MA 589 Project 1 Solutions

Luis Carvalho

1

(a)

```
inv.upper.tri <- function (R, transpose = FALSE)
  backsolve(R, diag(nrow(R)), transpose = transpose)
```

(b)

```
norm2 <- function (v) {
  m <- max(abs(v))
  ifelse(m == 0, 0, m * sqrt(sum((v / m) ^ 2)))
}
u <- 1e200 * rep(1, 100); norm2(u) # test
```

```
## [1] 1e+201
```

(c)

```
normalize.cols <- function (A) sweep(A, 2, apply(A, 2, norm2), `/`)
```

(d)

```
proj <- function (a, u) {
  m <- max(abs(u), abs(a))
  u <- u / m
  ifelse(m == 0, 0, sum(u * a) / sum(u * u)) * u
}
proj(1:100, u) # test
```

```
##   [1] 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5
##  [15] 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5
##  [29] 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5
##  [43] 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5
##  [57] 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5
##  [71] 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5
##  [85] 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5 50.5
##  [99] 50.5 50.5
```

(e)

```
vandermonde <- function (a, d) outer(a, 0:d, `^`)
```

2

(a)

```
eps <- local({ # closure on epsilon (optional)
  epsilon <- 1
  function () {
    while (1 + epsilon / 2 > 1) epsilon <<- epsilon / 2
    epsilon
  }
})
```

(b–d)

```
LOGEPS <- log(eps() / 2)
log1pe <- function (x) { # vectorized version: `x` can be a vector
  l <- ifelse(x > 0, x, 0) # shift
  x <- ifelse(x > 0, -x, x) # range reduction: `x = -abs(x)`
  ifelse(x < LOGEPS, l, l + log(1 + exp(x)))
}
c(log1pe(0), log1pe(-80), log1pe(800)) # test
```

```
## [1]   0.6931472   0.0000000 800.0000000
```

3

(a)

By the construction of Q via Gram-Schmidt we can see that $\mathbf{u}_i^\top \mathbf{u}_j = \delta_{ij}\|\mathbf{u}_i\|^2$ with δ_{ij} the Kronecker delta ($\delta_{ii} = 1$ and $\delta_{ij} = 0$ for $i \neq j$). In general,

$$C_{ki} = \mathbf{q}_k^\top \mathbf{a}_i = \frac{1}{\|\mathbf{u}_k\|}\mathbf{u}_k^\top \left(\mathbf{u}_i + \sum_{j=1}^{i-1} \frac{\mathbf{u}_j^\top \mathbf{a}_i}{\|\mathbf{u}_j\|^2}\mathbf{u}_j \right) = \delta_{ki}\|\mathbf{u}_k\| + \sum_{j=1}^{i-1} \delta_{kj}\mathbf{q}_j^\top \mathbf{a}_i$$

and so $C_{ki} = 0$ for $k > i$, $C_{kk} = \|\mathbf{u}_k\| > 0$, and $C_{ki} = \mathbf{q}_k^\top \mathbf{a}_i$ for $k < i$. Thus, since C is upper triangular, $A^\top A = (QC)^\top (QC) = C^\top C$, and the diagonal entries of C are positive, C is also the Cholesky factor of $A^\top A$.

(b)

```
vandermonde.Q <- function (x, d) { # from Gram-Schmidt orthogonalization
  U <- matrix(nrow = length(x), ncol = d + 1)
```

```
  U[, 1] <- a <- rep(1, length(x))
  for (i in 2:(d + 1)) {
    U[, i] <- a <- a * x
    for (j in 1:(i - 1))
      U[, i] <- U[, i] - proj(a, U[, j])
  }
  normalize.cols(U)
}
```

(c)

```
decompress.Q <- function (x, d, eta, alpha) {
  U <- matrix(nrow = length(x), ncol = d + 1)
  U[, 1] <- rep(1, length(x))
  U[, 2] <- x - alpha[1]
  for (i in 2:d)
    U[, i + 1] <- (x - alpha[i]) * U[, i] - eta[i + 1] / eta[i] * U[, i - 1]
  normalize.cols(U)
}
```

(d)

Since $\mathbf{u}_1 = 1_n$, $\eta_2 = 1_n^\top 1_n = n$, $\alpha_1 = 1_n^\top \mathrm{Diag}(\mathbf{x}) 1_n / (1_n^\top 1_n) = \sum_{i=1}^n x_i/n = \bar{x}$, and because $\mathbf{u}_2 = \mathbf{x} - \bar{x}1_n$, $\eta_3 = \mathbf{u}_2^\top \mathbf{u}_2 = \sum_{i=1}^n (x_i - \bar{x})^2 = (n-1)s_x^2$.

```
compress.Q <- function (x, d) {
  alpha <- numeric(d); eta <- numeric(d + 2)
  eta[1] <- 1
  U <- matrix(nrow = length(x), ncol = d + 1)
  U[, 1] <- a <- rep(1, length(x))
  eta[2] <- sum(U[, 1] ^ 2)
  alpha[1] <- sum(x * U[, 1] ^ 2) / eta[2]
  for (i in 2:(d + 1)) {
    U[, i] <- a <- a * x
    for (j in 1:(i - 1))
      U[, i] <- U[, i] - proj(a, U[, j])
    eta[i + 1] <- sum(U[, i] ^ 2)
    alpha[i] <- sum(x * U[, i] ^ 2) / eta[i + 1]
  }
  list(eta = eta, alpha = alpha)
}
```

4

(a)

If $H_0 : \beta_z = 0$ for a contiguous set of indices $z = \{j, j+1, \ldots, p\}$, define the complement $[-z] = \{1, \ldots, j-1\}$ and consider the partition $\beta = (\beta_{[-z]}, \beta_z)$. Then

$$R\beta = \begin{bmatrix} R_{[-z],[-z]} & R_{[-z],z} \\ 0 & R_{z,z} \end{bmatrix} \begin{bmatrix} \beta_{[-z]} \\ \beta_z \end{bmatrix} = \begin{bmatrix} \gamma_{[-z]} \\ \gamma_z \end{bmatrix},$$

where $R_{z,z}$ is also upper triangular by the structure of R and full rank, and so $R_{z,z}\beta_z = \gamma_z$ implies $\beta_z = 0$ is equivalent to $\gamma_z = 0$.

(b)

The MLE is $\hat{\gamma} = (Q^\top Q)^{-1} Q^\top \mathbf{y} = Q^\top \mathbf{y}$ because Q has orthogonal columns ($Q^\top Q = I_p$). Moreover, $\hat{\gamma}$ is normally distributed and has variance $\sigma^2 (Q^\top Q)^{-1} = \sigma^2 I_p$, so all components are uncorrelated and thus independent.

(c)

The variance of $\hat{\beta}$ is $\Sigma = \sigma^2 (X^\top X)^{-1} = \sigma^2 (R^\top R)^{-1}$, and so its correlation is

$$\mathrm{Diag}_i\{\Sigma_{ii}\}^{-1/2} \Sigma \, \mathrm{Diag}_i\{\Sigma_{ii}\}^{-1/2} = \mathrm{Diag}_i\{1/\|\mathbf{r}_i\|\} R^{-1} R^{-\top} \, \mathrm{Diag}_i\{1/\|\mathbf{r}_i\|\} = (R_\perp^{-\top})^\top R_\perp^{-\top},$$

where $\mathbf{r}_i$ is the i-th column of R and $R_\perp^{-\top}$ is the column normalization of $R^{-\top}$.

```
beta.hat <- backsolve(R, gamma.hat) # (i)
beta.cor <- crossprod(normalize.cols(inv.upper.tri(R, transpose = TRUE))) # (ii)
```

(d)

Due to high correlation in $\hat{\beta}$, common in polynomial fits based on monomial covariates, the standard error estimates are inflated and thus so are the p-values for testing each $\beta_j = 0$. The tests are more reliable when using γ since each component of $\hat{\gamma}$ is independent. Based on these tests, we can only reasonably reject up to a quadratic term.

```
data(cars)
y <- cars$dist; x <- cars$speed; d <- 3
# (i)
Q <- vandermonde.Q(x, d)
(gamma.hat <- drop(crossprod(Q, y)))
```

```
## [1] 303.91449 145.55226  22.99576  13.79688
```

```
(coef(lm(dist ~ Q - 1, data = cars))) # same estimates
```

```
##        Q1        Q2        Q3        Q4
## 303.91449 145.55226  22.99576  13.79688
```

```
# (ii)
R <- crossprod(Q, vandermonde(x, d))
(beta.hat <- backsolve(R, gamma.hat))
```

```
## [1] -19.50504910   6.80110597  -0.34965781   0.01025205
```

```
(coef(lm(dist ~ vandermonde(x, d) - 1, data = cars))) # same estimates
```

```
## vandermonde(x, d)1 vandermonde(x, d)2 vandermonde(x, d)3
##       -19.50504910         6.80110597        -0.34965781
## vandermonde(x, d)4
##         0.01025205
```

```
# (iii) (*)
summary(lm(dist ~ Q - 1, data = cars))
```

```
##
## Call:
## lm(formula = dist ~ Q - 1, data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.670  -9.601  -2.231   7.075  44.691
##
## Coefficients:
##    Estimate Std. Error t value Pr(>|t|)
## Q1    303.9       15.2  19.988  < 2e-16
## Q2    145.6       15.2   9.573  1.6e-12
## Q3     23.0       15.2   1.512    0.137
## Q4     13.8       15.2   0.907    0.369
##
## Residual standard error: 15.2 on 46 degrees of freedom
## Multiple R-squared:  0.9149, Adjusted R-squared:  0.9075
## F-statistic: 123.6 on 4 and 46 DF,  p-value: < 2.2e-16
```

```
summary(lm(dist ~ vandermonde(x, d) - 1, data = cars))
```

```
##
## Call:
## lm(formula = dist ~ vandermonde(x, d) - 1, data = cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.670  -9.601  -2.231   7.075  44.691
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## vandermonde(x, d)1 -19.50505   28.40530  -0.687    0.496
## vandermonde(x, d)2   6.80111    6.80113   1.000    0.323
## vandermonde(x, d)3  -0.34966    0.49988  -0.699    0.488
## vandermonde(x, d)4   0.01025    0.01130   0.907    0.369
##
## Residual standard error: 15.2 on 46 degrees of freedom
## Multiple R-squared:  0.9149, Adjusted R-squared:  0.9075
## F-statistic: 123.6 on 4 and 46 DF,  p-value: < 2.2e-16
```

(e) (∗)

Since R is upper triangular and $R^\top R = X^\top X$, the only missing condition is that the diagonal entries of R are positive. To guarantee that, we can define the vector $s_R = [\text{sgn}(R_{ii})]_i$, where $\text{sgn}(x) = I(x > 0) - I(x < 0)$, and build a new QR decomposition $QR = (Q\text{Diag}\{s_R\})(\text{Diag}\{s_R\}R) = \tilde{Q}\tilde{R}$. Note that since $\text{Diag}\{s_R\}$ is orthogonal, $\tilde{Q} = Q\text{Diag}\{s_R\}$ is still orthogonal, and $\tilde{R} = \text{Diag}\{s_R\}R$ is still upper triangular but now $\tilde{R}_{ii} = |R_{ii}| > 0$.

```
fix.chol.qr <- function (Q, R) {
  sr <- sign(diag(R))
  list(R = sr * R, # diag(R) = abs(diag(R))
       Q = sweep(Q, 2, sr, `*`)) #
}

# Check:
qx <- qr(vandermonde(x, d))
(R <- qr.R(qx)) # note that some diagonal entries are < 0
```

```
##           [,1]       [,2]       [,3]       [,4]
## [1,] -7.071068 -108.89444 -1870.7217 -34660.960
## [2,]  0.000000   37.01351  1117.9377  27771.535
## [3,]  0.000000    0.00000  -230.0513 -10089.202
## [4,]  0.000000    0.00000     0.0000  -1345.769
```

```
(fix.chol.qr(qr.Q(qx), R)$R) # now a Cholesky factor
```

```
##          [,1]      [,2]      [,3]      [,4]
## [1,] 7.071068 108.89444 1870.7217 34660.960
## [2,] 0.000000  37.01351 1117.9377 27771.535
## [3,] 0.000000   0.00000  230.0513 10089.202
## [4,] 0.000000   0.00000    0.0000  1345.769
```