gneiting_-_probabilistic_forecasts_calibration_sharpness

Evan L. Ray

8/21/2020

Article

This is about the article Probabilistic forecasts, calibration and sharpness by Gneiting et al. https://rss.onlinelibrary.wiley.com/doi/10.1111/j.1467-9868.2007.00587.x

Summary

Introduction

G_t is data generating process, F_t is a forecasting model.

PIT is $p_t = F_t(x_t)$. $p_t \sim Unif(0, 1)$ is a necessary but not sufficient condition for a forecaster to be ideal.

Example:

DGP is:

$$\mu_t \sim N(0, 1)$$
$$X_t \sim N(\mu_t, 1)$$

Three other models in table.

```
r_dgp <- function(n) {
  mu_t <- rnorm(n)
  x_t <- rnorm(n = length(mu_t), mean = mu_t)
  return(list(mu_t = mu_t, x_t = x_t))
}

d_dgp <- function(x_t, mu_t) {
  dnorm(x = x_t, mean = mu_t, sd = 1)
}

p_dgp <- function(x_t, mu_t) {
  if(missing(mu_t)) {
    mu_t <- rnorm(length(x_t))
  }
  pnorm(q = x_t, mean = mu_t, sd = 1)
}

q_dgp <- function(p, mu_t) {
  qnorm(p = p, mean = mu_t, sd = 1)
```

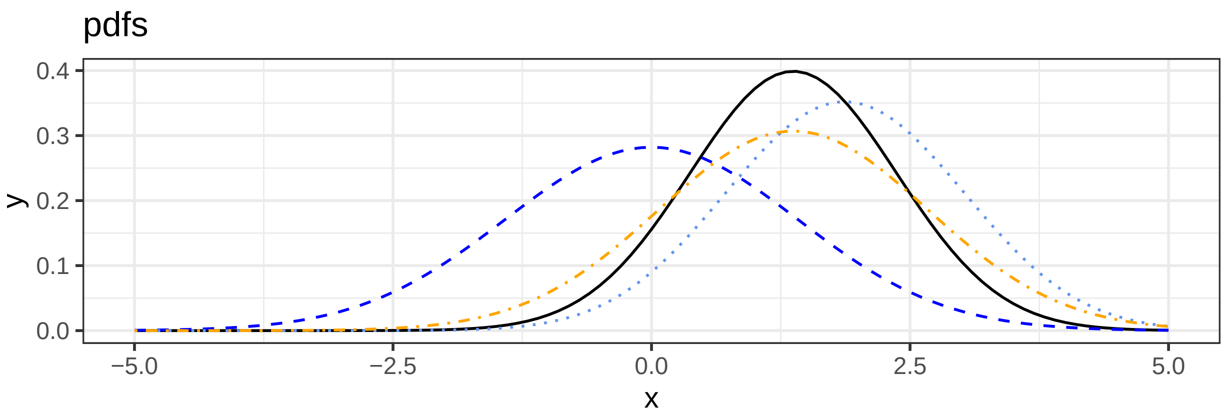
```
}

r_f1 <- function(n) {
  list(
    x_t = rnorm(n = n, mean = 0, sd = sqrt(2))
  )
}

d_f1 <- function(x_t, ...) {
  dnorm(x = x_t, mean = 0, sd = sqrt(2))
}

p_f1 <- function(x_t, ...) {
  pnorm(q = x_t, mean = 0, sd = sqrt(2))
}

q_f1 <- function(p, ...) {
  qnorm(p = p, mean = 0, sd = sqrt(2))
}

r_f2 <- function(n) {
  mu_t <- rnorm(n)
  tau_t <- sample(x = c(-1, 1), size = n, replace = TRUE)
  x_t <- 0.5 * (rnorm(n = n, mean = mu_t) + rnorm(n = n, mean = mu_t + tau_t))
  return(list(mu_t = mu_t, tau_t = tau_t, x_t = x_t))
}

d_f2 <- function(x_t, mu_t, tau_t) {
  x_t <- 0.5 * (dnorm(x = x_t, mean = mu_t) + dnorm(x = x_t, mean = mu_t + tau_t))
}

p_f2 <- function(x_t, mu_t, tau_t) {
  if(missing(tau_t)) {
    tau_t <- sample(x = c(-1, 1), size = length(x_t), replace = TRUE)
  }
  x_t <- 0.5 * (pnorm(q = x_t, mean = mu_t) + pnorm(q = x_t, mean = mu_t + tau_t))
}

q_f2 <- function(p, mu_t, tau_t) {
  if(missing(tau_t)) {
    tau_t <- sample(x = c(-1, 1), size = length(p), replace = TRUE)
  }
  x_t <- 0.5 * (qnorm(p = p, mean = mu_t) + pnorm(q = x_t, mean = mu_t + tau_t))
}

r_f3 <- function(n) {
  mu_t <- rnorm(n)
  mix_ind <- sample(x = 1:3, size = n, replace = TRUE)
  delta_t <- c(0.5, -0.5, 0)[mix_ind]
  sigma_sq_t <- c(1, 1, 169/100)[mix_ind]

  list(
    x_t = rnorm(n = n, mean = mu_t + delta_t, sd = sqrt(sigma_sq_t)),
```

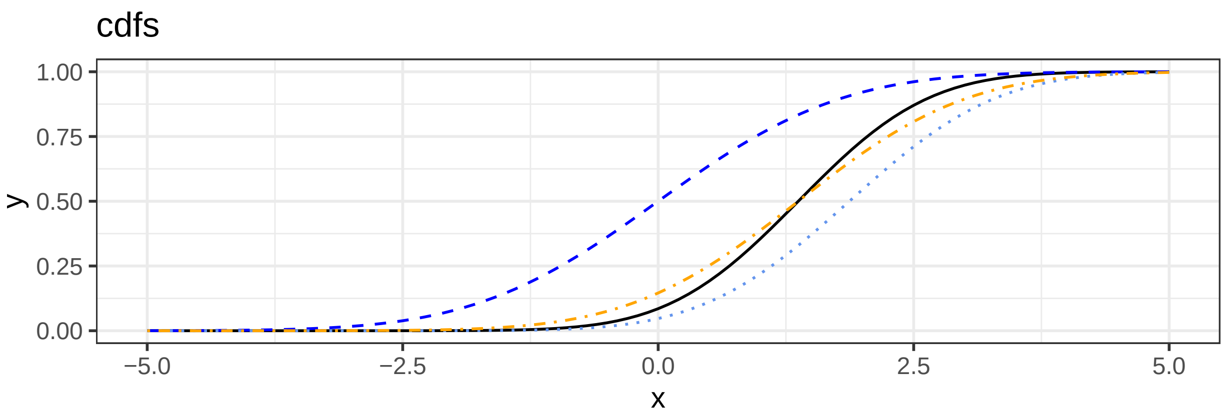
```r
    mix_ind = mix_ind
  )
}

d_f3 <- function(x_t, mu_t, mix_ind) {
  delta_t <- c(0.5, -0.5, 0)[mix_ind]
  sigma_sq_t <- c(1, 1, 169/100)[mix_ind]

  dnorm(x = x_t, mean = mu_t + delta_t, sd = sqrt(sigma_sq_t))
}

p_f3 <- function(x_t, mu_t, mix_ind) {
  if(missing(mix_ind)) {
    mix_ind <- sample(x = 1:3, size = length(x_t), replace = TRUE)
  }
  delta_t <- c(0.5, -0.5, 0)[mix_ind]
  sigma_sq_t <- c(1, 1, 169/100)[mix_ind]

  pnorm(q = x_t, mean = mu_t + delta_t, sd = sqrt(sigma_sq_t))
}

q_f3 <- function(p, mu_t, mix_ind) {
  delta_t <- c(0.5, -0.5, 0)[mix_ind]
  sigma_sq_t <- c(1, 1, 169/100)[mix_ind]

  qnorm(p = p, mean = mu_t + delta_t, sd = sqrt(sigma_sq_t))
}
```

Example predictive distributions for a single time point:

```r
set.seed(42)
sample_dgp <- r_dgp(n = 1)
sample_f1 <- r_f1(n = 1)
sample_f2 <- r_f2(n = 1)
sample_f3 <- r_f3(n = 1)


p1 <- ggplot(data = data.frame(x = c(-5, 5)), mapping = aes(x = x)) +
  stat_function(
    fun = d_dgp, args = list(mu_t = sample_dgp$mu_t)
  ) +
  stat_function(
    fun = d_f1,
    color = "blue",
    linetype = 2
  ) +
  stat_function(
    fun = d_f2, args = list(mu_t = sample_dgp$mu_t, tau_t = sample_f2$tau_t),
    color = "cornflowerblue",
    linetype = 3
  ) +
  stat_function(
```

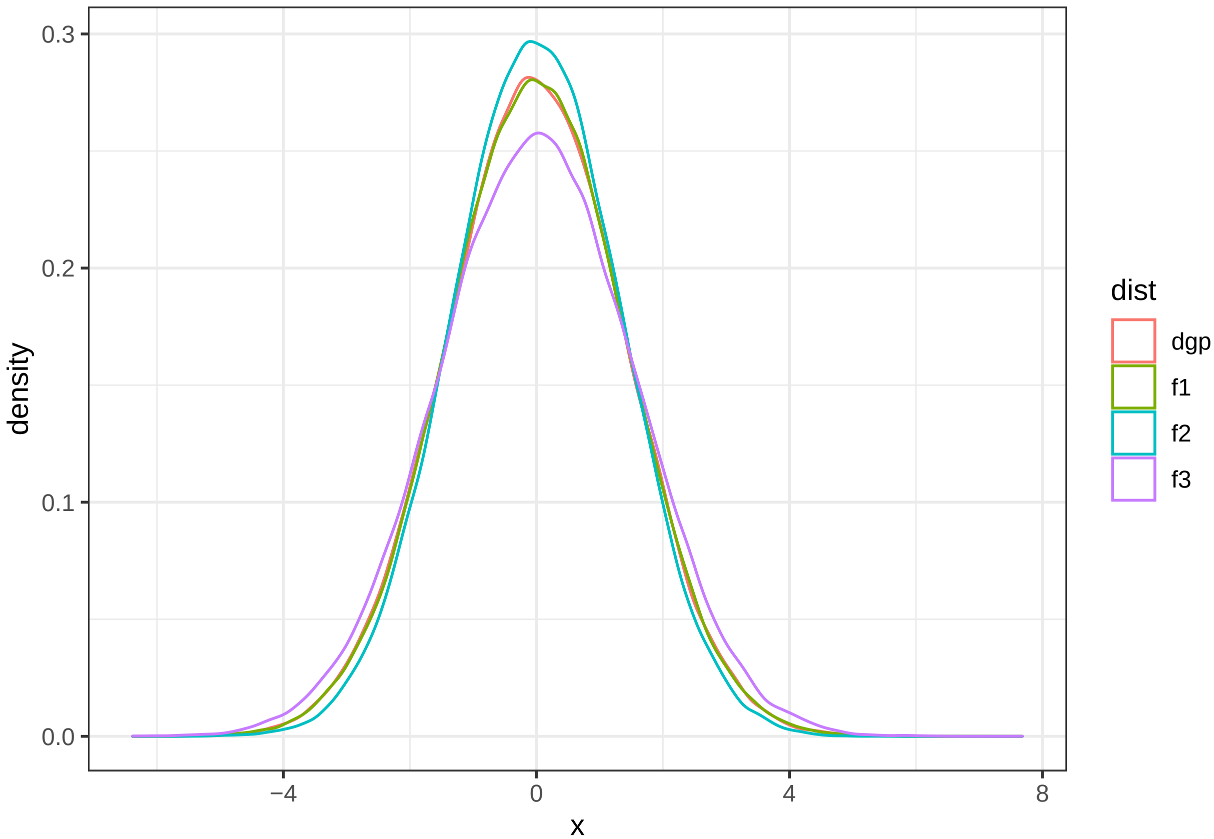
```r
    fun = d_f3, args = list(mu_t = sample_dgp$mu_t, mix_ind = sample_f3$mix_ind),
    color = "orange",
    linetype = 4
  ) +
  ggtitle("pdfs") +
  theme_bw()


p2 <- ggplot(data = data.frame(x = c(-5, 5)), mapping = aes(x = x)) +
  stat_function(
    fun = p_dgp, args = list(mu_t = sample_dgp$mu_t)
  ) +
  stat_function(
    fun = p_f1,
    color = "blue",
    linetype = 2
  ) +
  stat_function(
    fun = p_f2, args = list(mu_t = sample_dgp$mu_t, tau_t = sample_f2$tau_t),
    color = "cornflowerblue",
    linetype = 3
  ) +
  stat_function(
    fun = p_f3, args = list(mu_t = sample_dgp$mu_t, mix_ind = sample_f3$mix_ind),
    color = "orange",
    linetype = 4
  ) +
  ggtitle("cdfs") +
  theme_bw()

gridExtra::grid.arrange(p1, p2)
```





Marginal predictive distributions:

```r
n <- 100000

samples <- purrr::map_dfr(
  c("dgp", paste0("f", 1:3)),
  function(dist_name) {
    data.frame(
      dist = dist_name,
      x = do.call(
        paste0("r_", dist_name),
        args = list(n = n)
      )$x_t,
      stringsAsFactors = FALSE
    )
  }
)

ggplot(data = samples, mapping = aes(x = x, color = dist)) +
  geom_density() +
  theme_bw()
```



PITs

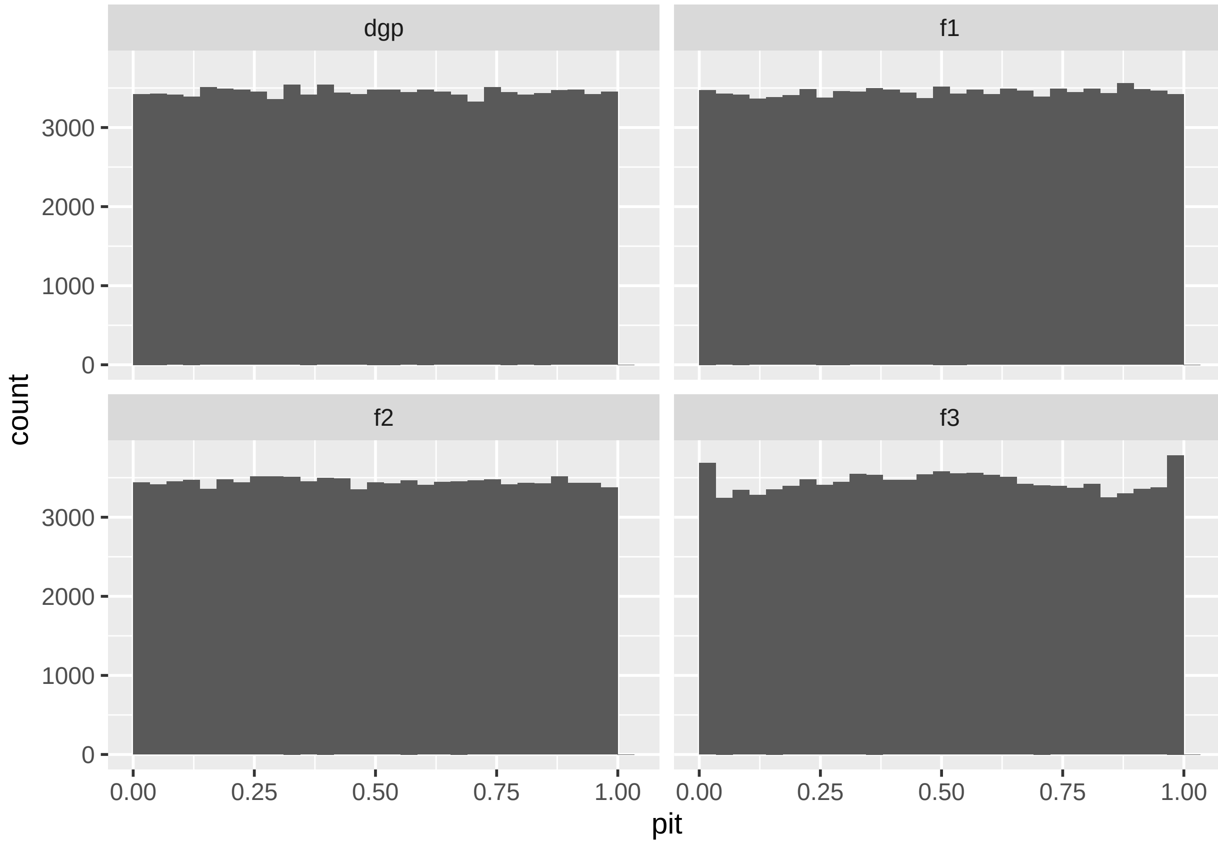
```
samples <- r_dgp(n)

pits <- purrr::map_dfr(
  c("dgp", paste0("f", 1:3)),
  function(dist_name) {
    data.frame(
      dist = dist_name,
      pit = do.call(
        paste0("p_", dist_name),
        args = samples
      )
    )
  }
)
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
```

```
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector
```

```
ggplot(data = pits, mapping = aes(x = pit)) +
  geom_histogram(boundary = 0) +
  facet_wrap( ~ dist)
```

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



PITs, but making the marginal model the DGP and the original DGP a predictive model

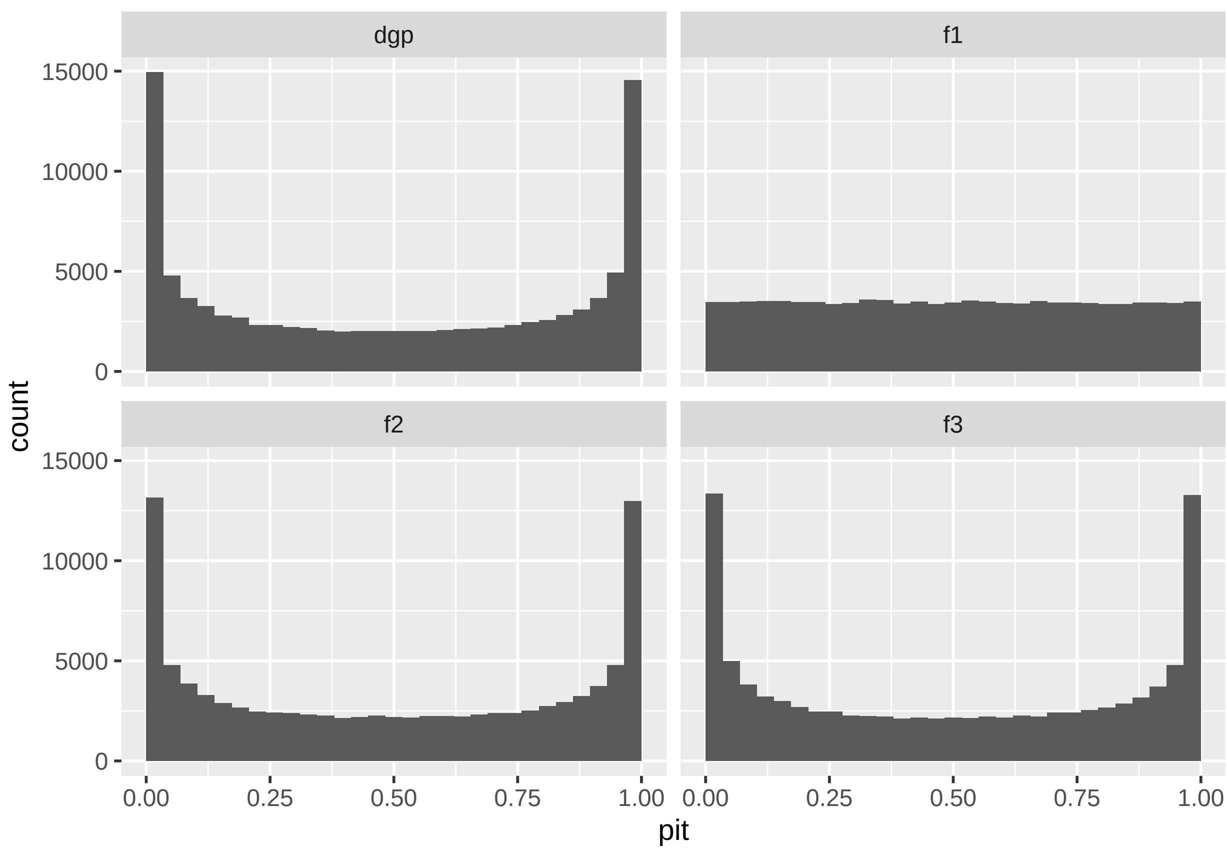
```
samples <- r_f1(n)
dgp_samples <- r_dgp(n)

pits <- purrr::map_dfr(
  c("dgp", paste0("f", 1:3)),
  function(dist_name) {
    call_args <- samples
    if(dist_name != "f1") {
      call_args$mu_t <- dgp_samples$mu_t
    }
    data.frame(
      dist = dist_name,
      pit = do.call(
```

```
        paste0("p_", dist_name),
        args = call_args
      )
    )
  }
)
```

```
## Warning in bind_rows_(x, .id): Unequal factor levels: coercing to character

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector, coercing
## into character vector
```

```
ggplot(data = pits, mapping = aes(x = pit)) +
  geom_histogram(boundary = 0) +
  facet_wrap( ~ dist)
```

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



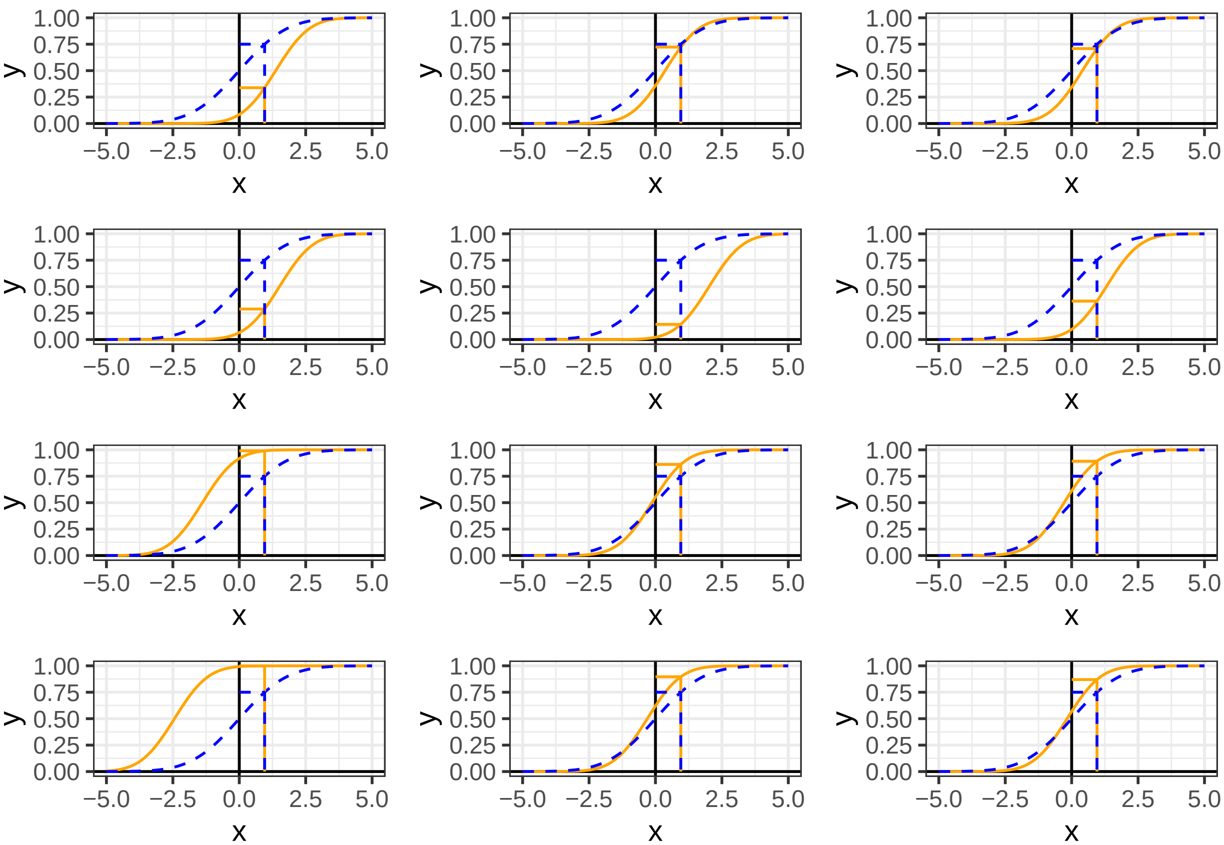
Section 2: Modes of Calibration

climatological forecaster (marginal distribution) is probabilistically calibrated

We pick a particular value $p = 0.75$ at which to explore probabilistic calibration.

```
prob_val <- 0.75

set.seed(42)

make_prob_cal_plot <- function() {
  sample_dgp <- r_dgp(n = 1)

  x_val <- q_f1(p = prob_val)

  if(x_val > 0) {
    dgp_lines <- data.frame(
      x = c(0, rep(x_val, 2)),
      y = c(rep(p_dgp(x_val, mu_t = sample_dgp$mu_t), 2), 0),
      model = "dgp"
    )

    f1_lines <- data.frame(
      x = c(0, rep(x_val, 2)),
      y = c(rep(prob_val, 2), 0)
    )
  } else {
    dgp_lines <- data.frame(
      x = c(rep(x_val, 2), 0),
      y = c(0, prob_val, prob_val),
      model = "dgp"
    )

    f1_lines <- data.frame(
      x = c(rep(x_val, 2), 0),
      y = c(0, rep(p_f1(x_val), 2))
    )
  }

  p <- ggplot(data = data.frame(x = c(-5, 5)), mapping = aes(x = x)) +
    geom_hline(yintercept = 0) +
    geom_vline(xintercept = 0) +
    stat_function(
      fun = p_dgp, args = list(mu_t = sample_dgp$mu_t),
      color = "orange"
    ) +
    stat_function(
      fun = p_f1,
      color = "blue",
      linetype = 2
    ) +
    geom_line(data = dgp_lines, mapping = aes(x = x, y = y),
              color = "orange") +
    geom_line(data = f1_lines, mapping = aes(x = x, y = y),
```

```
                color = "blue", linetype = 2) +
    theme_bw()

  return(p)
}

plots <- lapply(1:12, function(i) make_prob_cal_plot())

grid.arrange(grobs = plots)
```

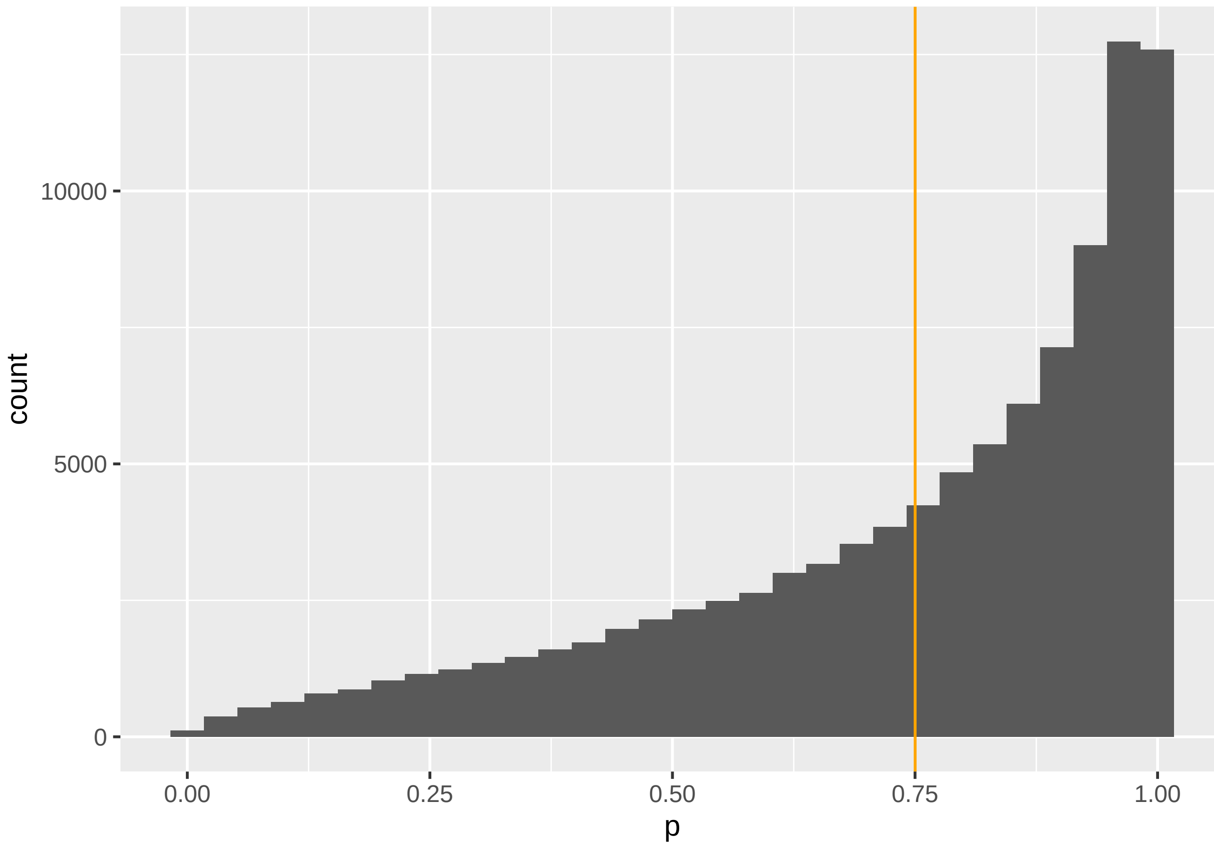


Probabilistic calibration: at each probability level p, on average across all time points, the true probability of being less than the p'th quantile of the predictive distribution is p. Note that this does not have to hold for every time point.

```
set.seed(42)
get_f1_prob_cal_rel_dgp <- function() {
  sample_dgp <- r_dgp(n = 1)

  x_val <- q_f1(p = prob_val)

  return(p_dgp(x_val, mu_t = sample_dgp$mu_t))
}

to_plot <- data.frame(
  p = purrr::map_dbl(1:100000, function(i) get_f1_prob_cal_rel_dgp())
)
```

```r
ggplot(data = to_plot, mapping = aes(x = p)) +
  geom_histogram() +
  geom_vline(xintercept = mean(to_plot$p), color = "orange")
```

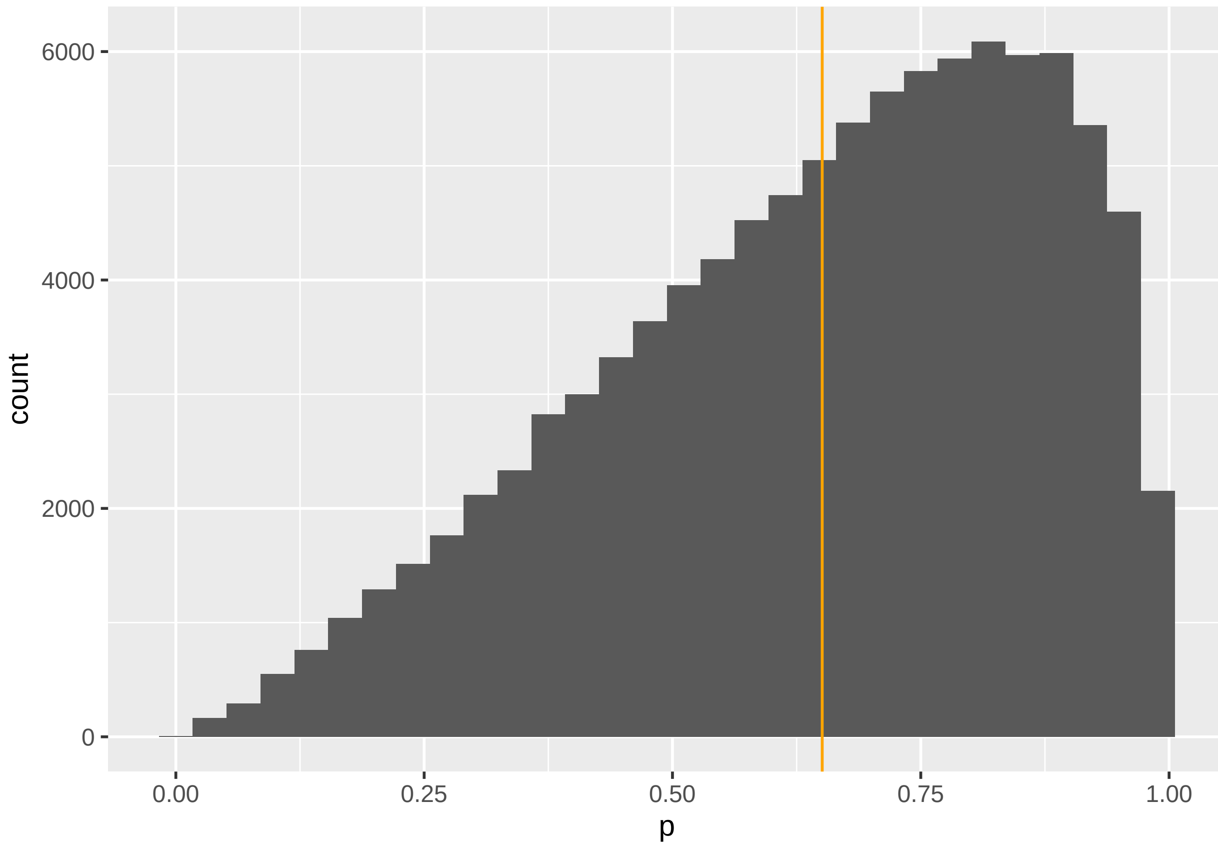
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Note: I accidentally code it up backwards at first and the relationship does not work in reverse!

```r
set.seed(42)
get_f1_prob_cal_rel_dgp <- function() {
  sample_dgp <- r_dgp(n = 1)

  x_val <- q_dgp(p = prob_val, mu_t = sample_dgp$mu_t)

  return(p_f1(x_val))
}

to_plot <- data.frame(
  p = purrr::map_dbl(1:100000, function(i) get_f1_prob_cal_rel_dgp())
)

ggplot(data = to_plot, mapping = aes(x = p)) +
  geom_histogram() +
  geom_vline(xintercept = mean(to_plot$p), color = "orange")
```

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



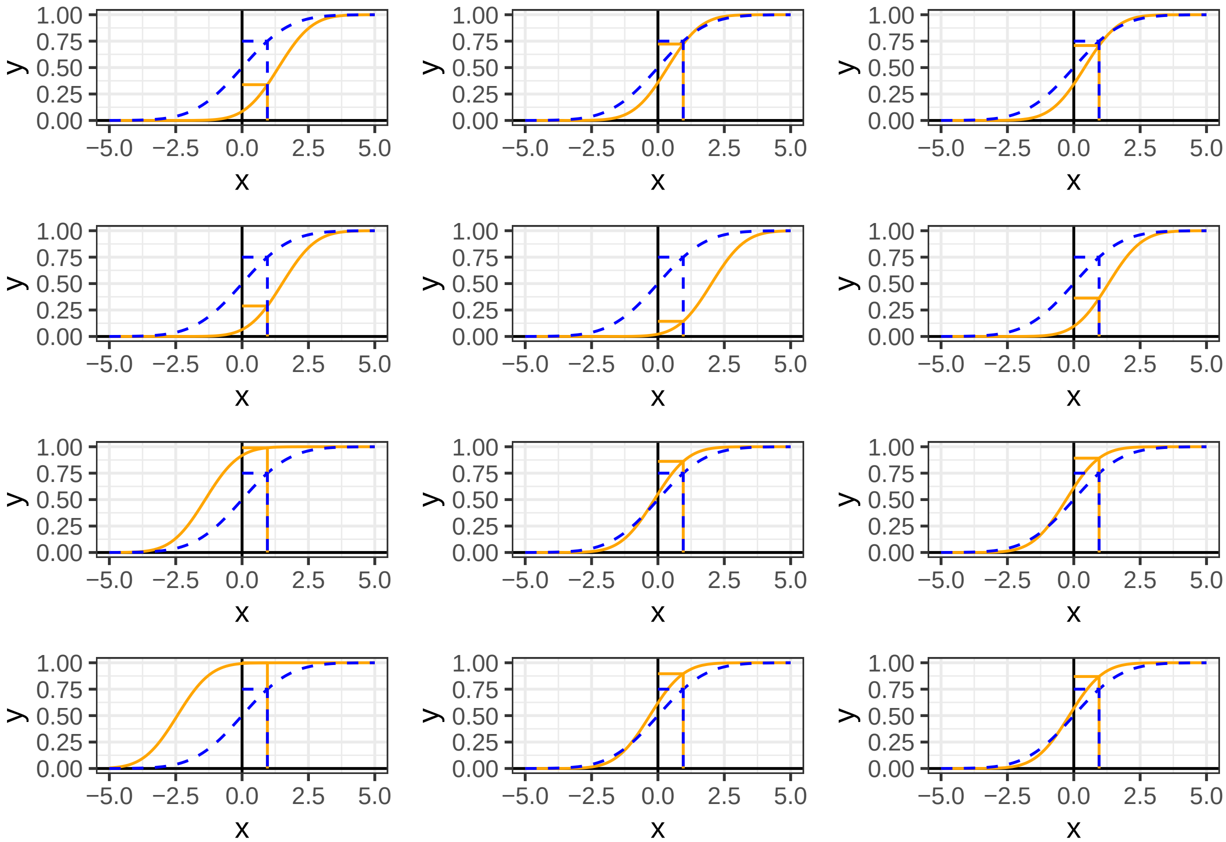
This is also an illustration that marginal calibration does not imply probabilistic calibration.

Marginal distribution is exceedance calibrated

We pick a particular value $x = 0.75$ at which to explore exceedance calibration.

```
x_val <- 0.75

set.seed(42)

make_prob_cal_plot <- function() {
  sample_dgp <- r_dgp(n = 1)

  x_val <- q_f1(p = prob_val)

  if(x_val > 0) {
    dgp_lines <- data.frame(
      x = c(0, rep(x_val, 2)),
      y = c(rep(p_dgp(x_val, mu_t = sample_dgp$mu_t), 2), 0),
      model = "dgp"
    )

    f1_lines <- data.frame(
      x = c(0, rep(x_val, 2)),
      y = c(rep(prob_val, 2), 0)
    )
  } else {
```

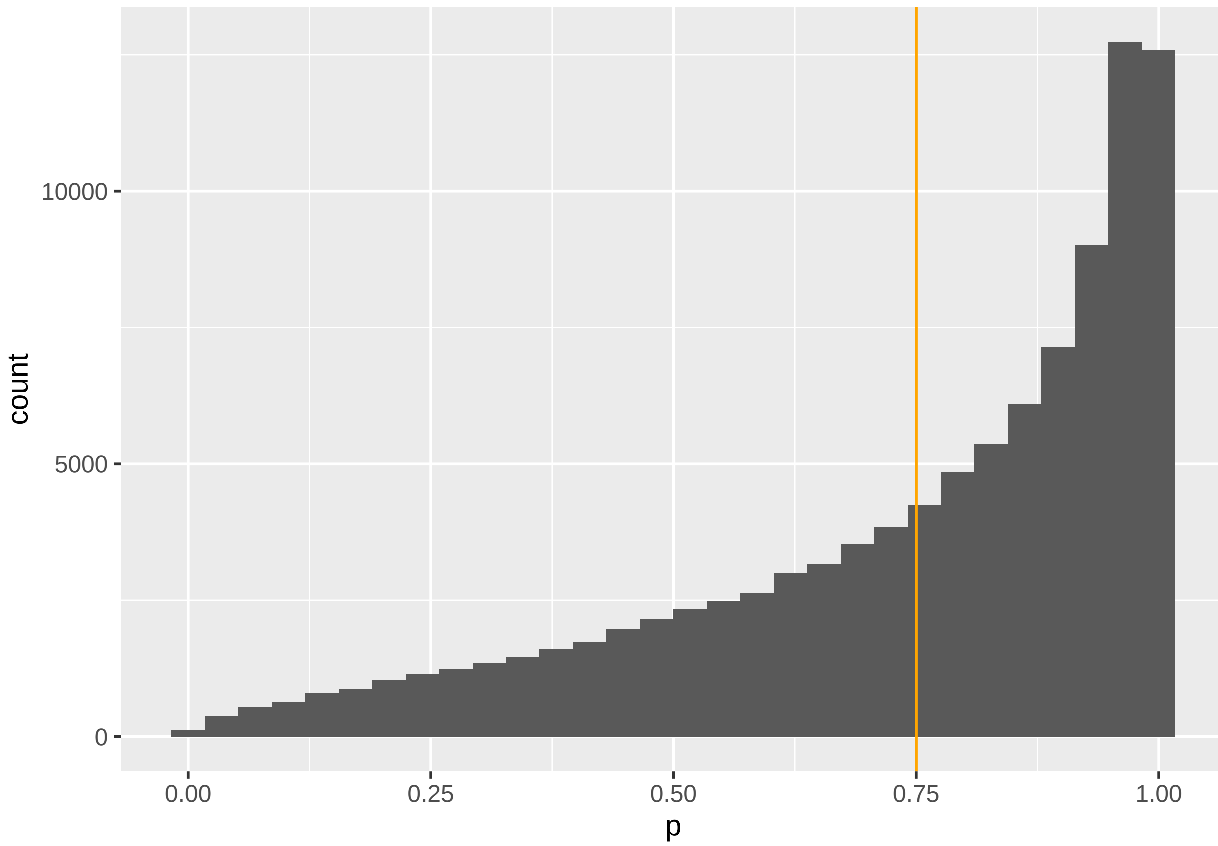
```r
    dgp_lines <- data.frame(
      x = c(rep(x_val, 2), 0),
      y = c(0, prob_val, prob_val),
      model = "dgp"
    )

    f1_lines <- data.frame(
      x = c(rep(x_val, 2), 0),
      y = c(0, rep(p_f1(x_val), 2))
    )
  }

  p <- ggplot(data = data.frame(x = c(-5, 5)), mapping = aes(x = x)) +
    geom_hline(yintercept = 0) +
    geom_vline(xintercept = 0) +
    stat_function(
      fun = p_dgp, args = list(mu_t = sample_dgp$mu_t),
      color = "orange"
    ) +
    stat_function(
      fun = p_f1,
      color = "blue",
      linetype = 2
    ) +
    geom_line(data = dgp_lines, mapping = aes(x = x, y = y),
              color = "orange") +
    geom_line(data = f1_lines, mapping = aes(x = x, y = y),
              color = "blue", linetype = 2) +
    theme_bw()

  return(p)
}

plots <- lapply(1:12, function(i) make_prob_cal_plot())

grid.arrange(grobs = plots)
```



Probabilistic calibration: at each probability level p, on average across all time points, the true probability of being less than the p'th quantile of the predictive distribution is p. Note that this does not have to hold for every time point.

```r
set.seed(42)
get_f1_prob_cal_rel_dgp <- function() {
  sample_dgp <- r_dgp(n = 1)

  x_val <- q_f1(p = prob_val)

  return(p_dgp(x_val, mu_t = sample_dgp$mu_t))
}

to_plot <- data.frame(
  p = purrr::map_dbl(1:100000, function(i) get_f1_prob_cal_rel_dgp())
)

ggplot(data = to_plot, mapping = aes(x = p)) +
  geom_histogram() +
  geom_vline(xintercept = mean(to_plot$p), color = "orange")
```

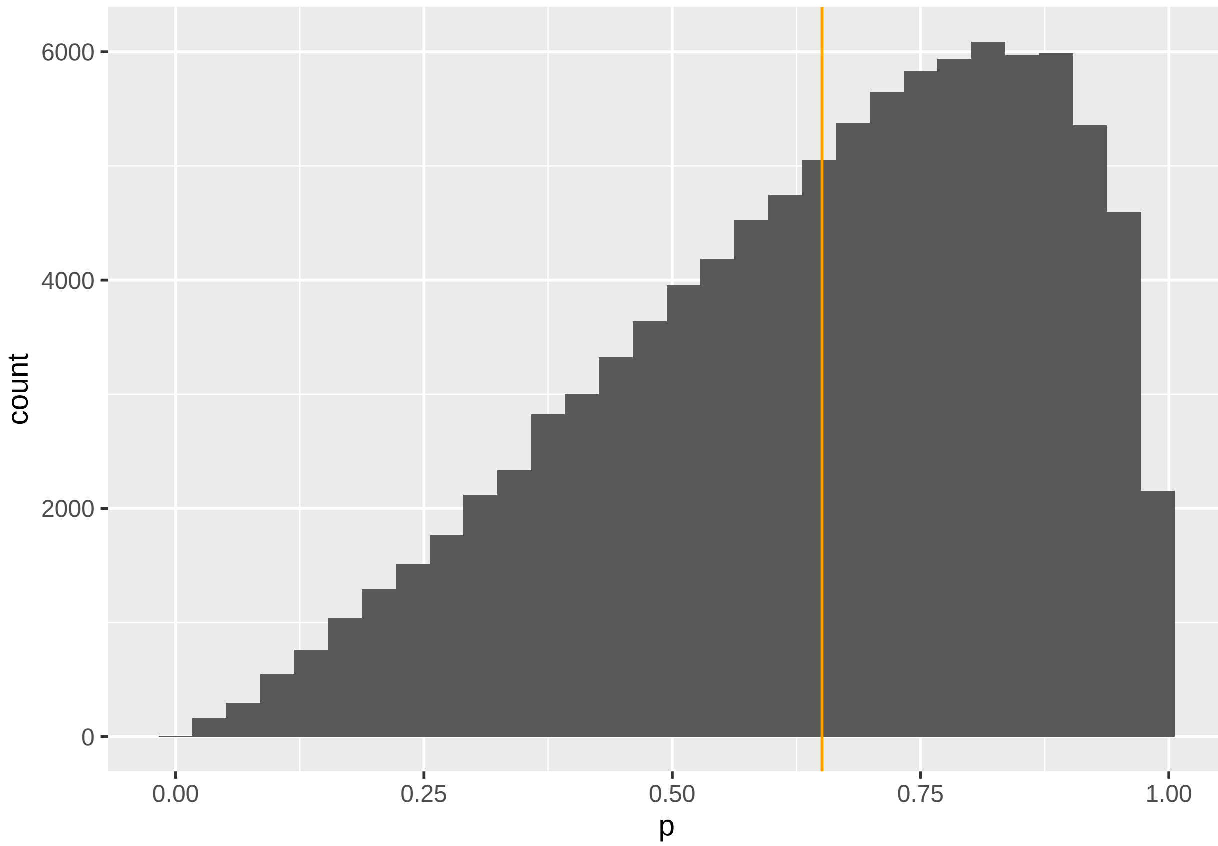
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Note: I accidentally code it up backwards at first and the relationship does not work in reverse!

```
set.seed(42)
get_f1_prob_cal_rel_dgp <- function() {
  sample_dgp <- r_dgp(n = 1)

  x_val <- q_dgp(p = prob_val, mu_t = sample_dgp$mu_t)

  return(p_f1(x_val))
}

to_plot <- data.frame(
  p = purrr::map_dbl(1:100000, function(i) get_f1_prob_cal_rel_dgp())
)

ggplot(data = to_plot, mapping = aes(x = p)) +
  geom_histogram() +
  geom_vline(xintercept = mean(to_plot$p), color = "orange")
```

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



This is also an illustration that marginal calibration does not imply probabilistic calibration.