Solution proposals

Mikkel Meyer Andersen

2023-01-27

Possible topics to study

Exercise 1

1. Related to Section [Linear models]:
   - The orthoginal projection matrix onto the span of the model matrix X is $P = X(X^\top X)^{-1}X^\top$. The residuals are $r = (I - P)y$. From this one may verify that these are not all independent.
   - If one of the factors is ignored, then the model becomes a one-way analysis of variance model, at it is illustrative to redo the computations in Section [Linear models] in this setting.
   - Likewise if an interaction between the two factors is included in the model. What are the residuals in this case?

```
R> # Part 1: not all independent
R> nr <- 2
R> nc <- 2
R> y <- matrix_sym(nr, nc, "y")
R> dim(y) <- c(nr*nc, 1)
R> dat <- expand.grid(r=factor(1:nr), s=factor(1:nc))
R> X <- model.matrix(~r+s, data=dat) |> as_sym()
R> b <- vector_sym(ncol(X), "b")
R> mu <- X %*% b
R> XtX <- t(X) %*% X
R> XtXinv <- inv(XtX)
R> Xty <- t(X) %*% y
R> b_hat <- XtXinv %*% Xty
R> print(b_hat, rowvec = FALSE)
```

```
## [c]: [3*y11   y12   y21   y22]
##      [----- + --- + --- - ---]
##      [  4      4     4     4 ]
##      [                       ]
##      [  y11   y12   y21   y22]
##      [- --- - --- + --- + ---]
##      [   2     2     2     2 ]
##      [                       ]
##      [  y11   y12   y21   y22]
##      [- --- + --- - --- + ---]
##      [   2     2     2     2 ]
```

$$Cov(r) = Cov((I - P)y) = (I - P)vI(I - P)^T = (I - P)v(I - P) = (I - P)v. \tag{1}$$

$I - P$ is proportional to covariance matrix for residuals (and complementary projection of P), hence we see if $I - P$ is a diagonal matrix:

```
R> P <- X %*% inv(t(X) %*% X) %*% t(X)
R> IP <- eye(nrow(P), ncol(P)) - P
R> IP
```

```
## [c]: [1/4   -1/4  -1/4  1/4 ]
##      [                      ]
##      [-1/4  1/4   1/4   -1/4]
##      [                      ]
##      [-1/4  1/4   1/4   -1/4]
##      [                      ]
##      [1/4   -1/4  -1/4  1/4 ]
```

We see that it is not a diagonal matrix, hence there are non-zero covariances.

```
R> # Part 2: one-way analysis of variance
R>
R> X <- model.matrix(~r, data=dat) |> as_sym()
R> b <- vector_sym(ncol(X), "b")
R> mu <- X %*% b
R> XtX <- t(X) %*% X
R> XtXinv <- inv(XtX)
R> Xty <- t(X) %*% y
R> b_hat <- XtXinv %*% Xty
R> print(b_hat, rowvec = FALSE)
```

```
## [c]: [       y11   y12       ]
##      [       --- + ---       ]
##      [        2     2        ]
##      [                       ]
##      [  y11   y12   y21   y22]
##      [- --- - --- + --- + ---]
##      [   2     2     2     2 ]
```

```
R> # Part 3: interaction
R>
R> X <- model.matrix(~r*s, data=dat) |> as_sym()
R> b <- vector_sym(ncol(X), "b")
R> mu <- X %*% b
R> XtX <- t(X) %*% X
R> XtXinv <- inv(XtX)
R> Xty <- t(X) %*% y
R> b_hat <- XtXinv %*% Xty
R> print(b_hat, rowvec = FALSE)
```

```
## [c]: [         y11         ]
##      [                     ]
##      [     -y11 + y21      ]
##      [                     ]
##      [     -y11 + y12      ]
##      [                     ]
##      [y11 - y12 - y21 + y22]
```

```
R> X %*% b_hat - y
```

```
## [c]: [0  0  0  0]^T
```

Exercise 2

1. Related to Section [Logistic regression]:
   - In [Each component of the likelihood], Newton-Rapson can be implemented to solve the likelihood equations and compared to the output from `glm()`. Note how sensitive Newton-Rapson is to starting point. This can be solved by another optimisation scheme, e.g. Nelder-Mead (optimising the log likelihood) or BFGS (finding extreme for the score function).
   - The example is done as logistic regression with the logit link function. Try other link functions such as cloglog (complemantary log-log).

```
R> data(budworm, package = "doBy")
R> bud <- subset(budworm, sex == "male")
```

```
R> # Part 1: Newton-Rapson to solve the likelihood equations / compare to glm()
R> def_sym(y, n, p, s)
R> logLp_ <- y * log(p) + (n - y) * log(1 - p)
R> p_ <- exp(s) / (exp(s) + 1)
R> logLs_ <- subs(logLp_, p, p_)
R>
R> b <- vector_sym(2, "b")
R> x <- vector_sym(2, "x")
R> s_ <- sum(x * b)
R> logLb_ <- subs(logLs_, s, s_)
R>
R> nms <- c("x1", "x2", "y", "n")
R> logLb_list <- lapply(seq_len(nrow(bud)), function(r){
+   vls <- c(1, log2(bud$dose[r]), bud$ndead[r], bud$ntotal[r])
+   subs(logLb_, nms, vls)
+ })
R> logLb_total <- Reduce(`+`, logLb_list)
R> logLb_total_func <- as_func(logLb_total, vec_arg = TRUE)
```

```
R> Sb_ <- score(logLb_, b) |> simplify()
R> Sb_list <- lapply(seq_len(nrow(bud)), function(r){
+   vls <- c(1, log2(bud$dose[r]), bud$ndead[r], bud$ntotal[r])
+   subs(Sb_, nms, vls)
+ })
R> Sb_total <- Reduce(`+`, Sb_list)
R> Sb_total_func <- as_func(Sb_total, vec_arg = TRUE)
R>
R> Hb_ <- hessian(logLb_, b) |> simplify()
R> Hb_list <- lapply(seq_len(nrow(bud)), function(r){
+   vls <- c(1, log2(bud$dose[r]), bud$ndead[r], bud$ntotal[r])
+   subs(Hb_, nms, vls)
+ })
R> Hb_total <- Reduce(`+`, Hb_list)
R> Hb_total_func <- as_func(Hb_total, vec_arg = TRUE)
```

```
R> # Maximising f
R> objf <- function(x) {
+   logLb_total_func(x)
+ }
R> of <- optim(c(0, 0), objf, control = list(fnscale = -1))
R>
R> # Finding roots of g = \nabla f
R> objg <- function(x) {
+   norm(Sb_total_func(x), type = "2")
+ }
R> og <- optim(c(0, 0), objg)
R>
R> of$par
```

```
## [1] -2.82  1.26
```

```
R> og$par
```

```
## [1] -2.82  1.26
```

```
R> m <- glm(cbind(ndead, ntotal - ndead) ~ log2(dose), bud, family = binomial())
R> m
```

```
##
## Call:  glm(formula = cbind(ndead, ntotal - ndead) ~ log2(dose), family = binomial(),
##     data = bud)
##
## Coefficients:
## (Intercept)   log2(dose)
##       -2.82         1.26
##
## Degrees of Freedom: 5 Total (i.e. Null);  4 Residual
## Null Deviance:       71.1
## Residual Deviance: 1.88  AIC: 20.2
```

```
R> cfs <- m |> coef() |> unname()
R> of$par - cfs
```

```
## [1] -0.000358  0.000193
```

```
R> onr <- c(0, 0) # Note fragile to starting points, try e.g. c(1, 1)
R> for (iter in 1:7) {
+   Hinv <- solve(Hb_total_func(onr))
+   newnr <- onr - Hinv %*% Sb_total_func(onr)
+   print(newnr)
+   print(norm(newnr - onr, type = "2"))
+   onr <- newnr
+ }
```

```
##         [,1]
## [1,] -1.848
## [2,]  0.806
## [1] 2.02
##       [,1]
## [1,] -2.56
## [2,]  1.13
## [1] 0.78
##       [,1]
## [1,] -2.79
## [2,]  1.25
## [1] 0.265
##       [,1]
## [1,] -2.82
## [2,]  1.26
## [1] 0.0265
##       [,1]
## [1,] -2.82
## [2,]  1.26
## [1] 0.00024
##       [,1]
## [1,] -2.82
## [2,]  1.26
## [1] 1.97e-08
##       [,1]
## [1,] -2.82
## [2,]  1.26
## [1] 0
```

Others:

```
R> optim(c(0, 0), objf, control = list(fnscale = -1))
```

```
## $par
## [1] -2.82  1.26
```

```
##
## $value
## [1] -48.1
##
## $counts
## function gradient
##       69       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```
R> optim(c(0, 0), objf, method = "BFGS", control = list(fnscale = -1), gr = Sb_total_func)
```

```
## $par
## [1] -2.82  1.26
##
## $value
## [1] -48.1
##
## $counts
## function gradient
##       31        9
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Exercise 3

FIXME: An -> Another?

1. Related to Section [Maximum likelihood under constraints]:
   - Identifiability of the parameters was handled by not including r_1 and s_1 in the specification of p_{ij}. An alternative is to impose the restrictions $r_1 = 1$ and $s_1 = 1$, and this can also be handled via Lagrange multipliers. Another alternative is to regard the model as a log-linear model where $\log p_{ij} = \log u + \log r_i + \log s_j = \tilde{u} + \tilde{r}_i + \tilde{s}_j$. This model is similar in its structure to the two-way ANOVA for Section [Linear models].
   - This model can be fitted as a generalized linear model with a Poisson likelihood and log as link function. Hence, one may modify the results in Section [Logistic regression] to provide an alternative way of fitting the model.
   - A simpler task is to consider a multinomial distribution with three categories, counts y_i and cell probabilities p_i, $i = 1, 2, 3$ where $\sum_i p_i = 1$. For this model, find the maximum likelihood estimate for p_i.

```
R> nr <- 2
R> nc <- 2
R> dat <- expand.grid(r=factor(1:nr), s=factor(1:nc))
R> X <- model.matrix(~r+s, data=dat) |> as_sym()
R> y_val <- c(2, 4, 5, 19)
```

```
R> XX <- as.data.frame(as_expr(X))
R> ddd <- cbind(y = as_expr(y_val), XX)
R> fit <- glm(y~-1+V1+V2+V3, family=poisson, data=ddd)
R> fit
```

```
##
## Call:  glm(formula = y ~ -1 + V1 + V2 + V3, family = poisson, data = ddd)
##
## Coefficients:
##    V1     V2     V3
## 0.336  1.190  1.386
##
## Degrees of Freedom: 4 Total (i.e. Null);  1 Residual
## Null Deviance:       89.8
## Residual Deviance: 0.395     AIC: 20.5
```

```
R> predlogn <- predict(fit, type = "link")
R> predn <- exp(predlogn)
R> predp <- matrix(predn / sum(predn), nrow = nr, ncol = nc)
```

From paper:

```
R> y_ <- c("y_11", "y_21", "y_12", "y_22")
R> y  <- as_sym(y_)
R> def_sym(u, r2, s2, lambda_)
R> p <- as_sym(c("u", "u*r2", "u*s2", "u*r2*s2"))
R> logL  <- sum(y * log(p))
R> Lag  <- -logL + lambda_ * (sum(p) - 1)
R> vars <- list(u, r2, s2, lambda_)
R> gLag <- der(Lag, vars)
R> sol <- solve_sys(gLag, vars)
R> sol <- sol[[1]]
R> p11 <- sol$u
R> p21 <- sol$u * sol$r2
R> p12 <- sol$u * sol$s2
R> p22 <- sol$u * sol$r2 * sol$s2
R> p.hat <- matrix_(c(p11, p21, p12, p22), nrow = 2)
R> p.hat0 <- subs(p.hat, y_, y_val) |> as_expr()
R> p.hat0
```

```
##        [,1]  [,2]
## [1,] 0.0467 0.187
## [2,] 0.1533 0.613
```

```
R> predp - p.hat0
```

```
##          [,1]      [,2]
## [1,] 2.01e-16  5.55e-17
## [2,] 4.16e-16 -5.55e-16
```

Alternatively, with explicit likelihood and `optim()`:

```
R> N <- nrow(X)
R> q <- ncol(X)
R>
R> y <- vector_sym(N, "y")
R> b <- vector_sym(q, "b")
R> #m <- vector_sym(N, "m")
R> s <- vector_sym(N, "s")
R>
R> ## log-likelihood as function of m
R> #logLm  <- sum(y * log(m))
R> #logLm  <- sum(y * m - exp(m))
```

```
R> ## log-likelihood as function of s
R> #m_ <- exp(s)
R> #logLs <- subs(logLm, m, m_)
```

```
R> logLs  <- sum(y*s - exp(s))
R>
R> ## linear predictor as function of regression coefficients:
R> s_  <- X %*% b
R> ## log-Likelihood as function of regression coefficients:
R> logLb <- subs(logLs, s, s_)
R>
R> logLb_ <- subs(logLb, y, y_val)
R> logLb_func <- as_func(logLb_, vec_arg = TRUE)
```

```
R> o <- optim(c(0, 0, 0), logLb_func, control = list(fnscale = -1))
R> o$par
```

```
## [1] 0.336 1.190 1.386
```

```
R> coef(fit) - o$par
```

```
##        V1        V2        V3
##  3.52e-05 -9.02e-05  4.49e-05
```

Exercise 4

1. Related to Section [Auto regressive models]: Find (approximate) standard error and confidence interval for the parameter a. Modify the model in Equation @ref(eq:ar1) by setting $x_1 = ax_n + e_1$ and see what happens to the pattern of zeros in the concentration matrix. Extend the $AR(1)$ model to and $AR(2)$ model and investigate this model along the same lines.

Exercise 5

1. Related to Section [Variance of the average of correlated data]: It is illustrative to study such behaviours for other covariance functions. For example, an auto correlation (AR(1)) structure as studied in Section [Auto regressive models] where $\mathbf{Cov}(x_i, x_j) = vr^{|i-j|}$ and. Alternatively, one may study a covariance structure where $\mathbf{Cov}(x_i, x_i) = v$, $\mathbf{Cov}(x_i, x_j) = vr$ if $|i - j| = 1$ and $\mathbf{Cov}(x_i, x_j) = 0$ if $|i - j| > 1$.