Bio Stats II : Lab 3

Acknowledgements: Punt & Branch Labs (U Washington)

Gavin Fay

02/10/2016

Lab schedule

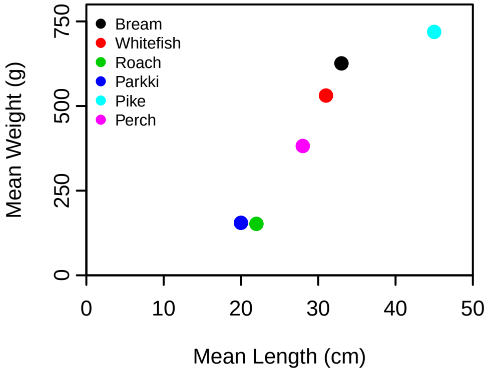
1/27: Introduction to R and R Studio, working with data
2/03: Intro to plotting, manipulating data
2/10: Probability, linear modeling, PCA
2/17: Programming practices, conditional statements
2/24: Creating functions, debugging
3/02: Permutation analysis
3/09: Advanced plotting

Review

```
> cabbage.mu <- tapply(cabbages$VitC,cabbages$Cult,
+                      mean,na.rm=TRUE)
> cabbage.mu
 c39  c52
51.5 64.4
```

```
> plot(weight~length,data=fish,xlim=c(0,50),
+       xlab="Mean Length (cm)",
+       ylab="Mean Weight (g)",
+       ylim=c(0,800),xaxs="i",yaxs="i",col=1:6,
+       pch=16,cex.lab=0.7,cex.axis=0.7,cex.main=0.7,
+       tcl=-0.25,mgp=c(1.5,0.25,0),
+       yaxp=c(0,750,3),
+       main="Laengelmavesi fish mean lengths and weights")
> legend(x="topleft",   #position in plot, also x=0, y=600
+        legend=fish$species,  #vector of text strings
+        col=1:6, # color of points
+        pt.cex=0.7,
+        pch=16,   # vector of symbol type
+        bty="n",cex=0.5)     # no box around legend
```

Laengelmavesi fish mean lengths and weights



Recommended reading

An introduction to R (Venables et al.)
– `http://cran.r-project.org/doc/manuals/R-intro.pdf` -
Today's material: Chapters 8, 11.

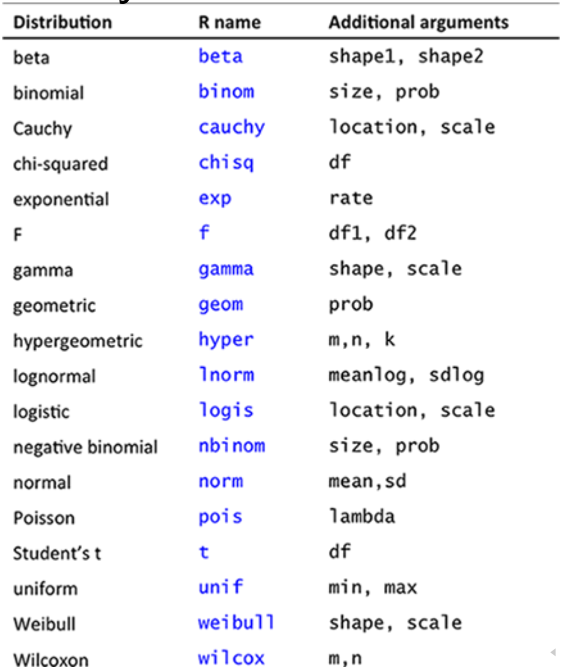
Probability distributions in `R`

`R` includes a set of probability distributions that can be used to simulate and model data.

If the function for the probability model is named `xxx`

- `pxxx`: the cumulative distribution $P(X \leq x)$
- `dxxx`: the probability distribution/density function $f(x)$
- `qxxx`: the quantile q, the smallest x such that $P(X \leq x) > q$
- `rxxx`: generate a random variable from the model `xxx`

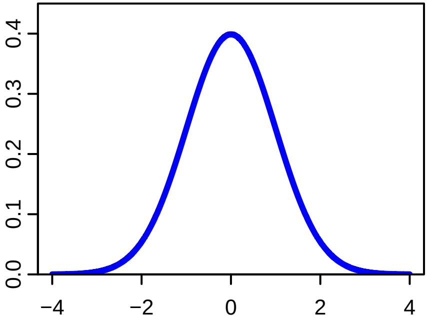
Probability distributions in R



Distribution	R name	Additional arguments
beta	beta	shape1, shape2
binomial	binom	size, prob
Cauchy	cauchy	location, scale
chi-squared	chisq	df
exponential	exp	rate
F	f	df1, df2
gamma	gamma	shape, scale
geometric	geom	prob
hypergeometric	hyper	m,n, k
lognormal	lnorm	meanlog, sdlog
logistic	logis	location, scale
negative binomial	nbinom	size, prob
normal	norm	mean,sd
Poisson	pois	lambda
Student's t	t	df
uniform	unif	min, max
Weibull	weibull	shape, scale
Wilcoxon	wilcox	m,n

Standard normal distribution

$(\mu = 0,\ \sigma^2 = 1)$



Functions for normal distribution

Values of x for different quantiles

`qnorm(p, mean=0, sd=1, lower.tail=TRUE, log.p=FALSE)`

```
> quants <- qnorm(c(0.01,0.025,0.05,0.95,0.975,0.99))
> round(quants,2)
[1] -2.33 -1.96 -1.64  1.64  1.96  2.33
```

$P(X \leq x)$

`pnorm(q, mean=0, sd=1, lower.tail=TRUE, log.p=FALSE)`

```
> pnorm(quants)
[1] 0.010 0.025 0.050 0.950 0.975 0.990
```

Functions for normal distribution

Density (probability ‘mass’ per unit value of x)

```
> dnorm(quants, mean = 0, sd = 1)
[1] 0.02665214 0.05844507 0.10313564 0.10313564 0.05844507
```

Generating standard normal random variables

```
> rnorm(n=10, mean = 0, sd = 1)
 [1]  1.5896384 -0.9677328  0.8573313  1.1280046 -1.0401407
 [7]  0.5077449 -1.0893859  1.0221718 -2.0941789
```

Generating random numbers

Computers generate pseudorandom numbers using a sequence of specially chosen numbers and algorithms.

Each sequence of numbers starts at a random seed with values in `.Random.seed`

By default the random sequence is initialized based on the start time of the program.

For repeatable pseudorandom sequences first call `set.seed(seed)` with `seed` = any integer between -2147483648 (-2^{31}) and 2147483647 ($2^{31} - 1$).

Generating random numbers

Often a good idea to use `set.seed()` and save the script detailing which number was used.

This ensures you can exactly repeat your results.

```
> set.seed(42)
> rnorm(3)
[1]  1.3709584 -0.5646982  0.3631284
> rnorm(3)
[1]  0.6328626  0.4042683 -0.1061245
> set.seed(42)
> rnorm(3)
[1]  1.3709584 -0.5646982  0.3631284
> rnorm(3)
[1]  0.6328626  0.4042683 -0.1061245
```

The sample() function

To generate random numbers from discrete sets of values:
- With or without replacement
- Equal or weighted probability

Extremely useful function that underlies many modern statistical techniques: - Resampling
- Bootstrapping
- Markov-chain Monte-Carlo (MCMC)

e.g. Roll 10 dice

```
> sample(1:6,size=10,replace=T)
 [1] 6 2 3 6 6 1 3 4 6 1
```

Pick 3 students from this class

```
> students <- c("Alex","Arjun","Ashley","Brooke","Chris",
+ "Liberty","Megan","Tammy")
> sample(students,size=3,replace=FALSE)
[1] "Tammy" "Megan" "Alex"
```

Lab Exercise 1/4

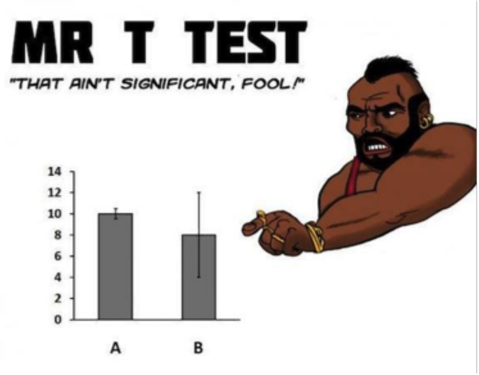
1. Generate 100 random normal numbers with mean 24 and standard deviation 10. Find the proportion that are ≥ 2 standard deviations from the mean.
2. Flip a (fair) coin six times.
3. Find the probability of getting six heads on those six flips (i.e. $P(X = 6)$ given $n = 6$).
4. How much more likely is it to get three heads than six?
5. For a standard normal random variable, find the number x such that $P(-x \leq X \leq x) = 0.24$.
6. The mean rate of arrival of alewives at a weir is 3.5 per hour. Plot the probability distribution function for the number of alewife arrivals.
7. Find the 95% confidence interval for the number of alewives arriving per day.

Basic statistical tests in R

LOTS of built-in functions to perform classical statistical tests.

e.g.,

- Correlation `cor.test()`
- Chi-squared `chisq.test()`
- t-test `t.test()`
- F-test `var.test()`
- Analysis of Variance `anova()`



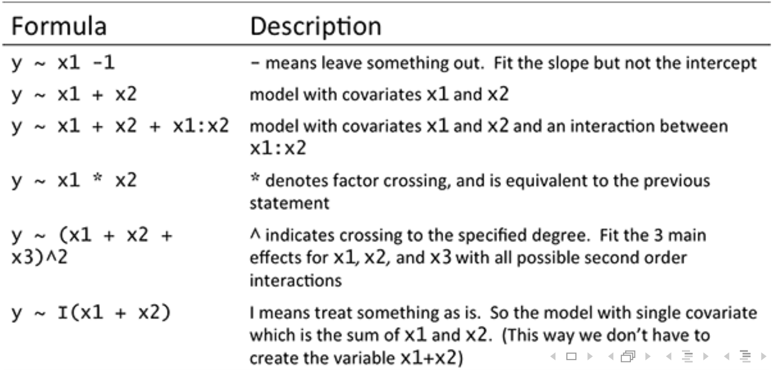
Linear models in R

Recall linear regression model:

$$y_i = \sum_{j=0}^{p} \beta_j x_i j + \varepsilon_i \ , \ \varepsilon_i \sim N(0,\sigma^2) \ , \ i = 1,\ldots,n$$

In matrix form: $y = X\beta + \varepsilon$

Model formulae in R, $y \sim x$, try `?formula`



Formula	Description
`y ~ x1 -1`	– means leave something out. Fit the slope but not the intercept
`y ~ x1 + x2`	model with covariates x1 and x2
`y ~ x1 + x2 + x1:x2`	model with covariates x1 and x2 and an interaction between x1:x2
`y ~ x1 * x2`	* denotes factor crossing, and is equivalent to the previous statement
`y ~ (x1 + x2 + x3)^2`	^ indicates crossing to the specified degree. Fit the 3 main effects for x1, x2, and x3 with all possible second order interactions
`y ~ I(x1 + x2)`	I means treat something as is. So the model with single covariate which is the sum of x1 and x2. (This way we don't have to create the variable x1+x2)

Linear models, `lm()`

The basic function for fitting ordinary multiple models is `lm()`

`fitted.model <- lm(formula, data = data.frame)`

e.g. species richness on beaches (Zuur Chapters 5 & 27)

```
> RIKZ <- read.table(file = "RIKZ.txt",header = TRUE)
> RIKZ$Richness <- rowSums(RIKZ[,2:76] > 0)
> RIKZ.lm1 <- lm(Richness ~ NAP, data = RIKZ)
```

Extracting model information

The value of `lm()` is a fitted model object.
- a list of results of class `lm`.

Information about the fitted model can be extracted, displayed, plotted, using some generic functions, inlcuding:

- `anova(object1,object2)` Compares a submodel with an outer model and produces an analysis of variance table
- `coef(object)` Extract the regression coefficient
- `deviance(object)` Residual sum of squares
- `formula(object)` Extract the model formula
- `plot(object)` Produce four plots, showing residuals, fitted values and some diagnostics
- `predict(object, newdata=data.frame)` Model predictions on new data
- `residuals(object)` or `resid(object)` Extract the matrix of residuals
- `step(object)` forward or backward model selection using AIC
- `summary(object)` Print a comprehensive summary of the results
- `vcov(object)` Return variance-covariance matrix of main parameters

Summary objects

```
summary(lm1)



Call:
lm(formula = Richness ~ NAP, data = RIKZ)

Residuals:
    Min      1Q  Median      3Q     Max
-5.0675 -2.7607 -0.8029  1.3534 13.8723

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.6857     0.6578  10.164 5.25e-13 ***
NAP          -2.8669     0.6307  -4.545 4.42e-05 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

Residual standard error: 4.16 on 43 degrees of freedom
Multiple R-squared:  0.3245,    Adjusted R-squared:  0.3088
F-statistic: 20.66 on 1 and 43 DF,  p-value: 4.418e-05
```

Are model assumptions met?

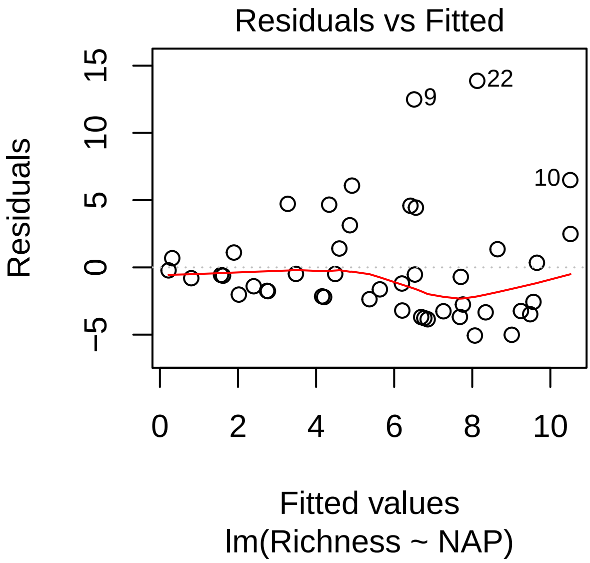
In the MASS library there are many functions.
Are samples independent? (Sample design.)
Normally distributed?
- Histograms, qq-plots: qqplot() and qqline()
- Kolmogorov-Smirnov normality test: ks.test()
- Shapiro-Wilk normality test: shapiro.test()
Similar variance among samples?
- Boxplots
- Bartlett's test for equal variance: bartlett.test()
- Fligner-Killeen test for equal variance: fligner.test()

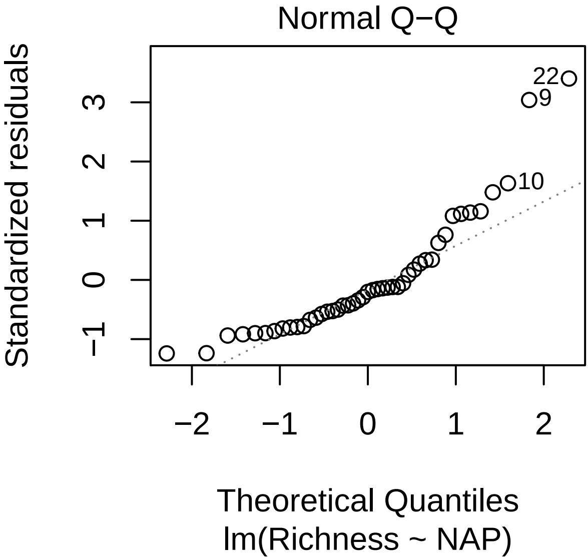
Checking assumptions

Model assumptions can be evaluated by plotting the model object.

```
> plot(RIKZ.lm1)
```

Click through the four plots, or use the argument `which` to specify which plot to draw.


Residuals vs Fitted
Residuals
−5
0
5
10
15
0
2
4
6
8
10
Fitted values
lm(Richness ~ NAP)
22
9
10


Normal Q−Q
Standardized residuals
−1
0
1
2
3
22
9
10
−2
−1
0
1
2
Theoretical Quantiles
lm(Richness ~ NAP)

Building and Comparing models

An alternative model for the RIKZ species richness is:

```
> RIKZ.lm2 <- lm(Richness ~ NAP+factor(week), data = RIKZ)
> #Can also create this model using the `update()` function:
> RIKZ.lm2 <- update(RIKZ.lm1, .~. + factor(week))
```

Compare models with `AIC()` and via `anova()`

```
> RIKZ.lm1 <- lm(Richness ~ NAP, data = RIKZ)
> RIKZ.lm2<-lm(Richness ~ NAP+factor(week), data = RIKZ)
> anova(RIKZ.lm1,RIKZ.lm2)
Analysis of Variance Table

Model 1: Richness ~ NAP
Model 2: Richness ~ NAP + factor(week)
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     43 744.12
2     40 357.00  3    387.11 14.458 1.581e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Lab exercise 2/4

1. Extract the residuals from the RIKZ.lm1 model
2. Check for normality using the residuals (you will need the MASS package loaded for tests)
3. Check whether the variances are equal
4. Are the assumptions met?
5. Compute the AIC for the two RIKZ models using the `AIC()` function
6. Use the logLik() function to extract both the log-likelihood and number of parameters and then compute the AIC for the two RIKZ models from the equation $AIC = -2 * \ln(likelihood) + 2p$
7. Compute AICc for both models $\left(AICc = -2 * \ln(likelihood) + 2p * \frac{n}{n-p-1}\right)$

Generalized linear modeling

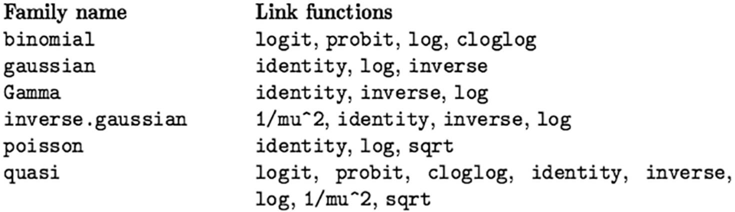
Recall from lecture:

$$\eta = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

$$f_Y(y; \mu, \varphi) = \exp\left[\frac{A}{\varphi} y \lambda(\mu) - \gamma(\lambda(\mu)) + \tau(y, \varphi)\right]$$

$$\mu = m(\eta),\ \eta = m^{-1}(\mu) = l(\mu)$$

The combination of a response distribution, a link function and other information needed to carry out the modeling exercise is called the *family* of the generalized linear model.



Family name	Link functions
`binomial`	`logit`, `probit`, `log`, `cloglog`
`gaussian`	`identity`, `log`, `inverse`
`Gamma`	`identity`, `inverse`, `log`
`inverse.gaussian`	`1/mu^2`, `identity`, `inverse`, `log`
`poisson`	`identity`, `log`, `sqrt`
`quasi`	`logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2`, `sqrt`

The `glm()` function

The R function to fit a generalized linear model is `glm()` which uses the form:
`fitted.model <- glm(formula,`
`family=family.generator, data=data.frame)`

Only new piece is the call to ‘family.generator’
Although complex, its use is fairly simple.
Where there is a choice of link, link supplied with the family name as a parameter.

Simple (inefficient) use: The following are equivalent.

```
> RIKZ.lm1 <- lm(Richness ~ NAP, data = RIKZ)
> RIKZ.glm1 <- glm(Richness ~ NAP, family = gaussian,
+                  data = RIKZ)
```

Most of the extraction functions that can be applied to `lm()` can also be used with `glm()`.

Poisson regression

$$P(X = x) = \frac{e^{-\mu}\mu^{x}}{x!},\ \mu_i = e^{\alpha+\beta_1 x_1,i+\cdots+\beta_j,i}$$

Les poissons. Les Poissons.
How I LOOOVE les poissons!
Useful-Disney-Quotes.Tumblr.com

RIKZ example

```
> RIKZ_poisson <- glm(Richness ~ NAP, data = RIKZ,
+                     family = poisson)
```

Note that the default link for the poisson is `log` so we don't have to specify here (see `?family`).

summary(RIKZ_poisson)

```
Call:
glm(formula = Richness ~ NAP, family = poisson, data = RIKZ)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2029  -1.2432  -0.9199   0.3943   4.3256

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.79100    0.06329  28.297  < 2e-16 ***
NAP         -0.55597    0.07163  -7.762 8.39e-15 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 179.75  on 44  degrees of freedom
Residual deviance: 113.18  on 43  degrees of freedom
AIC: 259.18

Number of Fisher Scoring iterations: 5
```

Lab exercise 3/4

1. Fit a poisson model to RIKZ species richness that includes NAP, and exposure and week as nominal variables (`factor`).
2. Plot the deviance residuals versus the linear predictor.
3. Compare the new model with the original RIKZ_poisson model using deviance and AIC.

Logistic regression

$$Y_i \sim B(1, P_i) \text{ and } E[Y_i] = P_i = \mu_i = frace^{g(x_i)}1 + e^{g(x_i)}$$

The logit transform in R

```
> library(boot)
> (p <- seq(0.1,0.9,0.1))
[1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
> logit(p)
[1] -2.1972246 -1.3862944 -0.8472979 -0.4054651  0.0000000
[7]  0.8472979  1.3862944  2.1972246
> logit.p <- log(p/(1-p))
> logit.p
[1] -2.1972246 -1.3862944 -0.8472979 -0.4054651  0.0000000
[7]  0.8472979  1.3862944  2.1972246
> exp(logit.p)/(1+exp(logit.p))
[1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
> inv.logit(logit.p)
[1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
```

Binomial GLM in R

Use `family = binomial`.

```
> Solea <- read.table("Solea.txt", header = T)
> solea_glm.1 <- glm(Solea_solea ~ salinity,
+                     data = Solea, family = binomial)
> solea_glm.1

Call:  glm(formula = Solea_solea ~ salinity, family = binomial,

Coefficients:
(Intercept)     salinity
     2.6607      -0.1299

Degrees of Freedom: 64 Total (i.e. Null);  63 Residual
Null Deviance:      87.49
Residual Deviance: 68.56    AIC: 72.56
```

Extract the odds

```
> odds <- exp(coef(solea_glm.1)[-1])
> odds
 salinity
0.8782228
```

predict.glm()

As with `lm()`, the `predict()` function can be used to obtain predictions from a fitted model object to a new data frame.

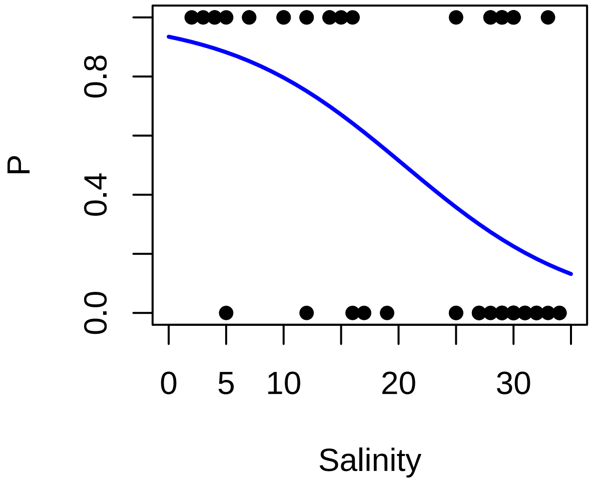
This can be useful when:
- a subset of the data was reserved from the fitting process ('test data')
- want to obtain model predictions at values other than the data (for example when plotting fitted values)

```
> newobject <- predict(oldobject, newdata,
+                      type = c("linK","response","terms"),
+                      se.fit = FALSE)
```

Note: default for `type` is on the scale of the linear predictors. Set to `"response"` to obtain predictions on the scale of the response variable.

Observed and fitted values for Solea

```
> solea_glm.1 <- glm(Solea_solea ~ salinity,data = Solea,
+                    family = binomial)
> plot(Solea$salinity, Solea$Solea_solea,xlab = "Salinity",
+      ylab = "P",xlim=c(0,35),ylim=c(0,1),pch=16)
> #create new values for salinity
> new.salinity <- seq(0,35,1)
> #predict values for response based on new salinity
> new.pred <- predict(solea_glm.1,data.frame(salinity = new
+                                 type = "response")
> lines(new.salinity,new.pred,col="blue",lwd=2)
```


P
0.0
0.4
0.8
0
5
10
20
30
Salinity

More on interrogating model results

You can also interrogate model objects directly. Type `names(object)` to get a list of the components of `object`.

```
> names(solea_glm.1)
 [1] "coefficients"      "residuals"          "fitted.values
 [4] "effects"           "R"                  "rank"
 [7] "qr"                "family"             "linear.predic
[10] "deviance"          "aic"                "null.deviance
[13] "iter"              "weights"            "prior.weights
[16] "df.residual"       "df.null"            "y"
[19] "converged"         "boundary"           "model"
[22] "call"              "formula"            "terms"
[25] "data"              "offset"             "control"
[28] "method"            "contrasts"          "xlevels"
```

`str()` can also be used.

More on interrogating model results

Note that summary(object) is also a list, and components can be extracted from here too.

```
> names(summary(solea_glm.1))
 [1] "call"           "terms"          "family"         "de
 [5] "aic"            "contrasts"      "df.residual"    "nu
 [9] "df.null"        "iter"           "deviance.resid" "co
[13] "aliased"        "dispersion"     "df"             "co
[17] "cov.scaled"
> str(summary(solea_glm.1))
List of 17
 $ call          : language glm(formula = Solea_solea ~ sal
 $ terms         :Classes ’terms’, ’formula’ length 3 Solea
  .. ..- attr(*, "variables")= language list(Solea_solea, s
  .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:2] "Solea_solea" "salinity"
  .. .. .. ..$ : chr "salinity"
  .. ..- attr(*, "term.labels")= chr "salinity"
```

Lab Exercise 4/4

1. Calculate the Bayesian Information Criterion (BIC) for the `solea_glm.1` salinity term ($BIC = z^2 - \ln(n)$).
2. Plot the fitted probability of presence of *Solea* as a function of depth. (*hint* fit a new model, then use `predict()` with a new data frame of new depth values)
3. **bonus** Fit a model with both salinity and depth and show the fitted response (*hint* use `expand.grid` to create the new data frame with combinations of both linear predictors)
4. Fit a model of *Solea solea* presence/absence that includes temperature, salinity, gravel, and month as linear predictors. (note that month is a nominal variable)
5. Check the fit using validation plots.

6. Print the ANOVA table for the new model.
7. Extract the model coefficients and their 95% confidence intervals.
8. Display the odds for the model. Comment on the magnitide of effect and relative influence of the different variables given their range within the data.
9. Show how AIC changes when each linear predictor term is dropped from the model, and show the results of chi-square tests that compare these reduced models to the original model. (*hint* see the help for `?add1`)
10. **bonus** Plot the partial fits of the individual explanatory variables while taking into account the other model variables (*hint* see the help on `termplot()`)