Stochastic Simulation and Power Analysis

Michael Noonan

Biol 520C: Statistical modelling for biological data

Table of contents

THE UNIVERSITY OF BRITISH COLUMBIA
Okanagan Campus

Stochastic Simulation and the Monte Carlo Method

Simulation

Everything we've been doing so far has been focused on parameter estimation. This can been seen as *'inverse'* modelling (i.e., we have the data and we try to identify the process that generated it).

Simulation can be seen as *'forward'* modelling. If we pick/build a model, what patterns can we expect to see in data.

Biologists often use simulation in order to explore: i) patterns that would emerge from a given model(s); or ii) plan future studies.

If we chain together simulations from multiple models we can generate rich and complex descriptions of biological systems.

Simulation cont.

Simulations can be useful for exploring the models and distributions you've used to model a dataset (if you can $\sim$recreate your data via simulation you know your model is probably reasonable).

Simulations can also be used to test parameter estimation procedures and how models perform when assumptions are broken in known ways (what we've been using them for throughout this course).

With real data we will never know the 'true' model, so we can never know how close we're getting. If we want to know how our estimations will perform under different scenarios we need to rely on best-case situations where the true model and distributions are known.

In the past any of these types of projects would require sophisticated analyses but simulations make these accessible to a broader range of biologists (learning sim. methods is easier than learning high level math).

Stochastic simulation

Stochastic simulation is a special type of simulation that relies on computational algorithms to randomly sample values from probability distributions to emulate a system's stochasticity (e.g. `rnorm()`, `runif()`, etc.).

Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results.

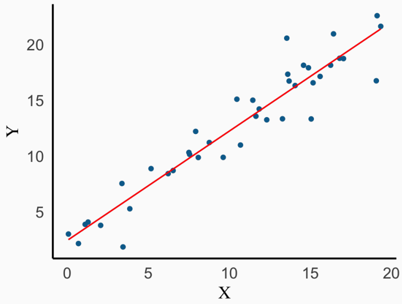
$$\text{Monte Carlo} = \text{Stochastic Simulation}$$

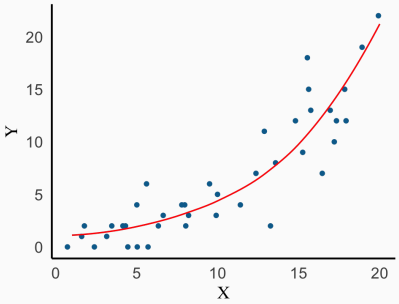
Stochastic sims: Single groups

If we know something about our system, we can generate simulated data using the methods we've seen throughout the course.

```
Linear <- function(x) {
  B_0 <- 2
  B_1 <- 1
  mu = B_0 + B_1*x
  rnorm(n = length(x), mean = mu, sd = 2)}

X <- runif(40, 0, 20)
Y <- Linear(X)
```

```
Counts <- function(x) {
  B_0 <- 0.01
  B_1 <- 0.15
  eta = exp(B_0 + B_1*x)
  rpois(n = length(x), lambda = eta)}

X <- runif(40, 0, 20)
Y <- Counts(X)
```





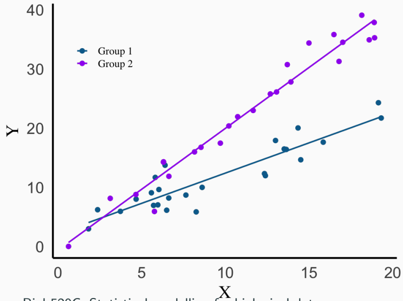
Stochastic sims: Multiple groups

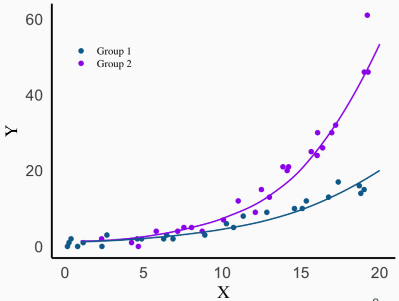
We're often interested in differences between groups (species, treatments, study sites, etc.).

```
group <- factor(rep(1:2, each = 25))
```

```
Linear <- function(x) {
  B_0 <- c(2,0)
  B_1 <- c(1,2)
  mu = B_0[group] + B_1[group]*x
  rnorm(n = length(x), mean = mu, sd = 2)}
X <- runif(50, 0, 20)
Y <- Linear(X)
```

```
Counts <- function(x) {
  B_0 <- c(0.01, 0)
  B_1 <- c(0.15, 0.2)
  eta = exp(B_0[group] + B_1[group]*x)
  rpois(n = length(x), lambda = eta)}
X <- runif(50, 0, 20)
Y <- Counts(X)
```





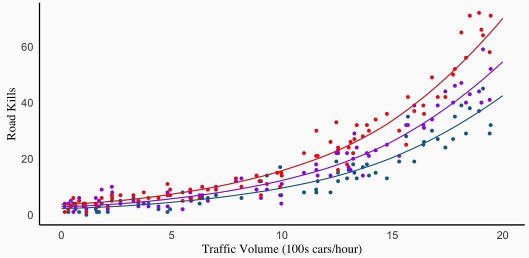
Stochastic sims: Multiple parameters

Our models often have multiple parameters and simulations can allow us to explore individual effects holding all else constant.

Imagine a system where traffic volume and temperature influence the number of road killed animals. We can use simulations to explore how climate change might influence road kills.

```r
Road_Kills <- function(x, x_2) {
  B_0 <- 0.01
  B_1 <- 0.15
  B_2 <- 0.05
  eta = exp(B_0 + B_1*x + B_2*x_2)
  rpois(n = length(x), lambda = eta)}
```

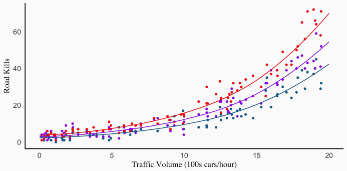
```r
X <- runif(80, 0, 20)

Deaths_15C <- Road_Kills(X, 15)
Deaths_20C <- Road_Kills(X, 20)
Deaths_25C <- Road_Kills(X, 25)
```

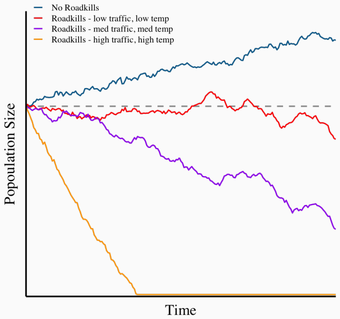


Stochastic sims: Chaining simulations

If we chain together simulations from multiple models we can generate rich descriptions of complex systems.

```
POP <- as.vector(200)
for(i in 1:200){
  Births <- rpois(1, 40)
  Deaths <- rpois(1, 38)
  POP[i+1] <- POP[i] + Births - Deaths}

POP2 <- as.vector(200)
for(i in 1:200){
  Births <- rpois(1, 40)
  Deaths <- rpois(1, 38)
  RK_Deaths <- Road_Kills(2, 15)
  POP2[i+1]<-POP2[i]+Births-Deaths-RK_Deaths}

...
  RK_Deaths <- Road_Kills(5, 20)
...

...
  RK_Deaths <- Road_Kills(10, 25)
...
```





Simulation take homes

With simulation studies you can easily manipulate any ingredient in your models (model params, data, params of the distributions). This makes simulations a very **powerful** tool for exploring biological systems and making data informed predictions.

With simulation studies you can easily manipulate any ingredient in your models (model params, data, params of the distributions). This makes simulations a potentially **dangerous** tool for exploring biological systems and making data informed predictions.

Always approach simulation studies with care and make sure the computational system you put together matches the biological reality of the system you are trying to study.

Power Analysis

Power analysis

Power analysis, in a traditional sense, means identifying the minimum sample size needed to detect the presence of a real effect if one is present.

It is traditionally associated with design based inference and pairwise hypothesis testing (ANOVAs, t-tests, etc.).

In model based inference each parameter has it's own effect, so the concept translates, but the tools required are different. There's also more to a model than just it's power.

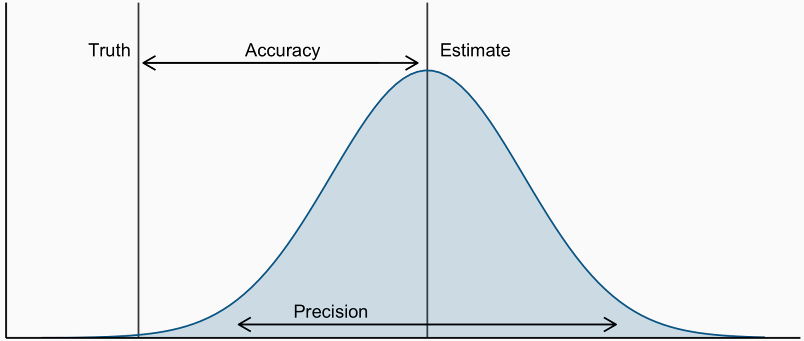
Power analysis in our context involves a special kind of simulation study aimed at exploring how much data you would need in order to get reasonably accurate estimates of your parameters, detect significance of parameters with true effects, and/or detect differences between groups.

Precision and accuracy

Ultimately, the quality of the inference we can make from our analyses boils down to two main factors, a model's **accuracy** and it's **precision**.

Accuracy specifies how likely your answer is to be correct

Precision describes variability in the estimates.



Precision and accuracy cont.

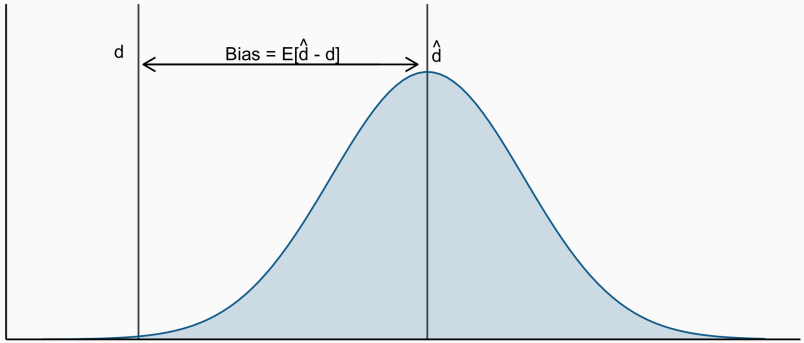
The precision and accuracy of an estimator can be estimated through a number of different measures:

1. Bias (accuracy)
2. Variance (precision)
3. Confidence interval width (precision)
4. Mean squared error (MSE: accuracy and precision)
5. Coverage (accuracy)
6. Power (precision)

Statistical bias (accuracy)

Bias is the expected difference between an estimate ($\hat{d}$) of a parameter and its true value (d).

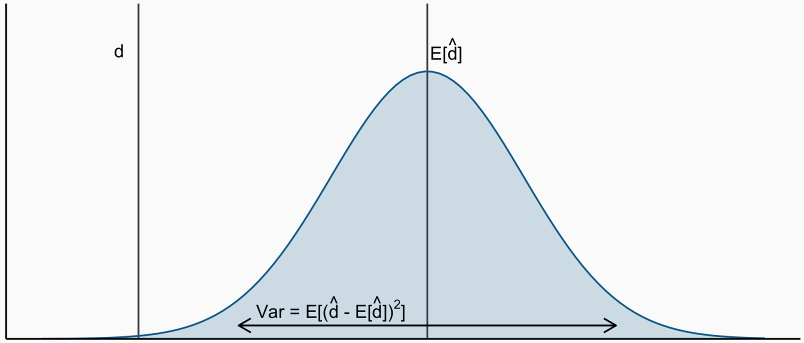
Ideally estimators should be asymptotically unbiased (as $n \to \infty \quad E[\hat{d} - d] \to 0$)



Variance and CI width (precision)

Variance measures the variability of individual estimates ($\hat{d}$) around the mean estimate ($E[\hat{d}]$).

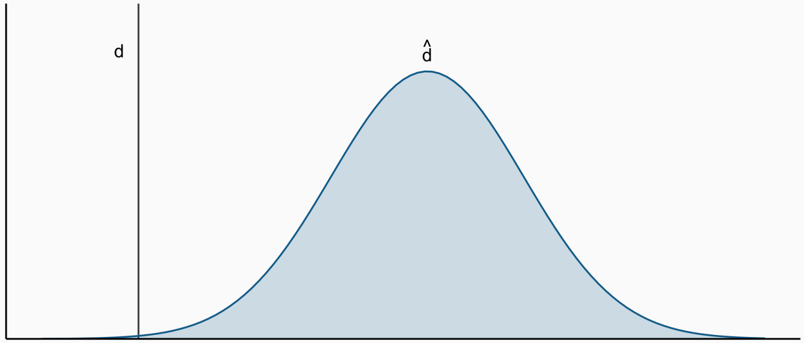
Low variance will give narrow CIs, large variance will give wide CIs



MSE (accuracy & precision)

Mean squared error (MSE) is a measure that combines both accuracy and precision and is calculated as $E[(\hat{d} - d)^2]$.

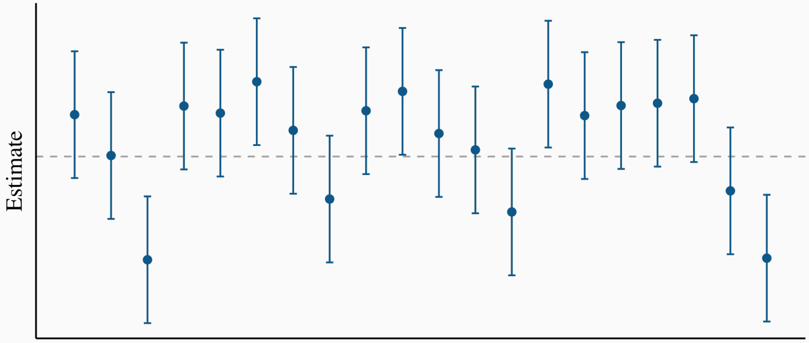
It provides a measure of total variation around the truth.



CI Coverage (accuracy)

CI coverage described the accuracy of a set of confidence intervals.

If 95% CIs are behaving like they should, they will include (cover) the truth 95% of the time.

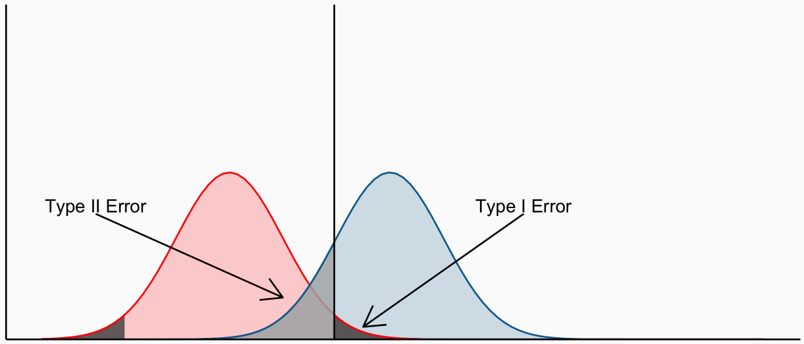


Statistical power (precision)

Type I (α) – False positive (i.e., observe an effect that isn't present).

Type II (β) – False Negative (don't detect an effect that is present).

Statistical power (1-β) – a measure of our ability to detect a real effect.

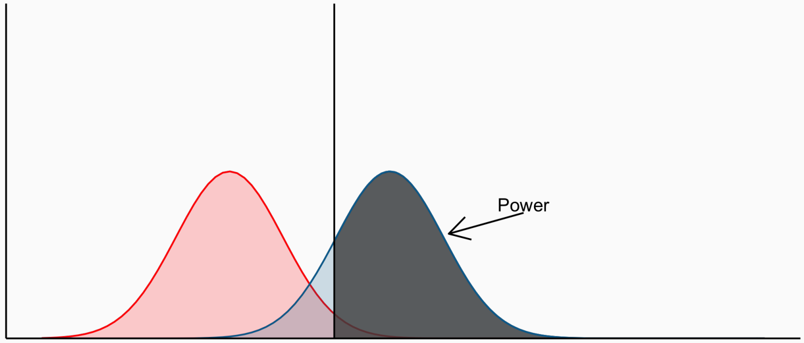


Statistical power (precision)

Type I (α) – False positive (i.e., observe an effect that isn't present).

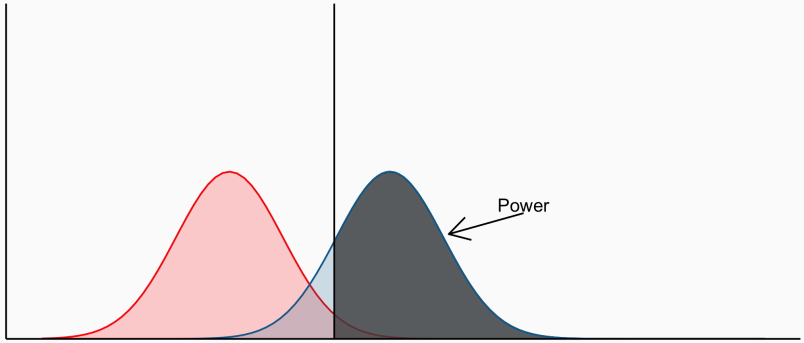
Type II (β) – False Negative (don't detect an effect that is present).

Statistical power (1-β) – a measure of our ability to detect a real effect.



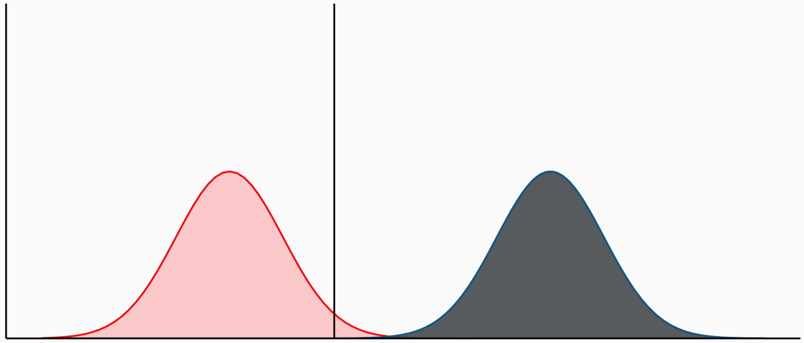
Statistical power (precision)

If we want to increase the power of a test there are two options:



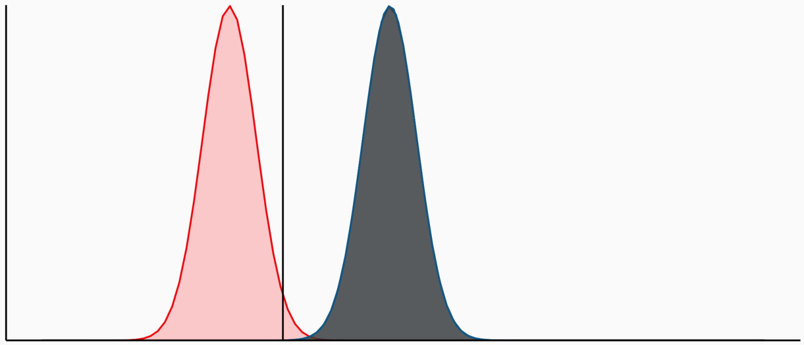
Statistical power (precision)

If we want to increase the power of a test there are two options:
1) increase effect size



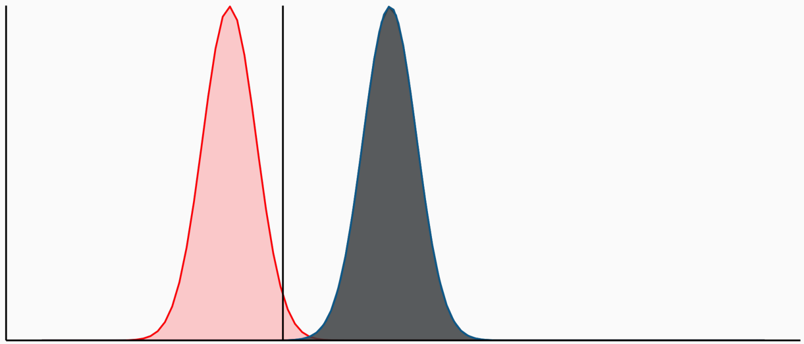
Statistical power (precision)

If we want to increase the power of a test there are two options:
1) increase effect size; or 2) make the distributions narrower ($\uparrow N$)



Statistical power (precision)

Some experimental designs can change effects sizes but we usually don't have control over this, so we typically $\uparrow$ sample sizes to $\uparrow$ power.



Power Analysis in Action

Power Analysis: Linear Regression

Let's estimate the statistical power of detecting a linear trend with a sample size of 20.

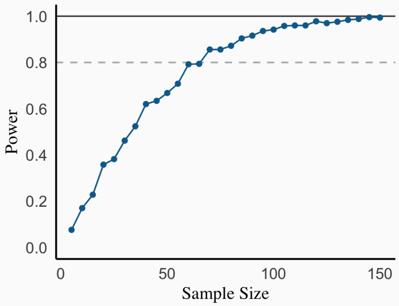
```
Linear <- function(x) {
  B_0 <- 2
  B_1 <- 0.5
  mu = B_0 + B_1*x
  rnorm(n = length(x), mean = mu, sd = 8)}

nSims <- 500

pval <- numeric(nSims)

for(i in 1:nSims){
X <- runif(20, 0, 20)
Y <- Linear(X)
fit <- lm(Y ~ X)
pval[i] <- coef(summary(fit))["X", "Pr(>|t|)"]}

sum(pval < 0.05)/nSims
0.326
```

Power Analysis: Linear Regression

Most of the time were interested in power for multiple sample sizes.

We can code this using nested for loops.

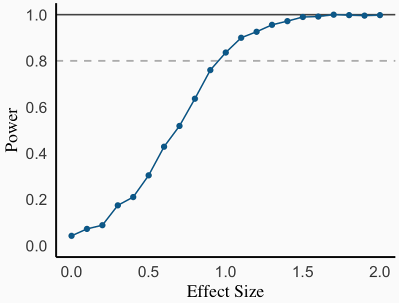
```
nSims <- 500
n <- seq(5,150,5)

POWER <- numeric(length(n))
for(j in 1:length(n)){

  pval <- numeric(nSims)
  for(i in 1:nSims){
  X <- runif(n[j], 0, 20)
  Y <- Linear(X)
  fit <- lm(Y ~ X)
  pval[i] <-coef(summary(fit))["X", "Pr(>|t|)"]
  }

  POWER[j] <- sum(pval < 0.05)/nSims
  }
```



Power Analysis: Linear Regression

Let's say we knew our maximum sample size was 20, we could also estimate the smallest effect size (slope) we could reasonably detect.

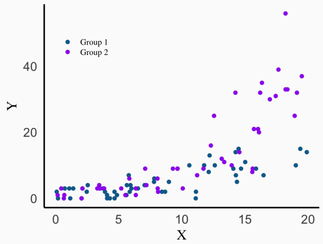
```
Linear <- function(x, B_1) {
  B_0 <- 2
  B_1 <- B_1
  mu = B_0 + B_1*x
  rnorm(n = length(x), mean = mu, sd = 8)}

nSims <- 500
B_1 <- seq(0,2,0.1)
POWER <- numeric(length(B_1))

for(j in 1:length(B_1)){
  pval <- numeric(nSims)
  for(i in 1:nSims){
    X <- runif(20, 0, 20)
    Y <- Linear(X, B_1[j])
    fit <- lm(Y ~ X)
    pval[i]<-coef(summary(fit))["X", "Pr(>|t|)"]
  }
  POWER[j] <- sum(pval < 0.05)/nSims
}
```

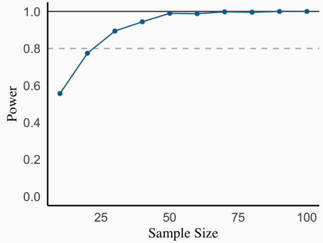


Detecting differences between groups

We might also be interested in estimating our ability to detect differences between groups.

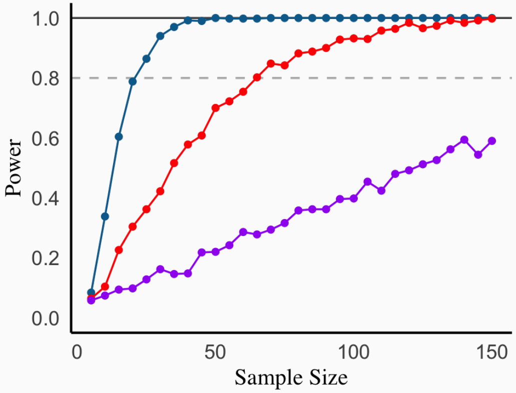
```
Counts <- function(x,group) {
  B_0 <- c(0.01, 0)
  B_1 <- c(0.15, 0.2)
  eta = exp(B_0[group] + B_1[group]*x)
  rnbinom(n = length(x), mu = eta, size = 10)
}

nSims <- 500
n <- seq(10,100,10)
POWER <- numeric(length(n))

for(j in 1:length(n)){
  pval <- numeric(nSims)
  for(i in 1:nSims){
    X <- runif(n[j], 0, 20)
    group <- factor(rep(1:2, each = n[j]/2))
    Y <- Counts(X,group)
    fit <- glm.nb(Y ~ X + group,link = "log")
    pval[i] <- coef(summary(fit))["group2", "Pr
          (>|z|)"]
  }
  POWER[j] <- sum(pval < 0.05)/nSims
}
```





Power to detect multiple effects

Regression models often contain multiple parameters with different effect sizes and we might be interested in knowing the power for different effects.

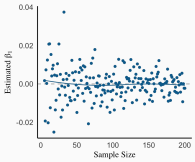


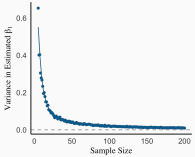
Blue: $\beta_1 = 1$; Red: $\beta_2 = 0.5$; Purple: $\beta_3 = .01$

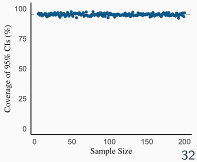
Quality of estimates

Sometimes we're not just interested in whether we can detect significance, but in how close to the 'true' parameter values we can get. This requires understanding bias, variance and coverage.

```
Linear <- function(x) {
  B_0 <- 2
  B_1 <- 0.5
  mu = B_0 + B_1*x
  rnorm(n = length(x), mean = mu, sd = 8)}

nSims <- 500
n <- seq(5,200,1)
BIAS <- numeric(length(n))
VARIANCE <- numeric(length(n))
COVERAGE <- numeric(length(n))
for(j in 1:length(n)){
  bias <- numeric(nSims)
  coverage <- numeric(nSims)
  for(i in 1:nSims){
    X <- runif(n[j], 0, 20)
    Y <- Linear(X)
    fit <- lm(Y ~ X)
    bias[i] <- coef(fit)[2] - 0.5
    coverage[i]<-(0.5<=confint(fit)[2,2]&0.5>= confint(fit)[2,1])
  }
  BIAS[j] <- mean(bias)
  VARIANCE[j] <- var(bias)
  COVERAGE[j] <- sum(coverage)/nSims
}
```







Pseudocode

Simulations and pseudocode

All of the tools we've discussed today involve putting together R code with multiple, interconnected steps.

When designing simulation studies or power analyses it is always a good idea to start by writing down a pseudocode.

Pseudocode is a plain language description of the steps in a computer algorithm (i.e., a recipe that you expect to follow).

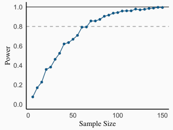
The more detail you put in your pseudocode, the easier it will be to write the computer code (you can also use this as a guide when describing your methods).

A pseudocode can also be translated to another coding language more easily.

Example pseudocode

THE UNIVERSITY OF BRITISH COLUMBIA
Okanagan Campus

The pseudocode for estimating the statistical power of detecting a linear trend would look like this:



```
1. Define $\sigma$, $\beta_0$, $\beta_1$, the # of sims, and the sample sizes to test.
2. Build a function for simulating from simple linear model.
3. Build a for loop that simulates data and fits models nSims times.
4. Extract and store the p values from each iteration.
5. Nest this in a for loop that iterates over the different sample sizes.
6. Calculate the power for each sample size and store the results.
7. Plot the results for interpretation.
```

Pseudocode examples

Hilborn & Mangel (1997) is full of examples of pseudocode if you’re interested in seeing what that looks like.

Take home messages

Simulations can put practicing biologists on equal footing with experienced mathematicians. This makes them potentially powerful tools for understanding biological systems and generalising the results of our analyses.

Simulations can also help us understand how a system might be expected to respond to conditions that we can not/couldn't measure.

Always remember that the more moving pieces a simulation setup has, the harder the outcomes can be to understand. Carefully tailored simulations are often more informative than complex simulations that we can't keep track of.

Simulations can also help us understand the statistical power of our data/model ... but good experimental design is more important than simulation based power analysis (don't overthink it).

References

Hilborn, R. & Mangel, M. (1997). *The ecological detective: confronting models with data*. vol. 28. Princeton University Press.

Bolker, B. M. (2008). Ecological models and data in R. Princeton University Press.