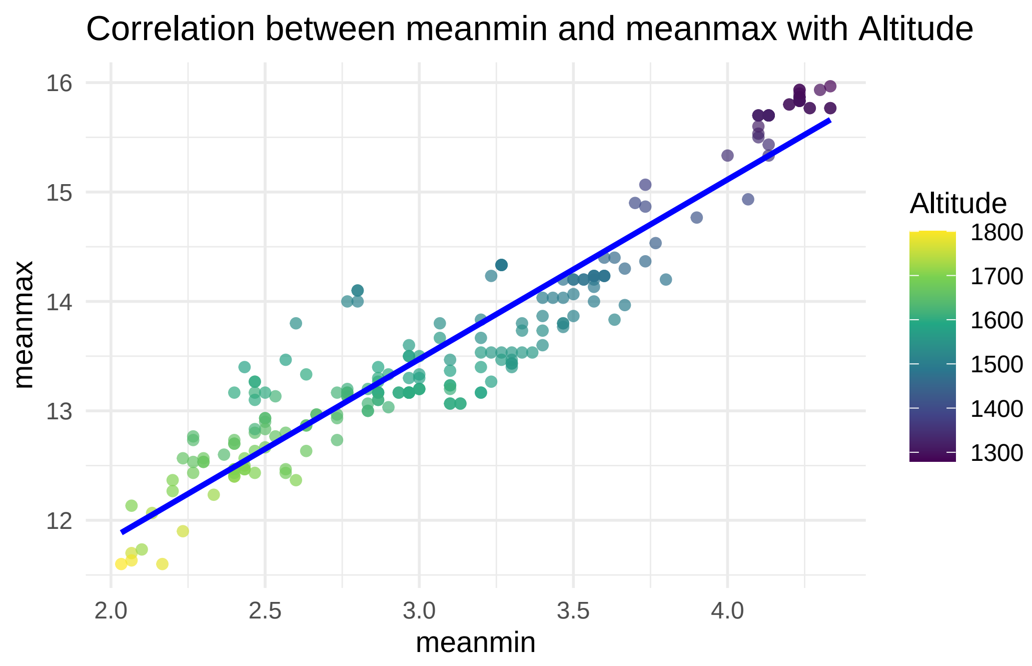
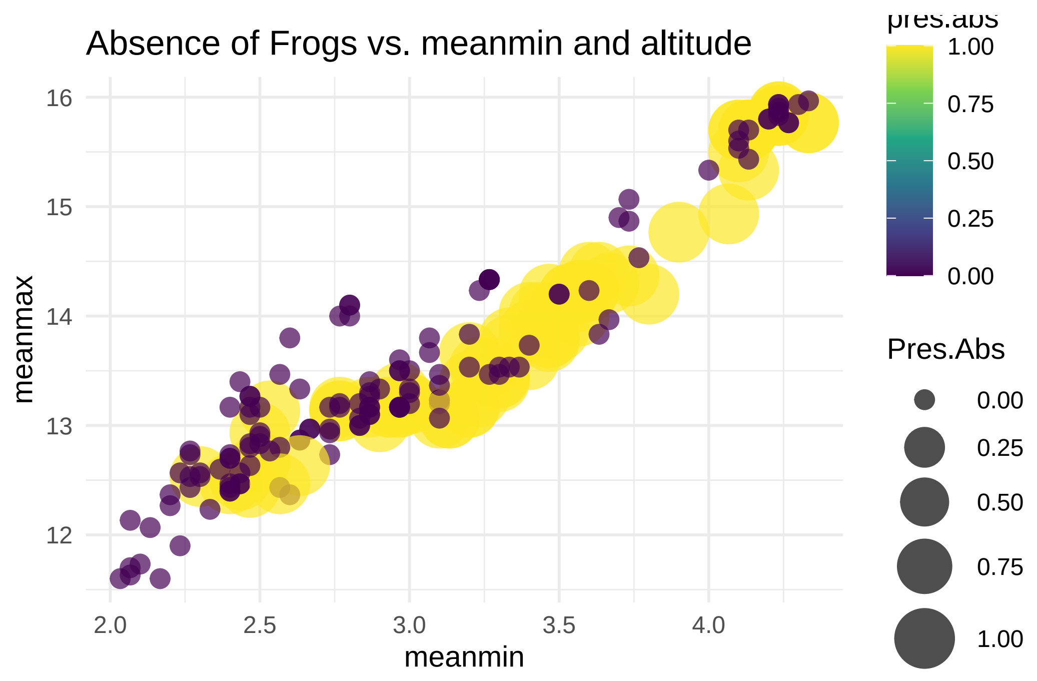
#Analysis of the data set

```r
ggplot(data, aes(x = meanmin, y = meanmax, color = altitude)) +
  geom_point(alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add linear regression line
  scale_color_viridis_c() +  # You can change the color scale if needed
  labs(title = "Correlation between meanmin and meanmax with Altitude",
       x = "meanmin",
       y = "meanmax",
       color = "Altitude") +
  theme_minimal()
```

```
`geom_smooth()` using formula = 'y ~ x'
```

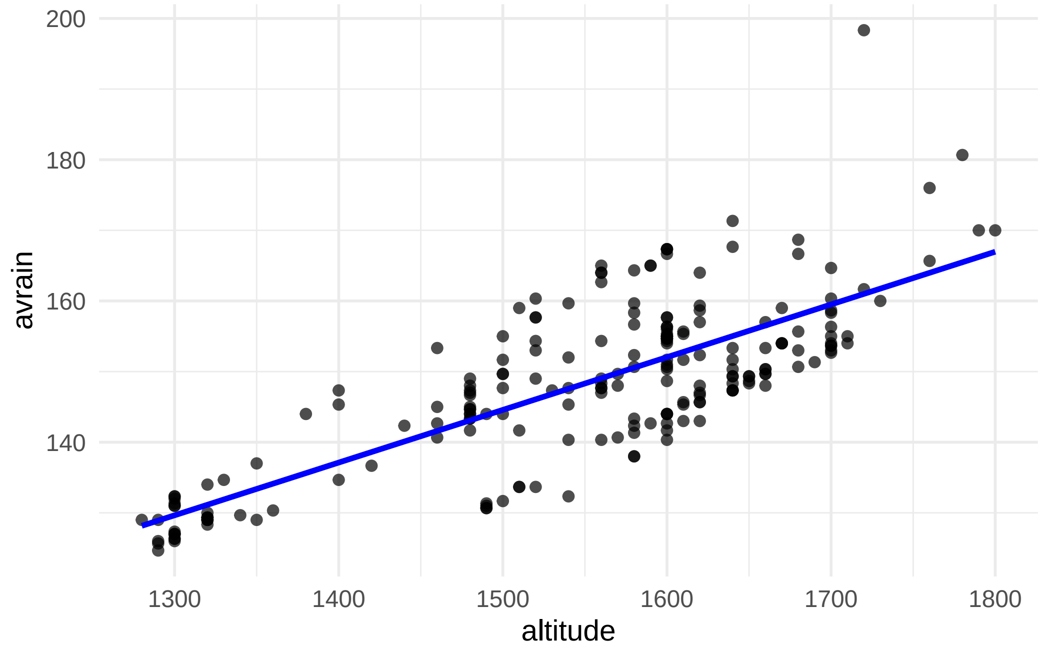


```r
# Plot the absence of frogs as a function of meanmin and altitude
ggplot(data, aes(x = meanmin, y = meanmax, color = pres.abs, size = pres.abs)) +
  geom_point(alpha = 0.7) +
  scale_color_viridis_c() +  # You can change the color scale if needed
  scale_size_continuous(range = c(3, 10)) +
  labs(title = "Absence of Frogs vs. meanmin and altitude",
       x = "meanmin",
       y = "meanmax",
       size = "Pres.Abs") +
  theme_minimal()
```

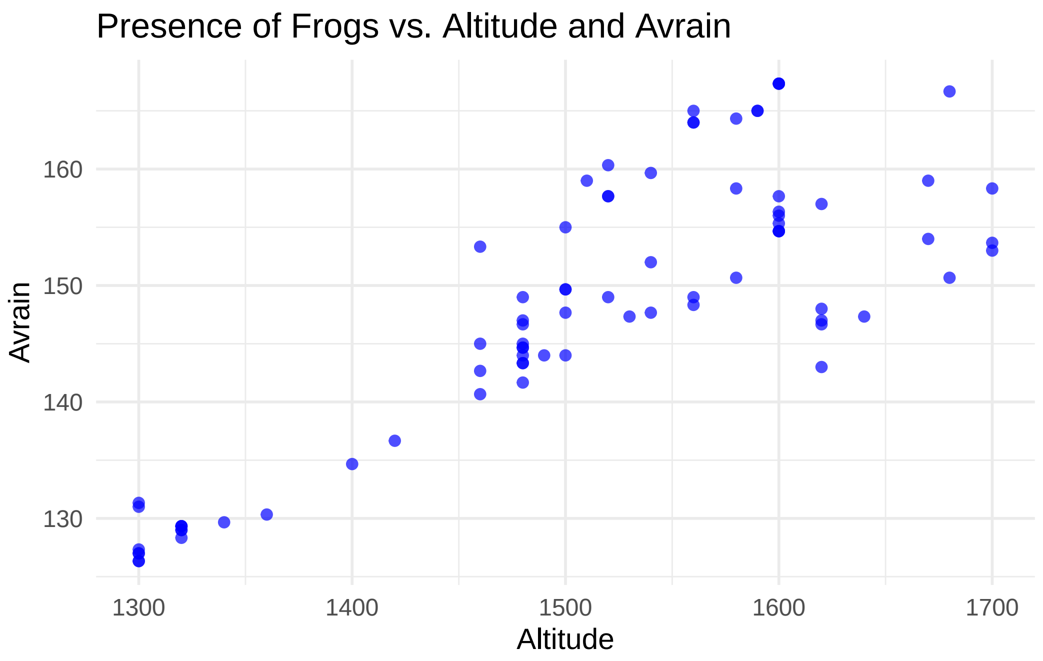


```r
ggplot(data, aes(x = altitude, y = avrain)) +
  geom_point(alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +  # Add linear regression line
  scale_color_viridis_c() +  # You can change the color scale if needed

  theme_minimal()
```

```
`geom_smooth()` using formula = 'y ~ x'
```



```
# Filter the data to include only the presence of frogs
presence_data <- data[data$pres.abs == 1, ]

# Plot the presence of frogs with respect to altitude and avrain
ggplot(presence_data, aes(x = altitude, y = avrain)) +
  geom_point(alpha = 0.7, color = "blue") +
  labs(title = "Presence of Frogs vs. Altitude and Avrain",
       x = "Altitude",
       y = "Avrain") +
  theme_minimal()
```

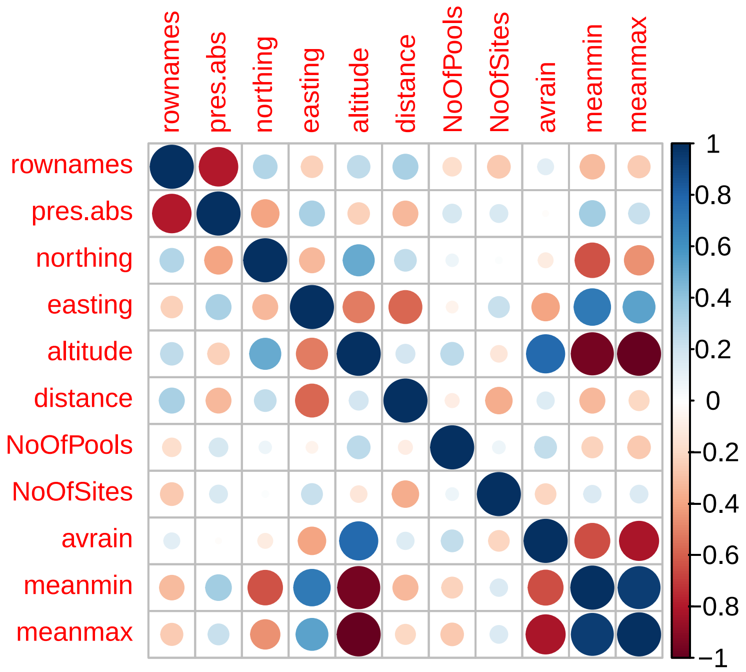


```
# Calculate correlations
correlation_matrix <- cor(data)
print(correlation_matrix)
```

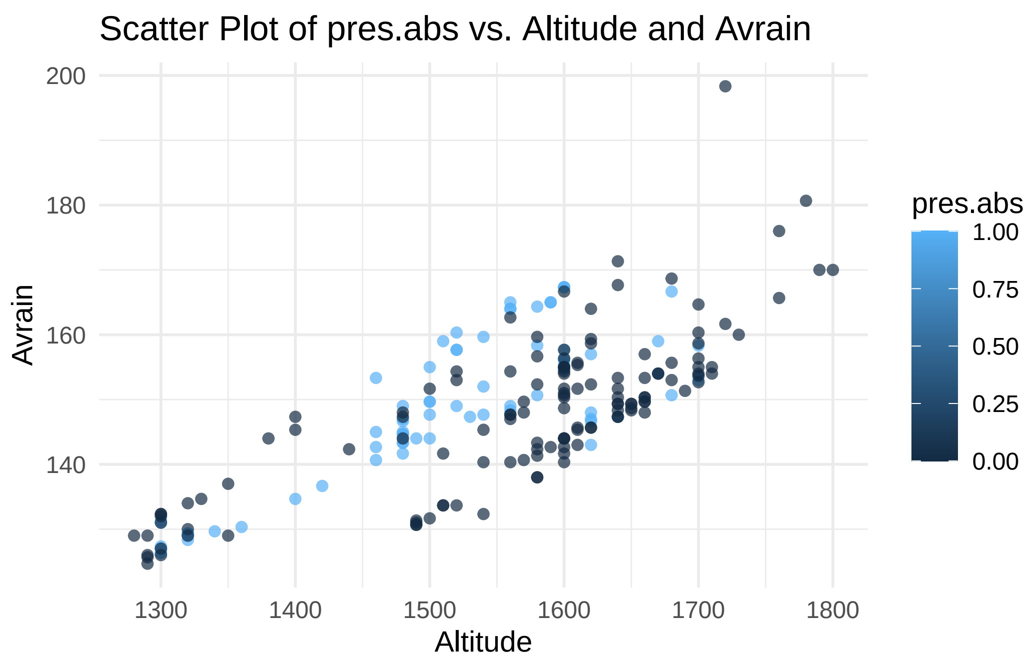
```
            rownames    pres.abs    northing     easting   altitude   distance
rownames   1.0000000 -0.79379062  0.29039960 -0.23733975  0.2594501  0.3251945
pres.abs  -0.7937906  1.00000000 -0.39875706  0.32493331 -0.2384100 -0.3217480
northing   0.2903996 -0.39875706  1.00000000 -0.32390138  0.5098837  0.2416126
easting   -0.2373398  0.32493331 -0.32390138  1.00000000 -0.5147038 -0.5753061
altitude   0.2594501 -0.23841002  0.50988367 -0.51470383  1.0000000  0.1820645
distance   0.3251945 -0.32174803  0.24161260 -0.57530607  0.1820645  1.0000000
NoOfPools -0.1727391  0.17562416  0.07394787 -0.06503599  0.2667913 -0.0959017
NoOfSites -0.2665277  0.16070989  0.01818349  0.22410207 -0.1368677 -0.3656936
avrain     0.1257059 -0.01358179 -0.10914493 -0.39538028  0.7780271  0.1484831
meanmin   -0.3117578  0.34316473 -0.63320255  0.70890060 -0.9536610 -0.3263706
meanmax   -0.2505741  0.22265134 -0.45335004  0.53773711 -0.9965570 -0.2045681
            NoOfPools   NoOfSites      avrain     meanmin    meanmax
```

```
rownames   -0.17273915 -0.26652773  0.12570585 -0.3117578 -0.2505741
pres.abs    0.17562416  0.16070989 -0.01358179  0.3431647  0.2226513
northing    0.07394787  0.01818349 -0.10914493 -0.6332025 -0.4533500
easting    -0.06503599  0.22410207 -0.39538028  0.7089006  0.5377371
altitude    0.26679134 -0.13686765  0.77802713 -0.9536610 -0.9965570
distance   -0.09590170 -0.36569361  0.14848311 -0.3263706 -0.2045681
NoOfPools   1.00000000  0.07803560  0.24377919 -0.2237348 -0.2633172
NoOfSites   0.07803560  1.00000000 -0.21811450  0.1510355  0.1576344
avrain      0.24377919 -0.21811450  1.00000000 -0.6492240 -0.8186997
meanmin    -0.22373480  0.15103548 -0.64922402  1.0000000  0.9462741
meanmax    -0.26331723  0.15763440 -0.81869972  0.9462741  1.0000000
```

```r
# Create a prettier plot of the correlation matrix
corrplot(correlation_matrix, method = "circle", type = "full", tl.cex = 0.8)
```



```r
# Plot the correlations
library(ggplot2)

ggplot(data, aes(x = altitude, y = avrain, color = pres.abs)) +
  geom_point(alpha = 0.7) +
  labs(title = "Scatter Plot of pres.abs vs. Altitude and Avrain",
       x = "Altitude",
       y = "Avrain",
       color = "pres.abs") +
  theme_minimal()
```



1 Forming bayesian regression models using Stan

```
priors1 <- c(
  prior(normal(0,100), coef = "avrain"),
  prior(normal(0,100), coef = "NoOfPools"),
  prior(normal(15,10), coef = "meanmax"),
  prior(normal(4,3), coef = "meanmin"),
  prior(normal(1500,300), coef = "altitude")
)


fit1 <- brms::brm(
  # This specifies the formula
  pres.abs ~ altitude + avrain + NoOfPools + meanmin + meanmax,
  # This specifies the dataset
  data = data,
  # This specifies the observation model family
  family = "bernoulli",
  # This passes the priors specified above to brms
  prior = priors1,
  # This causes brms to cache the results
  file = "~/Documents/R/Project/fit1"
)

priors2 <- c(
  prior(normal(0,100), coef = "NoOfPools"),
  prior(normal(4,3), coef = "meanmin"),
```

```r
  prior(normal(1500,300), coef = "altitude")
)


fit2 <- brms::brm(
  # This specifies the formula
  pres.abs ~ NoOfPools + meanmin + altitude,
  # This specifies the dataset
  data = data,
  # This specifies the observation model family
  family = bernoulli(link="logit"),
  # This passes the priors specified above to brms
  prior = priors2,
  # This causes brms to cache the results
  file = "~/Documents/R/Project/fit2"
)

priors3 <- c(
  prior(normal(4,3), coef = "meanmin"),
  prior(normal(1500,300), coef = "altitude")
)

fit3 <- brms::brm(
  # This specifies the formula
  pres.abs ~  meanmin + altitude^2,
  # This specifies the dataset
  data = data,
  # This specifies the observation model family
  family = bernoulli(link="logit"),
  # This passes the priors specified above to brms
  prior = priors3,
  # This causes brms to cache the results
  file = "~/Documents/R/Project/fit3"
)
```

2 Posterior predictive checking and model analysis

```r
pp_check_fit1 <- brms::pp_check(fit1, "stat")
```

```
Using all posterior draws for ppc type 'stat' by default.
```

```r
traceplot_fit1 <- stanplot(fit1, type = "trace", prob= 0.95)
```

```
Warning: Method 'stanplot' is deprecated. Please use 'mcmc_plot' instead.
```

```
Warning: The following arguments were unrecognized and ignored: prob
```

```
No divergences to plot.
```

```r
pp_check_fit2 <- brms::pp_check(fit2,"stat")
```

```
Using all posterior draws for ppc type 'stat' by default.
```

```r
traceplot_fit2 <- stanplot(fit2, type = "trace")
```

```
Warning: Method 'stanplot' is deprecated. Please use 'mcmc_plot' instead.
```

```
No divergences to plot.
```

```r
pp_check_fit3 <- brms::pp_check(fit3,"stat")
```

```
Using all posterior draws for ppc type 'stat' by default.
```

```r
traceplot_fit3 <- stanplot(fit3, type = "trace")
```

```
Warning: Method 'stanplot' is deprecated. Please use 'mcmc_plot' instead.
```

```
No divergences to plot.
```

```r
s1 <- summary(fit1)
s2 <- summary(fit2)
s3 <- summary(fit3)

print(s1)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: pres.abs ~ altitude + avrain + NoOfPools + meanmin + meanmax
   Data: data (Number of observations: 212)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   -78.25    118.16  -317.67   143.88 1.00     1103     1168
altitude      0.03      0.03    -0.04     0.10 1.00     1135     1189
```

```
avrain         0.05      0.05    -0.06     0.16 1.00     1263     2004
NoOfPools      0.03      0.01     0.02     0.05 1.00     2634     2654
meanmin        6.14      1.36     3.52     8.92 1.00     2295     2448
meanmax        0.81      4.29    -7.26     9.41 1.00     1103     1328

Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
print(s2)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: pres.abs ~ NoOfPools + meanmin + altitude
   Data: data (Number of observations: 212)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   -60.90     11.79   -84.75   -39.50 1.00     1404     1870
NoOfPools     0.03      0.01     0.01     0.05 1.00     2383     2031
meanmin       6.72      1.16     4.55     9.08 1.00     1381     1881
altitude      0.02      0.01     0.01     0.04 1.00     1463     1894

Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
print(s3)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: pres.abs ~ meanmin + altitude^2
   Data: data (Number of observations: 212)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   -60.02     11.51   -82.70   -38.71 1.00     1171     1309
meanmin       6.38      1.12     4.30     8.61 1.00     1185     1419
altitude      0.03      0.01     0.02     0.04 1.00     1191     1371

Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
r1 <- rhat(fit1)
r2 <- rhat(fit2)
r3 <- rhat(fit3)
print(r1)
```

```
b_Intercept  b_altitude    b_avrain b_NoOfPools   b_meanmin   b_meanmax 
   1.001682    1.001412    1.002236    1.000579    1.001079    1.001654 
     lprior        lp__ 
   1.000519    1.002178 
```

```r
print(r2)
```

```
b_Intercept b_NoOfPools   b_meanmin  b_altitude      lprior        lp__ 
   1.004646    1.000935    1.004281    1.004620    1.004556    1.001010 
```

```r
print(r3)
```

```
b_Intercept   b_meanmin  b_altitude      lprior        lp__ 
   1.002800    1.002660    1.002540    1.001230    1.000921 
```

```r
mcse(fit1)
```

```
    Parameter         MCSE
1 b_Intercept 3.5691039928
2  b_altitude 0.0010223064
3    b_avrain 0.0015271118
4 b_NoOfPools 0.0001690033
5   b_meanmin 0.0285918130
6   b_meanmax 0.1295099968
```

```r
mcse(fit2)
```

```
    Parameter         MCSE
1 b_Intercept 0.3127218690
2 b_NoOfPools 0.0001736244
3   b_meanmin 0.0311892344
4  b_altitude 0.0001374450
```

```r
mcse(fit3)
```

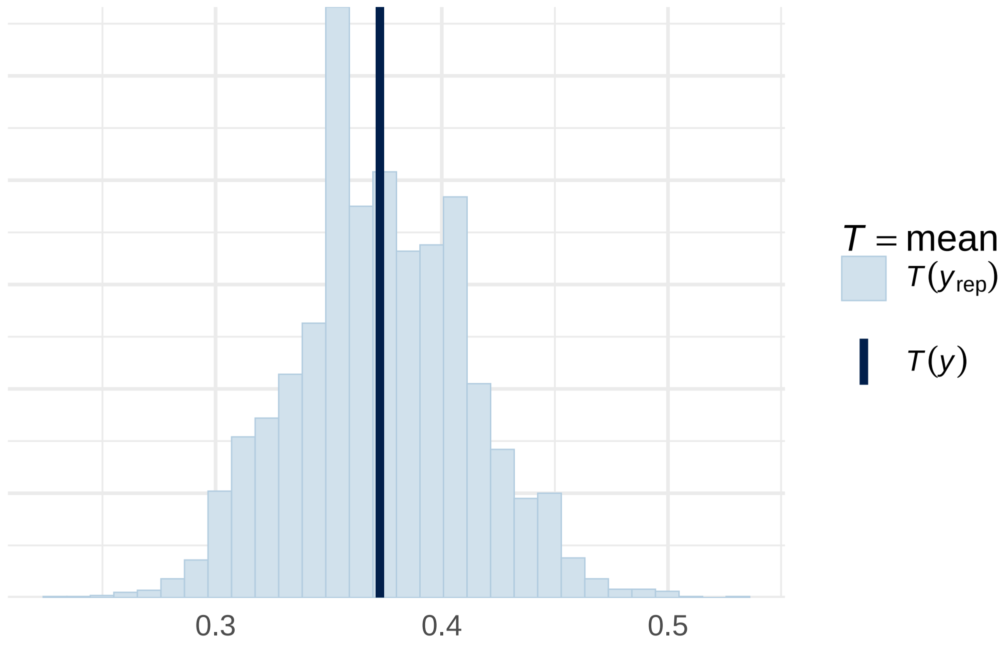
```
    Parameter         MCSE
1 b_Intercept 0.3386960020
2   b_meanmin 0.0327356654
3  b_altitude 0.0001516418
```

```r
loo_fit1 <- loo(fit1)
loo_fit2 <- loo(fit2)
loo_fit3 <- loo(fit3)

loo_compare(loo_fit1,loo_fit2,loo_fit3)
```

```
     elpd_diff se_diff
fit2  0.0       0.0   
fit1 -1.3       1.4   
fit3 -6.5       3.9   
```

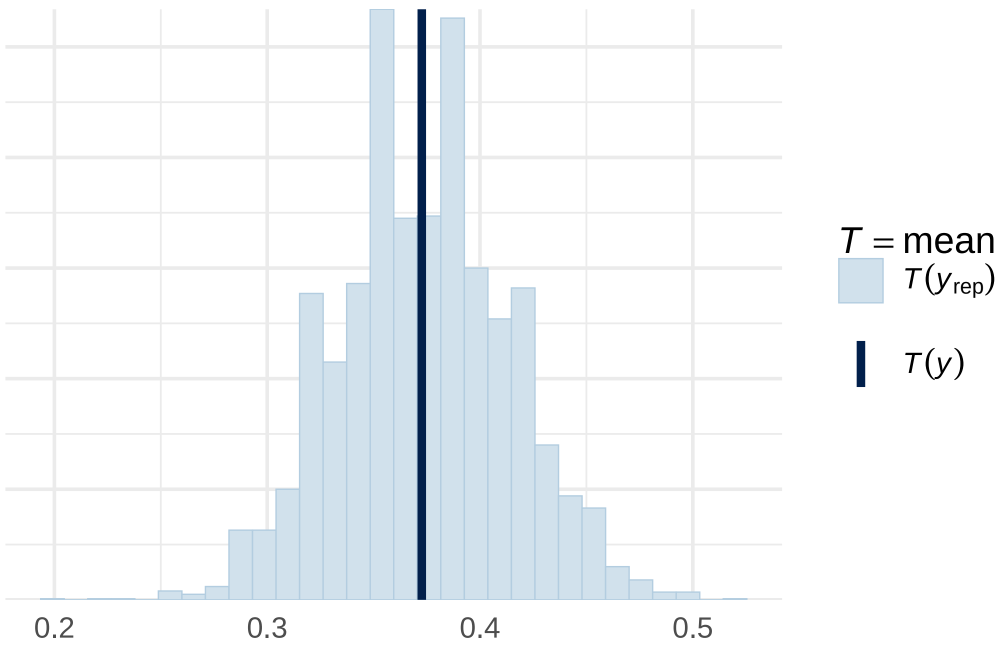
```r
print(pp_check_fit1)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



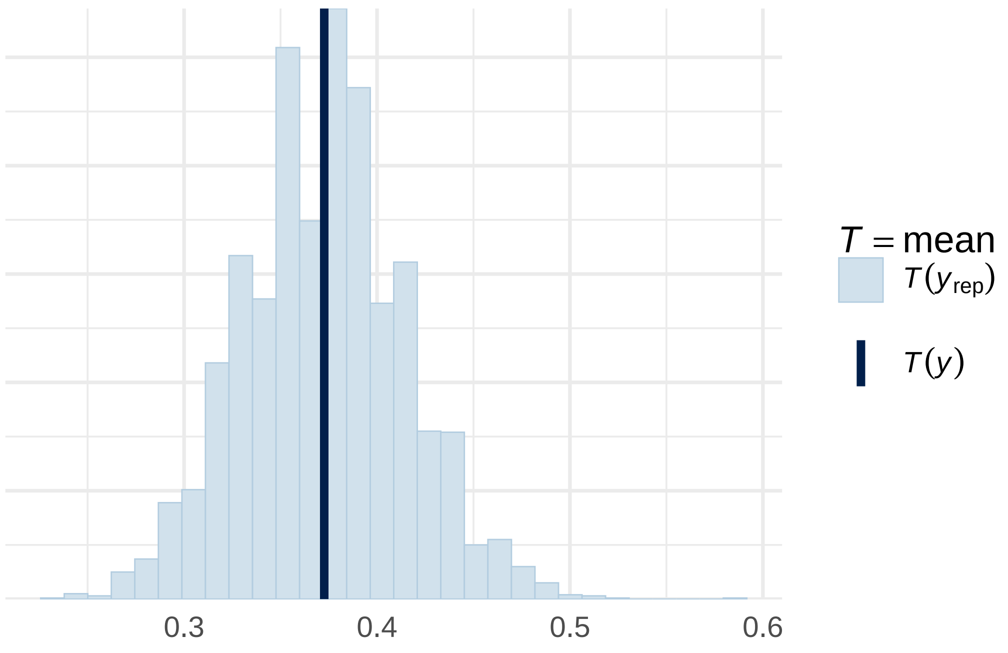
```r
print(pp_check_fit2)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
print(pp_check_fit3)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



#Prior Sensitivity analysis

```r
new_priors1_new <- c(
  prior(normal(0,100), coef = "avrain"),
  prior(normal(0,100), coef = "NoOfPools"),
  prior(normal(20,20), coef = "meanmax"),
  prior(normal(0,20), coef = "meanmin"),
  prior(normal(1000,1000), coef = "altitude")
)


fit_1_new_priors <- brms::brm(
  # This specifies the formula
  pres.abs ~ altitude + avrain + NoOfPools + meanmin + meanmax,
  # This specifies the dataset
  data = data,
  # This specifies the observation model family
  family = "bernoulli",
  # This passes the new_priors specified above to brms
  prior = new_priors1_new,
  # This causes brms to cache the results
  file = "~/Documents/R/Project/fit_1_new_priors"
)

new_priors2_new <- c(
  prior(normal(0,100), coef = "NoOfPools"),
  prior(normal(0,20), coef = "meanmin"),
  prior(normal(1000,1000), coef = "altitude")
)


fit_2_new_priors <- brms::brm(
  # This specifies the formula
  pres.abs ~ NoOfPools + meanmin + altitude,
  # This specifies the dataset
  data = data,
  # This specifies the observation model family
  family = bernoulli(link="logit"),
  # This passes the new_priors specified above to brms
  prior = new_priors2_new,
  # This causes brms to cache the results
  file = "~/Documents/R/Project/fit_2_new_priors_new_prior"
)

new_priors3_new <- c(
  prior(normal(0,20), coef = "meanmin"),
  prior(normal(1000,1000), coef = "altitude")
)

fit_3_new_priors <- brms::brm(
  # This specifies the formula
```

```r
    pres.abs ~  meanmin + altitude^2,
    # This specifies the dataset
    data = data,
    # This specifies the observation model family
    family = bernoulli(link="logit"),
    # This passes the new_priors specified above to brms
    prior = new_priors3_new,
    # This causes brms to cache the results
    file = "~/Documents/R/Project/fit_3_new_priors"
  )

  pp_check_fit_1_new_priors <- brms::pp_check(fit_1_new_priors, "stat")
```

```
Using all posterior draws for ppc type 'stat' by default.
```

```r
traceplot_fit_1_new_priors <- stanplot(fit_1_new_priors, type = "trace", prob= 0.95)
```

```
Warning: Method 'stanplot' is deprecated. Please use 'mcmc_plot' instead.

Warning: The following arguments were unrecognized and ignored: prob

No divergences to plot.
```

```r
pp_check_fit_2_new_priors <- brms::pp_check(fit_2_new_priors, "stat")
```

```
Using all posterior draws for ppc type 'stat' by default.
```

```r
traceplot_fit_2_new_priors <- stanplot(fit_2_new_priors, type = "trace")
```

```
Warning: Method 'stanplot' is deprecated. Please use 'mcmc_plot' instead.

No divergences to plot.
```

```r
pp_check_fit_3_new_priors <- brms::pp_check(fit_3_new_priors, "stat")
```

```
Using all posterior draws for ppc type 'stat' by default.
```

```r
traceplot_fit_3_new_priors <- stanplot(fit_3_new_priors, type = "trace")
```

```
Warning: Method 'stanplot' is deprecated. Please use 'mcmc_plot' instead.

No divergences to plot.
```

```r
s1_new <- summary(fit_1_new_priors)
s2_new <- summary(fit_2_new_priors)
s3_new <- summary(fit_3_new_priors)

print(s1_new)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: pres.abs ~ altitude + avrain + NoOfPools + meanmin + meanmax
   Data: data (Number of observations: 212)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   -26.07    124.76  -264.68   220.92 1.01     1278     2048
altitude      0.01      0.04    -0.06     0.08 1.01     1314     2058
avrain        0.02      0.06    -0.09     0.13 1.01     1417     1994
NoOfPools     0.03      0.01     0.02     0.05 1.00     2312     2480
meanmin       6.66      1.49     3.87     9.67 1.00     2271     2420
meanmax      -1.28      4.58   -10.39     7.36 1.01     1282     1923

Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
print(s2_new)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: pres.abs ~ NoOfPools + meanmin + altitude
   Data: data (Number of observations: 212)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   -66.66     13.73   -93.98   -40.28 1.00     1630     2012
NoOfPools     0.03      0.01     0.01     0.05 1.00     2124     1691
meanmin       7.29      1.36     4.68    10.00 1.00     1631     1983
altitude      0.03      0.01     0.02     0.04 1.00     1660     2005

Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
print(s3_new)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: pres.abs ~ meanmin + altitude^2
   Data: data (Number of observations: 212)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
Intercept   -63.24     13.20   -90.41   -39.12 1.01     1215     1407
meanmin       6.70      1.29     4.34     9.34 1.01     1202     1416
altitude      0.03      0.01     0.02     0.04 1.01     1228     1472

Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
r1 <- rhat(fit_1_new_priors)
r2_new <- rhat(fit_2_new_priors)
r3_new <- rhat(fit_3_new_priors)
print(r1)
```

```
b_Intercept  b_altitude    b_avrain b_NoOfPools   b_meanmin   b_meanmax
   1.006017    1.005760    1.005071    1.001669    1.003104    1.006261
     lprior        lp__
   1.005584    1.000081
```

```r
print(r2_new)
```

```
b_Intercept b_NoOfPools   b_meanmin  b_altitude      lprior        lp__
   1.001738    1.001123    1.001881    1.001626    1.002028    1.000560
```

```r
print(r3_new)
```

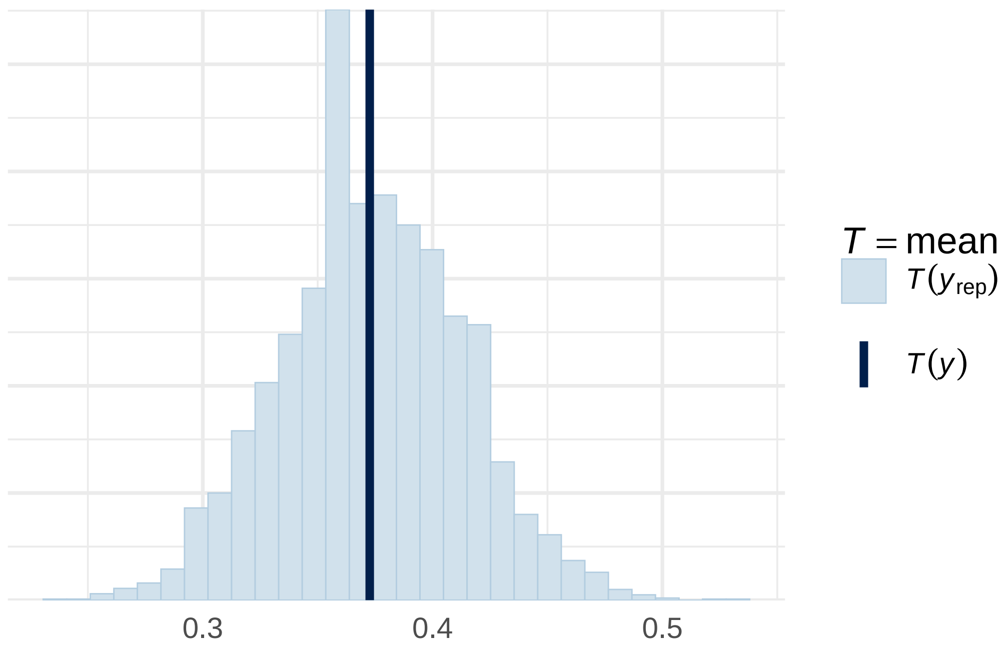
```
b_Intercept   b_meanmin  b_altitude      lprior        lp__
   1.005434    1.005309    1.005380    1.004929    1.002796
```

```r
loo_fit_1_new_priors <- loo(fit_1_new_priors)
loo_fit_2_new_priors <- loo(fit_2_new_priors)
loo_fit_3_new_priors <- loo(fit_3_new_priors)

loo_compare(loo_fit_1_new_priors,loo_fit_2_new_priors,loo_fit_3_new_priors)
```

```
                  elpd_diff se_diff
fit_2_new_priors  0.0       0.0
fit_1_new_priors -1.0       1.3
fit_3_new_priors -6.5       3.9
```

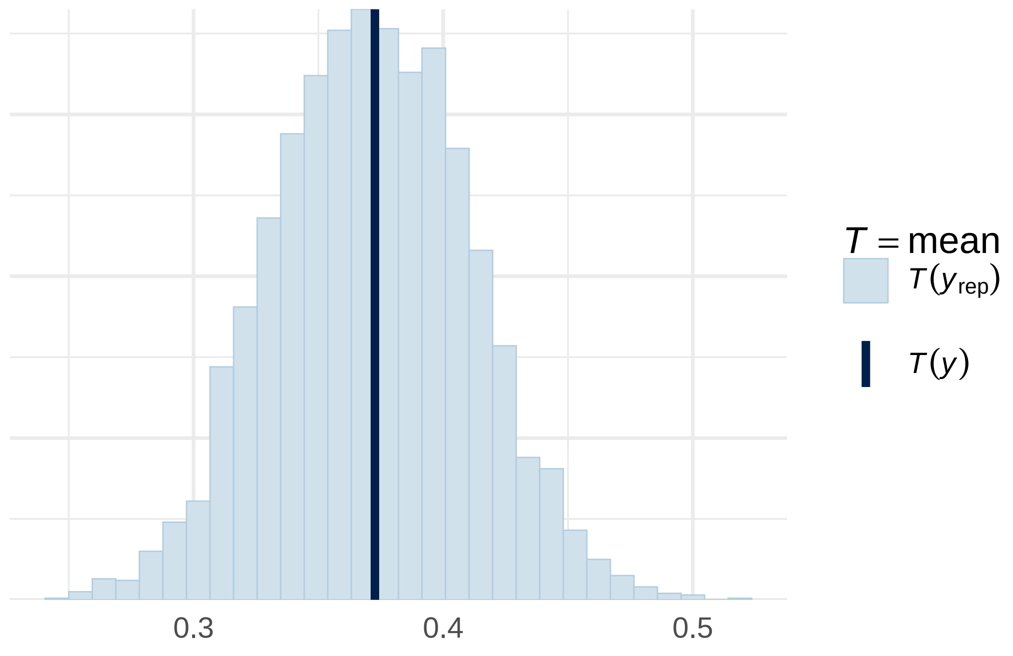
```r
print(pp_check_fit_1_new_priors)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



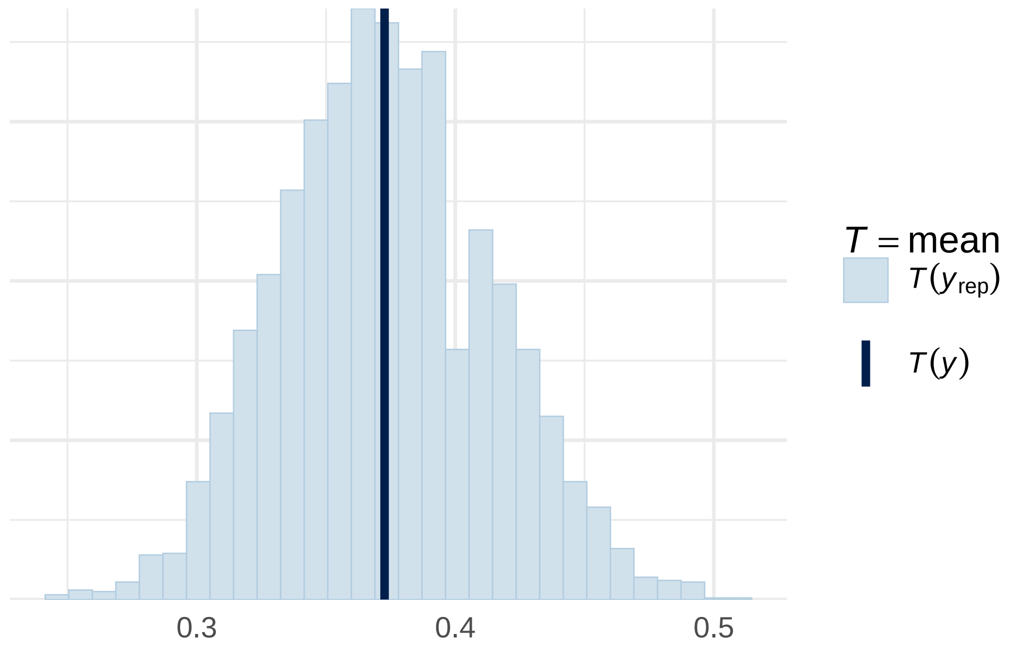
```r
print(pp_check_fit_2_new_priors)
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

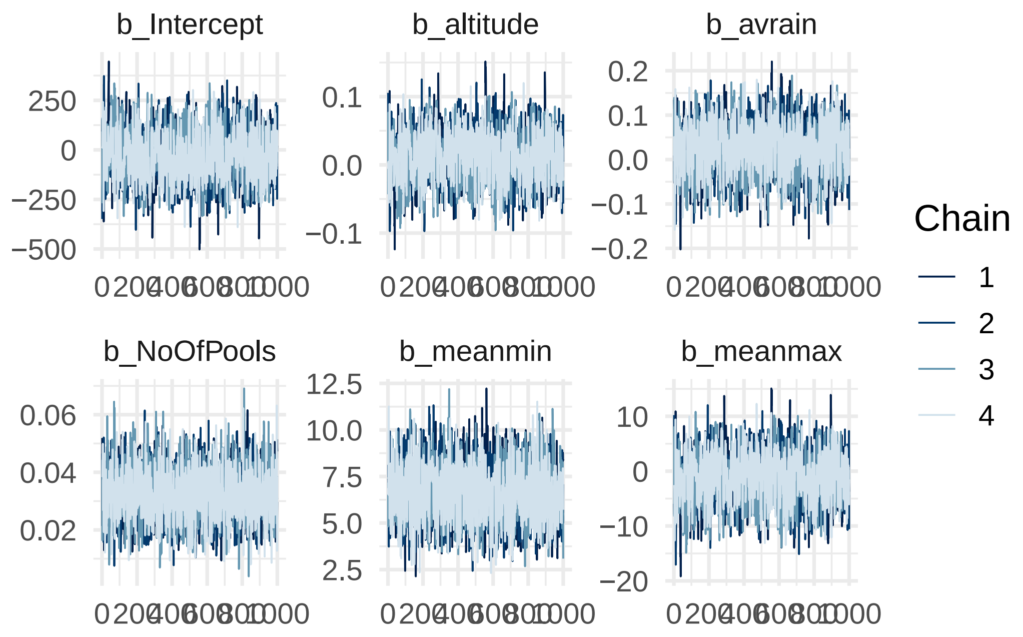


```r
print(pp_check_fit_3_new_priors)
```

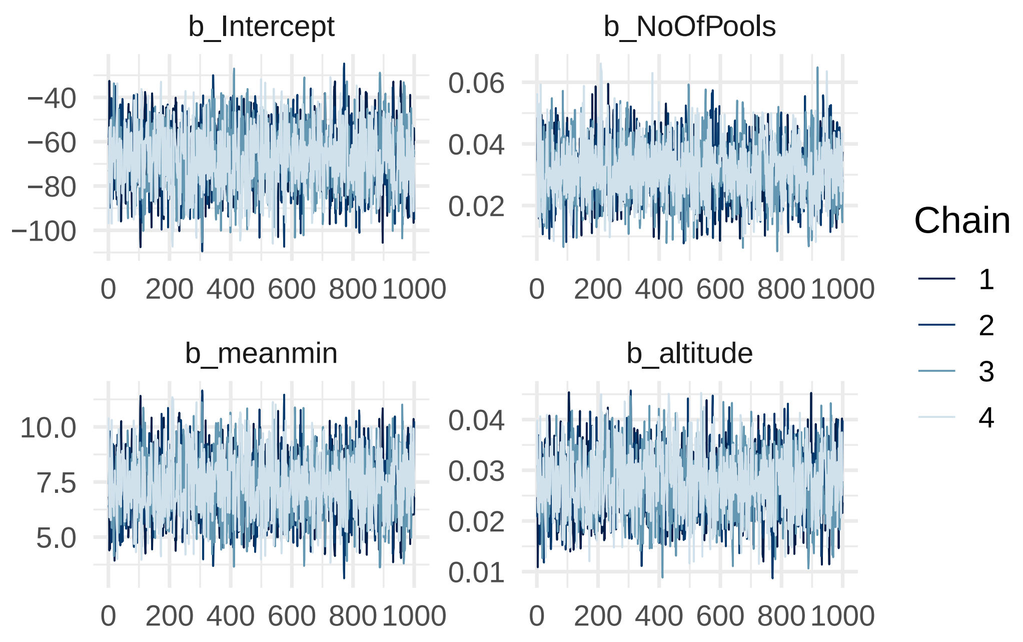
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



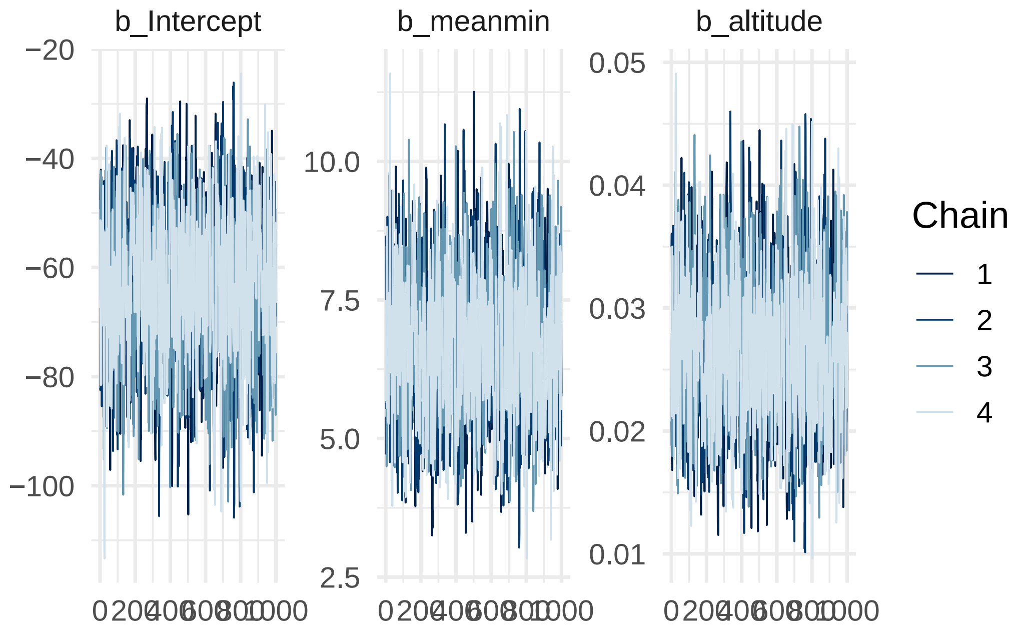
```r
print(traceplot_fit_1_new_priors)
```



```
print(traceplot_fit_2_new_priors)
```



```
print(traceplot_fit_3_new_priors)
```



```r
stancode(fit1)
```

```
// generated with brms 2.20.4
functions {

}
data {
  int<lower=1> N; // total number of observations
  array[N] int Y; // response variable
  int<lower=1> K; // number of population-level effects
  matrix[N, K] X; // population-level design matrix
  int<lower=1> Kc; // number of population-level effects after centering
  int prior_only; // should the likelihood be ignored?
}
transformed data {
  matrix[N, Kc] Xc; // centered version of X without an intercept
  vector[Kc] means_X; // column means of X before centering
  for (i in 2 : K) {
    means_X[i - 1] = mean(X[ : , i]);
    Xc[ : , i - 1] = X[ : , i] - means_X[i - 1];
  }
}
parameters {
  vector[Kc] b; // regression coefficients
  real Intercept; // temporary intercept for centered predictors
}
transformed parameters {
  real lprior = 0; // prior contributions to the log posterior
  lprior += normal_lpdf(b[1] | 1500, 300);
  lprior += normal_lpdf(b[2] | 0, 100);
  lprior += normal_lpdf(b[3] | 0, 100);
  lprior += normal_lpdf(b[4] | 4, 3);
  lprior += normal_lpdf(b[5] | 15, 10);
  lprior += student_t_lpdf(Intercept | 3, 0, 2.5);
}
model {
  // likelihood including constants
  if (!prior_only) {
    target += bernoulli_logit_glm_lpmf(Y | Xc, Intercept, b);
  }
  // priors including constants
  target += lprior;
}
generated quantities {
  // actual population-level intercept
  real b_Intercept = Intercept - dot_product(means_X, b);
}
```

```
stancode(fit2)
```

```
// generated with brms 2.20.4
functions {

}
data {
  int<lower=1> N; // total number of observations
  array[N] int Y; // response variable
  int<lower=1> K; // number of population-level effects
  matrix[N, K] X; // population-level design matrix
  int<lower=1> Kc; // number of population-level effects after centering
  int prior_only; // should the likelihood be ignored?
}
transformed data {
  matrix[N, Kc] Xc; // centered version of X without an intercept
  vector[Kc] means_X; // column means of X before centering
  for (i in 2 : K) {
    means_X[i - 1] = mean(X[ : , i]);
    Xc[ : , i - 1] = X[ : , i] - means_X[i - 1];
  }
}
parameters {
  vector[Kc] b; // regression coefficients
  real Intercept; // temporary intercept for centered predictors
}
transformed parameters {
  real lprior = 0; // prior contributions to the log posterior
  lprior += normal_lpdf(b[1] | 0, 100);
  lprior += normal_lpdf(b[2] | 4, 3);
  lprior += normal_lpdf(b[3] | 1500, 300);
  lprior += student_t_lpdf(Intercept | 3, 0, 2.5);
}
model {
  // likelihood including constants
  if (!prior_only) {
    target += bernoulli_logit_glm_lpmf(Y | Xc, Intercept, b);
  }
  // priors including constants
  target += lprior;
}
generated quantities {
  // actual population-level intercept
  real b_Intercept = Intercept - dot_product(means_X, b);
}
```

```
stancode(fit3)
```

```
// generated with brms 2.20.4
functions {

}
data {
  int<lower=1> N; // total number of observations
  array[N] int Y; // response variable
  int<lower=1> K; // number of population-level effects
  matrix[N, K] X; // population-level design matrix
  int<lower=1> Kc; // number of population-level effects after centering
  int prior_only; // should the likelihood be ignored?
}
transformed data {
  matrix[N, Kc] Xc; // centered version of X without an intercept
  vector[Kc] means_X; // column means of X before centering
  for (i in 2 : K) {
    means_X[i - 1] = mean(X[ : , i]);
    Xc[ : , i - 1] = X[ : , i] - means_X[i - 1];
  }
}
parameters {
  vector[Kc] b; // regression coefficients
  real Intercept; // temporary intercept for centered predictors
}
transformed parameters {
  real lprior = 0; // prior contributions to the log posterior
  lprior += normal_lpdf(b[1] | 4, 3);
  lprior += normal_lpdf(b[2] | 1500, 300);
  lprior += student_t_lpdf(Intercept | 3, 0, 2.5);
}
model {
  // likelihood including constants
  if (!prior_only) {
    target += bernoulli_logit_glm_lpmf(Y | Xc, Intercept, b);
  }
  // priors including constants
  target += lprior;
}
generated quantities {
  // actual population-level intercept
  real b_Intercept = Intercept - dot_product(means_X, b);
}
```