# F1 Specific Simulations

Cole

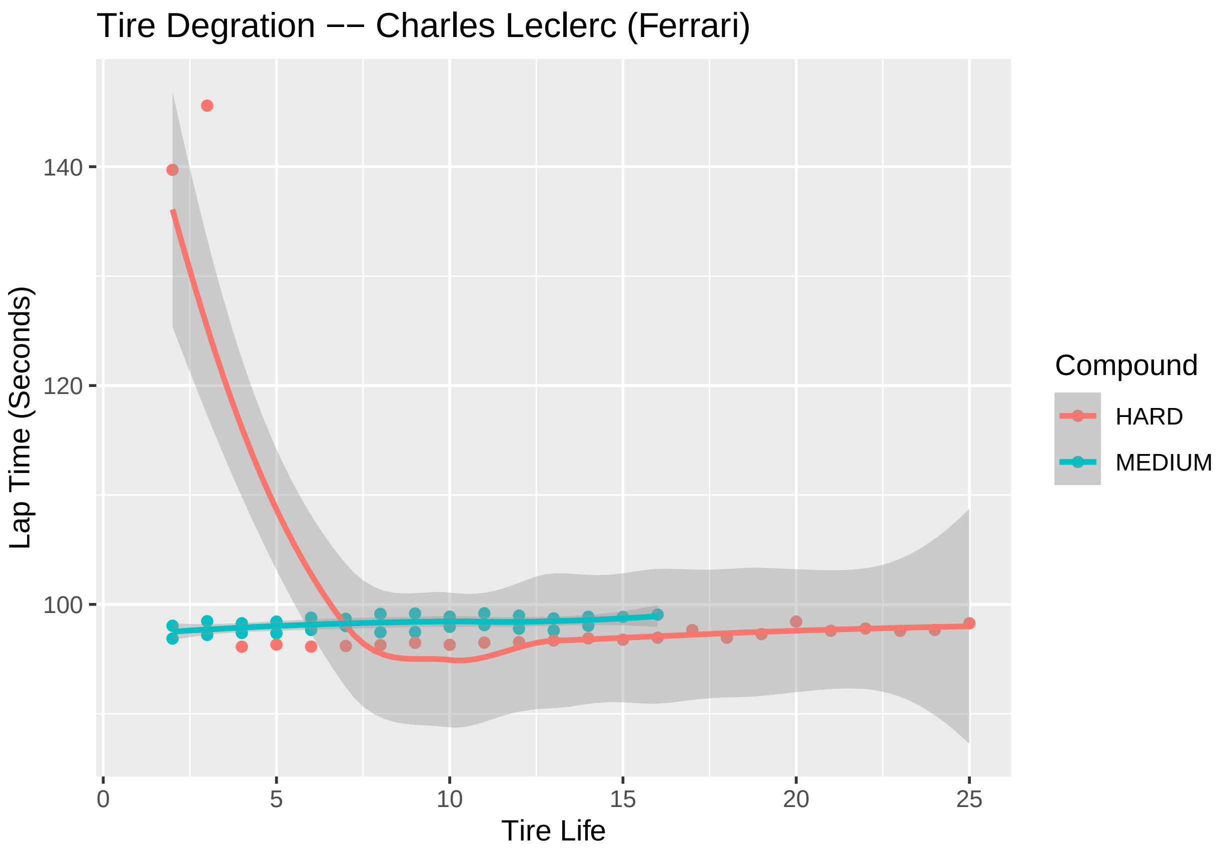
06-11-2025

Here are a couple of example tire degradation plots from actual race data.

```r
bahrain_laps <- read_csv("bahrain_laps.csv")

filter(bahrain_laps, Driver == "LEC", is.na(PitOutTime) & is.na(PitInTime)) %>%
    ggplot(mapping = aes(x = TyreLife, y = LapTime, colour = Compound)) +
    geom_point() +
    geom_smooth() +
    labs(title = "Tire Degration -- Charles Leclerc (Ferrari)",
         x = "Tire Life",
         y = "Lap Time (Seconds)")
```

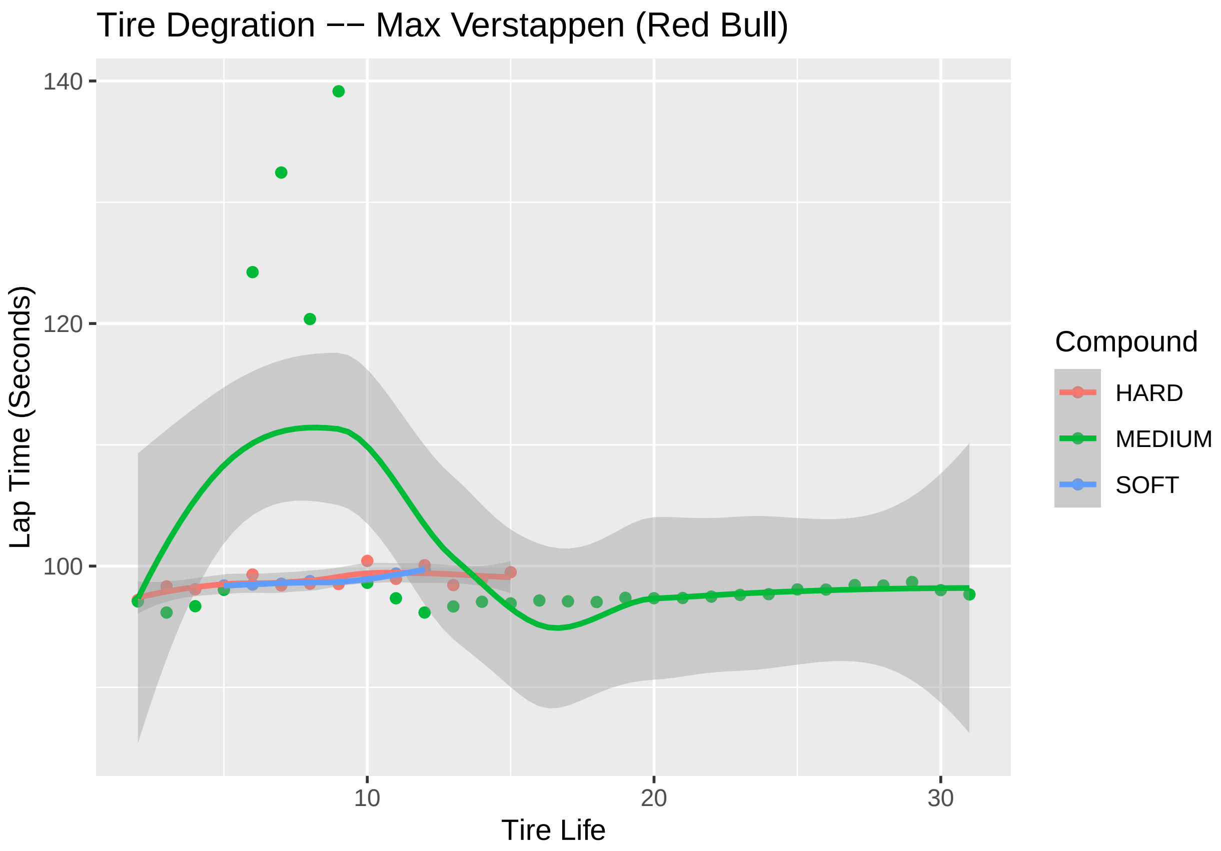
```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



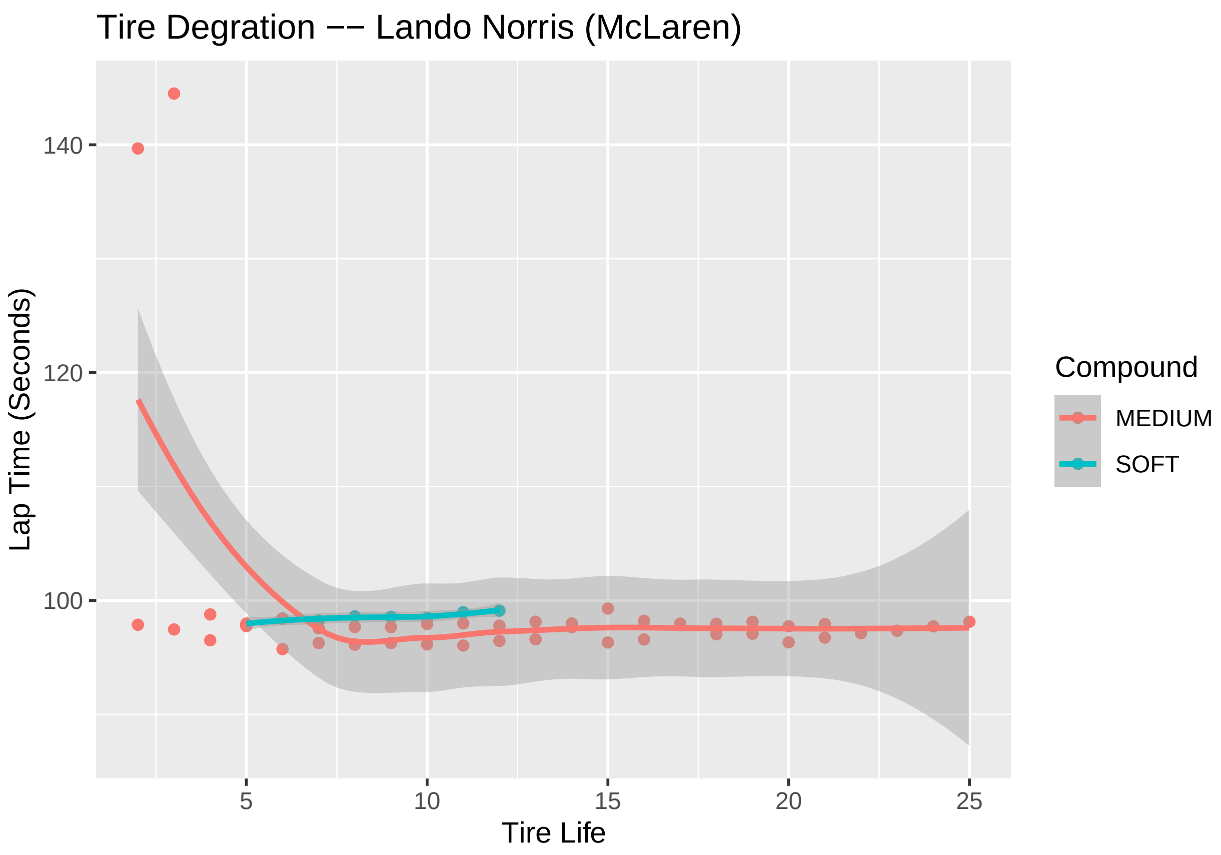
```r
filter(bahrain_laps, Driver == "VER", is.na(PitOutTime) & is.na(PitInTime)) %>%
    ggplot(mapping = aes(x = TyreLife, y = LapTime, colour = Compound)) +
    geom_point() +
    geom_smooth() +
    labs(title = "Tire Degration -- Max Verstappen (Red Bull)",
         x = "Tire Life",
         y = "Lap Time (Seconds)")
```

```
## Warning: Removed 1 row containing non-finite outside the scale range (`stat_smooth()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



```r
filter(bahrain_laps, Driver == "NOR", is.na(PitOutTime) & is.na(PitInTime)) %>%
    ggplot(mapping = aes(x = TyreLife, y = LapTime, colour = Compound)) +
    geom_point() +
    geom_smooth() +
    labs(title = "Tire Degration -- Lando Norris (McLaren)",
         x = "Tire Life",
         y = "Lap Time (Seconds)")
```

```
## Warning: Removed 1 row containing non-finite outside the scale range (`stat_smooth()`).
## Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



Based on these, it is reasonable to assume a maximum stint length of 30 laps (that seems extreme though, so I'll probably knock it down to 25). We will also assume that the fastest lap pace is 96 seconds and the true rate of degredation is .1 seconds per lap.

## Deterministic Process

The first model we simulate from will be

$$y_t = \alpha_t + \epsilon_t \tag{1}$$
$$\alpha_{t+1} = \alpha_t + \nu \tag{2}$$

where $\epsilon_t \sim N(0, .25)$ and $\nu = .1$.

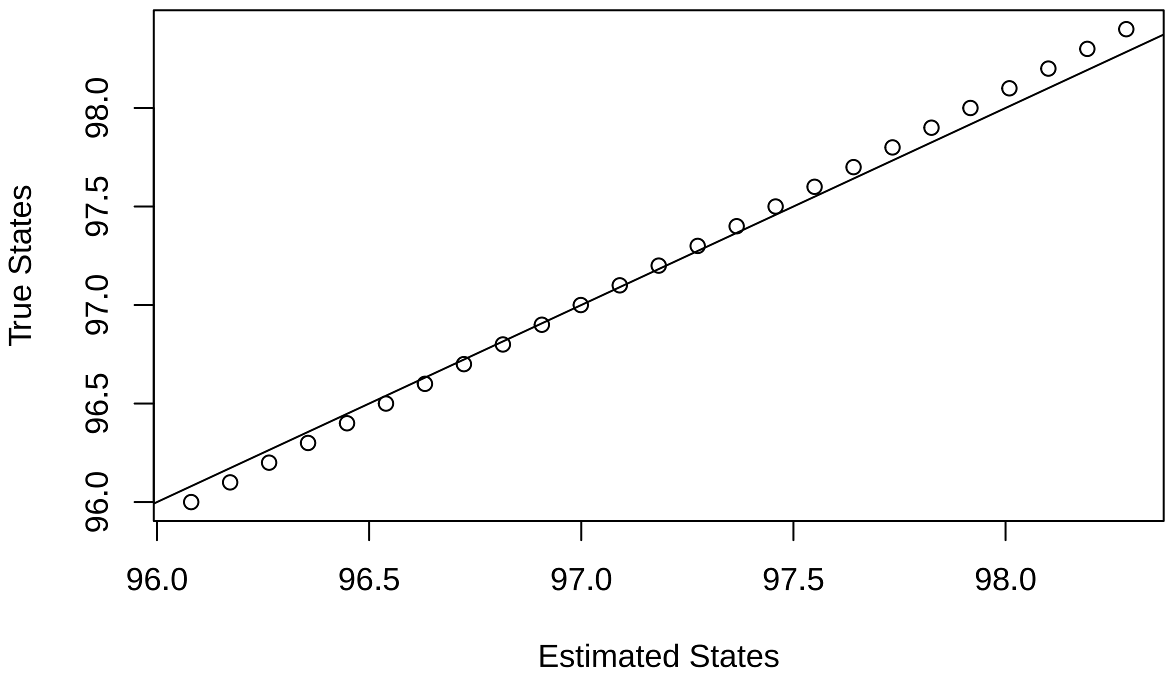
```r
# Number of states
N <- 25

# Nu
nu <- .1

# Initialize alpha vector
alpha <- rep(NA, N)
alpha[1] <- 96

# Store true latent states
for(i in 1:(length(alpha)-1)) {
    alpha[i+1] <- alpha[i] + nu
}

# Vector of errors
epsilon <- rnorm(n = N, mean = 0, sd = .25)

# Simulated observations
y <- alpha + epsilon
```

Now we will fit the model with stan and try to recover the parameters. The Linear_SSM.stan file fits a linear SSM with observation error and a deterministic linear process equation.

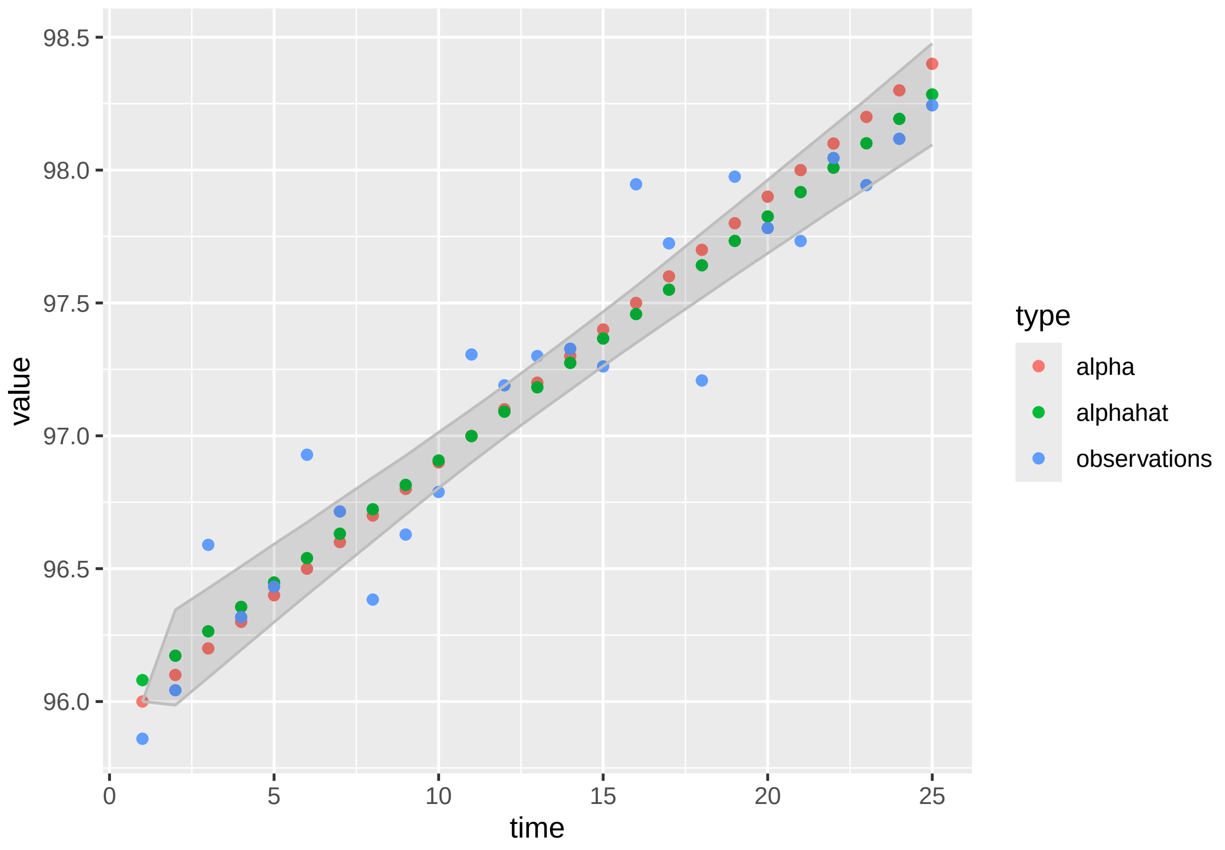
```r
data_stan <- list(TT = length(y), y = y, z0 = 0)

stan_test <- stan(file = "Linear_SSM.stan",
                  data = data_stan,
                  chains = 3, iter = 3000)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.22 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 1: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 1: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 1: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 1: Iteration: 1200 / 3000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1500 / 3000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1501 / 3000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1800 / 3000 [ 60%]  (Sampling)
## Chain 1: Iteration: 2100 / 3000 [ 70%]  (Sampling)
## Chain 1: Iteration: 2400 / 3000 [ 80%]  (Sampling)
## Chain 1: Iteration: 2700 / 3000 [ 90%]  (Sampling)
## Chain 1: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.02 seconds (Warm-up)
## Chain 1:                0.018 seconds (Sampling)
## Chain 1:                0.038 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 1e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 2: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 2: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 2: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 2: Iteration: 1200 / 3000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1500 / 3000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1501 / 3000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1800 / 3000 [ 60%]  (Sampling)
## Chain 2: Iteration: 2100 / 3000 [ 70%]  (Sampling)
## Chain 2: Iteration: 2400 / 3000 [ 80%]  (Sampling)
## Chain 2: Iteration: 2700 / 3000 [ 90%]  (Sampling)
## Chain 2: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.026 seconds (Warm-up)
## Chain 2:                0.019 seconds (Sampling)
## Chain 2:                0.045 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 1e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.01 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 3000 [  0%]  (Warmup)
## Chain 3: Iteration:  300 / 3000 [ 10%]  (Warmup)
## Chain 3: Iteration:  600 / 3000 [ 20%]  (Warmup)
## Chain 3: Iteration:  900 / 3000 [ 30%]  (Warmup)
## Chain 3: Iteration: 1200 / 3000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1500 / 3000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1501 / 3000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1800 / 3000 [ 60%]  (Sampling)
## Chain 3: Iteration: 2100 / 3000 [ 70%]  (Sampling)
## Chain 3: Iteration: 2400 / 3000 [ 80%]  (Sampling)
## Chain 3: Iteration: 2700 / 3000 [ 90%]  (Sampling)
## Chain 3: Iteration: 3000 / 3000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.029 seconds (Warm-up)
## Chain 3:                0.02 seconds (Sampling)
## Chain 3:                0.049 seconds (Total)
## Chain 3:
```

```r
states <- c(mean(extract(stan_test, pars = c("z1"))[[1]]), colMeans(extract(stan_test, pars = c("z"))[[

plot(x = states, y = alpha,
     xlab = "Estimated States",
     ylab = "True States")
abline(0,1)
```



```r
# Include a plot of time vs. estimated state, true state, and observations
time <- seq(from = 1, to = N, by = 1)

z_stan <- extract(stan_test, pars = c("z"))[[1]]
z_CIl <- c(96,apply(z_stan, 2, quantile, probs=0.025))
z_CIu <- c(96,apply(z_stan, 2, quantile, probs=0.975))

df1 <- tibble(time = time,
              alpha = alpha,
              alphahat = states,
              observations = y,
              z_CIl = z_CIl,
              z_CIu = z_CIu)


df_long <- df1 %>%
  pivot_longer(cols = c(alpha, alphahat,observations), names_to = "type", values_to = "value")

ggplot(df_long, aes(x = time, y = value, color = type)) +
    geom_point() +
    geom_ribbon(aes(ymin = z_CIl, ymax = z_CIu), fill = "black",color = "gray", alpha = .1)
```



```
# Estimated Slope:
mean(extract(stan_test, pars = c("v"))[[1]])
```

```
## [1] 0.09183116
```

```
# Estimated Observation Error:
mean(extract(stan_test, pars = c("sdo"))[[1]])
```

```
## [1] 0.2437622
```

## Process Error

This time we'll simulate from a model that has process error:

$$y_t = \alpha_t + \epsilon_t \tag{3}$$
$$\alpha_{t+1} = \alpha_t + \nu + \omega_t \tag{4}$$

where $\epsilon_t \sim N(0, .15)$, $\omega_t \sim N(0, .05)$ and $\nu = .1$. We'll suppose we have an initial starting alpha of 96.

Now if $\alpha_t$ represents something like the underlying/true pace of the tire, then the "observation error" could really be considered something like error due to driver inconsistencies over a lap. Because of this I think it makes sense to use a process error that is similar to or less than the "observation error".

I will also admit I am a bit partial to this because the model seems to perform much worse when the process error increases to be greater than the observation error.

```r
# Number of states
N <- 25

# Nu
nu <- .1

# Process errors
omega <- rnorm(n = N, mean = 0, sd = .05)

# Initialize alpha vector
alpha <- rep(NA, N)
alpha[1] <- 96

# Store true latennt states
for(i in 1:(length(alpha)-1)) {
    alpha[i+1] <- alpha[i] + nu + omega[i+1]
}

# Vector of observation errors
epsilon <- rnorm(n = N, mean = 0, sd = .15)

# Simulated observations
y <- alpha + epsilon
```

Again we will fit the model with stan and try to recover the parameters. The Basic_SSM2.stan file fits a linear SSM with observation error and process error.

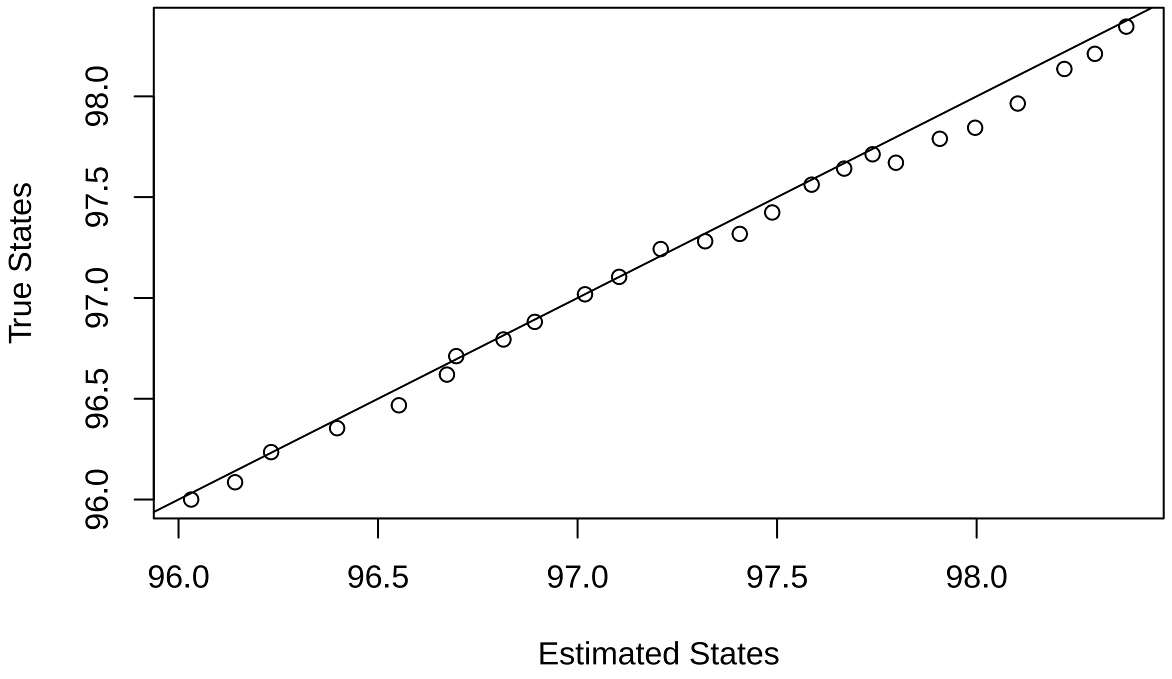
```r
data_stan <- list(TT = length(y), y = y, z0 = 96)

stan_test <- stan(file = "Basic_SSM2.stan",
                  data = data_stan,
                  chains = 3, iter = 15000)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.2e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.22 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 15000 [  0%]  (Warmup)
## Chain 1: Iteration:  1500 / 15000 [ 10%]  (Warmup)
## Chain 1: Iteration:  3000 / 15000 [ 20%]  (Warmup)
## Chain 1: Iteration:  4500 / 15000 [ 30%]  (Warmup)
## Chain 1: Iteration:  6000 / 15000 [ 40%]  (Warmup)
## Chain 1: Iteration:  7500 / 15000 [ 50%]  (Warmup)
## Chain 1: Iteration:  7501 / 15000 [ 50%]  (Sampling)
## Chain 1: Iteration:  9000 / 15000 [ 60%]  (Sampling)
## Chain 1: Iteration: 10500 / 15000 [ 70%]  (Sampling)
## Chain 1: Iteration: 12000 / 15000 [ 80%]  (Sampling)
## Chain 1: Iteration: 13500 / 15000 [ 90%]  (Sampling)
## Chain 1: Iteration: 15000 / 15000 [100%]  (Sampling)
## Chain 1:
```

```
## Chain 1:  Elapsed Time: 0.426 seconds (Warm-up)
## Chain 1:                0.412 seconds (Sampling)
## Chain 1:                0.838 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 15000 [  0%]  (Warmup)
## Chain 2: Iteration:  1500 / 15000 [ 10%]  (Warmup)
## Chain 2: Iteration:  3000 / 15000 [ 20%]  (Warmup)
## Chain 2: Iteration:  4500 / 15000 [ 30%]  (Warmup)
## Chain 2: Iteration:  6000 / 15000 [ 40%]  (Warmup)
## Chain 2: Iteration:  7500 / 15000 [ 50%]  (Warmup)
## Chain 2: Iteration:  7501 / 15000 [ 50%]  (Sampling)
## Chain 2: Iteration:  9000 / 15000 [ 60%]  (Sampling)
## Chain 2: Iteration: 10500 / 15000 [ 70%]  (Sampling)
## Chain 2: Iteration: 12000 / 15000 [ 80%]  (Sampling)
## Chain 2: Iteration: 13500 / 15000 [ 90%]  (Sampling)
## Chain 2: Iteration: 15000 / 15000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 0.722 seconds (Warm-up)
## Chain 2:                0.555 seconds (Sampling)
## Chain 2:                1.277 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 15000 [  0%]  (Warmup)
## Chain 3: Iteration:  1500 / 15000 [ 10%]  (Warmup)
## Chain 3: Iteration:  3000 / 15000 [ 20%]  (Warmup)
## Chain 3: Iteration:  4500 / 15000 [ 30%]  (Warmup)
## Chain 3: Iteration:  6000 / 15000 [ 40%]  (Warmup)
## Chain 3: Iteration:  7500 / 15000 [ 50%]  (Warmup)
## Chain 3: Iteration:  7501 / 15000 [ 50%]  (Sampling)
## Chain 3: Iteration:  9000 / 15000 [ 60%]  (Sampling)
## Chain 3: Iteration: 10500 / 15000 [ 70%]  (Sampling)
## Chain 3: Iteration: 12000 / 15000 [ 80%]  (Sampling)
## Chain 3: Iteration: 13500 / 15000 [ 90%]  (Sampling)
## Chain 3: Iteration: 15000 / 15000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 0.516 seconds (Warm-up)
## Chain 3:                0.555 seconds (Sampling)
## Chain 3:                1.071 seconds (Total)
## Chain 3:
```

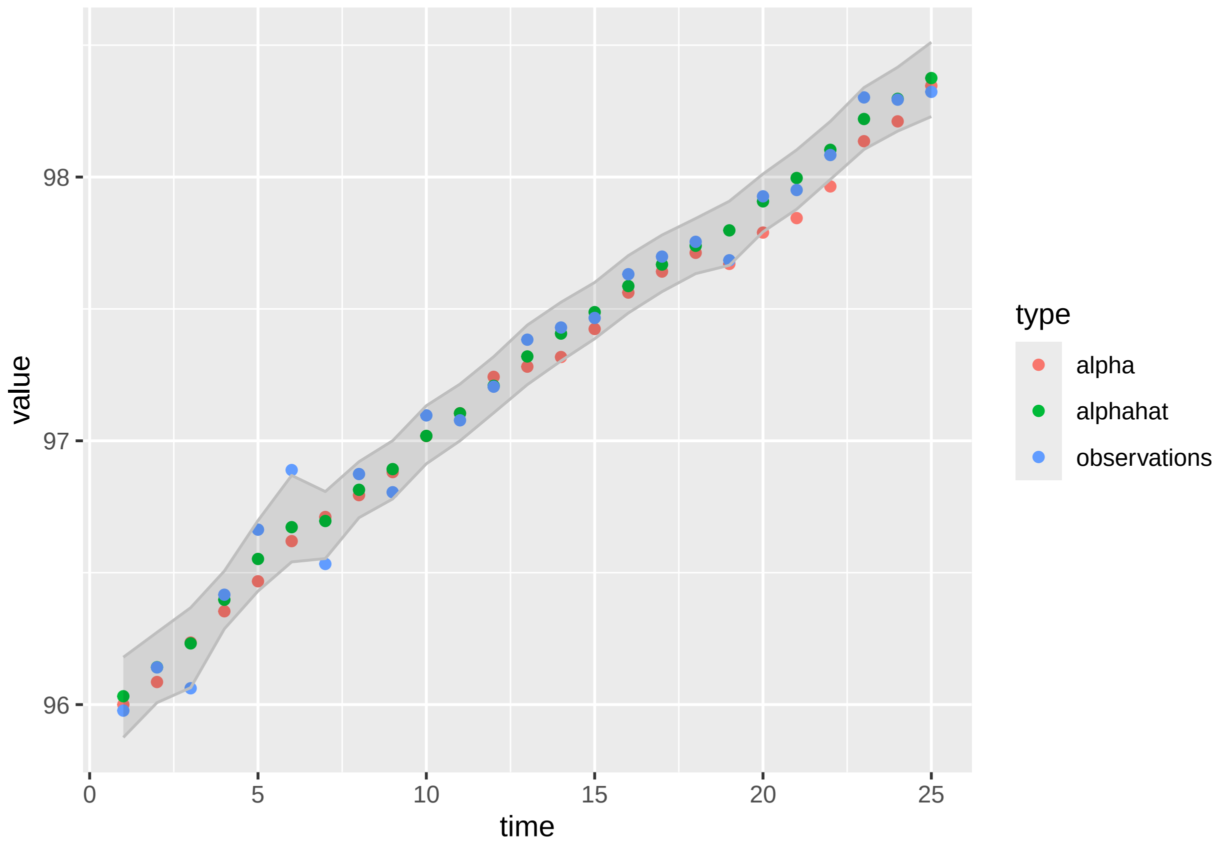
```
## Warning: There were 58 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

## Warning: There were 2 chains where the estimated Bayesian Fraction of Missing Information was low. Se
## https://mc-stan.org/misc/warnings.html#bfmi-low

## Warning: Examine the pairs() plot to diagnose sampling problems

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quant
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

```r
states <- colMeans(extract(stan_test, pars = c("z"))[[1]])

plot(x = states, y = alpha,
     xlab = "Estimated States",
     ylab = "True States")
abline(0,1)
```



```r
# Include a plot of time vs. estimated state, true state, and observations
time <- seq(from = 1, to = N, by = 1)

z_stan <- extract(stan_test, pars = c("z"))[[1]]
z_CIl <- apply(z_stan, 2, quantile, probs=0.025)
z_CIu <- apply(z_stan, 2, quantile, probs=0.975)

df1 <- tibble(time = time,
              alpha = alpha,
              alphahat = states,
              observations = y,
              z_CIl = z_CIl,
              z_CIu = z_CIu)
```

```r
df_long <- df1 %>%
  pivot_longer(cols = c(alpha, alphahat,observations), names_to = "type", values_to = "value")

ggplot(df_long, aes(x = time, y = value, color = type)) +
    geom_point() +
    geom_ribbon(aes(ymin = z_CIl, ymax = z_CIu), fill = "black",color = "gray", alpha = .1)
```



```r
# Estimated Slope:
mean(extract(stan_test, pars = c("v"))[[1]])
```

```
## [1] 0.09773307
```

```r
# Estimated Observation Error:
mean(extract(stan_test, pars = c("sdo"))[[1]])
```

```
## [1] 0.1063137
```

```r
# Estimated Process Error
mean(extract(stan_test, pars = c("sdp"))[[1]])
```

```
## [1] 0.06182498
```

```r
calc_rMSE(alpha,states)
```

```
## [1] 0.06962572
```

```r
calc_rMSE(alpha, y)
```

```
## [1] 0.1122
```

## Multiplicative and Additive Decay

The data seems to support the idea that the decay rate increases as we get further along in the stint. So we can try using the multiplicative and additive decay model.

Here we'll simulate from the model

$$y_t = \alpha_t + \epsilon_t \tag{5}$$
$$\alpha_{t+1} = \beta \cdot \alpha_t + \nu + \omega_t \tag{6}$$

where $\epsilon_t \sim N(0, .15)$, $\beta = 1.05$, $\nu = .1$, $\omega_t \sim N(0, .05)$

```r
# Number of states
N <- 25

# Beta
beta <- 1.005

# Nu
nu <- 0

# Initialize process errors

omega <- rnorm(n = N, mean = 0, sd = .05)

# Initialize alpha vector
alpha <- rep(NA, N)
alpha[1] <- 96

# Store true latent states
for(i in 1:(length(alpha)-1)) {
    alpha[i+1] <- beta*alpha[i] + nu + omega[i+1]
}

# Vector of errors
epsilon <- rnorm(n = N, mean = 0, sd = .15)

# Simulated observations
y <- alpha + epsilon
```

The Mult_Add_SSM.stan file contains the code to fit this model.

```r
data_stan <- list(TT = length(y), y = y, z0 = 96)

stan_test <- stan(file = "Mult_Add_SSM.stan",
                  data = data_stan,
                  chains = 3, iter = 15000)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.7e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.27 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
```

```
## Chain 1:
## Chain 1: Iteration:     1 / 15000 [  0%]  (Warmup)
## Chain 1: Iteration:  1500 / 15000 [ 10%]  (Warmup)
## Chain 1: Iteration:  3000 / 15000 [ 20%]  (Warmup)
## Chain 1: Iteration:  4500 / 15000 [ 30%]  (Warmup)
## Chain 1: Iteration:  6000 / 15000 [ 40%]  (Warmup)
## Chain 1: Iteration:  7500 / 15000 [ 50%]  (Warmup)
## Chain 1: Iteration:  7501 / 15000 [ 50%]  (Sampling)
## Chain 1: Iteration:  9000 / 15000 [ 60%]  (Sampling)
## Chain 1: Iteration: 10500 / 15000 [ 70%]  (Sampling)
## Chain 1: Iteration: 12000 / 15000 [ 80%]  (Sampling)
## Chain 1: Iteration: 13500 / 15000 [ 90%]  (Sampling)
## Chain 1: Iteration: 15000 / 15000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 3.125 seconds (Warm-up)
## Chain 1:                1.443 seconds (Sampling)
## Chain 1:                4.568 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 15000 [  0%]  (Warmup)
## Chain 2: Iteration:  1500 / 15000 [ 10%]  (Warmup)
## Chain 2: Iteration:  3000 / 15000 [ 20%]  (Warmup)
## Chain 2: Iteration:  4500 / 15000 [ 30%]  (Warmup)
## Chain 2: Iteration:  6000 / 15000 [ 40%]  (Warmup)
## Chain 2: Iteration:  7500 / 15000 [ 50%]  (Warmup)
## Chain 2: Iteration:  7501 / 15000 [ 50%]  (Sampling)
## Chain 2: Iteration:  9000 / 15000 [ 60%]  (Sampling)
## Chain 2: Iteration: 10500 / 15000 [ 70%]  (Sampling)
## Chain 2: Iteration: 12000 / 15000 [ 80%]  (Sampling)
## Chain 2: Iteration: 13500 / 15000 [ 90%]  (Sampling)
## Chain 2: Iteration: 15000 / 15000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 3.174 seconds (Warm-up)
## Chain 2:                4.739 seconds (Sampling)
## Chain 2:                7.913 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 15000 [  0%]  (Warmup)
## Chain 3: Iteration:  1500 / 15000 [ 10%]  (Warmup)
## Chain 3: Iteration:  3000 / 15000 [ 20%]  (Warmup)
## Chain 3: Iteration:  4500 / 15000 [ 30%]  (Warmup)
## Chain 3: Iteration:  6000 / 15000 [ 40%]  (Warmup)
## Chain 3: Iteration:  7500 / 15000 [ 50%]  (Warmup)
## Chain 3: Iteration:  7501 / 15000 [ 50%]  (Sampling)
## Chain 3: Iteration:  9000 / 15000 [ 60%]  (Sampling)
## Chain 3: Iteration: 10500 / 15000 [ 70%]  (Sampling)
## Chain 3: Iteration: 12000 / 15000 [ 80%]  (Sampling)
## Chain 3: Iteration: 13500 / 15000 [ 90%]  (Sampling)
## Chain 3: Iteration: 15000 / 15000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 3.149 seconds (Warm-up)
## Chain 3:                3.099 seconds (Sampling)
## Chain 3:                6.248 seconds (Total)
## Chain 3:
```

```
## Warning: There were 6180 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: There were 6 transitions after warmup that exceeded the maximum treedepth. Increase max_tre
## https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

```
## Warning: There were 2 chains where the estimated Bayesian Fraction of Missing Information was low. S
## https://mc-stan.org/misc/warnings.html#bfmi-low
```

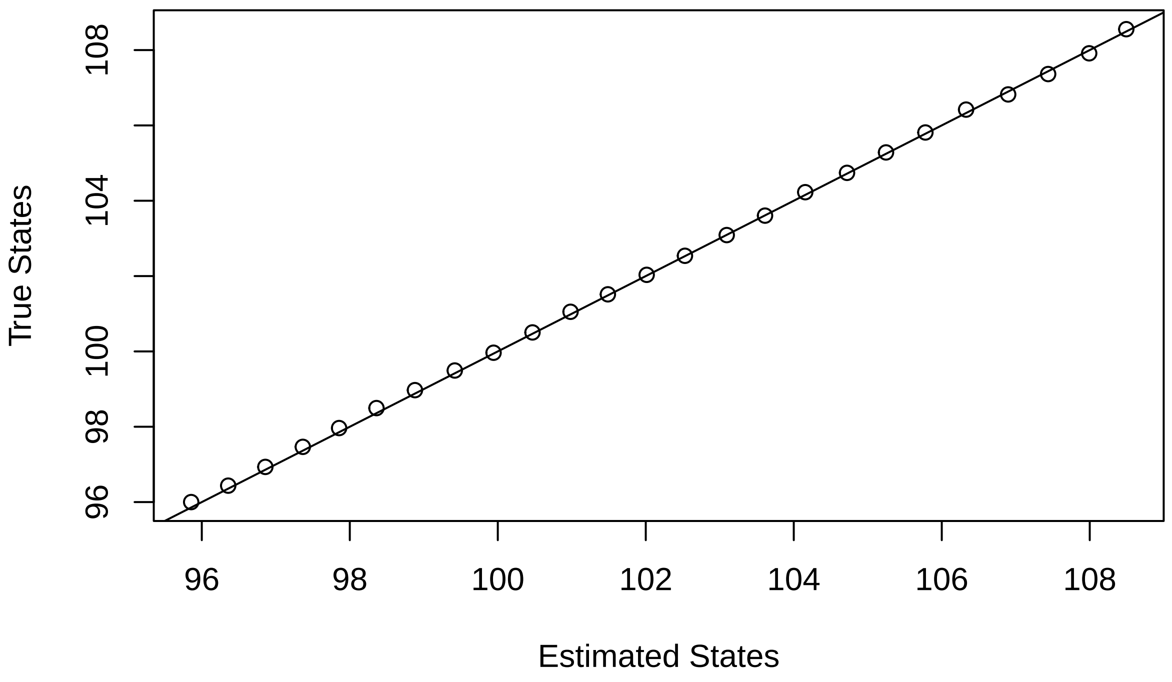
```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
## Warning: The largest R-hat is 1.13, indicating chains have not mixed.
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#r-hat
```

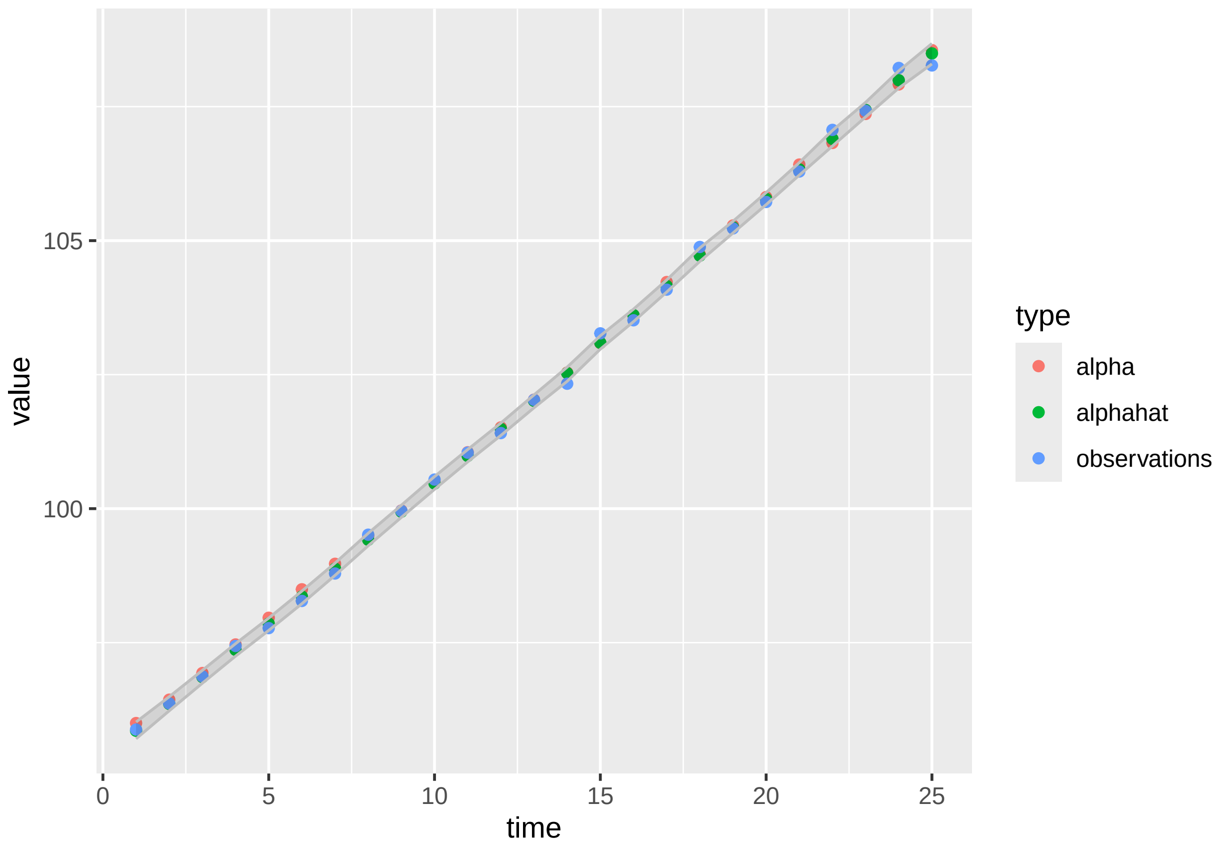
```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail quant
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess
```

```r
states <- colMeans(extract(stan_test, pars = c("z"))[[1]])

plot(x = states, y = alpha,
     xlab = "Estimated States",
     ylab = "True States")
abline(0,1)
```



```r
# Include a plot of time vs. estimated state, true state, and observations
time <- seq(from = 1, to = N, by = 1)

z_stan <- extract(stan_test, pars = c("z"))[[1]]
z_CIl <- apply(z_stan, 2, quantile, probs=0.025)
z_CIu <- apply(z_stan, 2, quantile, probs=0.975)

df1 <- tibble(time = time,
              alpha = alpha,
              alphahat = states,
              observations = y,
              z_CIl = z_CIl,
              z_CIu = z_CIu)


df_long <- df1 %>%
  pivot_longer(cols = c(alpha, alphahat,observations), names_to = "type", values_to = "value")

ggplot(df_long, aes(x = time, y = value, color = type)) +
    geom_point() +
    geom_ribbon(aes(ymin = z_CIl, ymax = z_CIu), fill = "black",color = "gray", alpha = .1)
```



```
# Estimated Slope:
mean(extract(stan_test, pars = c("v"))[[1]])
```

```
## [1] 0.1373057
```

```
# Estimated Observation Error:
mean(extract(stan_test, pars = c("sdo"))[[1]])
```

```
## [1] 0.1376522
```

```
# Estimated Process Error
mean(extract(stan_test, pars = c("sdp"))[[1]])
```

```
## [1] 0.04993984
```

```
calc_rMSE(alpha,states)
```

```
## [1] 0.07261506
```

```
calc_rMSE(alpha, y)
```

```
## [1] 0.1478524
```

Unfortunately, this just doesn't really seem to work. Doesn't really matter how I try to change beta. If I make it small enough to work with the interval I want, it ends up basically just being linear. I have another idea though.

## Linearly Increasing Slope

What if we just let the slope increase linearly with each step to capture some of that increasing decay rate behavior?

Here we'll simulate from the model

$$y_t = \alpha_t + \epsilon_t \tag{7}$$
$$\alpha_{t+1} = \alpha_t + \nu_t + \omega_t \tag{8}$$
$$\nu_{t+1} = \nu_t + \beta \tag{9}$$

where $\epsilon_t \sim N(0, .3)$, $\beta = .01$, $\nu_1 = .01$, $\omega_t \sim N(0, .1)$, and $\alpha_1 = 96$.

```
# Number of states
N <- 25

# Beta
beta <- .01

# Initialize nu vector
nu <- rep(NA, N)
nu[1] <- .01

# Initialize alpha vector
alpha <- rep(NA, N)
alpha[1] <- 96

# Initialize process errors
omega <- rnorm(n = N, mean = 0, sd = .1)

# Store true latent states
for(i in 2:(length(alpha))) {
    alpha[i] <- alpha[i-1] + nu[i-1] + omega[i]
    nu[i] <- nu[i-1] + beta
}

# Vector of observation errors
epsilon <- rnorm(n = N, mean = 0, sd = .3)

# Simulated observations
y <- alpha + epsilon
```

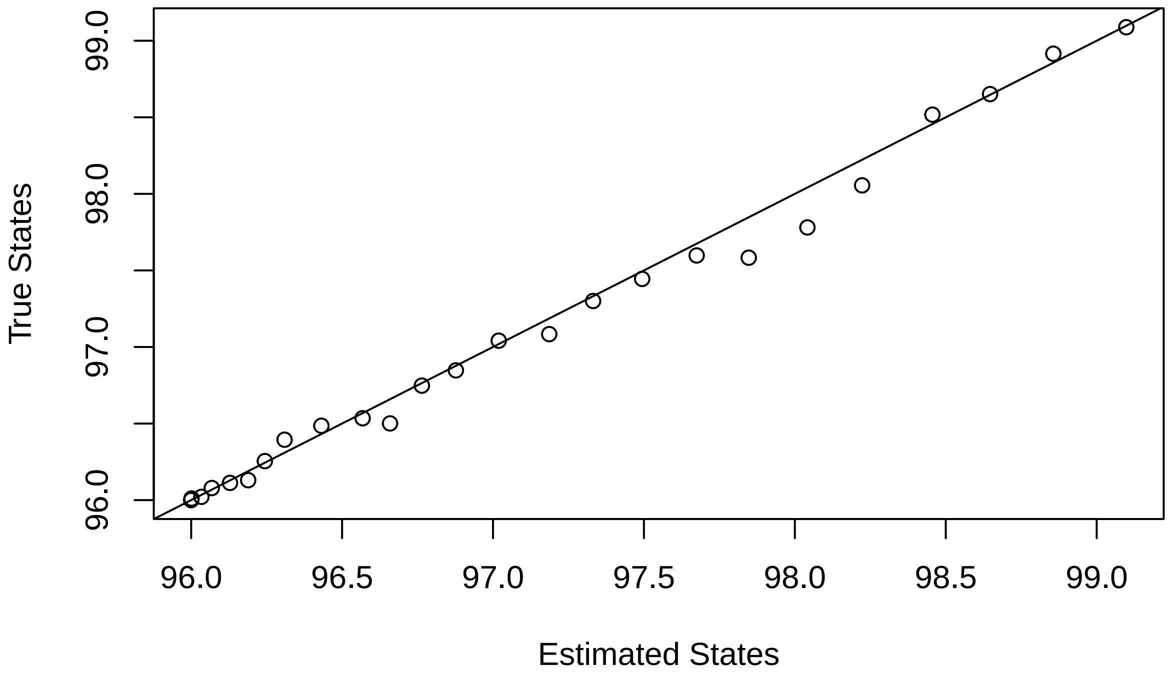
The Increasing_Slope_SSM.stan file fits the above model. Increasing_Slope_SSM_Parameterized.stan fits a parameterized version of the model to try to help with divergences.

```
data_stan <- list(TT = length(y), y = y, z0 = 96, v0 = .01, sdo0 = .3)

stan_test <- stan(file = "Increasing_Slope_SSM_Parameterized.stan",
                  data = data_stan,
                  chains = 3, iter = 30000,
                  control = list(adapt_delta = .99))
```

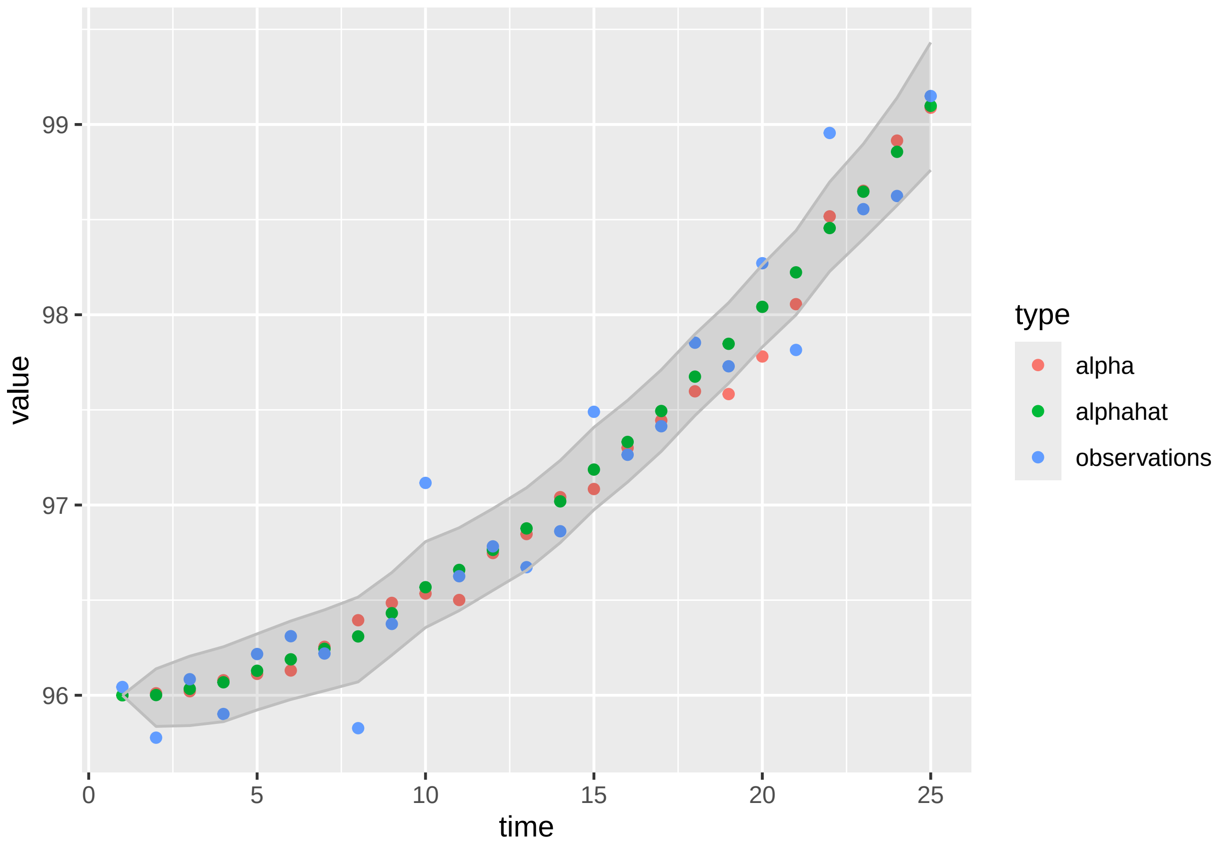
```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.5e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.35 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
```

```
## Chain 1:
## Chain 1: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 1: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 1: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 1: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 1: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 1: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 1: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 1: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 1: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 1: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 1: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 1: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 3.935 seconds (Warm-up)
## Chain 1:                3.506 seconds (Sampling)
## Chain 1:                7.441 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 2: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 2: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 2: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 2: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 2: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 2: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 2: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 2: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 2: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 2: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 2: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 3.904 seconds (Warm-up)
## Chain 2:                4.182 seconds (Sampling)
## Chain 2:                8.086 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 4e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 3: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 3: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 3: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 3: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 3: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 3: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 3: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 3: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 3: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 3: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 3: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 3.912 seconds (Warm-up)
## Chain 3:                3.577 seconds (Sampling)
## Chain 3:                7.489 seconds (Total)
## Chain 3:
```

```
states <- colMeans(extract(stan_test, pars = c("z"))[[1]])

plot(x = states, y = alpha,
     xlab = "Estimated States",
     ylab = "True States")
abline(0,1)
```



```
# Include a plot of time vs. estimated state, true state, and observations
time <- seq(from = 1, to = N, by = 1)

z_stan <- extract(stan_test, pars = c("z"))[[1]]
z_CIl <- apply(z_stan, 2, quantile, probs=0.025)
z_CIu <- apply(z_stan, 2, quantile, probs=0.975)

df1 <- tibble(time = time,
              alpha = alpha,
              alphahat = states,
              observations = y,
               z_CIl = z_CIl,
               z_CIu = z_CIu)


df_long <- df1 %>%
  pivot_longer(cols = c(alpha, alphahat,observations), names_to = "type", values_to = "value")

ggplot(df_long, aes(x = time, y = value, color = type)) +
    geom_point() +
    geom_ribbon(aes(ymin = z_CIl, ymax = z_CIu), fill = "black",color = "gray", alpha = .1)
```



```
# Estimated Observation Error:
mean(extract(stan_test, pars = c("sdo"))[[1]])
```

```
## [1] 0.2752279
```

```
# Estimated Process Error
mean(extract(stan_test, pars = c("sdp"))[[1]])
```

```
## [1] 0.07597588
```

```
calc_rMSE(alpha,states)
```

```
## [1] 0.09697471
```

```
calc_rMSE(alpha, y)
```

```
## [1] 0.2630121
```

This looks pretty good! It seems similar to the pattern observed in the laptime data and we are able to infer back to the true states quite well so long as the observation error is higher than the process error.

Now I'd like to implement some of the various model checks I've been reading about.

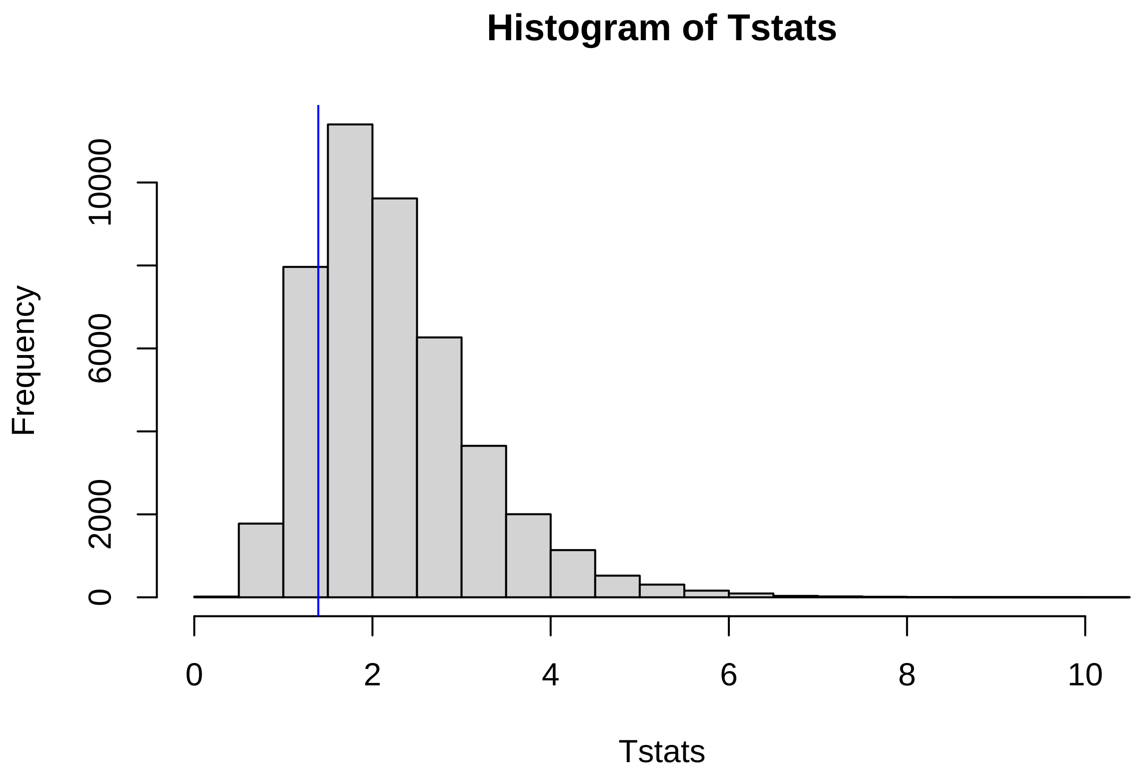
## Posterior Predictive Checks

For the previous model, we will do a posterior predictive check with the test statistic:

$$T(\mathbf{y},\theta) = \sum_{t=1}^{T}(y_t - \hat{y}_{t|1:T})^2$$

where $\hat{y}_{t|1:T}$) is the state estimate for the model at time t.

```r
Tstat <- function(y) {
    return(sum((y - df1$alphahat)^2))
}

posterior <- extract(stan_test)
yrep <- posterior$y_rep

Tstats <- rep(NA,dim(yrep)[1])
for(i in 1:dim(yrep)[1]) {
    Tstats[i] <- Tstat(yrep[i,])
}

hist(Tstats)
abline(v = Tstat(df1$observations), col = 'blue')
```



```r
Tobs <- Tstat(df1$observations)
count <- 0
```

```r
for(i in 1:length(Tstats)) {
    if(Tstats[i] > Tobs) {
        count <- count + 1
    }
}

pval <- count/length(Tstats)
pval
```

```
## [1] 0.8322444
```

## WAIC Calculation

```r
library(loo)

log_pd <- posterior$log_pd

waic(log_pd)
```

```
## Warning:
## 4 (16.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
##
## Computed from 45000 by 25 log-likelihood matrix.
##
##           Estimate  SE
## elpd_waic     -4.7 3.7
## p_waic         4.3 1.2
## waic           9.3 7.5
##
## 4 (16.0%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

## Forward Chaining Cross-Validation

The problem with Cross-Validation in this setting is that we have a pretty small sample size. Typically, a stint will be between 10 and 25 laps. For the 25 lap case above, the way I'd like to implement CV will be like so:

fold 1: training [1-20], test [21]

fold 2: training [1-21], test [22]

fold 3: training [1-22], test [23]

fold 4: training [1-23], test[24]

fold 5: training [1-24], test [25]

This seems like the best method out of those that I have come across for doing cross validation on a small sample size time series.

I could also potentially decrease the initial training size (maybe 15 instead of 20).

To estimate the one step ahead prediction, I have included code in the generated quantities block that will draw a predicted state and predicted observation for each posterior draw.

Lastly, the summary statistic we'll use is the Squared Prediction Error:

$$(y_{T+1} - \hat{y}_{T+1})^2$$

where $\hat{y}_{T+1}$ is the mean of the y_pred generated quantities.

```
# Number of folds
K <- 5

stan.models <- vector("list", length = K)
posteriors <- vector("list", length = K)
ypred_hat <- rep(NA, K)

for(i in 1:K) {

  data_stan <- list(TT = length(y[1:(i+N-K-1)]), y = y[1:(i+N-K-1)], z0 = 96, v0 = .01, sdo0 = .3)
  stan.models[[i]] <- stan(file = "Increasing_Slope_SSM_Parameterized.stan",
                           data = data_stan,
                           chains = 3, iter = 30000,
                           control = list(adapt_delta = .99))
  posteriors[[i]] <- extract(stan.models[[i]])
  ypred_hat[i]  <- mean(posteriors[[i]]$y_pred)
}
```

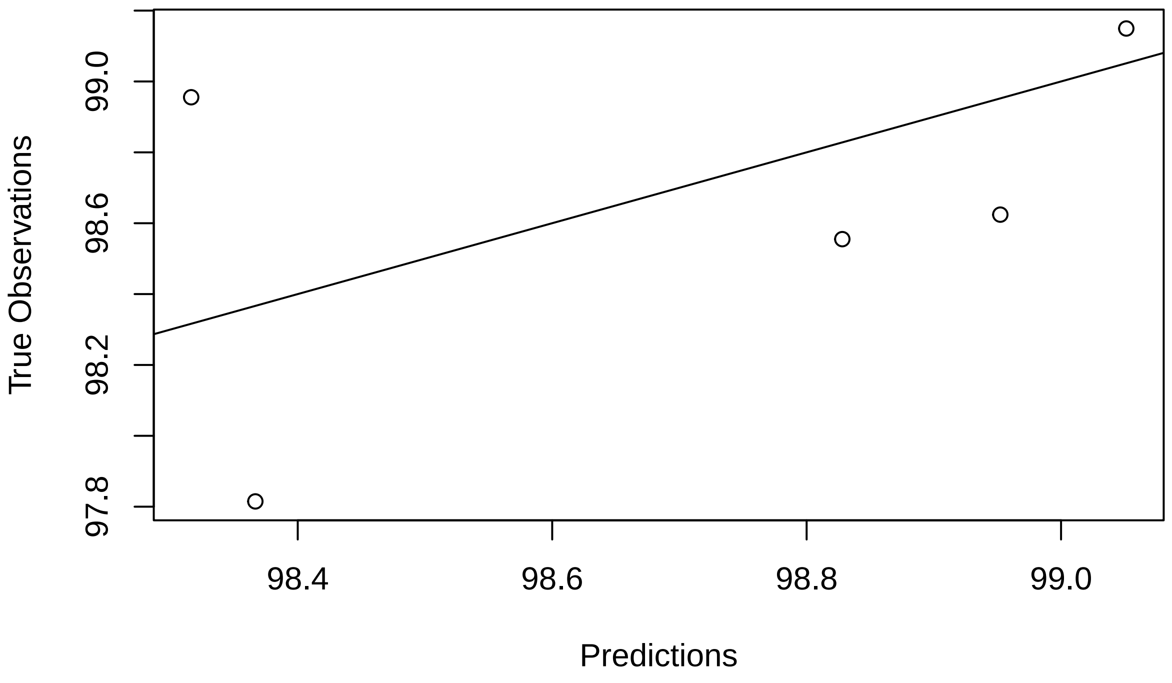
```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 1: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 1: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 1: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 1: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 1: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 1: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 1: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 1: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 1: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 1: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 1: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 3.47 seconds (Warm-up)
## Chain 1:                3.654 seconds (Sampling)
## Chain 1:                7.124 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 2: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 2: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 2: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 2: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 2: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 2: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 2: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 2: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 2: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 2: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 2: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 3.416 seconds (Warm-up)
## Chain 2:                3.164 seconds (Sampling)
## Chain 2:                6.58 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 3: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 3: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 3: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 3: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 3: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 3: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 3: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 3: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 3: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 3: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 3: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 3.404 seconds (Warm-up)
## Chain 3:                3.238 seconds (Sampling)
## Chain 3:                6.642 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 1: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 1: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 1: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 1: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 1: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 1: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 1: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 1: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 1: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 1: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 1: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 3.5 seconds (Warm-up)
## Chain 1:                3.051 seconds (Sampling)
## Chain 1:                6.551 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 2: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 2: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 2: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 2: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 2: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 2: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 2: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 2: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 2: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 2: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 2: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 3.623 seconds (Warm-up)
## Chain 2:                3.283 seconds (Sampling)
## Chain 2:                6.906 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 3: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 3: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 3: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 3: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 3: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 3: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 3: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 3: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 3: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 3: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 3: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 3.522 seconds (Warm-up)
## Chain 3:                4.007 seconds (Sampling)
## Chain 3:                7.529 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 1: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 1: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 1: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 1: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 1: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 1: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 1: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 1: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 1: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 1: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 1: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 3.509 seconds (Warm-up)
## Chain 1:                5.718 seconds (Sampling)
## Chain 1:                9.227 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 2: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 2: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 2: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 2: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 2: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 2: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 2: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 2: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 2: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 2: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 2: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 3.498 seconds (Warm-up)
## Chain 2:                2.877 seconds (Sampling)
## Chain 2:                6.375 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 3: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 3: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 3: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 3: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 3: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 3: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 3: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 3: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 3: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 3: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 3: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 3.489 seconds (Warm-up)
## Chain 3:                3.256 seconds (Sampling)
## Chain 3:                6.745 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 1: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 1: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 1: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 1: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 1: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 1: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 1: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 1: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 1: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 1: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 1: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 3.607 seconds (Warm-up)
## Chain 1:                6.576 seconds (Sampling)
## Chain 1:                10.183 seconds (Total)
## Chain 1:
##
```

```
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 2: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 2: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 2: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 2: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 2: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 2: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 2: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 2: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 2: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 2: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 2: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 3.694 seconds (Warm-up)
## Chain 2:                3.528 seconds (Sampling)
## Chain 2:                7.222 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 3: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 3: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 3: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 3: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 3: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 3: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 3: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 3: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 3: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 3: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 3: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 3.634 seconds (Warm-up)
## Chain 3:                3.456 seconds (Sampling)
## Chain 3:                7.09 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4e-06 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 1: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 1: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 1: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 1: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 1: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 1: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 1: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 1: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 1: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 1: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 1: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 3.84 seconds (Warm-up)
## Chain 1:                6.659 seconds (Sampling)
## Chain 1:                10.499 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 4e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.04 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 2: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 2: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 2: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 2: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 2: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 2: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 2: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 2: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 2: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 2: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 2: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 3.859 seconds (Warm-up)
## Chain 2:                3.491 seconds (Sampling)
## Chain 2:                7.35 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.02 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 3: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 3: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 3: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 3: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 3: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 3: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 3: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 3: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 3: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 3: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 3: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 3.897 seconds (Warm-up)
## Chain 3:                3.517 seconds (Sampling)
## Chain 3:                7.414 seconds (Total)
## Chain 3:
```

```
MSPE <- mean((y[(N-K+1):N] - ypred_hat)^2)
RMSPE <- sqrt(MSPE)

RMSPE
```

```
## [1] 0.4253955
```

```
plot(x = ypred_hat, y = y[(N-K+1):N],
     xlab = "Predictions",
     ylab = "True Observations")
abline(0,1)
```



## Linearly Increasing Slope 2.0

Lastly I'd just like to show what happens to the model when we make the observation error equal to the process error.

```r
# Number of states
N <- 50

# Beta
beta <- .01

# Initialize nu vector
nu <- rep(NA, N)
nu[1] <- .01

# Initialize alpha vector
alpha <- rep(NA, N)
alpha[1] <- 96

# Initialize process errors
omega <- rnorm(n = N, mean = 0, sd = .1)

# Store true latent states
for(i in 2:(length(alpha))) {
    alpha[i] <- alpha[i-1] + nu[i-1] + omega[i]
    nu[i] <- nu[i-1] + beta
}

# Vector of observation errors
epsilon <- rnorm(n = N, mean = 0, sd = .1)

# Simulated observations
y <- alpha + epsilon
```

The Increasing_Slope_SSM.stan file fits the above model.

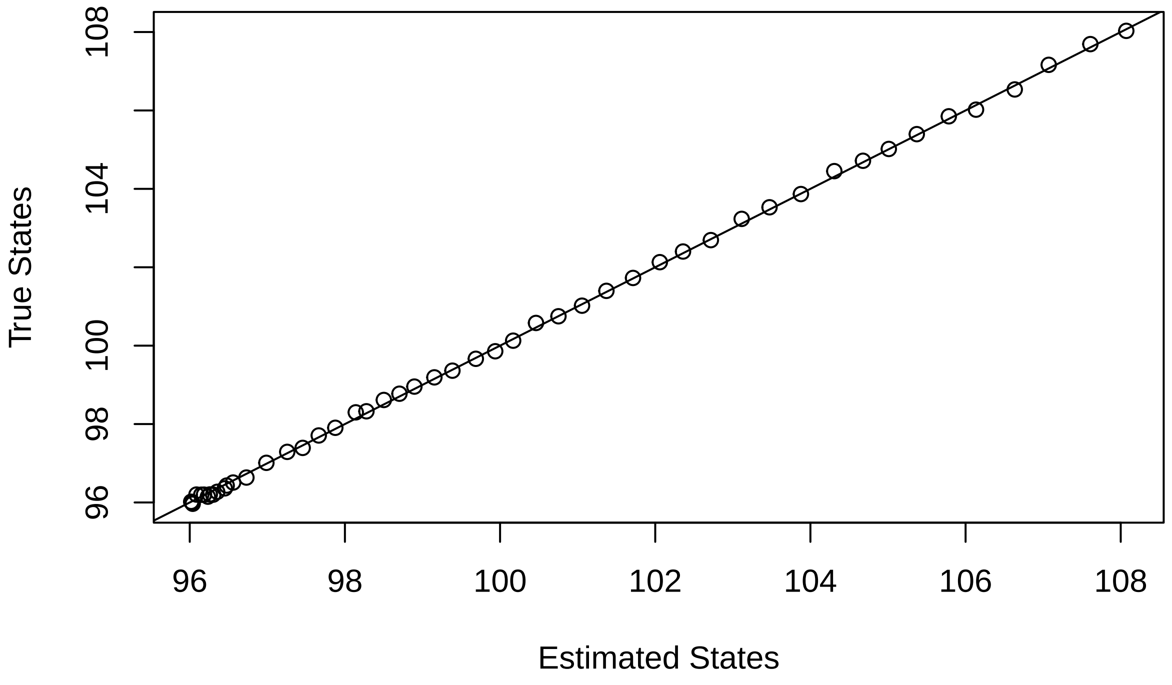
```r
data_stan <- list(TT = length(y), y = y, z0 = 96, v0 = .01, sdo0 = .1)

stan_test <- stan(file = "Increasing_Slope_SSM.stan",
                  data = data_stan,
                  chains = 3, iter = 30000,
                  control = list(adapt_delta = .95))
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 3.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.39 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 1: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 1: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 1: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 1: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 1: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 1: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 1: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 1: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 1: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 1: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 1: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 2.731 seconds (Warm-up)
## Chain 1:                1.897 seconds (Sampling)
## Chain 1:                4.628 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3e-06 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 2: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 2: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 2: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 2: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 2: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 2: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 2: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 2: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 2: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 2: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 2: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 2:
## Chain 2:  Elapsed Time: 2.609 seconds (Warm-up)
## Chain 2:                1.852 seconds (Sampling)
## Chain 2:                4.461 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3e-06 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.03 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:     1 / 30000 [  0%]  (Warmup)
## Chain 3: Iteration:  3000 / 30000 [ 10%]  (Warmup)
## Chain 3: Iteration:  6000 / 30000 [ 20%]  (Warmup)
## Chain 3: Iteration:  9000 / 30000 [ 30%]  (Warmup)
## Chain 3: Iteration: 12000 / 30000 [ 40%]  (Warmup)
## Chain 3: Iteration: 15000 / 30000 [ 50%]  (Warmup)
## Chain 3: Iteration: 15001 / 30000 [ 50%]  (Sampling)
## Chain 3: Iteration: 18000 / 30000 [ 60%]  (Sampling)
## Chain 3: Iteration: 21000 / 30000 [ 70%]  (Sampling)
## Chain 3: Iteration: 24000 / 30000 [ 80%]  (Sampling)
## Chain 3: Iteration: 27000 / 30000 [ 90%]  (Sampling)
## Chain 3: Iteration: 30000 / 30000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 2.73 seconds (Warm-up)
## Chain 3:                2.128 seconds (Sampling)
## Chain 3:                4.858 seconds (Total)
## Chain 3:
```

```
## Warning: There were 16 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

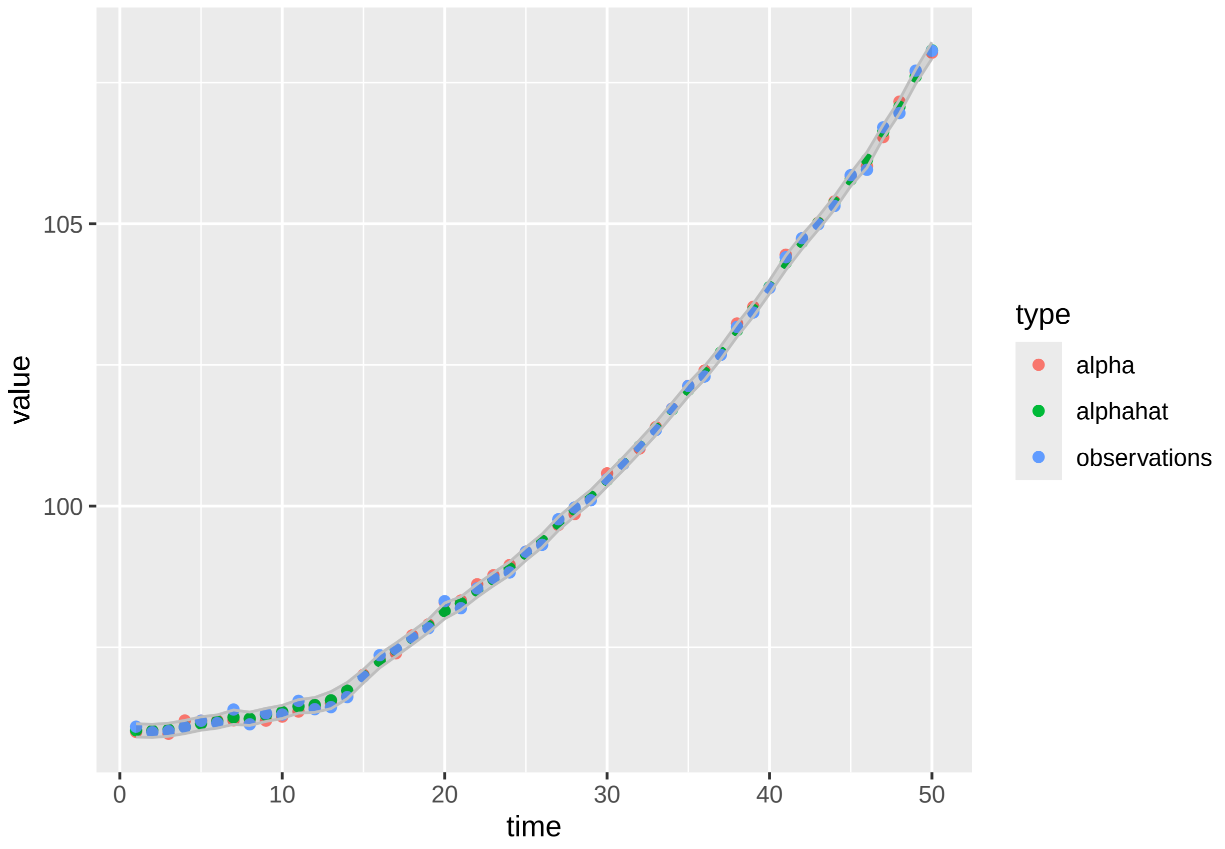
```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
states <- colMeans(extract(stan_test, pars = c("z"))[[1]])

plot(x = states, y = alpha,
     xlab = "Estimated States",
     ylab = "True States")
abline(0,1)
```



```
# Include a plot of time vs. estimated state, true state, and observations
time <- seq(from = 1, to = N, by = 1)

z_stan <- extract(stan_test, pars = c("z"))[[1]]
z_CIl <- apply(z_stan, 2, quantile, probs=0.025)
z_CIu <- apply(z_stan, 2, quantile, probs=0.975)

df1 <- tibble(time = time,
              alpha = alpha,
              alphahat = states,
              observations = y,
              z_CIl = z_CIl,
              z_CIu = z_CIu)
```

```r
df_long <- df1 %>%
  pivot_longer(cols = c(alpha, alphahat,observations), names_to = "type", values_to = "value")

ggplot(df_long, aes(x = time, y = value, color = type)) +
    geom_point() +
    geom_ribbon(aes(ymin = z_CIl, ymax = z_CIu), fill = "black",color = "gray", alpha = .1)
```



```r
# Estimated Slope:
mean(extract(stan_test, pars = c("v"))[[1]])
```

```
## [1] 0.2458543
```

```r
# Estimated Observation Error:
mean(extract(stan_test, pars = c("sdo"))[[1]])
```

```
## [1] 0.09625382
```

```r
# Estimated Process Error
mean(extract(stan_test, pars = c("sdp"))[[1]])
```

```
## [1] 0.08112045
```

```r
calc_rMSE(alpha,states)
```

```
## [1] 0.07038458
```

```r
calc_rMSE(alpha, y)
```

```
## [1] 0.08225244
```

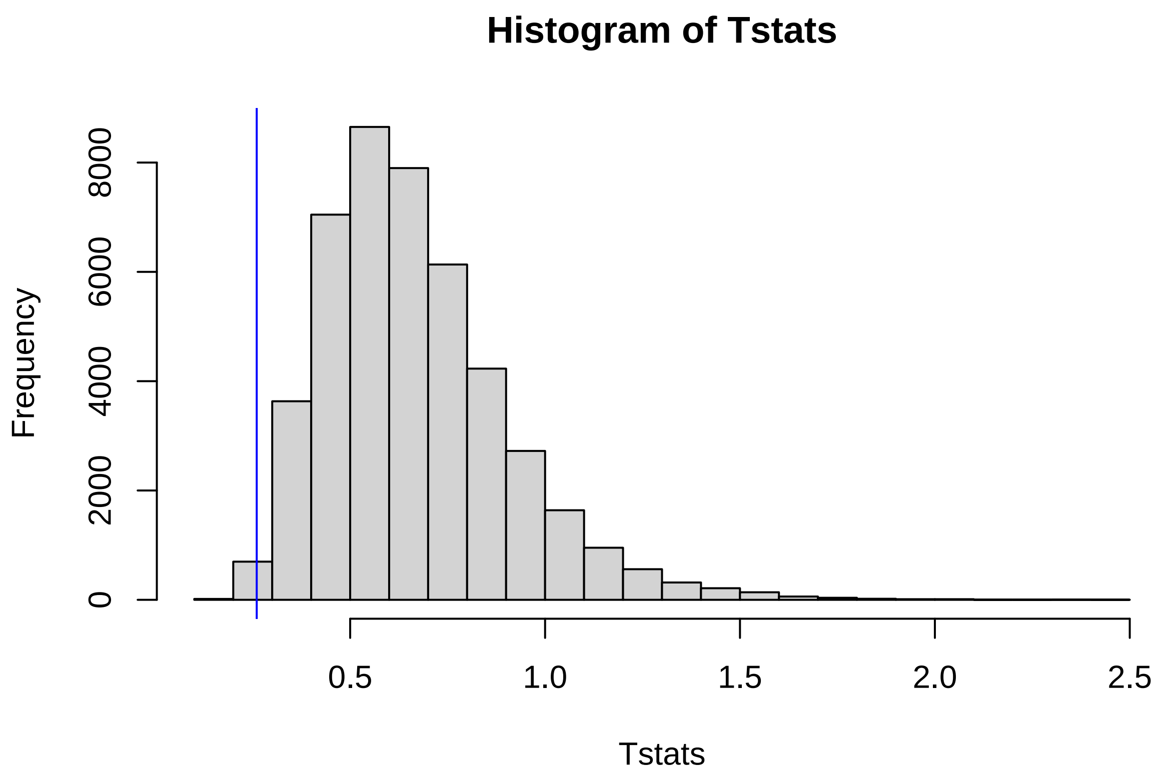
## Posterior Predictive Checks

For the previous model, we will do a posterior predictive check with the test statistic:

$$T(\mathbf{y}, \theta) = \sum_{t=1}^{T} (y_t - \hat{y}_{t|1:T})^2$$

where $\hat{y}_{t|1:T})^2$ is the state estimate for the model at time t.

```r
Tstat <- function(y) {
    return(sum((y - df1$alphahat)^2))
}

posterior <- extract(stan_test)
yrep <- posterior$y_rep

Tstats <- rep(NA,dim(yrep)[1])
for(i in 1:dim(yrep)[1]) {
    Tstats[i] <- Tstat(yrep[i,])
}

hist(Tstats)
abline(v = Tstat(df1$observations), col = 'blue')
```



```r
Tobs <- Tstat(df1$observations)
count <- 0
for(i in 1:length(Tstats)) {
    if(Tstats[i] > Tobs) {
        count <- count + 1
    }
}

pval <- count/length(Tstats)
pval
```

```
## [1] 0.9945778
```

It seems a bit concerning to me that I've specified the data generating process exactly in stan with very informative priors and yet our posterior predictive p-val is still very close to 1.