Microeconomics 3 problem set 1

Oliver Snellman

22 March, 2020

1. Introduction

This exercise set deals with modeling of limited dependant variables, namely the binary case $y \in \{0, 1\}$, and providing interpretations for the results; marginal effects of explanatory variables. The models try to map transformations of explanatory variables into probability estimates $p := \mathbf{P}(y = 1|X)$. We define a suitable latent variable y^* (index or utility), for which a linear regression is conducted based on explanatory variables $\mathbf{x}$. y is defined with a cutoff on y^*. Typically the cutoff is at zero:

$$y_i = \begin{cases} 1, & \text{if } y_i^* > 0 \\ 0, & \text{if } y_i^* \leq 0 \end{cases}$$

Typically a first glance of the data can be conducted by using LPM (OLS), which is also a handy way of approximating the sample average marginal effect of the explanatory variables on the p. Then, depending on the Data Generating Process (DGP), typically either probit or logit model is chosen for the actual analysis. More specificly, the DGP is here defined by the distribution of the error term. The accuracy of the model can be assessed for example with pseudo-R^2 or the ROC/AUC framework.

The general model for limited dependent variable y_i with a latent variable $\mathbf{x_i}\beta + \varepsilon_i =: y_i^*$ where the cutoff is set to 0, is

$$\begin{aligned} p_i := P(y_i = 1|\mathbf{x}) &= P(y^* > 0) \\ &= P(\mathbf{x_i}'\beta + \varepsilon_i > 0) \\ &= P(-\varepsilon_i < \mathbf{x_i}'\beta) \\ &= F_\varepsilon(\mathbf{x_i'}\beta) \end{aligned}$$

where F() is (typically) some CDF, $\underbrace{\mathbf{x_i}}_{Kx1}$ includes values of K observable explanatory variables for observation i, $\underbrace{\beta}_{Kx1}$ has unknown parameters and $\mathbf{x_i}\beta + \varepsilon_i =: y_i^*$, which actually define the state of y_i.

Verbal intuition for the last two lines: y_i is defined by y_i^*, which includes noise ε_i, hence $\mathbf{x_i}'\beta$ doesn't fully determine y_i. Our confidence on the signal issued by $\mathbf{x_i}'\beta$ (the value being positive or negative) depends on the distribution of ε_i. The reason is, that if the error ε_i is larger than $\mathbf{x_i}'\beta$ with the opposite sign, $-\varepsilon_i > \mathbf{x_i}'\beta$, then their sum will cross the threshold of 0 and cause the y^* to have the opposite sign from the $\mathbf{x_i}'\beta$. Hence, at each value in the range of $\mathbf{x_i}$, our confidence for $y = 1$ has to equal: one minus the probability, that ε reaches a larger and opposite value than $\mathbf{x}'\beta$. This is by definition $P(-\varepsilon_i > \mathbf{x_i}'\beta) = 1 - F_\varepsilon(-\mathbf{x_i}'\beta) = F_\varepsilon(\mathbf{x_i'}\beta)$.

2. Set up

Linear Probability Model (LPM)

Let's start off by defining the DGP to be a Linear Probability Model, which has additively separable error term ε.

$$y^* = \mathbf{X}\beta + \varepsilon$$

where $y_i^* \in [0,1]$, $\underbrace{X}_{NxK} = (\mathbf{x_1}', \ldots, \mathbf{x_N}')$, $x_i = (1, x_{i1}, \ldots x_{iK})$, $\beta = (\beta_0, \beta_1, \ldots \beta_K)'$, and $\mathbb{P}(y_i = 1|x_i) = F(x_i'\beta) = y^*$. The error term $\varepsilon_i = y - y^*$ captures everything that the model can't explain in more straight forward manner than in the forecoming models.

The link to the actual binary observable variable y is defined by the cutoff at zero.

Probit

Next we specify the Probit link. It is a mapping from the latent $y^* \in \mathbb{R}$ to the [0,1] interval using the CDF of the standard normal distribution $F_\varepsilon() = \Phi()$:

$$
\begin{aligned}
p &= P(y_i = 1|\mathbf{x}) \\
&= \Phi(\mathbf{x_i}'\beta) \qquad\qquad |z = \mathbf{x_i}'\beta \\
&= \int_{-\infty}^{\mathbf{x}'\beta} \phi(z)dz \\
&= \int_{-\infty}^{\mathbf{x}'\beta} \frac{1}{\sqrt{2\pi}} e^{\frac{-z^2}{2}} dz \quad | \times \Phi^{-1} \\
\Phi^{-1}(p) &=: y^* = \mathbf{x_i}\beta
\end{aligned}
$$

which is reasonable tool to use, if the error term ε can be expected to be normally distributed. Here again, y^* is the real random variable, which defines the state of y.

Logit

Logit is similar to probit, but with different CDF for the error term; that of the logistic function

$$
\begin{aligned}
p &= P(y_i = 1|\mathbf{x}) \\
&= \Lambda(\mathbf{x_i}'\beta) \\
&= \frac{e^{\mathbf{x_i}'\beta}}{1 + e^{\mathbf{x_i}'\beta}} \qquad | * \frac{e^{-\mathbf{x_i}'\beta}}{e^{-\mathbf{x_i}'\beta}} \\
&= \frac{1}{1 + e^{-\mathbf{x_i}'\beta}}
\end{aligned}
$$

There are three representations for this model, which can be used when interpreting the marginal impacts of the explanatory variables $\mathbf{x_i}$ with estimated coefficients β.

$$p = \frac{e^{\mathbf{x_i}'\beta}}{1 + e^{\mathbf{x_i}'\beta}} \quad | \text{ Dividing by } (1-p) \text{ and rearranging}$$

$$\frac{p}{1-p} = e^{\mathbf{x_i}'\beta} \quad | \, ln()$$

$$ln(\frac{p}{1-p}) = \mathbf{x_i}'\beta$$

where $\frac{p}{1-p}$ is called the odds-ratio. β can be interpreted as semi-elasticity: unit increase in regressor increase the odds ratio by a multiple of β, i.e.

$$\begin{aligned}
\Delta\frac{p}{1-p} &= \Delta e^{\mathbf{x_i}'\beta} \\
&= e^{(\mathbf{x_i}'+\gamma)\beta} - e^{\mathbf{x_i}'\beta} \\
&= e^{\mathbf{x_i}'\beta+\gamma\beta} - e^{\mathbf{x_i}'\beta} \\
&= e^{\mathbf{x_i}'\beta}e^{\gamma\beta} - e^{\mathbf{x_i}'\beta} \\
&= (e^{\gamma\beta} - 1)e^{\mathbf{x_i}'\beta}
\end{aligned}$$

Let's define the change in x_j to be of a unit size $\gamma = 1$. For small β_j this entails $e^{\beta_j} - 1 \simeq \beta_j$, hence

$$\Delta\frac{p}{1-p} = \beta_j e^{\mathbf{x_i}'\beta}$$

2.1.

Define an (micro-) economic decision y_i that results in a binary choice 0/1, with some observable variable x_i affecting the choice.

My choice for $y_i \in \{0, 1\}$ is whether acquires a college degree or not, and $x_i \in \mathbb{N}$ is the education level (in years) of the mother. It is a common sociological finding, that mother's education level strongly predicts the child's educational attainment.

2.2. Specify parameters

Set $\beta_0 = -0.25$ and $\beta_1 = 0.35$, indicating mild positive relation between the explanatory and explained variables.

2.3.

The indexes $i = 1, 2, ..., N$ refer to particular households, from where we obtain the data for the mother and son. The latent variable y^* could be an index of unobserved propensity, like IQ or motivation. It could also relate to the differential utility gained from the choice of higher education, like the outcome of cost benefit

analysis weighting time and effort spent pursuing a diploma against the increased earning potential it creates. The costs vary across individuals depending on their attributes and values, whereas the expected increse in lifetime earnings due to education is more of a sociological impersonal factor.

2.4.

The marginal effect of x on y in general form is given by

$$\begin{aligned}\frac{\partial \mathbf{P}[y_i = 1|\mathbf{x_i}]}{\partial x_{ij}} &= \frac{\partial F(\mathbf{x_i}'\beta)}{\partial x_{ij}} \\ &= F'(\mathbf{x_i}'\beta)\beta_j \\ &= f(\mathbf{x_i}'\beta)\beta_j\end{aligned}$$

i.e. the density function times the inner derivative

For LPM the marginal effect is simply

$$\frac{\partial(\mathbf{x_i}'\beta + \varepsilon_i)}{\partial x_{ij}} = \beta_j$$

Probit

$$\frac{\partial \Phi(\mathbf{x_i}'\beta)}{\partial x_{ij}} = \phi(\mathbf{x_i}'\beta)\beta_j$$

Logit with $\Lambda(\mathbf{x_i}'\beta) = \frac{e^{\mathbf{x_i}'\beta}}{1+e^{\mathbf{x_i}'\beta}}$

$$\frac{\partial \Lambda(\mathbf{x_i}'\beta)}{\partial x_{ij}} = \Lambda(\mathbf{x_i}'\beta)[1 - \Lambda(\mathbf{x_i}'\beta)]\beta_j$$

The mean marginal effect of a model can be calculated in several ways in the Monte Carlo framework. All of the following examples are for the Probit model.

The easiest way is just to take the average of estimated coefficients and use them to calculate the marginal effect at $\bar{x}$.

$$\phi(\bar{\mathbf{x}}_\mathbf{s}'\hat{\beta})\hat{\beta}_j$$

Here j refers to the index of the coefficient of j^{th} explanatory variable x_j, for which the mean marginal effect is being calculated for.

Another way is to calculate the marginal effects separately for each S iterations (with different estimated coefficients) and then taking the average of the marginal effects.

$$\frac{1}{S}\sum_{s=1}^{S} \phi(\bar{\mathbf{x}}_\mathbf{s}'\hat{\beta}_\mathbf{s})\hat{\beta}_{j,s}$$

These procedures will give different results, but the difference isn't large.

The third way suggested by Cameron&Trivedi is more laborous: For each of the S iteration, calculate the marginal effect for each of the sampled x and take the mean of those. Finally, calculate the mean of the S estimates.

For Probit this would mean

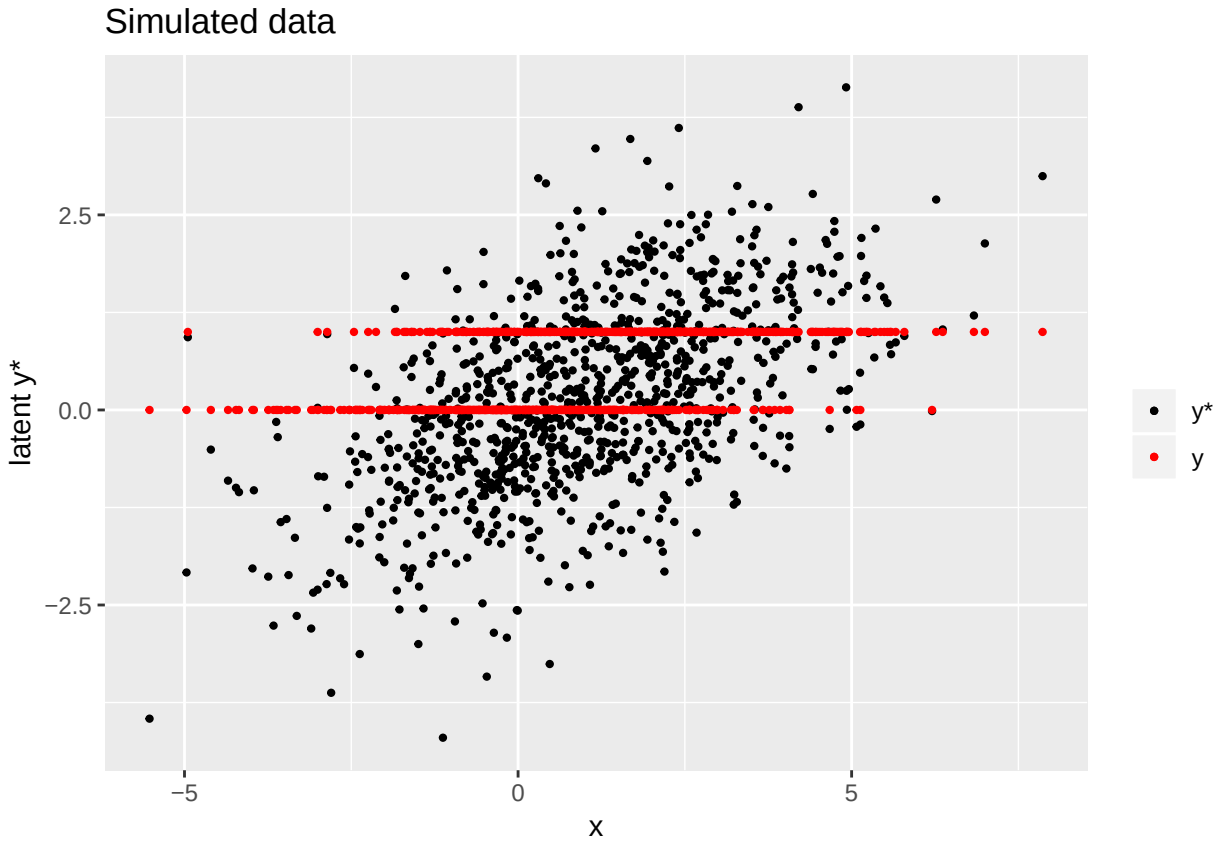
$$\frac{1}{S}\sum_{s=1}^{S}\left[\frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{x}'_{\mathbf{i,s}}\hat{\beta}_{\mathbf{s}})\hat{\beta}_{j,s}\right]$$

I will demonstrate all of the three techniques in practice shortly.

2.5.

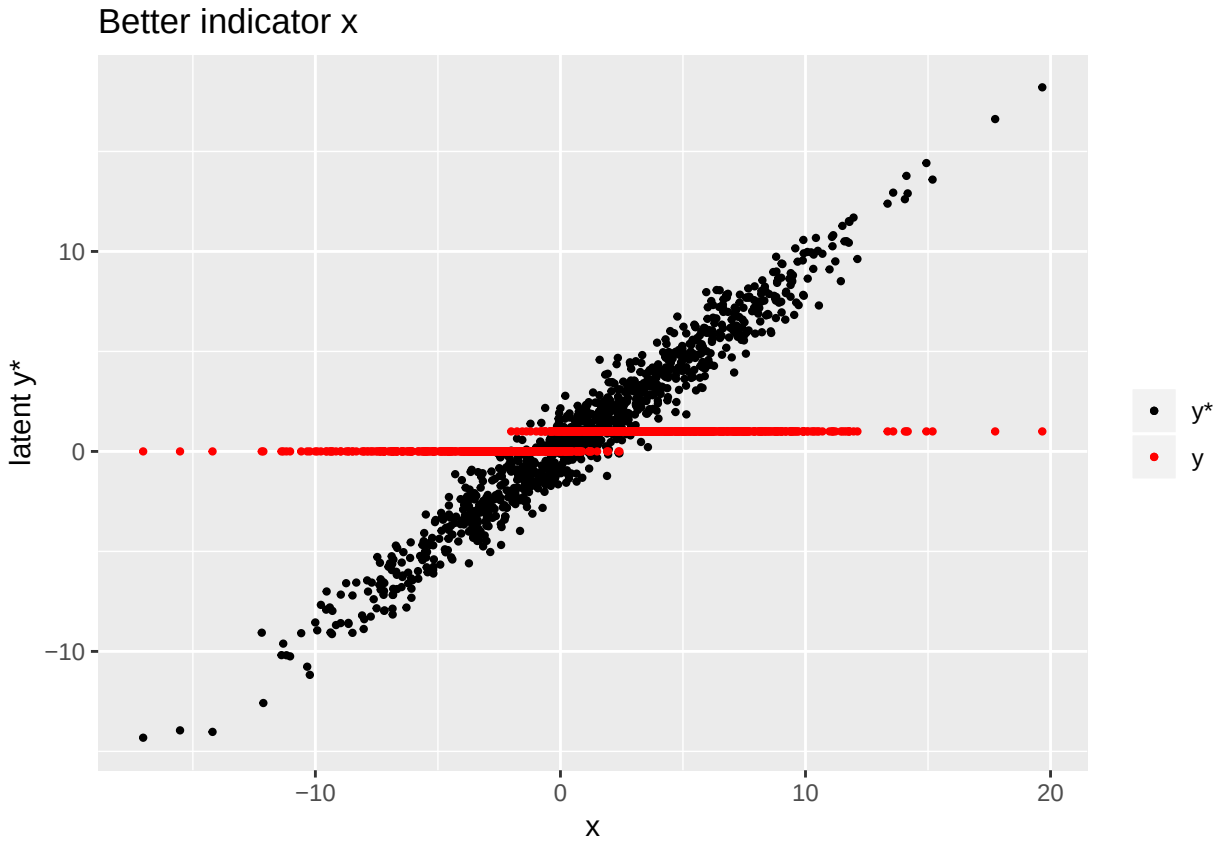
Simulate data from the model

```
simulate_data <- function(n, beta=c(-0.25, 0.35), var_x=2, sd){
      x <- cbind(1, rnorm(n, 1, var_x))
      epsilon <- rnorm(n,0,sd)
      y_latent <- x %*% beta + epsilon
      y <- ifelse(y_latent < 0, 0, 1)
      data <- cbind(y, x[,2], y_latent)

      return(data)
}

data <- as.data.frame(simulate_data(n=1000, sd=1))

ggplot(data=data, aes(x=data[,2], y=data[,3], color="black")) +
    geom_point(size=0.8) +
    geom_point(data = data, aes(y = data[,1], colour = 'red'), size = 0.8) +
    labs(title="Simulated data", x ="x", y = "latent y*") +
    scale_colour_manual(name = '', values =c('black'='black','red'='red'), labels = c('y*','y'))
```



In contrast, if the relation between x and y* was stronger; β_1 was higher, and x was was more dispersed, without the error term having too large of a variance σ^2, the signaling quality of x_i would be much greater:

```
data <- as.data.frame(simulate_data(n=1000, beta=c(0,0.9), var_x=5, sd=1))

ggplot(data=data, aes(x=data[,2], y=data[,3], color="black")) +
    geom_point(size=0.8) +
    geom_point(data = data, aes(y = data[,1], colour = 'red'), size = 0.8) +
    labs(title="Better indicator x", x ="x", y = "latent y*") +
    scale_colour_manual(name = '', values =c('black'='black','red'='red'), labels = c('y*','y'))
```



3

3.1 $\varepsilon_i \sim \mathbb{N}(0, 1)$

3.1.2 Monte Carlo: Create 100 sets of 1000 observations and estimate LPM, Probit and Logit models for each of them.

Specifying the error term in this manner means, that the real dgp is a probit model. Therefore we can expect it to perform better than alternatives also in the estimation.

The following algorithm simulates a set of data and estimates LPM and Probit coefficients for it.

```
monte_carlo <- function(n_draws=1000, n_simulations=100, sigma=1, counter=0){

    # Lists to save results
    xs <- list()
    ys <- list()
    lpms <- list()
    probits <- list()
    logits <- list()

    values <- list()

    probit_avg_marg_simple <- c()
    probit_avg_marginals <- c()
    real_marginals <- c()
    probit_marginals <- c()
```

```
logit_marginals <- c()
logit_avg_marginals <- c()

# Perform the simulations and save results
for(i in 1:n_simulations){
    set.seed(1917+counter)

    # SIMULATE DATA
    data <- as_tibble(simulate_data(n=n_draws, sd=sigma))
    y <- data[,1]
    x <- data[,2]
    colnames(data) <- c("y", "x")

    # ESTIMATE MODELS

    # Linear probability model (OLS)
    model_lpm <- lm(y ~ x, data=data)
    coefs_lpm <- model_lpm$coefficients
    values_lpm <- model_lpm$fitted.values

    # Estimate Probit
    model_probit <- glm(y ~ x, family = binomial(link = "probit"),
                        data = data)
    coefs_probit <- model_probit$coefficients
    values_probit <- model_probit$fitted.values
    probit_avg_marginals[[i]] <- 1/n_draws * sum(dnorm(values_probit)*coefs_probit[[2]]) # average
    probit_avg_marg_simple[[i]] <- dnorm(coefs_probit[[1]] +
                                         coefs_probit[[2]])*coefs_probit[[2]] # mean

    # Estimate Logit
    model_logit <- glm(y ~ x, family = binomial(link = "logit"),
                       data = data)
    coefs_logit <- model_logit$coefficients
    values_logit <- model_logit$fitted.values
    gamma <- exp(values_logit)/(1 + exp(values_logit))
    logit_avg_marginals[[i]] <-  1/n_draws *sum(gamma * (1 - gamma) * coefs_logit[[2]])


    # Save one batch of actual and fitted values for plotting purposes.
    if(i==1){
        values <- cbind(y, x, values_lpm, values_probit, values_logit)
        colnames(values) <- c("y", "x", "lpm_fit", "probit_fit", "logit_fit")
        #real_marginals <- dnorm(-0.25 + 0.35*x) * 0.35 # Real marginal effects at each x.
        probit_marginals <- dnorm(values_probit) * coefs_probit[[2]] # all
        logit_marginals <-  gamma * (1 - gamma) * coefs_logit[[2]]
    }

    # Save the coefficients each round
    lpms[[i]] <- c(coefs_lpm)
    probits[[i]] <- c(coefs_probit)
    logits[[i]] <- c(coefs_logit)
    xs[[i]] <- x
    ys[[i]] <- y
```

```
        counter <- counter + 1 # for the seed
    }


    # Save all coefficients
    lpms <- do.call(rbind,lapply(lpms,matrix,ncol=2,byrow=TRUE))
    probits <- do.call(rbind,lapply(probits,matrix,ncol=2,byrow=TRUE))
    logits <- do.call(rbind,lapply(logits,matrix,ncol=2,byrow=TRUE))

    all_coefficients <- cbind(lpms, probits, logits) # Intercept and one coefficient in each

    # Find mean coefficients
    lpms_mean <- round(apply(X=lpms, MARGIN=2, FUN=mean), 2)
    probit_mean <- round(apply(X=probits, MARGIN=2, FUN=mean), 2)
    logit_mean <- round(apply(X=logits, MARGIN=2, FUN=mean), 2)

    avg_coefficients <- cbind(c(-0.25, 0.35),lpms_mean, probit_mean, logit_mean)
    colnames(avg_coefficients) <- c("Real", "LPM", "Probit", "Logit")
    rownames(avg_coefficients) <- c("Intercept", "Beta 1")

    return(list(avg_coeffs=avg_coefficients, fitted_values=values,
                all_coeffs=all_coefficients,
                marginals = list(probit=probit_avg_marginals, probit_alt=probit_avg_marg_simple, logit=l
                all_marginals=list(probit=probit_marginals, logit=logit_marginals, real=real_marginals)
}
```

3.1.3.

Let's run the MC-algorithm 100 times, each time drawing 1000 sets of data including x, y* and y. For each dataset, the algorithm estimates LPM, Probit and Logit models and saves the results. The counter will guarantee, that each dataset is different, but reproducible.

The average values for the coefficients for all methods are displayed in the following table, accompanied by the real parameter values of the dgp.

```
mc_results <- monte_carlo(n_draws=1000, n_simulations=100, sigma=1)
```

```
## Warning: `as_tibble.matrix()` requires a matrix with column names or a `.name_repair` argument. Using
## This warning is displayed once per session.
```

```
avg_coeffs <- mc_results$avg_coeffs
lpm_marginal <- avg_coeffs[2,2]
avg_coeffs
```

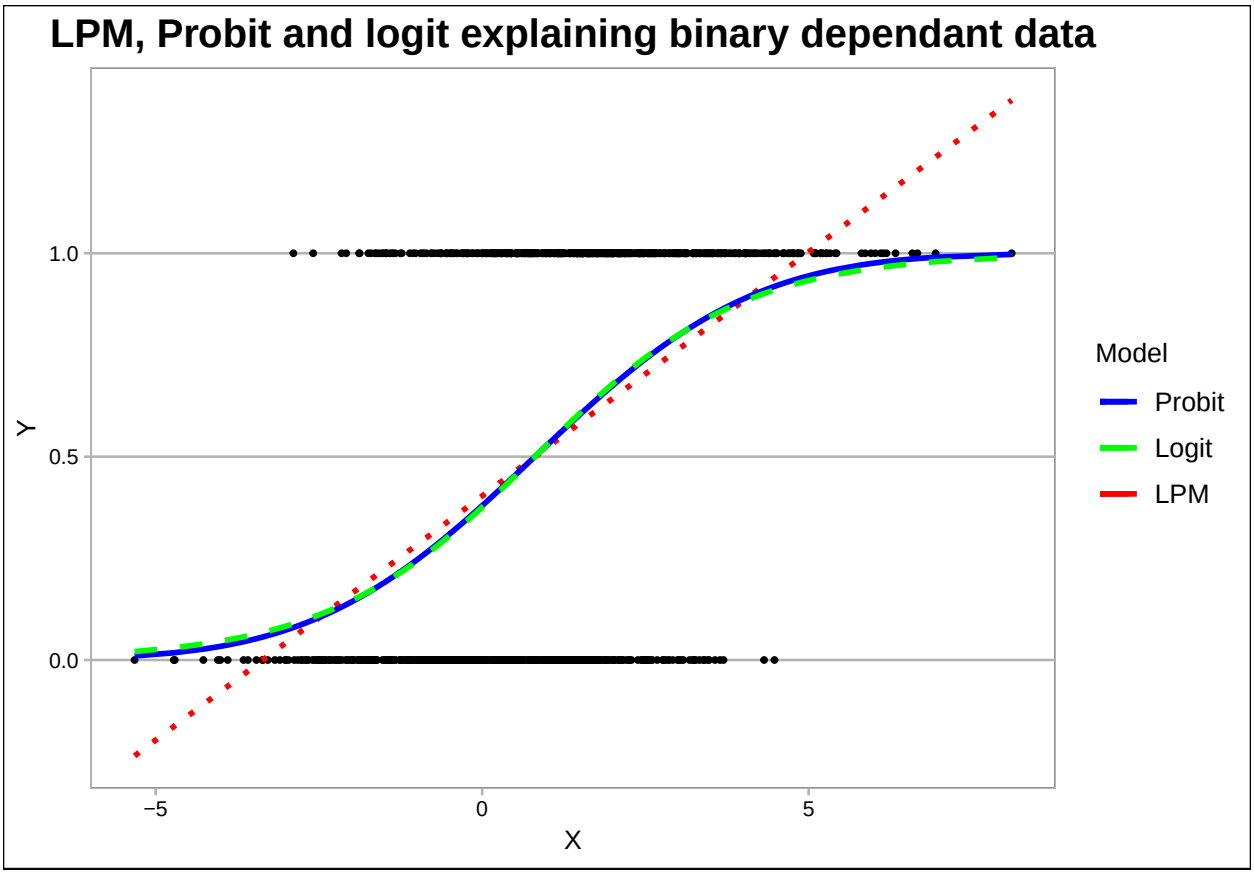
```
##           Real  LPM Probit Logit
## Intercept -0.25 0.42  -0.24 -0.41
## Beta 1     0.35 0.11   0.35  0.58
```

LPM estimates seem pretty badly off whereas Probit is really accurate, as was expected. Logit perform better than LPM, but is not too close to the real values.

Let's visualize the fitted values of the models for p.

```
# Draw fitted values for the plot
fitted_values <- mc_results$fitted_values
```

```r
fitted_values <- arrange(fitted_values, x)
colnames(fitted_values) <- c("y", "x", "lpm_fit", "probit_fit", "logit_fit")

ggplot(fitted_values, aes(x, y)) +
    geom_point(size=0.7) +
    geom_line(data = fitted_values, aes(y = lpm_fit, color = 'red'), size = 1, linetype=3) +
    geom_line(data = fitted_values, aes(y = probit_fit, color = 'blue'), size = 1) +
    geom_line(data = fitted_values, aes(y = logit_fit, color = 'green'), size = 1, linetype=2) +
    theme_calc() +
    labs(title="LPM, Probit and logit explaining binary dependant data", x ="X", y = "Y") +
    theme(plot.title = element_text(color="black", size=15, face="bold", hjust = 0.5)) +
    scale_colour_manual(name = 'Model', values =c('red'='red','blue'='blue', 'green'='green'),
                        labels = c('Probit','Logit', 'LPM'))
```

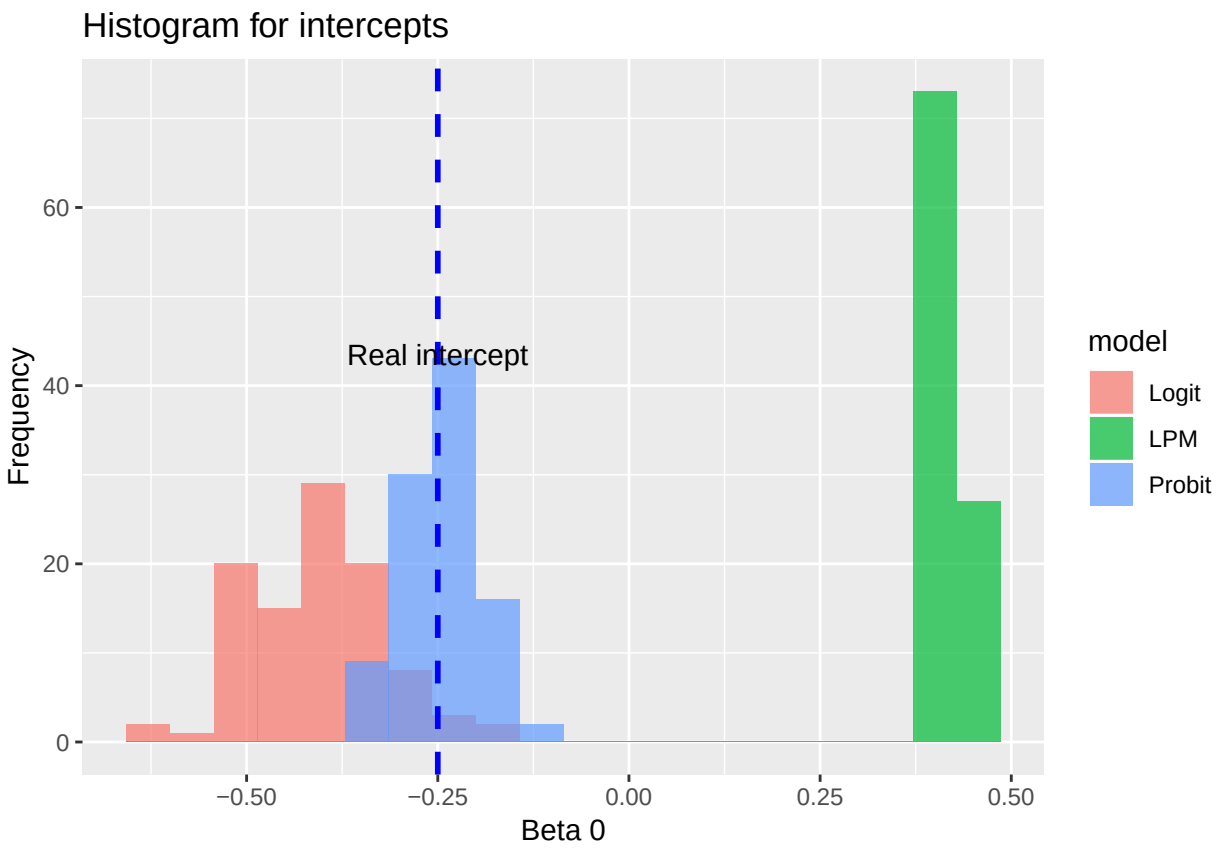


This reveals, that LPM approximates the true model quite well in the mid-range of x, but can't obviously handle the non-linear tails. Here Probit and Logit seem to perform almost identically, despite having identified different parameters.

Probit model seems consistent, whereas the LPM systemically overestimates the intercept and underestimates teh slope coefficient. This is hardly surprising, though, as the DGP was non-linear with respect to p to start with.

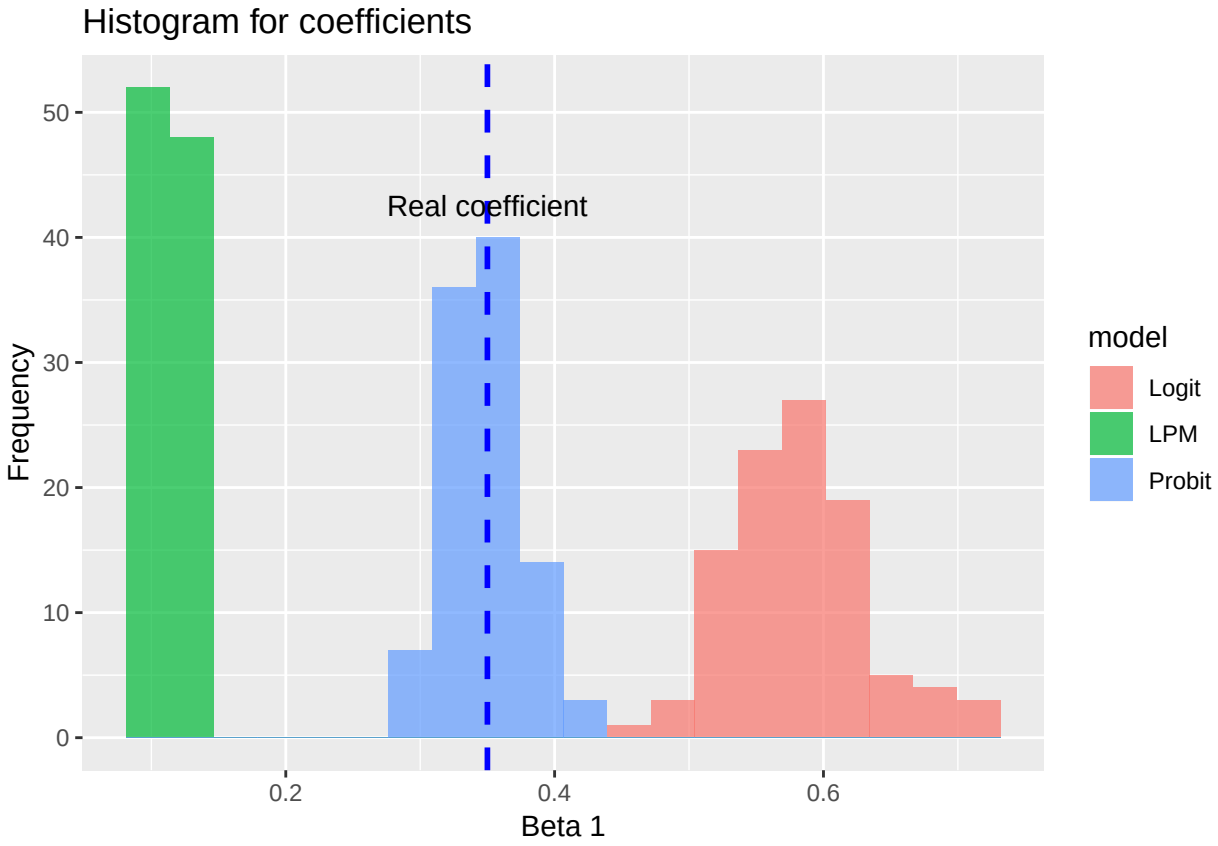
The average fit only gives a snapshot of the estimation process. Next I visualize the distribution of values for coefficients over the simulated runs on histograms, to see how similar results the models produced in different runs.

```r
# Draw fitted values for the plot
all_coefficients <- as.data.frame(mc_results$all_coeffs)

intercepts <- data.frame(values = c(all_coefficients[,1],all_coefficients[,3],all_coefficients[,5]),
                         model = rep(c("LPM", "Probit", "Logit"),each = length(all_coefficients[,1])))

coefficients <- data.frame(values = c(all_coefficients[,2],all_coefficients[,4],all_coefficients[,6]),
                           model = rep(c("LPM", "Probit", "Logit"),each = length(all_coefficients[,2])))


ggplot(intercepts, aes(x = values, fill = model)) +
    geom_histogram(alpha=0.7, position = 'identity',bins=bins) +
    geom_vline(aes(xintercept=-0.25), color="blue", linetype="dashed", size=1) +
    labs(title="Histogram for intercepts", x ="Beta 0", y = "Frequency") +
    annotate(geom="text", label="Real intercept", x=-0.25, y=40, vjust=-1)
```



Probit is the only consistent model here, whit the distribution centered around the real value for intercept. It also has the tightest distribution, referring to efficiency.

```r
ggplot(coefficients, aes(x = values, fill = model)) +
    geom_histogram(alpha=0.7, position = 'identity',bins=bins) +
    geom_vline(aes(xintercept=0.35), color="blue", linetype="dashed", size=1) +
    labs(title="Histogram for coefficients", x ="Beta 1", y = "Frequency") +
    annotate(geom="text", label="Real coefficient", x=0.35, y=40, vjust=-1)
```



With the slope coefficient the difference in quality is even more in favor of Probit.

3.1.4. Marginal effects

True marginal effect

The true marginal effect can be calculated from the underlying probit model, by evaluating the marginal effect at the sample average of x using the true parameter values. As the dgp has $x_i \sim N(1, 2)$, the average is simply $\bar{x} = 1$. Probit model yields (the formula for marginal effect was derived earlier)

```
real_me <- dnorm(-0.25 + 0.35) * 0.35
real_me
```

```
## [1] 0.1389334
```

$$\phi(\mathbf{x}_i'\beta)\beta_j = \phi(-0.25 + 1 * 0.35)0.35 = 0.1389$$

The interpretation for the number is, that a unit increase in x increases leads to ..?

Estimated marginal effects

As noted in part 2, the marginal effects of LPM is plainly given by the coefficient $\beta_1 = 0.11$. Hence the above histogram of estimated coefficient values for LPM also depicts its distribution of marginal effects.

For the estimated Probit model, I calculate the marginal effect in three different ways, just for fun.

First techinque: I calculate marginal effects for each of the N=1000 $x_i's$ in a simulation round and find the mean of that round: $\frac{1}{N}\sum_{i=1}^{N} F'(\mathbf{x'_{i,s}}\beta_{\mathbf{s}})\beta_{j,s} = \hat{ME}_s$. I then calculate the average marginal effect among the results from each S=100 rounds. In other words, marginal impact of each sampled $x_{i,s} \forall i, s$ on p, summed up and divided by the sample size:

```
probit_marginal_1 <- as.numeric(unlist(mc_results$marginals$probit))

mean(probit_marginal_1)
```

```
## [1] 0.1182047
```

Hence,

$$\frac{1}{S}\sum_{s=1}^{S}\left[\frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{x'_{i,s}}\beta_{\mathbf{s}})\beta_{j,s}\right] = 0.1182$$

which is closer to the real value than that of lpm. However, it's not as close as one would expect from a model which mirrors the dgp. The reason for that might derive from the technique used in the estimation, which should converge in the limit. I find evidence for this later on in part 3.1.7. This technique is the most laborous, demanding more than $100 * 1000 = 100000$ calculations in this case.

Secondly, the average marginal effect is calculated at the mean of x in each of the $S = 100$ samples, using the coefficients estimated on that round. The mean marginal effect of the model is then the mean of those sample estimates.

```
probit_marginal_alt <- as.numeric(unlist(mc_results$marginals$probit_alt))
mean(probit_marginal_alt)
```

```
## [1] 0.1381093
```

$$\frac{1}{S}\sum_{s=1}^{S}\phi(\bar{\mathbf{x}}'_{\mathbf{s}}\hat{\beta}_{\mathbf{s}})\hat{\beta}_{j,s} = 0.1381$$

The result is almost identical to the real marginal effect. I will use the estimates calculated with this technique in the plot, although Cameron&Trivedi prefers the first one. This technique also only required 100 calculations.

Finally, the easiest way is to just use the final coefficient estimates to calculate the mean marginal effect

```
avg_coeffs[2,3]*dnorm(avg_coeffs[1,3] + avg_coeffs[2,3])
```

```
## [1] 0.1387876
```

$$\phi(\bar{\mathbf{x}}'\hat{\beta})\hat{\beta}_j = 0.1388$$

This method only involves one additional calculation on top of the simulation procedure, and the estimate is really close to the one calculated using the second technique.

Marginal effects of the logit model is given by

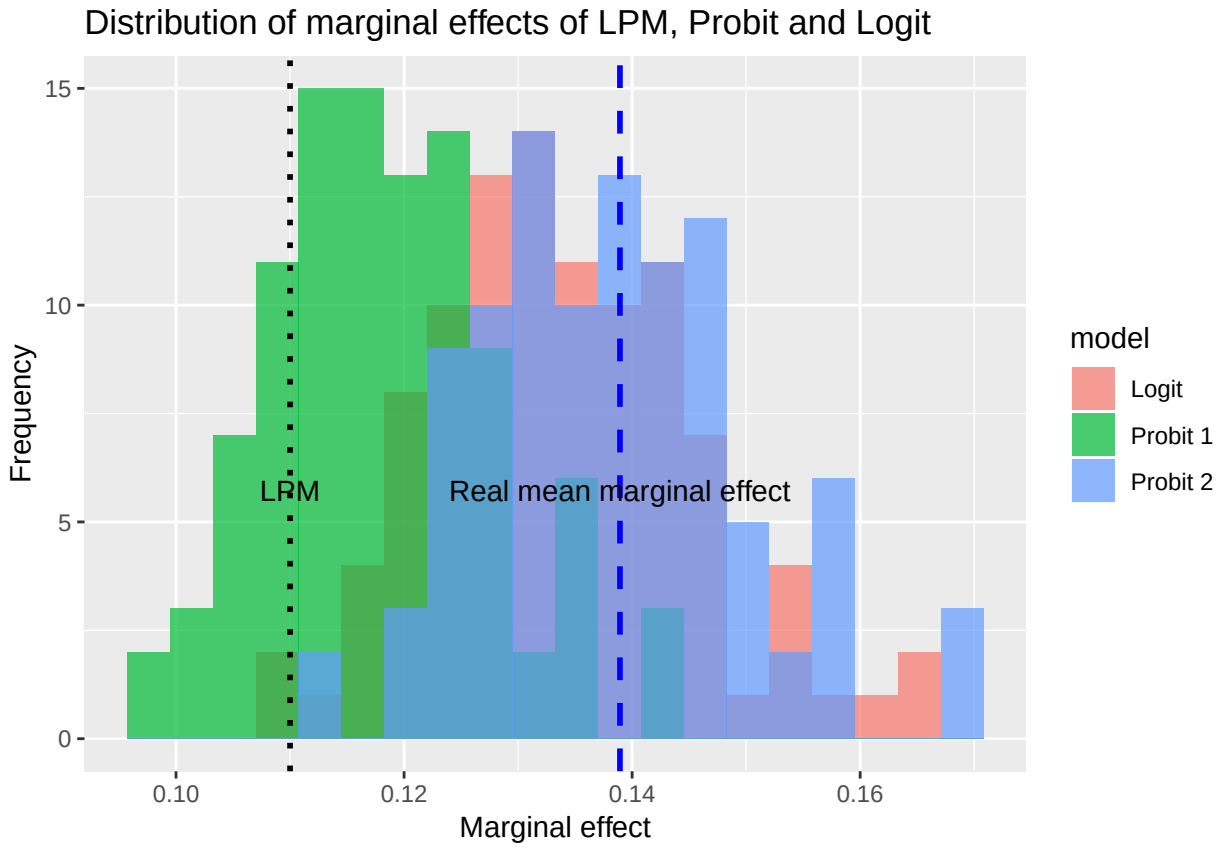
```
logit_marginal <- as.numeric(unlist(mc_results$marginals$logit))
mean(logit_marginal)
```

```
## [1] 0.1337806
```

which is quite close as well.

Plotting the marginal effects including both techniques for Probit, but only using the mean for LPM to simplify the plot

```
all_marginals <- data.frame(values = c(probit_marginal_1, probit_marginal_alt, logit_marginal),
                            model = rep(c("Probit 1", "Probit 2", "Logit"), each = length(probit_margina

ggplot(all_marginals, aes(x = values, fill = model)) +
    geom_histogram(alpha=0.7, position = 'identity', bins=bins) +
    labs(title="Distribution of marginal effects of LPM, Probit and Logit", x ="Marginal effect", y = "
    geom_vline(aes(xintercept=real_me), color="blue", linetype="dashed", size=1) +
    annotate(geom="text", label="Real mean marginal effect", x=real_me, y=5, vjust=-1) +
    geom_vline(aes(xintercept=0.11), color="black", linetype=3, size=1) +
    annotate(geom="text", label="LPM", x=0.11, y=5, vjust=-1)
```



Probit is the anticipated king here, although it is noteworthy that there is a quite substantial spread in the distribution. This means, that the marginal effects are rather sensitive to the particular dataset used, even if there was a thin tailed distribution and 1000 observations (which are something unheard of, for a macro-minded person). Logit is also performing well, even better than the Probit calculated from the sample averages of marginal effects.

3.1.5. Biases

Coefficients

Absolute sizes of the biases $\hat{\beta} - \beta$ are

```
avg_coeffs[,2:4] - avg_coeffs[,1]
```

```
##             LPM Probit Logit
## Intercept  0.67   0.01 -0.16
## Beta 1    -0.24   0.00  0.23
```

and relative (%)

```
round((avg_coeffs[,2:4] - avg_coeffs[,1])/avg_coeffs[,2:4], 2)
```

```
##             LPM Probit Logit
## Intercept  1.60  -0.04  0.39
## Beta 1    -2.18   0.00  0.40
```

No surprises here.

Marginal effects

Absolute sizes of the biases $\hat{ME} - ME$ are

```
cat("LPM: ", 0.11 - real_me, " , Probit: ", probit_marginal_alt - real_me)
```

```
## LPM:  -0.02893339  , Probit:  0.01230868 -0.004795238 -0.008604746 0.008243698 -0.007576046 0.002973
```

and relative (%)

```
cat("LPM: ", (0.11-real_me)/0.11, " % , Probit: ", (probit_marginal_alt-real_me)/probit_marginal_alt, "
```

```
## LPM:  -0.2630308  % , Probit:  0.08138395 -0.0357485 -0.06602344 0.0560121 -0.05767508 0.02095148 0.
```

3.1.6. Differences between the coefficients

Probit model performs well, as it has the same structure as the dgp. Additional to that, the data only exerts mild randomness (normality) and there are plenty of observations for the LLN and CLT to kick in.

Contrary to that, the parameters in LPM are not consistent with those of the dgp. Hence, it is not equipped to learn the correct representation to explain the data.

3.1.7. Create another dataset and do everything all over again

a) Coefficients

```
mc_results <- monte_carlo(n_draws=1000, n_simulations=1, sigma=1, counter=101) # Counter issues new pro
avg_coeffs <- mc_results$avg_coeffs
avg_coeffs
```

```
##            Real  LPM Probit Logit
## Intercept -0.25 0.41  -0.26 -0.43
## Beta 1     0.35 0.11   0.34  0.57
```

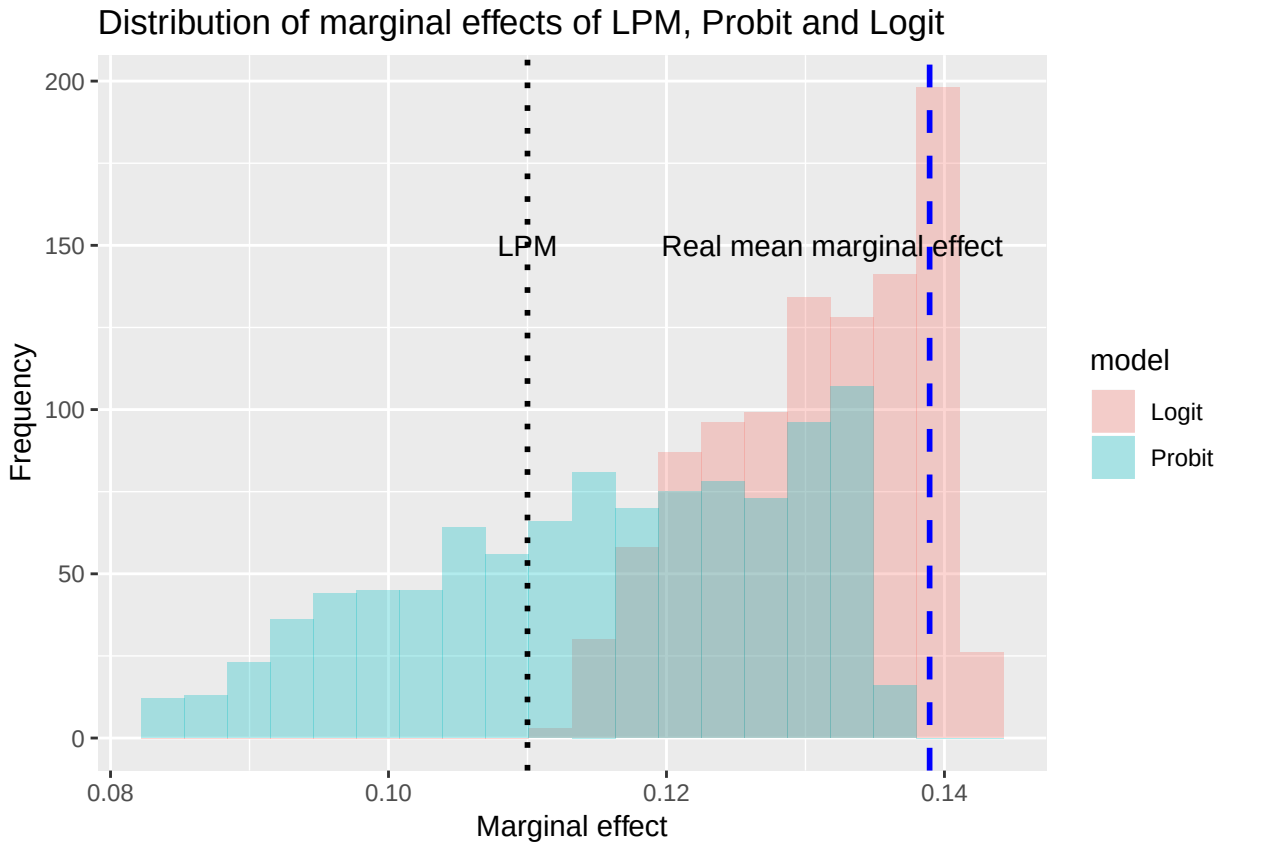
Quite similar results as previously.

b) Marginal effects

Marginal effect of LPM is the same for each observationx_i, and is given by $\beta_1 = 0.11$.

Let's create a histogram again to see the distribution of marginal effects in the Probit and Logit models. The real marginal value is the same as before, 0.1389, as we still have the same dgp.

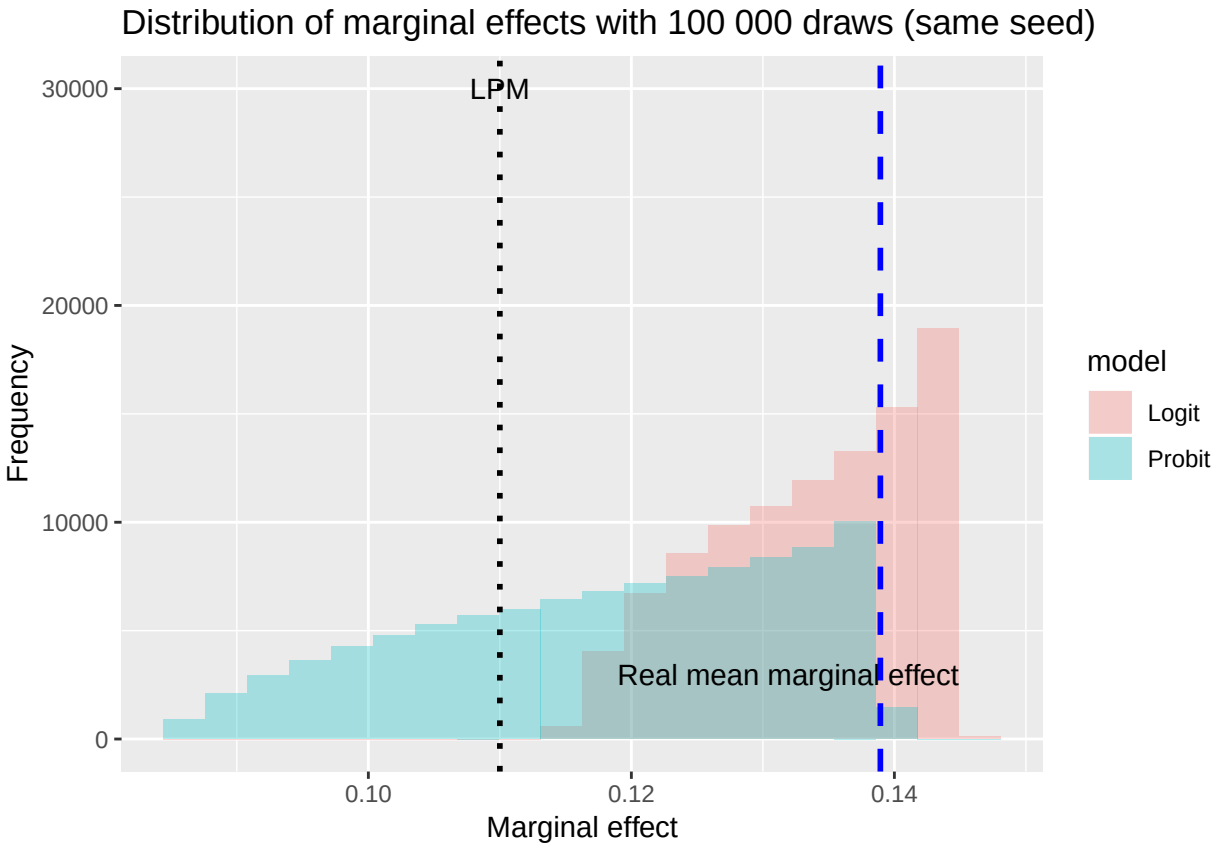
```
real_marginal <- mc_results$all_marginals$real
probit_marginal <- mc_results$all_marginals$probit
logit_marginal <- mc_results$all_marginals$logit

all_margins <- data.frame(values = c(probit_marginal, logit_marginal),
                          model = rep(c("Probit", "Logit"), each = length(probit_marginal)))

ggplot(all_margins, aes(x = values, fill = model)) +
    geom_histogram(alpha=0.3, position = 'identity', bins=bins) +
    labs(title="Distribution of marginal effects of LPM, Probit and Logit",
         x ="Marginal effect", y = "Frequency") +
    geom_vline(aes(xintercept=real_me), color="blue", linetype="dashed", size=1) +
    annotate(geom="text", label="Real mean marginal effect", x=(real_me-0.007), y=150) +
    geom_vline(aes(xintercept=0.11), color="black", linetype=3, size=1) +
    annotate(geom="text", label="LPM", x=0.11, y=150)
```



Here we see something interesting - the Logit model actually seems to perform better than Probit, by having the mode of the distribution of marginal effects at the same value where the real mean marginal effect is. This is not a fluke, I tested the result with different seeds and quite often Logit seemed to peak closer to the real value and more distinctly than Probit.

However, there are two things to consider. First, we don't see the distribution of the real marginal effects, which could be (and probably are) closer to the Probit model. Also, increasing the number of draws from 1000 to 100 000 changes the situation, as can be seen in the plot below, making the peak of the distribution from Probit model approach the real value. This suggests, that the marginal effects calculated in this fashion

do indeed converge to the real ones in the Probit model.

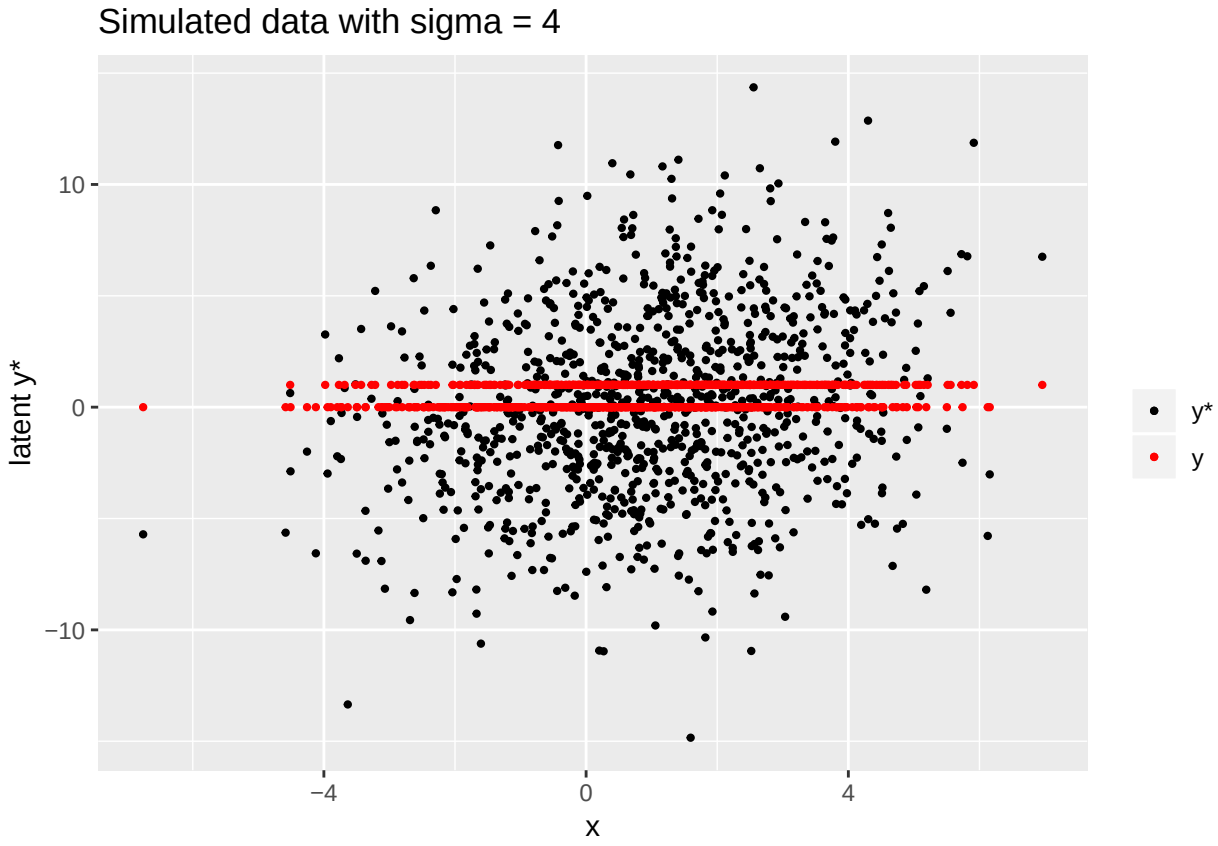


3.2 $\varepsilon \sim N(0, \sigma^2)$

3.2.1 Different variances

$\sigma^2 = 16$

The data denerated by this process is much more dirpersed than before.



Conducting a new MC-manouver

```
mc_results <- monte_carlo(n_draws=1000, n_simulations=100, sigma=4) # variance 16
average_coeffs_2 <- mc_results$avg_coeffs
average_coeffs_2
```

```
##             Real  LPM Probit Logit
## Intercept -0.25 0.48  -0.06 -0.09
## Beta 1     0.35 0.03   0.09  0.14
```

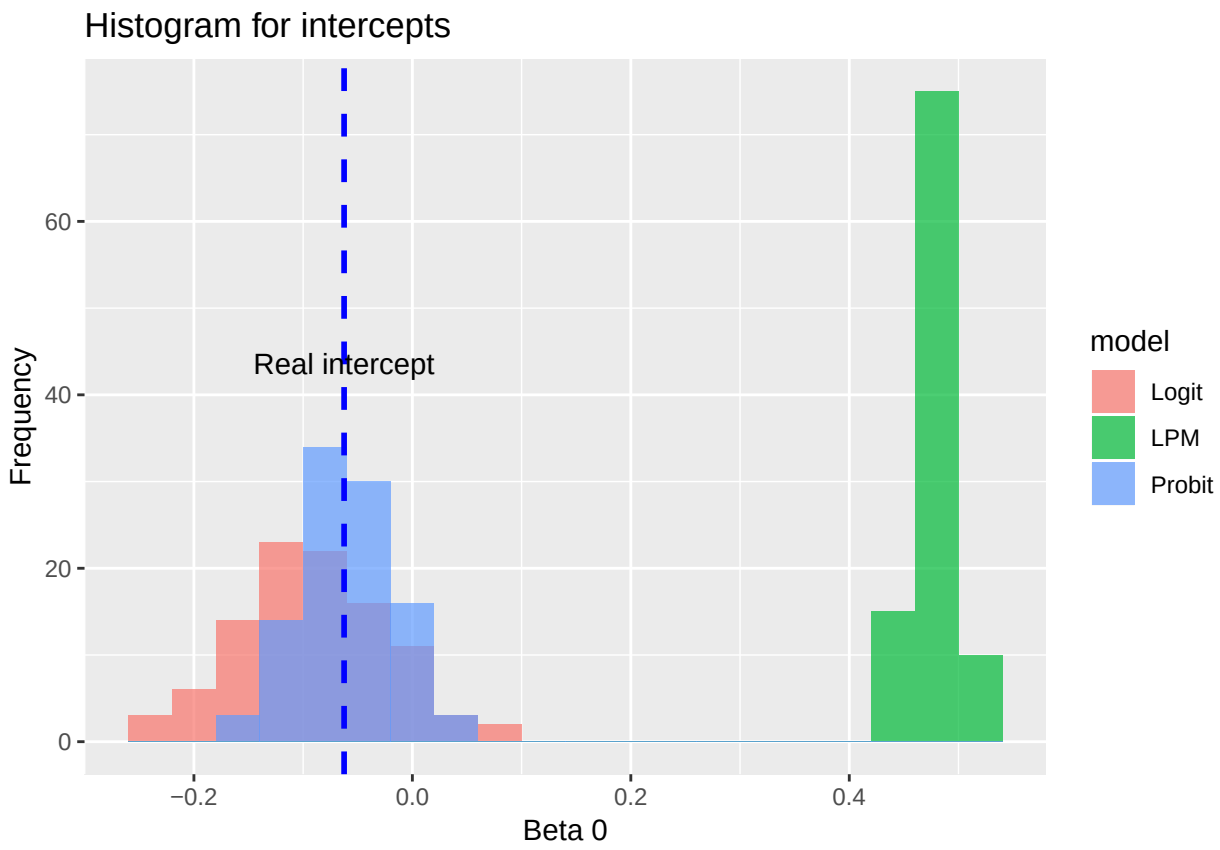
The Probit model now identifies $\frac{\beta_i}{\sigma}$, which are the real coefficients of the DGP. The Probit estimates are not anymore consistent for the β_0 and β_1 of the linear regression for latent y*.

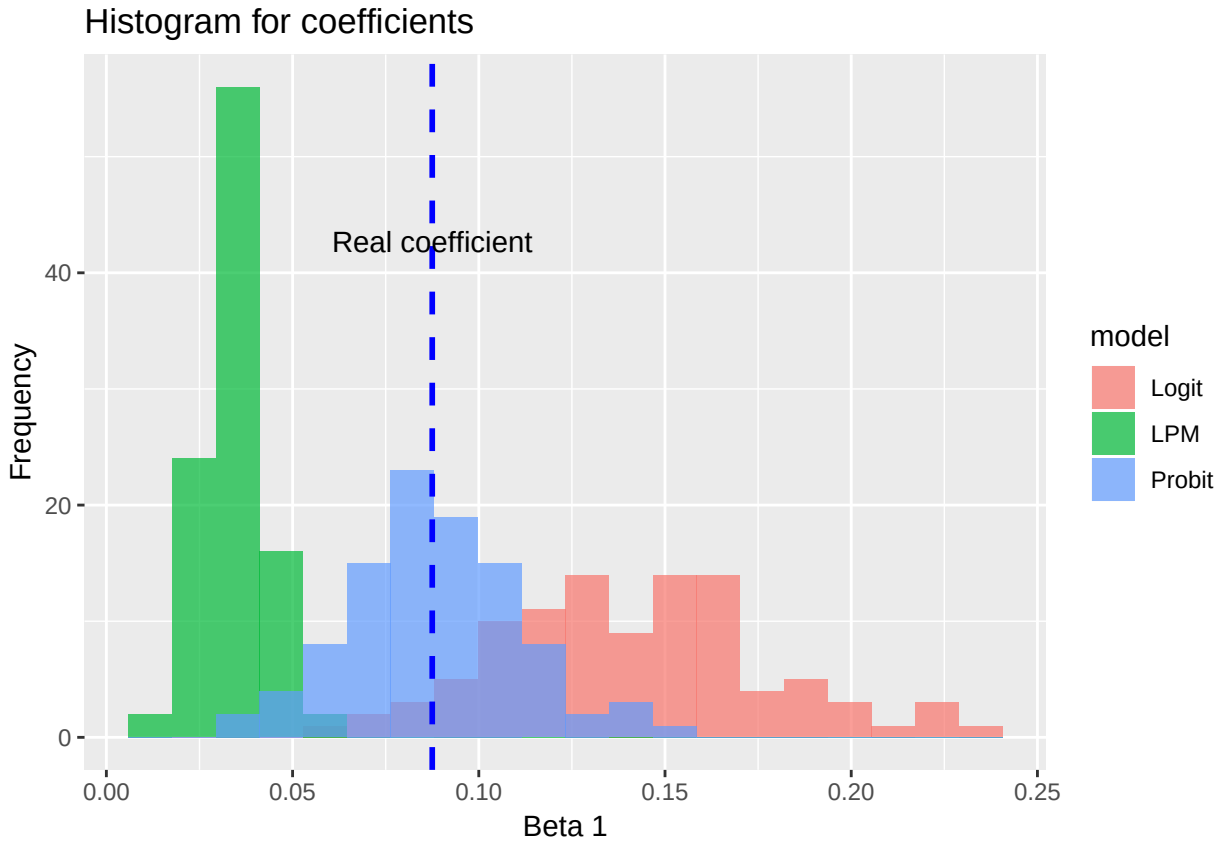
The new real coefficients of the dgp can be obtained by re-normalizing with σ.

```
beta0_real <- -0.25/4
beta1_real <- 0.35/4

cat("Real Intercept: ", beta0_real, " and coefficient: ", beta1_real)
```

```
## Real Intercept:  -0.0625  and coefficient:  0.0875
```

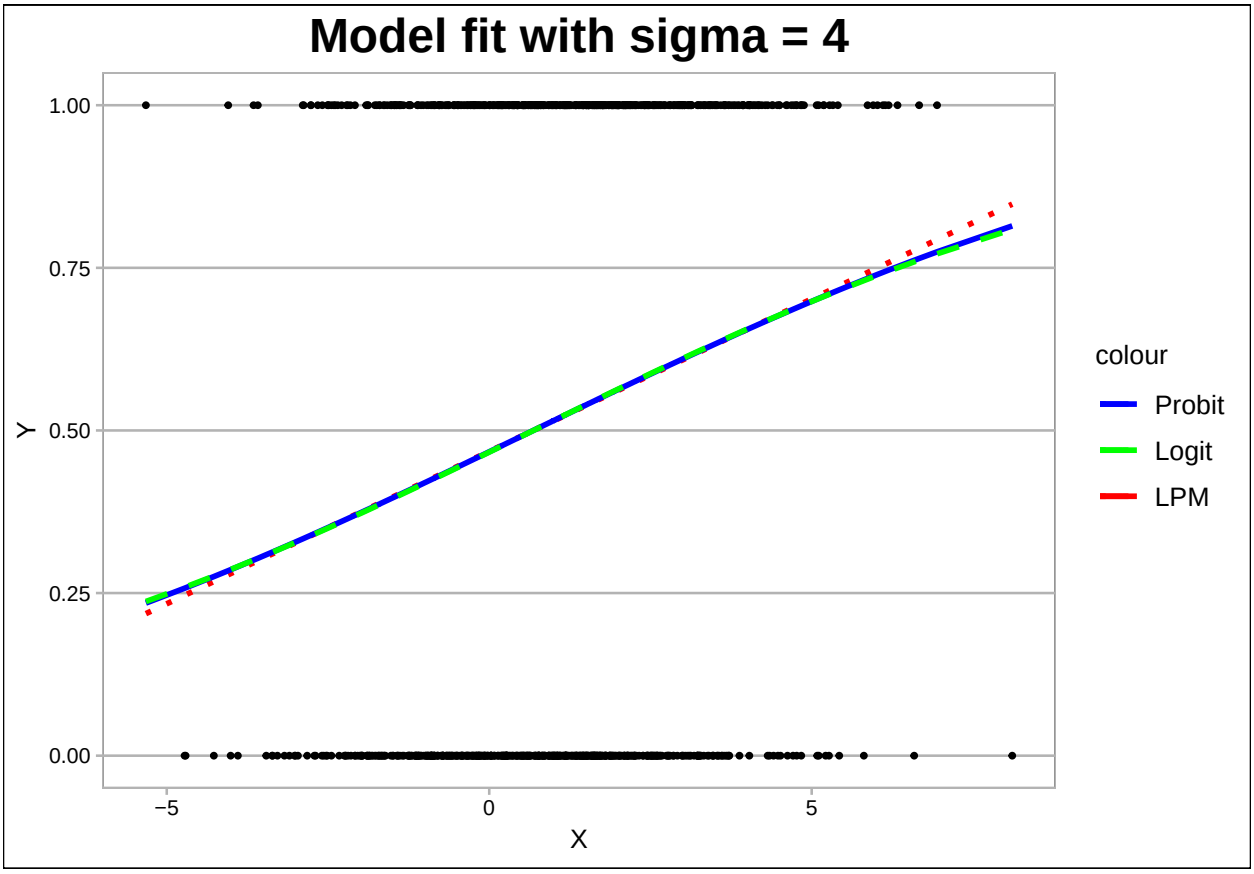
which are really close to the Probit estimates.


Histogram for intercepts
Real intercept
Frequency
Beta 0
model
Logit
LPM
Probit


Histogram for coefficients
Real coefficient
Frequency
Beta 1
model
Logit
LPM
Probit

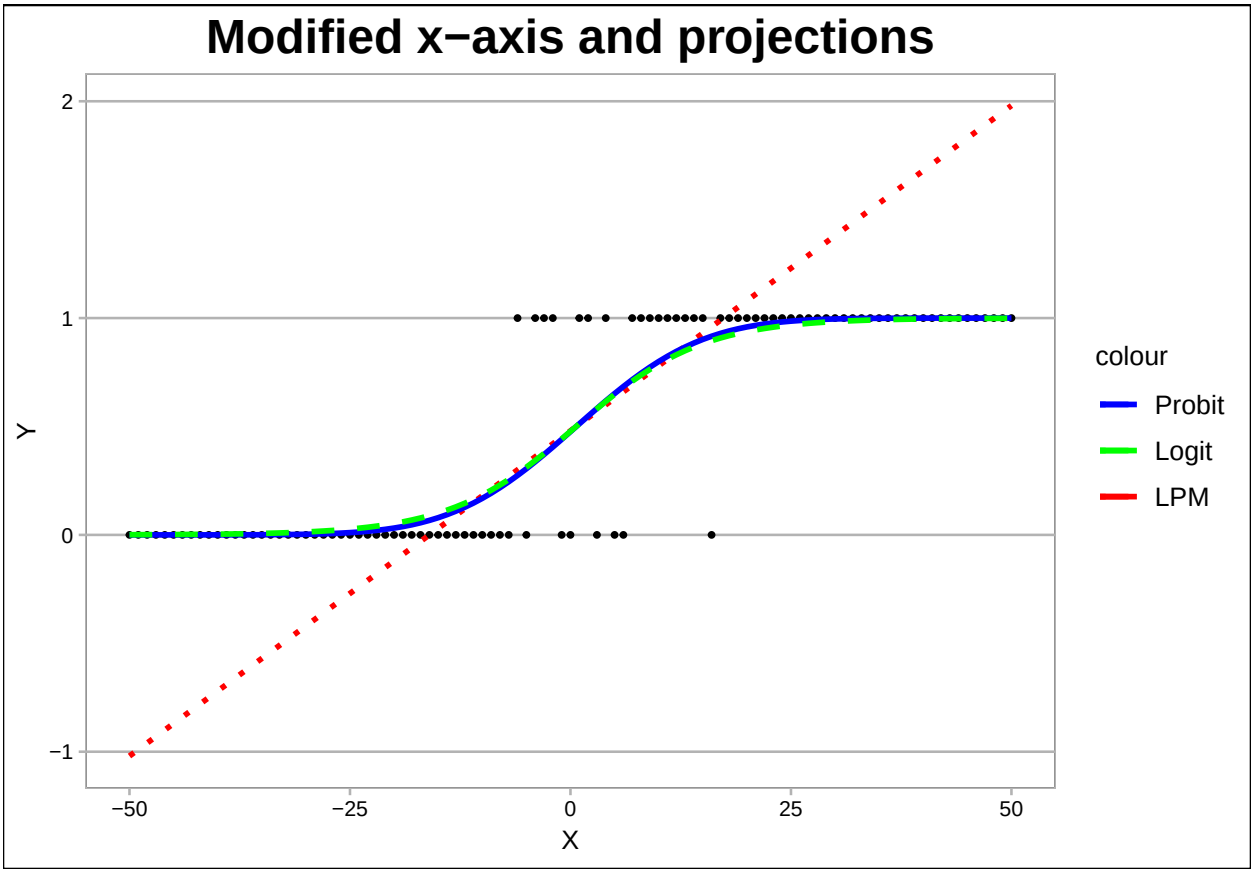
Pretty much the same story here as before.

Fitted values



This seems unfamiliar at first, as all of the models resemble linear regression on the support of x. Probit and logit models are much more flat here than before, because it pays off to issue predictions which are closer to $\bar{y} = 0.5$, as the signal of x is now diluted. The curves are not flat, however, as theunderlying dgp still expresses a relation between x and y, even though it is not as clear anymore close to the mean of x.

By projecting the model predictions for extended range of x-values we can see when the nonlinearity of Probit and Logit starts to show.



The Probit and Logit models indeed obtain their familiar shape when we "zoom out". In the far end of tail of the distribution of x, you can still be almost sure whether the accompanying y is 0 or 1.

Mean marginal effects

The real mean marginal effect can be obtained, by using estimated parameters which are re-normalized by σ.

```
real_me2 <- 0.35/4*dnorm(-0.25/4 + 0.35/4)
cat("Real mean marginal effect: ", real_me2)
```

```
## Real mean marginal effect:  0.03489654
```

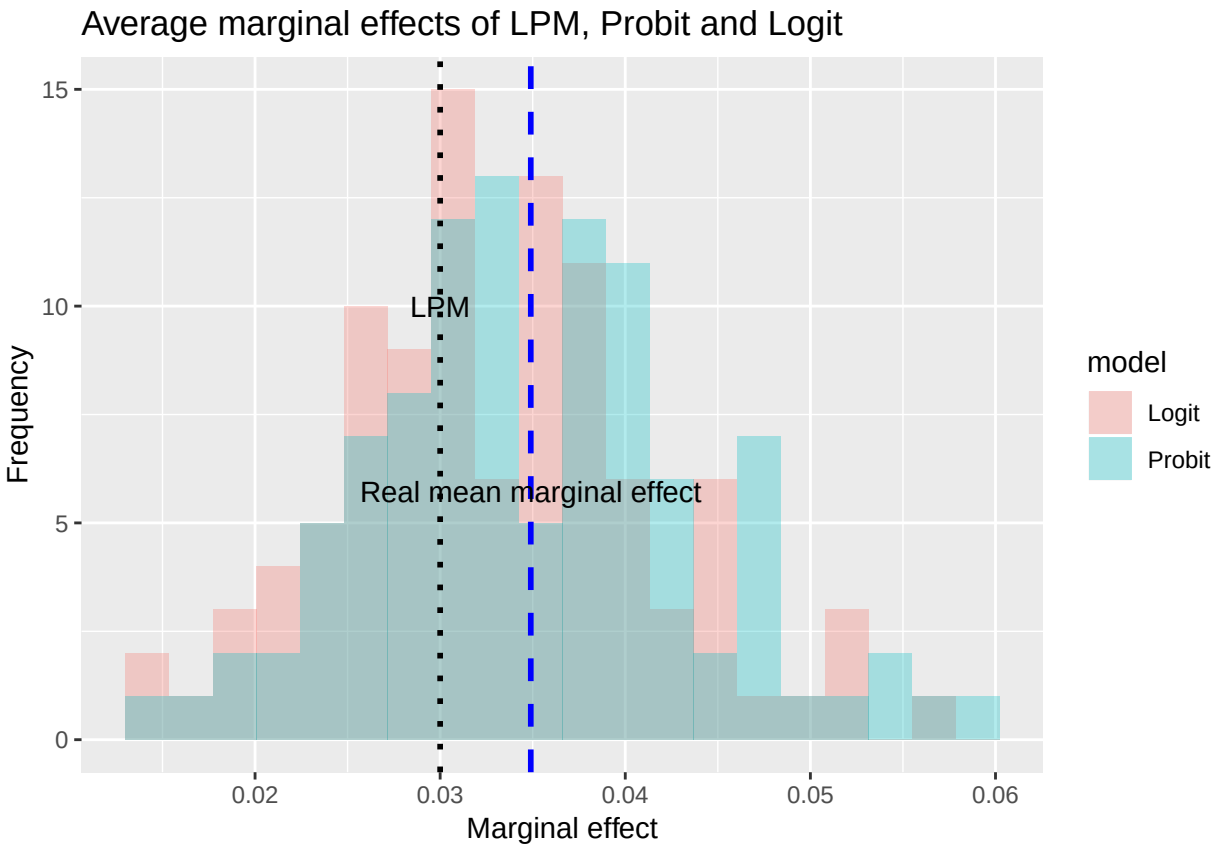
The mean marginal effect of LPM is given by $\beta_1 = 0.03$ as before.

The Probit and Logit values are the averages of the simple mean marginal effects in 100 samples (second technique presented above).

```
## Probit:  0.0350383  , Logit:  0.03299298
```

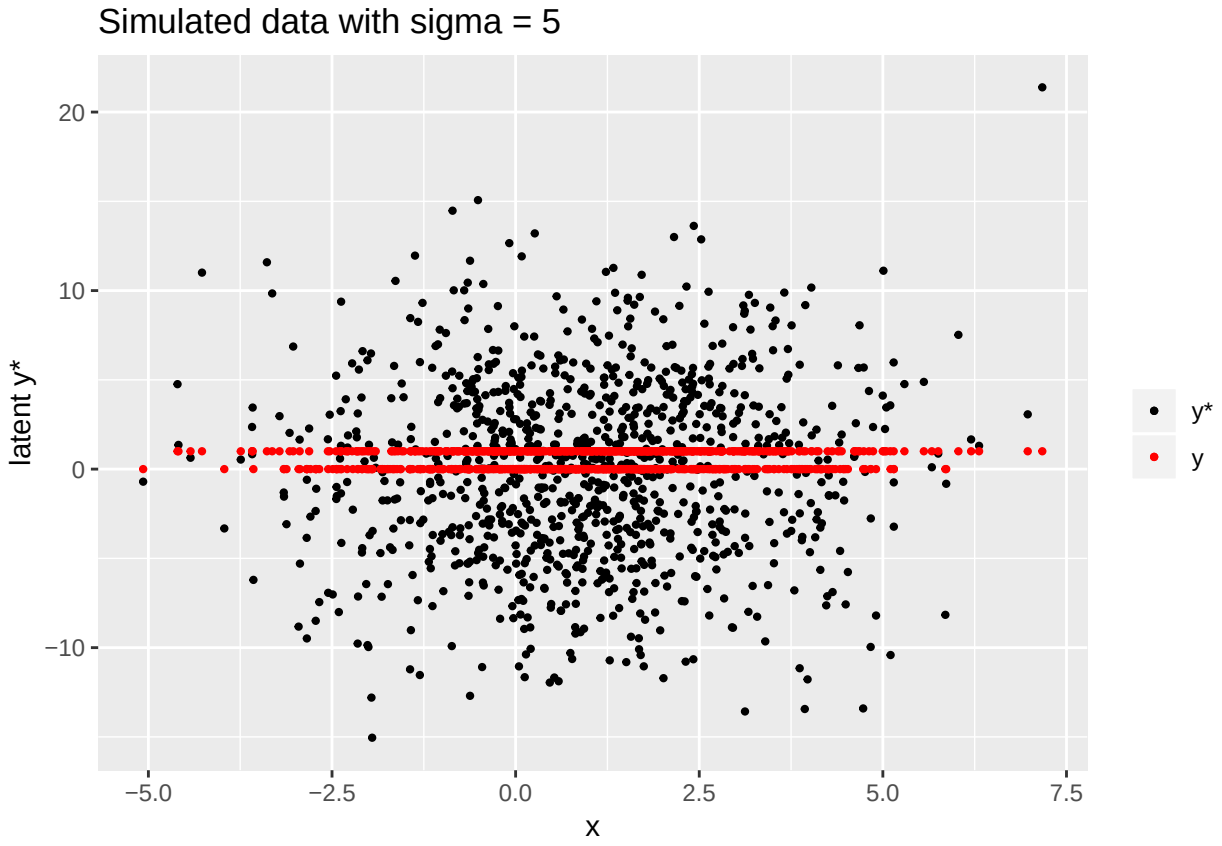
Both do well also in the situation with higher variance, this time also LPM seems quite accurate.

Distribution of marginal effects



$\sigma^2 = 25$

One more to go



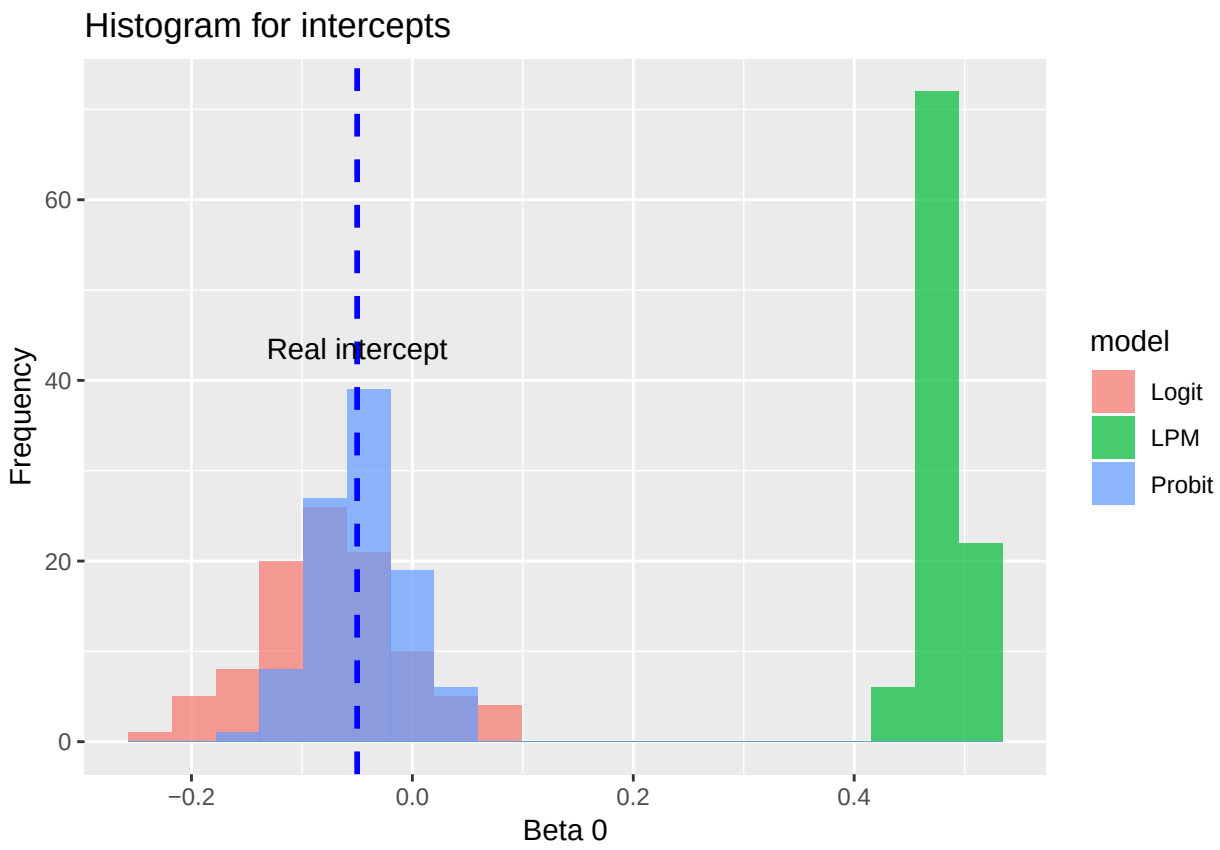

```
##            Real  LPM Probit Logit
## Intercept -0.25 0.48  -0.05 -0.07
## Beta 1     0.35 0.03   0.07  0.11
```

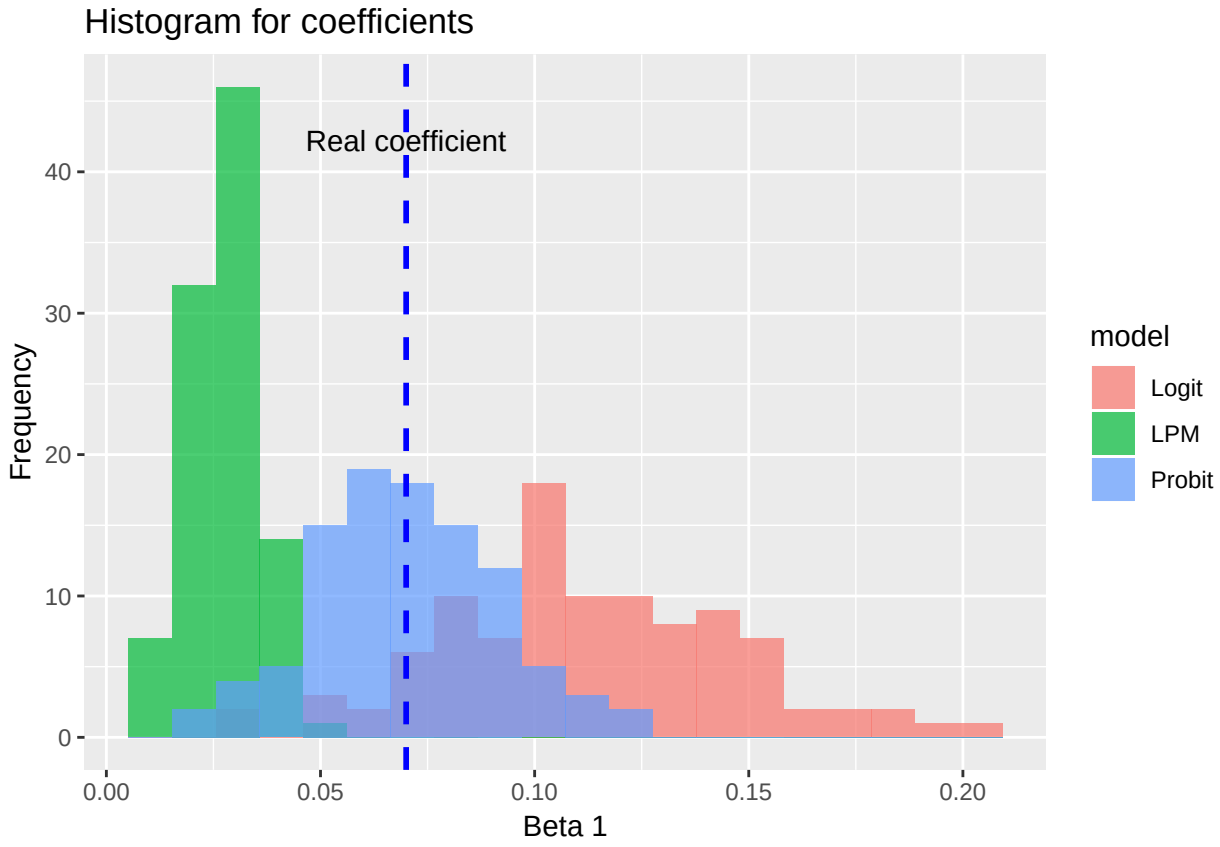
This time the real parameter values of the dgp are

```
## Real Intercept:  -0.05  and coefficient:  0.07
```

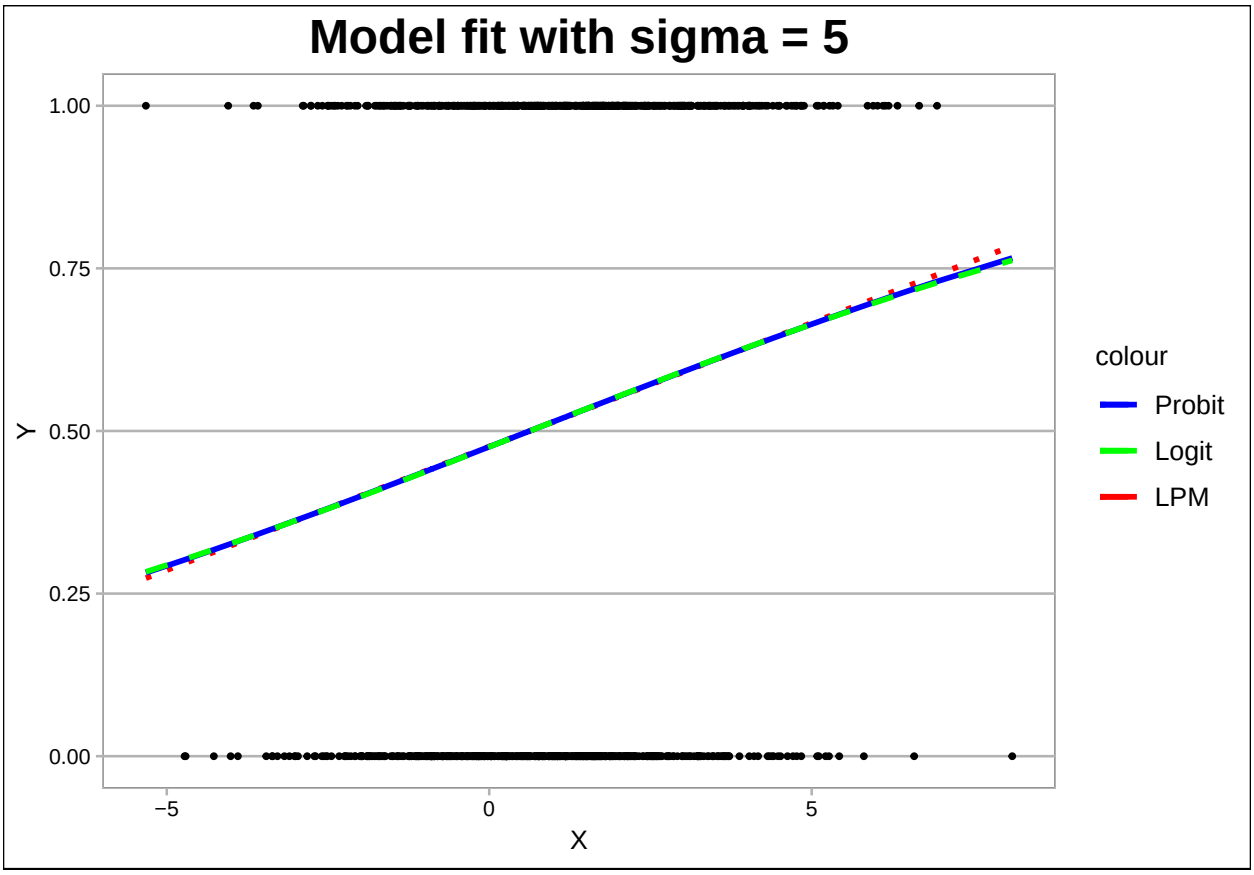
which are still close to the estimates dispite the large variance.

Distribution of coefficient values


Histogram for intercepts
Real intercept
Frequency
Beta 0
model
Logit
LPM
Probit

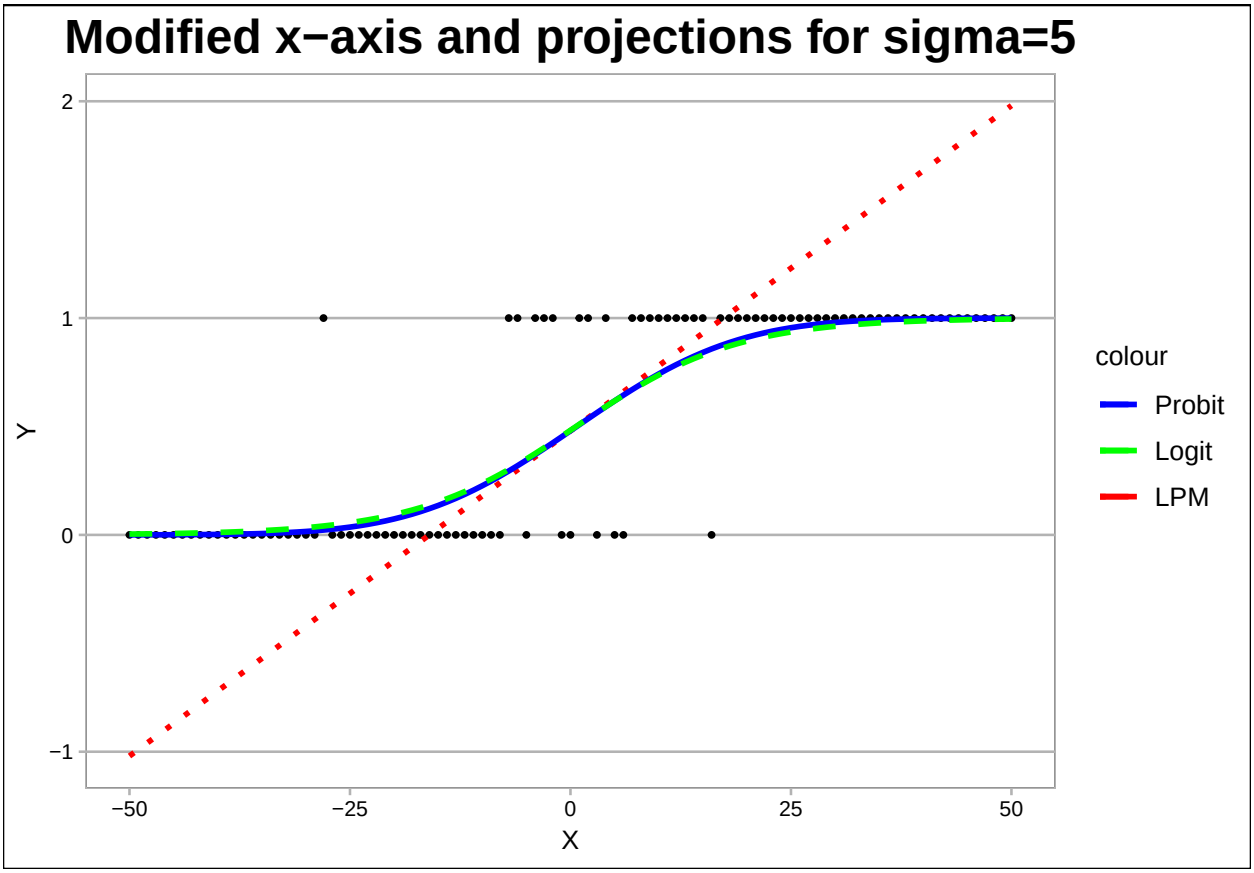

Histogram for coefficients
Real coefficient
Frequency
Beta 1
model
Logit
LPM
Probit

Fitted values



Probit and logit models are even more flat than previously, and scale back to the 0 and 1 on y axis much further.

Let's project the models further again.



The curvature sneaks in, but this time there are couple of outliers on y.

Marginal effects

The real mean marginal effect.

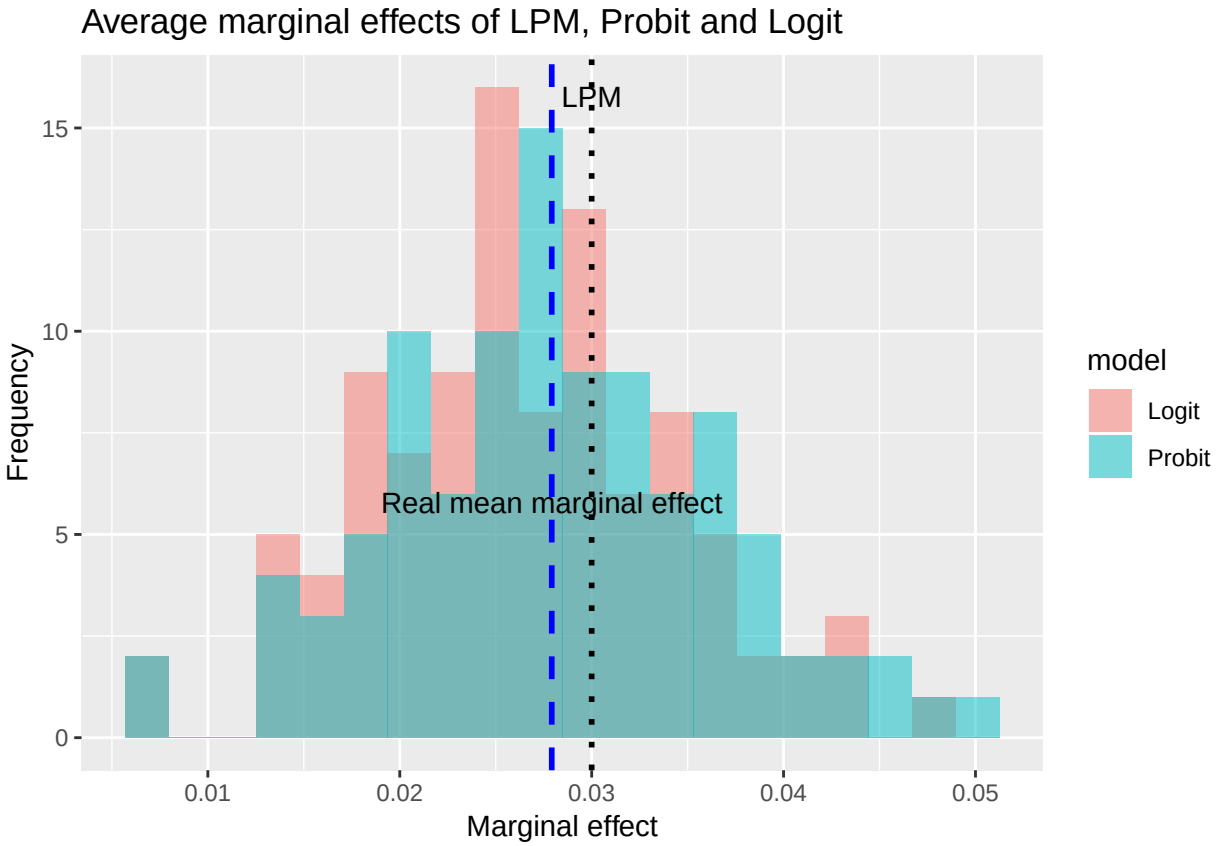
```
## Real mean marginal effect:  0.02792037
```

Estimated

LPM: 0.03

```
## Probit:  0.02800981  , Logit:  0.02635595
```

Distribution of marginal effects



Comparison of coefficients

Let's compare the values of estimated coefficients from each of the three runs with different sigmas.

```
# Real values
avg_coeffs[,1]
```

```
## Intercept    Beta 1
##     -0.25      0.35
```

```
# Sigma=1
avg_coeffs[,-1]
```

```
##             LPM Probit Logit
## Intercept 0.41  -0.26 -0.43
## Beta 1    0.11   0.34  0.57
```

```
# Sigma = 4
average_coeffs_2[,-1]
```

```
##             LPM Probit Logit
## Intercept 0.48  -0.06 -0.09
## Beta 1    0.03   0.09  0.14
```

```
# Sigma = 5
average_coeffs_3[,-1]
```

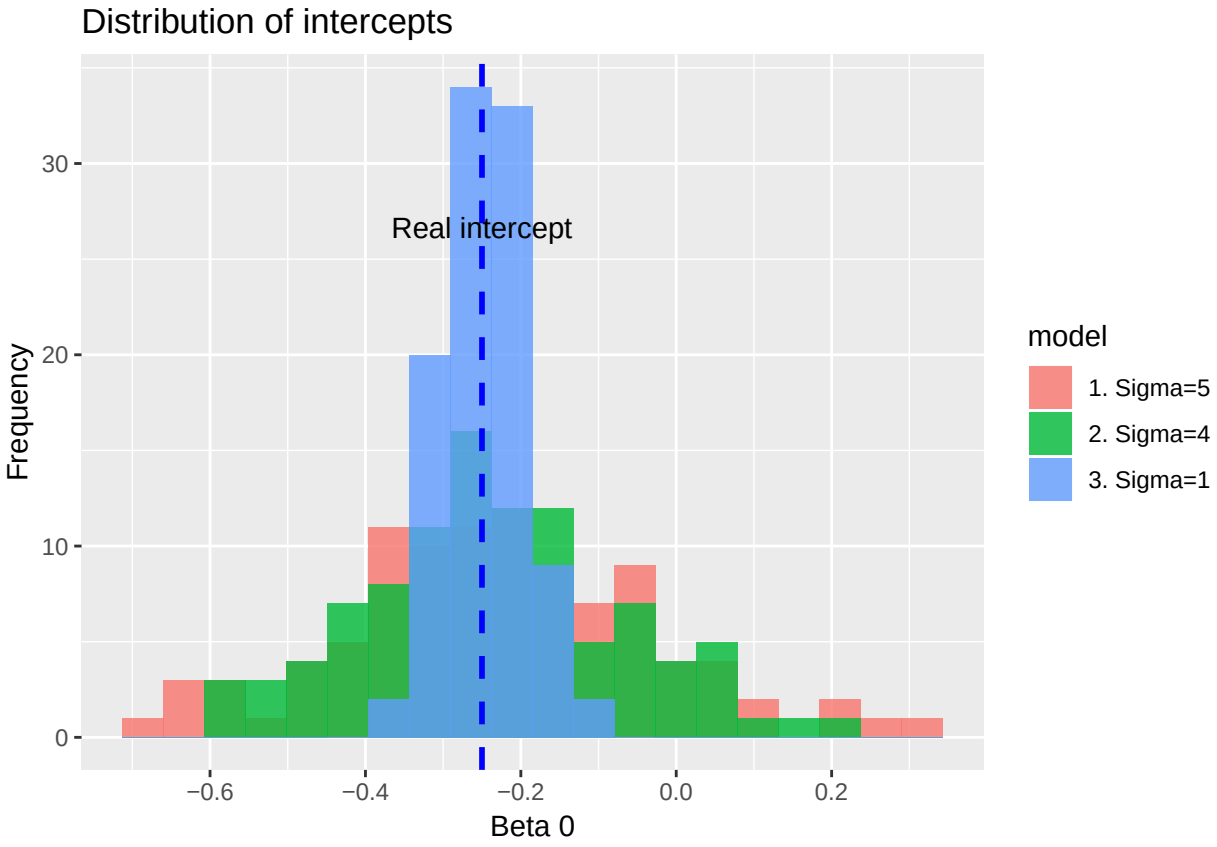
```
##             LPM Probit Logit
## Intercept 0.48  -0.05 -0.07
## Beta 1    0.03   0.07  0.11
```

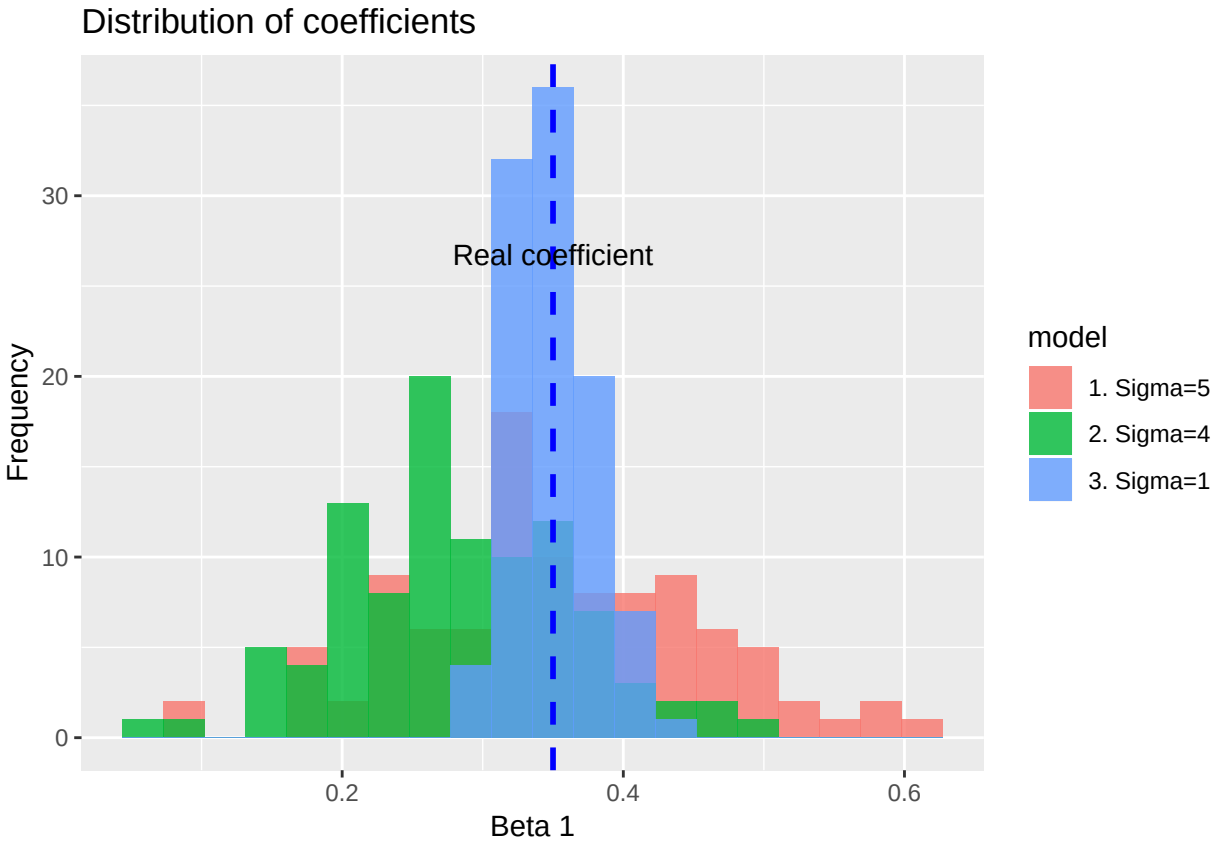
The comparison of the probit coefficients is not meaningful here, as all three models identified different coefficients $\frac{\beta}{\sigma}$.
The coefficients can be meaningfully compared, by scaling up the estimates from last two trials by the respective variances.

```
##           Probit 1 Probit 4 Probit 5
## Intercept    -0.26    -0.24    -0.25
## Beta 1        0.34     0.36     0.35
```

which is highly consistent.

Plot the distribution of the Probit coefficients for different sigma, properly standardized



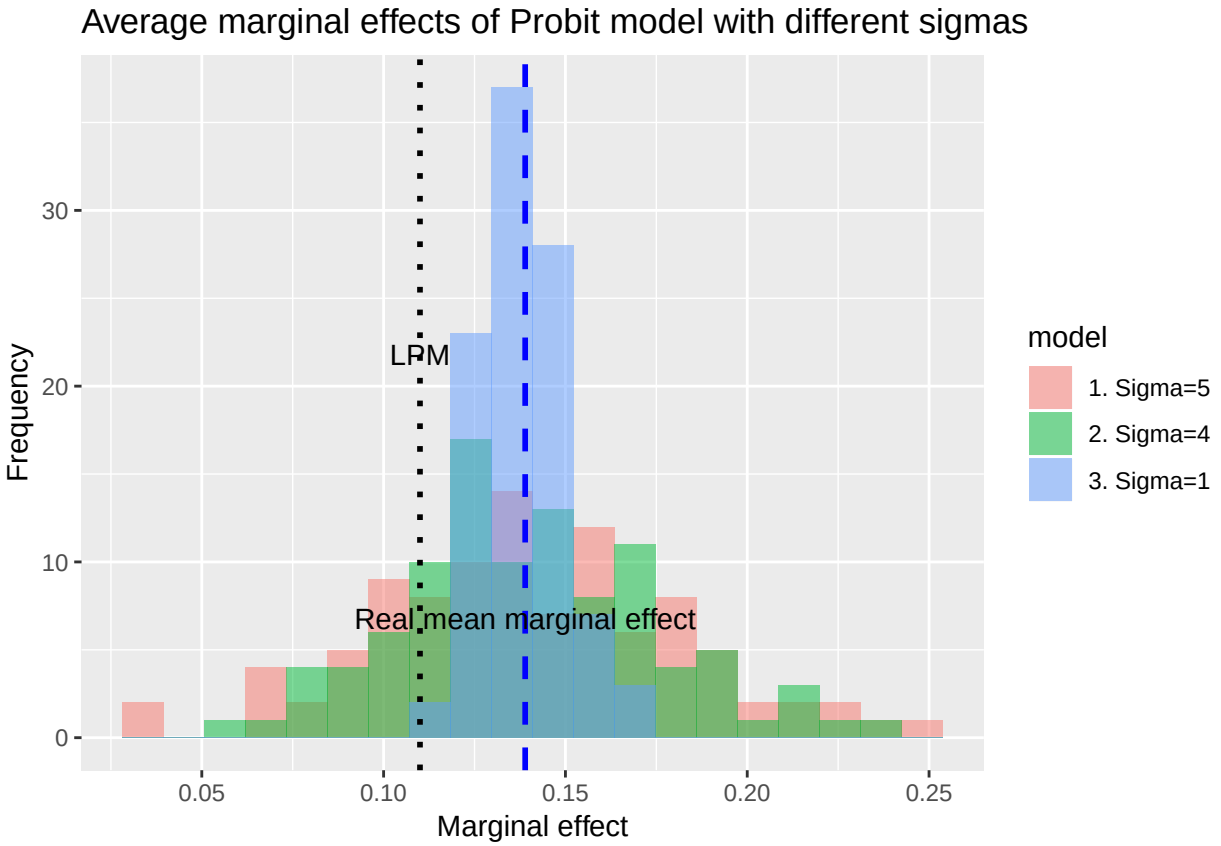


Comparison of marginal effects

The mean marginal effect is shrinking by increase of the variance, which is understandable, as the steepest part of the slope of the probit curve is flattening, as there is no clear cutoff in x which predict y.

```
## Mean marginal effects with sigma=1:  0.1381093
##  , sigma=4:  0.0350383  and
##  sigma=5:  0.02800981
```

Plot of the distribution of marginal effects of Probit models from three different variance experiments



3.3 Single firm entry

```
setwd("~/Documents/R/Data")
data <- readxl::read_xlsx("data_exercise1.xlsx")
colnames(data) <- c("x", "y")
data <- as.data.frame(data)
```

1. a)

Create a model for the indicator function determining, whether a firm i enters the market or not:

$$\mathbf{1}[\pi_i - K_i \geq 0]$$

where the expected gross profits from entry are given by $\mathbb{E}[\pi_i] = \alpha_0 + \alpha_1 x_i$ and K_i denotes the costs from entry.

Let's start off by defining the form of uncertainty related to both profits and costs: $\pi_i = \alpha_0 + \alpha_1 x_i + \varepsilon_i^\pi,\ \ \varepsilon_i^\pi \sim N(0,1)$ and $K_i = k_0 + \varepsilon_i^K,\ \ \varepsilon_i^K \sim N(0,1)$. In the cost equation k_0 refers to a fixed cost of entry known a priori and ε_i^K to the unknown variable costs, which differ for each firm. I got the idea for this clever decomposition of K_i form Jetro Anttonen.

Now we can define the intercept $\beta_0 = \alpha_0 + k_0$, the coefficient $\beta_1 = \alpha_1$ and error term $\varepsilon_i = \varepsilon_i^\pi - \varepsilon_i^K,\ \varepsilon_i \sim N(0,1)$ including the uncertainty related to profits and costs. Let's also define a latent variable $y^* = \beta_0 + \beta_1 x_i + \varepsilon_i$. The variable y^* determines the state of binary $y \in \{0,1\}$ of interest, which denotes the positivity or negativity

of the net profits. This gives us the relation between the explanatory variable x_i and the probability of positive profits from entering.

$$\begin{aligned} P(y = 1|x) &= P(y^* \geq 0) \\ &= P(\beta_0 + \beta_1 x_i - K_i + \varepsilon_i \geq 0) \\ &= F_\varepsilon(\beta_0 + \beta_1 x_i) \end{aligned}$$

where F_ε is the CDF of the standard normal, as all of the randomness in the model was granted standard normal.

a)

The interpretation for β_0 is that of a starting point cost-benefit assessment, if $x_i = 0$ is drawn. For β_1 the interpretation is, that a unit change in regressor x_i leads to $\beta_1 * \sigma_\varepsilon$ change in the latent $y* = \beta_0 + \beta_1 x_i + \varepsilon_i$.

b) The error term includes the uncertainty related to profits and costs of entering the market. As it was allowed to assume normality, I did so with regards to random variables ε_i^π and ε_i^K, inheriting normality also for their sum.

c)

Decision rule for the entrant:

$$\text{Decision} = \begin{cases} \text{Enter}, & \text{if } \beta_0 + \beta_1 x_i > 0 \\ \text{Don't enter}, & \text{if } \beta_0 + \beta_1 x_i \leq 0 \end{cases}$$

3.3.2. Estimate the model

```
model <- glm(formula = y ~ x, family = binomial(link = "probit"),
                              data = data)
params <- model$coefficients
params
```

```
## (Intercept)           x
## -0.98533493  0.04559989
```

3.3.3. Ministry of Finland

How would different tax regimes impact the decision?

a) Corporate tax $\tau > 0$

Assuming that this tax is imposed on $y^* = \beta_0 + \beta_1 x_i + \varepsilon_i$ and not on π_i. Let's also assume, that the taxes are only paid for positive net profits. The relationship between y and latent y* then becomes

$$\begin{aligned} P(y = 1|x) &= P(\tau(\beta_0 + \beta_1 x_i + \varepsilon) \geq 0) \\ &= P(\tau\beta_0 + \tau\beta_1 x_i + \tau\varepsilon \geq 0) \quad | \ \frac{1}{\tau} \\ &= P(\beta_0 + \beta_1 x_i + \varepsilon \geq 0) \\ &= F_\varepsilon(\beta_0 + \beta_1 x_i) \end{aligned}$$

which is the same as before. The reson for this is, that the proportional tax, no matter how small or large, will never change the sign of the net profit y^*. Therefore also the sign of the prediction for net profit $\beta_0 + \beta_1 x_i$ will be unaffected. By the same token, the probability of y being 1 remains unaltered.

Hence, our simple model can't help the ministry to evaluate the impact of this tax.

b) Lump sum tax τ_{lump}

This term can be added to the equation, and blended with the intercept term, as the tax is an additive level shift, just like the intercept itself.

$$\begin{aligned} P(y=1|x) &= P(\beta_0 + \beta_1 x_i - \tau_{\text{lump}} + \varepsilon \geq 0) \quad | \ \beta_0^* = \beta_0 - \tau_{\text{lump}} \\ &= P(\beta_0^* + \beta_1 x_i + \varepsilon \geq 0) \\ &= F_\varepsilon(\beta_0^* + \beta_1 x_i) \end{aligned}$$

3.3.4 Counterfactual

Let's conduct an actual counterfactual situation, where $\tau_{\text{lump}} = 1$

First we have to find out the mean marginal effect of the model, which is given by

```
params[[2]]*dnorm(params[[1]] + params[[2]])
```

```
## [1] 0.01169797
```

The tax changes the intercept, so calculating $\beta_0^* = \beta_0 - \tau_{\text{lump}}$ gives

```
beta_0_new <- params[[1]]-1
beta_0_new
```

```
## [1] -1.985335
```

This impacts the mean marginal effect, which is now

```
params[[2]]*dnorm(beta_0_new + params[[2]])
```

```
## [1] 0.002772309
```

The change in the mean marginal effect means, that on average, a unit increase in x_i will increase the probability of entry significantly less than before the tax. Hence, the ministry can tell the social democratic MP/Minister proposing the tax of this kind and magnitude, that it will have a large negative impact on the entries.