Week 5, Lecture 09
Advanced statistical methods, part I: Ecological analyses, ordinal data, and dimensionality reduction

Richard E.W. Berl

Spring 2019

Ecological analyses

Download the data from the annual Audubon Christmas Bird Count here: http://netapp.audubon.org/CBCObservation/Historical/ResultsByCount.aspx

Let's get all of the data available for Fort Collins: For "Start Year," select Count 1 in 1900; leave "End Year" at 2017; select "United States" and "Colorado" and flip through the pages until you find "Fort Collins" (was at the bottom of page 2 for me). Click the bubble, select CSV and Export.

Place the data in your `/data` folder.

If you open up the CSV and take a look, you'll see that the data are an absolute mess. Multiple tables are all provided one after another in the same spreadsheet. To get the data that we want, we need to skip some rows, read some rows, and skip some more rows. To make this easier, I opted to use `read_csv()` from the `readr` package, rather than the base `read.csv()` function, because it tries to figure out what the columns should be for messy data.

```
library(readr)
```

```
fcbird = as.data.frame(read_csv("./data/HistoricalResultsByCount [COFC-1901-2018].csv",
                    skip=208, n_max=18031))
```

```
## Parsed with column specification:
## cols(
##   COM_NAME = col_character(),
##   CountYear = col_character(),
##   how_manyCW = col_character(),
##   NumberByPartyHours = col_double(),
##   Flags = col_character()
## )
```

```
## Warning: 2 parsing failures.
## row # A tibble: 2 x 5 col     row col          expected        actual   file
```

```
head(fcbird)
```

```
##                                           COM_NAME
## 1 Greater White-fronted Goose\r\n[Anser albifrons]
## 2 Greater White-fronted Goose\r\n[Anser albifrons]
## 3 Greater White-fronted Goose\r\n[Anser albifrons]
## 4 Greater White-fronted Goose\r\n[Anser albifrons]
## 5 Greater White-fronted Goose\r\n[Anser albifrons]
## 6 Greater White-fronted Goose\r\n[Anser albifrons]
```

```
##                                                                                               CountYea
## 1       1926 [27]\r\nCount Date: 12/25/1926\r\n# Participants: \r\n# Species Reported: \r\nTotal Hrs
## 2       1927 [28]\r\nCount Date: 12/23/1927\r\n# Participants: \r\n# Species Reported: \r\nTotal Hrs
## 3  1947 [48]\r\nCount Date: 12/27/1947\r\n# Participants: \r\n# Species Reported: \r\nTotal Hrs.: 8.5
## 4 1948 [49]\r\nCount Date: 12/30/1948\r\n# Participants: \r\n# Species Reported: \r\nTotal Hrs.: 28.0
## 5 1949 [50]\r\nCount Date: 12/29/1949\r\n# Participants: \r\n# Species Reported: \r\nTotal Hrs.: 25.0
## 6 1950 [51]\r\nCount Date: 12/29/1950\r\n# Participants: \r\n# Species Reported: \r\nTotal Hrs.: 18.0
##   how_manyCW NumberByPartyHours Flags
## 1       <NA>                 NA  <NA>
## 2       <NA>                 NA  <NA>
## 3       <NA>                 NA  <NA>
## 4       <NA>                 NA  <NA>
## 5       <NA>                 NA  <NA>
## 6       <NA>                 NA  <NA>
```

```
tail(fcbird)
```

```
##                                     COM_NAME
## 18026 House Sparrow\r\n[Passer domesticus]
## 18027 House Sparrow\r\n[Passer domesticus]
## 18028 House Sparrow\r\n[Passer domesticus]
## 18029 House Sparrow\r\n[Passer domesticus]
## 18030 House Sparrow\r\n[Passer domesticus]
## 18031 House Sparrow\r\n[Passer domesticus]
##
## 18026  2012 [113]\r\nCount Date: 12/15/2012\r\n# Participants: 71\r\n# Species Reported: 94\r\nTotal
## 18027  2013 [114]\r\nCount Date: 12/14/2013\r\n# Participants: 71\r\n# Species Reported: 84\r\nTotal
## 18028  2014 [115]\r\nCount Date: 12/20/2014\r\n# Participants: 75\r\n# Species Reported: 95\r\nTotal
## 18029 2015 [116]\r\nCount Date: 12/19/2015\r\n# Participants: 77\r\n# Species Reported: 100\r\nTotal
## 18030  2016 [117]\r\nCount Date: 12/17/2016\r\n# Participants: 82\r\n# Species Reported: 88\r\nTotal
## 18031 2017 [118]\r\nCount Date: 12/16/2017\r\n# Participants: 90\r\n# Species Reported: 100\r\nTotal
##       how_manyCW NumberByPartyHours Flags
## 18026       2462            18.2370   HC,
## 18027       1694            11.4537  <NA>
## 18028       1409             9.6972  <NA>
## 18029       1443             9.5880  <NA>
## 18030        760             5.7445  <NA>
## 18031       1022             7.1469  <NA>
```

Let's clean up our variables a bit.

```
library(stringr)

fcbird$SPEC_NAME = str_split_fixed(fcbird$COM_NAME, "\\r\\n", 2)[,2]
fcbird$SPEC_NAME = gsub("\\[|\\]", "", fcbird$SPEC_NAME)
fcbird$COM_NAME = str_split_fixed(fcbird$COM_NAME, "\r\n", 2)[,1]
fcbird$CountYear = as.integer(substr(fcbird$CountYear, 1, 4))

fcbird = fcbird[,c("COM_NAME","SPEC_NAME","CountYear","how_manyCW")]

head(fcbird)
```

```
##                      COM_NAME       SPEC_NAME CountYear how_manyCW
## 1 Greater White-fronted Goose Anser albifrons      1926       <NA>
## 2 Greater White-fronted Goose Anser albifrons      1927       <NA>
## 3 Greater White-fronted Goose Anser albifrons      1947       <NA>
## 4 Greater White-fronted Goose Anser albifrons      1948       <NA>
```

```
## 5 Greater White-fronted Goose Anser albifrons      1949       <NA>
## 6 Greater White-fronted Goose Anser albifrons      1950       <NA>
```

```r
tail(fcbird)
```

```
##             COM_NAME         SPEC_NAME CountYear how_manyCW
## 18026 House Sparrow Passer domesticus      2012       2462
## 18027 House Sparrow Passer domesticus      2013       1694
## 18028 House Sparrow Passer domesticus      2014       1409
## 18029 House Sparrow Passer domesticus      2015       1443
## 18030 House Sparrow Passer domesticus      2016        760
## 18031 House Sparrow Passer domesticus      2017       1022
```

Now, for the analyses we'll be doing (with the `vegan` package), we need our species as columns and our years (typically different sampling "sites") as rows. So we need to *spread* the rows of our `COM_NAME` variable across columns, using `how_manyCW` as its values.

If we refer back to the Data Wrangling Cheat Sheet, we see that we need to use the `spread()` function from `tidyr`.

```r
library(tidyr)
```

```r
fcbirdW = spread(fcbird[,-2], "COM_NAME", "how_manyCW")
```

```r
fcbirdW[1:5,1:10]
```

```
##   CountYear Accipiter sp. African Collared-Dove
## 1      1926          <NA>                  <NA>
## 2      1927          <NA>                  <NA>
## 3      1947          <NA>                  <NA>
## 4      1948          <NA>                  <NA>
## 5      1949          <NA>                  <NA>
##   American Black Duck x Mallard (hybrid) American Coot American Crow
## 1                                   <NA>          <NA>          <NA>
## 2                                   <NA>          <NA>          <NA>
## 3                                   <NA>          <NA>             9
## 4                                   <NA>          <NA>             4
## 5                                   <NA>          <NA>           192
##   American Dipper American Goldfinch American Kestrel American Pipit
## 1            <NA>               <NA>             <NA>           <NA>
## 2            <NA>               <NA>                1           <NA>
## 3            <NA>               <NA>             <NA>           <NA>
## 4              10               <NA>               cw           <NA>
## 5               4               <NA>                1           <NA>
```

Looks good. However, we can see there are a lot of missing values, and `vegan` can't deal with any missing values.

`complete.cases()` would remove every row, so we want to find a way to include as much of our data as we can while eliminating missing values. This is an optimization problem that R doesn't have a base function for, so I Googled it and came to this thread on StackOverflow.

From one of the responses, I copied the code below to find the "best" subset of the data:

```r
l1 = combn(2:length(fcbirdW[,-1]), 2, function(x) fcbirdW[,-1][x[1]:x[2]], simplify = FALSE)
# If you also need "combinations" of only single columns, then uncomment the next line
# l1 = c(d[-1], l1)
l2 = sapply(l1, function(x) sum(complete.cases(x)))

score = sapply(1:length(l1), function(i) NCOL(l1[[i]]) * l2[i])
```

```r
best_score = which.max(score)
best = l1[[best_score]]
```

Source: dww on StackOverflow, 12/4/18

And then I want to take the complete cases of those variables, so that we do have complete data with no missing values.

```r
rownames(best) = fcbirdW$CountYear
best = best[complete.cases(best),]
# best = apply(best, as.numeric)
best = data.frame(lapply(best, function(x) as.numeric(as.character(x))),
                  check.names=F, row.names=rownames(best))
```

```r
head(best)
```

```
##      American Crow American Dipper American Goldfinch American Kestrel
## 1952           353              12                 16                2
## 1956             5               2                 36                2
## 1957             3               3                  7                3
## 1958           168               8                  3                7
## 1960             2               5                  3                6
## 1962           590              20                  6                2
```

```r
str(best)
```

```
## 'data.frame':    60 obs. of  4 variables:
##  $ American Crow     : num  353 5 3 168 2 590 130 13 100 390 ...
##  $ American Dipper   : num  12 2 3 8 5 20 15 10 2 5 ...
##  $ American Goldfinch: num  16 36 7 3 3 6 1 3 6 32 ...
##  $ American Kestrel  : num  2 2 3 7 6 2 5 4 2 12 ...
```

Great. Now we can load up vegan.

```r
install.packages("vegan")
```

```r
library(vegan)
```

```
## Loading required package: permute
```

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

```
## This is vegan 2.5-4
```

Diversity

```r
?diversity
```

```r
diversity(best, index="shannon")
```

```
##      1952      1956      1957      1958      1960      1962      1963
## 0.3437796 0.6994078 1.3032836 0.4172591 1.3050964 0.2188448 0.5043830
##      1964      1965      1966      1967      1968      1969      1970
## 1.2274905 0.3910243 0.4453950 0.3129922 0.3027719 0.2456579 0.3681573
##      1971      1972      1973      1974      1975      1976      1977
## 0.2114731 0.5172964 1.0273063 0.3687373 0.7055312 0.3574685 0.5070725
##      1979      1980      1981      1982      1983      1984      1985
## 0.5238491 0.6494850 0.8190258 1.1390141 0.9851331 1.1136518 1.1115508
##      1986      1987      1988      1989      1990      1991      1992
## 1.0876915 1.0321125 1.0849486 0.6690217 0.9653766 1.1289526 1.2249457
```

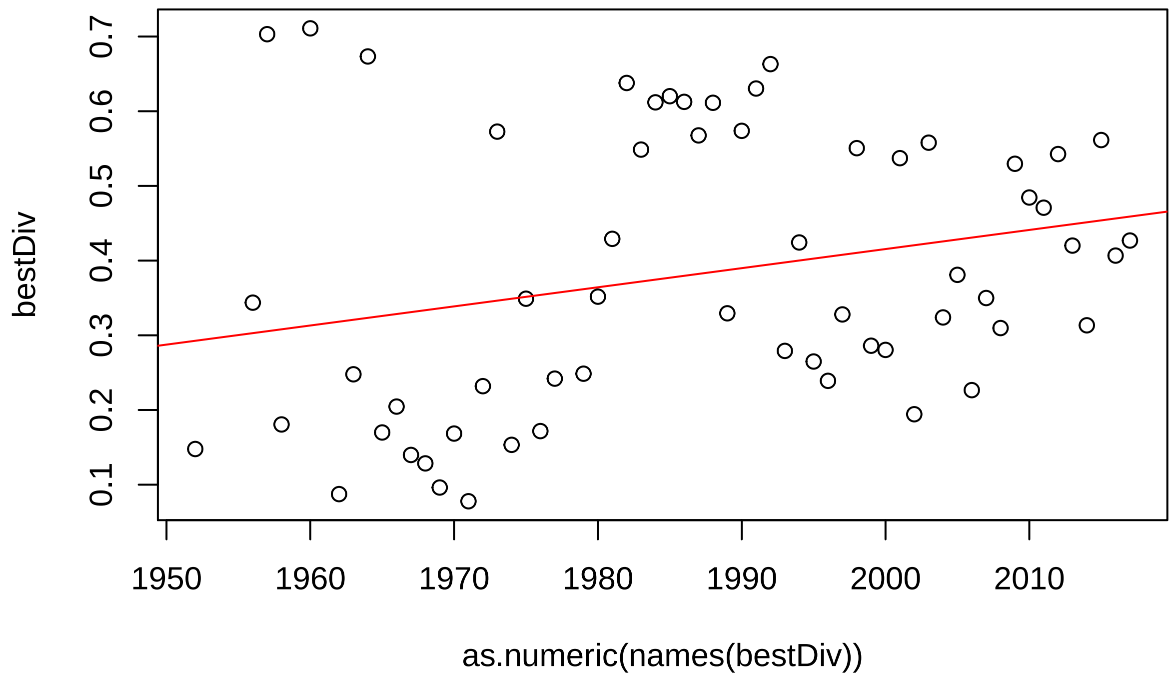
```
##      1993      1994      1995      1996      1997      1998      1999 
## 0.5457064 0.7918858 0.5299561 0.5084034 0.6102280 0.9501821 0.5673978 
##      2000      2001      2002      2003      2004      2005      2006 
## 0.5532326 0.8874201 0.4275090 0.9922356 0.6638118 0.7163728 0.4833601 
##      2007      2008      2009      2010      2011      2012      2013 
## 0.6654255 0.6276081 0.9762043 0.8541593 0.8501107 0.9471467 0.7681333 
##      2014      2015      2016      2017 
## 0.6260655 0.9791286 0.7936965 0.7690246
```

```
diversity(best, index="simpson")
```

```
##       1952       1956       1957       1958       1960       1962 
## 0.14776841 0.34370370 0.70312500 0.18065672 0.71093750 0.08741006 
##       1963       1964       1965       1966       1967       1968 
## 0.24779615 0.67333333 0.16991736 0.20458590 0.13981213 0.12852485 
##       1969       1970       1971       1972       1973       1974 
## 0.09622533 0.16844073 0.07785600 0.23192323 0.57269965 0.15336187 
##       1975       1976       1977       1979       1980       1981 
## 0.34899996 0.17176848 0.24199691 0.24858277 0.35173546 0.42913703 
##       1982       1983       1984       1985       1986       1987 
## 0.63781217 0.54863182 0.61186583 0.62013317 0.61254071 0.56760808 
##       1988       1989       1990       1991       1992       1993 
## 0.61126005 0.32947021 0.57373279 0.63037522 0.66306406 0.27912875 
##       1994       1995       1996       1997       1998       1999 
## 0.42428440 0.26492143 0.23901937 0.32795545 0.55056497 0.28605894 
##       2000       2001       2002       2003       2004       2005 
## 0.28045643 0.53715014 0.19434426 0.55787305 0.32392225 0.38094189 
##       2006       2007       2008       2009       2010       2011 
## 0.22659745 0.34990480 0.30970734 0.52964575 0.48446848 0.47083788 
##       2012       2013       2014       2015       2016       2017 
## 0.54263525 0.42010744 0.31339904 0.56141183 0.40671627 0.42687500
```

```
bestDiv = diversity(best, index="simpson")
```

```
plot(as.numeric(names(bestDiv)), bestDiv)
abline(lm(bestDiv ~ as.numeric(names(bestDiv))), col="red")
```




```r
cor.test(as.numeric(names(bestDiv)), bestDiv)
```

```
##
##  Pearson's product-moment correlation
##
## data:  as.numeric(names(bestDiv)) and bestDiv
## t = 2.01, df = 58, p-value = 0.04909
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.001355045 0.478133794
## sample estimates:
##       cor
## 0.2551919
```

Evenness

```r
diversity(best, index="shannon") / log(specnumber(best))
```

```
##      1952      1956      1957      1958      1960      1962      1963
## 0.2479846 0.5045161 0.9401204 0.3009888 0.9414280 0.1578631 0.3638354
##      1964      1965      1966      1967      1968      1969      1970
## 0.8854472 0.2820644 0.3212846 0.2257761 0.2184037 0.1772047 0.2655694
##      1971      1972      1973      1974      1975      1976      1977
## 0.1525456 0.3731505 0.7410449 0.2659878 0.5089332 0.2578590 0.3657755
##      1979      1980      1981      1982      1983      1984      1985
## 0.3778773 0.4685044 0.5908022 0.8216250 0.7106233 0.8033300 0.8018144
```

```
##      1986      1987      1988      1989      1990      1991      1992
## 0.7846036 0.7445118 0.7826250 0.4825972 0.6963720 0.8143671 0.8836115
##      1993      1994      1995      1996      1997      1998      1999
## 0.3936440 0.5712249 0.3822826 0.3667355 0.4401865 0.6854115 0.4092910
##      2000      2001      2002      2003      2004      2005      2006
## 0.3990730 0.6401383 0.3083826 0.7157467 0.4788390 0.5167537 0.3486706
##      2007      2008      2009      2010      2011      2012      2013
## 0.4800030 0.4527236 0.7041825 0.6161457 0.6132252 0.6832219 0.5540911
##      2014      2015      2016      2017
## 0.4516108 0.7062920 0.5725310 0.5547340
```

Richness

```
?rarefy
```

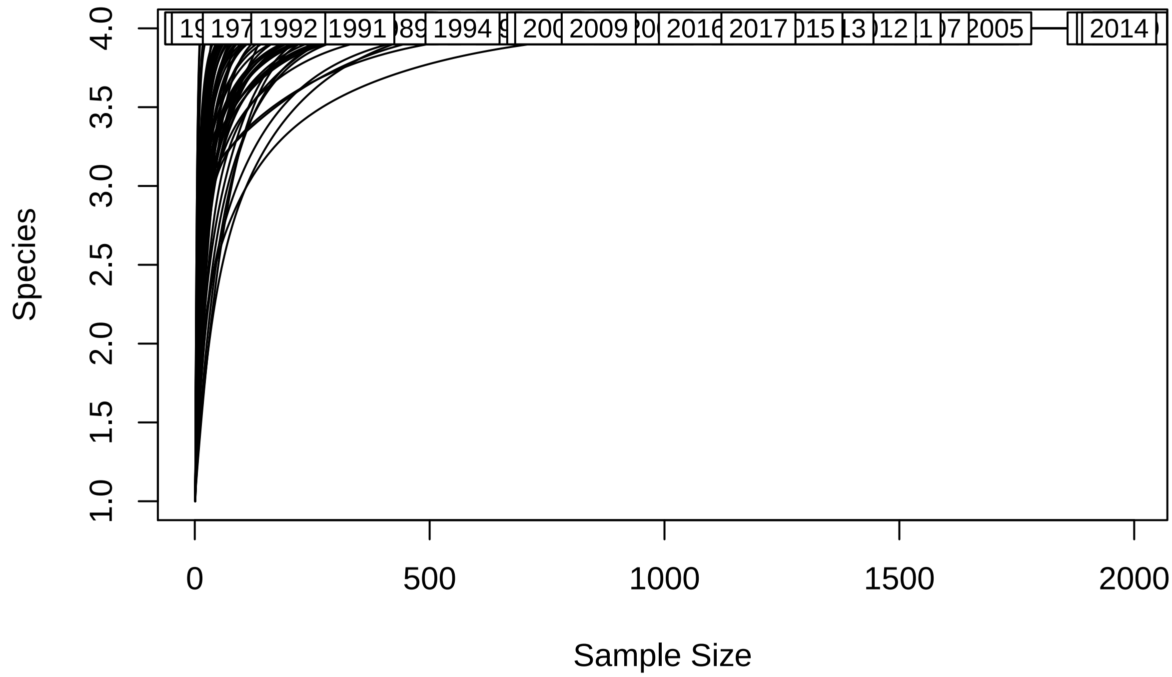
```
rarefy(best, sample=10)
```

```
##     1952     1956     1957     1958     1960     1962     1963     1964
## 1.677800 2.532271 3.892857 1.841751 3.837787 1.407843 2.020324 3.535623
##     1965     1966     1967     1968     1969     1970     1971     1972
## 1.792072 1.888261 1.607926 1.580720 1.454925 1.721593 1.378540 2.059964
##     1973     1974     1975     1976     1977     1979     1980     1981
## 2.963610 1.719705 2.448158 1.700102 2.021976 2.065888 2.229997 2.630100
##     1982     1983     1984     1985     1986     1987     1988     1989
## 3.167738 2.858432 3.165771 3.121535 3.038983 3.003950 3.026253 2.362203
##     1990     1991     1992     1993     1994     1995     1996     1997
## 2.658250 3.145499 3.461613 2.057313 2.557292 2.051739 2.021945 2.158362
##     1998     1999     2000     2001     2002     2003     2004     2005
## 2.683295 2.116202 2.104663 2.536735 1.850560 2.841343 2.356356 2.375419
##     2006     2007     2008     2009     2010     2011     2012     2013
## 1.972593 2.282606 2.260685 2.904340 2.573524 2.595579 2.762594 2.485045
##     2014     2015     2016     2017
## 2.259514 2.778704 2.597883 2.445090
## attr(,"Subsample")
## [1] 10
```

```
head(rarefy(best, sample=c(5, 15)))
```

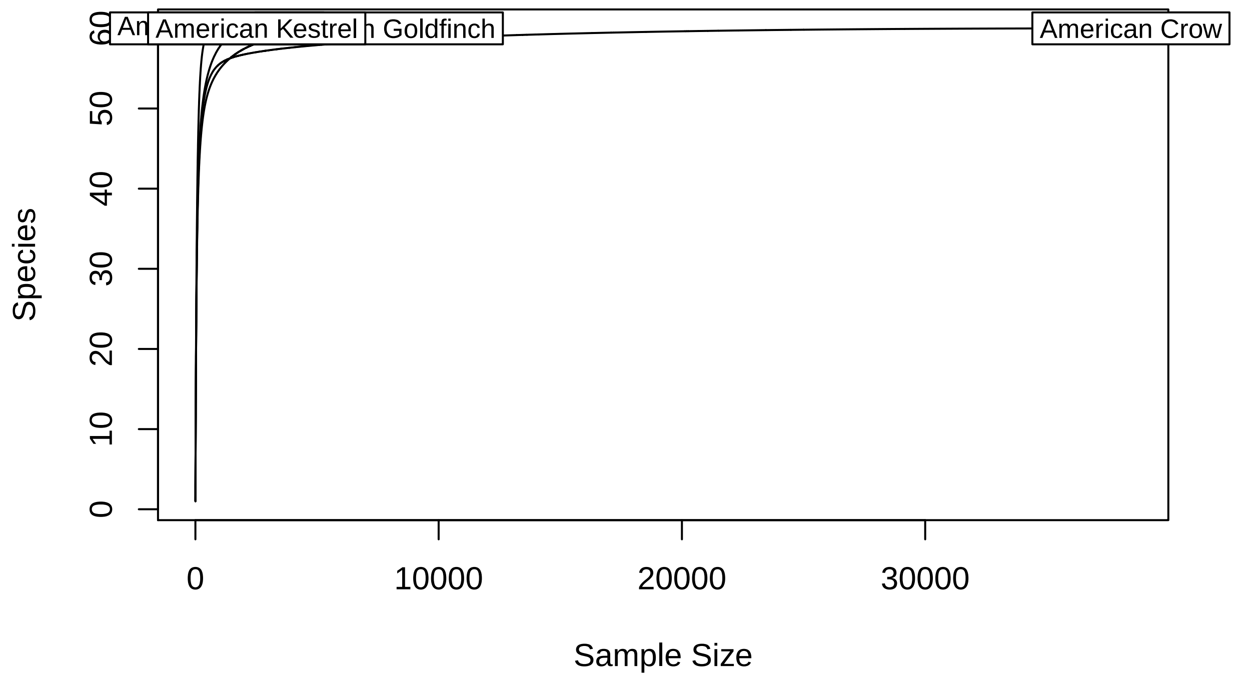
```
##            N5      N15
## [1,] 1.366906 1.941366
## [2,] 1.885565 3.004572
## [3,] 3.087225 4.000000
## [4,] 1.454503 2.171092
## [5,] 3.083562 4.000000
## [6,] 1.216024 1.578600
```

```
rarecurve(best)
```



We can also transpose our matrix with `t()` to look at it by species:

```
rarecurve(t(best))
```



Species accumulation curves

```
?specaccum
```

The `diverse` package has a number of different measures for "complex systems" research, which includes social sciences. A thorough description is available in a published paper on the package.

Ordinal data

For these exercises, let's get some Human Dimensions data for once! Go to the US Forest Service page for the 2004 visitor preference and usage data set for the Bob Marshall Wilderness Complex in Montana: https://www.fs.usda.gov/rds/archive/Product/RDS-2017-0016

At the bottom, click "Download data publication," which gives you a ZIP archive. Open it up, go into the "Data" folder and pull out both CSVs for your `/data` directory. You can hang on to the other files in the archive as well, for the metadata.

For now, let's load in the onsite data:

```
bm = read.csv("./data/BMWC2004_onsitedata.csv", header=T, na.strings="88",
              stringsAsFactors=F)
```

```
head(bm)
```

```
##     id. newweigh    first_ma   reminder      resend date_ret group_.
## 1 2000    1.215 13-JUL-2004 24.07.2004 07-AUG-2004  9/16/04       1
## 2 2001    1.215 13-JUL-2004 24.07.2004 07-AUG-2004  9/16/04       1
## 3 2002    1.215 13-JUL-2004                         7/19/04       2
## 4 2003    1.215 13-JUL-2004 24.07.2004              7/26/04       2
```

```
## 5 2004    1.215 13-JUL-2004 24.07.2004 07-AUG-2004   8/9/04       3
## 6 2005    1.215 13-JUL-2004 24.07.2004 07-AUG-2004                3
##        city st stcode poolstcd zip_code trailhea    date_con sumfall
## 1      Troy MT      1        1    59935       12 18-JUN-2004       1
## 2      Troy MT      1        1    59935       12 18-JUN-2004       1
## 3 Kalispell MT      1        1    59901       12 18-JUN-2004       1
## 4 Kalispell MT      1        1    59901       12 18-JUN-2004       1
## 5  Florance MT      1        1    59833       12 18-JUN-2004       1
## 6  Missoula MT      1        1    59801       12 18-JUN-2004       1
##   time_of entering wilderne overnigh length_o lengcats outfitte type_of
## 1    1900        2        1        1        7        5        2       2
## 2    1900        2        1        1        7        5        2       2
## 3    2000        1        1        1        2        2        2       1
## 4    2000        1        1        1        2        2        2       1
## 5    2030        2        1        1        1        2        2       2
## 6    2030        2        1        1        1        2        2       2
##   hikehors stocknum stockcat numnons         reason_f visitbef prvsvist
## 1        2        7        3       1 Mentally impared        2        0
## 2        2       NA       NA      NA                         2        0
## 3        1        0        0       0                         1       12
## 4        1       NA       NA      NA                         1       10
## 5        2        5        2       0                         2        0
## 6        2       NA       NA      NA                         1        3
##   aware_of affect_p
## 1        1        2
## 2        1        2
## 3        1        1
## 4        1        2
## 5        1        2
## 6        1        2
##                                                          how v28 v29
## 1                                                              2
## 2                                                              2
## 3 The area was basically shut down there was so much caution   2
## 4                                                              2
## 5                                                              2
## 6                                                              2
##   natural remotnes scenic_b hunting fishing recent_f test_ski familiar
## 1       1        1        2       1       1        1        3        2
## 2       1        1        2       1       1        1        3        2
## 3       3        3        3       2       3        1        2        3
## 4       3        3        3       3       3        1        2        2
## 5       2        3        3       3       3        2        2        2
## 6       3        3        3       1       3        1        1        3
##   variety friend_s date_of age agecats educatio female filter_.
## 1       2        1      50  54      54       NA      2        1
## 2       2        1      52  52      52       NA      1        1
## 3       1        2      81  23      23       16      2        0
## 4       2        2      82  22      22       16      2        0
## 5       2        2      61  43      43       14      2        1
## 6       1        1      63  41      41       16      2        1
```

Likert data

```
summary(bm[,36:45])
```

```
##     natural         remotnes        scenic_b        hunting     
##  Min.   :1.00   Min.   :1.000   Min.   :1.000   Min.   :1.000  
##  1st Qu.:2.00   1st Qu.:3.000   1st Qu.:3.000   1st Qu.:1.000  
##  Median :3.00   Median :3.000   Median :3.000   Median :1.000  
##  Mean   :2.67   Mean   :2.768   Mean   :2.844   Mean   :1.554  
##  3rd Qu.:3.00   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:2.000  
##  Max.   :3.00   Max.   :3.000   Max.   :3.000   Max.   :3.000  
##  NA's   :57     NA's   :56      NA's   :56      NA's   :73     
##     fishing         recent_f        test_ski        familiar   
##  Min.   :1.000   Min.   :0.000   Min.   :1.000   Min.   :1.00  
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:1.00  
##  Median :3.000   Median :1.000   Median :2.000   Median :2.00  
##  Mean   :2.221   Mean   :1.494   Mean   :1.744   Mean   :1.62  
##  3rd Qu.:3.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:2.00  
##  Max.   :3.000   Max.   :3.000   Max.   :3.000   Max.   :3.00  
##  NA's   :61      NA's   :61      NA's   :62      NA's   :62    
##     variety         friend_s    
##  Min.   :1.000   Min.   :1.000  
##  1st Qu.:2.000   1st Qu.:1.000  
##  Median :2.000   Median :2.000  
##  Mean   :2.156   Mean   :1.835  
##  3rd Qu.:3.000   3rd Qu.:3.000  
##  Max.   :3.000   Max.   :3.000  
##  NA's   :62      NA's   :70     
```

You can't take the mean of an ordinal variable!

Why not? Remember: we can't assume the intervals between ordinal levels are equal. For example, for this survey, the participants' perceptions of the distance between "Not Important" and "Somewhat Important" may be different from the distance between "Somewhat Important" and "Very Important." So, a mean doesn't make sense.

But you can take the median.

We can also see that one of the Likert variables, `recent_f` has a 0 in it.

```
bm$recent_f
```

```
##   [1]  1  1  1  1  2  1  2  1  1  3  1  2  1  1  1  1  1  1  1  1  1  1  1
##  [24]  2  3  1  1  3  1  1  2  1  2  2  3  2  1  3  2  1  3  1  3  1  3  1
##  [47]  1  1  3  2  1  1  1  1  1  1  2  1  1  1  1  2  1  1  2  1  2  2  1
##  [70]  1  2  1  1  1  2  1  1  2  2  3  1  2  1  1  2  3  3  3  2  2  1  1
##  [93]  3  1  3  1  1  3  1  1  1  2  1  2  1  2  1  1  1 NA  1  1  1  1  1
## [116] NA  2 NA  2 NA  1  1  1  1  2  1  2  3 NA NA  3  3  1  1  1  2  2  1
## [139]  1  1  1  2  2  2  1  1  1  2  1  1  2  2  2  2  1  1  1  2  2  1  1
## [162]  1 NA  1  1  1  1  2 NA NA  3  1  1  2 NA  1 NA  1  2  2  1  2 NA  2
## [185]  1  1  1  1  2  1  1  1  2 NA NA  1 NA  1  1 NA  2 NA NA  2  2  1  2
## [208]  1  3  1  2  1 NA  2  1  2 NA  1  3  1  2  3  2  1  2  1  1  3  0  2
## [231]  1  1  1  1  1  3  1  1  2  1  2  2  1  1  2  1  1  3  1  1  1  2  2
## [254]  2  1  1  1 NA  1  2  1  3 NA  2  2  2  1  1  2  2  1  1  1  1  2  1
## [277] NA  1  1  1  2  1  1  2  2 NA  1  2  1  1  1  1  1  2  2  1  1 NA NA
## [300]  2 NA NA  3  3  1 NA NA NA NA  1 NA  2 NA  1  2  2  1  1  1  1  1  1
## [323]  3  2  1  2 NA NA NA NA  1 NA  2  1  1  1  1  2  1  2  2  2  2  2 NA
```

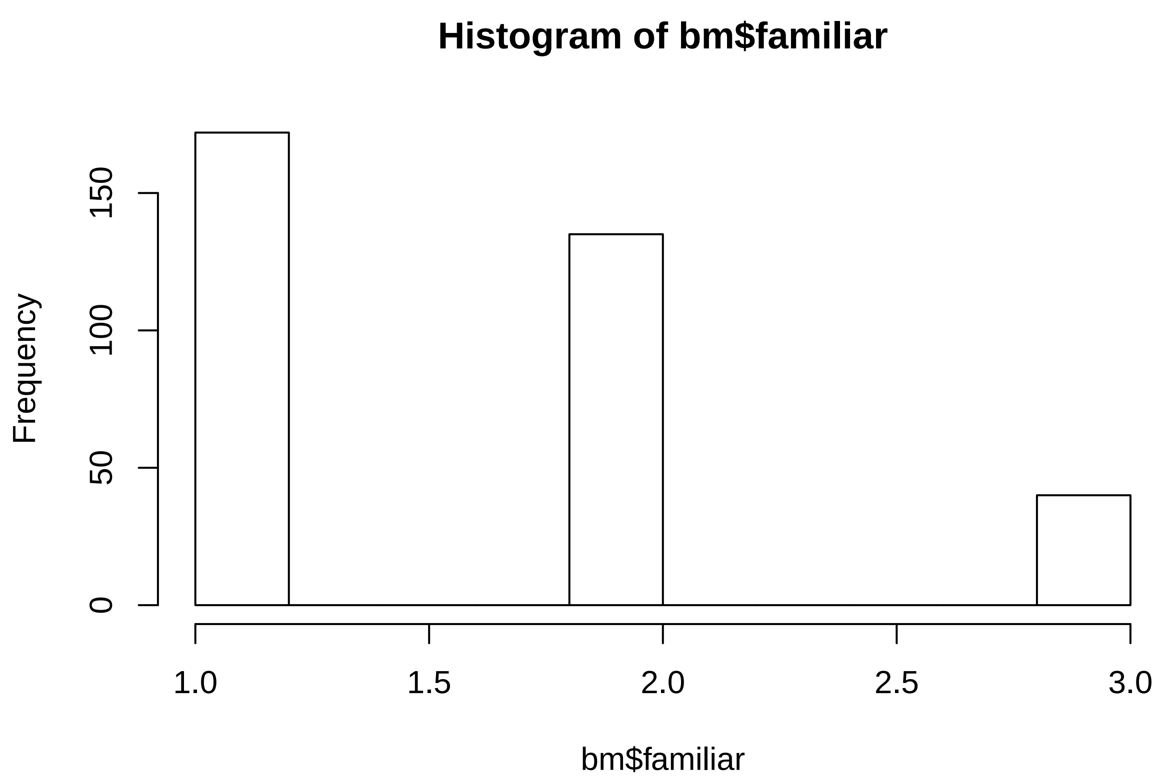
```
## [346] NA NA  2 NA NA  1  2  2  1  2  1  2  2 NA NA NA NA NA  1  1  1  1  1
## [369]  1  1  2 NA NA  1  1  1 NA NA NA NA NA NA NA NA NA NA  1  2  1  1  2
## [392]  2  2  2  2  2  1  2  2  1  1  1  2  1  1  1  1  3  1
```

Since there's only one 0, and the scale goes from 1 to 3 on the survey, this is likely a coding mistake. Might as well fix it and recode it as a missing value.

```
bm$recent_f[bm$recent_f == 0] = NA
```

A good way of visualizing ordinal variables is through the use of a histogram.

```
hist(bm$familiar)
```



A specialized package named `likert` has some additional options.

Hypothesis testing

How do we test whether Likert responses are different by group?

Permutation tests

Permutation tests shuffle around the data to see how often the observed result occurs, to generate a *p*-value. They don't require the assumptions that normal parametric tests do, and can work regardless of the expected distribution, which is why they're good for ordinal data. This is a one-way test, but others are described in the online textbook by Mangiafico.

Functions for permutation tests in R are in the `coin` package:

```
install.packages("coin")
```

```
library(coin)
```

```
## Loading required package: survival
```

Let's examine whether the importance of familiarity was different for Montana residents versus visitors from elsewhere. So, we recode the `st` (state) variable to represent this. We also go ahead and declare our Likert variable as ordered, to make sure R treats it as ordinal.

```
bmLik = bm
bmLik$st = factor(ifelse(bmLik$st != "MT", "Not MT", "MT"))
bmLik$familiar = ordered(bmLik$familiar)
```

We can take a look at the contingency table.

```
table(bmLik$st, bmLik$familiar)
```

```
##
##           1  2  3
##   MT     94 87 26
##   Not MT 78 48 14
```

Difficult to tell, since the row sums are different.

```
?independence_test
```

```
independence_test(familiar ~ st, data=bmLik)
```

```
##
##  Asymptotic General Independence Test
##
## data:  familiar (ordered) by st (MT, Not MT)
## Z = 1.7192, p-value = 0.08558
## alternative hypothesis: two.sided
```

Not significant. Interesting, because we might have expected Montanans to care more about familiarity with the natural area.

As mentioned above, two-way tests, regression, etc. are available on the Mangiafico page.

Polychoric correlations

For ordinal data, we can't use regular Pearson correlations (or Spearman, etc.). Instead, we need to calculate polychoric correlations, which assume that each variable is actually normally distributed, but represented ordinally in the data.

I like to use the `lavCor()` function in the `lavaan` package, because it can take a mix of variable types (numeric, ordinal) and calculate appropriate correlations for each. Other options are available, such as the `tetrachor()` function in the `psych` package:

```
?psych::tetrachor
```

But for now we'll stick with `lavaan`, which we will also use later for structural equation modeling.

```
install.packages("lavaan")
```

```
library(lavaan)
```

```
## This is lavaan 0.6-3
```

```
## lavaan is BETA software! Please report any bugs.
```

Change all of the numeric Likert variables to ordered:

```r
bmLik[,36:45] = lapply(bmLik[,36:45], function(x) ordered(x))
str(bmLik)
```

```
## 'data.frame':	409 obs. of  51 variables:
##  $ id.     : int  2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 ...
##  $ newweigh: num  1.22 1.22 1.22 1.22 1.22 ...
##  $ first_ma: chr  "13-JUL-2004" "13-JUL-2004" "13-JUL-2004" "13-JUL-2004" ...
##  $ reminder: chr  "24.07.2004" "24.07.2004" "" "24.07.2004" ...
##  $ resend  : chr  "07-AUG-2004" "07-AUG-2004" "" "" ...
##  $ date_ret: chr  "9/16/04" "9/16/04" "7/19/04" "7/26/04" ...
##  $ group_. : int  1 1 2 2 3 3 3 4 4 6 ...
##  $ city    : chr  "Troy" "Troy" "Kalispell" "Kalispell" ...
##  $ st      : Factor w/ 2 levels "MT","Not MT": 1 1 1 1 1 1 1 1 1 1 ...
##  $ stcode  : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ poolstcd: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ zip_code: chr  "59935" "59935" "59901" "59901" ...
##  $ trailhea: int  12 12 12 12 12 12 12 12 12 12 ...
##  $ date_con: chr  "18-JUN-2004" "18-JUN-2004" "18-JUN-2004" "18-JUN-2004" ...
##  $ sumfall : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ time_of : int  1900 1900 2000 2000 2030 2030 2030 900 900 1215 ...
##  $ entering: int  2 2 1 1 2 2 2 1 1 1 ...
##  $ wilderne: int  1 1 1 1 1 1 1 1 1 2 ...
##  $ overnigh: int  1 1 1 1 1 1 1 1 1 2 ...
##  $ length_o: int  7 7 2 2 1 1 1 7 7 0 ...
##  $ lengcats: int  5 5 2 2 2 2 2 5 5 1 ...
##  $ outfitte: int  2 2 2 2 2 2 2 1 1 2 ...
##  $ type_of : int  2 2 1 1 2 2 2 4 4 1 ...
##  $ hikehors: int  2 2 1 1 2 2 2 0 0 1 ...
##  $ stocknum: int  7 NA 0 NA 5 NA NA 0 NA 0 ...
##  $ stockcat: int  3 NA 0 NA 2 NA NA 0 NA 0 ...
##  $ numnons : int  1 NA 0 NA 0 NA NA 2 NA 2 ...
##  $ reason_f: chr  "Mentally impared" "" "" "" ...
##  $ visitbef: int  2 2 1 1 2 1 1 2 1 1 ...
##  $ prvsvist: int  0 0 12 10 0 3 10 0 6 9 ...
##  $ aware_of: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ affect_p: int  2 2 1 2 2 2 2 2 2 1 ...
##  $ how     : chr  "" "" "The area was basically shut down there was so much caution" "" ...
##  $ v28     : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ v29     : chr  "" "" "" "" ...
##  $ natural : Ord.factor w/ 3 levels "1"<"2"<"3": 1 1 3 3 2 3 3 3 3 3 ...
##  $ remotnes: Ord.factor w/ 3 levels "1"<"2"<"3": 1 1 3 3 3 3 3 3 3 3 ...
##  $ scenic_b: Ord.factor w/ 3 levels "1"<"2"<"3": 2 2 3 3 3 3 3 3 3 3 ...
##  $ hunting : Ord.factor w/ 3 levels "1"<"2"<"3": 1 1 2 3 3 1 3 2 3 3 ...
##  $ fishing : Ord.factor w/ 3 levels "1"<"2"<"3": 1 1 3 3 3 3 3 3 3 3 ...
##  $ recent_f: Ord.factor w/ 3 levels "1"<"2"<"3": 1 1 1 1 2 1 2 1 1 3 ...
##  $ test_ski: Ord.factor w/ 3 levels "1"<"2"<"3": 3 3 2 2 2 1 2 2 2 3 ...
##  $ familiar: Ord.factor w/ 3 levels "1"<"2"<"3": 2 2 3 2 2 3 2 1 1 3 ...
##  $ variety : Ord.factor w/ 3 levels "1"<"2"<"3": 2 2 1 2 2 1 2 3 3 3 ...
##  $ friend_s: Ord.factor w/ 3 levels "1"<"2"<"3": 1 1 2 2 2 1 2 1 1 3 ...
##  $ date_of : int  50 52 81 82 61 63 77 65 63 82 ...
##  $ age     : int  54 52 23 22 43 41 27 39 41 22 ...
##  $ agecats : int  54 52 23 22 43 41 27 39 41 22 ...
##  $ educatio: int  NA NA 16 16 14 16 12 16 13 13 ...
##  $ female  : int  2 1 2 2 2 2 2 1 2 2 ...
```

```
## $ filter_.: int 1 1 0 0 1 1 1 NA NA 0 ...
```

And calculate our correlation matrix:

```
?lavCor
```

```
bmLikCor = lavCor(bmLik[,36:45])
bmLikCor
```

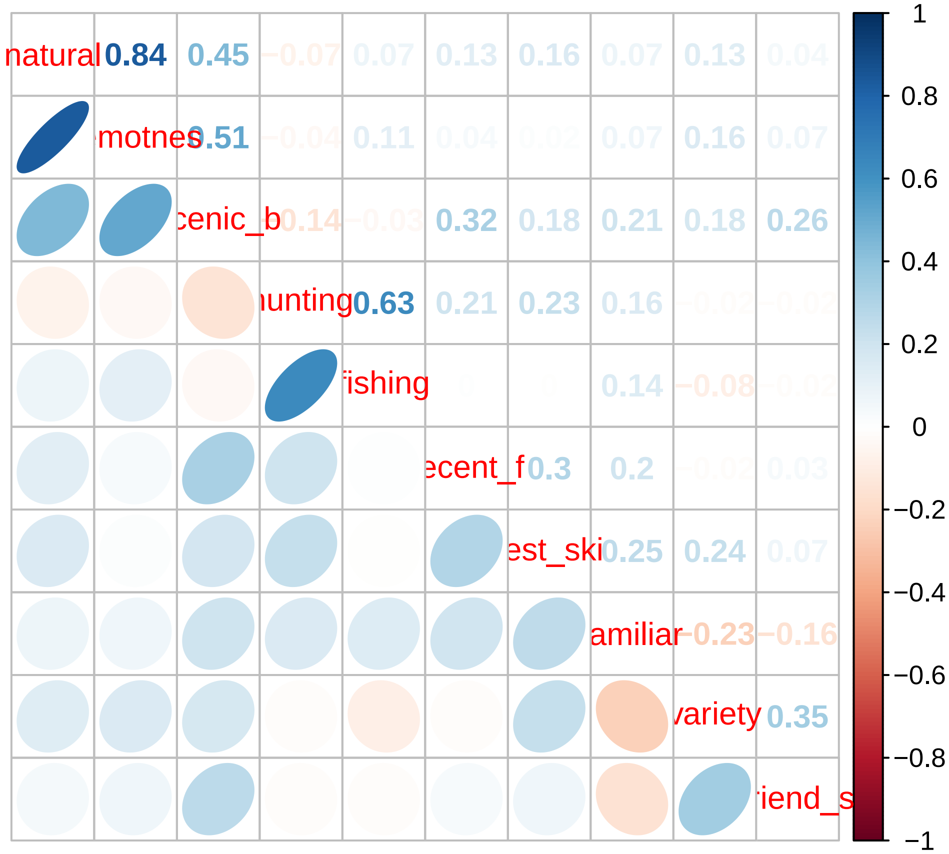
```
##          naturl remtns scnc_b huntng fishng rcnt_f tst_sk familr varity
## natural   1.000
## remotnes  0.837  1.000
## scenic_b  0.447  0.514  1.000
## hunting  -0.068 -0.036 -0.144  1.000
## fishing   0.075  0.112 -0.034  0.633  1.000
## recent_f  0.129  0.037  0.324  0.207  0.003  1.000
## test_ski  0.157  0.018  0.181  0.230 -0.001  0.297  1.000
## familiar  0.074  0.069  0.205  0.156  0.142  0.199  0.255  1.000
## variety   0.131  0.157  0.175 -0.017 -0.084 -0.017  0.237 -0.230  1.000
## friend_s  0.044  0.070  0.262 -0.018 -0.018  0.031  0.067 -0.158  0.348
##          frnd_s
## natural
## remotnes
## scenic_b
## hunting
## fishing
## recent_f
## test_ski
## familiar
## variety
## friend_s  1.000
```

We can also plot it to have a look. I like `corrplot` for its different visualization options.

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
corrplot.mixed(bmLikCor, lower="ellipse", upper="number")
```



Treating ordinal data as continuous

If you have fewer than 5 levels (like we did here), **don't do it**. Your data are unlikely to meet the assumptions of the tests you want to run. You often won't get errors or warnings for doing so, and R will spit out a result, but it's statistically incorrect and your results won't mean anything.

If you have at least 5 levels and good sample size, you're usually okay. Data with 6 or 7 levels are essentially indistinguishable from continuous data. So, when you're designing a survey, go for 6 or 7.

See:

Rhemtulla, M., et al. (2012). When can categorical variables be treated as continuous? A comparison of robust continuous and categorical SEM estimation methods under suboptimal conditions. Psychological Methods, 17(3), 354. doi: 10.1037/a0029315

Saving data

Before we go, let's save our cleaned bird count data for next time.

We can save it in CSV format, similar to the way we read CSVs in:

```
?write.csv
```

```
write.csv(fcbirdW, "./data/fcbirdW.csv")
write.csv(best, "./data/fcbirdbest.csv")
```

If we have a substantially larger data frame (or other object), and we know we'll only need to work with it in R or share it with others using R, we can save any R object as a compressed RDS file to save space:

```
?saveRDS
```

```
saveRDS(fcbirdW, "./data/fcbirdW.RDS")
saveRDS(best, "./data/fcbirdbest.RDS")
```

We could then reload it later with `readRDS()`.

Saving as RDS is also useful if you're working with and processing large files (geospatial raster layers, for example), and want to save the result to load later instead of having to do the processing steps every time.

(pdf / Rmd)