# JAGS Examples from Kery's book Introduction to WinBugs

Code in Kery's book can be downloaded from here:

http://www.mbr-pwrc.usgs.gov/software/kerybook/R/_WB/_code.txt

# JAGS Examples from Kery's book Introduction to WinBugs

Code in Kery's book can be downloaded from here:

http://www.mbr-pwrc.usgs.gov/software/kerybook/R/_WB/_code.txt

Most of the code will run, with a few modifications, using JAGS. In particular, we have to add `BUGSoutput` when referencing any of the output from JAGS.

- In Kery's booK: `plot(out$mean$predicted, out$mean$residual)`
- We need to use:
  `plot(out$BUGSoutput$mean$predicted, out$BUGSoutput$mean$residual)`.

# Trends in Wallcreeper Counts



- Proportion of sample quadrats where the bird is observed
- Assume deviations from a linear trend are normally distributed

# Ch 8: Linear regression

Prior and likelihood is very similar to what we have already seen.

```
linreg<-function(){

# Priors
 alpha ~ dnorm(0,0.001)
 beta ~ dnorm(0,0.001)
 sigma ~ dunif(0, 100)

# Likelihood
 for (i in 1:n) {
    y[i] ~ dnorm(mu[i], tau)
    mu[i] <- alpha + beta*x[i]
 }
```

# Derived quantities

```
# Derived quantities
tau <- 1/ (sigma * sigma)
p.decline <- 1-step(beta)        # Probability of decline
```

Step() = 1 if what is in the "()" is > 0 (and 0 otherwise). So, this calculates the probability that $\beta < 0$.

# Residuals

```
for (i in 1:n) {
    residual[i] <- y[i]-mu[i]       # Residuals for observed data
    predicted[i] <- mu[i]        # Predicted values
    sq[i] <- pow(residual[i], 2)    # Squared residuals for observed d
```

Note: we will get `n.iter` residuals for each data point! Why?

# Residuals

```
for (i in 1:n) {
    residual[i] <- y[i]-mu[i]        # Residuals for observed data
    predicted[i] <- mu[i]         # Predicted values
    sq[i] <- pow(residual[i], 2)    # Squared residuals for observed d
```

Note: we will get n.iter residuals for each data point! Why?

Residual: r[i] = $y_i - (alpha + beta * x[i])$.

# Residuals

```
for (i in 1:n) {
    residual[i] <- y[i]-mu[i]        # Residuals for observed data
    predicted[i] <- mu[i]        # Predicted values
    sq[i] <- pow(residual[i], 2)     # Squared residuals for observed d
```

Note: we will get n.iter residuals for each data point! Why?

Residual: r[i] = $y_i - (alpha + beta * x[i])$.

We generate $n_{MCMC}$ values of alpha and beta, so $n_{MCMC}$ values of r[i]!

# Bayesian p-value

- Generate new ("ideal") data: `y.new[i]` $\sim$ `dnorm(mu[i], tau)`
  - Using the assumed model
  - Using parameters drawn from the posterior distribution (accounts for parameter uncertainty)

# Bayesian p-value

- Generate new ("ideal") data: `y.new[i] $\sim$ dnorm(mu[i], tau)`
  - Using the assumed model
  - Using parameters drawn from the posterior distribution (accounts for parameter uncertainty)
- Calculate measure of "fit" (or lack of fit) for for the observed data and simulated data
  - `fit <- sum(sq[])` sum of squared residuals for the observed data
  - `fit.new <- sum(sq.new[])` (same for simulated data)

# Bayesian p-value

If the model fits the data well: "goodness-of-fit" statistics for real and simulated data should be similar

# Bayesian p-value

If the model fits the data well: "goodness-of-fit" statistics for real and simulated data should be similar

If the model does not fit the data well

- sum of squared residuals for the observed data > sum of squared residuals for the simulated data

# Bayesian p-value

If the model fits the data well: "goodness-of-fit" statistics for real and simulated data should be similar

If the model does not fit the data well

- sum of squared residuals for the observed data > sum of squared residuals for the simulated data

```
test <- step(fit.new - fit)
```

# Bayesian p-value

If the model fits the data well: "goodness-of-fit" statistics for real and simulated data should be similar

If the model does not fit the data well

- sum of squared residuals for the observed data > sum of squared residuals for the simulated data

```
test <- step(fit.new - fit)
```

- test = 1 if SS(simulated data) > SS(observed data), i.e., the model fits the observed data better than the simulated data.

# Bayesian p-value

If the model fits the data well: "goodness-of-fit" statistics for real and simulated data should be similar

If the model does not fit the data well

- sum of squared residuals for the observed data > sum of squared residuals for the simulated data

```
test <- step(fit.new - fit)
```

- test = 1 if SS(simulated data) > SS(observed data), i.e., the model fits the observed data better than the simulated data.
- bpvalue <- mean(test) (how often does the model fit observed data better than the simulated data)
- Want bpvalue to be large (can't conclude there is a lack of fit).

# Kery's code: Predictions = CI for the line = $E[Y|X]$

We have $n_{MCMC}$ = `n.chains(n.iter - n.burn)` draws from the posterior distribution of $(\alpha, \beta, \sigma)$.

# Kery's code: Predictions = CI for the line = $E[Y|X]$

We have $n_{MCMC}$ = n.chains(n.iter - n.burn) draws from the posterior distribution of $(\alpha, \beta, \sigma)$.

For each value of $X = x$ (loop), he generates $n_{MCMC}$ predictions.

```
for(i in 1:length(x)){
   predictions[i,] <- out$BUGSoutput$sims.list$alpha +
     out$BUGSoutput$sims.list$beta*x[i]
}
```

# Kery's code: Predictions = CI for the line = $E[Y|X]$

We have $n_{MCMC}$ = n.chains(n.iter - n.burn) draws from the posterior distribution of $(\alpha, \beta, \sigma)$.

For each value of $X = x$ (loop), he generates $n_{MCMC}$ predictions.

```
for(i in 1:length(x)){
   predictions[i,] <- out$BUGSoutput$sims.list$alpha +
     out$BUGSoutput$sims.list$beta*x[i]
}
```

- Gives us $n_{MCMC}$ draws of $E[Y|X = x] = \alpha + x\beta$. What does this tell us?

# Kery's code: Predictions = CI for the line = $E[Y|X]$

We have $n_{MCMC}$ = n.chains(n.iter − n.burn) draws from the posterior distribution of $(\alpha, \beta, \sigma)$.

For each value of $X = x$ (loop), he generates $n_{MCMC}$ predictions.

```
for(i in 1:length(x)){
   predictions[i,] <- out$BUGSoutput$sims.list$alpha +
     out$BUGSoutput$sims.list$beta*x[i]
}
```

- Gives us $n_{MCMC}$ draws of $E[Y|X = x] = \alpha + x\beta$. What does this tell us?
- This is the posterior distribution of the fitted line (tells us about the line and its uncertainty)

# Prediction Intervals

What about prediction intervals? Intervals that are likely to contain a new observation, if collected?

# Prediction Intervals

What about prediction intervals? Intervals that are likely to contain a new observation, if collected?
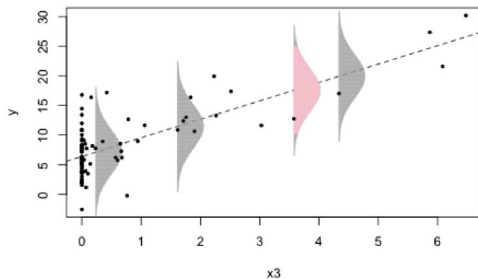


- Requires we also consider the variability about the line

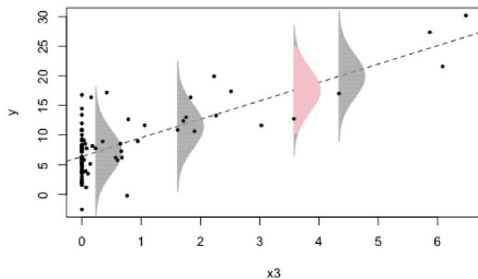# Prediction Intervals

What about prediction intervals? Intervals that are likely to contain a new observation, if collected?



- Requires we also consider the variability about the line
- This variability is described by the errors $\sim N(0, \sigma^2)$.

# Exercise 1: Bayesian prediction intervals

Start with the posterior for the line:

- We have $n_{MCMC}$ values of $E[Y|X=x] = \alpha + x\beta$

# Exercise 1: Bayesian prediction intervals

Start with the posterior for the line:

- We have $n_{MCMC}$ values of $E[Y|X = x] = \alpha + x\beta$

We need to add. . .

- random normal deviates

# Exercise 1: Bayesian prediction intervals

Start with the posterior for the line:

- We have $n_{MCMC}$ values of $E[Y|X = x] = \alpha + x\beta$

We need to add...

- random normal deviates
- add: rnorm($n_{MCMC}$, mean = 0, sd = reference posterior for $\sigma$)

# Exercise 1: Bayesian prediction intervals

Start with the posterior for the line:

- We have $n_{MCMC}$ values of $E[Y|X = x] = \alpha + x\beta$

We need to add...

- random normal deviates
- add: rnorm($n_{MCMC}$, mean = 0, sd = reference posterior for $\sigma$)
- gives us the posterior of $Y_{new}|X$ (rather than $E[Y|X]$)

# Exercise 1: Bayesian prediction intervals

Start with the posterior for the line:

- We have $n_{MCMC}$ values of $E[Y|X = x] = \alpha + x\beta$

We need to add...

- random normal deviates
- add: rnorm($n_{MCMC}$, mean = 0, sd = reference posterior for $\sigma$)
- gives us the posterior of $Y_{new}|X$ (rather than $E[Y|X]$)
- Try to modify Kery's code to do this! Hint: to figure out $n_{MCMC}$, inspect the dimension of `out$BUGSoutput$sims.list$alpha`

# Bayesian prediction intervals

Note: we could have added the normal random deviates within our `linreg` function.

# Bayesian prediction intervals

Note: we could have added the normal random deviates within our `linreg` function.

Actually, we did do this when we generated y.new.

# Bayesian prediction intervals

Note: we could have added the normal random deviates within our `linreg` function.

Actually, we did do this when we generated y.new.

We could have just added `y.new` to our "parameters.to.save" vector and then summarized the quantiles of the jags output to get the prediction intervals.

# Exercise 2: Prediction Intervals (frequentist)

For linear models fit in R, we can generate confidence and prediction intervals using the `predict` function, and more specifically, `predict.lm`

- Look up the help file for `predict.lm` and examples at bottom
- Calculate these intervals for the fitted linear model.
- Plot these intervals using the `lines` function along with Bayesian prediction intervals

Chapter 11: ANCOVA model (importance of priors and scaling)

# Relationship between Body Mass and Body Length



- Asp viper's from 3 populations (Pyrenees, Massif Central, Jura Mountains) in Switzerland
- Interested in population-specific differences in the body-mass relationship

# Linear regression model

$$y_i = \alpha_0 + \beta_1 x_i + \epsilon_i$$

- $y_i$ = body mass of individual $i$
- $x_i$ = body length of snake $i$

How dow we capture "population-specific differences in the body-mass relationship"?

# Linear regression model

$$y_i = \alpha_0 + \beta_1 x_i + \epsilon_i$$

- $y_i$ = body mass of individual $i$
- $x_i$ = body length of snake $i$

How dow we capture "population-specific differences in the body-mass relationship"?

Include interactions!

# Example CH 11.2

Effects parameterization:

$$y_i = \alpha_{pyr} + \beta_1 I_{MC,i} + \beta_2 I_{Jura,i} + \beta_3 x_i + \beta_4 x_i I_{MC,i} + \beta_5 x_i I_{Jur,i} + \epsilon_i$$

# Example CH 11.2

Effects parameterization:

$$y_i = \alpha_{pyr} + \beta_1 I_{MC,i} + \beta_2 I_{Jura,i} + \beta_3 x_i + \beta_4 x_i I_{MC,i} + \beta_5 x_i I_{Jur,i} + \epsilon_i$$

Means parameterization:

$$y_i = \alpha_{j(i)} + \beta_{j(i)} x_i + \epsilon_i$$

# Example CH 11.2

Effects parameterization:

$$y_i = \alpha_{pyr} + \beta_1 I_{MC,i} + \beta_2 I_{Jura,i} + \beta_3 x_i + \beta_4 x_i I_{MC,i} + \beta_5 x_i I_{Jur,i} + \epsilon_i$$

Means parameterization:

$$y_i = \alpha_{j(i)} + \beta_{j(i)} x_i + \epsilon_i$$

Identify the parameters used in the simulation:

- Effects model ($\alpha$ and $\beta_1 - \beta_5$)
- Means model ($\alpha$ and $\beta$) for each population.

# Parameters

Effects parameterization

- $\alpha$ = -250
- $\beta_1$ = 150
- $\beta_2$ = 200
- $\beta_3$ = 6
- $\beta_4$ -3
- $\beta_5$ -4

Means parameterization

- Pyrenees: $\alpha$ = -250, $\beta$ = 6
- Massif: $\alpha$ = -100, $\beta$ = 3
- Jura: $\alpha$ = -50, $\beta$ = 2

# What happenened?

Kery shows that standardizing works. Centering would also help. Why?

# What happenened?

Kery shows that standardizing works. Centering would also help. Why?

Lets look at the prior distributions for $\alpha \sim N(0, 0.001)$. Plot this distribution.

# What happenened?

Kery shows that standardizing works. Centering would also help. Why?

Lets look at the prior distributions for $\alpha \sim N(0, 0.001)$. Plot this distribution.

```
curve(dnorm(x, 0, sqrt(1/0.001)), from=-250, to=250)
```