plots for the gene expression levels for each patient.
5. Determine a set of "important genes" using logistic regression, once this set has been determined fit a model with your important genes.
6. Classification with cross-validation to test your predictive model!

Numerical Optimization

- As mentioned before we are computing maximum likelihood estimates of our parameter vector β - Although glm is really the way to go in this context often we need to perform direct numerical optimization of some objective function.
- In the current context we can explore a couple of `R` utilities for performing maximization or minimization of an objective function. Here our objective function will be the log likelihood which is displayed in equation (20) for the binomial outcome.

$$\sum_{i=1}^{n} y_i \log(f(x_i^t\beta)) + (1 - y_i)\log(1 - f(x_i^t\beta)) \tag{20}$$

Numerical Optimization: R

The main R functions for numerical optimization are the following:[2]

1. optim General-purpose optimization based on Nelder-Mead, quasi-Newton and conjugate-gradient algorithms. It includes an option for box-constrained optimization and simulated annealing.
2. nlm This function carries out a minimization of the function 'f' using a Newton-type algorithm. See the references for details.
3. nlminb Unconstrained and constrained optimization using PORT routines.

Numerical Optimization: R

4. optimize The function 'optimize' searches the interval from 'lower' to 'upper' for a minimum or maximum of the function 'f' with respect to its first argument.The method used is a combination of golden section search and successive parabolic interpolation.
5. uniroot The function 'uniroot' searches the interval from 'lower' to 'upper' for a root (i.e., zero) of the function 'f' with respect to its first argument.

[2]These descriptions are taken right out of the help files.

Using optim

```
> fx <- function(x) {
+       -return(4 + x^2)
+ }
> optimize(fx, c(-100, 100))$objective
[1] 4
```

1. Try to optimize a function of two variables (use optim instead).

Function Closure

- Function closure exists in a number of "high-level" languages, such as `scheme` and `Lisp`.
- We can think of functions as "objects" in that they have state. This is quite different from `C` and `perl` and is a very powerful concept.
- The idea is that a function has an "environment" which it was created with, generally this is the special environment `.GlobalEnv`

```
> a <- rnorm(100)
> fx <- function(x) {
+     mean(x * a)
+ }
> fx(10)
```

Function Closure

- As we have seen before functions can be defined within other functions. When this happens the function defined inside the outer function inherits it's environment.

```
> makeF <- function(vec) {
+     return(function(x) {
+         mean(x * vec)
+     })
+ }
> fx <- makeF(rnorm(100))
> fx(10)
```

Function Closure

Another example which demonstrates more clearly that functions have "state" and can be treated like objects is the following simple example of a function which returns a function which accepts a message which instructs the function what action to perform. This can be very useful if there is a high-cost of setting up some objects, for instance database connections.

```
> makeModelFitter <- function(X) {
+     xtxInv <- (solve(t(X) %*% X) %*%
+         t(X))
+     function(msg, ...) {
+         switch(msg, fit = {
+             Y <- list(...)[[1]]
+             if (is.null(Y))
+                 stop("fit is called with Y.")
```

Function Closure

```
+                 betas <<- (xtxInv %*%
+                     Y)
+             }, coef = {
+                 if (!exists("betas")) {
+                     stop("must call fit first")
+                 }
+                 return(betas)
+             })
+         }
+ }
```

Function Closure

This will be very useful in context of numerical optimization of a number of likelihood functions. The common problem can be seen when we look at functions such as optim which takes a function of the parameters which we wish to optimize. In addition to this we can see that the likelihood equation depends on x and y, as well as β. As we are optimizing we don't change the data, therefore we will use closure to generate a function which takes only the parameters which we will be optimizing over.

- The other option is to have X and Y as global variables, but good programmers avoid global variables as much as possible.

Example

Write a function which takes the data which we generated above to be used in the logisitic regression, something like:

```
> makeLikelihoodFunction <- function(x,
+     y) {
+     return(function(betas) {
+         return("something")
+     })
+ }
```

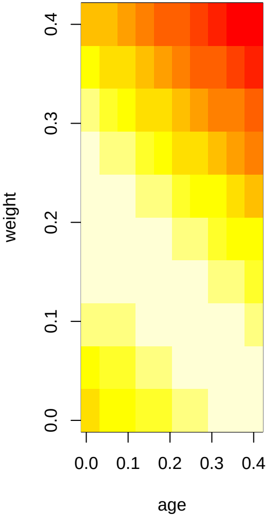
Use optim, and nlm to find the maximum likelihood estimates of β.

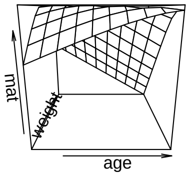
Likelihood Surfaces

When optimizing a function directly you have to pick sensible starting values, this is not always easy, but oftwen we can be aided by things like plots telling us where the likelihood is high. In the context of generalized linear models we have the luxury of a convex optimization problem, thus we can be confident that when we find the maximimizer it is the global maximizer.

```
> par(mfrow = c(1, 2))
> image(X, Y, mat, xlab = "age",
+       ylab = "weight")
> persp(X, Y, mat, xlab = "age",
+       ylab = "weight")
```

Likelihood Surfaces



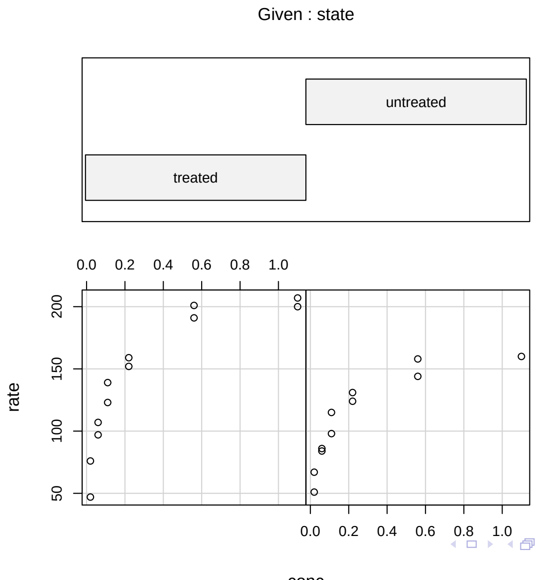


Non-linear modeling in R

- We have now seen both linear models as well as general linear models, however we have barely scratched the surface of things we might wish to model.
- We recall the puromycin data set which is a data set of rates of reaction for various concentration of enzyme in “treated” and “untreated” cells.

```
> data(Puromycin)
> coplot(rate ~ conc | state, data = Puromycin)
```

Non-linear modeling in R



NLS Continued

We could maybe model the rate as a function of the log concentration, however it turns out that the Michaelis-Menten model is the one that is really appropriate

$$\text{rate} = \frac{V_{max}\text{concentration}}{K_m + \text{concentration}} \tag{21}$$

We have data for rates and concentrations now we want to get estimates for the V_{max} which is the maximum initial velocity of the reaction, and K_m which is the Michaelis-Menten rate constant.

NLS

- In order to do maximum likelihood we need a probability model for the response variable rate, another option is to choose a loss function and then minimize that loss. We'll follow essentially the same path.

$$\arg\min_{\beta} = \sum_{i} (\mathrm{rate} - f(\beta; \mathrm{concentration}))^2 \qquad (22)$$

Here $f(\cdot)$ is an arbitrary function of β and concentration. We see we have another optimization problem which we can solve using one of the numerical optimization routines described above, or we can try to take advantage of the fact that we "know" $f(\cdot)$ up to the parameters - if we can take derivatives we might be able ease our optimization problems.

NLS

- The main idea behind non-linear least squares is to use a Taylor expansion of $f(\cdot)$ with respect to β around some initial β^0. We'll go through this for the Michaelis-Menten model.
- Before going on it is instructive to use optim to minimize the loss function and get estimates for our two parameters.

NLS Fit

```
> makeLF <- function(conc, rate) {
+     return(function(Vm, K) {
+         sum(rate - (Vm * conc/(K +
+             conc)))^2
+     })
+ }
> fx <- makeLF(Puromycin$conc, Puromycin$rate)
> makeImageMat <- function(N) {
+     mat <- matrix(NA, N, N)
+     X <- seq(0, 200, length = N)
+     Y <- seq(0, 0.1, length = N)
+     for (i in 1:N) {
+         for (j in 1:N) {
+             mat[i, j] <- fx(X[i],
```

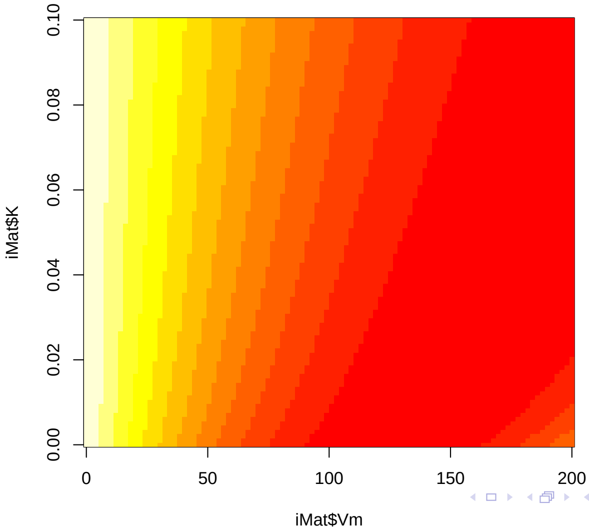
NLS Fit

```
+                   Y[j])
+           }
+       }
+       return(list(z = mat, Vm = X,
+           K = Y))
+ }
> iMat <- makeImageMat(100)
> matrixWhichMin <- function(mat) {
+       N <- nrow(mat)
+       mm <- which.min(mat)
+       c(mm%%N, (mm - (mm%%N))/N +
+           1)
+ }
> Vm.0 <- iMat$Vm[matrixWhichMin(iMat$z)[1]]
> K0 <- iMat$K[matrixWhichMin(iMat$z)[2]]
```

NLS Fit

```
> nl.P <- nls(rate ~ Vm * conc/(K +
+      conc), data = Puromycin, start = c(Vm = Vm.0,
+      K = K0), trace = FALSE)
> image(iMat$Vm, iMat$K, iMat$z)
```

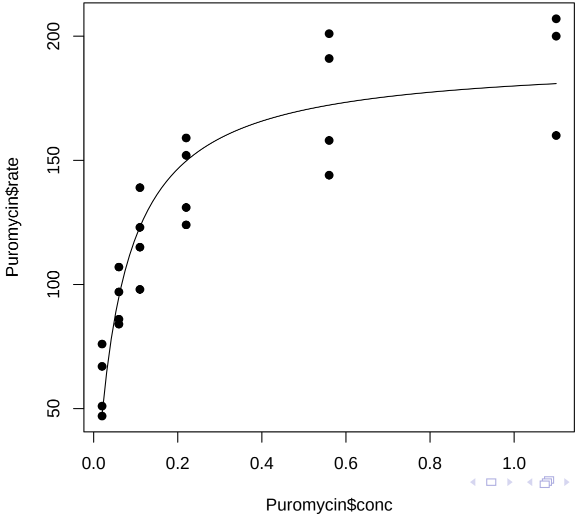
NLS Fit



NLS: fit

```
> makeCF <- function(Vm, K) {
+     return(function(conc) {
+         return((Vm * conc/(K +
+             conc)))
+     })
+ }
> plot(Puromycin$conc, Puromycin$rate,
+     pch = 16)
> conc <- seq(min(Puromycin$conc),
+     max(Puromycin$conc), length = 100)
> rateHat <- makeCF(coefficients(nl.P)[1],
+     coefficients(nl.P)[2])(conc)
> lines(conc, rateHat, type = "l")
```

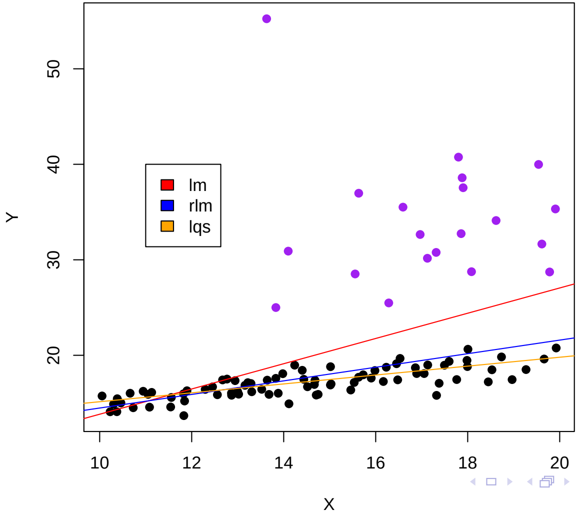
NLS: fit



MASS

- The R package MASS which accompanies the book "Modern Applied Statistics with S" has a wealth of useful tools for statistical modeling in R.
- One such tool is rlm which provides robust linear modeling.
- rlm uses m-estimation to fit linear models more robustly. lqs fits only the "good" data points. See the help file for more details.
- It is not always easy to see what is an outlier and what is not, but sometimes some knowledge of how the data was collected/what assay was used can help - For instance, it might be possible for a machine to give off spurious results at high values of it's dynamic range.

MASS



Online Resources

1. Vector General Additive models can be fit with the vgam package. This is probably the most popular way to fit multinomial outcomes in R
2. In addition to fitting both linear models as well as glms R also provides lme and nlme to fit mixed effects models (See also the fantastic pdf on these models in the resources directory by John Fox).