Overview of Linear Models, Bayesian Inference, and STAN

Key Concepts

- Linear Model Specification
- Simulating Data in R
- Fitting Linear Models in R
- Bayesian Inference
- Fitting Bayesian Models with Stan

Linear Model Specification

Linear models provide the foundation for most statistical analyses:

- *univariate regression models:* $y = \beta_0 + \beta_1 x + \epsilon$, *where* $\epsilon \sim N(0, \sigma^2)$

- *regression models(matrix notation):* $\mathbf{y} = X\beta + \epsilon$, *where* $\epsilon \sim N(\mathbf{0}, \Sigma)$ *and* $\Sigma = \sigma^2 I$

- *two sample t-test:* $y = \beta_0 + \beta_1 x_{I=group1} + \epsilon$, *where* $\epsilon \sim N(0, \sigma^2)$

One assumption, that is often violated in spatial statistics is that the errors are independently distributed.

Simulating Data in R

Simulating “fake” data will be a cornerstone of fitting models in this class.

```r
set.seed(01112021)
# initialize parameters
num_obs <- 100
beta <- 1
sigma <- 2

# simulate data
x <- runif(num_obs, min = -10, max = 10)
y <- x * beta + rnorm(num_obs, mean = 0, sd = sigma)
```

```r
# create figure
tibble(x=x, y=y) %>%
  ggplot(aes(x=x, y=y)) +
  geom_point() +
  theme_bw() +
  geom_smooth(formula = 'y~x', method = 'lm')
```

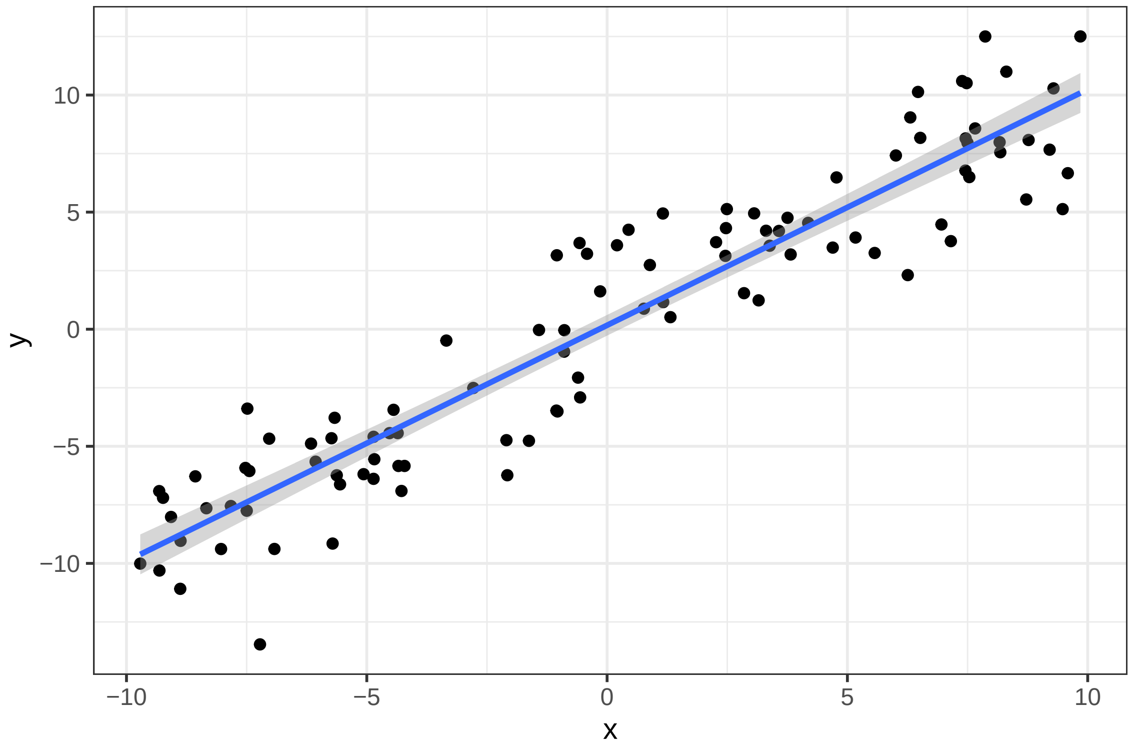

Figure 1: Scatterplot of synthetic data with best fit regression line.

Fitting Linear Models in R

The standard method for fitting linear models in R is with `lm()`.

```
fit_lm <- lm(y ~ x)
summary(fit_lm)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3524 -1.4720  0.1297  1.5266  4.4066
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.16819    0.22278   0.755    0.452
## x            1.00733    0.03755  26.827   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.228 on 98 degrees of freedom
## Multiple R-squared:  0.8802, Adjusted R-squared:  0.8789
## F-statistic: 719.7 on 1 and 98 DF,  p-value: < 2.2e-16
```

A quick interlude about model interpretation.

- *Avoid casual language or even language that suggest casuality. For instance "a one unit increase in x results in a one unit increase in y". Rather than discussing changes* ***within a unit****, we should talk about changes* ***between units.*** *Hence, the expected difference between two units that differ by one unit in x is XYZ.*

- P-values… The ASA Statement on p-values is "required reading". *Know what the can and cannot do. If you choose to use p-values, always include uncertainty in the effect size: evidence (p-value) and effects.*

The `arm` *package, associated with Gelman and Hill's textbook,* Data Analysis Using Regression and Multilevel/Hierarchical Models, *has a function that hides p-values.*

```
display(fit_lm)
```

```
## lm(formula = y ~ x)
##             coef.est coef.se
## (Intercept) 0.17     0.22
## x           1.01     0.04
## ---
## n = 100, k = 2
## residual sd = 2.23, R-Squared = 0.88
```

An alternative framework for fitting regression models is to use the `rstanarm` package and the associated `stan_glm()` functionality.

```
synthetic_data <- tibble(x = x, y = y)
fit_stan <- stan_glm(y ~ x, data = synthetic_data, refresh = 0)
print(fit_stan)
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      y ~ x
##  observations: 100
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 0.2    0.2
## x           1.0    0.0
##
## Auxiliary parameter(s):
##       Median MAD_SD
## sigma 2.2    0.2
##
## ------
## * For help interpreting the printed output see ?print.stanreg
## * For info on the priors used see ?prior_summary.stanreg
```

Bayesian Inference

`stan_glm()` is a Bayesian procedure that uses Markov Chain Monte Carlo to generate samples that represent distribution for each parameter.

While there are coefficient estimates similar to `lm`

```
coef(fit_stan)
```

```
## (Intercept)           x
##   0.1747493   1.0075860
```

Rather than just a point estimate that maximizes the likelihood, the object actually contains a set of samples for each of the parameter values.

```
fit_stan %>% as.data.frame() %>% head(10)
```

```
##    (Intercept)         x    sigma
## 1   0.18706157 1.0098023 2.529174
## 2   0.09967871 0.9939963 2.079094
## 3   0.08007354 0.9818507 2.215735
## 4   0.04590595 0.9796351 2.108727
## 5   0.02927909 0.9952074 2.133086
## 6   0.49101367 0.9909962 2.364030
## 7   0.26350585 0.9921205 2.271885
## 8   0.22166709 0.9987937 2.009585
## 9   0.10027736 1.0141302 2.488189
## 10  0.67024064 0.9608660 2.247425
```

While the estimation procedure and interval interpretations are fundamentally different, the actual values on the interval are fairly similar.

```
posterior_interval(fit_stan, prob = .95)
```

```
##                   2.5%     97.5%
## (Intercept) -0.2773379 0.6195185
## x            0.9331275 1.0801073
## sigma        1.9687029 2.5849950
```

```
confint(fit_lm)
```

```
##                  2.5 %    97.5 %
## (Intercept) -0.2739015 0.6102871
## x            0.9328161 1.0818436
```

So what is hiding in this model specification?

```r
synthetic_data <- tibble(x = x, y = y)
fit_stan <- stan_glm(y ~ x, data = synthetic_data, refresh = 0)
```

priors, of course

```r
prior_summary(fit_stan)
```

```
## Priors for model 'fit_stan'
## ------
## Intercept (after predictors centered)
##   Specified prior:
##     ~ normal(location = 0.24, scale = 2.5)
##   Adjusted prior:
##     ~ normal(location = 0.24, scale = 16)
##
## Coefficients
##   Specified prior:
##     ~ normal(location = 0, scale = 2.5)
##   Adjusted prior:
##     ~ normal(location = 0, scale = 2.7)
##
## Auxiliary (sigma)
##   Specified prior:
##     ~ exponential(rate = 1)
##   Adjusted prior:
##     ~ exponential(rate = 0.16)
## ------
## See help('prior_summary.stanreg') for more details
```

Bayes Rule *Bayes Rule (Theorem) is a common topic in probability courses, and while it carries the "Bayes" moniker it is not synonymous with Bayesian inference.*

$$Pr(A|B) = \frac{Pr(A \cap B)}{Pr(B)} = \frac{Pr(B|A)Pr(A)}{Pr(B)},$$

where $Pr(A|B)$ is a conditional probability of event A, given event B.

The classic example of Bayes Rule focuses on medical testing. So consider a COVID-19 testing scenario. Assume we want to know the probability that an individual is positive given they received a positive test or Pr(An individual is positive | test is positive). Let:

- Pr(An individual is Positive) = .10 (Note this is generally a population prevalence)
- Pr(test is positive | an individual is positive) = .93 **(sensitivity)**
- Pr(test is positive | an individual is not positive) = .02 **(1 - specificity)**
- Pr(test is positive) = Pr(test is positive | an individual is positive) x Pr(an individual is positive) + Pr(test is positive | an individual is not positive) x Pr(an individual is not positive)

```r
prev <- .10
sens <- .93
spec <- .98

Pr_pos_pos = (sens * prev) / (sens * prev + (1 - spec) * (1 - prev))
```

Then the probability that an individual with a positive test has COVID-19 is 0.84.

As mentioned, using Bayes's theorem does not equate with Bayesian inference, but rather what we have just done is an exercise with conditional probability.

Bayesian inference requires a prior probability distribution for model parameters that is specified before collecting data (or analyzing data). This represents prior "beliefs" about the plausible range of parameter values. Let this be generically denoted $p(\theta)$.

Then once data has been collected, a sampling distribution for the data is specified. In many cases (t-test, regression, etc...) this is a normal distribution. Let the probability distribution be stated as $p(X|\theta)$.

Maximum likelihood estimator use the sampling distribution, or more specifically the likelihood, to select parameter values ($\hat{\theta}_{MLE}$) that maximize the likelihood.

With MLEs, uncertainty intervals are calculated analytically using the distribution of $\hat{\theta}_{MLE}$*, sometimes requiring asymptotics.*

Bayesian inference uses another mechanism, Bayes's Theorem, to conduct inference.

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)},$$

Inferences are made using the posterior distribution, $p(\theta|x)$ *which is a probability distribution that contains the range of plausible values for the parameter(s)* θ.

In some situations, the posterior distribution can be analytically calculated. However, in many scenarios, $p(x)$ requires integration and does not have an analytical solution.

In these situations, Markov Chain Monte Carlo (MCMC) is used to compute the posterior distribution. The ***`stan_glm()`*** *functions call STAN to implement MCMC behind the curtain to find posterior distributions of parameters. MCMC can be written in any language, but STAN has some nice advantages (Hamiltonian Monte Carlo).*

STAN can be called directly from R (or R Markdown documents.). The basic structure of a STAN script looks like:

```
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;
}
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma;
}
model {
  y ~ normal(alpha + beta * x, sigma);
}
```