統計モデリング

$ conda install -c r r-essentials

In [1]:

```
"+" <- function(e1, e2) {
   if (is.character(c(e1, e2))) {
      paste(e1, e2, sep = "")
   } else {
      base::"+"(e1, e2)
   }
}
```

2章 確率分布と統計モデルの最尤推定

データ読み込み

In [2]:

```
# 2-1. 例題:種子数の統計モデリング
DIR="/var/jupyter/10_r/"
load(DIR + "data.RData")
```

データをみる

In [3]:

```
data
```

```
 2  2  4  6  4  5  2  3  1  2  0  4  3  3  3  3  4  2  7  2  4  3  3  3
 4  3  7  5  3  1  7  6  4  6  5  2  4  7  2  2  6  2  4  5  4  5  1  3
 2  3
```

In [4]:

```
length(data)
```

```
50
```

In [5]:

```
summary(data)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  0.000   2.000   3.000   3.560   4.750   7.000 
```

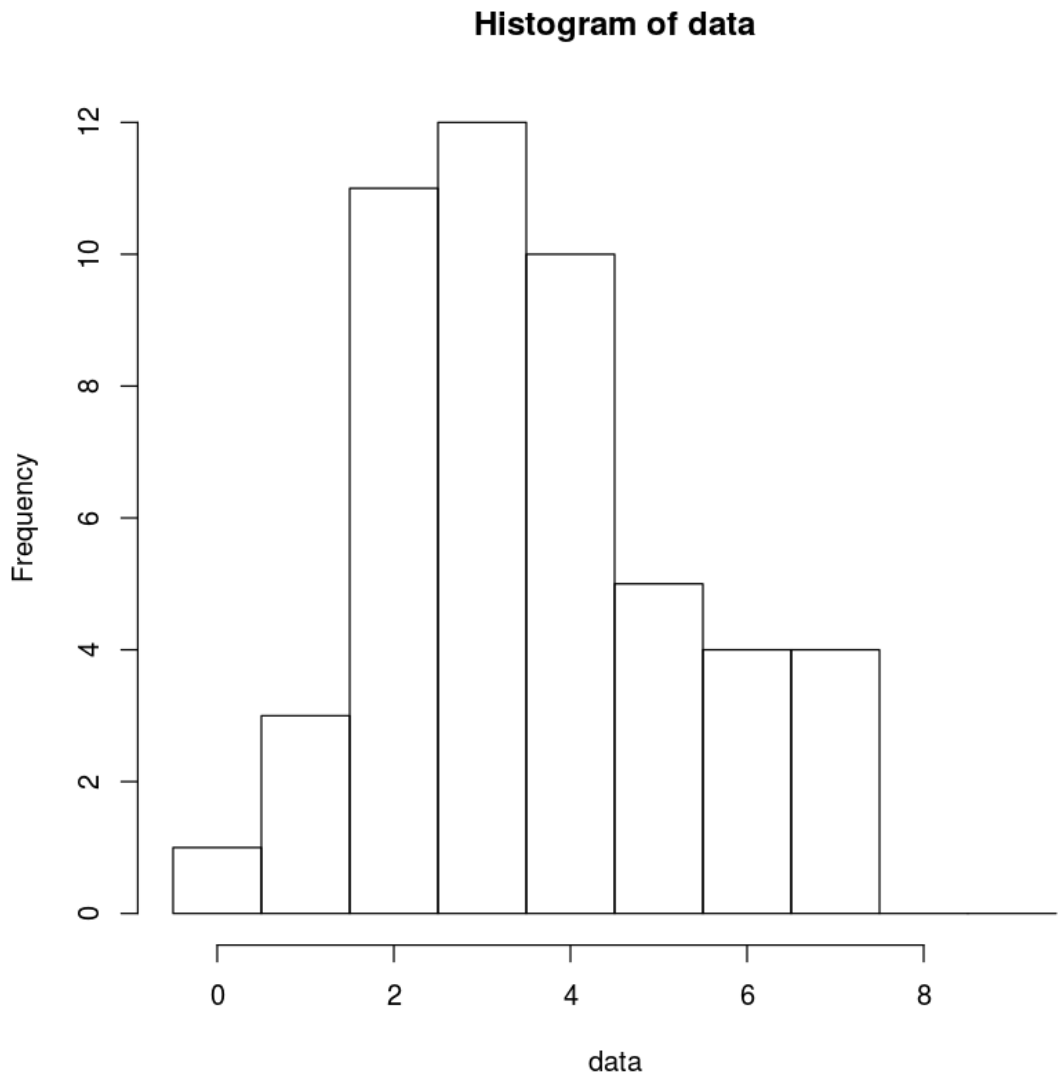
In [6]:

```
table(data)
```

```
data
 0  1  2  3  4  5  6  7
 1  3 11 12 10  5  4  4
```

In [7]:

```
hist(data, breaks=seq(-0.5, 9.5, 1))
```



In [8]:

```
var(data)
```

```
2.98612244897959
```

In [9]:

```
sd(data)
```

```
1.72804006000428
```

In [10]:

```
sqrt(var(data))
```

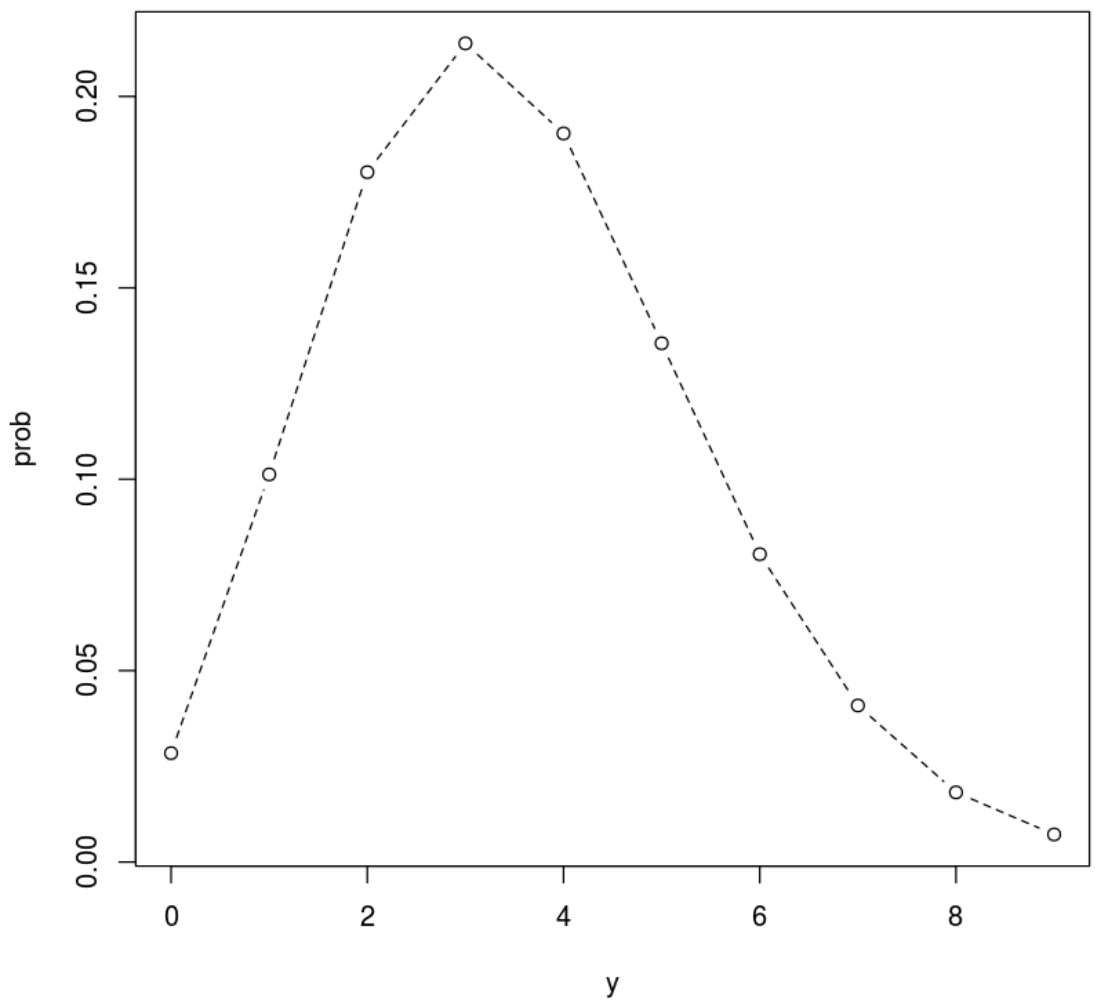
1.72804006000428

平均と分散が3前後で比較的近いので、ポアソン分布で近似できそう。

ポアソン分布をプロット

In [11]:

```
# 2-2. データと確率分布の対応関係をながめる
y = 0:9
prob = dpois(y, lambda=3.56)
plot(y, prob, type="b", lty=2)
```

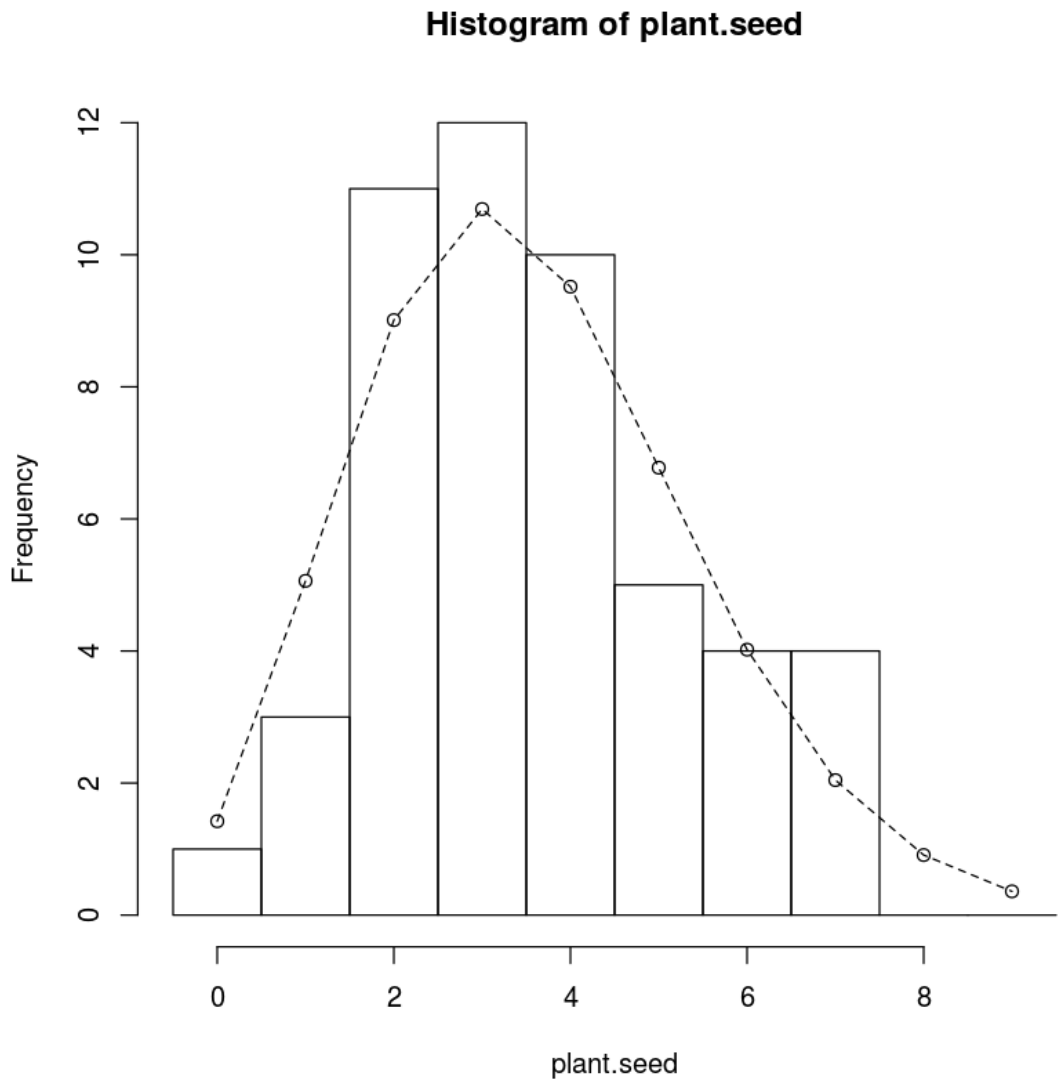


データとポアソン分布の対応をみる

In [12]:

```
plant.seed = data
length(plant.seed)

hist(plant.seed, breaks = seq(-0.5, 9.5, 1))
points(y, prob * 50)
lines(y, prob * 50, lty = 2)
```

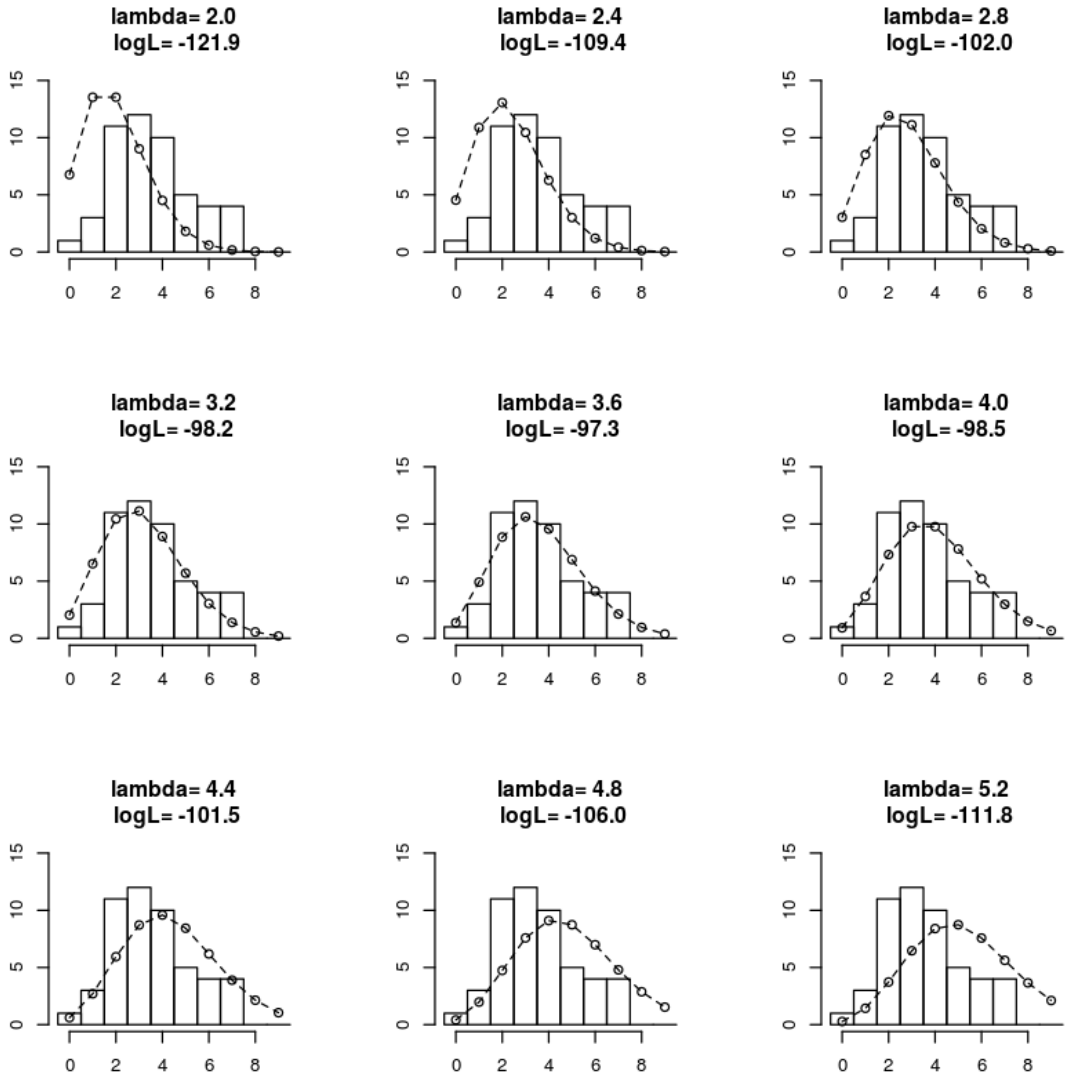
50



lambda の値ごとのlog likelihoodをみる

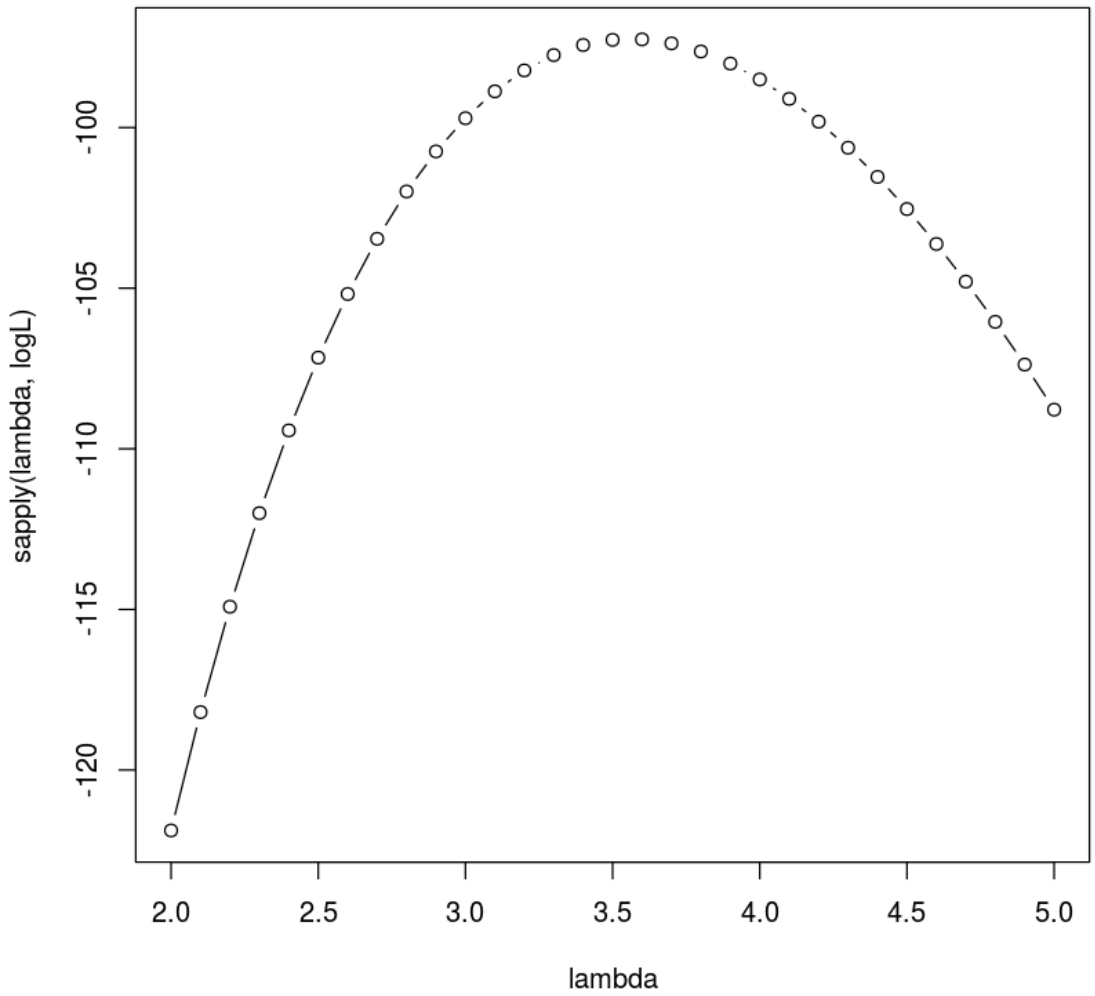
In [13]:

```
logL <- function(m) sum(dpois(data, m, log = TRUE))

plot.poisson <- function(lambda) {
  y <- 0:9
  prob <- dpois(y, lambda = lambda)

  hist(plant.seed, breaks = seq(-0.5, 9.5, 1), ylim = c(0, 15),
       main = "", xlab = "", ylab = "")
  points(y, prob * 50)
  lines(y,  prob * 50, lty = 2)

  title(sprintf("lambda= %.1f\n logL= %.1f", lambda, logL(lambda)))
}

layout(matrix(1:9, byrow = T, ncol = 3))
junk <- sapply(seq(2, 5.2, 0.4), plot.poisson)
```



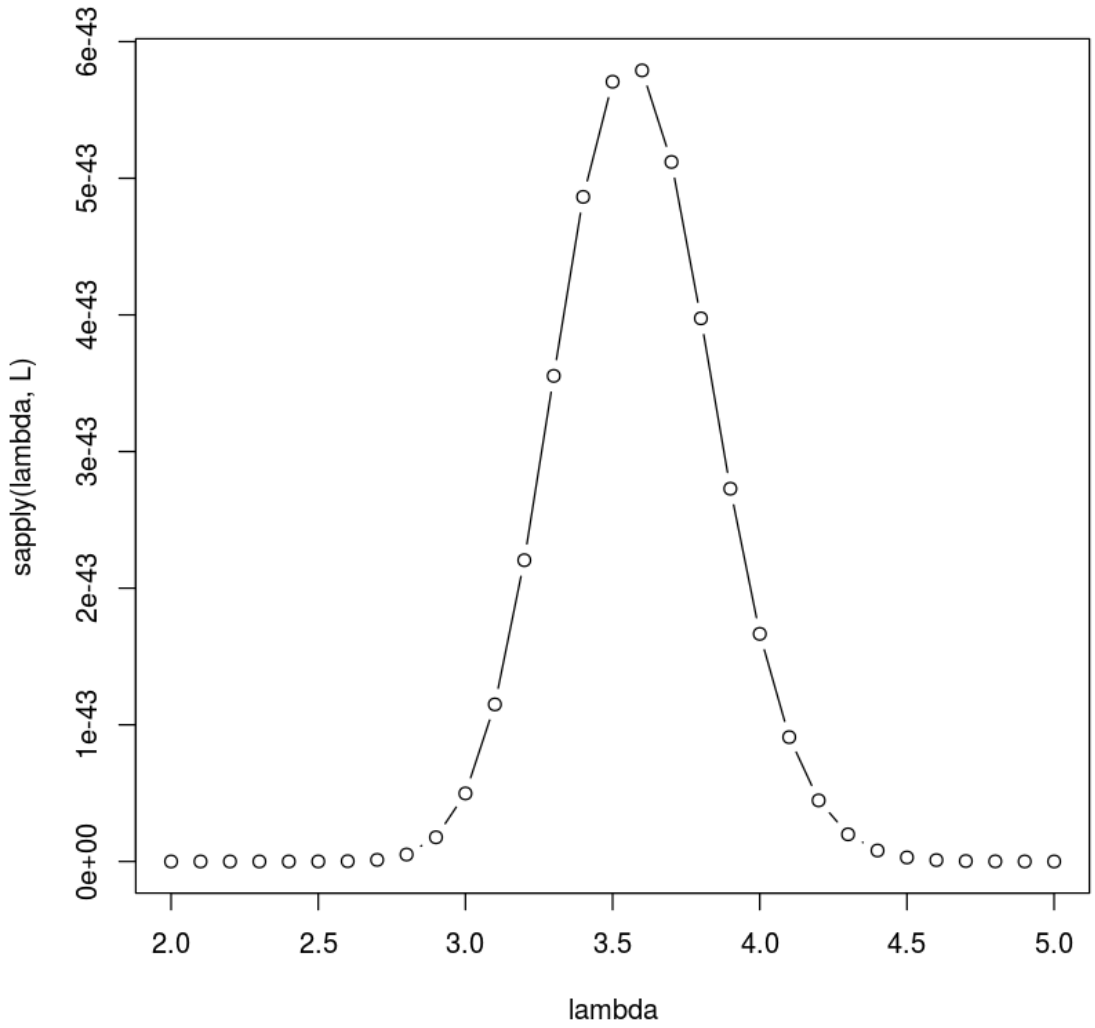
In [14]:

```
# Likelihood functionの形態をみる
logL <- function(m) sum(dpois(data, m, log = TRUE))
lambda <- seq(2, 5, 0.1)
plot(lambda, sapply(lambda, logL), type = "b")
```



In [15]:

```
L <- function(m) prod(dpois(data, m, log = FALSE))
plot(lambda, sapply(lambda, L), type = "b")
```



最尤法を用いるポアソン回帰を行って切片のexp()を求めると同じ値

In [16]:

```
fit.poisson <- glm(plant.seed ~ 1, family = poisson)
exp(coef(fit.poisson))
```

(Intercept): 3.5600000007512

3章 一般化線形モデル(GLM)

In [17]:

```
d <- read.csv(DIR + "data3a.csv")
```

In [18]:

```
d
```

y	x	f
6	8.31	C
6	9.44	C
6	9.50	C
12	9.07	C
10	10.16	C
4	8.32	C
9	10.61	C
9	10.06	C
9	9.93	C
11	10.43	C
6	10.36	C
10	10.15	C
6	10.92	C
10	8.85	C
11	9.42	C
8	11.11	C
3	8.02	C
8	11.93	C
5	8.55	C
5	7.19	C
4	9.83	C
11	10.79	C
5	8.89	C
10	10.09	C
6	11.63	C
6	10.21	C
7	9.45	C
9	10.44	C
3	9.44	C
10	10.48	C
⋮	⋮	⋮
10	10.54	T
8	11.30	T
8	12.40	T

y	x	f
7	10.18	T
5	9.53	T
6	10.24	T
8	11.76	T
9	9.52	T
9	10.40	T
6	9.96	T
7	10.30	T
10	11.54	T
6	9.42	T
11	11.28	T
11	9.73	T
11	10.78	T
5	10.21	T
6	10.51	T
4	10.73	T
5	8.85	T
6	11.20	T
5	9.86	T
8	11.54	T
5	10.03	T
9	11.88	T
8	9.15	T
6	8.52	T
8	10.24	T
7	10.86	T
9	9.97	T

In [19]:

```
d$x
```

```
8.31  9.44  9.5  9.07  10.16  8.32  10.61  10.06  9.93  10.43  10.36
10.15  10.92  8.85  9.42  11.11  8.02  11.93  8.55  7.19  9.83  10.79
8.89  10.09  11.63  10.21  9.45  10.44  9.44  10.48  9.43  10.32  10.33
8.5  9.41  8.96  9.67  10.26  10.36  11.8  10.94  10.25  8.74  10.46  9.37
9.74  8.95  8.74  11.32  9.25  10.14  9.05  9.89  8.76  12.04  9.91  9.84
11.87  10.16  9.34  10.17  10.99  9.19  10.67  10.96  10.55  9.69  10.91
9.6  12.37  10.54  11.3  12.4  10.18  9.53  10.24  11.76  9.52  10.4  9.96
10.3  11.54  9.42  11.28  9.73  10.78  10.21  10.51  10.73  8.85  11.2
9.86  11.54  10.03  11.88  9.15  8.52  10.24  10.86  9.97
```

In [20]:

```
d$y
```

```
6  6  6  12  10  4  9  9  9  11  6  10  6  10  11  8  3  8  5  5  4
11  5  10  6  6  7  9  3  10  2  9  10  5  11  10  4  8  9  12  8  9
8  6  6  10  10  9  12  6  14  6  7  9  6  7  9  13  9  13  7  8  10
7  12  6  15  3  4  6  10  8  8  7  5  6  8  9  9  6  7  10  6  11  11
11  5  6  4  5  6  5  8  5  9  8  6  8  7  9
```

In [21]:

```
d$f
```

```
C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C
C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C  C
C  C  C  C  C  C  C  C  T  T  T  T  T  T  T  T  T  T  T  T  T  T  T
T  T  T  T  T  T  T  T  T  T  T  T  T  T  T  T  T  T  T  T  T  T  T
T  T  T  T  T  T  T  T  T  T  T
```

In [22]:

```
class(d)
```

'data.frame'

In [23]:

```
class(d$y)
```

'integer'

In [24]:

```
class(d$x)
```

'numeric'

In [25]:

```
class(d$f)
```

'factor'

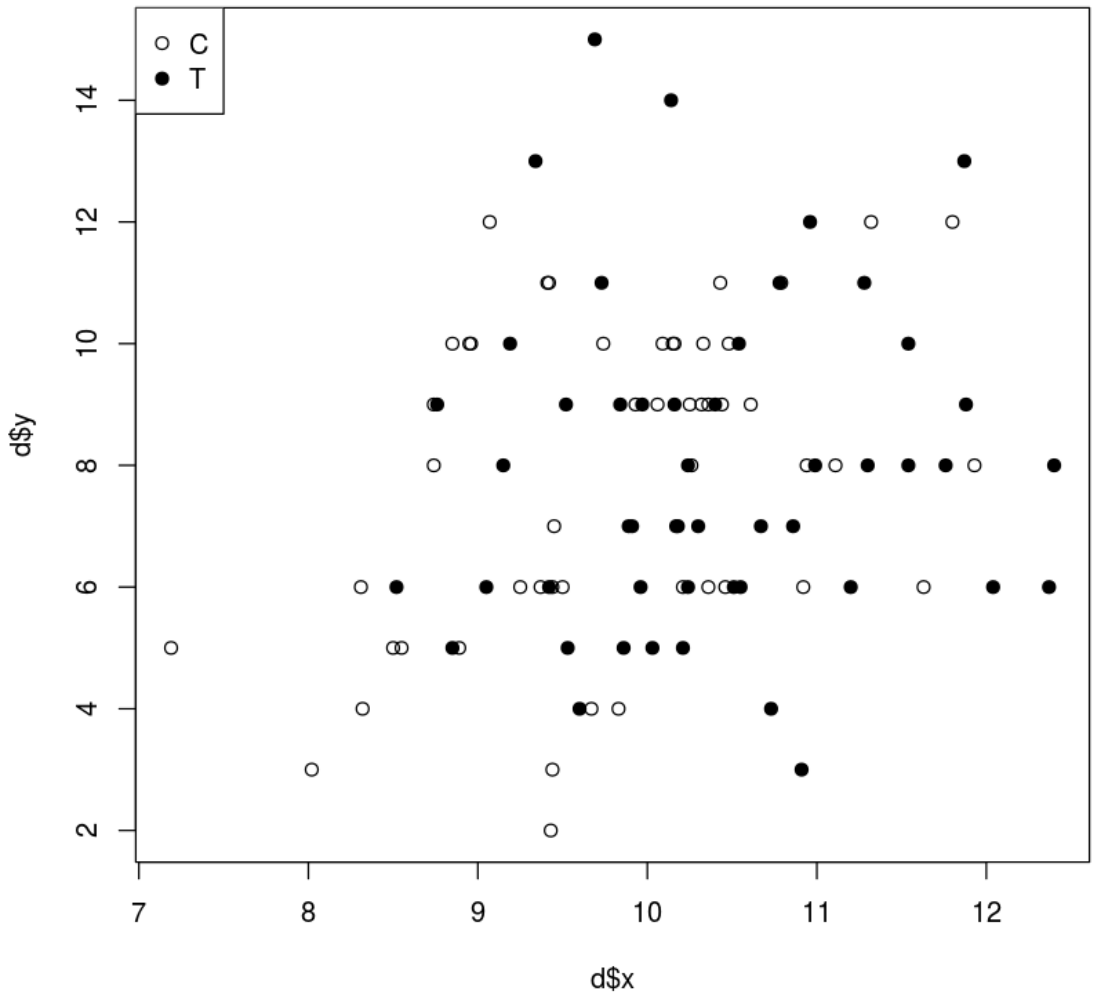
In [26]:

```
summary(d)
```

```
      y               x         f       
 Min.   : 2.00   Min.   : 7.190   C:50  
 1st Qu.: 6.00   1st Qu.: 9.428   T:50  
 Median : 8.00   Median :10.155         
 Mean   : 7.83   Mean   :10.089         
 3rd Qu.:10.00   3rd Qu.:10.685         
 Max.   :15.00   Max.   :12.400         
```

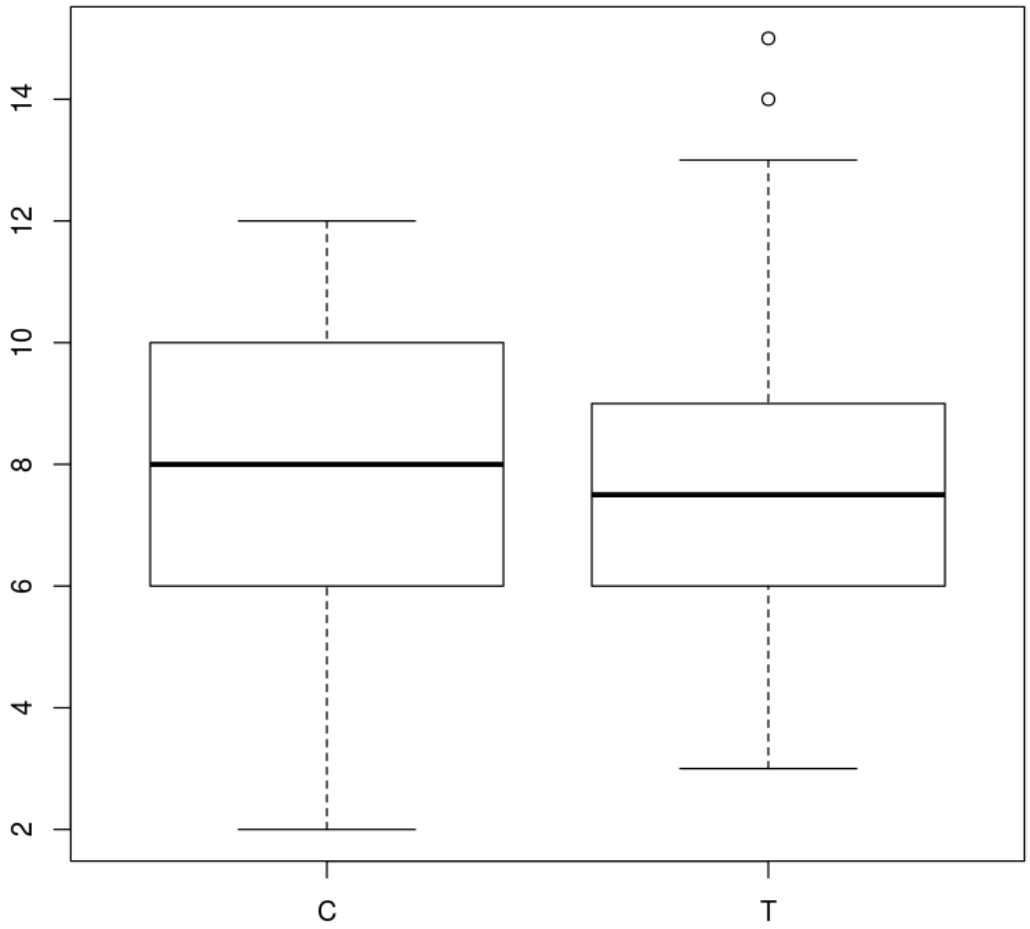
In [27]:

```
plot(d$x, d$y, pch=c(21,19)[d$f])
legend("topleft", legend=c("C", "T"), pch=c(21,19))
```



In [28]:

```
plot(d$f, d$y, pch=c(21,19)[d$f])
```



対数リンク関数を使って当てはめを行う。

In [29]:

```
fit <- glm(y ~ x, data = d, family = poisson(link = 'log'))
fit
summary(fit)
```

```
Call:  glm(formula = y ~ x, family = poisson(link = "log"), data = d)

Coefficients:
(Intercept)          x
    1.29172      0.07566

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:          89.51
Residual Deviance: 84.99     AIC: 474.8
```

```
Call:
glm(formula = y ~ x, family = poisson(link = "log"), data = d)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3679  -0.7348  -0.1775   0.6987   2.3760

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.29172    0.36369   3.552 0.000383 ***
x            0.07566    0.03560   2.125 0.033580 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 89.507  on 99  degrees of freedom
Residual deviance: 84.993  on 98  degrees of freedom
AIC: 474.77

Number of Fisher Scoring iterations: 4
```

Estimate によると、切片β_1は1.29, 傾きβ_2は0.076。 Std. Error (SE) はこの推定値がどのくらいばらつくかを表している。 対数尤度は最大尤度で最大値を取る凸関数。推定のばらつきが正規分布であると仮定し、さらに対数尤度関数は最大値付近での形が正規分布に近いと仮定すると（Wald統計量）、このように SE を求めることができる。 z value: Wald 統計量。最尤推定値を SE で除した値。推定値が 0 から離れているかの目安になる値。

logLik() で、最大尤度（＝あてはまりの良さ）を算出することができる。df は自由度 (degrees of freedom)。

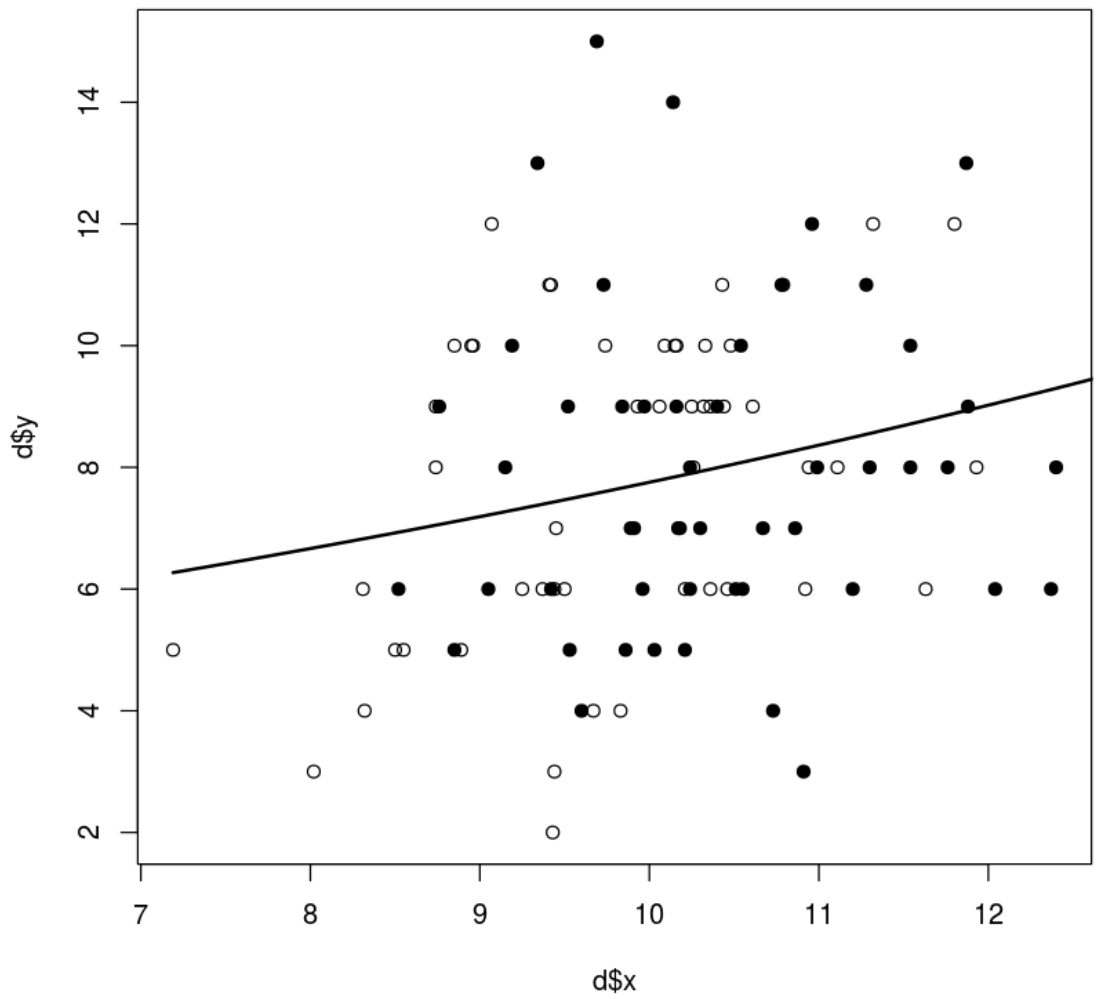
In [30]:

```
logLik(fit)
```

```
'log Lik.' -235.3863 (df=2)
```

In [31]:

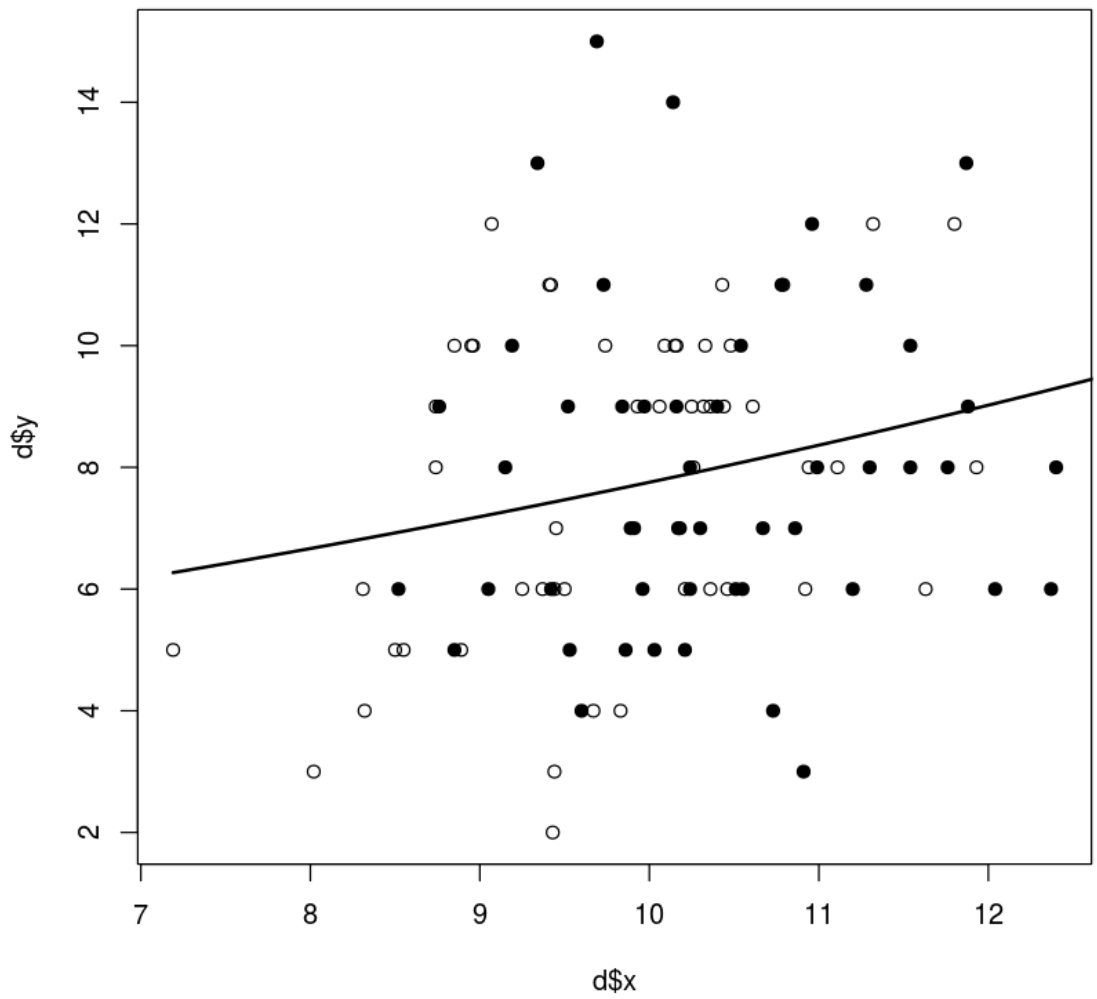
```
plot(d$x, d$y, pch=c(21,19)[d$f])
xx <- seq(min(d$x), max(d$y), length=100)
lines(xx, exp(1.29172 + 0.07566 * xx), lwd = 2)
```



predict() でも描ける。

In [32]:

```
plot(d$x, d$y, pch=c(21,19)[d$f])
yy <- predict(fit, newdata = data.frame(x = xx), type="response")
lines(xx, yy, lwd = 2)
```



では、施肥効果 f を使った場合は？

In [33]:

```
fit.f <- glm(y ~ f, data = d, family = poisson(link = 'log'))
summary(fit.f)
```

```
Call:
glm(formula = y ~ f, family = poisson(link = "log"), data = d)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.47515  -0.69941   0.04264   0.72467   2.25204

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.05156    0.05070  40.463   <2e-16 ***
fT           0.01277    0.07148   0.179    0.858
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 89.507  on 99  degrees of freedom
Residual deviance: 89.475  on 98  degrees of freedom
AIC: 479.25

Number of Fisher Scoring iterations: 4
```

わずかに、個体サイズ x だけを使った時より、誤差が大きくなっている。

In [34]:

```
logLik(fit.f)
```

```
'log Lik.' -237.6273 (df=2)
```

では、素性総動員ではどうか？
施肥効果は逆にマイナスになる。

In [35]:

```
fit.all <- glm(y ~ x + f, data = d, family = poisson(link = 'log'))
summary(fit.all)
```

```
Call:
glm(formula = y ~ x + f, family = poisson(link = "log"), data = d)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.3977  -0.7337  -0.2023   0.6795   2.4317

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.26311    0.36963   3.417 0.000633 ***
x            0.08007    0.03704   2.162 0.030620 *
fT          -0.03200    0.07438  -0.430 0.667035
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 89.507  on 99  degrees of freedom
Residual deviance: 84.808  on 97  degrees of freedom
AIC: 476.59

Number of Fisher Scoring iterations: 4
```

なんとか、誤差は一番小さくなっている。
でも、これが良いモデルとは限らない。（まだ「あてはまり」しか評価していない）

In [36]:

```
logLik(fit.all)
```

```
'log Lik.' -235.2937 (df=3)
```

4章 GLMのモデル選択

In [37]:

```
d <- read.csv('data3a.csv')
fit <- glm(y ~ x, data = d, family = poisson(link = 'log'))
fit
```

```
Call:  glm(formula = y ~ x, family = poisson(link = "log"), data = d)

Coefficients:
(Intercept)            x
    1.29172      0.07566

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:          89.51
Residual Deviance: 84.99     AIC: 474.8
```

この　Null Deviance　と Residual Deviance が何を示しているのか？ Null Deviance は、切片のみを用いた一定モデルの逸脱度（＝最大逸脱度）と、全データ点を追従したモデル（＝フルモデル）の逸脱度（＝最小逸脱度）の差を表している。 Residual Deviance は、与えられたモデルの逸脱度と、最小逸脱度の差を表している。

Given モデルの逸脱度

In [38]:

```
-2 * logLik(fit)
```

'log Lik.' 470.7725 (df=2)

フルモデルの逸脱度

In [39]:

```
-2 * sum(log(dpois(d$y, lambda = d$y)))
```

385.779505048992

In [40]:

```
470.7725 - 385.779505048992
```

84.992994951008

ほぼ Residual Deviance に一致していることがわかる。

一方、一定モデルの逸脱度は、

In [41]:

```
fit <- glm(y ~ 1, data = d, family = poisson(link = 'log'))
-2 * logLik(fit)
```

'log Lik.' 475.2864 (df=1)

In [42]:

```
475.2864 - 385.779505048992
```

89.506894951008

ほぼ Null Deviance に一致していることがわかる。

参考文献

データ解析のための統計モデリング入門: 一般化線形モデル・階層ベイズモデル・MCMC
http://hosho.ees.hokudai.ac.jp/~kubo/ce/IwanamiBook.html
(http://hosho.ees.hokudai.ac.jp/~kubo/ce/IwanamiBook.html)

一般化線形モデル（と ベイズ推定）
平成２７年 数理統計短期集合研修（応用編）
農環研・生物多様性研究領域, 山村光司
http://cse.naro.affrc.go.jp/yamamura/Images/kenshuu_slide_glm_2015_applied.pdf
(http://cse.naro.affrc.go.jp/yamamura/Images/kenshuu_slide_glm_2015_applied.pdf)

第7章 一般化線形混合モデル まとめ
https://ysk24ok.github.io/2017/01/05/midoribon_section7.html
(https://ysk24ok.github.io/2017/01/05/midoribon_section7.html)

「データ解析のための統計モデリング入門」読書ノート
7章 GLMMとGLMを比較する
http://yagays.github.io/blog/2012/11/02/glm-mcmc-chp7/ (http://yagays.github.io/blog/2012/11/02/glm-mcmc-chp7/)