Johan Södling
SMHI, 2020-10-13

A method for creating synthetic gaps

Introduction

- At SMHI we have an active homogenization project where we are working with tools such as HOMER and Climatol. We apply these tools to monthly data (in particular temperature data) to calculate a ”climate indicator” which is a summarized temperature value for Sweden.
- Both of these tools have several features, but the one that is of most interest for this particular presentation is the gap filling feature. Both HOMER and Climatol can fill in any missing data whenever there is a gap in the time series. This is done because many statistics and measures require full data coverage from the data set.
- When we use gap filling on real station data it is impossible to know how well this gap filling performed, since we are missing these data values. This is where the idea of ”gap making” comes in: by creating synthetic gaps on time series with full data coverage, and then applying these gap filling methods, we can then compare the gap filled values to the true values.
- Of course, in order for this to work well, the method that creates these gaps must do so in a ”good” way (more on this soon). This is where this method comes in!

Some terminology

- Some quick terminology before we dive in:
  - A *gap mask* is defined as the binary representation of if we have data or a gap. We usually model observed data with a 1 and a gap with 0. This is the type of output that our model will create, and then we can apply it this mask on any data series of the same length. So if we want to create a gap mask for a monthly time series that is 1200 months long, the output from this model will be a list that is 1200 elements long filled with 0:s and 1:s, with 0 for timestep where the model wants a gap, and 1 otherwise.
  - A gap is any section of missing data in the time series. In this presentation we are working with monthly data, so the time unit will be ”months”. But in theory it could be on any temporal resolution.
  - We define an *uptime* (the opposite of ”downtime, or gap”) as any uninteruppted section in the timeseries that have measurements.
- So to use our terminology in an short example: a gap mask is made of alternating uptimes and gaps of varying length.

What is a ”good” gap mask?

- There are a few aspects that make a synthetically made gap mask ”good”:
  1. The length of the uptimes should be comparable to what we see in station data
  2. Also, the length of the gaps should be close to the station data gap lengths.
  3. The proportion of missing data should also be comparable to station data. Note that this goes hand in hand with points 1 and 2, but should be investigated separately.

Method overview

- The main idea behind this model is to create a gap mask by iteratively placing sections of uptime and gaps.

- The length of each uptime and gap is a random value created from the training data (more on this soon)

- The model needs training data to ”learn” the length of uptimes and gaps. If you have several stations as input data (in our study we have 100 stations with monthly mean temperatures) you can split your stations into a training set and a validation set.
- In this study we used 75 of the stations as training data, and left the remaining 25 stations for validation. So we will compare the gaps created from the model, trained on these 75 stations, against these 25 validation stations. This is to make sure that the model has no knowledge of the data we will evaluate it against.
  - The 75 training stations are selected randomly from the 100 stations.

Some notes about the data

- As mentioned, the analysis was done on 100 Swedish stations with monthly temperature data. More exactly, these are ”pseudo-stations”, i.e. each of the time series can consist of several stations that were considered similar enough to be put into the same timeseries. The purpose of this is to get long timeseries that stretch back to 1860 in our case (at least for some of the stations).
- In many cases a pseudostation is made by connecting the time series from a discontinued manual station to the automatic station that replaced it (many stations in Sweden got replaced by automatic stations in 1995/1996).
- These 100 (pseudo)-stations were selected to accurately represent Sweden’s climate, so they are spread out over the country.

Method

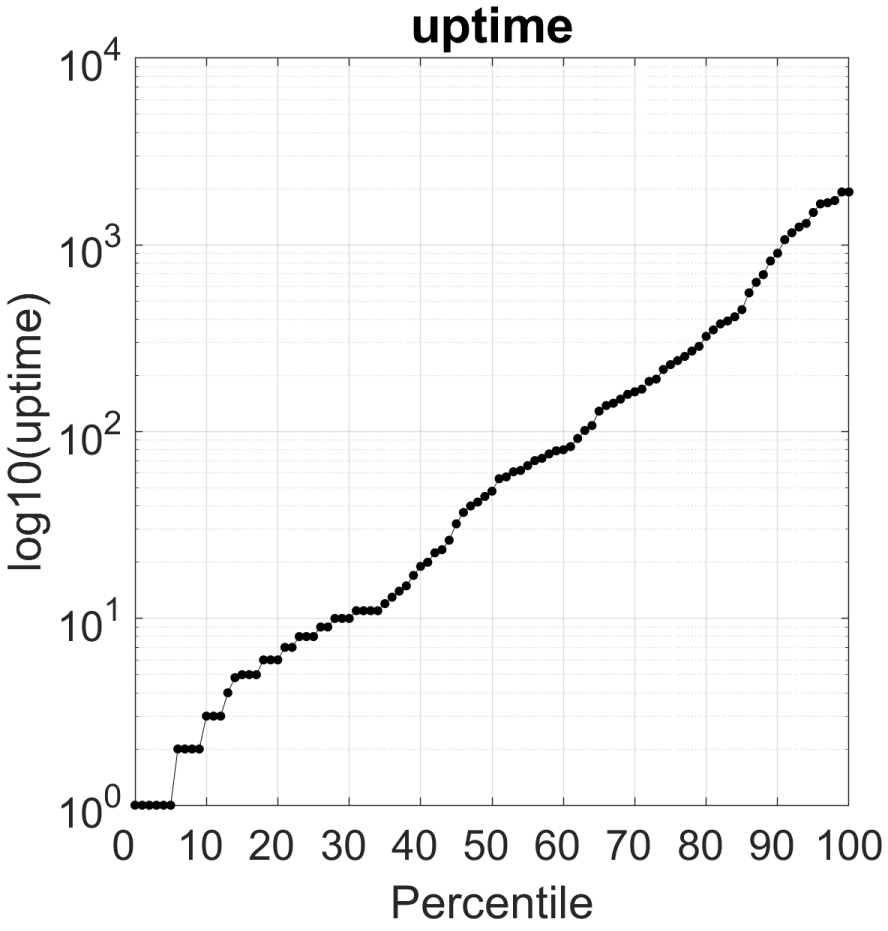
- From the stations categorized as training data, extract the length of all the uptimes. Whenever the model is to create a new uptime section, we extract a random percentile from this training set. More formally, create random number between 0 and 100, and draw that percentile from the training data. This gives us how long an uptime section we will place. Once we have placed this uptime section, we do the same process but with gaps.

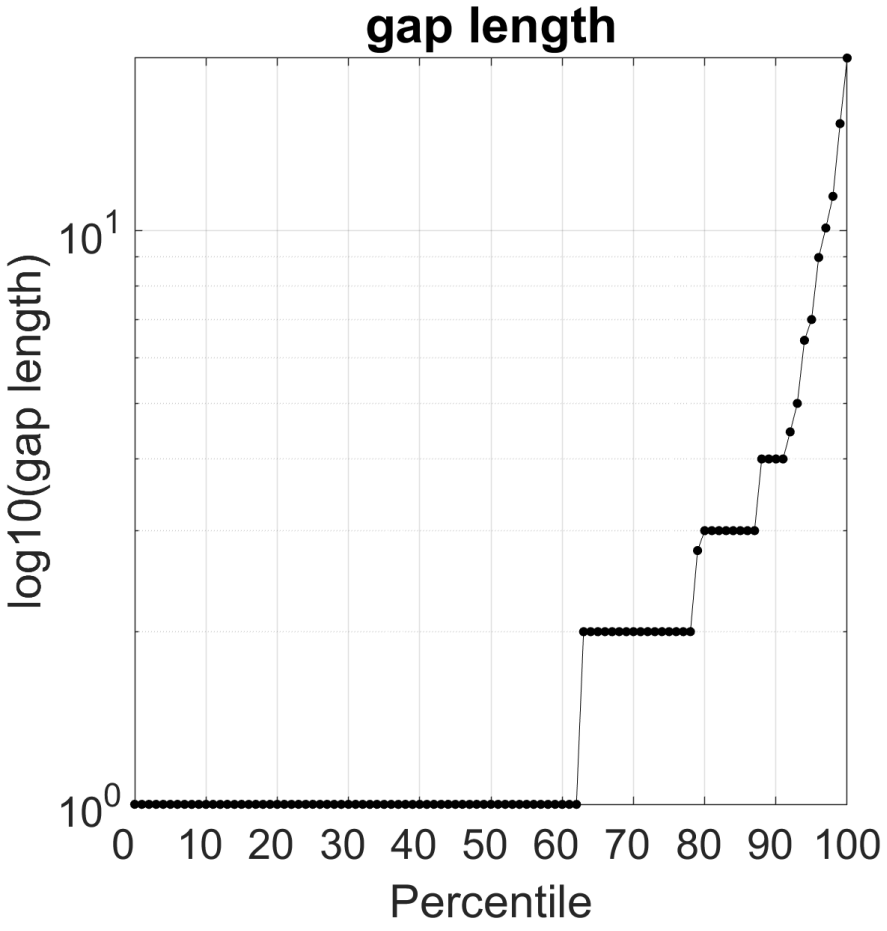
- Determining uptime/gap length by sampling random percentiles from training data has advantages and disadvantages. In particular, we are bound by the range of the training data; no uptime/gap can be shorter/longer than the shortest/longest value in the training data (which would correspond to the 0 and 100 percentile respectively).
  - A good thing about this method is that it should work for any dataset, since it makes no assumptions on how the gaps and uptimes in the data is distributed.
  - … and since we only place values in the range that we have observed, we should never get unrealistic uptime/gap lengths.

Method

- So far all this has been a bit abstract, so let's look at an example. This is a plot of the 0,1,…,100 percentiles for uptime length from the training data. So these are the percentiles for uptime length for all the uptimes found in the 75 training stations. We also include the same type of plot but for gap length on the right.
  - Note that the *Y*-axis is logarithmic in both plots!

Uptime / gap plots





Uptime / gap plots

- Before we go through the example of how the model creates a gap mask, let's reflect on the two figures for a few seconds.
  - The uptime plot looks quite linear on a logarithmic axis, meaning that it can take on a wide range of values. The median uptime length is about 45 months, but the longest one is an impressive 1920 months = 160 years (which is the total length of the dataset, so there is a station in the training data with no missing values).
  - If we look at the gap length figure we see that most gaps are just 1 month long (which makes sense, most gaps in data get fixed after just 1 month). The longest gap is 20 months for these training stations.

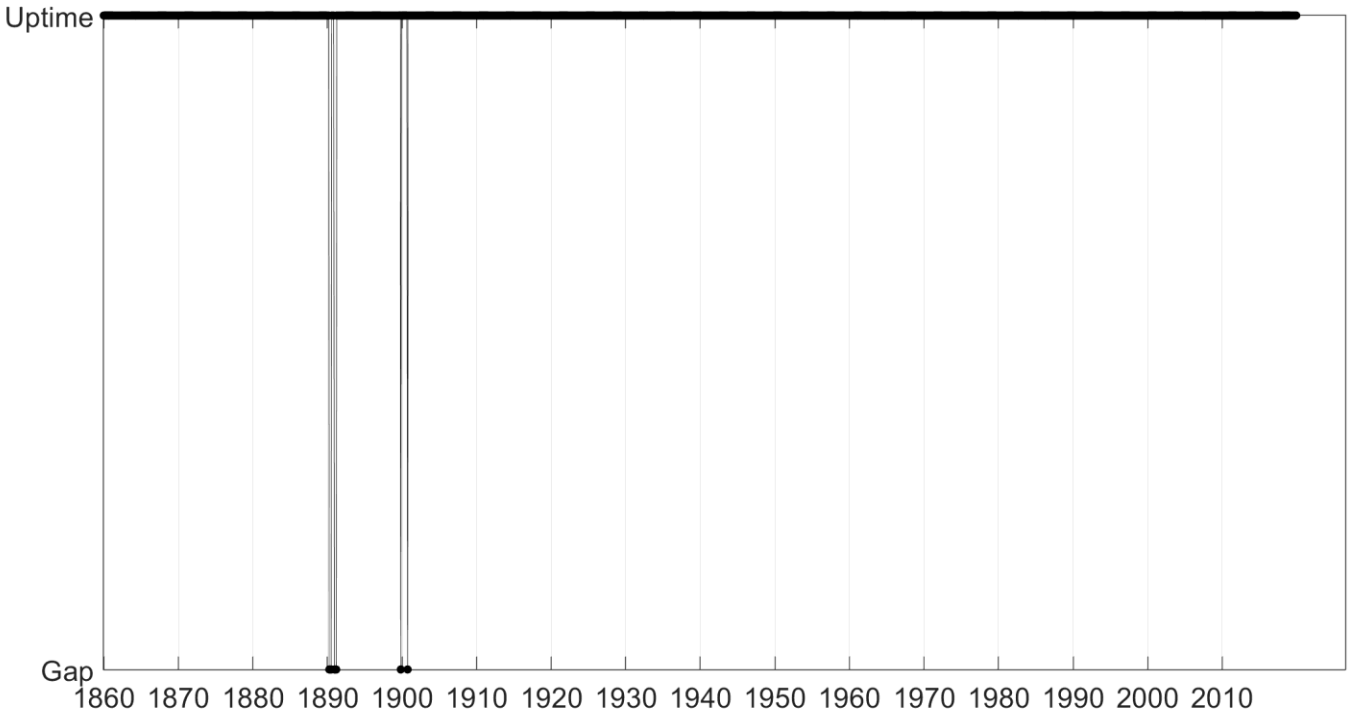
Constructing gap masks – an example

- The two plots on the previous slide is the model's ”knowledge” about uptime length and gap length. Let's go through a very quick example of how the model creates gap masks:
  - The model starts by placing a section of uptime. It generates a random number between 0 and 100, let's say we get 81. We look up what the uptime value is on percentile 81, in this case it is 350.8. We round this to the nearest integer, giving us 351. So the first section of the gap mask will be 351 months (about 29 years) of interrupted data.
  - After this step we create a section of gap in the same way. A random number between 0 and 100 is generated, let's say we get 26. If we read this value on the gap figure we see that this is a gap of 1 month. So we place 1 month of missing data in the gap mask, and then repeat this process by placing a section of uptime again. Once the length of the gap mask meets the desired length, we break to model. If it exceeds the desired length (which can happen) we cut off any excess time steps in the gap mask.

Constructing gap masks – an example

- Note that the percentile we generate is a float and not an integer, but in this example I used whole percentiles since it is easier to read in the figures.
- It is important to note that if you use another RNG seed before running the program (giving you other random numbers) the gap mask will look very different, since we draw other values from the training data.
  - This allows us to use this model for monte carlo simulation and similiar kind of studies, since we can create as many gap masks as we want.
- I present two pictures here with created gap masks for the time period 1860 to 2019. So these gap masks could be applied to time series that have the same data period.

A generated gap mask with rng seed = 0



SMHI

… and another one with rng seed = 1

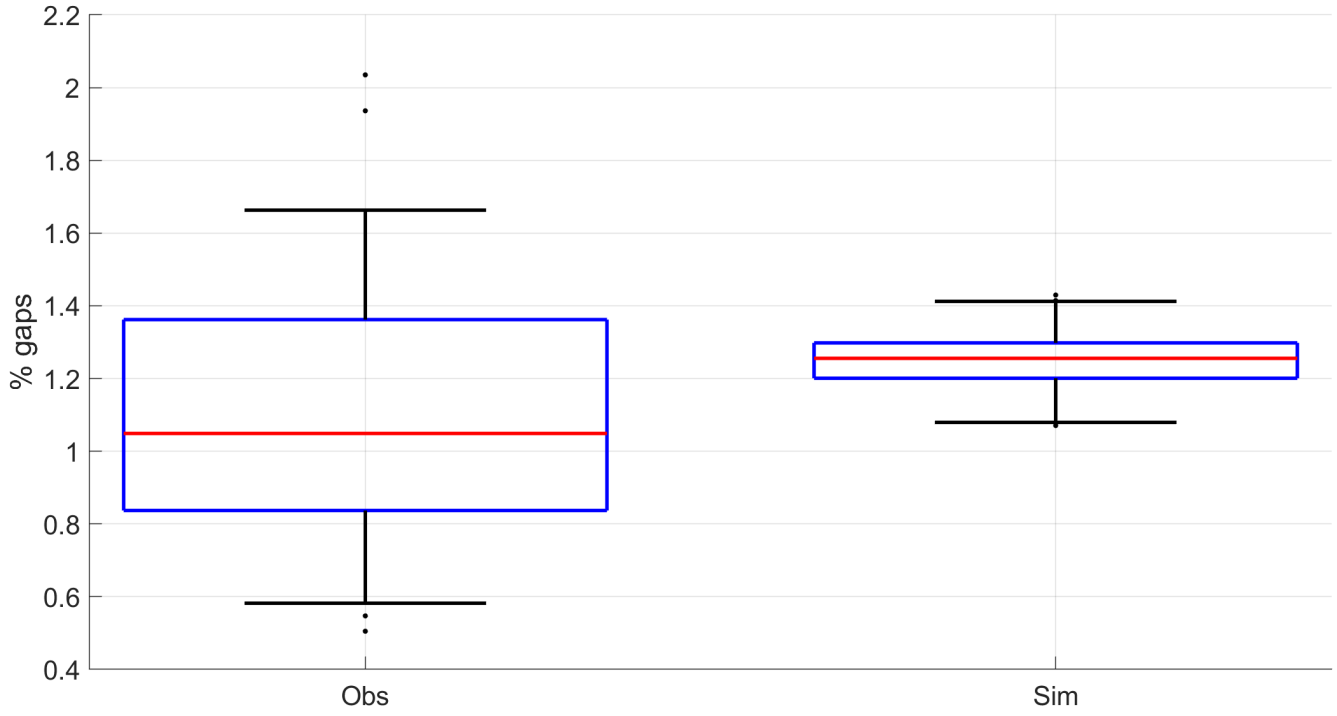
Additional notes

- We now basically know everything about how the model works, but there are a few more things I would like to bring up before we look at the results:
  - The model has a feature where you can train different models on different timewindows, for example a 30 year moving window with 10 year steps. This allows the model to have different datasets for how gaps/uptimes look for different timewindows. So there is one dataset for 1901-1930, another for 1911-1940, etc. This can be useful if you believe that the behaviour of gaps changes during your timeperiod.

Results

- Also, I should mention that here I have trained several models, each on a different random selection of 75 stations. This was done to avoid the results depending on the particularly chosen training set. I trained a total of 100 models for these results. Of course, each model is only compared to its validation data, so no model will have any knowledge of the ”ground truth”. I will explain details for each figure that follows.

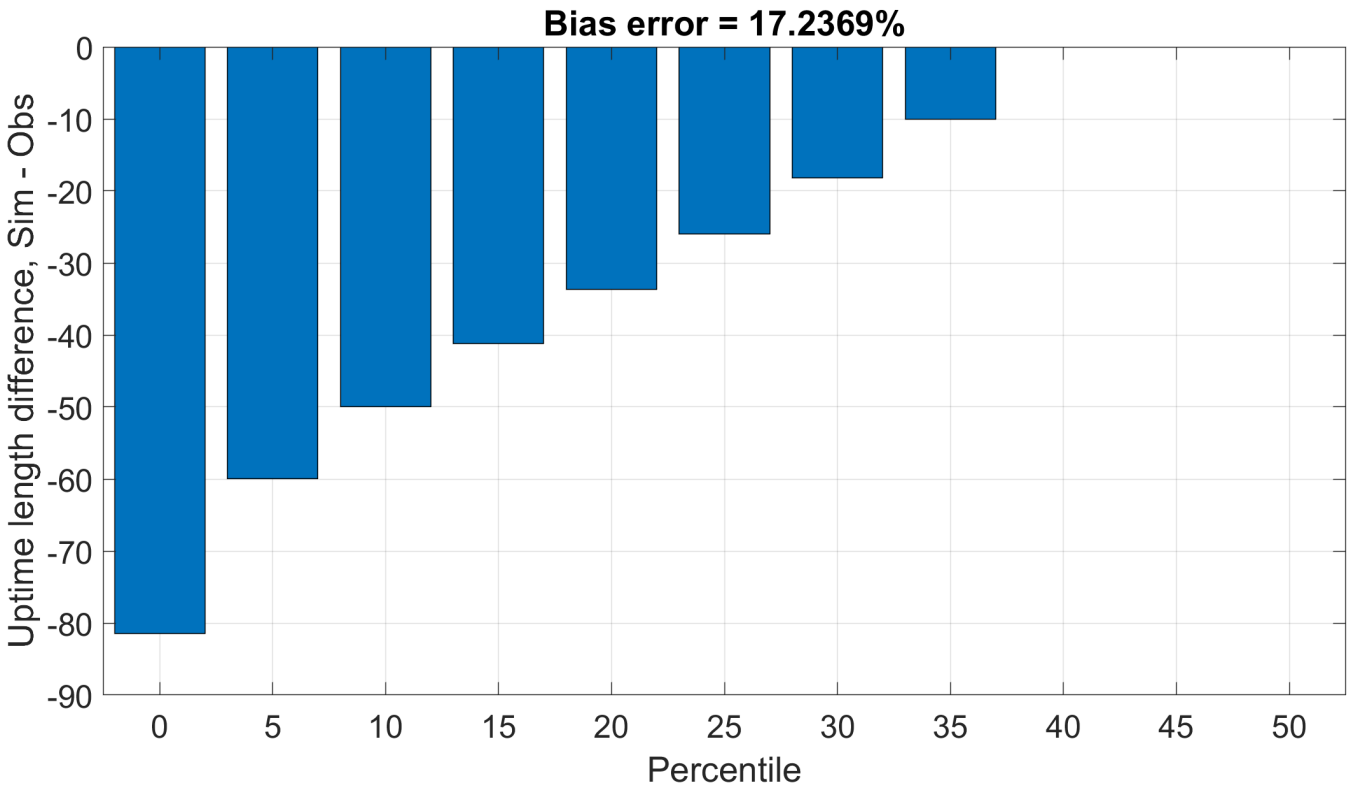
Boxplot of % gaps



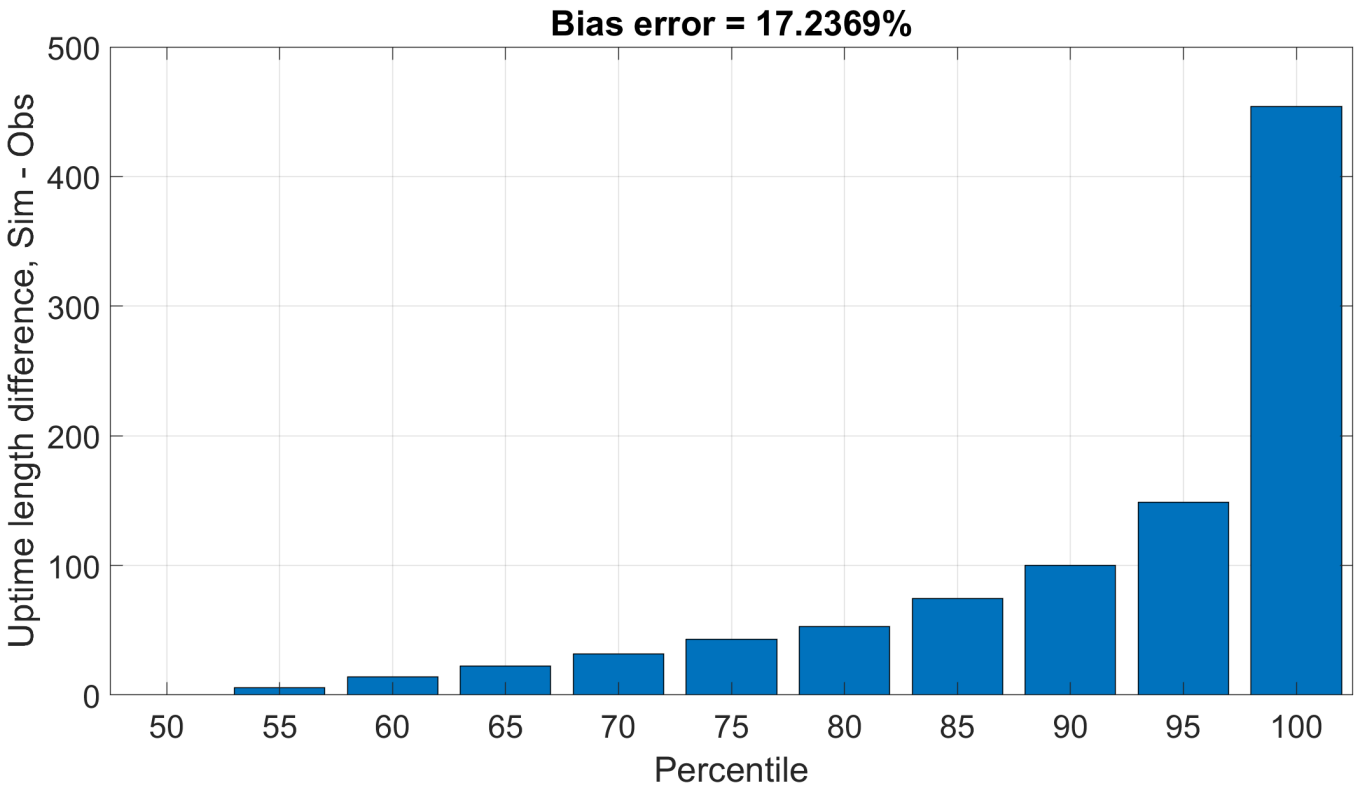
Barplots of percentile differences

- The upcoming figures show the difference in gap/uptime length. More exactly, for each of the 100 trained models we calculate percentiles 0,1,2,…,100 for the model and station data (from the validation stations). We now calculate the diffence model – station for uptime length and gap length.
- So if the model is perfect all these differences would be 0, but in reality at least the highest/lowest percentiles can differ quite a lot. So we get 101 percentiles for each trained model, and we have 100 models, and we do this for all of them. This gives us a data set of 101 * 100 = 10 100 values. It is on this data set that the percentiles 0,1,…,100 in the figures below have been calculated.
- Here I present the bottom 50 and top 50 percentiles in separate pictures (since they have different y-scales). In the title I present the bias, i.e. if the model systematically under- or overestimates the gap/uptime length. The unit is presented in % of obs, so if the value is for example +25% it means that on this percentile the model length is 25% higher than station data.

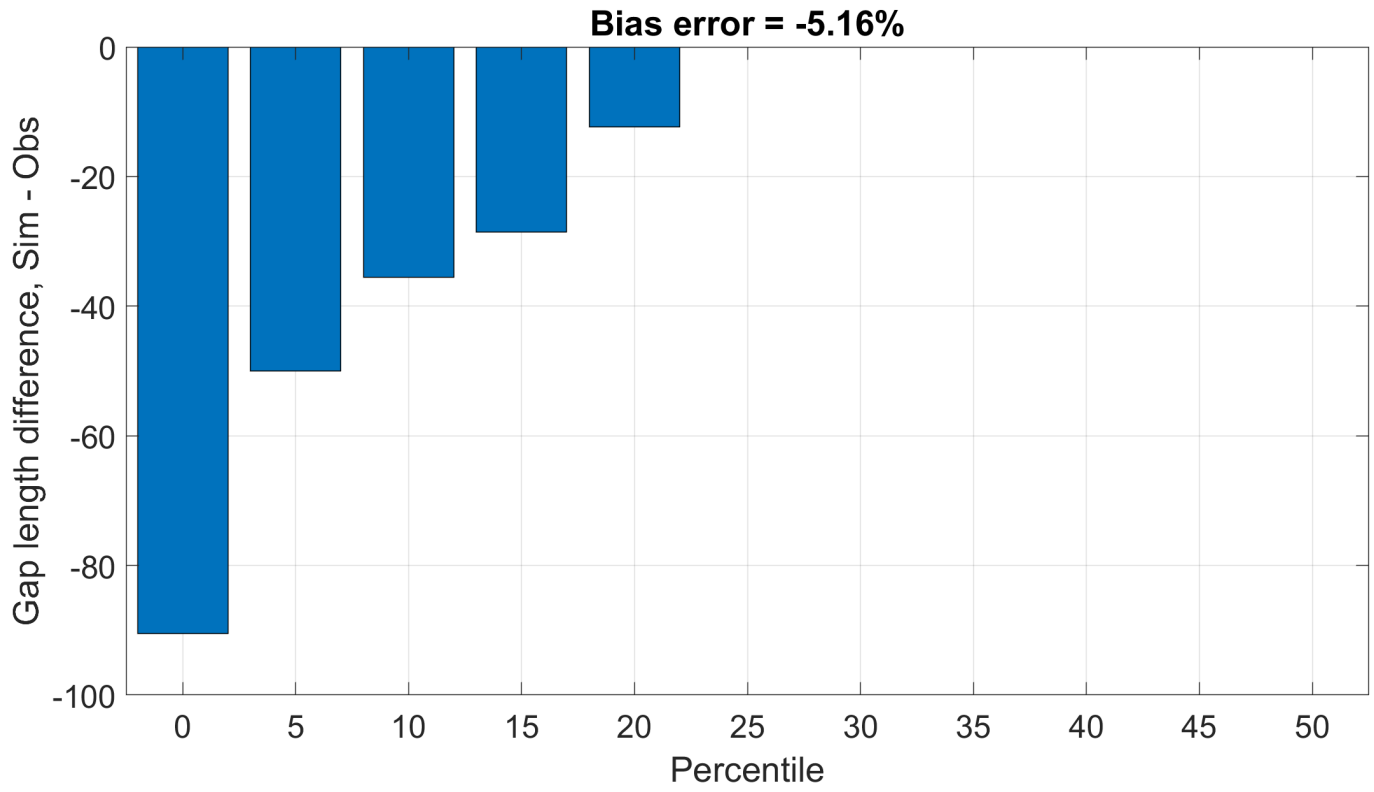
Barplots of percentile differences



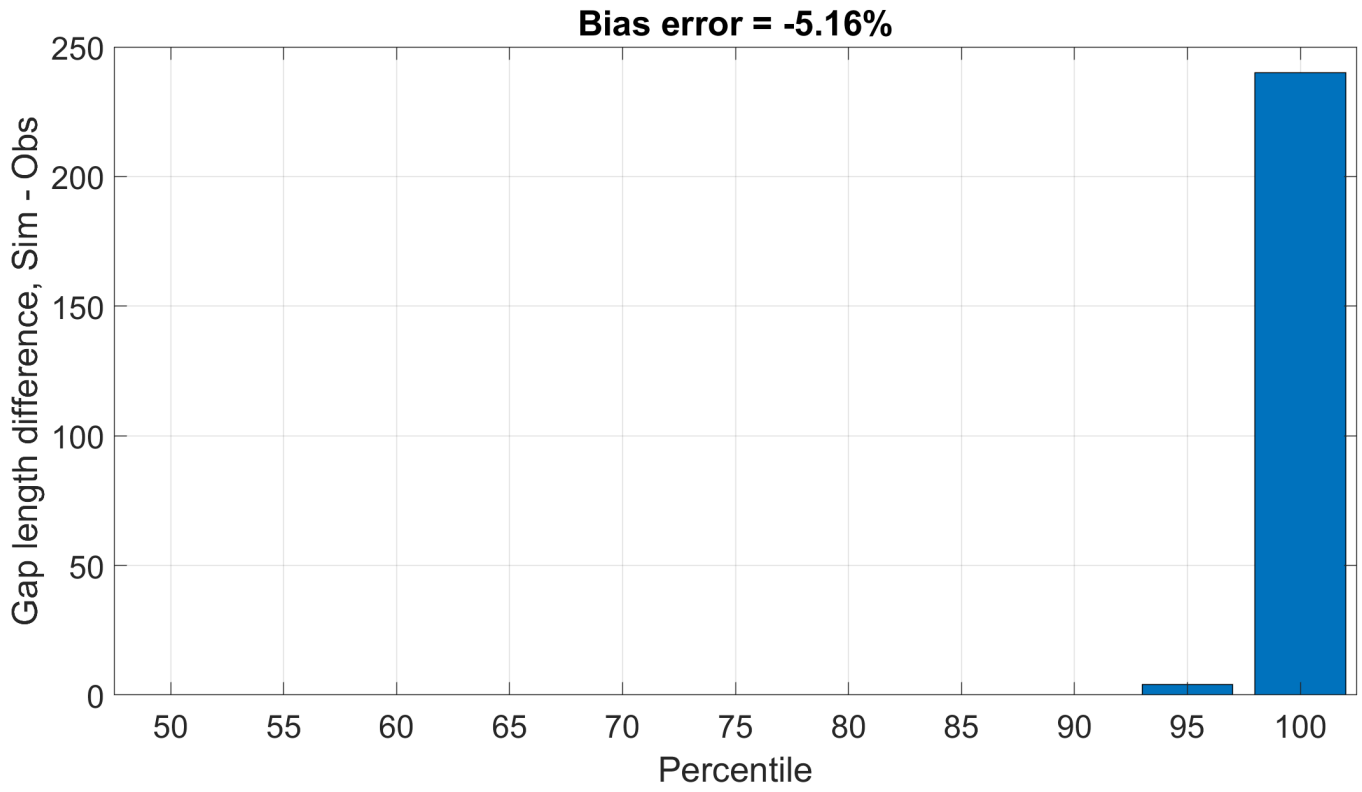
Barplots of percentile differences



Barplots of percentile differences



Barplots of percentile differences



Conclusions

- The gap making model presented here seems to perform quite well. The percentage of missing data is in the same range as in the validation data set (although the spread is a bit too small).
- When looking at the percentiles of the difference in gap length/uptime to station data, most of the percentiles are low. The model seems to sometimes produce too long uptimes. On average the uptime length from the model is 17% higher than in the station data.
- There is no problem with underestimating uptime length, which can be seen by the lower uptime percentiles. Only the lowest 5% of the data seems to underestimte by any noteworthy amount.
- As for the length of the generated gaps, the model underestimates them a little bit, by about 5%.

Discussion

- By pooling all our stations into one big dataset, we make the assumption that the length of uptimes/gaps from all our training stations can be pooled into one big data set, or in other words that they all follow the same distribution and that their data can be pooled.This assumption works moderately well, but by visually looking at timeseries of individual stations some of them seem more ”problematic” than others, having many short gaps in short successsion. So perhaps it could be wise to split stations into different categories, perhaps by some clustering method.
- It might be possible to model the length of gaps/uptimes with a distribution, but I had no luck in doing this (after trying the most common distributions: exp, normal, GEV, gumbel, gamma…)
- In theory, this method works for any kind of process that jumps between ”states”. I think it could be used to model precipitation as well (rain/no rain), though perhaps the fact that the model does not take the length of the previous gap/uptime into account could be a problem here..
- Also note that although we used 2 states here (gap/uptime) we could have more states, and successively place them one at a time in the output mask (so first place a section of state 0, then 1, then 2, etc. until we come back to state 0)

Some final words

- If you want to get in touch with me: johan.sodling@smhi.se
- Thanks to Erik, Magnus och Christophe in my group, who have helped me a lot with all this.
- I skipped a few details here, for example analysing independence between gap length and the length of the previous gap (which is relevant since the model has no memory, and we need to know that this is a reasonable assumption). The short story here is that there does not seem to be any dependence.
- To my knowledge, no work has been done in this field (”gap making”) before – but if you know of anything, please mail me about it! Neither I, nor any of my colleagues, have found any previous work in this field.
- I am writing a paper on this right now, and many of the figures and results are of course from this paper. Right now the paper is a ”work in progress”.
- On the whole I think the model is interesting. It is a first step, with some areas for improvements. It would be very interesting to see if anyone else wants to do anything on gap making.

That’s it, thank you!