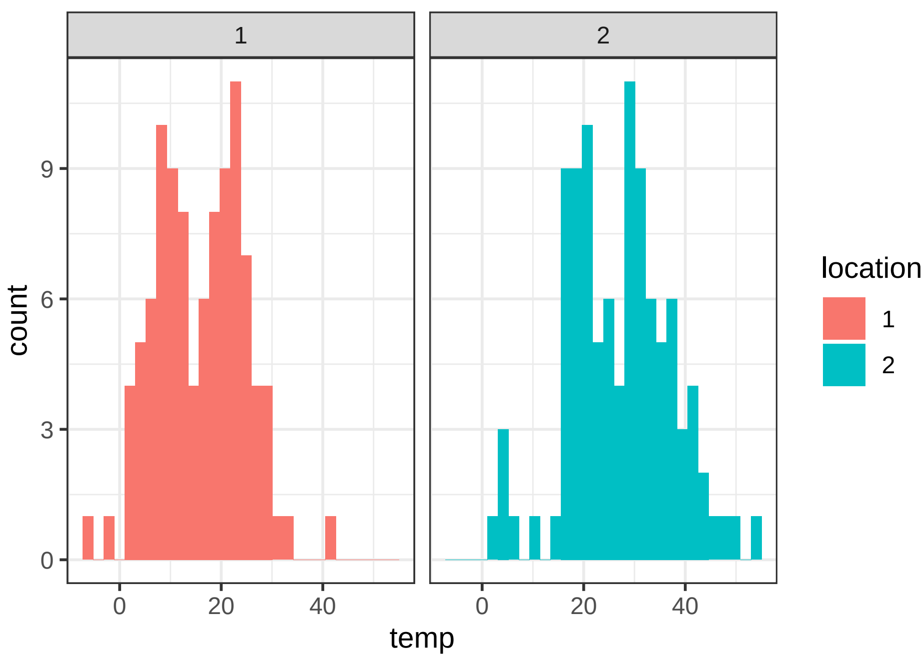
GP demo

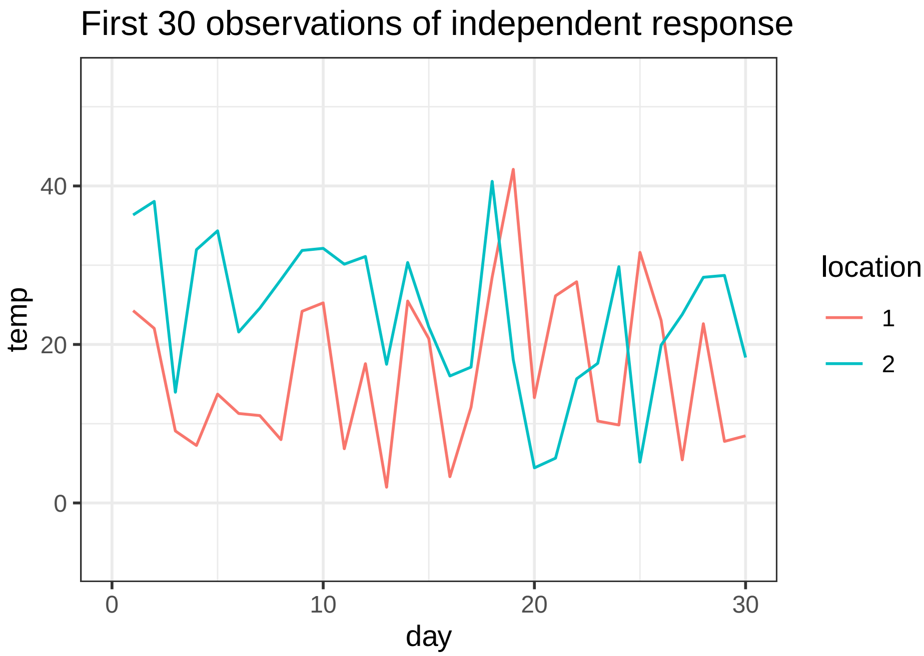
Multivariate Normal Distribution First we will start with the a bivariate normal distribution: y ~ N(theta,sigma), where theta is a mean vector and sigma = sigmasq * I is a covariance matrix.

To provide a motivating context, not consider jointly estimating the temperature at Bridger Bowl *and* Big Sky Resort.

Independent bivariate normal Simulate a set of temperature values from each location, where the temperature values are independent (sigma = sigmasq * I)

```
library(mnormt)
n <- 100
theta <- c(15,25)
sigma <- diag(2) * 100
fake_temperatures <- rmnorm(n, theta , sigma)
```

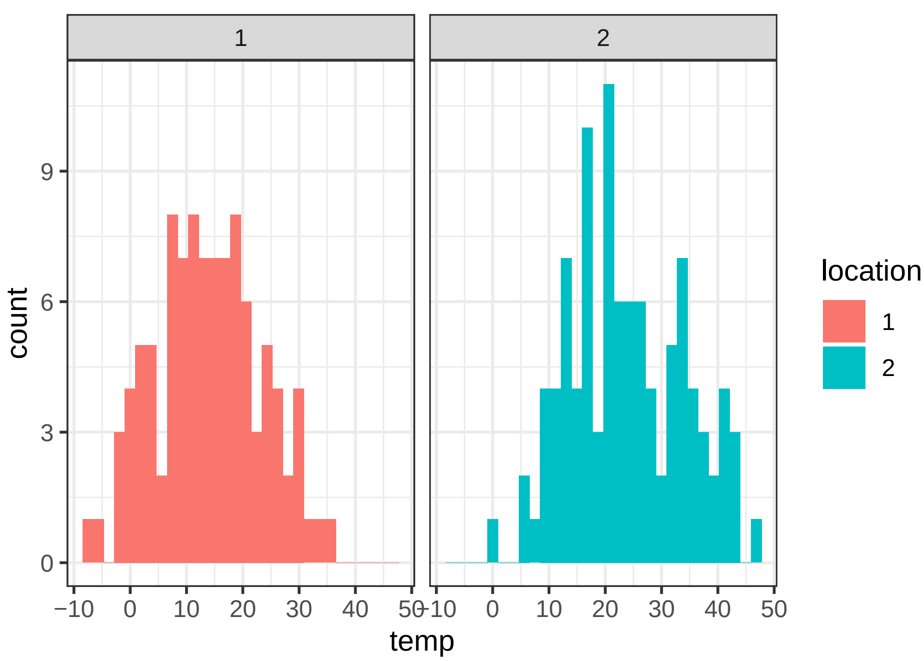


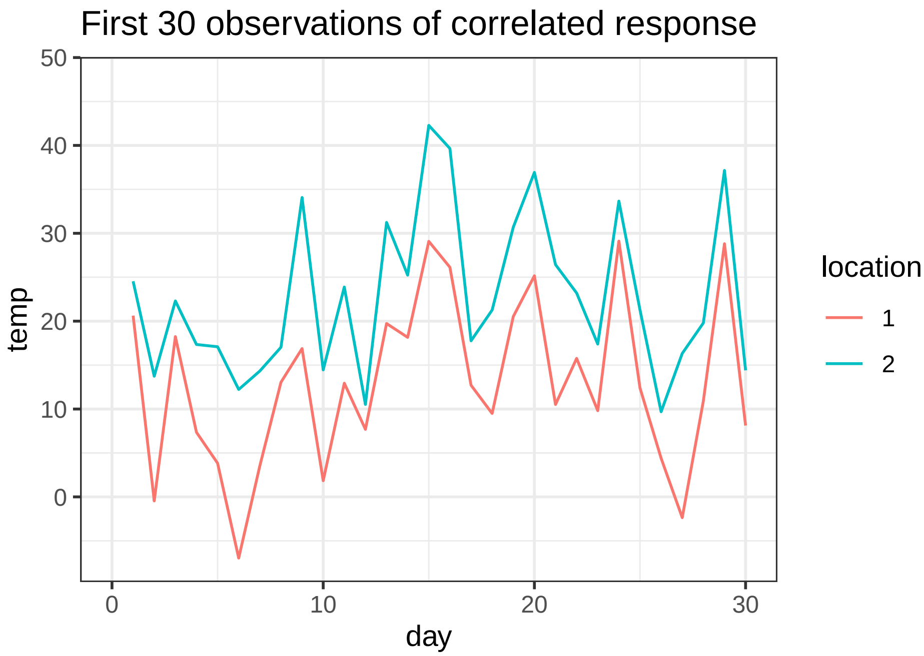


Correlated bivariate normal Simulate a set of temperature values from each location, where the temperature values are not independent (sigma = sigmasq * H), where H is a correlation matrix. (Note there are some constraints we will discuss later)

```
sigma <- matrix(c(1, .9, .9, 1), nrow = 2, ncol = 2) * 100
fake_temperatures_corr <- rmnorm(n, theta , sigma)
```

Then create a few graphs to show marginal distribution of temperature as well as how the temperatures evolve in time.





In many statistical models there is an assumption about independence. When independence is violated, uncertainty is under estimated and in incorrect inferences can be made.

While lack of independence often has a negative connotation, in spatial statistics we can actually exploit correlation. For instance, by knowing the temperature at the weather station at Bozeman High School or Bridger Bowl, we can estimate temperature at other locations.

Conditional Normal distribution In general,

$$\underline{y_1}|\underline{y_2} \sim N\left(X_1\beta + \Sigma_{12}\Sigma_{22}^{-1}\left(\underline{y_2} - X_2\beta\right), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)$$

Now there is one more location that we are interested in learning the temperature, maybe Rendezvous Ski Trails in West Yellowstone.

Let's assume that

$$\begin{bmatrix} y_{bridger} \\ y_{bigsky} \\ y_{rendezvous} \end{bmatrix} \sim N\left(\begin{bmatrix} 15 \\ 15 \\ 15 \end{bmatrix}, 100 \begin{bmatrix} 1 & .3 & .2 \\ .3 & 1 & .5 \\ .2 & .5 & 1 \end{bmatrix}\right)$$

$$y_r|\underline{y_2} \sim N\left(X_1\beta + \Sigma_{12}\Sigma_{22}^{-1}\left(\underline{y_2} - X_2\beta\right), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right)$$

Conditional on the values from Bridger Bowl and Big Sky, we can construct the distribution for the Rendezvous temperature. Add this to a graph with a marginal temperature.

```
fake_temperatures1 <- c(30,30)
mu_given <- mu[3] + Sigma[3,1:2] %*% solve(Sigma[1:2,1:2]) %*% (fake_temperatures1 - mu[1:2])
sigma_given <- Sigma[3,3] - Sigma[3,1:2] %*% solve(Sigma[1:2,1:2]) %*% Sigma[1:2, 3]

x_seq <- seq(-15, 55, by = 1)

tibble(x = rep(x_seq,2),
       dens = c(dnorm(x_seq, mu[3], sqrt(Sigma[3,3])),
                dnorm(x_seq, mu_given, sqrt(sigma_given))),
```

```
    type = rep(c('marginal','conditional'), each = length(x_seq) )) %>%
ggplot(aes(x = x, y = dens, group = type, color = type)) +
geom_line() + theme_bw() +
geom_vline(xintercept = fake_temperatures1)
```

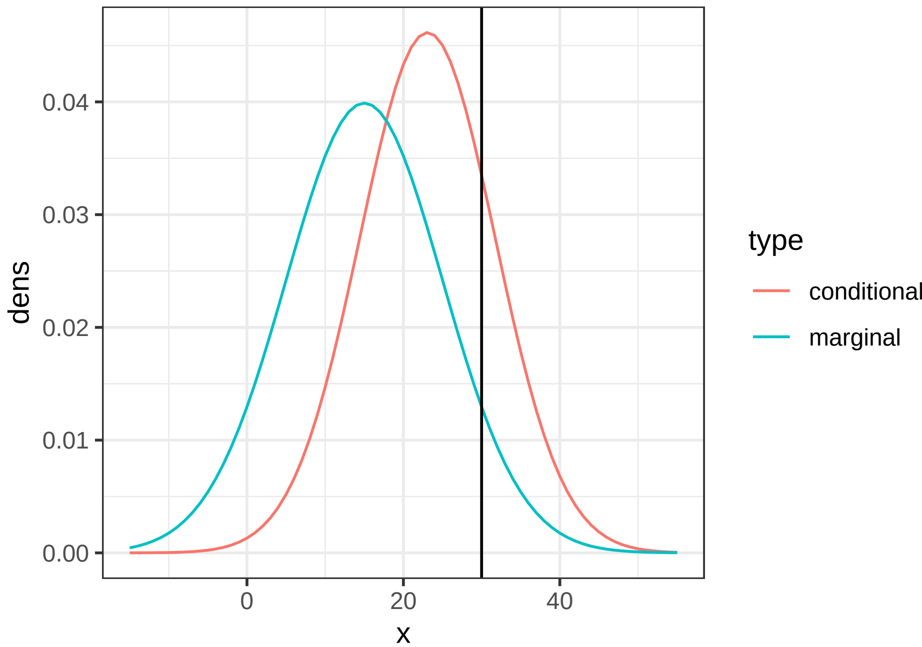


Figure 1: Black bars represent observed temperature at Big Sky and Bridger

GP regression We will simulate a Gaussian process regression, where

$$\underline{y} \sim N\left(X\underline{\beta}, \sigma^2 H(\phi) + \tau^2\right),$$

where $H(\phi)$ is a matrix where entry $h_{ij} = \exp(-d_{ij}/\phi)$ where d_{ij} is the distance between points i and j.

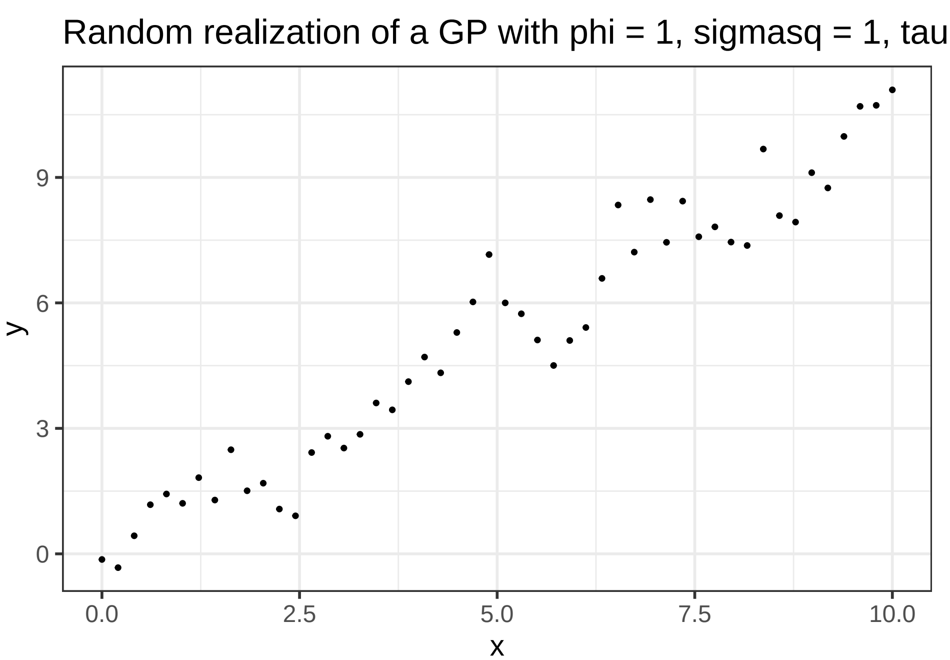
1. Set up the model parameters

```
phi <- 1
sigmasq <- 1
tausq <- .2
n <- 50
x <- seq(0, 10, length.out = n)
beta <- 1
d <- sqrt(plgp::distance(x))
eps <- sqrt(.Machine$double.eps)
H <- exp(-d/phi) + diag(eps, n)
```

2. Simulate a finite realization from the process

```
y <- rmnorm(1, x * beta,sigmasq * H + tausq * diag(n))
```

```
reg_fig <- tibble(y = y, x = x) %>% ggplot(aes(y=y, x=x)) +
  theme_bw() + ggtitle('Random realization of a GP with phi = 1, sigmasq = 1, tausq = .2') +
  geom_point(size = .5)
reg_fig
```




```
n_preds <- 50
x_preds <- seq(-1, 11, length.out = n_preds)
d_12 <- sqrt(plgp::distance(x, x_preds))
d_preds <- sqrt(plgp::distance(x_preds))
```

STAN CODE Let's first write stan code to estimate phi, sigmasq, tausq, and mu

```
data {
```

```
  int<lower=0> N; // number of data points
  vector[N] y; // response
  matrix[N,N] dist; // distance matrix
  vector[N] x; // covariate
  int<lower=0> N_preds;
  matrix[N_preds, N_preds] dist_preds;
  matrix[N, N_preds] dist_12;
  vector[N_preds] x_preds; // covariate
}

parameters {
  real<lower = 0.25, upper = 9> phi;
  real<lower = 0> sigmasq;
  real<lower = 0> tausq;
  real beta;
}

transformed parameters{
  vector[N] mu_vec;
  vector[N] tausq_vec;
  corr_matrix[N] Sigma;

  for(i in 1:N) mu_vec[i] = x[i] * beta;
  for(i in 1:N) tausq_vec[i] = tausq;

  for(i in 1:(N-1)){
   for(j in (i+1):N){
     Sigma[i,j] = exp((-1)*dist[i,j]/ phi);
     Sigma[j,i] = Sigma[i,j];
   }
 }
 for(i in 1:N) Sigma[i,i] = 1;

}

model {
  y ~ multi_normal(mu_vec ,sigmasq * Sigma + diag_matrix(tausq_vec));
  phi ~ inv_gamma(10, 10);
  sigmasq ~ inv_gamma(10, 10);
  tausq ~ inv_gamma(10, 2);
  beta ~ normal(0, 10);
}


generated quantities {
  vector[N_preds] y_preds;
  vector[N] y_diff;
  vector[N_preds] mu_preds;
  corr_matrix[N_preds] Sigma_preds;
  vector[N_preds] tausq_preds;
  matrix[N, N_preds] Sigma_12;

  for(i in 1:N_preds) tausq_preds[i] = tausq;
  for(i in 1:N_preds) mu_preds[i] = x_preds[i] * beta;
```

```
  for(i in 1:N) y_diff[i] = y[i] - x[i] * beta;


  for(i in 1:(N_preds-1)){
   for(j in (i+1):N_preds){
     Sigma_preds[i,j] = exp((-1)*dist_preds[i,j]/ phi);
     Sigma_preds[j,i] = Sigma_preds[i,j];
   }
 }
 for(i in 1:N_preds) Sigma_preds[i,i] = 1;

   for(i in 1:(N)){
   for(j in (1):N_preds){
     Sigma_12[i,j] = exp((-1)*dist_12[i,j]/ phi);
   }
 }

 y_preds = multi_normal_rng(mu_preds + (sigmasq * Sigma_12)' * inverse(sigmasq * Sigma) * (y_diff),
 sigmasq * Sigma_preds + diag_matrix(tausq_preds) - (sigmasq * Sigma_12)' * inverse(sigmasq * Sigma + di
}
```

```r
Reg_params <- stan("GP_reg.stan",
                  data=list(N = n,
                            y = y,
                            x = x,
                            dist = d,
                            N_preds = n_preds,
                            dist_preds = d_preds,
                            dist_12 = d_12,
                            x_preds = x_preds),
                  iter = 2000)

#shinystan::launch_shinystan(Reg_params)
```

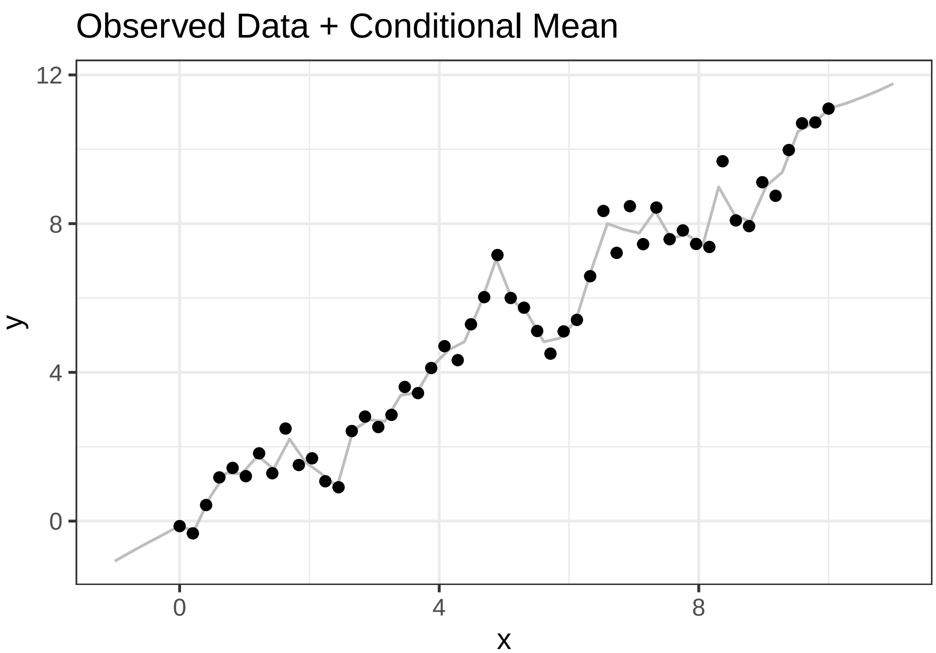
```r
print(Reg_params, pars = c('phi', 'beta','sigmasq', 'tausq',
                           'y_preds[1]', 'y_preds[50]'))
```

```
## Inference for Stan model: GP_reg.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##              mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## phi          1.03    0.01 0.31  0.59  0.82  0.98  1.19  1.77  3559    1
## beta         1.05    0.00 0.07  0.93  1.01  1.05  1.09  1.19  3103    1
## sigmasq      0.92    0.00 0.24  0.55  0.76  0.89  1.06  1.49  3582    1
## tausq        0.20    0.00 0.05  0.12  0.16  0.19  0.23  0.32  3696    1
## y_preds[1]  -1.08    0.02 1.01 -3.03 -1.76 -1.10 -0.40  0.96  3840    1
## y_preds[50] 11.76    0.02 1.09  9.65 11.01 11.79 12.49 13.92  3780    1
##
## Samples were drawn using NUTS(diag_e) at Tue Mar  2 14:56:04 2021.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

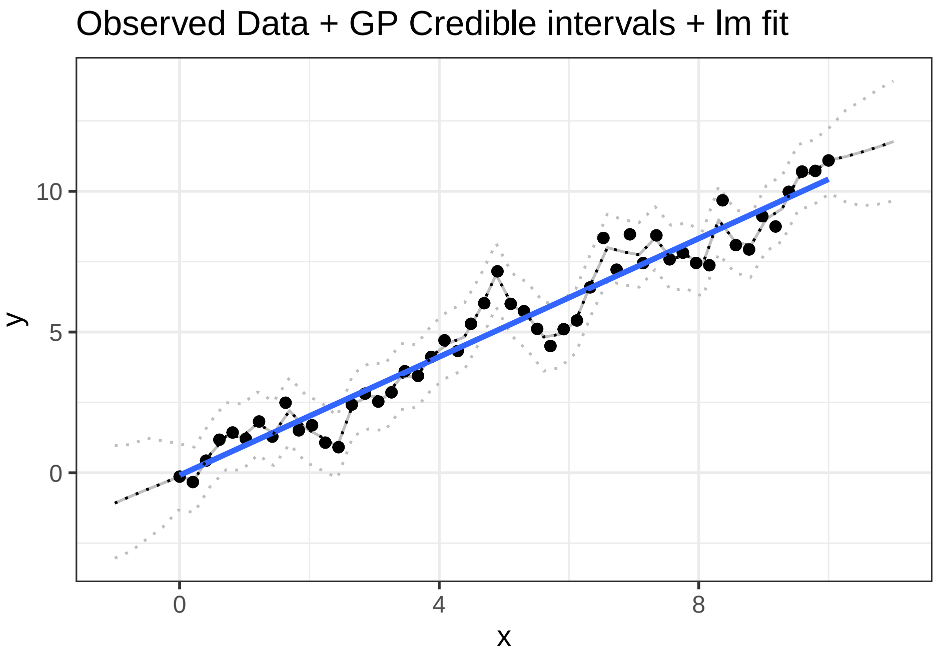
```r
preds <- extract(Reg_params)['y_preds']$y_preds
mean_preds <- colMeans(preds)
lower_preds <- apply(preds, 2, quantile, probs = .025)
upper_preds <- apply(preds, 2, quantile, probs = .975)

mean_line <- tibble(y_mean = mean_preds, xnew = x_preds,
                    lower = lower_preds, upper = upper_preds)
data_and_mean <- reg_fig +
  geom_line(aes(y = y_mean, x = xnew), inherit.aes = F, data = mean_line, color = 'gray') +
  geom_point() +
  ggtitle("Observed Data + Conditional Mean")
data_and_mean
```

Making Predictions




```r
data_and_mean +
  geom_line(aes(y = upper, x = xnew), inherit.aes = F,
            data = mean_line, color = 'gray', linetype = 3) +
  geom_line(aes(y = lower, x = xnew), inherit.aes = F,
            data = mean_line, color = 'gray', linetype = 3) +
  ggtitle('Observed Data + GP Credible intervals + lm fit') +
  geom_point() +
  geom_smooth(method ='lm', formula = 'y~x', se = F) +
  geom_line(aes(y = y_mean, x = xnew), inherit.aes = F,
            data = mean_line, color = 'black', linetype = 3)
```


Observed Data + GP Credible intervals + lm fit
y
10
5
0
0
4
8
x