FISH 604: Modern Applied Statistics for Fisheries
F. Mueter

Lab 2: Another bootstrap example and exploring statistical distributions

Work through this lab in small groups or (if you have to) on your own.

Please submit ONE set of solutions per group via Canvas as a text submission. That is, paste your code for each exercise, along with any written answers to questions, as text into the text submission box in Canvas. Clearly indicate each exercise.

Files needed:

Lab 2.pdf	Lab instructions (this file)
Lab 2.R	R script with code
M2winds.csv	Data file with daily wind speed observations over the Bering Sea shelf

1. Use a bootstrap to estimate the distribution of extreme wind events!

The data we will be using are daily wind speed in the East-West direction (u winds) and in the N-S direction (v winds) at a location on the middle shelf of the eastern Bering Sea (centered at 57 N 165 W). The daily data are contained in the comma-delineated file 'M2winds.csv' (take a look at the file using a spreadsheet). Columns are 'Date', u.speed', and 'v.speed' (Wind speeds in m/s).

Import the data:
Make sure you copy the data set (M2winds.csv) to your working directory first.
If you don't know what your current working directory is, use:

```
getwd()
```

Then import the data file using ‘read.csv’ (or ‘read_csv’ if using the tidyverse):

```
M2 <- read.csv("M2winds.csv", as.is=T)
```

The `'as.is'` argument suppresses importing of the Date column as a factor (default behavior for the `read_csv()` function from the ‘readr’ package.

Or, you can open a browser to find the csv file:

```
M2 <- read.csv(file.choose(), as.is=T)
head(M2)   # Examine first few lines of data
```

The Date column is recognized as a string of dates and you can extract days, years, months, etc. You may have to first run: `install.packages("chron")` if you don’t have the package.

```
library(chron)    # loads the chron package
years(M2$Date)    # FACTOR variable of years
months(M2$Date)
days(M2$Date)
```

Compute Julian day (days from beginning of year) and save as an additional column in M2 (for plotting):

```
M2$Julian <- julian(months(M2$Date), days(M2$Date), 1900,
                     origin=c(12,31,1899))
```

Check result:

```
head(M2, 20)
```

Exercise 1

Compute the absolute wind speed using Pythagoras' theorem and add it as another column ('speed') to the existing data frame! (i.e.:

```
M2$speed <- # insert a simple formula to compute absolute wind speed
```

You will need the wind speed column ('speed') before continuing!

Cross-tabulate the data by year and Julian day and save as a new matrix:

```
X <- tapply(M2$speed, list(years(M2$Date), M2$Julian), mean)
```

Note that there is only one value per day, hence the function 'mean' simply returns that value.

If you are not familiar with it, learn about the tapply function and examine the resulting object:

```
X       # The whole matrix (rather large!)
View(X) # To look at data in matrix form: years in rows
dim(X)  # Number of rows and columns
X[,1]   # First column: Wind speed on Julian Day 1 (January 1) for all years
X[1,]   # first row: Daily mean wind speeds for 1948
```

Display the data using a color palette (this is for illustration only, don't worry about figuring out the code - I simply provided it for reference)

```
library(fields)
```

(You may have to first run: `install.packages("fields")`)

```
par(mar=c(3,4,4,6))
image(1:365, 1948:2009, t(X), col=tim.colors(64), xlab="", ylab = "",
              main="Daily wind speeds (m/s), 1949-2009")
```

Add a legend:

```
image.plot(1:365, 1948:2009, t(X), xlab="", ylab = "", yaxt="n",
legend.only=T, col=tim.colors(64))
```

Note that you can do all of the above more efficiently and (to some) more intuitively using the 'tidyverse' approach (a collection of packages designed by Hadley Wickham, chief scientist at R Studio), which has it's own read_csv file (package 'readr') that works slightly differently:

```
library(tidyverse)
M2.tbl <- read_csv("M2winds.csv")
M2.tbl <- M2.tbl %>% transmute(Date, year = years(Date), julian =
julian(months(Date), days(Date), 1900, origin = c(12, 31, 1899)),
                 speed = [insert code for computing windspeed here])
ggplot(M2.tbl, aes(julian, year, fill = speed)) + geom_raster() +
       xlab("Day of Year") + ylab("Year") +
       scale_fill_gradientn(colors=tim.colors(100))
```

Exercise 2 (No need to show any results)
Interpret and discuss the output with your group! What is a typical seasonal pattern of wind speeds? Are there any obvious trends across years from 1948-2009?

Examine the distribution of daily wind speeds across years for each month:

```
library(lattice)
histogram( ~ speed | months(Date), data = M2)
```

Alternatively, using ‘ggplot’:

```
library(ggplot2)
ggplot(M2, aes(speed)) +
  geom_histogram(binwidth=2) +
  facet_wrap(~months(Date))
```

Again, interpret and discuss output with your group!

Exercise 3
Remember that we were going to examine extreme wind events! To do so, we first extract the maximum daily wind speed for each year:

```
W <- apply(X,1,max)  # This applies the function 'max' to each row (=year) of
the data and returns the maximum observed wind speed for each year
sum(is.na(W))         # Check for missing values
W                     # Note missing value in 2009
W <- W[1:61]          # Omit missing value for 2009 (incomplete data)
```

----- An optional alternative approach using the ‘tidyverse’:
Note that we don't need the 'X' matrix at all using the tidyverse syntax:

```
W <- M2.tbl %>% group_by(year) %>% summarize(Max = max(speed))
```

This results in a data frame with two columns, but we can extract ('pull out') the 'Max' column for inspection and for further computations:

```
W %>% pull(Max)
```

Note also that this includes the 2009 maximum:

```
tail(W)
```

This is because the "long-format" data set (`M2.tbl`) does not have any missing values in 2009 (hence computing the maximum does not result in ‘NA’ unlike above where we compute the maximum over a row of the full matrix that includes missing values).

Since the 2009 value may be biased, we'll remove it and save ‘W’ as a simple vector:

```
W <- W %>% filter(year != 2009) %>% pull(Max)
```

------ End of optional section

(a) Plot a histogram or empirical density estimate of maximum annual wind speeds
What does the distribution of extreme values look like? (symmetrical, skewed?)

As an aside, the distribution of extreme events is often approximated by a "Gumbel" distribution, something we will get back to at the end of the lab!

(b) Compute the 90th percentile of the distribution using the '`quantile`' function

The 90th percentile may be taken as the magnitude of a once in a decade wind event (1 out of 10 years have a wind event as large or larger)

(c) Using a bootstrap approach, we will estimate and visualize the distribution of a once-in-a-decade wind event and calculate the mean and variance of its bootstrap distribution.

Adapt the code for bootstrapping the median to bootstrapping the 90th percentile. You can simply replace the call to ‘median’ with an appropriate call to ‘quantile’ to compute the 90th percentile.

Make sure to understand the bootstrap code! The mean of the bootstrapped quantiles (call it $\hat{\theta}$) can be taken as an estimate of the average magnitude of a ‘once-in-a-decade’ wind event and the standard deviation is an estimate of the associated uncertainty. Compute two possible confidence intervals for our ‘once-in-a-decade’ winds:

1. One possibility is to assume that the estimates are approximately normally distributed, hence we can estimate a 95% confidence interval as: $\hat{\theta} \pm 1.96 * se(\hat{\theta})$ (Why?)
2. A second possibility is to use the central 95% of the bootstrap distribution (rather than a normal distribution) as a confidence interval. This requires computing the 2.5th and 97.5th percentiles of the bootstrapped values in ‘`out`’, which you can compute using '`quantile`'.

2. Distributions

We covered a number of discrete distributions in class. Each of these distributions are implemented in R and you can easily compute probabilities, quantiles corresponding to certain probabilities, and generate random numbers from these distributions using the following built-in functions. Most distributions have four functions associated with them that each serve a different purpose a detailed below. The functions start with '`r`', '`d`', '`p`', and '`q`', followed by the (abbreviated) name of the distribution, for example for the normal distribution:

```
dnorm()  # Compute density (pdf) of normal distribution
pnorm()  # Compute cumulative pdf of   "          "
qnorm()  # Compute quantiles of normal distribution
rnorm()  # Generate random numbers from normal distribution
```

Functions for discrete distributions are (among others available in other packages):

```
dbinom, pbinom, qbinom, rbinom          # (Bernoulli &) Binomial distribution
dmultinom, rmultinom (no pmultinom and qmultinom)      # Multinomial
dpois, ... (etc.)                                      # Poisson
dnbinom                                                # Negative binomial
```

We explored the Bernoulli & binomial distributions, and will encounter the Poisson distribution again later!

Multinomial distribution example

Example: Remember one of the examples of the multinomial distribution from class. Long-term frequencies suggest that of the sockeye salmon returning to Bristol Bay about 22% are 3 years old, 63% are 4 years old, and 15% are 5 years old

```
p <- c(0.22, 0.63, 0.15)
```

Say we catch 3 fish. What is the probability of catching exactly one fish of each age? We can use the formula that I showed in the recorded lecture, which is computed in R using `dmultinom()`:

```
dmultinom(c(1,1,1), size=3, prob=p)
```

Exercise 4

What is the probability of a sample of 8 fish consisting of 1 age-3, 4 age-4, and 3 age-5 fish given the above vector of probabilities (p)?

Do the computation both "by hand", i.e. based on the definition of the multinomial probability distribution given in the recorded lecture or in slide 15 of '*Mod2 Basics (2) distributions (discrete*.)*' (use the function "`factorial()`"). Compute the same probability using the function '`dmultinom()`'

Because there are lots of different possible combinations of fish at each age if we have more than 3 fish, it is generally not simple (and not particularly useful) to compute all of the corresponding probabilities. However, for n=3 fish in r=3 age classes as in the above example, here is how you could do it if you had to (this generalizes to more than 3, but for simplicity it is illustrated here for this simple case):
You can safely skip this part if you are running out of time!

```
X <- t(as.matrix(expand.grid(0:3, 0:3)))
X <- X[, colSums(X) <= 3]
X <- rbind(X, 3:3 - colSums(X))
dimnames(X) <- list(paste("age",1:3,sep="-"), NULL)
X
```

We can compute the probability of getting each combination of 3 fish (given p!) by applying '`dmultinom()`' to each column of X:

```
round(apply(X, 2, function(x) dmultinom(x, prob = p)), 3)
```

These probabilities should sum to 1 since there are only these 10 possible combinations

Exercise 5

Drawing random numbers from a multinomial distribution:
Given the above vector of probabilities (p), use the `rmultinom()` function to draw 100 random multinomial age distributions that you might expect in a sample of 50 salmon. Save the result (i.e. assign the result from the call to `rmultinom` to a named object!

Use the saved results to compute (1) the average number of fish by age class across all 100 samples (you can use the apply function with FUN = mean (Check `?apply` to find out more!)) and (2) the standard deviation (FUN = sd) of the numbers at age.

How do these values from your randomly generated 'data' compare to the expected numbers of fish per age class (=$n*p_i$ fish in age class i) and their standard deviation [= square root of $n* p_i$ $*(1-p_i)$]?

More on the Gumbel distribution and defining distributions:

As noted above, the distribution of extreme values is often approximated by a 'Gumbel' distribution. We can check if the Gumbel distribution provides a reasonable approximation to the observed wind speed maxima.

There is no built-in 'Gumbel' distribution in the R base packages! However, we can easily define the required functions if we know the formulas for the corresponding probability density function (pdf), cumulative density function (cdf) and quantile function. All of these can easily be found online or in stats books:

Probability density function:
```
dgumbel <- function(x,mu,s){
  exp((mu - x)/s - exp((mu - x)/s))/s
}
```

Cumulative probability function:
```
pgumbel <- function(q,mu,s){
  exp(-exp(-((q - mu)/s)))
}
```

Quantile function:
```
qgumbel <- function(p, mu, s){
  mu-s*log(-log(p))
}
```

Recall our maximum wind speeds:
```
W
```

Using the above functions, we can estimate the parameters of the distribution that provides the best fit to our wind speed data. We can use a function in the 'fitdistrplus' package called '`fitdist()`', which uses more advanced approaches to fitting a curve to data that we haven’t covered yet and has several options. We use maximum likelihood estimation (mle), a method that I will cover in more detail later. This uses an iterative algorithm and we need to provide reasonable starting values - using the empirical mean and standard deviation is very reasonable as the parameters of the Gumbel distribution correspond to the theoretical mean and standard deviation:

```
gumbel.fit <- fitdist(W, "gumbel", start=list(mu=mean(W), s=sd(W)),
           method="mle")
summary(gumbel.fit)
```

Let's check how the approximation works visually:

```
hist(W, freq=F)
x <- seq(16,24, by=0.2)
lines(x, dgumbel(x, 18.33, 1.046), col=2, lwd=2)
```

Or we can take advantage of the built-in plotting function to assess goodness of fit:

```
plot(gumbel.fit)
```

The Gumbel distribution appears to approximate the distribution of maximum wind speeds quite well.

For comparison, let's fit a log-normal distribution, which is also right-skewed:

```
lognormal.fit <- fitdist(W, "lnorm", start=list(meanlog=mean(W),
      sdlog=sd(W)), method="mle")
summary(lognormal.fit)
plot(lognormal.fit)
```

Because both models are fit using maximum likelihood, we can use the Akaike Information Criterion (AIC, more on that later), to compare the two fits. Note that the Gumbel distribution fits slightly better than the log-normal (AIC is lower by about 2.3)

A little searching online reveals that the Gumbel distribution has been implemented in the QRM package (among others):

```
library(QRM)
hist(W, freq=F)
lines(x, dGumbel(x, 18.33, 1.046), col=2, lwd=2)
```

# where '`dGumbel()`' computes the density of the Gumbel distribution (spelled with upper-case 'G') and the result is identical to what we get using our own function '`dgumbel()`'!