# FISH/MSL 604

Franz Mueter

# Lab 3: Data exploration, multicollinearity, robust regression

Please work through the lab and insert code and answers to questions directly in the script file, then upload script files to Canvas (one per group).

**Files needed:**

| | |
|---|---|
| *Lab 3 EDA multicollinearity.pdf* | This file |
| *Lab 3 EDA.R* | Script file |
| *Beaufort.csv* | Data file |

Import the Beaufort Sea data set (copy to your working directory):

```
Beaufort <- read.csv("Beaufort.csv")
```

## 1. Explore and analyze trends in environmental time series

The object 'Beaufort' contains 22 years of Arctic Cisco (*Coregonus autumnalis*) abundances (catch-per-unit-effort at age 0) and environmental data for 5 variables measured annually in the coastal Beaufort Sea. We wish to relate recruitment to the following explanatory variables:

| *Variable* | *Description* |
|---|---|
| summer.airT | Summer mean temperatures (June-September) at Barrow airport ($^{o}$C) |
| summer.SST | Mean July-September SST averaged over 70-71N, 146-152W based on Optimum Interpolation v. 2 SST |
| spring.ice | Mean June-July ice concentrations (%) over 70-71N, 146-152W off the Colville R. |
| Oct.ice | Mean Oct ice concentrations (%) over 70-71N, 146-152W off the Colville R. |
| Sag.discharge | Average daily discharge (cfs) for Sagavanirktok River as measured at Pump Station 3, May 1 - September 30 |
| Wind | Mean hourly wind speed (m/s) at Deadhorse Airport for the period July 1 - August 31 |

First, do a quick exploration of the environmental data by plotting the time series for each variable with a linear time trend (via simple linear regression on 'Year') and a loess smooth. [We do so by cycling over all variables using a 'for' loop. We could also use '`xyplot()`' {lattice package}, the 'ggplot2' package, or other tools for multi-panel displays – see script file]. The linear trend gives some indication of whether there is an overall increase or decrease and the loess smooth gives a better indication of the "mean" trend in the data (i.e. is it linear? are their decadal-scale patterns? is it stationary or "flat"?, etc).

We set `par(ask=T)`, which will pause after each plot until you click on the graphics window:

```
for(i in names(Beaufort)[3:8]) {     # cycle through names in data frame
                                     # (environmental data only - vars 3-8)
      x <- Beaufort$Year             # extract years for plotting
      y <- Beaufort[,i]              # extract ith variable
# Plot time series for variable i using both points and lines,
#  suppress default axis labels and add variable name as title:
      plot(x, y, type = "b", xlab="", ylab="", main = i)
# Fit linear time trend and add line to plot:
      abline(lm(y ~ x), col=4)
# Fit a LOWESS (= LOESS) smooth trend and add fitted line to plot:
      fit <- loess(y ~ x)
      lines(x, fitted(fit), col=2)  # Add fitted line
}
```

Of course you can also display all plots on a single page by changing the layout. For example:
`par(mfrow=c(3,2))`
(Re-run for loop)

For ease of comparison, plot all environmental time series in a single panel using '`matplot`' :
`matplot(Beaufort[,-(1:2)], type="l")`
*(Note that the first two columns with the 'Year'and 'Cisco' variables are exclude by using '-(1:2)' as subscript)*

In this case, this is not very helpful because of the large differences in scale among variables! To scale each variable you can use the '`scale()`' function. By default it will subtract the mean from each column and divide by the standard deviation to 'normalize' the variables. This normalization will not change the patterns of variability over time:

```
matplot(scale(Beaufort[,-1]), type="l")
legend("topright",names(Beaufort)[-1], lty=1:7, col=1:7)
```

The script file also includes code to plot trends with confidence bands using the '`ggplot()`' function, which provides powerful tools to visualize fits for a variety of relatively simple models.

## 2. Correlations among time series (Multicollinearity)

Examine all pairwise scatterplots (this is the default behaviour of '`plot()`' IF its first argument is a data frame with all numeric variables):

```
plot(Beaufort[,-(1:2)])
```

➔ Are there any obvious relationships among the variables?

We can more formally assess correlations by testing correlation coefficients for significance. Note that several of the variables are highly correlated with each other (by default, `cor()` computes Pearson's product moment correlations):

```
cor(Beaufort[,-(1:2)])
```

To test individual correlations for significance, use `cor.test()` for specific pairwise comparisons:

```
cor.test(Beaufort$summer.airT, Beaufort$summer.SST)
```

➔ Are air and water temperatures significantly correlated?

You can do all pairwise tests simultaneously using the '`rcorr()`' function written by Frank Harrell and located in his 'Hmisc' package. Note that '`rcorr()`', unlike `cor()`, does not take a data frame as input. Therefore we must convert the data frame to a matrix first:

```
library(Hmisc)
Env <- as.matrix(Beaufort[,3:8])  # Save environmental variables as a matrix
rcorr(Env)
```

The output includes the sample size for each pairwise comparison (N) and the p-values corresponding to the usual test of the null hypothesis that r = 0. You can save the output and extract the correlations and p-values as follows:

```
COR <- rcorr(Env)
COR$r   # Matrix of correlations
COR$P   # Matrix of p-value
```

For a more robust measure, we could compute rank-based correlations that are less susceptible to outliers but also have lower power! There are two rank-based tests based on 'Kendall's Tau' and 'Spearman's rho'. Both are implemented in the '`cor()`' and '`cor.test()`' functions, but only Spearman's rho is implemented in '`rcorr()`'

**Exercise 1:** Look at the helpfile for '`rcorr`' and compute pairwise rank-based correlations (how do they differ from the classical 'Pearson's product moment correlations'?) with the corresponding tests for significance. Compare results to those based on Pearson's product-moment correlations!

Clearly, many of the variables are related to temperature conditions and are highly correlated with each other. This is known as multicollinearity and can have important consequences for the variance of estimated regression coefficients and for evaluating the significance of individual variables (see Appendix 1).

One way to deal with a number of inter-correlated explanatory variables is to use a variable reduction technique called 'Principal Components Analysis' (see Appendix 2). This may allow us to reduce these 6 inter-correlated variables to 1 or 2 "principal components", which are linear combinations of the variables that, unlike the original variables, are (by definition) uncorrelated with each other:

```
pca <- princomp(x = Env, cor=T)  # Force use of correlation matrix
summary(pca2)
plot(pca)   # Scree plot (variance explained by each PC
print(loadings(pca2), cutoff=0)    # show all loadings
```

The results suggest that 56% of the overall variability is captured by a single principal component (PC 1), which is a new time series with the same length as the original variables. The loadings show that the first PC is primarily related to temperature conditions (the temperature and ice variables have high loadings for this PC). PC 2 is primarily related to wind conditions, and PC3 is primarily related to discharge. These three variables reflect the three main "dimensions" of the data (temperature conditions, local wind conditions, freshwater runoff).

The loadings show how the values for PC 1 are computed:

```
 PC 1 =  - 0.401*summer.airT  - 0.508*summer.SST + 0.482*spring.ice
           + 0.461*Oct.ice - 0.367*Sag.discharge - 0.033*Wind
```

The high proportion of variability explained by PC1 implies that there is a lot of redundancy in the data, which makes sense because SST, air temperature, and ice concentrations all reflect general "temperature conditions". Hence, we may want to reduce these variables to a single variable (i.e. 'principal component') using a PCA that includes only these variables:

```
pca2 <- princomp(x = Env[,1:4], cor=T)  # Force use of correlation matrix
summary(pca2)
plot(pca2)   # Scree plot (variance explained by each PCplot(pca)
print(loadings(pca2), cutoff=0)    # show all loadings
pca2$scores    # Show all principal component scores!
```

The results show that PC 1 accounts for 75% of the variability in these four variables, hence we may wish to use this variable that describes overall temperature conditions instead of the original 4 variables in a regression, which greatly simplifies further analyses (this is also known as "Principal Component Regression (PCR"). In our case, high values of PC1 imply cold conditions (positive loadings on ice, negative loadings on SST and air temp). We'll add the temperature PC to the original data frame for the analysis:

```
Beaufort$Temp.PC1 <- pca2$scores[,1]
```

The wind and discharge variables were not strongly correlated with the temperature variables, hence we include them individually in a model of Cisco abundances, along with our general "temperature" variable:

```
fit1 <- lm(Cisco ~ Temp.PC1 + Wind + Sag.discharge, data = Beaufort)
summary(fit1)
par(mfrow=c(2,2))
plot(fit1)
```

The result suggests that the temperature variable is not significant, hence we may want to eliminate it. However, the diagnostic plots suggest some large outliers and potential heteroscedasticity, along with some influential points (large residuals with high "leverage").

## 3. Box-Cox transformations

We will take a closer look at the distribution of the residuals to see if a transformation may be warranted! A histogram suggests that the distribution of Cisco abundances (or, rather, residual abundances) is somewhat right-skewed:

```
hist(r <- resid(fit1))
```

and a Shapiro-Wilks test rejects normality, which is somewhat troublesome for such a small dataset!

We can also test formally for skewness to see if a transformation may be needed, although the Box-Cox approach itself is probably preferable in determining an appropriate transformation (or no transformation). A test for skewness, against the null hypothesis that the data are normally distributed (symmetrical) can be found in the 'moments' package. The test is based on D'Agostino (D'Agostino R.B. 1970. Transformation to Normality of the Null Distribution of G1. Biometrika, 57, 3, 679-681):

You may have to install the package first:

```
install.packages("moments")
library(moments)
```

You can compute a measure of skewness using a function with that name. Normally distributed data have skewness = 0:

```
skewness(r)
```

As suspected, the data are positively skewed, but could this just occur by chance even with normally distributed data? To find out, we can test for significance using the approach of D'Agostino, which is implemented in the following function:

```
agostino.test(r)
```

The result is significant, providing some evidence of skewness.

We can use the `boxcox()` function to explore the effect of different transformations on the 'likelihood' of the data, given the model, to select a suitable transformation, if any. Note that we have to add a small positive value to 'Cisco' because there are zeros in the abundance data, which cannot be accommodated by the Box-Cox transformation. I dropped the temperature variable from the model because it was not significant:

```
boxcox(lm(Cisco+1 ~ Wind + Sag.discharge, data = Beaufort))
```

The resulting likelihood profile shows that a log-transformation may be appropriate. That is, the optimum lambda is approximately zero, corresponding to a log-transformation, and is significantly different from 1 (no transformation).

Hence we use a log-transformation and re-fit the model:

```
fit2 <- lm(log(Cisco+1) ~ Temp.PC1 + Wind + Sag.discharge, data = Beaufort)
summary(fit2); plot(fit2)
```

**Exercise 2:** Examine the output, in particular the t-tests for each of the explanatory variables, which test the null hypothesis that the corresponding coefficient is equal to zero. Are any of the

coefficients not significant? If so, simply eliminate the non-significant variable(s) and re-fit the model (We will discuss more 'rigorous' model selection approaches later)! Save the model output as 'fit3', examine model diagnostics (`plot(fit3)`), and test residuals for normality (`shapiro.test()`). Does the resulting model provide an adequate fit?

Plot the resulting fitted model with pointwise standard errors using ' `termplot()`' (partial fit or fits):

```
termplot(fit3, se=T, partial.resid=T)
```

An alternative that provides a more flexible approach to visualizing model fits is the '`visreg()`' function in a package of the same name (see script file).

**4. Outliers and robust regression:**

→ Exercise 3: Using the Beaufort data, plot summer sea-surface temperature against spring ice conditions, fit a simple linear regression of SST on ice concentration (using '`lm()`'), save the resulting object as '`fit.lm`', and add the fitted line to the plot. In using a regression approach (rather than, for example, a correlation between the two variables), we assume that the ice variable (x-variable) is measured without error. Assuming that is the case, is there a significant effect of spring ice concentration on summer SST? (Do you think this "effect" reflects a causal relationship?)

Note what looks like a possible outlier in the upper left. First identify what year the outlier corresponds to! To do so, you can use the following function:

```
identify(x=??,y=??)
```

where the inputs for arguments x and y are the same variables that you used to generate the plot (i.e. insert the time series of ice concentration and SST for x and y). You can then click on or near any point on the plot to identify points by row number. Use right-click or hit 'Esc' to exit interactive mode. To identify points by Year, use:

```
identify(x=??, y=??, labels = Beaufort$Year)
```

When you are done identifying points on the plot, be sure to right-click and select 'Stop'

You can examine whether the apparent outlier has a strong influence on the regression by looking, for example, at the "Cook's distance' measure and / or looking at diagnostic plots:

```
win.graph(8,4)                  # Open a second graphics window
par(mfrow=c(1,2))               # change layout (2 side-by-side plots)
plot(fit.lm, which=c(4,5))      # Select influence plots (see help for 'plot.lm')
```

Note that the first plot shows Cook's distances, which can be computed using

```
cooks.distance(fit.lm)
```

Note that the year which looked like a potential outlier has very large "leverage" and a very large 'Cook's distance', which means that it has a strong influence on the regression line.

```
dev.off()                    # Close current graphics device
```

To deal with this potential outlier, we could use a robust regression, which identifies outliers and down-weights them.

→ Exercise 4: Use the function '`rlm()`' (in library MASS) to fit a robust regression to the same data and add the fitted line to the scatterplot of summer SST against spring ice. The syntax for '`rlm()`' is very similar to that used for '`lm()`', i.e. you only have to specify a formula and the data frame where to find the variables. Save your output from the call to '`rlm()`'(for example as '`fit.rob`') and add the fitted line to the plot (you can simply use '`abline()`' which also works with objects of class 'rlm').

You can see which, if any, observations were down-weighted in the regression by looking at the weight used in the final iteration:

```
fit.rob$w
```

Note that two years were down-weighted (because the magnitudes of the residuals exceeded a pre-defined threshold – for more details on robust regression, check the references in the '`rlm`' documentation, in particular Hampel et al. 1986)

The year in question was an outlier because of very unusual (low) ice conditions in the spring, resulting from unusual wind patterns. If the goal is to predict summer temperatures from spring ice conditions for a "typical" year, we may want to eliminate the unusual observation entirely.

→ Exercise 5: Use the 'subset' argument to '`lm()`' to re-fit the model (`subset = -n`) where *n* is the row corresponding to the outlier. Add the resulting fitted line to the plot! How and why do the three fitted lines (OLS, robust regression, outlier excluded) differ?

## Appendix 1: Multi-collinearity and its effects

Often the independent variables in a regression analysis, analysis of variance, or another type of model are correlated among themselves. When two variables are highly correlated there will be a wide range of possible regression coefficients that fit the data almost equally well. Thus the coefficients have a large confidence interval and are themselves highly correlated. This also means that it is not possible to ascribe a certain effect to one or the other variable due to their correlation. Correlated variables make for redundancy in the data, thus adding a variable that is correlated with one or more of the other variables adds little information. Adding such a variable to a model may however greatly change the regression coefficients of correlated variables that were already in the model. If variables are uncorrelated, on the other hand, the effects ascribed to each variable are the same no matter which other variables are added. When correlated variables are present in the model a good fit can still be obtained and predictions can be made, but the regression coefficients will be poorly estimated and will have high variability. Very different combinations of the coefficients can lead to equally good predictions and the coefficients cannot easily be interpreted.

As another consequence of high correlations the confidence intervals and hypothesis tests on the coefficients are no longer valid. A test of a number of highly correlated coefficients may show that each of the coefficients, when tested individually, is no not significantly different from zero, although the corresponding variables taken together may have a lot of explanatory power.

One approach to dealing with multi-collinearity is to apply a Principal Components Analysis to the explanatory variables to find a reduced set of (independent) variables that capture most of the variability in the data.

## Appendix 2: Principal Components Analysis

The objective of PCA is, given a number (p) of variables, to produce p uncorrelated indices (principal components, PC) that are linear combinations of the p original variables. The indices are then ordered by the amount of variation they display. The total amount of variation in the indices is equal to the total variation in the original variables, but it is hoped that the variance of some of the indices (PC's) is so low that they can be ignored. Thus, ideally, a few PC's explain most of the variation present in the original variables.

The best results can be expected when the original variables are highly correlated. In that case they contain a lot of redundant information that can be effectively summarized using fewer PC's. If the original variables are completely uncorrelated a PCA will not be very useful.

Procedure: The goal is to find an index

$$Z_1 = a_{11} X_1 + a_{12} X_2 + ... + a_{1p} X_p$$

under the constraint: $a_{11}^2 + a_{12}^2 + ... + a_{1p}^2 = 1$
such that the variance of $Z_1$, var($Z_1$), is maximized.

The second PC is found likewise as a linear combination:

$$Z_2 = a_{21} X_1 + a_{22} X_2 + ... + a_{2p} X_p$$

under constraints that: $a_{21}^2 + a_{22}^2 + ... + a_{2p}^2 = 1$

and that: $Z_2$ is uncorrelated with $Z_1$

All other components are found likewise. The constraint that the sum of the squared coefficients is equal to 1 is necessary because otherwise the variance for any index can simply be increased by increasing the coefficient $a_{ij}$

Generally the variables are standardized before PC's are computed to avoid any one variable having an undue influence on the PC's. If the values for one variable were much larger for one of the variables than for the others the PC would all have a high coefficients for this variable. This is especially important if variables were measured at different scales or in completely different units.

In practice, the amount of variance explained by each PC and the coefficients for each PC are obtained by an eigenanalysis of the covariance (or correlation) matrix of the variables. The eigenvalues ($\lambda$) of this matrix are the variances of the principal components and the corresponding eigenvectors give the coefficients of the variables used to compute the principal components. In effect, PCA only rotates the data to orient the axes along directions of maximum spread.

To evaluate how important the contribution of each principal component is one can:

- divide each eigenvalue by the sum of all eigenvalues to determine how much of the variance is explained by each principal component (fraction), then pick a "cut-off point" e.g. use all those PC's who, taken together, account for 90% of the variability or more.
- only use PC's whose eigenvalues are larger than one, the reason being that eigenvalues smaller than 1 explain less of the overall variation than each of the original variables
- plot eigenvalues in decreasing order and see if a sharp drop indicates an appropriate cutoff point.
- A formal test for the number of "interpretable" principal components is implemented in the broken-stick approach (Jackson, D.A. 1993. Stopping Rules in Principal Components Analysis: A Comparison of Heuristical and Statistical Approaches. Ecology 74(8): 2204-2214). Not sure if it has been implemented in R but it is easy to code!