Solution of assignment 10, ST2304

Problem 1

1.
Copying the code in Handout 2 gives us the two functions multinomprobs and lnL:

```
> multinomialprobs <- function(par) {
+ pA <- par[1]
+ pB <- par[2]
+ p0 <- 1-pA-pB
+ c(pA^2 + 2*pA*p0, pB^2 + 2*pB*p0, 2*pA*pB, p0^2)
+ }
> lnL <- function(par,x) {
+ n <- sum(x)
+ -dmultinom(x,prob=multinomialprobs(par),log=T)
+ }
```

Using this function to estimate the parameters of the blood type data gives:

```
> x <- c(44,27,4,88)
> fit <- optim(c(.25,.25),lnL,x=x)
> fit
$par
[1] 0.1604618 0.1003531

$value
[1] 6.917786

$counts
function gradient
      65       NA

$convergence
[1] 0

$message
NULL
```

The estimated allele frequencies are given under $par, meaning that pA and pB are estimated to ca. 0.16 and 0.10, respectively.
You can also see the estimated genotype frequencies through:

```
> Phat <- multinomialprobs(fit$par)
> Phat
[1] 0.26296987 0.15842973 0.03220566 0.54639474
```

Other information that may be of interest is the expected numbers and the fit of the model:

```
> n <- sum(x)
> n*Phat
[1] 42.864089 25.824047  5.249522 89.062342
```

```
> D <- sum((x-n*Phat)^2/(n*Phat))
> D
[1] 0.3937418
> pchisq(D,df=1,lower.tail=F)
[1] 0.5303391
```

This informs us that the model fits rather well with the data.

Ok, now to the standard errors of the estimates. Using the method in section 2.1 in Handout 5 we get:

```
> fit.hess <- optim(c(.25,.25),lnL,x=x, hessian=T)
> sqrt(diag(solve(fit.hess$hessian)))
[1] 0.02129385 0.01711418
```

Voila!

2.
Editing the commands to fit the probit link function yields the following two functions:

```
> multinomialprobs.H2 <- function(par) {
+ pa <- par[1]
+ pb <- par[2]
+ c((1-pa^2)*pb^2, (1-pb^2)*pa^2, (1-pa^2)*(1-pb^2), pa^2*pb^2)
+ }
> lnL.H2 <- function(par,x) {
+ n <- sum(x)
+ -dmultinom(x,prob=multinomialprobs.H2(par),log=T)
+ }
```

Note that this function will estimate pa and pb instead of pA and pB.
Using this function to evaluate the MLE of the gene frequencies gives us:

```
> x <- c(44,27,4,88)
> fit.H2 <- optim(c(.25,.25),lnL.H2,x=x, lower=c(.001,.001),upper=c(.999,.999))
Warning message:
In optim(c(0.25, 0.25), lnL.H2, x = x, lower = c(0.001, 0.001),  :
  bounds can only be used with method L-BFGS-B
> fit.H2
$par
[1] 0.8399537 0.8998954

$value
[1] 9.563715

$counts
function gradient
      24       24

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

The estimated allele frequencies of pa and pb are ca. 0.84 and 0.90, respectively. The warning is nothing to worry about, it just informs us that optim will use the "L-BFGS-B" optimisation method (instead of the default "Nelder-Mead" method).
Calculating the standard errors of the estimates result in:

```
> fit.hess.H2 <- optim(c(.75,.75),lnL.H2,x=x, hessian=T)
> sqrt(diag(solve(fit.hess.H2$hessian)))
[1] 0.02125107 0.01707864
```

3.
Expected number of observations can be estimated through:

```
> Phat.H2 <- multinomialprobs.H2(fit.H2$par)
> Phat.H2
[1] 0.23847153 0.13418201 0.05600618 0.57134027
> n <- sum(x)
> n*Phat.H2
[1] 38.870860 21.871667  9.129008 93.128465
```

4.
The χ^2 statistic is calculated via:

```
> D.H2 <- sum((x-n*Phat.H2)^2/(n*Phat.H2))
> D.H2
[1] 5.043348
> pchisq(D.H2,df=1, lower.tail=F)
[1] 0.02472067
```

This is <0.025 which makes us reject H_0. The model can be rejected.

5.
Goodness-of-fit can sometimes be estimated also for more parameters, although there might appear "ridges" in the likelihood surface, which results in the absence of a maxima (or an infinite number of maxima).

Problem 2
Just to avoid spill-over gunk from problem 1

```
rm(list=ls(all=TRUE))
```

Now import the data:

```
> ovul <- read.table("http://www.math.ntnu.no/~jarlet/statmod/ovul2.dat")
> names(ovul)
[1] "time" "x"    "n"
> attach(ovul)
```

1.
First we create a new function, using the probability function from the probit link function, compare with the example in section 3 of Handout 5.

```
> lnLP <- function(par,x,n,time) {
+ beta0 <- par[1]
+ beta1 <- par[2]
+ p <- pnorm(beta0+beta1*time)
+ -sum(dbinom(x,size=n,prob=p,log=T))
+ }
```

The model parameters can now be optimized, using relatively reasonable initial values

```
> fitP <- optim(c(-10,0),lnLP,x=x,n=n,time=time)
> fitP
$par
[1] -18.0510392   0.0651633

$value
[1] 92.6427

$counts
function gradient
     101       NA

$convergence
[1] 0

$message
NULL
```

The values for β_0 and β_1 are -18.05 and 0.06516, which is almost exactly the same as we saw in the glm model in exercise 7.

2.
The second part includes the additional parameter q, which will cap the function at a lower level than 1, implying a model where not all females ovulate during the rut.

```
> lnLP.q <- function(par,x,n,time) {
+ beta0 <- par[1]
+ beta1 <- par[2]
+ q <- par[3]
+ p <- q*pnorm(beta0+beta1*time)
+ -sum(dbinom(x,size=n,prob=p,log=T))
+ }
```

Fitting the parameters gives us

```
> fitP.q <- optim(c(-18,0.06,1),lnLP.q,x=x,n=n,time=time)
Warning messages:
1: In dbinom(x, size, prob, log) : NaNs produced
2: In dbinom(x, size, prob, log) : NaNs produced
3: In dbinom(x, size, prob, log) : NaNs produced
4: In dbinom(x, size, prob, log) : NaNs produced
5: In dbinom(x, size, prob, log) : NaNs produced
6: In dbinom(x, size, prob, log) : NaNs produced
7: In dbinom(x, size, prob, log) : NaNs produced
8: In dbinom(x, size, prob, log) : NaNs produced
9: In dbinom(x, size, prob, log) : NaNs produced
10: In dbinom(x, size, prob, log) : NaNs produced
```

Note that it may be required to use reasonable start values, otherwise the optimization may fail. You will also get a warning due to optim trying to evaluate outside accepted values (q>1 or q<0). This is however just a warning, not an error, and no downstream errors are caused by this. The function will use use a non-default method to handle this.

```
> fitP.q
$par
[1] -44.6367971   0.1626905   0.8482704

$value
[1] 68.208

$counts
function gradient
     244       NA

$convergence
[1] 0

$message
NULL
```

The values for β_0 and β_1 are -44.64 and 0.1627, q is estimated as 0.8482.

3.
In order to find the standard errors we use the same method as in Problem 1

```
> fitP.q.hess <- optim(c(-18,0.06,1),lnLP.q,x=x,n=n,time=time, hessian=TRUE)
Warning messages:
1: In dbinom(x, size, prob, log) : NaNs produced
2: In dbinom(x, size, prob, log) : NaNs produced
3: In dbinom(x, size, prob, log) : NaNs produced
4: In dbinom(x, size, prob, log) : NaNs produced
5: In dbinom(x, size, prob, log) : NaNs produced
6: In dbinom(x, size, prob, log) : NaNs produced
7: In dbinom(x, size, prob, log) : NaNs produced
8: In dbinom(x, size, prob, log) : NaNs produced
9: In dbinom(x, size, prob, log) : NaNs produced
10: In dbinom(x, size, prob, log) : NaNs produced
> sqrt(diag(solve(fitP.q.hess$hessian)))
[1] 3.31537822 0.01216053 0.02221172
```

The standard errors for β_0, β_1 and q are 3.315, 0.01216 and 0.02221, respectively.

4.
Let us plot the data and our two models, plotting the data is rather straightforward.

```
> plot(time,ovul$x/ovul$n,cex=sqrt(ovul$n)*0.8, xlab="Time of year", ylab="x/n")
```

The curves for the models may look weird if we have residual values for x or n from Problem 1.

```
> curve(pnorm(fitP$par[1]+fitP$par[2]*x),col="orange",add=T)
> curve(fitP.q$par[3]*pnorm((fitP.q$par[1]+fitP.q$par[2]*x)),col="brown",add=T,lty="dott
```

5.
The models are nested, if the additional parameter q is set to 1, the models would be fitted to identical values of the parameters β_0 and β_1.

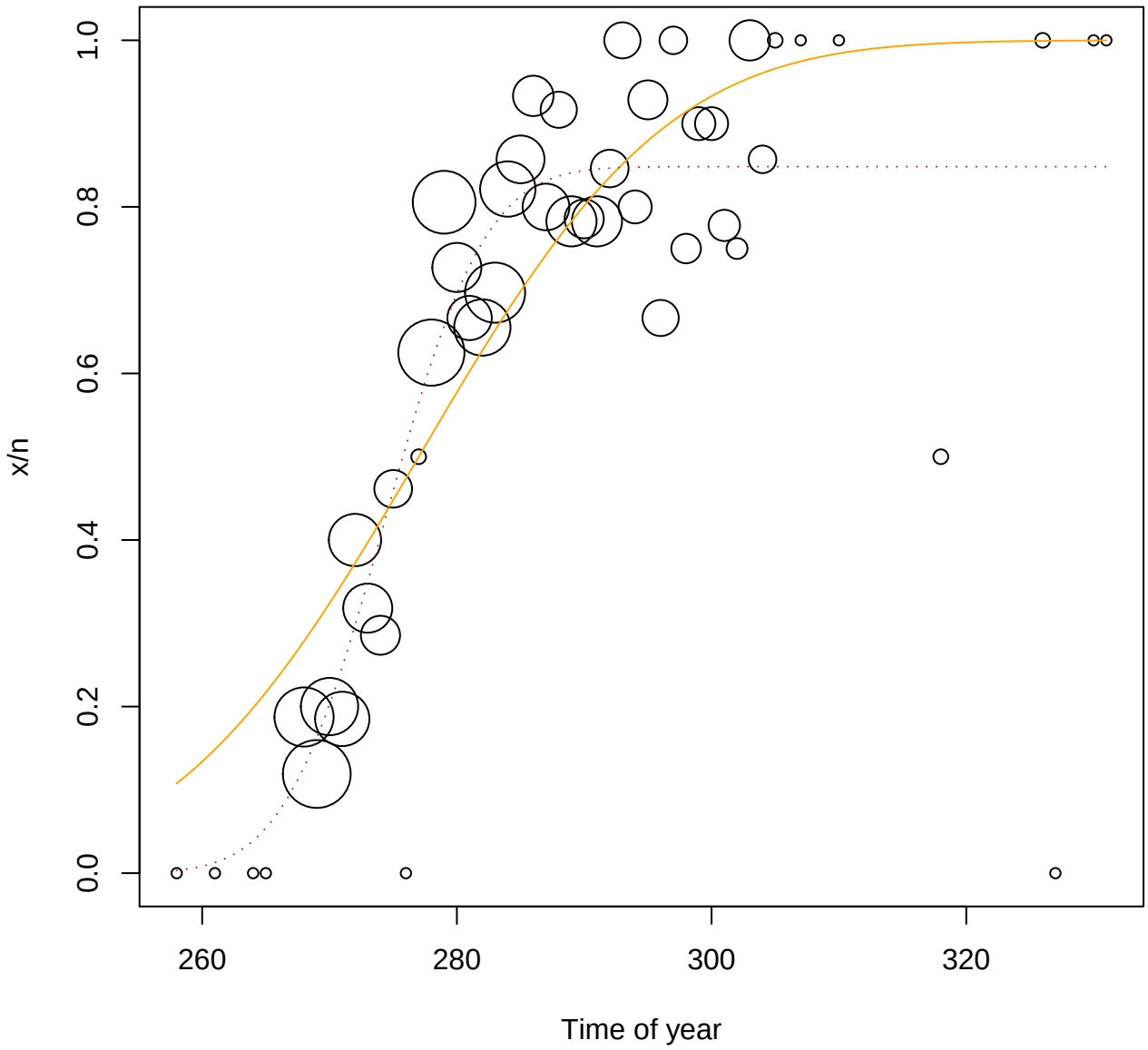

Figure 1: Proportion of x/n individuals having ovulated at different days against number of days since January 1. The probit function is solid orange and the probit function with the q parameter is dotted brown.