Basic Stats

Bill Last Updated:

17 March, 2020

Contents

Preface: Motivation

All the notes I have done here are about basic stats. While I have tried my best, probably there are still some typos and errors. Please feel free to let me know in case you find one. Thank you!

Chapter 1

443

1.1 Some basic concepts

1.1.1 Permutation

An ordered arrangement of a set of objects is known as a permutation.

e.g., The number of permutations of n distinguishable objects is $n!$. e.g., The number of permuations of n distinct objects taken r at a time is

$$ {}_nP_r = \frac{n!}{(n-r)!} $$

1.1.2 Combinations

If the order of the objects is not important, then one may simply be interested in the number of combinations.

$$ \binom{n}{r} = \frac{n!}{r!(n-r)!} $$

1.1.3 Partitioning

The number of ways of partitioning a set of n objects into k cells with r_1 objects into the first cell, r_2 in the second cell, and so forth is

$$ \frac{n!}{r_1!r_2!...r_k!} $$

1.2 Discrete Random Variables

1.2.1 Binomial

$$X \sim BIN(n, p)$$

$$\binom{n}{x} P^x (1-P)^{n-x}$$

mean: np

variance: npq

(Note that, Bernoulli is written as $BIN(1, p)$)

1.2.2 Poisson

$$X \sim POI(\mu)$$

$$\frac{e^{-\mu}\mu^x}{x!}$$

mean: μ

variance: μ

1.3 Continuous Random Variables

1.3.1 Uniform

$$X \sim UNIF(a, b)$$

$$\frac{1}{b-a}$$

Mean: $\frac{a+b}{2}$

Variance: $\frac{(b-a)^2}{12}$

1.3.2 Exponential

$$X \sim EXP(\theta)$$

$$\frac{1}{\theta} e^{-x/\theta}$$

Mean: θ

Variance: θ^2

1.3.3 Normal

$$X \sim N(\mu, \sigma^2)$$

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Mean: μ

Variance" σ^2

1.4 Large Sample Theory

1.4.1 Convergence in distribution

https://en.wikipedia.org/wiki/Law_of_large_numbers

$$\bar{X} \to \mu \quad (n \to \infty)$$

$$Var(\bar{X}) = Var(\frac{1}{n}(X_1 + ... + X_n)) = \frac{1}{n^2} Var(X_1 + ... + X_n) = \frac{n\sigma^2}{n^2} = \frac{\sigma^2}{n}$$

1.4.2 Weak law

There are two different versions of the Law of Large Numbers: Strong law of large numbers and Weak law of large numbers.

The weak law of large numbers: The sample average converges in probability towards the expected value.

$$\bar{X_n} \xrightarrow{p} \mu \quad (n \to \infty)$$

This, for any positive number ϵ

$$lim_{n\to\infty} Pr(|\bar{X_n} - \mu| > \epsilon) = 0$$

1.4.3 Strong law

$$\bar{X_n} \xrightarrow{a.s.} \mu \quad (n \to \infty)$$

This is,

$$Pr(lim_{n\to\infty} \bar{X_n} = \mu) = 1$$

1.4.4 Central limit theorem

If $X_1, ..., X_n$ is a random sample from a distribution with mean μ and variance $\sigma^2 < \infty$, then the limiting distribution of

$$Z_n = \frac{\sum_{i=1}^{n} X_i - n\mu}{\sqrt{n}\sigma}$$

is the standard normal, $Z_n \xrightarrow{d} Z \sim N(0, 1)$ as $n \to \infty$.

1.4.4.1 Bernoulli law of large number

$\hat{p_n}$ converges stochasticalltly to p as n approchaes infinity. For example, if a coin is tossed repeatedly, and $A = \{H\}$, then the successive relative frequencies of A correspond to a sequence of random variables that will converge stochastically to $p = 1/2$.

1.4.4.2 Normal approximation to Binomial

$$Z_n = \frac{Y_n - np}{\sqrt{npq}} \xrightarrow{d} Z \sim N(0, 1)$$

Example: The probability that a basketball player hits a shot is $p = 0.5$. If he takes 20 shorts, what is the probability that he hits at least 9?

$$\begin{aligned} P[Y_{20} \geq 9] &= 1 - P[Y_{20}] \\ &= 1 - \sum_{y-0}^{8} \binom{20}{y} 0.5^y 0.5^{20-y} \\ &= 0.7483 \end{aligned}$$

A normal approximation is

$$\begin{aligned} P[Y_{20} \geq 9] &= 1 - P[Y_{20}] \\ &= 1 - \Phi(\frac{8-10}{\sqrt{5}}) \\ &= 0.8133 \end{aligned}$$

1.4.4.3 Normal approximation to Poisson

$$\begin{aligned} P[10 \leq Y_{20} \leq 30] &= P[Y_{20} \leq 30] - P[Y_{20} \leq 10] \\ &= \Phi[\frac{30.5-20}{\sqrt{20}}] - \Phi[\frac{9.5-20}{\sqrt{20}}] \\ &= 0.981 \end{aligned}$$

1.4.5 Poisson approximation to binomial

We know that the mean for binomial is

$$\mu = np \rightarrow p = \frac{\mu}{n}$$

The moment generating function for Binomial is

$$M_n(t) = (1 - p + pe^t)^n = (1 + \frac{\mu(e^t - 1)}{n})^n$$
$$lim_{n\rightarrow\infty} M_n(t) = e^{\mu(e^t-1)}$$

Note that the MGF for Poisson is as follows.

$$POI(\lambda) : e^{\lambda(e^t-1)}$$

Thus,

$$Y_n \rightarrow Y \sim POI(\mu)$$

Chapter 2

556

2.1 Statistics and Sampling Distributions

2.1.1 Statistics

2.1.1.1 Definition of Statistic

P.264

A fucntion of observable random variables, $T = t(X_1, ...X_n)$, which does not depend on any unknown parameters is called statistic.

For example, Let $X_1, ..., X_n$ represent a random sample from a population with $pdf\ f(x)$. The sample mean provides an example of a statistic with the function

$$t(x_1, ..., x_n) = (x_1 + ... + x_n)/n$$

This statistic usually is denoted by

$$\bar{X} = \sum_{i=1}^{n} \frac{X_i}{n}$$

When a random sample is observed, the value of $\bar{X}$, computed from the data, usually is denoted by lower case $\bar{x}$.

$$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n}$$

2.1.1.2 Sample and parameters

P.265

If $X_1, ..., X_n$ denotes a random sample from $f(x)$ with $E(X) = \mu$ and $var(X) = \sigma^2$, then

$$E(\bar{X}) = \mu$$

$$Var(\bar{X}) = \frac{\sigma^2}{n}$$

For example, a random sample of size n from a Bernoulli distribution $X_i \sim BIN(1, p)$. We know Bernoulli has $\mu = p$ and $\sigma^2 = pq$. In this case, the sample mean is

$$\bar{X} = Y/n = \hat{p}$$

Thus,

$$E(\hat{p}) = p$$

$$Var(\hat{p}) = \frac{pq}{n}$$

Thus, sample mean is the unbiased estimate for the population mean. However, you can not use sample mean's variance to estimate pupulation variance. That lead to definition of sample variance.

P.266

Sample variance:

$$S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \bar{X})^2$$

$$E(S^2) = \sigma^2$$

2.1.2 $\chi^2, t, F, beta$

2.1.2.1 χ^2

Always squre from standard normal, and the standardization can be using μ or $\bar{X}$.

$$\frac{(n-1)S_n^2}{\sigma^2} \sim \chi^2(n-1)$$

(Thus, we can see this is a bit weired, as the numerator is $\bar{X}$ is from the sample, whereas σ^2 is from the population.)

Thus, we can

$$\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{\sigma^2} \sim \chi^2(n-1)$$

(You can compare $\bar{X}$ with μ, we can see the only difference is that the χ^2 has one less degree of freedom because we use this degree of freedom to calculate the mean.)

We need to know the mean and variance of χ^2.

Assume that

$$X \sim x^2(v)$$

$$mean : v$$

$$variance : 2v$$

2.1.2.2 t

Definition

$$t(k) = \frac{N(0,1)}{\sqrt{\frac{\chi^2(k)}{k}}}$$

Property 1: t distribution is symmetrical

Given that t-distribution is symmetrical, we can get

$$H(-c) = 1 - H(c)$$

Property 2: t distribution has heavier tails than the normal

My note: t-distribution only has a parameter of k, which is determined by the χ^2's degree of freedom. Of course, χ^2 also only has one parameter, namely the degree of freedom.

2.1.2.3 *F*

If $V_1 \sim \chi^2(v_1)$ and $V_2 \sim \chi^2(v_2)$ are independent, then the random variable

$$\frac{V_1/v_1}{V_2/v_2} \sim F(v_1, v_2)$$

2.1.2.4 Beta

If $X \sim F(v_1, v_2)$

$$Y = \frac{(v_1/v_2)X}{1 + (v_1/v_2)X} \sim Beta(\alpha, \beta)$$

2.1.3 Large-sample approximations

P.280

If $Y_v \sim x^2(x)$, then

$$Z_v = \frac{Y_v - v}{\sqrt{2v}} \xrightarrow{d} Z \sim N(0, 1)$$

(The proof is based on CLT.)

2.2 Point Estimation

2.2.1 Method of moments estimators

2.2.1.1 Definition

The joint MGF of $(X_1, ..., X_n)$ is defined as $M(t_1, ..., t_n) = E(e^{\sum_{i=1}^{n} t_i X_i})$

When $X_1, ..., X_n$ are independent if and only if

$$M(t_1, t_n) = \prod_{i=1}^{n} M_{X_i}(t_i)$$

where $M_{X_i}(t_i)$ is the MGF of X_i

2.2.1.2 Well-known MGF

(1) Bernoulli with success probability p: $1-p+pe^t$

(2) Binomial Bin(n,p): $(1-p+pe^t)^n$

(3) Poisson $POI(\lambda)$: $e^{\lambda(e^t-1)}$

(4) Normal $N(\mu,\sigma^2)$: $e^{\mu t+\frac{1}{2}\sigma^2t^2}$

(5) Gamma $GAM(\theta,k)$: $(1-\theta t)^{-k}$

Two special cases:

(6) Chi-square $\chi^2(v)=GAM(2,\frac{v}{2})$: $(1-2t)^{-\frac{v}{2}}$

(7) Exponential $EXP(\theta)=GAM(\theta,1)$: $(1-\theta t)^{-1}$

2.2.2 least squares estimators

2.2.3 likelihood function and maximum likelihood estimators

2.2.3.1 Likelihood function

P.293

The joint density function of n random variables $X_1,...X_n$ evaluated at $x_1,...x_n$, say $f(x_1,...,x_n;\theta)$, is refererred to as a *likelihood function.*

2.2.3.2 Maximum likelihood estimators

P.294

Let $L(\theta)=f(x_1,...x_n;\theta),\theta\in\Omega$, be the joint pdf of $X_1,...,X_n$. For a given set of observations, $(x_1,...x_n)$, a value $\widehat{\theta}$ in Ω at which $L(\theta)$ is a maximum is called a *maximum likelihood estimate (MLE)* of θ. That is $\widehat{\theta}$ is a value of θ that satisfies

$$f(x_1,...x_n;\theta)=max_{\theta\in\Omega}f(x_1,...,x_n;\theta)$$

2.2.4 Invariance property of MLEs

p296

If $\hat{\theta}$ is the MLE of θ and if $u(\theta)$ is a function of θ, then $u(\hat{\theta})$ is an MLE of $u(\theta)$.

For instance,

2.2.5 Unbiased estimators

An estimator T is said to be an unbiased estimator of $\tau(\theta)$ if

$$E(T) = \tau(\theta)$$

for all $\theta \in \Omega$. Otherwise, we said that T is biased stimator of $\tau(\theta)$.

For instance, if we want to estimate a percentile, say the 95th percentile of $N(\mu, 9)$. Note that the percentiles that we know are about standardized noraml (i.e., $N(0, 1)$). Thus, we need to have some calculation to get the non-standard one.

$$\frac{X_{95\,percentile} - \mu}{\sigma} = 1.645$$

Thus, we can get

$$X_{95\,percentile} = 1.645 \times \sigma + \mu$$

We know that $\bar{X}$ is the unbiased estimate for μ. Thus, we can get

$$X_{95\,percentile} = 1.645 \times \sigma + \mu = 4.94 + \mu$$

We know that

$$E(T) = E(\bar{X} + 4.94) = \mu + 4.94$$

Thus, $T = \bar{X} + 4.94$ is the unbiased estimator of $\tau(\mu) = \mu + 4.94$.

2.2.6 Cramer-Rao lower bound

2.2.7 Best linear unbiased estimation (BLUE or MVLUE)

2.2.8 Rao-Blackwell theorem, UMVUEs

2.2.9 Consistency, asymptotic unbiasedness

2.2.10 Efficiency, asymptotic efficiency

2.2.11 Asymptotic properties of MLEs

2.3 Sufficient and completeness

2.3.1 Sufficiency and minimal sufficiency

2.3.2 Neyman factorization theorem, minimal sufficiency of MLEs

2.3.3 completeness, Lehmann-Scheffe completeness theorem

2.3.4 Exponential class, complete sufficient statistics

Chapter 3

Logit and Probit

3.1 Logit

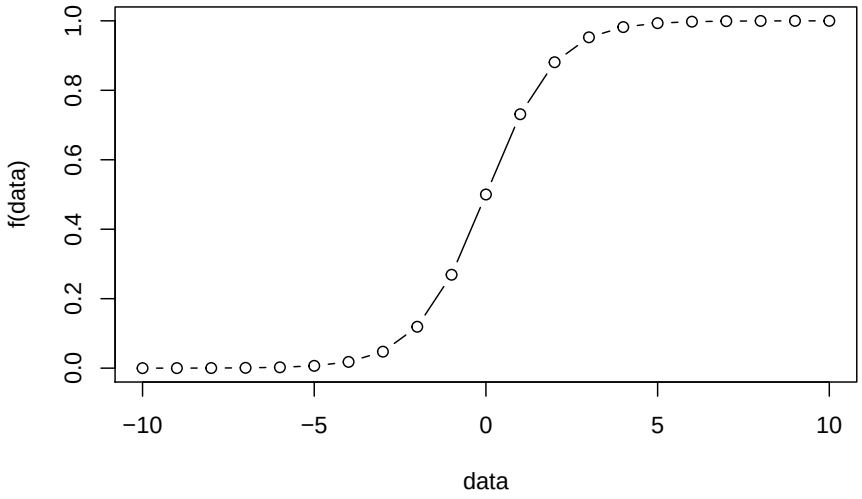
$$f(x) = log(\frac{p(y=1)}{1-p(y=1)})$$

The basic idea of logistic regression:

$$p(y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+...+\beta_n x_n)}} = \frac{e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}}{1+e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}}$$

Thus, $\beta_0 + \beta_1 x_1 + ... + \beta_n x_n$ can be from $-\infty$ to $+\infty$, and $p(y=1)$ will be always within the range of $(0, 1)$.

```
f<-function(x){exp(x)/(1+exp(x))}
data<-seq(-10,10,1)
plot(data,f(data),type = "b")
```



We can also write the function into another format as follows:

$$log\frac{p(y=1)}{1-p(y=1)} = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$$

Thus, we know that the regression coeficients of β_i actually change the "log-odds" of the event. Of course, note that the magnitude of β_i is dependent upon the units of x_i.

The following is an example testing whether that home teams are more likely to win in NFL games. The results show that the odd of winning is the same for both home and away teams.

```
mydata = read.csv(url('https://raw.githubusercontent.com/nfl-football-ops/Big-Data-Bowl
mydata$result_new<-ifelse(mydata$HomeScore>mydata$VisitorScore,1,0)
summary(mydata$result_new)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.4945  1.0000  1.0000
```

```
mylogit1 = glm(result_new~1, family=binomial, data=mydata)
summary(mylogit1)
```

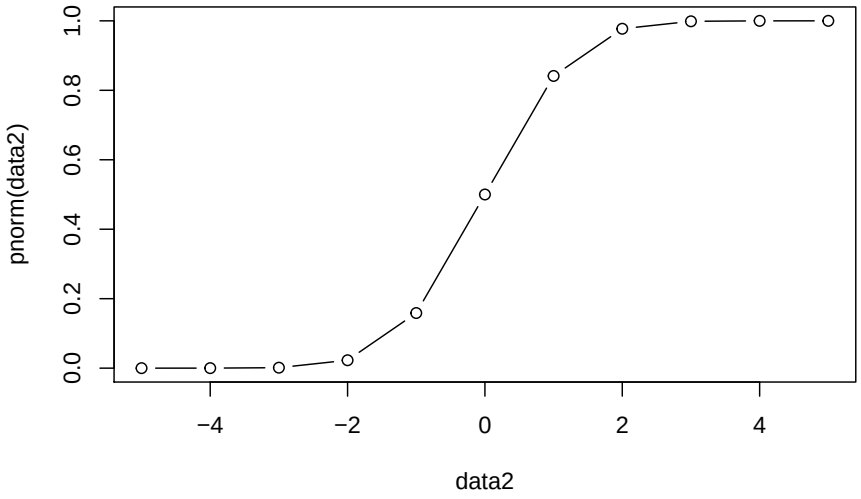
```
##
## Call:
```

```
## glm(formula = result_new ~ 1, family = binomial, data = mydata)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.168  -1.168  -1.168   1.187   1.187
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.02198    0.20967  -0.105    0.917
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 126.14  on 90  degrees of freedom
## Residual deviance: 126.14  on 90  degrees of freedom
## AIC: 128.14
##
## Number of Fisher Scoring iterations: 3
```

3.2 Probit

As noted above, logit $f(x) = log(\frac{p(y=1)}{1-p(y=1)})$ provides the resulting range of $(0, 1)$. Another way to provide the same rage is through the cdf of normal distribution.The following R code is used to illusrate this process.

```
data2<-seq(-5,5,1)
plot(data2,pnorm(data2),type = "b")
```



Thus, the cdf of normal distribution can be used to indicate the probability of $p(y = 1)$.

$$\Phi(\beta_0 + \beta_1 x_1 + ... + \beta_n x_n) = p(y = 1)$$

Similar to logit model, we can also write the inverse function of the cdf to get the function that can be from $-\infty$ to $+\infty$.

$$\beta_0 + \beta_1 x_1 + ... + \beta_n x_n = \Phi^{-1}(p(y = 1))$$

Thus, for example, if $X\beta$ = -2, based on $\Phi(\beta_0 + \beta_1 x_1 + ... + \beta_n x_n) = p(y = 1)$ we can get that the $p(y = 1) = 0.023$.

In contrast, if $X\beta = 3$, the $p(y = 1) = 0.999$.

```
pnorm(-2)
```

```
## [1] 0.02275013
```

```
pnorm(3)
```

```
## [1] 0.9986501
```

Let's assume that there is a latent variable called Y^* such that

$$Y^* = X\beta + \epsilon, \epsilon \sim N(0, \sigma^2)$$

You could think of Y^* as a kind of "proxy" between $X\beta + \epsilon$ and the observed $Y(1or0)$. Thus, we can get the following. Note that, it does not have to be zero, and can be any constant.

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

Thus,

$$y_i^* > 0 \Rightarrow \beta^{'} X_i + \epsilon_i > 0 \Rightarrow \epsilon_i > -\beta^{'} X_i$$

Thus, we can write it as follows. Note that $\frac{\epsilon_i}{\sigma} \sim N(0, 1)$

$$p(y = 1|x_i) = p(y_i^* > 0|x_i) = p(\epsilon_i > -\beta^{'} X_i) = p(\frac{\epsilon_i}{\sigma} > \frac{-\beta^{'} X_i}{\sigma}) = \Phi(\frac{\beta^{'} X_i}{\sigma})$$

We thus can get:

$$p(y = 0|x_i) = 1 - \Phi(\frac{\beta^{'} X_i}{\sigma})$$

For $p(y = 1|x_i) = \Phi(\frac{\beta^{'} X_i}{\sigma})$, we can not really estimate both β and σ as they are in a ratio. We can assume $\sigma = 1$, then $\epsilon \sim N(0, 1)$. We know y_i and x_i since we observe them. Thus, we can write it as follows.

$$p(y = 1|x_i) = \Phi(\beta^{'} X_i)$$

Chapter 4

Normal distribution

4.1 Basics

μ and σ determine the center and spread of the distribution.

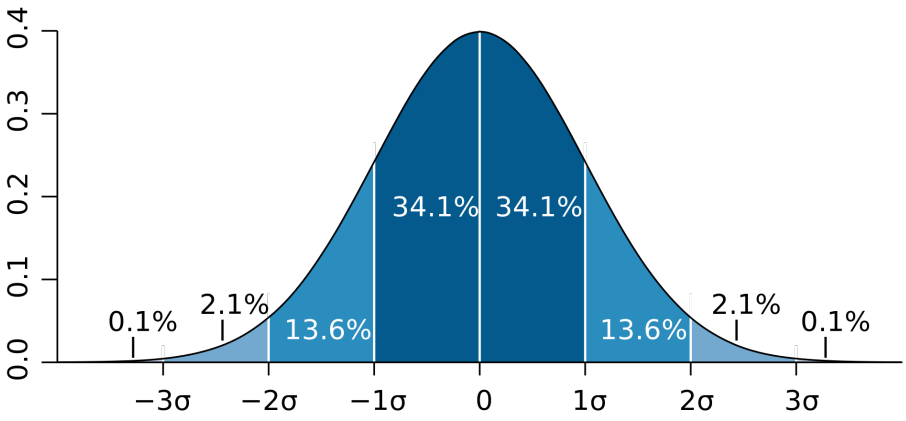


Figure 4.1: Normal

The empirical rule holds for all normal distributions:

(1) 68% of the area under the curve lies between $(\mu - \sigma, \mu + \sigma)$.

(2) 95% of the area under the curve lies between $(\mu - 2\sigma, \mu + 2\sigma)$.

(3) 99.7% of the area under the curve lies between $(\mu - 3\sigma, \mu + 3\sigma)$.

4.2 Confidence intervals for normal distributions

$$\bar{X} \pm Z\frac{\sigma}{\sqrt{n}}$$

where,

$\bar{X}$ is the mean

Z is the Z value (see the table below)

σ is the standard deviation

n is the number of observations

(We can see the connection between this formula and information shown in the *Basics* section.)

$Confidence\ Levels$	Z
80	1.282
85	1.440
90	1.645
95	1.960
99	2.576
99.5	2.807
99.9	3.291

4.3 Percentile

A percentile is a measure used in statistics indicating the value below which a given percentage of observations in a group of observations falls.

For example, the 20th percentile is the value (or score) below which 20% of the observations may be found.

For normal distribution,

-3 σ is the 0.13th percentile (i.e., $\frac{100-99.7}{2} = 0.15$);

-2 σ is the 2.28th percentile ((i.e., $\frac{100-95}{2} = 2.50$));

-1σ is the 15.87th percentile (i.e., $\frac{100-68}{2} = 16$);

0 σ is 50th percentile.

+2 σ is the 97.72nd percentile (i.e., $100 - \frac{100-95}{2} = 100 - 2.5 = 97.50$);

+3 σ is the 99.87th percentile (i.e., $100 - \frac{100-99.70}{2} = 100 - 0.15 = 99.85$).

This is related to the 68-95-99.7 rule or the three-sigma rule.

(Note that, it is *related*, not *direct* 68-95-99.7 rule, which is about symmetric situations. See the figure above)

Percentile	*Z*
90	1.282
–	1.440
95	1.645
–	1.960
–	2.576
–	2.807
99.9	3.000

Chapter 5

MLE

5.1 Basic idea of MLE

Suppose that we flip a coin, $y_i = 0$ for tails and $y_i = 1$ for heads. If we get p heads from n trials, we can get the proportion of heads is p/n, which is the sample mean. If we do not do any further calculation, this is our best guess.

Suppose that the true proablity is ρ, then we can get:

$$\mathbf{L}(y_i) = \begin{cases} \rho & y_i = 1 \\ 1-\rho & y_i = 0 \end{cases}$$

Thus, we can also write it as follows.

$$\mathbf{L}(y_i) = \rho^{y_i}(1-\rho)^{1-y_i}$$

Thus, we can get:

$$\prod \mathbf{L}(y_i|\rho) = \rho^{\sum y_i}(1-\rho)^{\sum(1-y_i)}$$

Further, we can get a log-transformed format.

$$log(\prod \mathbf{L}(y_i|\rho)) = \sum y_i log\rho + \sum(1-y_i)log(1-\rho)$$

To maximize the log-function above, we can calculate the derivative with respect to ρ.

$$\frac{\partial log(\prod \mathbf{L}(y_i|\rho))}{\partial \rho} = \sum y_i \frac{1}{\rho} - \sum(1-y_i)\frac{1}{1-\rho}$$

Set the derivative to zero and solve for ρ, we can get

$$\sum y_i \frac{1}{\rho} - \sum (1-y_i)\frac{1}{1-\rho} = 0$$

$$\Rightarrow (1-\rho)\sum y_i - \rho \sum (1-y_i) = 0$$

$$\Rightarrow \sum y_i - \rho \sum y_i - n\rho + \rho \sum y_i = 0$$

$$\Rightarrow \sum y_i - n\rho = 0$$

$$\Rightarrow \rho = \frac{\sum y_i}{n} = \frac{p}{n}$$

Thus, we can see that the ρ maximizing the likelihood function is equal to the sample mean.

5.2 Coin flip example, probit, and logit

In the example above, we are not really trying to estimate a lot of regression coefficients. What we are doing actually is to calculate the sample mean, or intercept in the regresion sense. What does it mean? Let's use some data to explain it.

Suppose that we flip a coin 20 times and observe 8 heads. We can use the R's glm function to esimate the ρ. If the result is consistent with what we did above, we should observe that the cdf of the esimate of β_0 (i.e., intercept) should be equal to $8/20 = 0.4$.

```
coins<-c(rep(1,times=8),rep(0,times=12))
table(coins)
```

```
## coins
##  0  1
## 12  8
```

```
coins<-as.data.frame(coins)
```

5.2.1 Probit

```
probitresults <- glm(coins ~ 1, family = binomial(link = "probit"), data = coins)
probitresults
```

```
##
## Call:  glm(formula = coins ~ 1, family = binomial(link = "probit"),
##     data = coins)
##
## Coefficients:
## (Intercept)
##     -0.2533
##
## Degrees of Freedom: 19 Total (i.e. Null);  19 Residual
## Null Deviance:       26.92
## Residual Deviance: 26.92     AIC: 28.92
```

```
pnorm(probitresults$coefficients)
```

```
## (Intercept)
##         0.4
```

As we can see the intercept is -0.2533, and thus $\Phi(-0.2533471) = 0.4$

5.2.2 Logit

We can also use logit link to calculate the intercept as well. Recall that

$$p(y=1) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\ldots+\beta_n x_n)}} = \frac{e^{\beta_0+\beta_1 x_1+\ldots+\beta_n x_n}}{1+e^{\beta_0+\beta_1 x_1+\ldots+\beta_n x_n}}$$

Thus,

$$p(y=1) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$$

```
logitresults <- glm(coins ~ 1, family = binomial(link = "logit"), data = coins)
logitresults$coefficients
```

```
## (Intercept)
##  -0.4054651
```

```
exp(logitresults$coefficients)/(1+exp(logitresults$coefficients))
```

```
## (Intercept)
##         0.4
```

Note that, the defaul link for the binomial in the glm function in logit.

5.3 Further on logit

The probablity of $y = 1$ is as follows:

$$p = p(y = 1) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x_1+...+\beta_n x_n)}} = \frac{e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}}{1 + e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}}$$

Thus, the likelihood function is as follows:

$$L = \prod p^{y_i}(1-p)^{1-y_i} = \prod (\frac{1}{1 + e^{-(\beta_0+\beta_1 x_1+...+\beta_n x_n)}})^{y_i} (\frac{1}{1 + e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}})^{1-y_i}$$

$$= \prod (1 + e^{-(\beta_0+\beta_1 x_1+...+\beta_n x_n)})^{-y_i} (1 + e^{\beta_0+\beta_1 x_1+...+\beta_n x_n})^{-(1-y_i)}$$

Thus, the log-likelihood is as follows:

$$logL = \sum (-y_i \cdot log(1+e^{-(\beta_0+\beta_1 x_1+...+\beta_n x_n)}) - (1-y_i) \cdot log(1+e^{\beta_0+\beta_1 x_1+...+\beta_n x_n}))$$

Typically, optimisers minimize a function, so we use negative log-likelihood as minimising that is equivalent to maximising the log-likelihood or the likelihood itself.

```
#Source of R code: https://www.r-bloggers.com/logistic-regression/

mle.logreg = function(fmla, data)
{
  # Define the negative log likelihood function
  logl <- function(theta,x,y){
    y <- y
    x <- as.matrix(x)
    beta <- theta[1:ncol(x)]

    # Use the log-likelihood of the Bernouilli distribution, where p is
    # defined as the logistic transformation of a linear combination
    # of predictors, according to logit(p)=(x%*%beta)
    loglik <- sum(-y*log(1 + exp(-(x%*%beta))) - (1-y)*log(1 + exp(x%*%beta)))
    return(-loglik)
  }

  # Prepare the data
  outcome = rownames(attr(terms(fmla),"factors"))[1]
  dfrTmp = model.frame(data)
  x = as.matrix(model.matrix(fmla, data=dfrTmp))
```

```
  y = as.numeric(as.matrix(data[,match(outcome,colnames(data))]))

  # Define initial values for the parameters
  theta.start = rep(0,(dim(x)[2]))
  names(theta.start) = colnames(x)

  # Calculate the maximum likelihood
  mle = optim(theta.start,logl,x=x,y=y, method = 'BFGS', hessian=T)
  out = list(beta=mle$par,vcov=solve(mle$hessian),ll=2*mle$value)
}
```

```
mydata = read.csv(url('https://stats.idre.ucla.edu/stat/data/binary.csv'))
mylogit1 = glm(admit~gre+gpa+as.factor(rank), family=binomial, data=mydata)

mydata$rank = factor(mydata$rank) #Treat rank as a categorical variable
fmla = as.formula("admit~gre+gpa+rank") #Create model formula
mylogit2 = mle.logreg(fmla, mydata) #Estimate coefficients


 print(cbind(coef(mylogit1), mylogit2$beta))
```

```
##                           [,1]         [,2]
## (Intercept)      -3.989979073 -3.772676422
## gre               0.002264426  0.001375522
## gpa               0.804037549  0.898201239
## as.factor(rank)2 -0.675442928 -0.675543009
## as.factor(rank)3 -1.340203916 -1.356554831
## as.factor(rank)4 -1.551463677 -1.563396035
```

5.4 References

http://www.columbia.edu/~so33/SusDev/Lecture_9.pdf

Chapter 6

Score, Gradient and Jacobian

6.1 Score

The score is the gradient (the vector of partial derivatives) of $logL(\theta)$, with respect to an m-dimensional parameter vector θ.

$$S(\theta) = \frac{\partial \ell}{\partial \theta}$$

Typically, they use ∇ to denote the partical derivative.

$$\nabla \ell$$

Such differentiation will generate a $m \times 1$ row vector, which indicates the sensitivity of the likelihood.

Quote from Steffen Lauritzen's slides: "Generally the solution to this equation must be calculated by iterative methods. One of the most common methods is the Newton–Raphson method and this is based on successive approximations to the solution, using Taylor's theorem to approximate the equation."

For instance, using logit link, we can get the first derivative of log likelihood logistic regression as follows. We can not really find β easily to make the equation to be 0.

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{n} x_i^T [y_i - \frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}}]$$
$$= \sum_{i=1}^{n} x_i^T [y_i - \hat{y_i}]$$

6.2 Fisher scoring

[I will come back to this later.]

https://www2.stat.duke.edu/courses/Fall00/sta216/handouts/diagnostics.pdf

https://stats.stackexchange.com/questions/176351/implement-fisher-scoring-for-linear-regression

6.3 Gradient and Jacobian

Remarks: This part discusses gradient in a more general sense.

When $f(x)$ is only in a single dimension space:

$\mathbb{R}^n \to \mathbb{R}$

$$\nabla f(x) = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}]$$

When $f(x)$ is only in a m-dimension space (i.e., Jacobian): $\mathbb{R}^n \to \mathbb{R}^m$

$$Jac(f) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \cdots & & & & \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \frac{\partial f_n}{\partial x_3} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

For instance,

$\mathbb{R}^n \to \mathbb{R}$:

$$f(x, y) = x^2 + 2y$$
$$\nabla f(x, y) = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}] = [2x, 2]$$

$\mathbb{R}^n \to \mathbb{R}^m$

$$f(x,y) = (x^2 + 2y, x^3)$$
$$Jac(f) = \begin{bmatrix} 2x & 2 \\ 2x^2 & 0 \end{bmatrix}$$

6.4 Hessian and Fisher Information

Hessian matrix or Hessian is a square matrix of second-order partial derivatives of a scalar-valued function, or scalar field.

$\mathbb{R}^n \to \mathbb{R}$

$$Hessian = \nabla^2(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} & \cdots & \frac{\partial^2 f}{\partial x_3 \partial x_n} \\ \cdots & & & & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \frac{\partial^2 f}{\partial x_n \partial x_3} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

As a special case, in the context of logit:

Suppose that the log likelihood function is $\ell(\theta)$. θ is a m demension vector.

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \cdots \\ \theta_m \end{bmatrix}$$

$$Hessian = \nabla^2(\ell) = \begin{bmatrix} \frac{\partial^2 \ell}{\partial \theta_1^2} & \frac{\partial^2 \ell}{\partial \theta_1 \partial \theta_2} & \frac{\partial^2 \ell}{\partial \theta_1 \partial \theta_3} & \cdots & \frac{\partial^2 \ell}{\partial \theta_1 \partial \theta_m} \\ \frac{\partial^2 \ell}{\partial \theta_2 \partial \theta_1} & \frac{\partial^2 \ell}{\partial \theta_2^2} & \frac{\partial^2 \ell}{\partial \theta_1 \partial \theta_3} & \cdots & \frac{\partial^2 \ell}{\partial \theta_1 \partial \theta_m} \\ \frac{\partial^2 \ell}{\partial \theta_3 \partial \theta_1} & \frac{\partial^2 \ell}{\partial \theta_3 \theta_2} & \frac{\partial^2 \ell}{\partial \theta_3^2} & \cdots & \frac{\partial^2 \ell}{\partial \theta_3 \partial \theta_m} \\ \cdots & & & & \\ \frac{\partial^2 \ell}{\partial \theta_m \partial \theta_1} & \frac{\partial^2 \ell}{\partial \theta_m \theta_2} & \frac{\partial^2 \ell}{\partial \theta_m \partial \theta_3} & \cdots & \frac{\partial^2 \ell}{\partial \theta_m \partial \theta_m} \end{bmatrix}$$

“In statistics, the observed information, or observed Fisher information, is the negative of the second derivative (the Hessian matrix) of the”log-likelihood” (the logarithm of the likelihood function). It is a sample-based version of the Fisher information.” (Direct quote from Wikipedia.)

Thus, the observed information matrix:

$$-Hessian = -\nabla^2(\ell)$$

Expected (Fisher) information matrix:

$$E[-\nabla^2(\ell)]$$

Chapter 7

Canonical link function

Inspired by a Stack Exchange post, I created the following figure:

$$\frac{Paramter}{\theta} \longrightarrow \gamma^{'}(\theta) = \mu \longrightarrow \frac{Mean}{\mu} \longrightarrow g(\mu) = \eta \longrightarrow \frac{Linearpredictor}{\eta}$$

For the case of n time Bernoulli (i.e., Binomial), its canonical link function is logit. Specifically,

$$\frac{Paramter}{\theta = \beta^T x_i} \longrightarrow \gamma^{'}(\theta) = \frac{e^{\beta^T x_i}}{1 + e^{\beta^T x_i}} \longrightarrow \frac{Mean}{\mu = \frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}} \longrightarrow g(\mu) = log\frac{\frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}}{1 - \frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}} \longrightarrow \frac{Linearpredictor}{\eta = \beta^T x_i}$$

Thus, we can see that,

$$\theta \equiv \eta$$

The link function $g(\mu)$ relates the linear predictor $\eta = \beta^T x_i$ to the mean μ.

Remarks:

(1) Parameter is $\theta = \beta^T x_i$ (Not μ!).

(2) $\mu = p(y = 1) = \frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}$ (Not logit!).

(3) Link function (i.e., $g(\mu)$) = logit = logarithm of odds = log $\frac{Event-Happened}{Event-Not-Happened}$.

(4) $g(\mu) = log\frac{\mu}{1-\mu} = \beta^T x_i$. Thus, link function = linear predictor = log odds!

(5) Quote from the Stack Exchange post "Newton Method and Fisher scoring for finding the ML estimator coincide, these links simplify the derivation of the MLE."

(Recall, we know that μ or $p(y = 1)$ is the mean function. Recall that, n trails of coin flips, and get p heads. Thus $\mu = \frac{p}{n}$.)

Chapter 8

Ordinary Least Squares (OLS)

Suppose we have n observation, and m variables.

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & ... & x_{1m} \\ x_{21} & x_{22} & x_{23} & ... & x_{2m} \\ ... & & & & \\ x_{n1} & x_{n2} & x_{n3} & ... & x_{nm} \end{bmatrix}$$

Thus, we can write it as the following n equations.

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + ... + \beta_m x_{1m} \\ y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + ... + \beta_m x_{2m} \\ y_3 &= \beta_0 + \beta_1 x_{31} + \beta_2 x_{32} + ... + \beta_m x_{3m} \\ &... \\ y_n &= \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + ... + \beta_m x_{nm} \end{aligned}$$

We can combine all the n equations as the following one:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_m x_{im} (i \in [1, n])$$

We can further rewrite it as a matrix format as follows.

$$y = X\beta$$

Where,

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ ... \\ y_n \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & ... & x_{1m} \\ 1 & x_{21} & x_{22} & x_{23} & ... & x_{2m} \\ ... & & & & & \\ 1 & x_{n1} & x_{n2} & x_{n3} & ... & x_{nm} \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ ... \\ \beta_m \end{bmatrix}$$

Since later we need the inverse of X, we need to make it into a square matrix.

$$X^T y = X^T X\hat{\beta} \Rightarrow \hat{\beta} = (X^T X)^{-1} X^T y$$

We can use R to implement this calculation. As we can see, there is no need to do any iterations at all, but rather just pure matrix calculation.

```
X<-matrix(rnorm(1000),ncol=2) # we define a 2 column matrix, with 500 rows
X<-cbind(1,X) # add a 1 constant
beta_true<-c(2,1,2) # True regression coefficients
beta_true<-as.matrix(beta_true)
y=X%*%beta_true+rnorm(500)

transposed_X<-t(X)
beta_hat<-solve(transposed_X%*%X)%*%transposed_X%*%y
beta_hat
```

```
##           [,1]
## [1,] 2.0669243
## [2,] 0.9325165
## [3,] 1.9621678
```

Side Notes The function of as.matrix will automatically make c(2,1,2) become the dimension of 3×1, you do not need to transpose the β.

8.1 Taylor series

$$\begin{aligned} f(x)|_a &= f(a) + \frac{f^{'}(a)}{1!}(x-a) + \frac{f^{'}(a)}{2!}(x-a)^2 + \frac{f^{''}(a)}{3!}(x-a)^3 + ... \\ &= \sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(x-a)^n \end{aligned}$$

For example:

$$\begin{aligned} e^x|_{a=0} &= e^a + \frac{e^a}{1!}(x-a) + \frac{e^a}{2!}(x-a)^2 + ... + \frac{e^a}{n!}(x-a)^n \\ &= 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + ... + \frac{1}{n!}x^n \end{aligned}$$

if $x = 2$

$e^2 = 7.389056$

$e^2 \approx 1 + \frac{1}{1!}x = 1 + \frac{1}{1!}2 = 3$

$e^2 \approx 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 = 1 + \frac{1}{1!}2 + \frac{1}{2!}2 = 5$ ...

$e^2 \approx 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \frac{1}{3!}x^2 + \frac{1}{4!}x^2 + \frac{1}{5!}x^2 = 7.2666...$

8.2 References

1. Steffen Lauritzen's slides:

http://www.stats.ox.ac.uk/~steffen/teaching/bs2HT9/scoring.pdf

2. The Stack Exchange post:

https://stats.stackexchange.com/questions/40876/what-is-the-difference-between-a-link-function-and-a-canonical-link-function

3. Wilipedia for OLS

https://en.wikipedia.org/wiki/Ordinary_least_squares

4. Gradient and Jacobian

https://math.stackexchange.com/questions/1519367/difference-between-gradient-and-jacobian

https://www.youtube.com/watch?v=3xVMVT-2_t4

https://math.stackexchange.com/questions/661195/what-is-the-difference-between-the-gradient-and-the-directional-derivative

5. Hessian

https://en.wikipedia.org/wiki/Hessian_matrix

6. Observed information

https://en.wikipedia.org/wiki/Observed_information

7. Fisher information

https://people.missouristate.edu/songfengzheng/Teaching/MTH541/Lecture%20notes/Fisher_info.pdf

8. Link function

https://en.wikipedia.org/wiki/Generalized_linear_model#Link_function

https://stats.stackexchange.com/questions/40876/what-is-the-difference-between-a-link-function-and-a-canonical-link-function

Chapter 9

Cholesky decomposition

9.1 Example 1

Use Cholesky decomposition to generate 1,000 trivariate normal deviates $X_1, ..., x_{1000}$ with mean $\mu = (-2, 4, 3)$ and covariance matrix

$$X = \begin{bmatrix} 2 & -1 & 0.5 \\ -1 & 4 & 1 \\ 0.5 & 1 & 5 \end{bmatrix}$$

```
Nsim = 10
means = c(-2,4,3)
N_columns = 3

# Generating random standard normal distribution numbers
Generated_numbers = matrix(rnorm(N_columns * Nsim), nrow = N_columns)

# The provided covariance matrix
cov_matrix = rbind(c(2, -1, 0.5), c(-1, 4, 1), c(0.5, 1, 5))

# Cholesky decomposition
Cholesky_decom_results = chol(cov_matrix)

# Data is transformed using the Cholesky decomposition
adjusted_data = t(Generated_numbers) %*% Cholesky_decom_results

Final_data = t(t(adjusted_data) + means)
```

```r
# calculating column means
colMeans(Final_data)
```

```
## [1] -2.01278  4.69800  3.33587
```

```r
# calculating column variances
apply(Final_data,2,var)
```

```
## [1] 2.703966 1.200539 1.605540
```

```r
# calculating covariance matrix
cov(Final_data)
```

```
##           [,1]      [,2]      [,3]
## [1,] 2.7039659 0.2838892 0.6641438
## [2,] 0.2838892 1.2005386 0.4852776
## [3,] 0.6641438 0.4852776 1.6055402
```

9.2 Example 2

AR(1) Covariance Matrix with Correlatiion Rho and Variance SigmaSq. Note that, there is only one individual or participant in this data simulation.

```r
n = 10;
SigmaSq = 5;
Rho = 0.8;

V = matrix(rep(n*n,0),n,n);

for (i in 1:n)
{
  for (j in i:n)
  {
    V[i,j]=SigmaSq*Rho^(j-i)
    V[j,i]=V[i,j]
  }
}

set.seed(123)
random_normal<-rnorm(n,2,1)
#chol(V) %*% random_normal
```

```
#colSums (chol(V))
b2<-t(as.matrix(random_normal))%*%chol(V)

pi = exp(b2)/(1 + exp(b2));

y<-ifelse(pi>runif(1),1,0)

y
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1    1    1    1    1    1    1    1    1     1
```

```
# The code above basically completes the generating job!
# The code below is to check

b = b2[2:n]
c = b2[1:(n-1)]
cor(b,c)
```

```
## [1] 0.8967058
```

```
sd(as.vector(b2))
```

```
## [1] 3.535119
```

```
# note that, you can not use var, as the mean is not zero, but rather it is 2
var(as.vector(b2))
```

```
## [1] 12.49707
```

```
#Not sure why the means are not the same ?
mean(as.vector(b2))
```

```
## [1] 10.01925
```

```
mean(random_normal)
```

```
## [1] 2.074626
```

9.3 Example 3

The following code very similar to the code shown above. However, it had only one observation. To illustrate the situation where there are more than one individual (or, participant), I did the code below.

```
n =25;   #the number of time points
m= 15;   # the number of participants or individuals, whichever ways you would like to
SigmaSq = 5;
Rho = 0.8;

filling_numbers<-rep(n*n,0)
V = matrix(filling_numbers,n,n);

for (i in 1:n)
{
  for (j in i:n)
  {
    V[i,j]=SigmaSq*Rho^(j-i)
    V[j,i]=V[i,j]
  }
}

set.seed(2345)
random_normal<-matrix(rnorm(m*n),nrow = m)
#chol(V) %*% random_normal
#colSums (chol(V))
b2<-random_normal%*%chol(V)

pi = exp(b2)/(1 + exp(b2));

random_unfirom<-matrix(runif(m*n),nrow = m)

y<-ifelse(pi>random_unfirom,1,0)
y
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]    0    0    0    0    0    0    1    1    1     1     1     0     1
##  [2,]    0    1    1    1    1    1    0    1    0     0     0     0     0
##  [3,]    1    0    0    0    0    1    0    1    0     1     1     0     0
##  [4,]    1    0    0    0    0    0    0    0    1     0     0     0     0
##  [5,]    0    0    0    0    1    0    0    0    0     0     0     0     1
##  [6,]    0    0    1    1    0    0    0    1    0     0     0     0     0
##  [7,]    0    0    0    0    0    1    0    0    1     0     0     0     1
##  [8,]    1    1    1    1    0    0    0    0    0     1     1     0     0
```

```
##  [9,]    1    1    1    1    0    1    1    1    0    1    0    0    0
## [10,]    1    0    1    1    1    1    1    1    1    1    0    0    0
## [11,]    1    1    1    1    1    1    1    1    1    0    1    0    0
## [12,]    1    1    1    1    1    1    0    1    1    1    1    1    0
## [13,]    0    1    0    0    0    0    0    0    1    1    1    1    0
## [14,]    1    1    0    1    1    1    1    1    1    0    0    0    0
## [15,]    1    1    0    1    1    1    1    0    0    1    0    0    1
##       [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
##  [1,]     1     1     1     1     0     1     0     0     1     0     0     0
##  [2,]     0     1     1     1     1     1     0     0     0     1     0     1
##  [3,]     0     1     0     0     1     1     1     1     0     0     1     1
##  [4,]     1     0     0     0     0     0     1     0     1     0     0     0
##  [5,]     0     0     1     1     1     0     1     1     1     0     1     1
##  [6,]     1     1     0     0     1     0     0     1     0     1     1     1
##  [7,]     0     0     0     0     0     1     0     0     1     0     0     0
##  [8,]     0     0     0     0     0     0     0     0     0     1     1     1
##  [9,]     0     1     1     1     0     0     1     0     0     0     0     0
## [10,]     1     1     1     1     1     1     1     1     1     1     1     1
## [11,]     0     0     0     1     0     0     0     0     1     1     1     0
## [12,]     0     0     0     0     1     1     1     0     0     0     1     0
## [13,]     0     1     0     0     0     1     0     1     1     1     1     1
## [14,]     0     0     0     0     1     0     1     0     0     0     0     0
## [15,]     0     0     0     0     0     0     1     0     1     0     0     1
```

```
# The code above basically completes the generating job! The code below is to check

# The following calcuates variance
# calculate variance of each column
mean(apply(b2, 2, var))
```

```
## [1] 4.330903
```

```
# calculate variance of each row
mean(apply(b2, 1, var))
```

```
## [1] 3.568107
```

```
# The whole table
var(as.vector(b2))
```

```
## [1] 4.299165
```

```
# The following code calculates the correlation
b = b2[,2:n]
c = b2[,1:(n-1)]

collected_cor<-rep(0,m-1) #creating an empty vector to collect correlation.
for (i in 1:(m-1))
{collected_cor[i]<-cor(b[i,],c[i,])}
collected_cor
```

```
##  [1] 0.8473037 0.7065013 0.6376223 0.5481540 0.7851062 0.6576329 0.4844481
##  [8] 0.6950847 0.6731673 0.6409116 0.7966547 0.7184030 0.8001861 0.7913736
```

```
mean(collected_cor)
```

```
## [1] 0.6987535
```

```
mean(y)
```

```
## [1] 0.456
```

```
log(mean(y)/(1-mean(y)))
```

```
## [1] -0.1764564
```

```
# It will always get a value close to zero, since we set the mean to be zero when simu
```