How Will SLR Increase Risk of Tidal Flooding in Providence, RI?

Curtis C. Bohlen

April 4, 2021

Table of Contents

# Introduction

In this notebook, we develop estimates of future tidal flooding risk for Providence, RI under one foot, two foot, and three foot sea level rise (SLR) scenarios. Our goal is to be able to say that analysis or simulation suggests an X% increase in frequency of flooding with a Y foot increase in SLR.

We conduct two different analyses to get at this question.

Our first method follows a process we were introduced to by the work of Peter Slovinsky, and the Maine Geological Survey. MGS has estimated future flooding risk at a variety of locations around Maine by adding a fixed SLR value to the historical record, and showing how frequent flooding would have been if base sea levels had been that much higher in the past. This provides a ready, and readily understood estimate of impact of SLR on frequency of flooding.

Details of their method and a data viewer for different locations in Maine are available at the [Maine Geological Survey sea level rise data viewer](https://mgs-collect.site/slr_ticker/slr_dashboard.html).

We repeat (a slightly modified version of) their analysis for selected SLR scenarios, and estimate percentage change in flooding under SLR.

Our second method uses simulation to estimate future flood frequencies.  
We then go on to simulate flooding histories based on the 19 year tidal epoch, and examining predicted flood frequencies using three different models under one foot, two foot, and three foot SLR scenarios.

For technical reasons, develop the methods looking at the official 19 year tidal epoch (1983 through 2001) on which current tidal predictions are based. We then reprise those analyses looking at current conditions, and forecast future conditions under an additional one foot, two foot, and three feet of SLR.

# Import Libraries

library(tidyverse)  
#> -- Attaching packages --------------------------------------- tidyverse 1.3.0 --  
#> v ggplot2 3.3.3 v purrr 0.3.4  
#> v tibble 3.0.5 v dplyr 1.0.3  
#> v tidyr 1.1.2 v stringr 1.4.0  
#> v readr 1.4.0 v forcats 0.5.0  
#> -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
#> x dplyr::filter() masks stats::filter()  
#> x dplyr::lag() masks stats::lag()  
library(readr)  
  
library(data.table) # for `fread()` to read large files  
#>   
#> Attaching package: 'data.table'  
#> The following objects are masked from 'package:dplyr':  
#>   
#> between, first, last  
#> The following object is masked from 'package:purrr':  
#>   
#> transpose  
  
library(moments) # for skewness and kurtosis; we could calculate, but why?  
library(forecast) # for `auto.arima()`  
#> Registered S3 method overwritten by 'quantmod':  
#> method from  
#> as.zoo.data.frame zoo  
library(lubridate) # to facilitate date and time imports  
#>   
#> Attaching package: 'lubridate'  
#> The following objects are masked from 'package:data.table':  
#>   
#> hour, isoweek, mday, minute, month, quarter, second, wday, week,  
#> yday, year  
#> The following objects are masked from 'package:base':  
#>   
#> date, intersect, setdiff, union  
  
library(CBEPgraphics) # to standardize graphic styles and colors  
load\_cbep\_fonts()   
theme\_set(theme\_cbep())

# Define Flood Level

One might want to complete this analysis with any of a variety of different defined flood elevations. To simplify rerunning the analysis, we define the flood elevation here. We chose the “HAT” tide elevation defined for Providence, RI, as expressed in meters above MLLW. (All the following analysis is conducted in those units). “HAT” stands for “highest astronomical tide”, and it refers to th highest tide expected during the nineteen year tidal epoch.

FLOOD\_ELEVATION = 1.987

# Import Data

Our primary source data is hourly data on observed and predicted water levels at the Providence tide station (Station 8454000). We accessed these data using python scripts to download and assemble data from the NOAA Tides and Currents API. The scripts and the data are provided in the Original\_Data" folder.

We downloaded the entire record of available data from the Providence gauge, even though we only use a portion of it in this analysis.

sibfldnm <- 'Original\_Data'  
parent <- dirname(getwd())  
sibling <- file.path(parent,sibfldnm)

The file of observed water levels is a very large CSV file. Loading the file with read\_csv() was crashing R. The fread() function from the data.table package handles large files more gracefully, so we used it for the file import. We then converted the data.table to a tibble – the tidyverse version of a data frame – to facilitate later steps using tidyverse idioms.

fn <- 'providence\_tides\_hourly.csv'  
fpath <- file.path(sibling, fn)  
  
# observed\_data <- read\_csv(fpath)  
observed\_data <- fread(fpath)  
  
observed\_data <- as\_tibble(observed\_data) %>%  
 rename(MLLW = `Water Level`,  
 theDate =`Date`) %>%  
 filter(! is.na(MLLW)) %>%  
 mutate(DateTime = ymd\_hm(DateTime),  
 theDate = as.Date(theDate),  
 Year = as.numeric(format(theDate, '%Y')),  
 MLLW\_ft = MLLW \* 3.28084,  
 Exceeds = MLLW > FLOOD\_ELEVATION)

observed\_data <- observed\_data %>%  
 mutate(Hour = as.numeric(format(DateTime, '%H')),  
 Month = as.numeric(format(theDate, '%m')),  
 Day = as.numeric(format(theDate, '%d')),  
 Year = as.numeric(format(theDate, '%Y'))) %>%  
 select(-Sigma)

For some reason, loading th predictions file, which is nearly as large as the previous file, did not crash R when loaded with read\_csv().

fn <- 'providence\_tides\_hourly\_predicts.csv'  
fpath <- file.path(sibling, fn)  
  
predict\_data <- read\_csv(fpath, col\_types = cols(Time = col\_time('%H:%M'))) %>%  
 rename(theDate =`Date`) %>%  
 mutate(Hour = as.numeric(format(DateTime, '%H')),  
 Month = as.numeric(format(theDate, '%m')),  
 Day = as.numeric(format(theDate, '%d')),  
 Year = as.numeric(format(theDate, '%Y'))) %>%  
 select(-DateTime, -theDate, -Time)

## Combine Data

The number of predicted and observed values are not the same. Presumably this reflects periods when the gauge was not in operation (but prediction was still possible based on harmonic constituents). That means we need to make sure we merge data appropriately, lining up dates and times correctly. We only want to work with data when we have actual observations of water level. We use left\_merge(), to add the predictions to the observed water level data. We merge by Year, Month, Day, and Hour. Alternatively, we could have merged on DateTime, but working with POSIXct times in R can be tricky.

combined <- observed\_data %>%  
 inner\_join(predict\_data,by = c("Year", "Month", "Day", "Hour"))  
  
## Calculate Deviations Between Predicted and Observed  
combined <- combined %>%  
 mutate(deviation = MLLW - Prediction)

rm(observed\_data)

# The Tidal Epoch

Tidal predictions are defined in terms of a specific 19 year long “Tidal Epoch.” The astronomical alignments of sun, moon, and earth repeat (at least closely enough for tidal prediction) every nineteen years, and tides are predicted based on astronomical processes. Tidal “Predictions” are based on a complex periodic function that is parameterized by “harmonic constituents”, themselves calculated based on observed tidal elevations over the 19 year Tidal Epoch.

For our purposes, the key insight is that tidal predictions are effectively a nineteen year-long periodic function, and thus do not take into account changes in sea level. Consequently, the deviations from tidal predictions we just calculated are not stationary. They tend to be more negative early in the tidal epoch, and more positive later.

However, during the Tidal Epoch, the average error of prediction should be zero (or very close to zero). Our best understanding of the distribution of deviations from (astronomical) predicted tide elevations, therefore, would come from looking at deviations during that tidal epoch, when deviations due to changing sea level are minimized.

That is why we base simulations of future flood risk on the tidal epoch.

According to the NOAA webpage for Providence, the current tidal epoch is 1983-2001. (The epoch is provided on the datums sub-page.) That means the current tidal predictions were based on data collected during a nearly twenty year period that ended about twenty years ago. To the extent that sea levels have been rising since, those predictions are likely to be low compared to observed sea levels in 2021. They are, in fact, about thirty yeas out of date. (We use thirty years because the midpoint of the tidal epoch was about thirty years ago.)

NOAA estimates average historical sea level trend in Providence at about 2.5 mm per year. that suggests current observed tidal elevations should be about above predicted levels – probably too small to be noticed, but it may affect frequency of flooding.

This raises a question about what constitutes a “fair” comparison between present-day and future flood frequencies. We do not want to compare future flood frequencies with frequencies from the the period of the tidal epoch, because we know we have higher flood frequencies today than a couple of decades ago. The “best” comparison should compare actual recent flood frequencies with “predicted” recent flood frequencies, and with “predicted” flood frequencies under different sea level rise scenarios. We follow that practice, below.

# Data for the tidal epoch  
epoch <- combined %>%  
 filter(Year> 1982 & Year < 2002)  
  
# And data for the most recent 19 years  
mock\_epoch <- combined %>%  
 filter(Year> 2002 & Year < 2021)

# How Many Flood Events Have Happened per Year?

## Function for Counting Days with Flooding

flood\_counts <- function(dat, dts, observed\_wl, flood = FLOOD\_ELEVATION) {  
 # dat is a dataframe  
 # dts is a data column in that data frame of identifiers by date / day  
 # observed\_wl is a data column of hourly observed water levels at  
 # flood is the selected level above which you declare an event to be a flood  
   
 # We quote data variables, and look them up by name  
 # Caution: there is no error checking. if things are not working it may be   
 # because the columns do not exist.  
   
 dts <- as.character(ensym(dts))  
 obs\_wl <- as.character(ensym(observed\_wl))  
   
 #create a dataframe, for convenience  
 df <- tibble(theDate = dat[[dts]], obs\_wl = dat[[obs\_wl]])  
   
 response <- df %>%   
 group\_by(theDate) %>%  
 summarize(exceeded = any(obs\_wl > flood),  
 .groups = 'drop') %>%  
 summarize(days = sum(! is.na(exceeded)),  
 floods = sum(exceeded, na.rm = TRUE),  
 floods\_p\_dy = floods/days,  
 floods\_p\_yr = 365.25 \* floods/days,  
 .groups = 'drop')  
 return(response)  
}

### Tidal Epoch

As a check on our methods, it’s nice to know how many flood events actually occurred during the official tidal epoch.

flood\_counts(epoch, theDate, MLLW)  
#> # A tibble: 1 x 4  
#> days floods floods\_p\_dy floods\_p\_yr  
#> <int> <int> <dbl> <dbl>  
#> 1 6917 64 0.00925 3.38

### Most Recent 19 Years

flood\_counts(mock\_epoch, theDate, MLLW)  
#> # A tibble: 1 x 4  
#> days floods floods\_p\_dy floods\_p\_yr  
#> <int> <int> <dbl> <dbl>  
#> 1 6524 152 0.0233 8.51

That suggests during the tidal epoch, we saw roughly one flood event every 100 days, or about three and a third floods a year. More recently, we’ve seen the rate of flooding more than double, even though sea level only rose a few inches over that period of time. That’s remarkable.

## Function for Mean and SD of Days of Flooding Per Year

Using slightly different methods, we get a slightly different numbers (+/- 2%), but more importantly, we can calculate variability in mean number of flood events each year. The disadvantage is that it is not obvious whether we should adjust for the few days a year when data was not being collected.

The prior method averages floods across the whole period of record and scales to the length of the average year. Here we count the number of floods per year, and calculate averages and standard deviations.

flood\_means <- function(dat, dts, observed\_wl, flood = FLOOD\_ELEVATION) {  
 # dat is a dataframe  
 # dts is a data column in that data frame dates from which we can extract years  
 # observed\_wl is a data column of hourly observed water levels at  
 # flood is the selected level above which you declare an event to be a flood  
   
 # We quote data variables, and look them up by name  
 # Caution: there is no error checking. if things are not working it may be   
 # because the columns do not exist.  
   
 dts <- as.character(ensym(dts))  
 obs\_wl <- as.character(ensym(observed\_wl))  
   
 #create a dataframe, for convenience  
 df <- tibble(theDate = dat[[dts]], obs\_wl = dat[[obs\_wl]]) %>%  
 mutate(Year = as.numeric(format(theDate, format = '%Y')))  
   
   
results <- df %>%  
 group\_by(theDate) %>%  
 summarize(Year = first(Year),  
 exceeded = any(obs\_wl > flood),  
 n = sum(! is.na(exceeded)), # Count days with ANY data...  
 .groups = 'drop') %>%  
 group\_by(Year) %>%  
 summarize(days = sum(n),  
 floods = sum(exceeded),  
 floods\_p\_d = floods / days,  
 floods\_p\_yr = floods\_p\_d \*   
 if\_else(Year %% 4 == 0 & (! Year %% 100 == 0), 365, 366),  
 .groups = 'drop') %>%  
 select(-floods\_p\_d) %>%  
 summarize(across(contains('floods'), c(mean = mean, sd = sd), na.rm = TRUE))   
  
 return(results)  
}

### Tidal Epoch

flood\_means(epoch, theDate, MLLW)  
#> # A tibble: 1 x 4  
#> floods\_mean floods\_sd floods\_p\_yr\_mean floods\_p\_yr\_sd  
#> <dbl> <dbl> <dbl> <dbl>  
#> 1 3.36 2.41 3.38 2.43

So, over the official tidal epoch, we saw on the order of three and a third flood events a year. ideally, our simulations should return a similar frequency.

### Most Recent 19 Years

Actual tidal levels today are several centimeters higher than during the tidal epoch. With average SLR on the order of 3 mm per year, observed elevations today should be on the order of .

flood\_means(mock\_epoch, theDate, MLLW)  
#> # A tibble: 1 x 4  
#> floods\_mean floods\_sd floods\_p\_yr\_mean floods\_p\_yr\_sd  
#> <dbl> <dbl> <dbl> <dbl>  
#> 1 8.42 3.94 8.52 4.03

So, just the SLR we have experienced over the past two decades has increased the number of expected flooding events by more than a factor of two. Paradoxically, that may make the relative impact of additional SLR look slightly smaller.

### Most Recent 10 Years

combined %>%  
 filter(Year > 2010) %>%  
 flood\_means(theDate, MLLW)  
#> # A tibble: 1 x 4  
#> floods\_mean floods\_sd floods\_p\_yr\_mean floods\_p\_yr\_sd  
#> <dbl> <dbl> <dbl> <dbl>  
#> 1 10.1 3.43 10.3 3.54

So the rate of flooding continues to increase…..

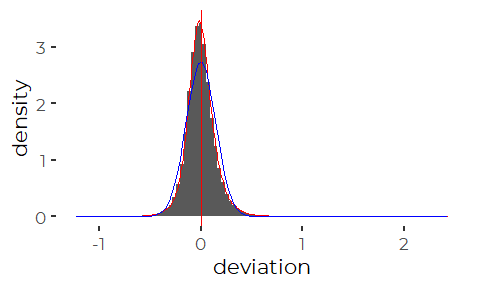
Note that this calculation leads to a higher estimate of recent flood frequency that does fitting a generalized linear model to historic flood frequencies. See the “Tidal\_Flooding\_Events.Rmd” notebook in the “Graphics” folder in the [GitHub repository](https://github.com/ccb60/Providence_SLR).

# Evaluation of Properties of Tidal Data for Simulation

## Distribution of Deviations

We plot a histogram of deviations, and overlay it with a kernel density estimator (in red) and a normal distribution with the same mean and variance (in blue).

ggplot() +  
 geom\_histogram(aes(x = deviation, y = ..density..),  
 bins = 100, data = epoch) +  
 geom\_density(aes(x = deviation), color = 'red', data = epoch) +  
 stat\_function(fun = dnorm,  
 args = list(mean = mean(epoch$deviation, na.rm = TRUE),  
 sd = sd(epoch$deviation, na.rm = TRUE)),  
 color = 'blue') +  
 geom\_vline (xintercept = 0, color = 'red')



Deviations are bell shaped, with mean close to zero, but the match with a normal distribution is not very good. The distribution is more peaked, with heavier tails than a normal distribution would suggest. Although it is a bit difficult to see, the distribution is also moderately skewed.

We are interested in floods, which correspond to events in the upper extremes of this distribution. The effects of skewness and kurtosis on frequency of extreme values reinforce each other. There were more extreme values during the tidal epoch than would have been predicted based on a normal distribution with a similar mean and variance.

## Temporal Autocorrelation

There is relatively high serial autocorrelation in deviations from tidal predictions. That is not much of a surprise. If tides are higher than expected now, it’s likely they will be in an hour too.  
Autocorrelation remains present and positive over a period of days (not shown). There is also some quasi-periodicity, centered on what may be tidal frequencies (12/13 or 24/25 hours), but it is relatively weak at Providence.

A simulation that does not address autocorrelation will probably not be adequate.

## Correlations Between Predictions and Deviations

There is no evidence in the historic data that there are correlations between deviations and the predicted tide.

# Estimates of Future Flood Frequency Using the MGS Analysis

The simplest way to evaluate future flooding risk is simply to take the existing tidal record and raise all elevations by a fixed amount, and count up how many flood events that suggests you might see in the future.

Pete Slovinsky, of Maine Geological Survey, has conducted this kind of analysis to highlight future flood risks. MGS added fixed estimates of SLR to the historic record, and counted hours of time with coastal flooding, defined as tidal elevation exceeding the HAT elevation.

We take a similar approach to identify coastal flooding under one foot, two foot, and three foot SLR scenarios. But instead of counting hours of flood, we count **days** in which flooding occurs. The idea is that most people are not going to worry whether that flood lasted one or two hours, but they are going to be interested in how many days their property floods.

A weakness of this method is that it provides no robust way to evaluate uncertainty of forecasts.

This methods uses relatively simple calculations that could readily be carried out in a spreadsheet program. In R, we use the Tidyverse to do the necessary math. We build a simple function to return expected number of future flood events under SLR.

## Function for SLR Scenarios

This function counts the number of days with flood events that would have occurred over a period of years under various sea level rise scenarios. It uses the result to estimate annual mean and standard deviation of rate of flooding.

The function is intended for use with data from an period of at least several years, with more or less complete records from each year. (The original use case was based on a 19 year period associated with the tidal epoch).

f\_flood\_means <- function(dat, dts, observed\_wl, slr = 0, flood = FLOOD\_ELEVATION) {  
 # dat is a dataframe  
 # dts is a data column in that data frame of dates  
 # observed\_wl is a data column of hourly observed water levels at  
 # slr is a a single value or vector of increases in elevation ("Scenarios")  
 # flood is the selected level above which you declare an event to be a flood  
   
 # We quote data variables, and look them up by name  
 # Caution: there is no error checking. if things are not working it may be   
 # because the columns do not exist.  
   
 dts <- as.character(ensym(dts))  
 obs\_wl <- as.character(ensym(observed\_wl))  
   
 #create a dataframe  
 df <- tibble(theDate = dat[[dts]], obs\_wl = dat[[obs\_wl]]) %>%  
 mutate(Year = as.numeric(format(theDate, format = '%Y')))  
  
 if (missing(slr) || (length(slr) == 1 && is.na(slr[[1]]))) {  
 # Without slr scenarios, we fill in with the null SLR case  
 df <- df %>%  
 mutate(slr\_0 = obs\_wl > flood)  
 }  
 else {  
 # we work through each slr scenario and calculate related data  
 for (n in seq\_along(slr)) {  
 df[[paste0('slr\_', n)]] <- df$obs\_wl + slr[[n]]  
 df[[paste0('exceeds\_', n)]] <- df[[paste0('slr\_', n)]] > flood  
 }  
 }  
   
 # Now we run through the remaining calculation steps.  
 result <- df %>%  
 select(-contains('slr')) %>%  
 # Group by day  
 group\_by(theDate) %>%  
 # Determine if a flood event occurs that day  
 summarize(Year = first(Year),  
 across(contains('exceeds'), any, na.rm = TRUE),  
 .groups = 'drop') %>%  
   
 # Group by Year  
 group\_by(Year) %>%  
 # Add up the number of days with at least one hour of flooding  
 summarize(Year = first(Year),  
 days = n(),  
 across(contains('exceeds'), sum),  
 .groups = 'drop') %>%  
 rename\_with( ~ sub('exceeds', 'floods', .x)) %>%  
 # Calculate a value rescaled by the number of days of data  
 mutate(across(contains('floods'), ~.x \* 365.25 / days,   
 .names = "{.col}\_p\_yr")) %>%  
   
 # Finally, calculate mean and standard deviations of yearly totals.  
 summarize(across(contains('floods'), c(mean = mean, sd = sd), na.rm = TRUE,  
 .names = "{.col}\_{.fn}"))  
  
 # We calculated everything in a tibble, but we want to output  
 # an array. We first make a vector.  
 result <- unname(unlist(result))   
   
 # That means we need to create row and column labels  
 num\_scenarios <- length(result) / 4  
 collabs <- c('Mean', 'SD')  
 rowlabs <- c(paste0('floods\_', 0:(num\_scenarios - 1)),   
 paste0('floods\_p\_yr', 0:(num\_scenarios - 1)))  
   
 # Then we reshape the vector by specifying its dimensions  
 dim(result) <- c(2, length(result)/ 2)  
 # And transpose it, so we end up with two columns, not two rows.  
 result = t(result)  
   
 ## Apply the names  
 rownames(result) <- rowlabs  
 colnames(result) <- collabs  
   
 # Add an attribute to retains the SLR values used.  
 attr(result, 'slr') <- slr  
 return(result)  
}

### Scenarios Based on the Tidal Epoch

We generate future frequency of flooding events estimates for one foot, two foot and three foot SLR scenarios, based on the tidal epoch.  
(The decimals are those SLR values in meters).

mgs\_est <- f\_flood\_means(epoch, theDate, MLLW, c(0, 0.3048, 0.6096, 0.9144))  
mgs\_est  
#> Mean SD  
#> floods\_0 3.368421 2.476793  
#> floods\_1 57.421053 19.483086  
#> floods\_2 231.052632 35.114849  
#> floods\_3 354.684211 6.807289  
#> floods\_p\_yr0 3.384380 2.498327  
#> floods\_p\_yr1 57.679943 19.881355  
#> floods\_p\_yr2 231.907452 35.869393  
#> floods\_p\_yr3 355.852906 5.950846  
#> attr(,"slr")  
#> [1] 0.0000 0.3048 0.6096 0.9144

That suggests a change in flooding frequency on the order of 17.05 with one foot of SLR.

Notice the high standard deviations, especially for moderate sea level rise scenarios. Those standard deviations reflect annual variation in sea levels between years, not modeling error. Yet it is not clear how we should interpret those standard deviations, as past years are not really a random sample of possible future sea levels, and we know the distribution of sea levels is not normally distributed.

### Scenarios Based on Recent Conditions

We generate future frequency of flooding events estimates for zero foot, one foot, two foot and three foot SLR scenarios, based (roughly) on recent conditions.

The tidal forecast is based on the tidal epoch, as that ensures the tidal deviations have mean zero (important in the next analysis). That means to estimate current flood frequencies, we need to add an estimated correction to take us from the tidal epoch to recent times. We used an estimate of about 75 mm of SLR over the past 30 years, based on the NOAA estimate of about 2.5 mm of SLR a year at Providence.

The function as drafted automatically reports the flooding frequency during the tidal epoch as the first value, so we add four more “scenarios”.

mgs\_est\_modern <- f\_flood\_means(epoch, theDate, MLLW,  
 c(0.075, 0.3048 + 0.075, 0.6096 + 0.075, 0.9144 + 0.075))  
mgs\_est\_modern  
#> Mean SD  
#> floods\_0 7.157895 4.621966  
#> floods\_1 89.947368 26.270968  
#> floods\_2 280.631579 30.775550  
#> floods\_3 359.526316 5.450468  
#> floods\_p\_yr0 7.193990 4.674937  
#> floods\_p\_yr1 90.322829 26.756018  
#> floods\_p\_yr2 281.630551 31.485638  
#> floods\_p\_yr3 360.704560 3.709245  
#> attr(,"slr")  
#> [1] 0.0750 0.3798 0.6846 0.9894

Notice that the forecast we get, of about 7.2 +/- 4.5 flood events per year under “current” conditions, is some what lower than has actually been observed, which was around 8.4 +/- 3.9 floods per year.

flood\_means(mock\_epoch, theDate, MLLW)  
#> # A tibble: 1 x 4  
#> floods\_mean floods\_sd floods\_p\_yr\_mean floods\_p\_yr\_sd  
#> <dbl> <dbl> <dbl> <dbl>  
#> 1 8.42 3.94 8.52 4.03

This analysis suggests a change in flooding frequency on the order of 4.02 with one foot of SLR, based (roughly) on current sea levels. That ratio is slightly lower than we saw looking at changes in flood frequency building from the low base of the tidal epoch. That is likely because the number of flood events during the tidal epoch was so low.

# Simulation Models

## Model Structure

We developed simulation models that estimate future flooding by adding simulated deviations to predicted tides. The primary advantage of this approach over the MGS analysis is that it offers the ability to estimate uncertainty (although not bias) by simulating future flood events many times, and examining the distribution of future flooding.

The simulation model we present here does the following: 1. It starts with predicted tides from the official tidal epoch (in meters). (As currently drafted, the simulation runs on the entire 19 year record.)

1. Adds (or not) a sea level rise value to those predicted tides.
2. Adds a random deviation from predicted tidal elevation, where the random values are drawn from a time series that has a similar autocorrelation structure to the historic deviations.
3. Counts up the number of days (over the simulated tidal epoch of 19 years) where the sum of prediction + SLR + deviation exceeds the flood threshold.
4. Calculates probability of flood events per day (total floods / total days) and the number of expected floods per year by multiplying that by the average number of days in a year (365.25).
5. Reruns that simulation 1000 times, and looks at properties of the resulting distribution of estimated rates of flooding per year.

The key step here is defining “random” deviations in a manner that creates deviations with statistical properties close to those of the real deviations of the past. We do that by simulating an ARMA process based on historic data. An ARMA process is a statistical model used to predict values from a time series (The acronym reflects the fact that the model includes both “autoregressive” and “moving average” model components.

We also developed a model that resamples deviations, but without modeling serial autocorrelation, the model consistently overestimated number of days with flood events, so we do not present it here.

## Model Weaknesses

(Note this is a model, not reality, so we need to evaluate what it does well and not so well. Here are some initial thoughts.

1. We may **still** underestimate flooding at the back end of a tidal epoch (and overestimate it early in the tidal epoch),because nothing in the model addresses sea level rise that occurs during simulated tidal epoch.
2. The ARMA-derived simulated deviations are based on stationary normal errors, and thus we do not expect simulated deviations to be as skewed as the real deviations. That also may underestimate flooding.
3. We are not handling seasonal phenomena in the ARMA model, even though seasonal patterns in storm intensities and wind direction may mean the deviations from tidal predictions are not independent of time of year.)
4. We are not addressing the residual (tide-related) periodic components of the autocorrelogram and partial correlogram. (Confusingly, the literature of time series modeling sometimes uses “seasonal” as a term to describe any periodic structure included in an ARIMA model, so we would use a “seasonal” term to model tide-based autocorrelation, as well as yearly ones.)

## Developing an ARIMA Simulator

We can fit an ARMA model to the tidal deviations data, and then simulate random deviations based on model parameters. We fit an ARIMA model using the auto.arima() function from the forcast package. auto.arima() searches for a “best” ARIMA model based on AIC or other information criterion.

By definition, the observed deviations should be stationary, with mean zero, so we need not do any preliminary differencing. We signal that to auto.arima() with d = 0. We signal a stationary, non-seasonal process, with zero mean as well, to take advantage of properties of the deviations.

(In fact, as we explain elsewhere, the deviations has a small positive slope, as the tidal predictions did not take into account sea level rise. The effect is small, and we chose to overlook it for modeling purposes.

We speed things up by setting stepwise and approximation arguments to FALSE and TRUE, respectively. You can request a trace to show how the model search progressed, as we have done. You can also decide whether to fit “seasonal” terms, but to do that, you have to pass a frequency to the time series object. We found “seasonal” models were very time consuming to fit, and we could not simulate from them readily, so we chose not to use them here.

### Select ARIMA Model

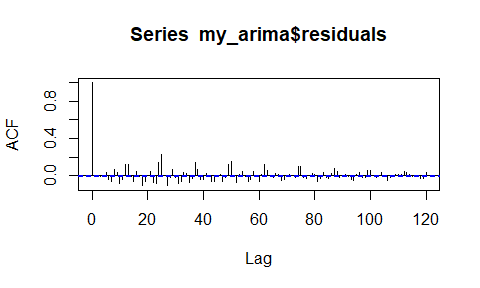
This takes around 20 seconds to run.

my\_arima <- auto.arima(epoch$deviation,  
 d = 0, # No differencing.  
 stationary = TRUE,  
 seasonal = FALSE,  
 allowdrift = FALSE, # not sure how this differs from "stationary"  
 allowmean = FALSE, # Our time series has mean = 0  
 stepwise = TRUE, # FALSE is faster, less accurate  
 approximation = TRUE, # FALSE is slower  
 trace = TRUE)  
#>   
#> Fitting models using approximations to speed things up...  
#>   
#> ARIMA(2,0,2) with zero mean : -435596.2  
#> ARIMA(0,0,0) with zero mean : -169284.9  
#> ARIMA(1,0,0) with zero mean : -411229.1  
#> ARIMA(0,0,1) with zero mean : -320279.9  
#> ARIMA(1,0,2) with zero mean : -432024.5  
#> ARIMA(2,0,1) with zero mean : -423536.1  
#> ARIMA(3,0,2) with zero mean : -440618.7  
#> ARIMA(3,0,1) with zero mean : -440071  
#> ARIMA(4,0,2) with zero mean : -440489  
#> ARIMA(3,0,3) with zero mean : -441158.7  
#> ARIMA(2,0,3) with zero mean : -436175.3  
#> ARIMA(4,0,3) with zero mean : -437015  
#> ARIMA(3,0,4) with zero mean : -445797.6  
#> ARIMA(2,0,4) with zero mean : -438286  
#> ARIMA(4,0,4) with zero mean : -445982.9  
#> ARIMA(5,0,4) with zero mean : -446049.8  
#> ARIMA(5,0,3) with zero mean : -445678.1  
#> ARIMA(5,0,5) with zero mean : -446294.9  
#> ARIMA(4,0,5) with zero mean : -445805.9  
#>   
#> Now re-fitting the best model(s) without approximations...  
#>   
#> ARIMA(5,0,5) with zero mean : -446301.2  
#>   
#> Best model: ARIMA(5,0,5) with zero mean  
#my\_arima  
(my\_coefs <- coef(my\_arima))  
#> ar1 ar2 ar3 ar4 ar5 ma1   
#> 0.19997814 0.54092109 0.03811914 -0.44522926 0.54275104 0.82350247   
#> ma2 ma3 ma4 ma5   
#> -0.14159188 -0.36185595 0.29005757 0.08291017  
(my\_sigma2 <- my\_arima$sigma2)  
#> [1] 0.003976376

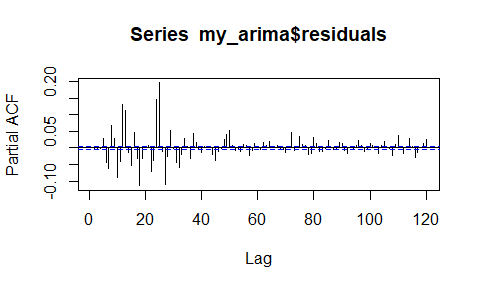
That fits a fairly complex ARIMA model, with order 5 autoregressive and moving average terms. Looking at the AIC values reported by the trace output, the ARIMA(4,0,4) and ARIMA(3,0,4) do nearly as well at reduced model complexity.

### Evaluate Model Residuals

acf(my\_arima$residuals, na.action = na.pass, lag.max = 24\*5)



pacf(my\_arima$residuals, na.action = na.pass, lag.max = 24\*5)



The ARIMA model successfully matched most of the non-periodic structure in the observed deviations. Model residuals still show higher-order periodic structure, especially around 12 to 13 and 24 to 25 hours, which correspond to tidal periods.

### Evaluating Simulated Deviations from the ARMA Model

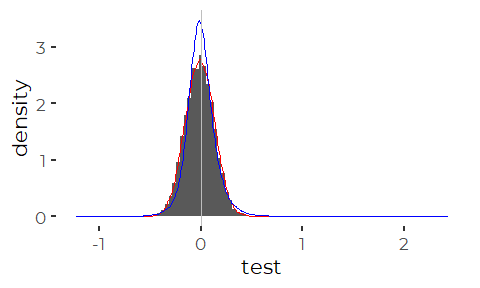
We can simulate a “random” time series that matches the estimated ARIMA structure of the observed deviations using arima.sim().

test <- arima.sim(n = 365\*24,  
 model = list(ar = my\_coefs[1:5], ma = my\_coefs[6:10]),  
 sd = sqrt(my\_sigma2))

#### Compare to Observed Deviations

We plot the actual deviations (blue) and the simulated deviations (red). The simulated deviations are simulated, so they vary slightly each time this code is run.

ggplot() +  
 geom\_histogram(aes(x = test, y = ..density..), bins = 100) +  
 geom\_density(aes(x = test), color = 'red') +  
 geom\_density(aes(x = deviation), color = 'blue', data = epoch) +  
  
 geom\_vline (xintercept = 0, color = 'grey')  
#> Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.

 As expected, the simulated data is less skewed, and less heavy-tailed than the original data. It does have heavier tails than a normal distribution.

#### Moments

Lets look at the moments of the simulated data. (Results will vary; this is a simulation).

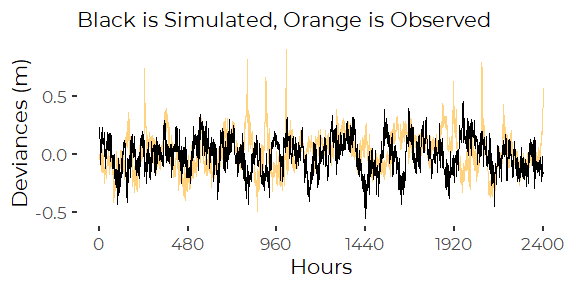
original <- c( round(mean(test),4),  
 round(sd(epoch$deviation),4),  
 round(skewness(epoch$deviation),4),  
 round(kurtosis(epoch$deviation),4))  
names(original) <- c('Mean', 'SD', 'Skewness', 'Kurtosis')  
  
simulated <- c( round(mean(test),4),  
 round(sd(test),4),  
 round(skewness(test),4),  
 round(kurtosis(test),4))  
names(simulated) <- c('Mean', 'SD', 'Skewness', 'Kurtosis')  
  
rbind(original, simulated)  
#> Mean SD Skewness Kurtosis  
#> original -0.0108 0.1453 0.5472 7.3838  
#> simulated -0.0108 0.1413 0.0011 2.9796

Mean and SD are excellent fits, but both skewness and kurtosis are lower than the historical data. That will tend to mean we underestimate extreme floods. It is less clear how important that will be for our estimates of frequency of moderate flood events.

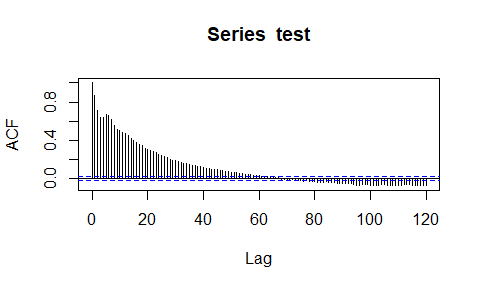
#### Autocorrelatons

We examine temporal patterns and autocorrelation.

ggplot() +  
 geom\_line(aes(x = (1:(24\*100)), y = epoch$deviation[1:(24\*100)]),  
 color = 'orange', alpha = 0.5) +  
 geom\_line(aes(x = (1:(24\*100)), y = test[1:(24\*100)])) +  
   
 scale\_x\_continuous(breaks = seq(0, 24\*100, 24\*20)) +  
 ylab('Deviances (m)') +  
 xlab('Hours') +  
 labs(subtitle = 'Black is Simulated, Orange is Observed')

 The key thing to notice is that the apparent overall structure of the pseudo-periodicity in the record looks visually similar. You can also see the small number of large deviations in the original data (‘floods’) that are poorly modeled by the ARIMA model. The simulated data does not mimic the effect of large storms well.

acf(test, na.action = na.pass, lag.max = 24\*5)



The overall temporal pattern of the simulation looks roughly comparable to the original deviations out to a couple of days (the timescale is in hours). The autocorrelation structure shows a similar initial decrease in correlation, but it does not level out at ACF ~ 0.2, or get to that level quite as quickly as did the original data. The modeled values also the periodic components of the real data.

#### Upper Percentiles

We compare extreme percentiles

cat("Original Data\n")  
#> Original Data  
round(quantile(epoch$deviation, c(.90, .95, .99, .999)),2)  
#> 90% 95% 99% 99.9%   
#> 0.17 0.24 0.42 0.76  
  
cat("\nSimulated Data\n")  
#>   
#> Simulated Data  
round(quantile(test, c(.90, .95, .99, .999)),2)  
#> 90% 95% 99% 99.9%   
#> 0.17 0.22 0.32 0.44

So the differences between simulated and original deviations are significant at the upper tail, where simulated deviations are much less extreme compared with the original deviations.

#### Tail Probabilitieis

We would actually be more interested in the inverse of that calculation. What is the probability of falling over a specific

probs\_dev <- numeric(6)  
probs\_sim <- numeric(6)  
levels <- seq(0.25, 0.5, 0.05)  
for (n in seq\_along(levels)) {  
 probs\_dev[[n]] <- round(mean(epoch$deviation > levels[[n]]),3)  
 probs\_sim[[n]] <- round(mean(test > levels[[n]]),3)  
}  
  
a <- rbind(levels, probs\_dev, probs\_sim)  
rownames(a) <- c('Deviation', 'Original Data', 'Simulated Data')  
a  
#> [,1] [,2] [,3] [,4] [,5] [,6]  
#> Deviation 0.250 0.300 0.350 0.400 0.450 0.500  
#> Original Data 0.045 0.028 0.018 0.012 0.008 0.005  
#> Simulated Data 0.032 0.014 0.006 0.002 0.000 0.000

So the probability of observing a deviation above moderate levels of about eight inches are similar, but the probabilities diverge somewhat for higher deviations.

# Simulation Function

We create a function that simulates one possible time series of tidal heights.

**Note that this function relies on existence of parameters and standard error from an existing ARIMA(5,0,5) model.** We pass them via a model parameter.

sim\_once <- function(dat, pr\_sl, dts, slr, flood = FLOOD\_ELEVATION,  
 coefs = my\_coefs,   
 sigma2 = my\_sigma2) {  
   
 # dat is a dataframe  
 # pr\_sl is the data column containing the (astronomical) sea level predictions  
 # dts is a data column of identifiers by date / day  
 # slr is the value for SLR for estimating future tidal elevations   
 # flood is the selected elevation for something to be considered a flood  
 # coefs is a list of coefficients as produced by arima() or auto.arima()  
 # sigma2 is the sigma2 measure of variation from arima() or auto.arima()  
   
 # Returns the mean number of floods per year over ONE simulated tidal epoch.  
   
 # We quote data variables, and look them up by name  
 # Caution: there is no error checking. if things are not working it may be   
 # because the columns do not exist.  
 pr\_sl <- as.character(ensym(pr\_sl))  
 #dev <- as.character(ensym(dev))  
 dts <- as.character(ensym(dts))  
   
 #Simulate one tidal epoch of hourly tidal elevations  
 val <- dat[[pr\_sl]] + slr + arima.sim(n = length(dat[[pr\_sl]]),  
 model = list(ar = coefs[1:5],   
 ma = coefs[6:10]),  
 sd = sqrt(sigma2))  
   
 #create a dataframe, for convenient calculations  
 df <- tibble(theDate = dat[[dts]], sim = val)  
   
 #Calculate results  
 res <- df %>%  
 group\_by(theDate) %>%  
 summarize(exceeded = any(sim > flood),  
 .groups = 'drop') %>%  
 summarize(days = sum(! is.na(exceeded)),  
 floods = sum(exceeded, na.rm = TRUE),  
 floods\_p\_yr = 365.25 \* floods/days) %>%  
 pull(floods\_p\_yr)  
   
 return(res)  
}

## Test The Function

sim\_once(epoch, Prediction, theDate, 0, flood = FLOOD\_ELEVATION)  
#> [1] 4.013156

That value is a random value that will vary each time the function is run.  
In general, it is fairly close to the observed frequency of flooding over the official tidal epoch (about 3.37).

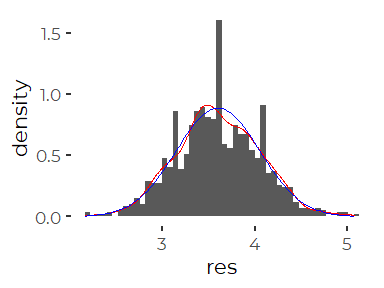
## Run 1000 Simulations, Without SLR

The following takes a couple of minutes to run. It runs the simulation function a thousand times and sticks the results into a vector (res).

set.seed(54321)  
samp = 1000  
res <- numeric(samp)  
for (iter in seq(1, samp)) {  
 res[[iter]] <- sim\_once(epoch, Prediction, theDate, 0)  
}

### Look at Simulation Results

ggplot() +  
 geom\_histogram(aes(x = res, y = ..density..), bins = 50) +  
 geom\_density(aes(x = res), color = 'red') +  
 stat\_function(fun = dnorm, args = list(mean = mean(res), sd = sd(res)),  
 color = 'blue')



mean(res)  
#> [1] 3.59885  
sd(res)  
#> [1] 0.4500711  
skewness(res)  
#> [1] 0.05404328  
kurtosis(res)  
#> [1] 2.996686

So, the core finding is that the expected number of flood events per year over a 19 year period without any sea level rise post tidal epoch is 3.60 +/- 0.45 events per year. That is in fairly close agreement to the value actually observed during the tidal epoch from 1983 through 2001.

Note that the SD reported here is the standard error of the mean number of flood events over a 19 year period based on 100 simulations. It is not a measure of interannual variation in the number of flood events. (This analysis does not provide an estimate of that figure.)

# Wrapping It All Up

How much will SLR increase the frequency of days with flooding? We focus on the period of the tidal epoch, when difference between observed and predicted tides were minimized. That makes these results only qualitatively related to change in frequency of flooding compared to recent times.

### Direct Substitution

We ran these calculations previously using the MGS approach. We run the code based on the tidal epoch, updated for about 30 years of average sea level rise, and for four scenarios, zero, one foot, two foot, and three foot of SLR.

hist\_slr <- 0.075  
mgs\_est\_modern <- f\_flood\_means(epoch, theDate, MLLW,  
 c(hist\_slr,   
 0.3048 + hist\_slr,   
 0.6096 + hist\_slr,   
 0.9144 + hist\_slr))  
mgs\_est\_modern  
#> Mean SD  
#> floods\_0 7.157895 4.621966  
#> floods\_1 89.947368 26.270968  
#> floods\_2 280.631579 30.775550  
#> floods\_3 359.526316 5.450468  
#> floods\_p\_yr0 7.193990 4.674937  
#> floods\_p\_yr1 90.322829 26.756018  
#> floods\_p\_yr2 281.630551 31.485638  
#> floods\_p\_yr3 360.704560 3.709245  
#> attr(,"slr")  
#> [1] 0.0750 0.3798 0.6846 0.9894  
  
`One Foot Multiplier` <- round(mgs\_est\_modern[3,1] / mgs\_est\_modern[2,1],2)  
`Two Foot Multiplier` <- round(mgs\_est\_modern[4,1] / mgs\_est\_modern[2,1],2)  
`Three Foot Multiplier` <- round(mgs\_est\_modern[5,1] / mgs\_est\_modern[2,1],2)  
  
the\_col <- c( mgs\_est\_modern[2,1],  
 mgs\_est\_modern[3,1],  
 mgs\_est\_modern[4,1],  
 mgs\_est\_modern[5,1],  
 `One Foot Multiplier`,  
 `Two Foot Multiplier`,  
 `Three Foot Multiplier`)  
rm(`One Foot Multiplier`,  
 `Two Foot Multiplier`,  
 `Three Foot Multiplier`)

(Note here the SD is an interannual SD, not a standard error from a simulation).

## ARIMA

We create a wrapper function to automate running the ARIMA simulation repeatedly for each sea level scenario.

The function sim\_auto() is not fully encapsulated, as it requires the data frame passed as the df parameter to contain a data column named “Prediction” and another named “theDate”.

sim\_auto <- function(df, slr, samp) {  
 res <- numeric(samp)  
 for (iter in seq(1, samp))  
 res[[iter]] <- sim\_once(df, Prediction, theDate, slr)  
 return(res)  
}

We run this simulation for the same sea level rise scenarios. The code here uses lapply() to apply a function over a list, and return a list. Here the function returns a list of vectors containing results of separate simulation runs.

The following takes several minutes to run.

# set.seed(12345)  
samp = 1000  
simulates <- lapply(((0:3 \* 0.3048) + 0.075),   
 function(x) sim\_auto(epoch, x, samp))

names(simulates) = c('Present', 'Plus One Foot SLR', 'Plus Two Foot SLR', 'Plus Three Foot SLR')  
simulates <- do.call(bind\_cols, simulates)  
  
simulates <- simulates %>%  
 summarize(across(everything(), mean, na.rm = TRUE)) %>%  
 rowwise() %>%  
 mutate(`One Foot Multiplier` = round(`Plus One Foot SLR`/`Present`,2),  
 `Two Foot Multiplier` = round(`Plus Two Foot SLR`/`Present`,2),  
 `Three Foot Multiplier` = round(`Plus Three Foot SLR`/`Present`,2))  
simulates  
#> # A tibble: 1 x 7  
#> # Rowwise:   
#> Present `Plus One Foot ~ `Plus Two Foot ~ `Plus Three Foo~ `One Foot Multi~  
#> <dbl> <dbl> <dbl> <dbl> <dbl>  
#> 1 8.88 94.8 276. 360. 10.7  
#> # ... with 2 more variables: `Two Foot Multiplier` <dbl>, `Three Foot  
#> # Multiplier` <dbl>  
  
v <- simulates$`One Foot Multiplier`

This analysis suggests a one foot sea level rise would increase flooding by a factor of about 10.7. That is not all that different from the ratio we calculated before.

## Make a Nice Table

a <- cbind(t(round(simulates,2)), round(the\_col,2))  
colnames(a) <- c('ARIMA', 'Add SLR')  
knitr::kable(a, caption ='Mean Predicted Annual Flood Events And Multipliers Two Models')

Mean Predicted Annual Flood Events And Multipliers Two Models

|  |  |  |
| --- | --- | --- |
|  | ARIMA | Add SLR |
| Present | 8.88 | 89.95 |
| Plus One Foot SLR | 94.79 | 280.63 |
| Plus Two Foot SLR | 275.95 | 359.53 |
| Plus Three Foot SLR | 360.29 | 7.19 |
| One Foot Multiplier | 10.67 | 3.12 |
| Two Foot Multiplier | 31.07 | 4.00 |
| Three Foot Multiplier | 40.57 | 0.08 |