VE414 Lecture 15

Jing Liu

UM-SJTU Joint Institute

July 2, 2019

- Recall we have looked at linear models

$$\mathbf{Y} \mid \{\mathbf{X}, \boldsymbol{\beta}, \sigma^2\} \sim \mathrm{Normal}\left(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}\right)$$

where some priors need to be specified in Bayesian context, i.e.

$$\boldsymbol{\beta} \sim \mathrm{Normal}\left(\boldsymbol{\beta}_0, \boldsymbol{\Sigma}_0\right)$$
$$\frac{1}{\sigma^2} \sim \mathrm{Gamma}\left(\frac{\nu_0}{2}, \frac{\nu_0\sigma_0^2}{2}\right)$$

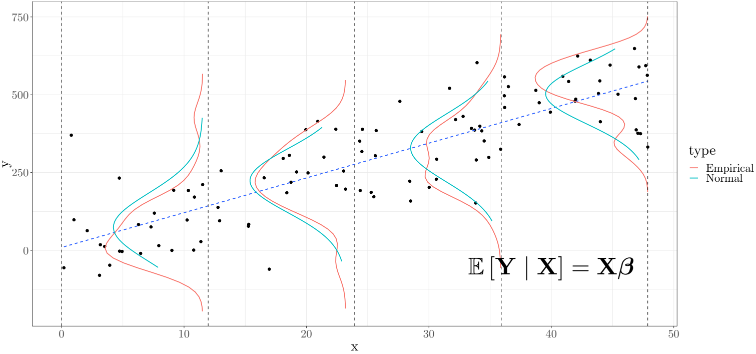
Q: Linear models assume the errors are normally distributed, they are the most widely used models by frequentists, why do frequentists like them so much?

- Without the normal assumption, the followings will not hold,

$$\hat{\boldsymbol{\beta}} \sim \text{Normal}\left(\boldsymbol{\beta}, \sigma^2\left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\right)$$
$$\hat{\mathbf{e}} \sim \text{Normal}\left(\mathbf{0}, \sigma^2\left(\mathbf{I} - \mathbf{P}\right)\right)$$

without which frequentists will not be able to do inference.

- By comparison, Bayesians are a lot less reliant on the normal assumption,



hence a Bayesian linear model can be as faithful to reality as possible,

$$
\begin{aligned}
\mathbf{Y} \mid \{\mathbf{X}, \boldsymbol{\beta}, \mathbf{Z}\} &\sim f_{\mathbf{Y} \mid \{\mathbf{X}, \boldsymbol{\beta}, \mathbf{Z}\}} \\
\boldsymbol{\beta} &\sim f_{\boldsymbol{\beta}} \\
\mathbf{Z} &\sim f_{\mathbf{Z}}
\end{aligned}
$$

- Having the wrong "shape" is often the least worry of the normal assumption.
- Consider the following dataset to see how the normal assumption is invalid

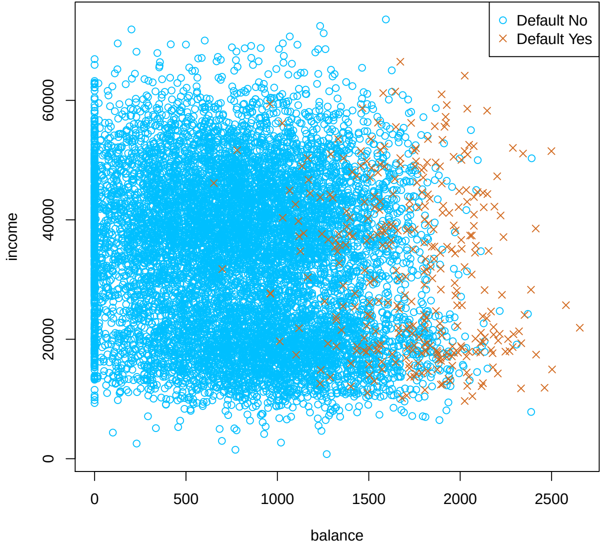
`Income`	Annual income
`Balance`	Credit card balance
`Default`	Whether the card holder has defaulted
`Student`	Whether the card holder is a student

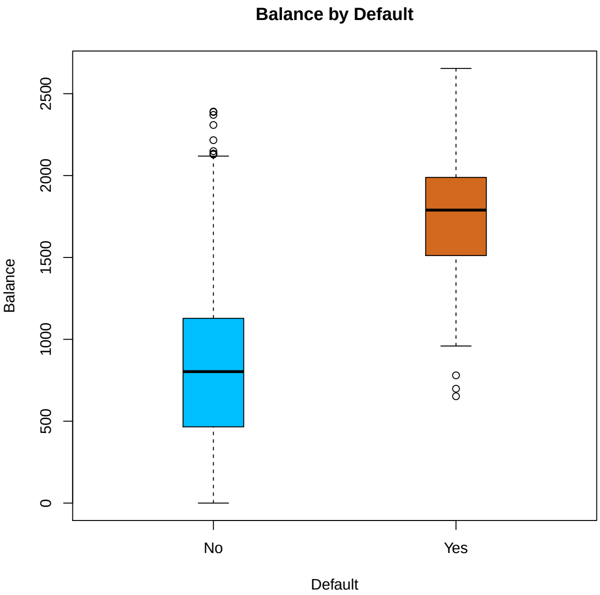
  which is a dataset that the bank want to use to predict credit card default.
- Notice the response is `Default` is categorical, i.e. the normal variable

$$\mathbf{Y} \mid \{\mathbf{X}, \boldsymbol{\beta}, \sigma^2\} \sim \mathrm{Normal}\left(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}\right)$$

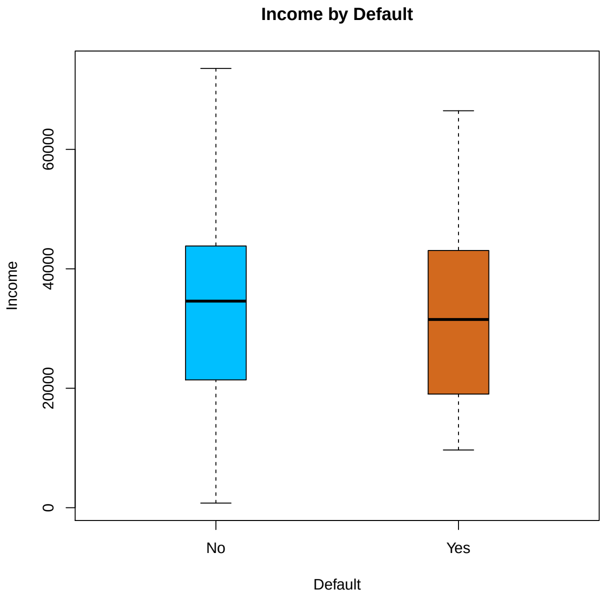
  is actually categorical, that is, it can only take a number of discrete values.
- The predictor variables here are categorical or continuous, which are fine.
- So the question here is vary similar to the one in Bayesian decision theory.

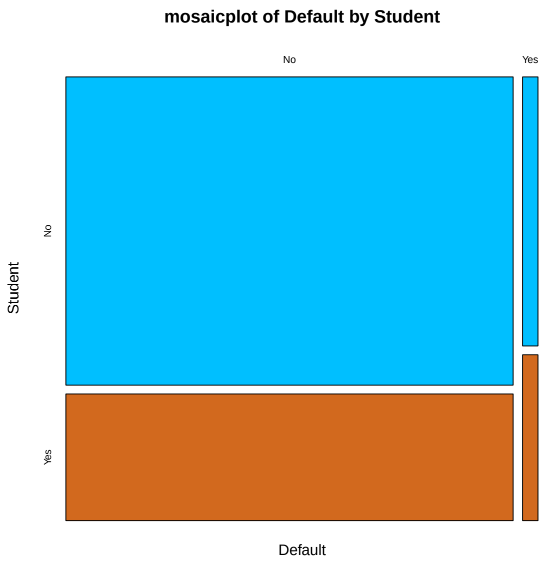
Credit Default





Balance by Default
Balance
0
500
1000
1500
2000
2500
No
Yes
Default


Income by Default
Income
0
20000
40000
60000
No
Yes
Default


mosaicplot of Default by Student
No
Yes
Student
No
Yes
Default

Q: Why a linear model is not going to be useful/meaningful here? e.g.

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

where X corresponds to `Balance` and Y corresponds `Default`

$$Y = \begin{cases} 0 & \text{if } \texttt{Default = No}; \\ 1 & \text{if } \texttt{Default = Yes}. \end{cases}$$

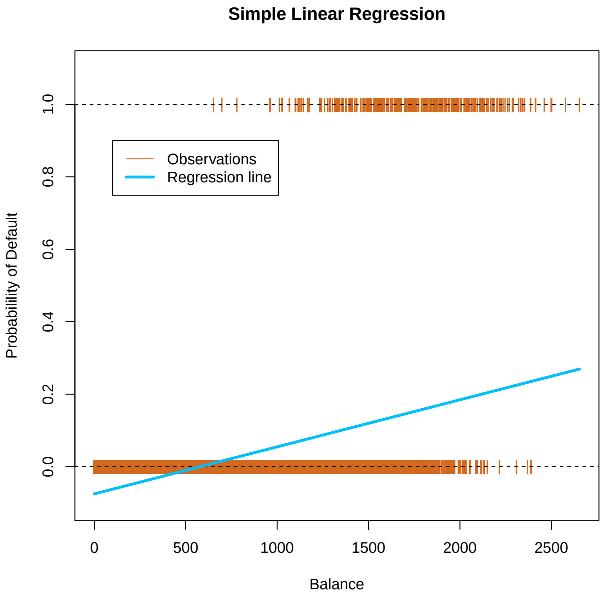
- Recall a simple linear regression models the conditional mean

$$\mathbb{E}\left[Y \mid X = x\right] = \beta_0 + \beta_1 x$$

by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ for β_0 and β_1, and we essentially use the estimate

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

to predict the conditional mean for every possible value of $X = x$.


Simple Linear Regression
Observations
Regression line
Probability of Default
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
Balance

- To avoid having an estimated probability outside of $[0,1]$, we model

$$\Pr\left(Y=1 \mid X=x\right)$$

using a function that has the range $[0,1]$ instead of using

$$\mathbb{E}\left[Y \mid X=x\right]=\Pr\left(Y=1 \mid X=x\right)=p\left(x,\boldsymbol{\beta}\right)=\beta_0+\beta_1 x$$

- There are many functions that meet this requirement, if the logistic function

$$\Pr\left(Y=1 \mid X=x\right)=p\left(x,\boldsymbol{\beta}\right)=\frac{\exp\left(\beta_0+\beta_1 x\right)}{1+\exp\left(\beta_0+\beta_1 x\right)}$$

is used, then we will end up with so-called logistic regression.

- In general, logistic regression can be thought as modelling the response by

$$Y_i \mid \{\mathbf{X}_i,\boldsymbol{\beta}\} \sim \mathrm{Binomial}\left(p,1\right) \quad \text{where} \quad p=\frac{\exp\left(\mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta}\right)}{1+\exp\left(\mathbf{x}_i^{\mathrm{T}}\boldsymbol{\beta}\right)}$$

- Frequentist would find β_0 and β_1 by maximising likelihood

$$\hat{\boldsymbol{\beta}} = \arg\max_{\mathbf{b}} \left\{ \mathcal{L}\left(\mathbf{b}; \mathbf{Y}, \mathbf{X}\right) \right\}$$

- Using the maximum likelihood estimate (MLE) of β_0 and β_1, we have

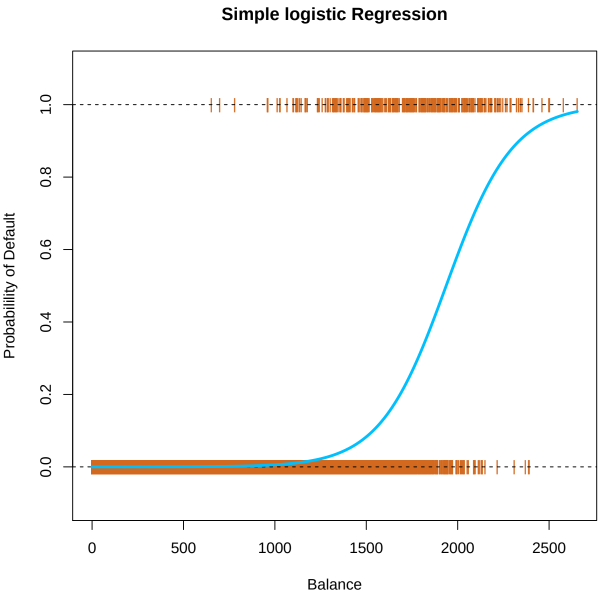
$$\hat{p}\left(x, \hat{\boldsymbol{\beta}}\right) = \frac{\exp\left(\hat{\beta}_0 + \hat{\beta}_1 x\right)}{1 + \exp\left(\hat{\beta}_0 + \hat{\beta}_1 x\right)}$$

which is an estimate of

$$\Pr\left(Y = 1 \mid X = x\right)$$

which gives the likelihood of defaulting in this example, e.g.

$$\Pr\left(\texttt{default=Yes} \mid \texttt{balance=1000}\right)$$


Simple logistic Regression
Probabilility of Default
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
Balance

Q: How to obtain MLEs $\hat{\beta}_0$ and $\hat{\beta}_1$? What is the likelihood function here?

$$\mathcal{L}\left(\beta_0, \beta_1; y_1, \ldots, y_n, x_1, \ldots, x_n\right) = \prod_{i=1}^{n} \Pr\left(Y = y_i\right)$$

- Under the assumption of independence and

$$\Pr\left(Y = 1 \mid X = x\right) = \frac{\exp\left(\beta_0 + \beta_1 x\right)}{1 + \exp\left(\beta_0 + \beta_1 x\right)}$$

the negative log-likelihood function is given by

$$\ln\left(\mathcal{L}\right) = \ln\left[\prod_{i=1}^{n}\Big(\Pr\left(Y = 1 \mid X = x_i\right)\Big)^{y_i}\Big(\Pr\left(Y = 0 \mid X = x_i\right)\Big)^{1-y_i}\right]$$

$$= \sum_{i=1}^{n}\Big(y_i\left(\beta_0 + \beta_1 x_i\right) - \ln\big(1 + \exp\left(\beta_0 + \beta_1 x_i\right)\big)\Big)$$

- We obtain two nonlinear equations when setting the first derivatives to zero

$$\sum_{i=1}^{n}\left(y_i - \frac{\exp\left(b_0 + b_1 x_i\right)}{1 + \exp\left(b_0 + b_1 x_i\right)}\right) = 0 \iff \sum_{i=1}^{n}\left(y_i - m\left(\mathbf{b}\right)\right) = 0$$

$$\sum_{i=1}^{n} x_i\left(y_i - \frac{\exp\left(b_0 + b_1 x_i\right)}{1 + \exp\left(b_0 + b_1 x_i\right)}\right) = 0 \iff \sum_{i=1}^{n} x_i\left(y_i - m\left(\mathbf{b}\right)\right) = 0$$

- We can then solve the two nonlinear equations numerically, for example,

```
> credit.LG = glm(default~balance, family = binomial,
+                 data = credit.df)
>
> coef(credit.LG)
```

```
  (Intercept)        balance
-10.651330614    0.005498917
```

R has a build-in logistic regression routine using Frequentist's approach.

- Thus, for the individual with `Balance = 1000`, our prediction is

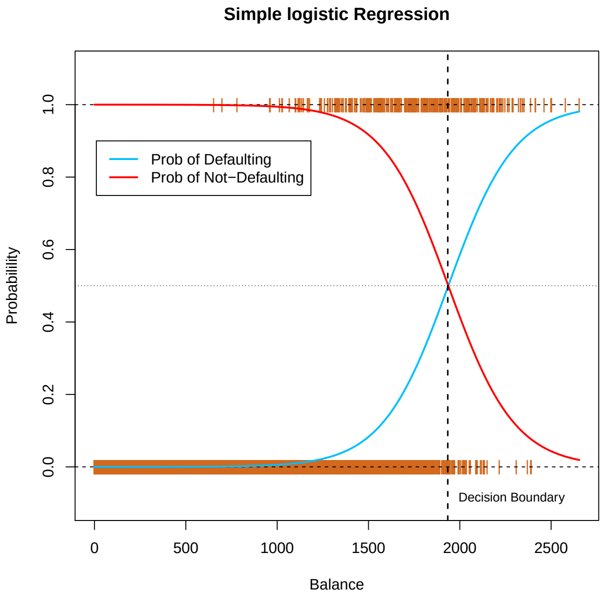
$$\hat{p} = \hat{\Pr}(Y = 1 \mid X = 1000) = \frac{\exp\left(\hat{\beta}_0 + \hat{\beta}_1 x\right)}{1 + \exp\left(\hat{\beta}_0 + \hat{\beta}_1 x\right)}$$

$$= \frac{\exp(-10.6513 + 0.0055 \cdot 1000)}{1 + \exp(-10.6513 + 0.0055 \cdot 1000)}$$

$$= 0.00575$$

```
> newdata.df = data.frame(balance = 1000)
>
> phat = predict(credit.LG, newdata = newdata.df,
+                  type = "response")
> phat
```

```
          1
0.005752145
```


Simple logistic Regression
Prob of Defaulting
Prob of Not−Defaulting
Probabilility
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
Balance
Decision Boundary

- In terms of decision theorem, the following can be used as a classifier

$$z = \arg\max_{z \in \{0,1\}} \left\{ z \cdot \hat{p} + (1 - z) \cdot (1 - \hat{p}) \right\}$$

which classifies the one with `Balance=1000` into the "not defaulting" class.

Q: How would a Bayesian approach the same problem?

$$Y \mid \{\beta_0, \beta_1, X\} \sim \text{Binomial}\,(p, 1)$$

where

$$p = \frac{\exp\left(\beta_0 + \beta_1 x\right)}{1 + \exp\left(\beta_0 + \beta_1 x\right)}$$

Q: Given n independent (x_i, y_i), what is the posterior if we use a normal prior?

$$\boldsymbol{\beta} \sim \text{Normal}\left(\boldsymbol{\mu}, \text{diag}\left(\sigma_0^2, \sigma_1^2\right)\right)$$

- In this case, the posterior takes the following form

$$f_{\boldsymbol{\beta}|\{\mathbf{Y},\mathbf{X}\}} \propto \mathcal{L} f_{\boldsymbol{\beta}} = \prod_{i=1}^{n} \left(\frac{\exp\left(\beta_0 + \beta_1 x_i\right)}{1 + \exp\left(\beta_0 + \beta_1 x_i\right)} \right)^{y_i} \left(\frac{1}{1 + \exp\left(\beta_0 + \beta_1 x_i\right)} \right)^{1-y_i} \cdot \exp\left(-\frac{1}{2\sigma_0^2} \left(\beta_0 - \mu_0\right)^2 - \frac{1}{2\sigma_1^2} \left(\beta_1 - \mu_1\right)^2 \right)$$

- The normalisation constant has no analytic form, but MAP can be used here

$$\begin{aligned}
\hat{\boldsymbol{\beta}}_{\mathsf{MAP}} &= \arg\max_{\mathbf{b}} \left\{ \mathcal{L}\left(\mathbf{b}; \mathbf{Y}, \mathbf{X}\right) \cdot f_{\boldsymbol{\beta}}\left(\mathbf{b}\right) \right\} \\
&= \arg\max_{\mathbf{b}} \left\{ \ln\left(\mathcal{L}\right) + \ln\left(f_{\boldsymbol{\beta}}\right) \right\} \\
&= \arg\max_{\mathbf{b}} \left\{ \sum_{i=1}^{n} \left(y_i \left(b_0 + b_1 x_i\right) - \ln\left(1 + \exp\left(b_0 + b_1 x_i\right)\right) \right) - \frac{1}{2\sigma_0^2} \left(b_0 - \mu_0\right)^2 - \frac{1}{2\sigma_1^2} \left(b_1 - \mu_1\right)^2 \right\} \\
&= \hat{\boldsymbol{\beta}}_{\mathsf{MLE}} \qquad \text{as} \qquad \sigma_0^2, \sigma_1^2 \to \infty
\end{aligned}$$

- Recall frequentist uses the following as the prediction

$$\hat{p} = \frac{\exp\left(\mathbf{x}^{\mathrm{T}}\hat{\boldsymbol{\beta}}_{\mathsf{MLE}}\right)}{1+\exp\left(\mathbf{x}^{\mathrm{T}}\hat{\boldsymbol{\beta}}_{\mathsf{MLE}}\right)} \quad \text{where} \quad \mathbf{x} = \begin{bmatrix} 1 \\ 1000 \end{bmatrix}$$

and the following could be used as a classifier

$$z = \underset{z\in\{0,1\}}{\arg\max}\left\{z\cdot\hat{p} + (1-z)\cdot(1-\hat{p})\right\}$$

Q: What is the MAP classifier in this case? How to obtain the MAP classifier?

$$\begin{aligned} z &= \underset{z\in\{0,1\}}{\arg\max}\, f_{Y^*|\{\mathbf{Y},\mathbf{X},X^*\}}\left(z \mid \mathbf{y},\mathbf{X},1000\right) \\ &= \underset{z\in\{0,1\}}{\arg\max} \int_{\mathcal{D}} f_{\{Y^*,\boldsymbol{\beta}\}|\{\mathbf{y},\mathbf{X},X^*\}}\left(z,\boldsymbol{\beta} \mid \mathbf{y},\mathbf{X},1000\right)\, d\boldsymbol{\beta} \\ &= \underset{z\in\{0,1\}}{\arg\max} \int_{\mathcal{D}} f_{Y^*|\{X^*,\boldsymbol{\beta}\}}\left(z \mid 1000,\boldsymbol{\beta}\right) f_{\boldsymbol{\beta}|\{\mathbf{Y},\mathbf{X}\}}\left(\boldsymbol{\beta} \mid \mathbf{y},\mathbf{X}\right)\, d\boldsymbol{\beta} \end{aligned}$$

- Thus it becomes a Bayesian computational problem again.
- Stan is a tool for building *standard* Bayesian model using *standard* MCMC.
- The best way to use Stan is via R/Python,

```
> # install.packages("rstan")
> library(rstan)
```

- A Stan program usually has three components:

```
data {
  int<lower=0> N;
  vector[N] x;
  int<lower=0,upper=1> y[N];
}
parameters {
  real beta0;
  real beta1;
}
model {
  y ~ bernoulli_logit(beta0 + beta1 * x);
}
```

- Once the Stan program on the last page is saved, it can be run within R

```
> fit = stan("logistic_credit_simple_414.stan",
+             data = list(
+                       x = credit.df$balance,
+                       y = credit.df$default,
+                       N = nrow(credit.df)),
+             chains = 4,
+             cores = 4,
+             iter = 1000,
+             warmup = 200))

> beta_simple_summary =
+   summary(fit, pars = c("beta0", "beta1"),
+           probs = c(0.025, 0.9725))$summary
> beta_simple_summary[, c("mean", "2.5%", "97.25%")]
```

```
              mean          2.5%        97.5%
beta0 -9.241026120 -1.128547e+01 -2.623256862
beta1  0.004594058 -1.155165e-05  0.005876377
```

- There are some difference even non-informative priors are used by default

```
> confint(credit.LG)
```

```
                    2.5 %       97.5 %
(Intercept) -11.383288936 -9.966565064
balance       0.005078926  0.005943365
```

```
> beta_simple_summary[, c("2.5%", "97.5%")]
```

```
               2.5%        97.5%
beta0 -1.128547e+01 -2.623256862
beta1 -1.155165e-05  0.005876377
```

- An informative prior can be specified in the model section

```
model {
  beta0 ~ normal(0, 1);          #New
  beta1 ~ normal(0, 0.1);        #New
  y ~ bernoulli_logit(beta0 + beta1 * x);
}
```

- Run this new stan program,

```
> fit_norm = stan("logistic_credit_norm_414.stan",
+               data = list(
+                         x = credit.df$balance,
+                         y = credit.df$default,
+                         N = nrow(credit.df)),
+               chains = 4, cores = 4,
+               iter = 1000, warmup = 200))
>
> beta_norm_summary =
+  summary(fit_norm, pars = c("beta0", "beta1"),
+           probs = c(0.025, 0.9725))$summary
>
> beta_norm_summary[, c("2.5%", "97.5%")]
```

```
                2.5%        97.5%
beta0 -1.046182e+01 -2.61198411
beta1  1.808603e-05  0.00539212
```

- The actual sample from stan can be extracted for further analysis, e.g

```
> beta0 = extract(object = fit_normal,
+                 pars = "beta0")
>
> head(beta0$beta0)
```

```
[1] -8.289337 -4.948232 -8.790315
[4] -9.277918 -9.023087 -4.939379
```

```
> getmode = function(v) {
+   uniqv = unique(v)
+   uniqv[which.max(tabulate(match(v, uniqv)))]
+ }
>
> getmode(beta0$beta0)
```

```
[1] -9.593992
```

- For very common models, the following package is useful,

```
> library(rstanarm)

> credit.bayesian.LG = stan_glm(default~balance,
+                      family = binomial(link = "logit"),
+                      prior_intercept = normal(0, 1),
+                      prior = normal(0,0.1), chains = 4,
+                      iter = 1000, data = credit.df)

> posterior_interval(credit.bayesian.LG, prob=0.95)
```

```
                  2.5%        97.5%
(Intercept) -7.431983 -6.769537753
balance      0.002993  0.003419604
```

```
> confint(credit.LG)
```

```
                   2.5 %       97.5 %
(Intercept) -11.383288936 -9.966565064
balance       0.005078926  0.005943365
```

- The posterior predictive distribution can obtained in the following way

```
> newdata.df = data.frame(balance = 1000)
```

- Take a sample from the posterior predictive predictive distribution

```
> post_pred_sample = posterior_predict(draws = 2000,
+   credit.bayesian.LG, newdata = newdata.df)
>
> post_pred_sample[1:20,]
```

```
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
```

- The estimate is simply the sample mean, which differs slightly from MLE

```
> mean(post_pred_sample); phat
```

```
[1] 0.0155
[1] 0.005752145
```

- It also classifies the one with `balance=1000` into the “not Defaulting” class.