MAST90104: Introduction to Statistical Learning

Assignment 4, 2019, Solutions

1. College juniors (undergraduates) at US universities were asked if they were "unlikely", "somewhat likely", or "very likely" to apply to graduate school. Data on whether the parents have graduate education (`pared`), whether the undergraduate institution is public (`public`), and current GPA (`gpa`), were also collected.

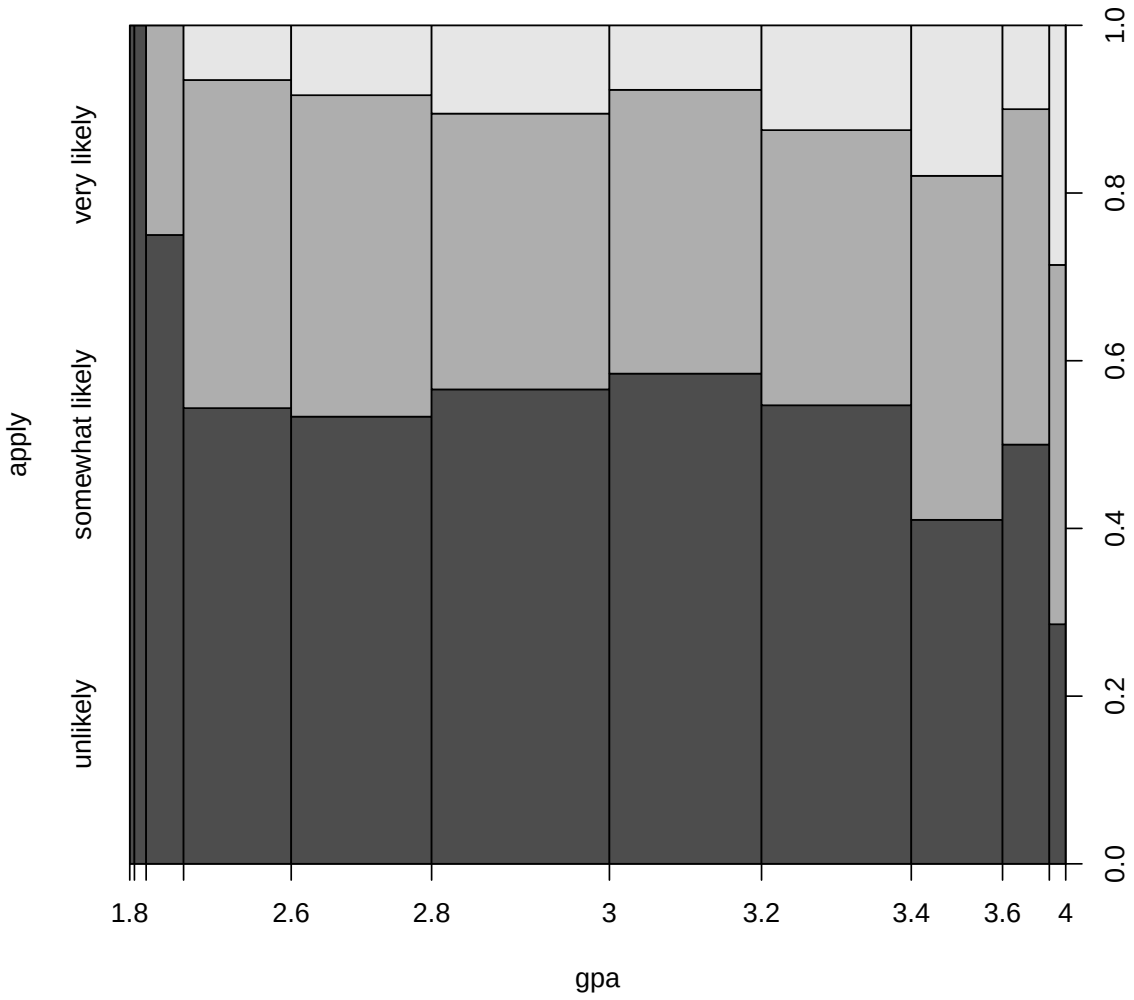
```
load("ologit.Rdata")
head(dat)
```

```
##             apply pared public  gpa
## 1     very likely     0      0 3.26
## 2 somewhat likely     1      0 3.21
## 3        unlikely     1      1 3.94
## 4 somewhat likely     0      0 2.81
## 5 somewhat likely     0      0 2.53
## 6        unlikely     0      1 2.59
```

```
ftable(xtabs(~ public + apply + pared, data = dat))
```

```
##                        pared   0   1
## public apply
## 0      unlikely              175  14
##        somewhat likely        98  26
##        very likely            20  10
## 1      unlikely               25   6
##        somewhat likely        12   4
##        very likely             7   3
```

```
plot(apply ~ gpa, data = dat)
```



(a) Fit a multinomial model to predict the probabilities for graduate application by category, "unlikely", "somewhat likely", or "very likely".

Solution:

```
library(nnet)
q3a_model1 = multinom(apply ~ pared*public*gpa, data = dat)
```

```
## # weights:  27 (16 variable)
## initial  value 439.444915
## iter  10 value 356.729468
## iter  20 value 350.788487
## iter  30 value 350.601860
## iter  40 value 350.590868
## iter  50 value 350.583458
## iter  60 value 350.582596
## final  value 350.582408
## converged
```

```
summary(q3a_model1)
```

```
## Call:
## multinom(formula = apply ~ pared * public * gpa, data = dat)
```

```
##
## Coefficients:
##                  (Intercept)     pared     public       gpa pared:public
## somewhat likely   -2.083894  4.481409 -5.5377440 0.5106691     5.456118
## very likely       -3.254924 -6.177962 -0.6065062 0.3699475     5.792676
##                  pared:gpa public:gpa pared:public:gpa
## somewhat likely -1.092795  1.6296648        -1.859499
## very likely      2.427065  0.4573643        -2.226742
##
## Std. Errors:
##                  (Intercept)    pared   public       gpa pared:public
## somewhat likely   0.9949367 2.983511 3.611404 0.3338763     7.806031
## very likely       1.8503088 5.211142 4.147947 0.6207361     9.965316
##                  pared:gpa public:gpa pared:public:gpa
## somewhat likely 0.9698168   1.109113         2.366396
## very likely     1.5971114   1.321666         2.948514
##
## Residual Deviance: 701.1648
## AIC: 733.1648

q3a_model2 <- step(q3a_model1)

## Start:  AIC=733.16
## apply ~ pared * public * gpa
##
## trying - pared:public:gpa
## # weights:  24 (14 variable)
## initial  value 439.444915
## iter  10 value 352.518781
## iter  20 value 351.033095
## iter  30 value 351.031368
## iter  30 value 351.031368
## final  value 351.031368
## converged
##                     Df      AIC
## - pared:public:gpa 14 730.0627
## <none>             16 733.1648
## # weights:  24 (14 variable)
## initial  value 439.444915
## iter  10 value 352.518781
## iter  20 value 351.033095
## iter  30 value 351.031368
## iter  30 value 351.031368
## final  value 351.031368
## converged
##
## Step:  AIC=730.06
## apply ~ pared + public + gpa + pared:public + pared:gpa + public:gpa
##
## trying - pared:public
## # weights:  21 (12 variable)
## initial  value 439.444915
## iter  10 value 353.710062
## iter  20 value 352.220325
## final  value 352.218611
## converged
## trying - pared:gpa
## # weights:  21 (12 variable)
```

```
## initial  value 439.444915
## iter  10 value 355.635036
## iter  20 value 354.982611
## final  value 354.982547
## converged
## trying - public:gpa
## # weights:  21 (12 variable)
## initial  value 439.444915
## iter  10 value 352.762475
## iter  20 value 352.000062
## final  value 351.999744
## converged
##                Df      AIC
## - public:gpa   12 727.9995
## - pared:public 12 728.4372
## <none>         14 730.0627
## - pared:gpa    12 733.9651
## # weights:  21 (12 variable)
## initial  value 439.444915
## iter  10 value 352.762475
## iter  20 value 352.000062
## final  value 351.999744
## converged
##
## Step:  AIC=728
## apply ~ pared + public + gpa + pared:public + pared:gpa
##
## trying - pared:public
## # weights:  18 (10 variable)
## initial  value 439.444915
## iter  10 value 354.101438
## iter  20 value 353.441487
## final  value 353.441474
## converged
## trying - pared:gpa
## # weights:  18 (10 variable)
## initial  value 439.444915
## iter  10 value 355.913353
## final  value 355.865699
## converged
##                Df      AIC
## - pared:public 10 726.8829
## <none>         12 727.9995
## - pared:gpa    10 731.7314
## # weights:  18 (10 variable)
## initial  value 439.444915
## iter  10 value 354.101438
## iter  20 value 353.441487
## final  value 353.441474
## converged
##
## Step:  AIC=726.88
## apply ~ pared + public + gpa + pared:gpa
##
## trying - public
## # weights:  15 (8 variable)
## initial  value 439.444915
```

```
## iter  10 value 354.812899
## final  value 354.741801
## converged
## trying - pared:gpa
## # weights:  15 (8 variable)
## initial  value 439.444915
## iter  10 value 357.012275
## final  value 356.996982
## converged
##             Df      AIC
## - public     8 725.4836
## <none>      10 726.8829
## - pared:gpa  8 729.9940
## # weights:  15 (8 variable)
## initial  value 439.444915
## iter  10 value 354.812899
## final  value 354.741801
## converged
##
## Step:  AIC=725.48
## apply ~ pared + gpa + pared:gpa
##
## trying - pared:gpa
## # weights:  12 (6 variable)
## initial  value 439.444915
## iter  10 value 358.535353
## final  value 358.535259
## converged
##             Df      AIC
## <none>       8 725.4836
## - pared:gpa  6 729.0705
```

```
summary(q3a_model2)
```

```
## Call:
## multinom(formula = apply ~ pared + gpa + pared:gpa, data = dat)
##
## Coefficients:
##                  (Intercept)     pared       gpa pared:gpa
## somewhat likely   -2.470841  5.780411 0.6293256 -1.559901
## very likely       -3.804172 -3.033925 0.6057269  1.332739
##
## Std. Errors:
##                  (Intercept)    pared       gpa pared:gpa
## somewhat likely   0.9319653 2.644279 0.3091534 0.8425412
## very likely       1.6081085 4.122495 0.5300470 1.2495708
##
## Residual Deviance: 709.4836
## AIC: 725.4836
```

```
q3a_model3 = multinom(apply ~ pared + public + gpa,data=dat)
```

```
## # weights:  15 (8 variable)
## initial  value 439.444915
## iter  10 value 357.012275
## final  value 356.996982
## converged
```

```
summary(q3a_model3)
```

```
## Call:
## multinom(formula = apply ~ pared + public + gpa, data = dat)
##
## Coefficients:
##                 (Intercept)     pared     public       gpa
## somewhat likely   -1.878955 0.9516492 -0.4188168 0.4487486
## very likely       -4.847763 1.3741555  0.3600661 0.9240376
##
## Std. Errors:
##                 (Intercept)     pared    public       gpa
## somewhat likely    0.863786 0.3170625 0.3432944 0.2902058
## very likely        1.449023 0.4221675 0.4434658 0.4741715
##
## Residual Deviance: 713.994
## AIC: 729.994
```

```
q3a_model4 = multinom(apply ~ pared + public + gpa + pared:gpa, data = dat)
```

```
## # weights:  18 (10 variable)
## initial  value 439.444915
## iter  10 value 354.101438
## iter  20 value 353.441487
## final  value 353.441474
## converged
```

```
summary(q3a_model4)
```

```
## Call:
## multinom(formula = apply ~ pared + public + gpa + pared:gpa,
##     data = dat)
##
## Coefficients:
##                 (Intercept)     pared     public       gpa pared:gpa
## somewhat likely   -2.622783  5.739196 -0.3916432 0.6963197 -1.543574
## very likely       -3.628411 -2.816939  0.3390075 0.5281235  1.263007
##
## Std. Errors:
##                 (Intercept)    pared    public       gpa pared:gpa
## somewhat likely   0.9446567 2.658737 0.3418006 0.3155827 0.8474836
## very likely       1.6134093 4.102775 0.4512468 0.5366104 1.2448418
##
## Residual Deviance: 706.8829
## AIC: 726.8829
```

```
anova(q3a_model1,q3a_model2,test="Chisq")
```

```
## Likelihood ratio tests of Multinomial Models
##
## Response: apply
##                      Model Resid. df Resid. Dev   Test    Df LR stat.
## 1 pared + gpa + pared:gpa       792   709.4836
## 2    pared * public * gpa       784   701.1648 1 vs 2     8 8.318787
##     Pr(Chi)
## 1
## 2 0.4029658
```

```
anova(q3a_model3,q3a_model4,test="Chisq")
```

```
## Likelihood ratio tests of Multinomial Models
##
```

```
## Response: apply
##                               Model Resid. df Resid. Dev   Test    Df
## 1             pared + public + gpa       792   713.9940
## 2 pared + public + gpa + pared:gpa       790   706.8829 1 vs 2     2
##   LR stat.    Pr(Chi)
## 1
## 2 7.111016 0.02856686
```

```
anova(q3a_model4,q3a_model2,test= "Chisq")
```

```
## Likelihood ratio tests of Multinomial Models
##
## Response: apply
##                               Model Resid. df Resid. Dev   Test    Df
## 1          pared + gpa + pared:gpa       792   709.4836
## 2 pared + public + gpa + pared:gpa       790   706.8829 1 vs 2     2
##   LR stat.   Pr(Chi)
## 1
## 2 2.600656 0.2724424
```

Accept with the additive model or the model with interaction tt pared:gpa.

[2]

(b) Repeat the analysis with an ordinal model. Comment on any differences.
Solution:

```
library(MASS)
        q3b_model1 = polr(apply ~ pared*public*gpa, data = dat)
        summary(q3b_model1)
```

```
##
## Re-fitting to get Hessian
```

```
## Call:
## polr(formula = apply ~ pared * public * gpa, data = dat)
##
## Coefficients:
##                     Value Std. Error t value
## pared             -0.6015     2.3661 -0.2542
## public            -2.5946     2.7115 -0.9569
## gpa                0.4416     0.3047  1.4495
## pared:public       3.5907     6.5266  0.5502
## pared:gpa          0.5871     0.7697  0.7627
## public:gpa         0.8548     0.8408  1.0166
## pared:public:gpa -1.3798     1.9674 -0.7013
##
## Intercepts:
##                              Value   Std. Error t value
## unlikely|somewhat likely      1.7070  0.9081     1.8797
## somewhat likely|very likely   3.8197  0.9254     4.1277
##
## Residual Deviance: 714.3552
## AIC: 732.3552
```

```
        q3b_model2 <- step(q3b_model1)
```

```
## Start:  AIC=732.36
## apply ~ pared * public * gpa
##
##                     Df    AIC
```

```
## - pared:public:gpa  1 730.85
## <none>                732.36
##
## Step:  AIC=730.85
## apply ~ pared + public + gpa + pared:public + pared:gpa + public:gpa
##
##                Df    AIC
## - pared:gpa     1 729.13
## - public:gpa    1 729.51
## - pared:public  1 730.56
## <none>            730.85
##
## Step:  AIC=729.13
## apply ~ pared + public + gpa + pared:public + public:gpa
##
##                Df    AIC
## - public:gpa    1 727.81
## - pared:public  1 728.60
## <none>            729.13
##
## Step:  AIC=727.81
## apply ~ pared + public + gpa + pared:public
##
##                Df    AIC
## - pared:public  1 727.02
## <none>            727.81
## - gpa           1 731.56
##
## Step:  AIC=727.02
## apply ~ pared + public + gpa
##
##          Df    AIC
## - public  1 725.06
## <none>      727.02
## - gpa     1 730.67
## - pared   1 740.60
##
## Step:  AIC=725.06
## apply ~ pared + gpa
##
##         Df    AIC
## <none>     725.06
## - gpa    1 728.79
## - pared  1 738.60
```

```
summary(q3b_model2)
```

```
##
## Re-fitting to get Hessian

## Call:
## polr(formula = apply ~ pared + gpa, data = dat)
##
## Coefficients:
##        Value Std. Error t value
## pared 1.0457     0.2656   3.937
## gpa   0.6042     0.2539   2.379
##
```

```
## Intercepts:
##                             Value  Std. Error t value
## unlikely|somewhat likely    2.1763 0.7671     2.8370
## somewhat likely|very likely 4.2716 0.7922     5.3924
##
## Residual Deviance: 717.0638
## AIC: 725.0638
```

```
        q3b_model3 = polr(apply ~ pared + public + gpa,data=dat)
        summary(q3b_model3)
```

```
##
## Re-fitting to get Hessian
```

```
## Call:
## polr(formula = apply ~ pared + public + gpa, data = dat)
##
## Coefficients:
##           Value Std. Error t value
## pared   1.04769     0.2658  3.9418
## public -0.05879     0.2979 -0.1974
## gpa     0.61594     0.2606  2.3632
##
## Intercepts:
##                             Value   Std. Error t value
## unlikely|somewhat likely     2.2039  0.7795     2.8272
## somewhat likely|very likely  4.2994  0.8043     5.3453
##
## Residual Deviance: 717.0249
## AIC: 727.0249
```

```
        q3b_model4 = polr(apply ~ pared + public + gpa + pared:gpa, data = dat)
        summary(q3b_model4)
```

```
##
## Re-fitting to get Hessian
```

```
## Call:
## polr(formula = apply ~ pared + public + gpa + pared:gpa, data = dat)
##
## Coefficients:
##              Value Std. Error t value
## pared      0.58745     2.1305  0.2757
## public    -0.06211     0.2984 -0.2081
## gpa        0.59181     0.2827  2.0938
## pared:gpa  0.14845     0.6819  0.2177
##
## Intercepts:
##                             Value   Std. Error t value
## unlikely|somewhat likely     2.1311  0.8463     2.5180
## somewhat likely|very likely  4.2279  0.8670     4.8764
##
## Residual Deviance: 716.9774
## AIC: 728.9774
```

```
        anova(q3b_model1,q3b_model2,test="Chisq")
```

```
## Likelihood ratio tests of ordinal regression models
##
## Response: apply
```

```
##                   Model Resid. df Resid. Dev   Test    Df LR stat.
## 1          pared + gpa       396   717.0638
## 2 pared * public * gpa       391   714.3552 1 vs 2     5 2.708574
##     Pr(Chi)
## 1
## 2 0.7448068
```

```
        anova(q3b_model3,q3b_model4,test="Chisq")
```

```
## Likelihood ratio tests of ordinal regression models
##
## Response: apply
##                               Model Resid. df Resid. Dev   Test    Df
## 1             pared + public + gpa       395   717.0249
## 2 pared + public + gpa + pared:gpa       394   716.9774 1 vs 2     1
##   LR stat.   Pr(Chi)
## 1
## 2 0.047445 0.8275703
```

```
        anova(q3b_model4,q3b_model2,test= "Chisq")
```

```
## Likelihood ratio tests of ordinal regression models
##
## Response: apply
##                               Model Resid. df Resid. Dev   Test    Df
## 1                       pared + gpa       396   717.0638
## 2 pared + public + gpa + pared:gpa       394   716.9774 1 vs 2     2
##     LR stat.   Pr(Chi)
## 1
## 2 0.08636135 0.9577383
```

The ordinal model and the multinomial with interaction term have similar AIC, but the interaction term is not needed in the ordinal model. It is simpler and possibly to be preferred. Also the coefficients are log's of odds ratios so their exponentials are easily interpretable.

[2]

(c) Under the ordinal model, what is the fitted probability that a student whose parents have graduate degrees and who went to a public university, with a gpa of 3.0, is very likely to undertake graduate study?
Solution:

```
predict(q3b_model2,newdata =
data.frame(pared = 1,public = 1, gpa = 3.0),
type = "probs")
```

```
##        unlikely somewhat likely     very likely
##       0.3357750       0.4684851       0.1957398
```

So the probability is 0.196.

[2]

2. The $t(3)$ distribution has pdf $p(x) = \frac{2}{\sqrt{3}\pi}\left(1+\frac{x^2}{3}\right)^{-2}$, $-\infty < x < \infty$, and the Cauchy$(\sqrt{3},0)$ distribution has pdf $g(y) = \frac{1}{\sqrt{3}\pi}\left(1+\frac{y^2}{3}\right)^{-1}$, $-\infty < y < \infty$.

(a) Suppose $U \stackrel{d}{=} U(0,1)$, and $Y = \sqrt{3}\tan(\pi(U-\frac{1}{2}))$. Show that $Y \stackrel{d}{=}$ Cauchy$(\sqrt{3},0)$.
Solution: The cdf of Y is

$$\begin{aligned} F_Y(y) &= P[Y \le y] = P[\sqrt{3}\tan(\pi(U-\frac{1}{2})) \le y] \\ &= P[U \le \frac{1}{2} + \pi^{-1}\arctan\frac{y}{\sqrt{3}}] = \frac{1}{2} + \pi^{-1}\arctan\frac{y}{\sqrt{3}}, \quad -\infty < y < \infty \end{aligned}$$

The pdf of Y is then

$$f_Y(y) = F'_Y(y) = \frac{1}{\sqrt{3}\pi}\left(1+\frac{y^2}{3}\right)^{-1}$$

which is the same as Cauchy$(\sqrt{3}, 0)$.

[2]

(b) Construct an A-R sampling algorithm (or a mixture of A-R and transformation algorithm) for generating random numbers from $t(3)$ by using the result of (a).

Solution: Since $p(x) = \frac{2}{\sqrt{3}\pi}\left(1+\frac{x^2}{3}\right)^{-2} = 2\left(1+\frac{x^2}{3}\right)^{-1} g(x) \le 2g(x)$, the following algorithm will generate random numbers from a $t(3)$ distribution:

1° Generate $U \sim U(0,1)$, and calculate $X = \sqrt{3}\tan(\pi(U-\frac{1}{2}))$.
2° Generate $Y \sim U(0, 2g(X))$ independently.
3° If $Y \le p(X)$ deliver X; otherwise go to 1°.

We can in fact make this a little more efficient. Note that if $V \sim U(0,1)$ then $2g(x)V \sim U(0, 2g(x))$ and $2g(x)V \le p(x)$ iff $V \le p(x)/(2g(x)) = \left(1+\frac{x^2}{3}\right)^{-1}$. Thus the following algorithm does the same as before, but instead of calculating $2g(x)$ and $p(x)$, it only has to calculate $\left(1+\frac{x^2}{3}\right)^{-1}$.

1° Generate $U \sim U(0,1)$, and calculate $X = \sqrt{3}\tan(\pi(U-\frac{1}{2}))$.
2° Generate $V \sim U(0,1)$ independently.
3° If $V \le 1/\left(1+X^2/3\right)$ deliver X; otherwise go to 1°.

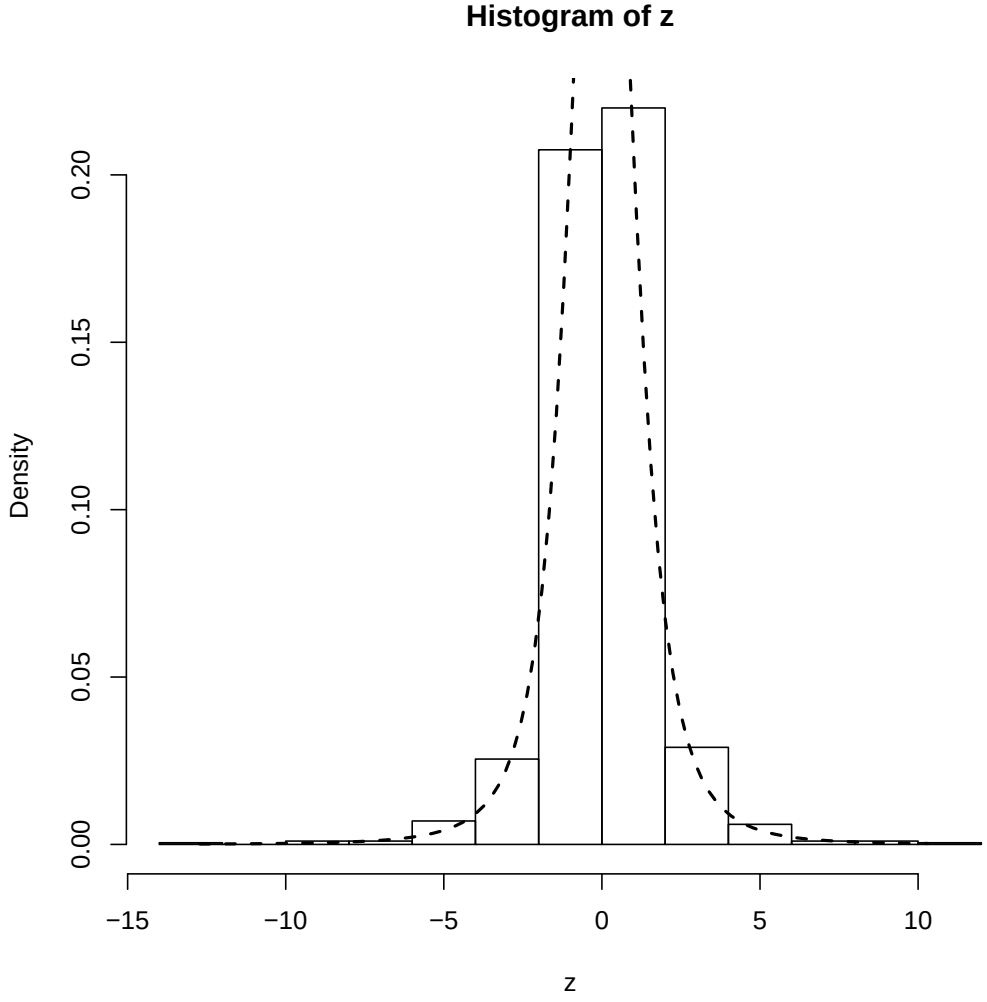
[2]

(c) Write an R function to implement (b). Then use it to generate a sample of 1000 numbers from $t(3)$, and compare the sample pdf curve with the actual $t(3)$ curve.

Solution:

```
t3sim <- function(){
# generate a t(3) random number.
u <- runif(1)
x <- sqrt(3.0)*tan(pi*(u-0.5))
v <- runif(1)
while(v > 1/(1+x^2/3)){
u <- runif(1)
x <- sqrt(3.0)*tan(pi*(u-0.5))
v <- runif(1)
}
return(x)
}

set.seed(3456)
n <- 1000
z <- rep(0, n)
for (i in 1:n) z[i] <- t3sim()
hist(z, freq=F)
curve(dt(x,3), from=min(z), to=max(z), add=T, lwd=2, lty=2)
```



[2]

3. The Dirichlet distribution is a multivariate generalisation of the beta distribution. It takes values in $\{\mathbf{x} = (x_1, \ldots, x_d) : x_i \in [0,1],\ \sum_i x_i = 1\}$, and, for $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d) \in \mathbb{R}_+^d$, has density

$$f(\mathbf{x}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{d} x_i^{\alpha_i - 1}, \text{ where } B(\boldsymbol{\alpha}) = \frac{\prod_i \Gamma(\alpha_i)}{\Gamma\left(\sum_i \alpha_i\right)}.$$

   (a) Prove that if $\mathbf{X} = (X_1, \ldots, X_d)$ has a Dirichlet distribution with parameter $\boldsymbol{\alpha}$ (we write $\mathbf{X} \sim \text{Dir}(\boldsymbol{\alpha})$), then $\mathbb{E}X_i = \alpha_i / \sum_i \alpha_i$. Hint: $\int \cdots \int f(\mathbf{x}) dx_1 \cdots dx_d = 1$.
   Solution:

$$\begin{aligned}
\mathbb{E}X_1 &= \int \cdots \int x_1 \frac{1}{B(\alpha_1, \ldots, \alpha_d)} \prod_{i=1}^{d} x_i^{\alpha_i - 1} dx_1 \cdots dx_d \\
&= \frac{B(\alpha_1 + 1, \ldots, \alpha_d)}{B(\alpha_1, \ldots, \alpha_d)} \int \cdots \int \frac{1}{B(\alpha_1 + 1, \ldots, \alpha_d)} x_1^{\alpha_1 + 1 - 1} \prod_{i=2}^{d} x_i^{\alpha_i - 1} dx_1 \cdots dx_d \\
&= \frac{B(\alpha_1 + 1, \ldots, \alpha_d)}{B(\alpha_1, \ldots, \alpha_d)} \\
&= \frac{\Gamma\left(\sum_i \alpha_i\right) \Gamma(\alpha_1 + 1) \prod_{i=2}^{d} \Gamma(\alpha_i)}{\prod_i \Gamma(\alpha_i) \Gamma\left(\alpha_1 + 1 + \sum_{i=2}^{d} \alpha_i\right)} \\
&= \frac{\alpha_1}{\sum_i \alpha_i} \text{ since } \Gamma(n) = (n-1)\Gamma(n-1).
\end{aligned}$$

[2]

(b) Show that the Dirichlet distribution is the conjugate prior for the multinomial. That is, if $\mathbf{p} \sim \text{Dir}(\boldsymbol{\alpha})$ and $\mathbf{X}|\mathbf{p} \sim \text{multinomial}(n, \mathbf{p})$, then $\mathbf{p}|\{\mathbf{X} = \mathbf{x}\} \sim \text{Dir}(\boldsymbol{\beta})$, where $\boldsymbol{\beta}$ depends on $\boldsymbol{\alpha}$ and $\mathbf{x}$.

Solution:

$$\begin{aligned} f(\mathbf{p}|\mathbf{x}) &\propto f(\mathbf{x}|\mathbf{p})f(\mathbf{p}) \\ &\propto \prod_i p_i^{x_i} \prod_i p_i^{\alpha_i - 1} \\ &= \prod_i p_i^{x_i + \alpha_i - 1} \end{aligned}$$

This is the kernel of a $\text{Dir}(x_1 + \alpha_1, \ldots, x_d + \alpha_d)$ random variable, as required.

[2]

(c) In 2003 Briggs, Ades and Price reported on a trial for the treatment of asthma. Patients received one of two treatments (seretide or fluticasone), and their status was monitored from week to week. Possible states were

STW Successfully treated week

UTW Unsuccessfully treated week

HEX Hospital managed exacerbation

PEX Primary-care managed exacerbation

TF Treatment failure (treatment ceased and patient removed from the trial)

For patients on seretide, the number of transitions from one state to another were

	To					
	STW	UTW	HEX	PEX	TF	
From						Total
STW	210	60	0	1	1	272
UTW	88	641	0	4	13	746
HEX	0	0	0	0	0	0
PEX	1	0	0	0	1	2
TF	0	0	0	0	81	81

The rows of this table can be considered as observations from independent multinomial random variables. Using Dir(1,...,1) priors, give Bayesian estimates (posterior means) for

$$p_{ij} = \mathbb{P}(\text{ state changes from } i \text{ to } j \text{ })$$

for $i = \text{STW}, \ldots, \text{PEX}$ and $j = \text{STW}, \ldots, \text{TF}$.

What prior would be appropriate for the transitions from state TF?

Solution: For transitions from STW we have prior $\text{Dir}(1, 1, 1, 1, 1)$ and posterior $\text{Dir}(211, 61, 1, 2, 2)$. The posterior means are thus 211/277, 61/277, 1/277, 2/277, 2/277.

We can proceed in the same way for the next three rows, however for the last row we know that TF is an absorbing state. Thus we can use the deterministic prior which gives probability 1 to the response TF, and probability 0 to all the others (if you like, you can think of this as a Dir(0,0,0,0,∞) prior). Thus we get estimated transition probabilities

	To				
	STW	UTW	HEX	PEX	TF
From					
STW	0.762	0.220	0.004	0.007	0.007
UTW	0.119	0.855	0.001	0.007	0.019
HEX	0.200	0.200	0.200	0.200	0.200
PEX	0.286	0.143	0.143	0.143	0.286
TF	0	0	0	0	1

[2]

4. There is a famous data set from RA Fisher's 1936 paper on *The use of multiple measurements in taxonomic problems* which consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres.

   The data is available in R in the data set `iris`.

   Perform a principal components and cluster analysis on the four variables using the R commands `prcomp` and `kmeans`. Use two separate methods to determine the number of clusters.

   Use the `ggplot2` and `ggfortify` libraries to do plots of the first two principal components that indicate where the directions of the original variables and the positions of the cluster means.

   Comment on your results.

   Solution: The following commands take the data set of the sepal and petal length and width, calculate the principal components for the standardised variables, give the proportions of total variance of the principal components contributed by each and give the weightings of each the original variables in the components. The option `retx` returns the values of the principal components in the column `x` of `irispc` - this is needed for computing the cluster means on the principal component scales.

```
library(ggplot2)
library(ggfortify)
irisscale <- scale(iris[c(1, 2, 3, 4)])
irispc <- prcomp(irisscale,retx = TRUE)
summary(irispc)
irispc$rotation
```

   The results show that the first two principal components account for 95.81% of the total variance. The first principal component has a high weight on petal length and smaller weights on sepal length and petal width, with little weight on sepal width. The second principal component has high weights on sepal length and width with lower weights on the petal dimensions.

```
## Importance of components%s:
##                             PC1    PC2     PC3     PC4
## Standard deviation     1.7084 0.9560 0.38309 0.14393
## Proportion of Variance 0.7296 0.2285 0.03669 0.00518
## Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
## Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
## Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
## Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
```

   The number of clusters can be decided using the commands in lectures. The plot of total within cluster sum of squares versus cluster size is given by the following commands:

```
library(cluster)
k.max <- 6 # Maximal number of clusters
wss <- sapply(1:k.max,
function(k){kmeans(irisscale, k, nstart=3 )$tot.withinss})
plot(1:k.max, wss,
type="b", pch = 19, frame = FALSE,
xlab="Number of clusters K",
ylab="Total within-clusters sum of squares")
abline(v = 3, lty =2)
```

   The results in Figure 1 show that there is a strong decrease up to 2 clusters with a smaller decrease to 3.

   The average silouhette plot of total is given by the following commands:

Figure 1: Total within cluster sum of squares vs. number of clusters

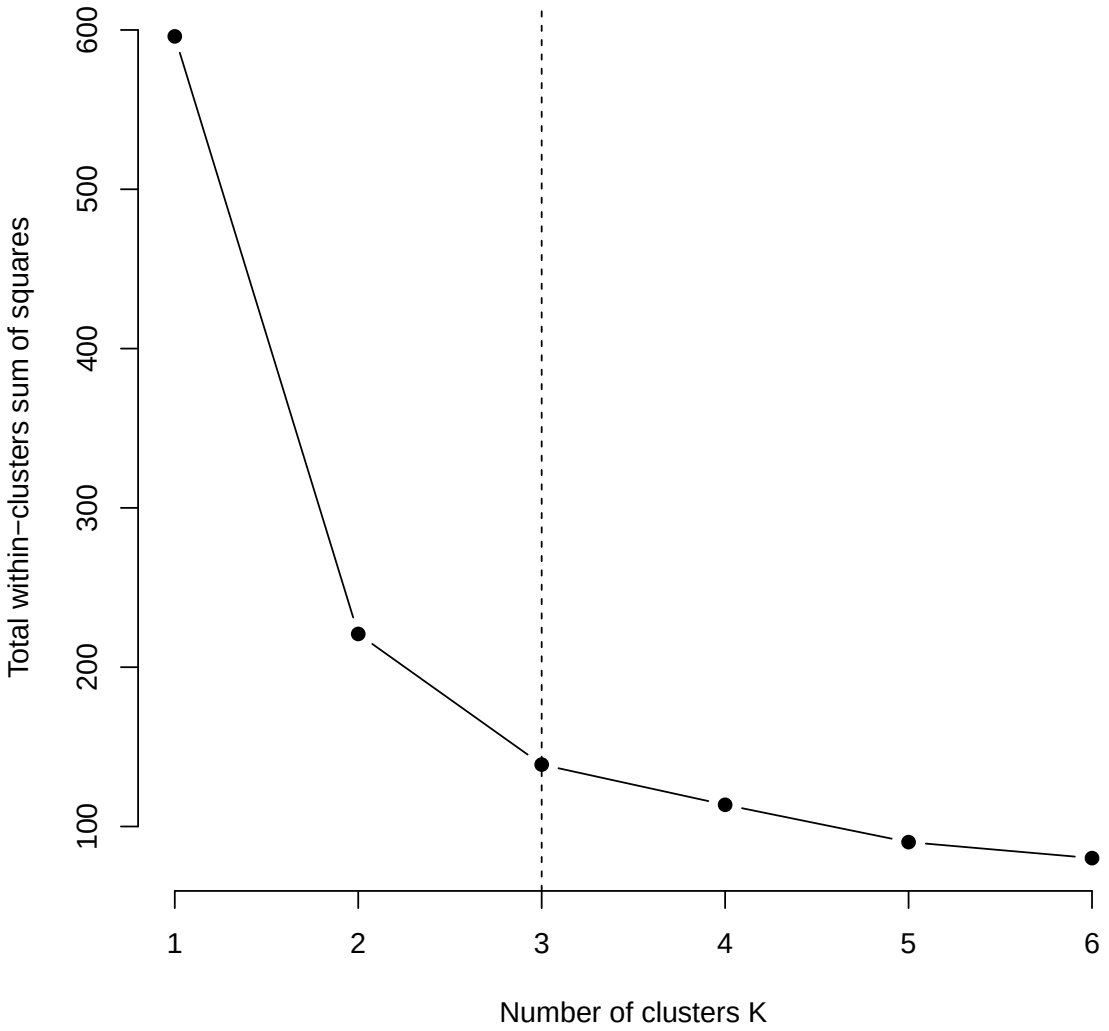
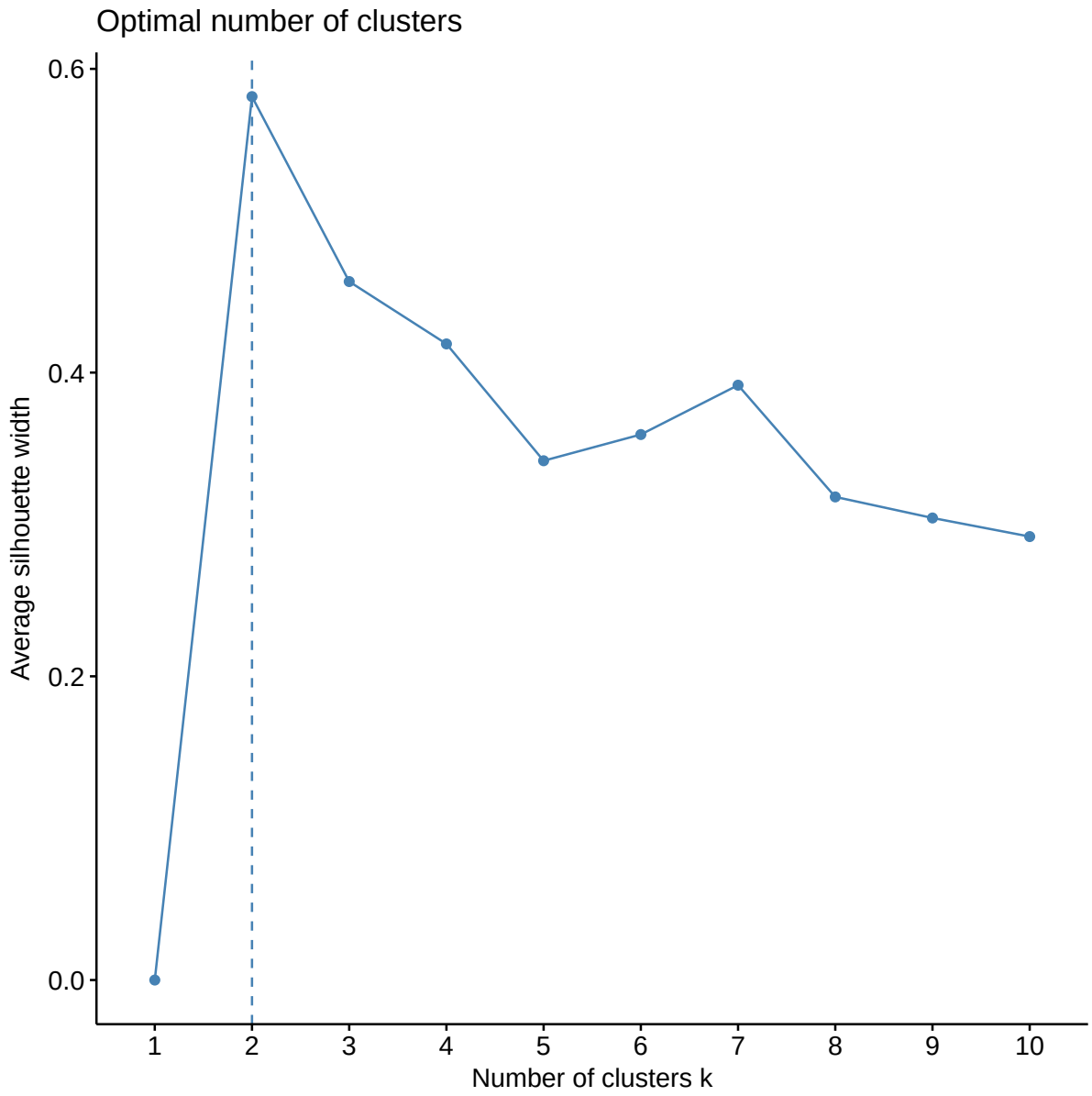


Figure 2: Average silohouette vs. number of clusters




```
library(factoextra)
library(NbClust)
library(fpc)
fviz_nbclust(irisscale, kmeans, method = c("silhouette"))
```

The maximum average silohouette shown in Figure 2 is for 2 clusters indicating that 2 is the optimal cluster number of clusters.

The two clusters are now found and plotted in Figure 3. The silohouette plot in Figure 4 shows that all observations have high silohouette values.

```
Kiris <- kmeans(irisscale,2)
autoplot(Kiris,data=irisscale)
dis = dist(GTYearScale)^2
sil = silhouette (KYear$cluster, dis)
plot(sil)
```

Figure 3: Plot of first two principal components with clusters

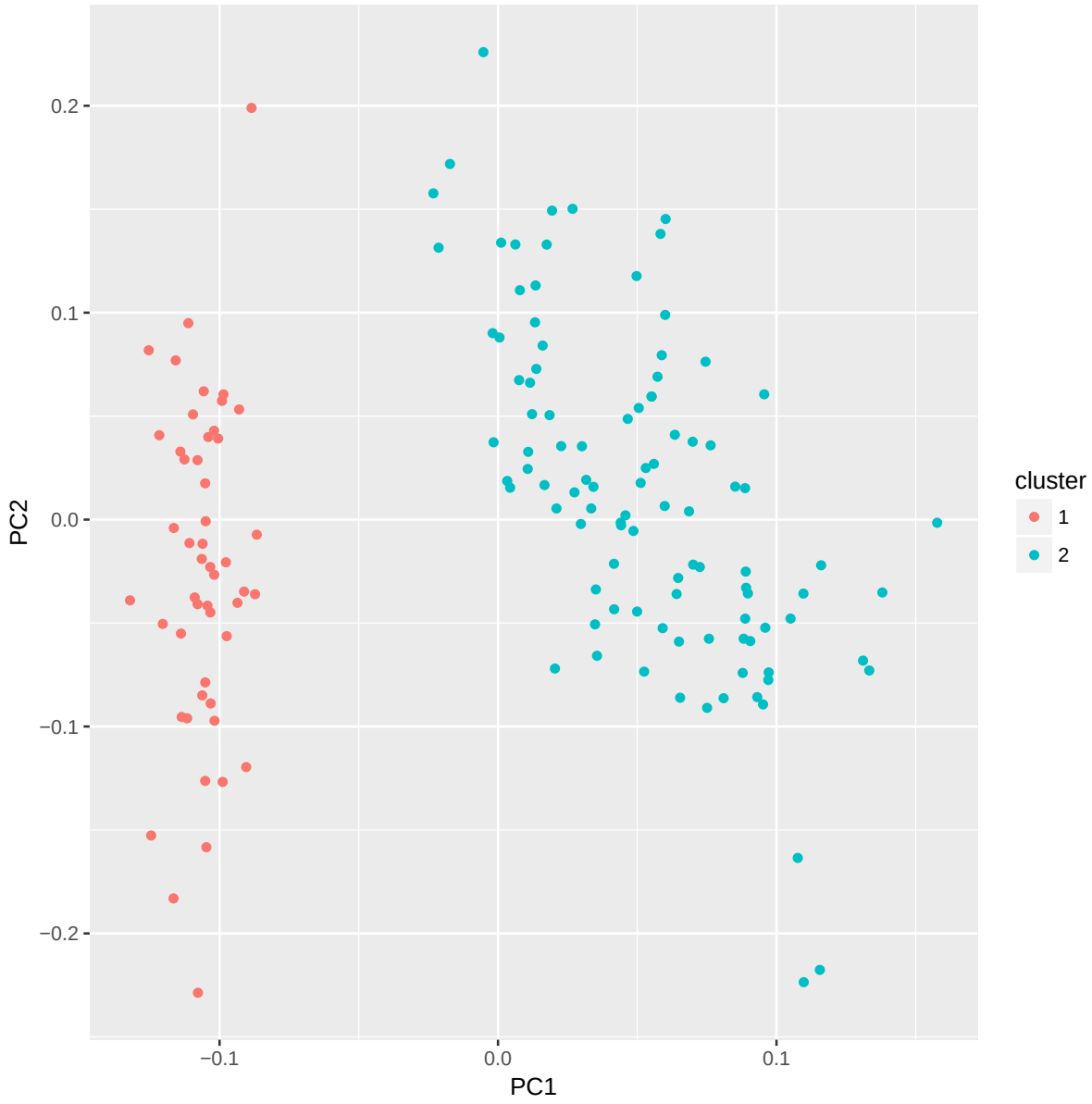
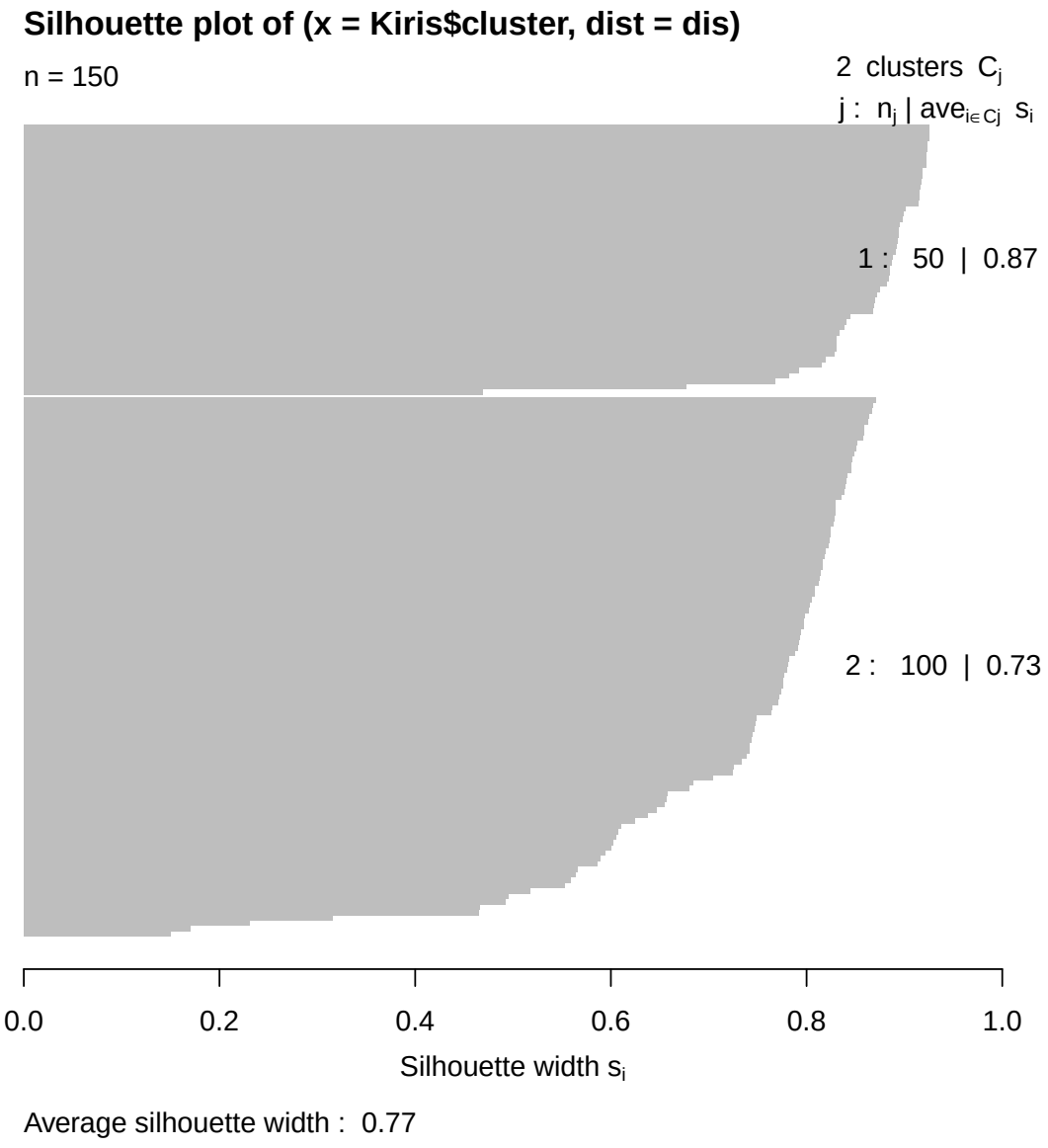


Figure 4: Silohouette plot



The question asks for a plot of the first two principal components against each other with the original variables identified by directions in the plot and the cluster centres identified. The `autoplot` command performs the plot of the principal components and the directions. The points for the cluster means and their identification are then added. Figure 5 has the result.

```
#  compute the cluster means by selecting the x values
# in the principal component outputs
x1 <- mean(irispc$x[,1][Kiris$cluster==1])
y1 <- mean(irispc$x[,2][Kiris$cluster==1])
x2 <- mean(irispc$x[,1][Kiris$cluster==2])
y2 <- mean(irispc$x[,2][Kiris$cluster==2])
# scale = 0 plots the principal components in the units of the data
autoplot(irispc, data = iris, colour = 'Species',
loadings = TRUE, loadings.colour = 'blue',
loadings.label = TRUE, loadings.label.size = 3,scale=0) +
# plot the cluster means and identifying text
geom_point(x = x1,y = y1,colour = "black") +
geom_text(label="cluster 1 mean",x = x1,y = y1,
colour = "black",size=3,hjust=-0.1,vjust=0.2) +
geom_point(x = x2,y = y2,colour = "black") +
geom_text(label="cluster 2 mean",x = x2,y = y2,
colour = "black",size=3,hjust=-0.1,vjust=0.2)
```

Here is the final requested plot:

[2]

Figure 5: Plot of first two principal components identifying species, original variable directions and cluster means