Homework 6

MacKenzie Ullman

4/9/2021

1. Load data and place into an unmarkedFramePCount object

```
setwd("F:/Quant Eco")
getwd()
```

```
## [1] "F:/Quant Eco"
```

```
HW6 <- read.csv(file ='count.csv')
head(HW6)
```

```
##   j1 j2 j3
## 1  0  0  0
## 2  1  1  1
## 3  8 13 12
## 4  2  1  1
## 5  3  8  3
## 6  0  0  0
```

```
summary(HW6)
```

```
##        j1              j2              j3       
##  Min.   : 0.00   Min.   : 0.00   Min.   : 0.00  
##  1st Qu.: 1.00   1st Qu.: 1.00   1st Qu.: 1.00  
##  Median : 2.00   Median : 2.00   Median : 2.00  
##  Mean   : 3.21   Mean   : 3.24   Mean   : 3.43  
##  3rd Qu.: 4.00   3rd Qu.: 4.00   3rd Qu.: 5.00  
##  Max.   :24.00   Max.   :25.00   Max.   :17.00  
```

```
library(unmarked)
```

```
## Warning: package 'unmarked' was built under R version 3.6.3
```

```
## Loading required package: lattice
```

```
count_mat <- as.matrix(HW6)
nmix_data <- unmarkedFramePCount(y = count_mat)
```

2. Fit an N-mixture model that assumes conditional detection probability is a function of the detection covariate provided, and expected abundance is an additive function of variables x1 and x2.

```
#detection covariates
p_covs <- read.csv('obs_covs.csv')
head(p_covs)
```

```
##          j1           j2         j3
## 1 0.9401255 -0.555760967 -0.3582067
## 2 2.4448899 -0.457783896  0.4253590
## 3 1.1633403  0.006345006 -0.2650346
```

```
## 4 0.7138189  0.857225770  1.5519564
## 5 2.0457416  0.077946090  1.9626749
## 6 0.7596043 -0.356677808 -0.1295560
```

```r
#Placing detection covariates in an unmarkedFramePCount object
det_covs <- list(
replicate = data.frame(p_covs[, c('j1', 'j2', 'j3')])
)

#Placing the list of detection covariates in to the unmarkedFramePCount object
nmix_data <- unmarkedFramePCount(y = as.matrix(count_mat), obsCovs = det_covs)

fit <- pcount(formula = ~ replicate ~ 1, data = nmix_data, K = 100)
```

```r
#site level covariates
sitecovs <- read.csv('site_covs.csv')
head(sitecovs)
```

```
##            x1 x2
## 1 -1.06733947  b
## 2 -0.98588873  a
## 3 -0.09409764  d
## 4  1.32241491  a
## 5  0.45689994  d
## 6 -0.89026419  b
```

```r
nmix_data <- unmarkedFramePCount(y = as.matrix(count_mat), siteCovs = sitecovs, obsCovs = det_covs)

fit <- pcount(~ replicate ~ x1 + x2,data = nmix_data, K = 100)
summary(fit)
```

```
##
## Call:
## pcount(formula = ~replicate ~ x1 + x2, data = nmix_data, K = 100)
##
## Abundance (log-scale):
##             Estimate     SE     z  P(>|z|)
## (Intercept)    0.915 0.1106  8.27 1.30e-16
## x1             0.370 0.0401  9.21 3.14e-20
## x2b           -0.161 0.1382 -1.16 2.45e-01
## x2c           -0.189 0.1522 -1.24 2.14e-01
## x2d            1.335 0.1195 11.17 5.63e-29
##
## Detection (logit-scale):
##             Estimate     SE     z  P(>|z|)
## (Intercept)    1.259 0.0925  13.6 4.01e-42
## replicate     -0.841 0.0639 -13.2 1.57e-39
##
## AIC: 1736.028
## Number of sites: 200
## optim convergence code: 0
## optim iterations: 46
## Bootstrap iterations: 0
```

3. Interpret the effect of x1 on the expected count at each site. Verity your interpretation in R. The expected count per each site increases by 0.3696188 when x1 increases by 1 unit.

```r
#verify
beta <- coef(fit)
beta
```

```
##     lam(Int)      lam(x1)     lam(x2b)     lam(x2c)     lam(x2d)       p(Int)
##    0.9151447    0.3696188   -0.1606306   -0.1891303    1.3351914    1.2585992
## p(replicate)
##   -0.8410807
```

```r
a <- beta[2]*1
a
```

```
##   lam(x1)
## 0.3696188
```

```r
b <- beta[2]*2
b
```

```
##   lam(x1)
## 0.7392376
```

```r
c <- b-a
c
```

```
##   lam(x1)
## 0.3696188
```

4. Predict and plot the effect of the supplied detection covariate. Do this over the range of this covariate.

```r
new <- data.frame(replicate = seq(from = min(det_covs$replicate), to = max(det_covs$replicate), length.o


prd <- predict(object = fit, newdata = new, type = 'det')

prd
```

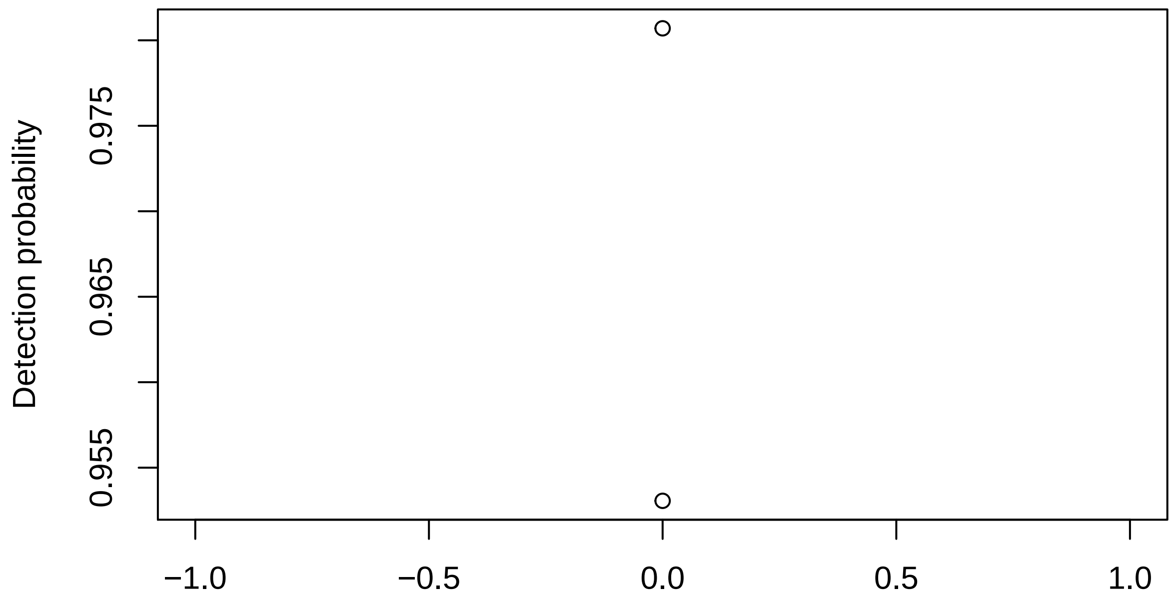
```
##     Predicted          SE     lower     upper
## 1   0.9698079 0.006852513 0.9530609 0.9807009
## 2   0.9682779 0.007075790 0.9510610 0.9795680
## 3   0.9666731 0.007303550 0.9489797 0.9783704
## 4   0.9649900 0.007535662 0.9468141 0.9771046
## 5   0.9632251 0.007771974 0.9445612 0.9757669
## 6   0.9613748 0.008012310 0.9422177 0.9743534
## 7   0.9594353 0.008256468 0.9397807 0.9728603
## 8   0.9574028 0.008504222 0.9372466 0.9712832
## 9   0.9552732 0.008755319 0.9346122 0.9696180
## 10  0.9530423 0.009009476 0.9318739 0.9678600
## 11  0.9507059 0.009266382 0.9290284 0.9660045
## 12  0.9482595 0.009525696 0.9260719 0.9640466
## 13  0.9456987 0.009787044 0.9230009 0.9619813
## 14  0.9430188 0.010050021 0.9198115 0.9598031
## 15  0.9402149 0.010314193 0.9165001 0.9575068
## 16  0.9372823 0.010579089 0.9130626 0.9550866
## 17  0.9342159 0.010844209 0.9094952 0.9525366
## 18  0.9310105 0.011109020 0.9057939 0.9498510
## 19  0.9276611 0.011372959 0.9019547 0.9470235
## 20  0.9241624 0.011635433 0.8979735 0.9440478
## 21  0.9205089 0.011895821 0.8938462 0.9409174
```

```
## 22  0.9166952 0.012153478 0.8895686 0.9376257
## 23  0.9127160 0.012407735 0.8851365 0.9341660
## 24  0.9085656 0.012657908 0.8805457 0.9305315
## 25  0.9042386 0.012903296 0.8757921 0.9267154
## 26  0.8997295 0.013143191 0.8708713 0.9227107
## 27  0.8950326 0.013376884 0.8657791 0.9185104
## 28  0.8901426 0.013603670 0.8605112 0.9141077
## 29  0.8850540 0.013822859 0.8550635 0.9094957
## 30  0.8797616 0.014033785 0.8494316 0.9046677
## 31  0.8742601 0.014235818 0.8436114 0.8996171
## 32  0.8685446 0.014428373 0.8375987 0.8943375
## 33  0.8626100 0.014610928 0.8313894 0.8888228
## 34  0.8564518 0.014783035 0.8249792 0.8830671
## 35  0.8500656 0.014944339 0.8183643 0.8770651
## 36  0.8434471 0.015094592 0.8115404 0.8708119
## 37  0.8365927 0.015233675 0.8045037 0.8643030
## 38  0.8294989 0.015361615 0.7972503 0.8575347
## 39  0.8221625 0.015478606 0.7897762 0.8505039
## 40  0.8145811 0.015585027 0.7820778 0.8432084
## 41  0.8067524 0.015681463 0.7741513 0.8356466
## 42  0.7986748 0.015768722 0.7659930 0.8278182
## 43  0.7903474 0.015847852 0.7575996 0.8197237
## 44  0.7817696 0.015920156 0.7489676 0.8113649
## 45  0.7729417 0.015987197 0.7400939 0.8027446
## 46  0.7638646 0.016050804 0.7309753 0.7938668
## 47  0.7545398 0.016113068 0.7216092 0.7847370
## 48  0.7449697 0.016176327 0.7119931 0.7753618
## 49  0.7351575 0.016243147 0.7021249 0.7657491
## 50  0.7251071 0.016316280 0.6920032 0.7559080
## 51  0.7148232 0.016398621 0.6816268 0.7458488
## 52  0.7043114 0.016493146 0.6709958 0.7355829
## 53  0.6935783 0.016602837 0.6601108 0.7251227
## 54  0.6826312 0.016730598 0.6489737 0.7144814
## 55  0.6714782 0.016879164 0.6375876 0.7036725
## 56  0.6601284 0.017051003 0.6259572 0.6927104
## 57  0.6485917 0.017248225 0.6140886 0.6816094
## 58  0.6368788 0.017472490 0.6019899 0.6703840
## 59  0.6250012 0.017724941 0.5896711 0.6590483
## 60  0.6129713 0.018006141 0.5771438 0.6476165
## 61  0.6008018 0.018316041 0.5644221 0.6361021
## 62  0.5885067 0.018653966 0.5515216 0.6245182
## 63  0.5761000 0.019018627 0.5384600 0.6128772
## 64  0.5635966 0.019408154 0.5252566 0.6011912
## 65  0.5510119 0.019820144 0.5119323 0.5894716
## 66  0.5383615 0.020251728 0.4985093 0.5777293
## 67  0.5256615 0.020699645 0.4850108 0.5659748
## 68  0.5129282 0.021160318 0.4714609 0.5542183
## 69  0.5001781 0.021629942 0.4578842 0.5424694
## 70  0.4874278 0.022104559 0.4443057 0.5307379
## 71  0.4746938 0.022580134 0.4307502 0.5190329
## 72  0.4619927 0.023052628 0.4172427 0.5073636
## 73  0.4493407 0.023518058 0.4038075 0.4957390
## 74  0.4367539 0.023972554 0.3904683 0.4841679
## 75  0.4242480 0.024412411 0.3772482 0.4726591
```

```
## 76  0.4118384 0.024834124 0.3641692 0.4612212
## 77  0.3995399 0.025234429 0.3512523 0.4498626
## 78  0.3873668 0.025610326 0.3385173 0.4385917
## 79  0.3753328 0.025959101 0.3259825 0.4274165
## 80  0.3634508 0.026278344 0.3136650 0.4163450
## 81  0.3517331 0.026565958 0.3015806 0.4053850
## 82  0.3401913 0.026820162 0.2897432 0.3945439
## 83  0.3288362 0.027039495 0.2781656 0.3838289
## 84  0.3176776 0.027222811 0.2668589 0.3732469
## 85  0.3067246 0.027369275 0.2558327 0.3628045
## 86  0.2959854 0.027478348 0.2450952 0.3525080
## 87  0.2854674 0.027549779 0.2346530 0.3423633
## 88  0.2751770 0.027583590 0.2245115 0.3323757
## 89  0.2651200 0.027580055 0.2146747 0.3225506
## 90  0.2553011 0.027539685 0.2051451 0.3128925
## 91  0.2457242 0.027463210 0.1959243 0.3034058
## 92  0.2363925 0.027351553 0.1870126 0.2940944
## 93  0.2273084 0.027205817 0.1784092 0.2849616
## 94  0.2184736 0.027027255 0.1701124 0.2760107
## 95  0.2098888 0.026817258 0.1621195 0.2672441
## 96  0.2015544 0.026577327 0.1544271 0.2586642
## 97  0.1934699 0.026309060 0.1470310 0.2502726
## 98  0.1856342 0.026014125 0.1399262 0.2420708
## 99  0.1780459 0.025694249 0.1331074 0.2340598
## 100 0.1707027 0.025351199 0.1265685 0.2262402
```

```r
plot(x = c(0,0), y = prd[1, c('lower', 'upper')],

ylab = 'Detection probability', xlab = '',)
```



5. Use contrasts to compare expected abundance between all pairwise levels of variable x2. Obtain p-values associated with each contrast and tell me whether you reject or fail to reject each null hypothesis tested.

```
x <- matrix(
c(0, 0, 1, -1, 0,
0, 0, 1, 0, -1,
0, 0, 0, 1, -1),
nrow = 3, byrow = T
)
x
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    0    1   -1    0
## [2,]    0    0    1    0   -1
## [3,]    0    0    0    1   -1
```

```
lin_com <- linearComb(obj = fit, coefficients = x, type = 'state')

lin_com
```

```
## Linear combination(s) of Abundance estimate(s)
##
##   Estimate     SE (Intercept) x1 x2b x2c x2d
## 1   0.0285 0.1332           0  0   1  -1   0
## 2  -1.4958 0.0935           0  0   1   0  -1
## 3  -1.5243 0.1140           0  0   0   1  -1
```

```
w <- coef(lin_com) / SE(lin_com)
w
```

```
## [1]   0.2140043 -15.9929898 -13.3713082
```

```r
#Calculating p-values
2 * pnorm(-1 * abs(w))
```

```
## [1] 8.305437e-01 1.430000e-57 8.896231e-41
```

We reject all the null hypotheses. There is a difference between b and c, a difference between b and d, and a difference between c and d, in terms of abundance probability