HW 6 KEY

Graded out of 62.5 points, scored out of 65 with a 1 point adjustment

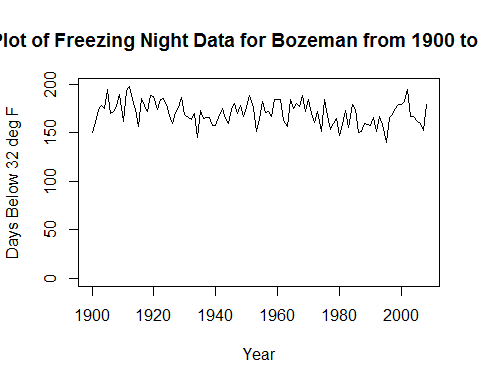
We will explore one series like those you found in the Vincent and Meki's paper, but for Bozeman.

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.

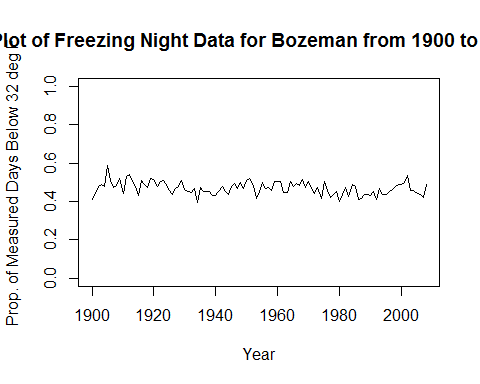
Bozeman<-read.csv("https://dl.dropboxusercontent.com/u/77307195/Bozeman.csv",header=T)  
  
monthsF<-sort(unique(Bozeman$MonthRE))  
countfun<-function(x) c(sum(x<32),sum(!is.na(x)))  
  
 monthcountMINF<-aggregate(Bozeman$TMIN..F.,by=list(Bozeman$MonthRE),FUN=countfun)  
 yearcountMINF<-aggregate(Bozeman$TMIN..F.,by=list(Bozeman$Year),FUN=countfun)  
   
 Data1<-data.frame(Year=yearcountMINF[,1],DaysBelow32=yearcountMINF$x[,1],MeasuredDays=yearcountMINF$x[,2],PropDays=yearcountMINF$x[,1]/yearcountMINF$x[,2])

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

plot(DaysBelow32~Year,data=Data1,type="l",ylim=c(0,max(Data1$DaysBelow32)),ylab="Days Below 32 deg F",main="Plot of Freezing Night Data for Bozeman from 1900 to 2008")



plot(PropDays~Year,data=Data1,type="l",ylim=c(0,1),ylab="Prop. of Measured Days Below 32 deg F",main="Plot of Freezing Night Data for Bozeman from 1900 to 2008")



1. 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).

* There is strong evidence of a linear trend in the proportion of days below freezing (t(107)=-3.17,p-value=0.002).
* For a 1 year increase, we estimate that the mean proportion of days below freezing changes by -0.0003, or for 100 years, the change in the mean proportion is estimated to be -0.03117.
* It is not possible to make causal inferences because the years can not be randomly assigned (so time is not causing changes). The inferences are limited to the years studied because this is not a random sample of years, so inferences pertain to 1900 to 2008. This is also just at a particular location, Bozeman, so inferences are only pertinent to this area.

model1<-lm(PropDays~Year,data=Data1)  
summary(model1)

##   
## Call:  
## lm(formula = PropDays ~ Year, data = Data1)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.078346 -0.021286 -0.001322 0.025965 0.100939   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.078e+00 1.920e-01 5.618 1.55e-07 \*\*\*  
## Year -3.117e-04 9.823e-05 -3.173 0.00197 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.03227 on 107 degrees of freedom  
## Multiple R-squared: 0.08602, Adjusted R-squared: 0.07748   
## F-statistic: 10.07 on 1 and 107 DF, p-value: 0.001968

1. 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?

* The proportion response seems more valid than the count of days since the number of days that were observed in each year varies. You would hate to see a trend toward more cold nights later in the series occur because missing days are less present in the modern era. It separates the rate of days from the amount measured.
* The count of days out of the total possible really is a Binomial response and a logistic regression model should be considered for the analysis here. Because the number of days (trials) is really large (usually over 300) the normal approximation is reasonable. But if the true proportion changed a lot then the variance would as well and the typical linear model would struggle. We could also consider an arc-sin-square-root or logit transformation here of the proportions to possibly stabilize variance and make the responses more normally distributed.

1. 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?

* The test statistic used is the slope coefficient which was estimated to be -0.0003 here. You could also use the t-statistic for the slope and would get the same answer. The p-value is 0.0014 in my 10,000 permutations.
* The permutation test would in theory allow us to relax the normality assumption. This is potentially helpful because the response is a proportion and that might not follow a normal distribution. But with proportions the distribution might be skewed because of the boundaries at 0 and 1 and in a Binomial setting the variance changes as a function of the mean. These two things mean that we might violate the "equal shape and variability" condition of the permutation test and so get no real benefit from using it here.
* With a large sample size, the differences between the parametric t-test and the permutation test are minimal as is seen here. But it does allow us to develop a test for something that we don't know its distribution like in mblm where the distribution of the slope (specifically its SE) may not be easy to derive.

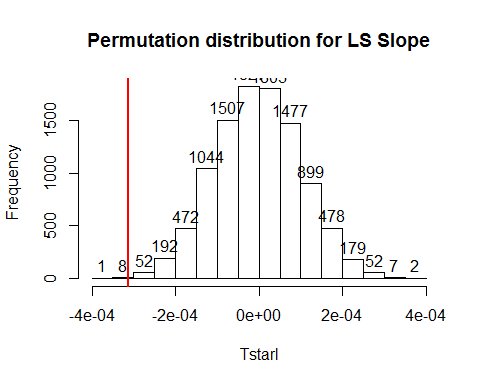
require(mosaic)  
require(mblm)  
set.seed(1356)  
 B<- 10000  
 Tstarl<-Tstars<-matrix(NA,nrow=B)  
 for (b in (1:B)){  
 Data1$Yearshuffle<-shuffle(Data1$Year)  
 Tstarl[b]<-lm(PropDays~Yearshuffle,data=Data1)$coefficients[2]  
 Tstars[b]<-mblm(PropDays~Yearshuffle,data=Data1,repeated=F)$coefficients[2]  
 }  
 Tobsl<-lm(PropDays~Year,data=Data1)$coefficients[2]  
 Tobsl

## Year   
## -0.0003117224

Tobss<-mblm(PropDays~Year,data=Data1,repeated=F)$coefficients[2]  
 Tobss

## Year   
## -0.0003161222

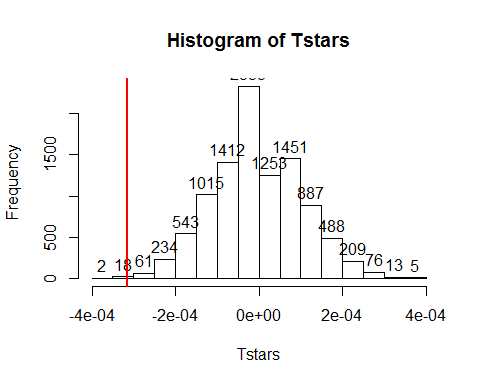
hist(Tstarl,labels=T,main="Permutation distribution for LS Slope")  
 abline(v=Tobsl,lwd=2,col="red")



pdata(abs(Tstarl),abs(Tobsl),lower.tail=F)

## Year   
## 0.0014

hist(Tstars,labels=T)  
 abline(v=Tobss,lwd=2,col="red")



pdata(abs(Tstars),abs(Tobss),lower.tail=F)

## Year   
## 0.0024

1. 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.

* The Theil-Sen slope is estimated to be -0.000316. The linear model slope is -0.000317 so there is little difference here. Some would use that close match as an argument that there are no extreme outliers present.
* The code for my permutation test is provided above in the same loop as for the linear model slope permutation test. The p-value is pretty much the same as the result found for the permutation test from the linear model with it being 0.0024 in my 10,000 permutations.
* The 95% confidence interval goes from -0.0003697871 to -0.0002404847. So we would reject the null hypothesis here and have a consistent result with the other approaches. But can we really trust this confidence interval to have the correct coverage rate? If a test based on it works at the 5% level then it is also going to have 95% coverage.
* If you dig into the methods, the confidence interval is based on the middle 95% of the pairwise slopes. I am still trying to decide if this is a method that provides a well-calibrated interval but I don't think it is...

require(mblm)  
model1s<-mblm(PropDays~Year,data=Data1,repeated=F)  
summary(model1s)

##   
## Call:  
## mblm(formula = PropDays ~ Year, dataframe = Data1, repeated = F)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.07703 -0.01971 0.00000 0.02734 0.10213   
##   
## Coefficients:  
## Estimate MAD V value Pr(>|V|)   
## (Intercept) 1.0856691 0.0331885 5995 <2e-16 \*\*\*  
## Year -0.0003161 0.0014556 6858654 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.0323 on 107 degrees of freedom

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

1. 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.
* The residual SE from above is 0.03227.

set.seed(93567)  
x<-Data1$Year  
  
Sims<-1000  
B<- 999  
Pval\_t<-CIresult\_s<-Pval\_p<-matrix(NA,nrow=Sims)  
for (k in (1:Sims)){  
ysim<-rnorm(n=109,0,0.03227)   
Pval\_t[k]<-summary(lm(ysim~x))$coef[2,4]  
model1s<-mblm(ysim~x,repeated=F)  
CI<-confint(model1s)[2,] #Extract CI and check whether 0 is in interval  
CIresult\_s[k]<-!((0>CI[1])&(0<CI[2])) #If 0 is not in interval, Reject H0  
  
 Tstar<-matrix(NA,nrow=B)  
 for (b in (1:B)){  
 Tstar[b]<-lm(ysim~shuffle(x))$coefficients[2]  
 }  
 Tobs<-lm(ysim~x)$coefficients[2]  
 Pval\_p[k]<-(pdata(abs(Tstar),abs(Tobs),lower.tail=F)\*B+1)/(B+1)  
 }  
  
ResultsSummary<-data.frame(TtestError=mean(Pval\_t<0.05),SentestError=mean(CIresult\_s),PermtestError=mean(Pval\_p<0.05))  
print(ResultsSummary)

## TtestError SentestError PermtestError  
## 1 0.047 0.661 0.046

* This suggests that the two linear-model based tests perform about as expected with close to 5% type I error rates with 5% tests. The Sen-based confidence interval rejects the null hypothesis around 60 or 70% of the time when the assumptions of the linear model are met. This means the real coverage rate for a 95% interval would be around 30 or 40%.
* Some got massively different answers here and I checked your code as best I could for errors. I am hoping it was just an error in transcribing simulation results because I can't explain the differences since the code purports to be the same.

1. 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 =0.3 and 0.6. In other words, 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.

* Show your derivation of the required white noise variances first for and .
* To simulate the process we can use this value in the arima.sim function in something like arima.sim(n=2000,list(ar=c(0.3)),sd=5) where n=2000 provides 2000 simulated observations, model=list(ar=c(0.3)) determines that we are using an AR(1) process with parameter of of 0.3, and sd=5 controls the SD of the normal white noise used to build the AR(1) process (this is *not* the variance of the AR(1) process). Check that you get about your expected results using something like:
* The variance of an AR(1) process is so the that we need to have the AR(1) process have a variance of is .
* For , the white noise procces needs a variance of 0.0009476311
* For , the white noise procces needs a variance of 0.0006664659

ar1sim<-arima.sim(n=2000,model=list(ar=c(0.3)),sd=sqrt(0.0009476311))  
var(ar1sim)

## [1] 0.001045496

ar1sim<-arima.sim(n=2000,model=list(ar=c(0.6)),sd=sqrt(0.0006664659))  
var(ar1sim)

## [1] 0.001004476

1. Repeat your simulation study of the parametric, permutation, and Theil-Sen linear trend test based on the CI. Report the estimated Type I error rates in the presence of AR(1) correlations with a parameter of 0.6 based on your work in the previous question for simulating the response time series.

set.seed(93567)  
x<-Data1$Year  
  
Sims<-1000  
B<- 999  
Pval\_t<-CIresult\_s<-Pval\_p<-matrix(NA,nrow=Sims)  
for (k in (1:Sims)){  
ysim<-arima.sim(n=109,model=list(ar=c(0.6)),sd=sqrt(0.0006664659))   
Pval\_t[k]<-summary(lm(ysim~x))$coef[2,4]  
model1s<-mblm(ysim~x,repeated=F)  
CI<-confint(model1s)[2,] #Extract CI and check whether 0 is in interval  
CIresult\_s[k]<-!((0>CI[1])&(0<CI[2])) #If 0 is not in interval, Reject H0  
  
 Tstar<-matrix(NA,nrow=B)  
 for (b in (1:B)){  
 Tstar[b]<-lm(ysim~shuffle(x))$coefficients[2]  
 }  
 Tobs<-lm(ysim~x)$coefficients[2]  
 Pval\_p[k]<-(pdata(abs(Tstar),abs(Tobs),lower.tail=F)\*B+1)/(B+1)  
 }  
  
ResultsSummary<-data.frame(TtestError=mean(Pval\_t<0.05),SentestError=mean(CIresult\_s),PermtestError=mean(Pval\_p<0.05))  
print(ResultsSummary)

## TtestError SentestError PermtestError  
## 1 0.333 0.834 0.334

* With a moderate or high level of positive autocorrelation present we would expect to have some issues with our linear model procedures since they assume independence. It is unclear what to expect with the Sen-CI but its original results and the fact that it doesn't account for autocorrelation would make me concerned.
* The results show higher Type I error rates when positive autocorrelation is present with the parametric and permutation tests from the linear both having inflated error rates. The Sen-CI gets even worse but that could just be from sampling variability.

1. The Zhang method you read about is also available in the zyp package but it only provides confidence intervals and I am not completely convinced by their discussion of the intervals provided without more exploration. But you can get estimates from zyp.sen and confidence intervals using confint.zyp on the results from zyp.sen. The confint function can also be applied to mblm results. Find and compare the two confidence intervals for the Sen-estimators for the proportion response time series. No simulation study here - just complete the analysis.

require(zyp)  
z1<-zyp.sen(PropDays~Year,data=Data1)  
confint.zyp(z1)

## 0.025 0.975  
## Intercept 1.0241223706 1.150026980  
## Year -0.0005073567 -0.000109903

* The 95% CI from the Zhang method is from -0.0005 to -0.00011. The mblm based method is from -0.00037 to -0.00024. The Zhang method's CI is about 3 times wider. Since the mblm method performed poorly we would expect this method to be better calibrated but would still need to explore it further to see if it works better or not. It certainly can't work worse.
* In digging further into this function, the code I had you run just provides a better version of the slope estimate than what is provided from mblm but is still just using the regular Theil-Sen method and there is no adjustment for autocorrelation incorporated. It is **not** using the Zhang method. To use the Zhang method you have to use their very odd functions: zyp.trend.dataframe(indat, metadata.cols, method=c("yuepilon", "zhang"),conf.intervals=TRUE, preserve.range.for.sig.test=TRUE) and zyp.trend.csv(filename, output.filename, metadata.cols, method=c("yuepilon", "zhang"), conf.intervals=TRUE,csv.header=TRUE, preserve.range.for.sig.test=TRUE). I'll leave that for further explorations if you want to figure this out.

1. Make a plot of the original proportion response time series with the parametric linear, Theil-Sen, and Zhang methods/models on the same plot. You may want to use plot(y~x,type="l") and then add lines to the plot.

plot(PropDays~Year,data=Data1,type="l")  
abline(model1)  
abline(model1s,col="blue",lwd=2,lty=2)  
abline(z1$coefficients,col="red",lwd=3,lty=3)

