Time Series HW 6

Kenny Flagg
Andrea Mack

October 18, 2016

```
## Error:  <text>:14:0:  unexpected end of input
## 12:  options(xtable.table.placement = "H", xtable.caption.placement = "top",
## 13:         #width = 80, scipen = 1, show.signif.stars = FALSE)
##     ^
```

We will explore one series like those you found in the Vincent and Meki's paper, but for Bozeman. The following code will count the days in Bozeman where the minimum temperature was measured to be below 32 degrees F (0 degrees C) and the number of days where information was available in "Data1".

1. *Make nice looking and labeled time series plots of the number of days below freezing and the proportion of measured days below freezing.*

```
 ggplot(data = Data1, aes(x = Year, y = DaysBelow32)) + geom_point() + scale_x_continuous(
```

```
## Error in eval(expr, envir, enclos):  could not find function "ggplot"
```

```
ggplot(data = Data1, aes(x = Year, y = PropDays)) + geom_point() + scale_x_continuous(brea
```

```
## Error in eval(expr, envir, enclos):  could not find function "ggplot"
```

2. *Estimate a linear trend model for the proportion of measured days below freezing and report the parametric (t-test) linear trend test results in a sentence. Also discuss scope of inference for this test in a sentence or two (random sampling and random assignment and their implications).*

3. *Discuss this proportion response versus using the count of days below zero per year, specific to this example and in general. What issues does using one or the other present?*

   Specific in to these data, temperature was not recorded every day of each year.To report the number of days below 32^oF may be misleading if there is a large discrepancy in the total number of days temperature was recorded each year. The proportion of days may still be misleading depending on which days of the year temperature was recorded, but it puts each year on the same scale.

4. *Generate a permutation test for the trend with the proportion response. I performed one in the syllabus (page 6) using the "shuffle" function from the "mosaic" package. Report a plot of the permutation distribution, the* **test statistic** *you used, and a p-value. Generally randomization based tests are more robust to violations of the normality assumption as long as the distribution (shape and variability) is the same for all observations except for differences in the center or mean. Why would that be advantageous with this response?*

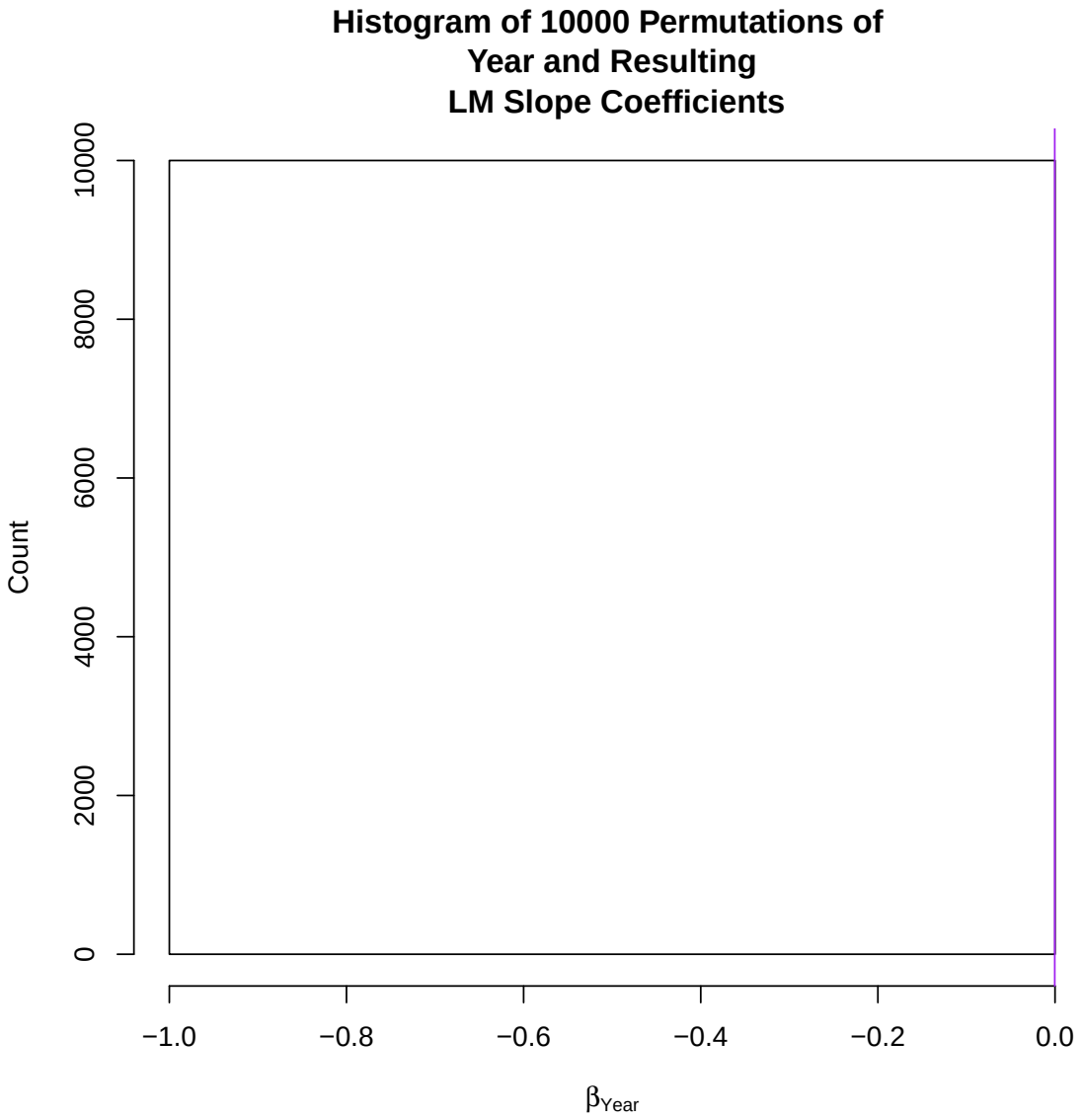
```
lm <- numeric(10000)
lm_year.b <- NULL

for(i in 1:10000){
  lm[i] <- coef(lm(Data1$PropDays ~ shuffle(Data1$Year)))[2]
  }

## Error in eval(expr, envir, enclos):  could not find function "shuffle"

lm.Year <- lm(PropDays ~ Year, data = Data1)

hist(lm, main = "Histogram of 10000 Permutations of \n Year and Resulting \n LM Slope Coef
abline(v = coef(lm.Year)[2], col = "purple")
```



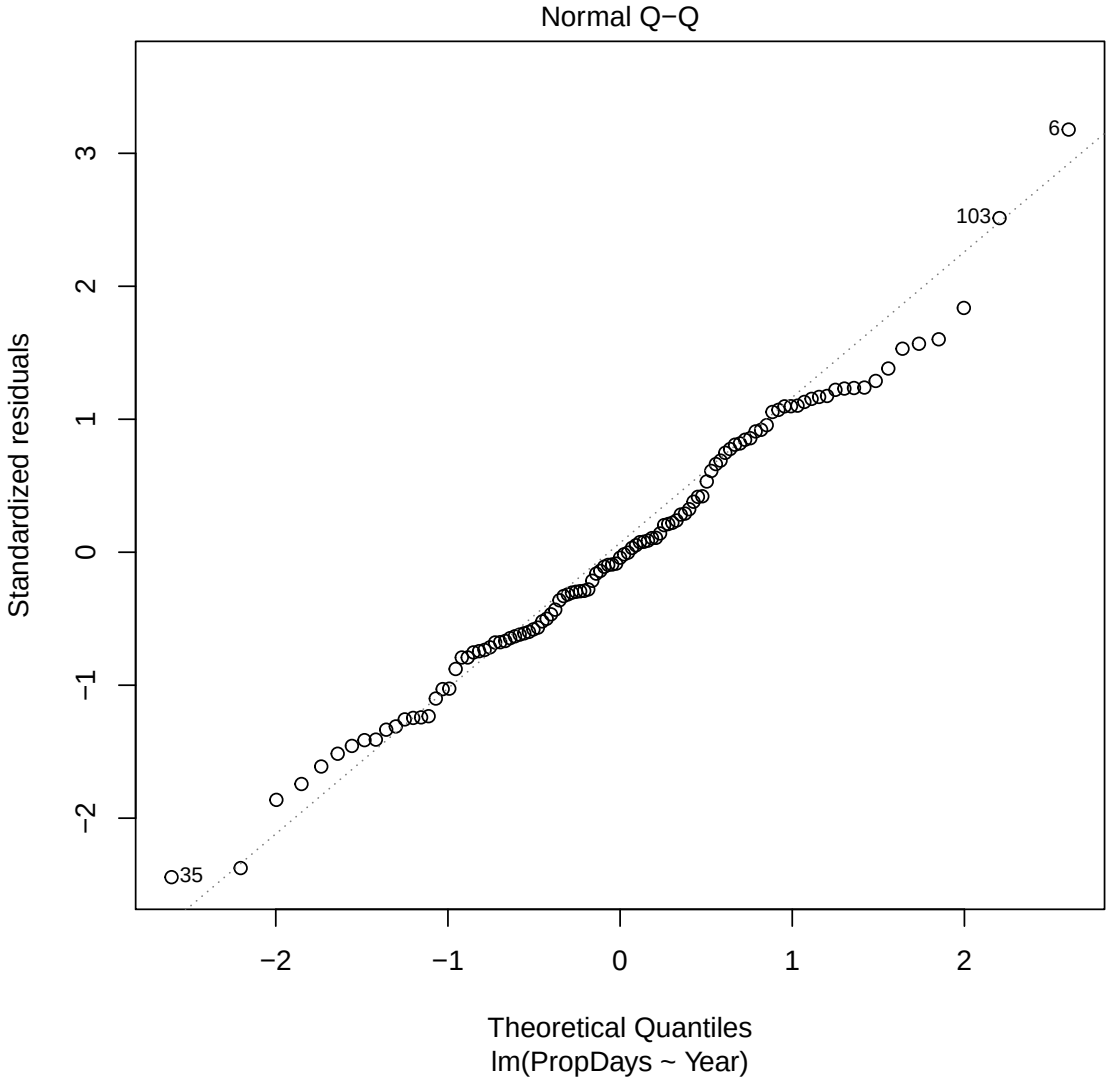

```
perm_pvalue <- (length(which(lm<=coef(lm.Year)[2])) + length(which(lm>=-coef(lm.Year)[2]))
```

H_o: $\beta_{Year} = 0$

H_a: $\beta_{Year} \neq 0$

Based on 10000 permutations, the resulted in a pvalue of 0, from a test statistic of -3×10^{-4}, which is also the estimated slope from a model with the observed data.

```
par(mfrow=c(1,1))
plot(lm.Year, which = 2)
```



The Normal Q-Q plot shows some slight deviations from normality in the tails. If the normality assumption is not met, the pvalues from the F tests for the coefficients in the model output may be incorrect. Because the residuals (and therefore the response) are not perfectly normally distributed, it may be advantageous to consider a test that is not based on the normality assumption, such as the permutation test.

5. *The Sen estimator or, more commonly, Theil-Sen is based on a single median of all the possible pairwise generated slopes. Its standard version is available in the "mblm" (median based linear models) R package developed by Lukasz Komsta. The package description provides more details (`https://cran.r-project.org/web/packages/mblm/mblm.pdf`). Note that with 'mblm', you need to use "repeated=FALSE" to get the Theil-Sen estimator and not the better estimator developed by Siegel. The package has a "summary" function that provides a test based on the nonparametric Wilcox test but it had terrible Type I error rates when I explored it.*

Without further explorations, I would recommend avoiding its use. Fortunately, our permutation approach can be used to develop a test based on the Theil-Sen slope coefficient. First, compare the estimated slope provided by "mblm" to what you found from the linear model and its permutation test. Then develop a permutation test based on the slope coefficient from 'mblm' - note that "mblm" conveniently has the same output structure as "lm". The confidence interval that runs on "mblm" seems to perform well enough to study, so we can make 95% confidence intervals and check whether 0 is in the interval or not as the following code suggests to use it to perform our 5% significance level hypothesis test.

What does he mean by "and its permutation test"? – AM

```
require(mblm)
model1s<-mblm(PropDays~Year,data=Data1,repeated=F)
model1s.coef <- coef(model1s)[2]

year <- data.frame(replicate(10000, shuffle(Data1$Year)))

## Error in FUN(X[[i]], ...):  could not find function "shuffle"

year$pdays <- Data1$PropDays

## Error in year$pdays <- Data1$PropDays:  object 'year' not found

shuffled.frame <- data.frame(pdays = year$pdays)

## Error in data.frame(pdays = year$pdays):  object 'year' not found

mblm.mod <- NULL
mblm.coeff <- numeric(10000)
mblm.mod <- lapply(1:10000, function(i){
  shuffled.frame$year <- year[,i]
  return(mblm(pdays~year, data=shuffled.frame,repeated = F))
})

## Error in FUN(X[[i]], ...):  object 'year' not found

# Excercise for Andrea!
# level of difficulty: 1 (after seeing mblm.mod of course ;)
mblm.coeff <- sapply(mblm.mod, function(x){coef(x)[2]})

pvalue.mblm <- length(c(which(mblm.coeff<=model1s.coef), which(mblm.coeff>=-model1s.coef))


confint(model1s)
```

```
##                      0.025         0.975
## (Intercept)  1.0804343058  1.0931490704
## Year        -0.0003697871 -0.0002404847

CI<-confint(model1s)[2,] #Extract CI and check whether 0 is in interval
(0>CI[1])&(0<CI[2]) #If 0 is in interval, FTR H0

## 0.025
## FALSE

confint(mblm.mod[[1]])[2,]

## Error in UseMethod("vcov"):  no applicable method for 'vcov' applied to an
object of class "NULL"

fn.confint <- function(x){
  confint(x)[2,]
}

out <- mblm.mod[[1]]

dec.mblm <- c(rep(0,10000))

#took about 3-4 hours to run
for(i in 1:10000){
  dec.mblm[i] <- ifelse(confint(mblm.mod[[i]])[2,1]<0 & confint(mblm.mod[[i]])[2,2]>=0, "F
}

## Error in UseMethod("vcov"):  no applicable method for 'vcov' applied to an
object of class "NULL"

cl.mblm.confint <- length(which(as.factor(dec.mblm) == "R"))/length(dec.mblm)
```

`lm()` gave an estimated slope coefficeint of $-3.1172237 \times 10^{-4}$ and `mblm()` gave an estimated slope coefficient of $-3.1612223 \times 10^{-4}$.

Using the same hypotheses from 4, but with the `mblm()` function fitting the model with the median response, the permutation test based on 10,000 iterations led to a pvalue of NaN using the test statistic of the observed slope coefficient ($-3.1612223 \times 10^{-4}$) and the null hypothesis of no relationship between year and temperature. The explicit hypotheses are stated below. There is strong evidence that year had an effect on median temperature in these data. H_o: $\beta_{Year.median} = 0$

H_a: $\beta_{Year.median} \neq 0$

It is estimated that the median temperature in this data set decreased by -3.1612223×10^{-4} F each year with an associated 95% confidence interval of -3.6978709×10^{-4} to -2.404847×10^{-4} F.

However, I would question the validity of the confidence level set using `confint()` on an `mblm()` object. If there was strong evidence of an effect of temperature, at the 95% confidence level, we would expect to "reject" 5% of the time, meaning that approximately 5% of confidence intervals, in the long run, should contain 0. We ran 10,000 simulations and 0% contained 0. Something is wrong.

6. *Use the residual error variance estimate from your linear model for the proportion responses to simulate a series with no trend (constant mean and you can leave it at 0) and normal white noise with that same variance. Use that simulation code to perform a simulation study of the Type I error rate for the parametric t-test for the slope coefficient, the test using the confidence interval from "mblm", and your permutation test (use 500 permutations and do 250 simulations to keep the run time somewhat manageable). Report the simulation-based Type I error rates when using a 5% significance level test for the three procedures with the same sample size as the original data set.*

   - *For the parametric test, the p-value can be extracted from the "lm" model "summary"'s using "summary(model1)coef*$[2,4]$".
   - *It is best and easiest if you do one loop for the simulations and then for each simulated data set in each loop generate the three test results, extracting the p-values that each produces. If you struggle to set this up, please send me an email or stop by with an attempt at your code for some feedback.*
   - *This will be computationally intensive. To avoid needing to re-run results in R-markdown, you can try the "cache=T" option for any of the permutation or simulation code chunks. Or for this section, you can just report the three error rates and comment out the code you used.*

7. *Instead of white noise errors, we might also be interested in Type I error rates when we have autocorrelation present (again with no trend in the true process). Use the results for an AR(1) process variance (derived in class) to calculate the white noise variance needed to generate a process with the same variance as you used for your previous simulation, but when ϕ=0.3 and 0.6. In other words, γ_0 of the AR(1) process needs to match the white noise variance used above and the white noise process driving the AR(1) process needs to be adjusted appropriately.*

   Let ϕ^k = the correlation between observations taken k time points apart (k lags apart), such that for time points 0 units apart, $\phi^0 = 1$.

   In class we derived that the variance at a given time point, $\text{Var}(y_t) = \text{Cov}(y_t,y_t) = \frac{\sigma_e^2}{(1-\phi^2)}$.

   (a) *Show your derivation of the required white noise variances first for $\phi = 0.3$ and $\phi = 0.6$.*

   Take residual standard error and input in here.

   White Noise Variance = σ^2

   $\sigma^2_{0.03} = \frac{}{1-0.03} = \frac{}{0.97} =$