1. Intro
2. adapt.pp.input and the structure of s3 and s5 commands
3. Section 3: pre-downscaling adjustment
4. Section 5: Post-downscaling adjustment

1.Intro

In the current FUDGE workflow, there are several steps that are not downscaling methods which are capable of changing the values of the input and output data. These are currently called section 3 and section 5 adjustments, from the bubbles located on the FUDGE schematic. Although these steps are not the downscaling method itself, it is important to document their workings and make sure that their behavior in the workflow is clearly understood, since these functions can have a significant effect on the downscaled output.

Looking at the FUDGE schematic, you will notice that there are two calls each to **callS3Adjustment** and **callS5Adjustment**: one outside of TrainDriver, and one within it. This allows the adjustment functions to make comparisons within the time windows, only comparing datasets that are similar to each other. A day that gets flagged as a temperature outlier in January, for example, might be a perfectly normal temperature for June. The point at which the s3 and s5 modifications are applied, either during LoopByTimeWindow or within the main driver program, is distinguished by the 'loc' parameter of the input instructions: if it is set to 'inloop', the adjustment takes place within the downscaling loops; if set to 'outloop', the adjustments take place outside of TrainDriver and LoopByTimeWindow. Since the loc parameter is currently not present in the XML, there is a function in Drivers/Utility_functions, adapt.pp.input, designed to convert the input precipitation and mask arguments into a form that fits into the current FUDGE framework. As a result of the conversion, almost all section 3 and section 5 adjustments currently take place within the downscaling loops. The only exception are the section 3 and section 5 adjustments associated with precipitation, which take place outside of the downscaling loops. The behavior of adapt.pp.input, and the construction of the argument lists for section 3 and section 5, is dealt with in more detail below:

2. adapt.pp.input and the structure of s3 and s5 commands

Adapt.pp.input currently takes up to two arguments from the R runcode: the parameter mask.list and/or the parameter pr_opts.

```
mask.list <- list( mask1=list(type='SBiasCorr',
                              adjust.out='off',qc.mask='on',
                              qc_options=list(botlim=-6 , toplim=6)))

pr_opts=list(pr_threshold_in='us_trace',pr_freqadj_in='off',pr_conserve_in='on',
        pr_threshold_out='us_trace',pr_conserve_out='on', #'us_trace'
        apply_0_mask='off')
```

In general, the section 3 and section 5 instructions output by adapt.pp.input look fairly similar:

```
section_3_instruction <- list(type='PR', var='pr', apply='all', loc='outloop',
        pp.args=list(thold='us_trace',
                        freqadj='on',
                        conserve='on',
                        apply_0_mask='off'))
```

type: What is the name of the adjustment that is being used?
var: What variable is this adjustment meant to be applied over? Can become very important as we move towards multivariate precipitation transforms.
apply: Is this modification applied to all time periods ('all'), only the historic target and historic predictor, or just the future predictor? Currently exists, but is not used by any functions, though it may be useful later.
loc: Does this adjustment take place inside (inloop) or outside (outloop) of looping by time windowing mask?
pp.args : What are the arguments needed for the section 3 function itself?

```
section_5_instruction <- list(type='SBiasCorr',
        adjust.out='on',
        qc.mask='off',
        loc='outloop',
        qc_args=list(toplim=6,
                        botlim=-6))
```

type: What is the name of the adjustment that is being used?
adjust.out: Will this particular step adjust the downscaled output ('on') or pass it on without adjustment ('off')?
qc.mask: Will this adjustment produce a qc mask ('on') or not ('off') ? Note that there can only be one section 5 method for which this is true at the moment; the QC functions within the R code will exit with an error if that is not the case.
loc: Does this adjustment take place inside (inloop) or outside (outloop) of looping by time windowing mask?
qc.args: What are the arguments needed for the section 5 function itself?

As stated earlier, the 'loc' parameter is currently set by the adapt.pp.input function itself, but all other arguments are obtained from the list of arguments passed in from the runcode. Note that there is no adjust.out or qc.mask parameter in the instruction list passed into s3. It is assumed for the moment that s3 adjustments will always modify data, and never return a qc mask made up of values flagged as suspicious.

3.Section 3: Pre-downscaling

Section 3 consists of those operations that might take place on downscaled data before it is downscaled that involve more that one dataset. Although spatial and temporal masking are both considered to be part of section 3 on the schema, the spatial and temporal masking functions, called via apply.any.mask, are lightweight and simple enough to be considered standalone functions.

Calls to callS3Adjustment are made with 6 arguments:

```
temp.output <- callS3Adjustment(s3.instructions=pre.ds,
                        hist.pred = list.hist$clim.in,
                        hist.targ = list.target$clim.in,
                        fut.pred = list.fut$clim.in,,
                        att.list =list('hist.pred'=list.hist$att_table,
                                'hist.targ'=list.target$att_table,
                                'fut.pred'=list.fut$att_table),
                        s5.instructions=post.ds)
```

where s3.instructions consists of a list of instructions produced from adapt.pp.input, hist.pred, fut.pred and hist.targ are the historic predictor, future predictor and target datasets respectively, and

s3 is complicated because adjustments should be both callable in sequence and capable of altering section 5 instructions. In the case of precipitation, for example, it is easy to imagine a case where one would want to be able to first call the wetday masking function MaskPRSeries, to remove the drizzle bias from the precipitation datasets and afterwards call a precipitation transform to give the data a gaussian distribution. In that case, it would also be important to save the settings of the transform in order to apply a backtransform after the downscaling process is done, which would be part of section 5.

The assumption is that any methods added to the section 5 instructions as a result of operations taken in section 3 will be in the form of a stack: the most recent change to the dataset will be the first change reversed or modified by section 5. For example, if 3 commands were operated on in succession in section 3, the section 5 adjustments might look like this:

s3 operation 1
s3 operation 2

- Loop over all time windows
  s3 operation 3
  == downscaling operation
  s5 operation 3
- End loop over all time windows

s5 operation 2
s5 operation 1

Note also that the convention would be for individual adjustments added to take place in the same loop as the . For example, if a section 3 method resulted in a backtransfrom that should be applied in section 5, if the section 3 adjustment took place outside of the looping by time window, then the backtransform would take place outside of that loop as well.

Internal to the file section3.R, the s3 adjustment functions accept three arguments:
test : The name of the downscaling adjustment to be performed. Currently, the only pre-downscaling adjustment present is
input : a list of the input data and attribute table, described in more detail below
s5.list : a list of instructions to be carried out in section 5 of FUDGE, post-downscaling adjustment. These instructions can be modified by functions in s3, and arguments to those functions (such as the coefficients of a gaussian backtransform) may be generated in section 3.

In general, the parameters in input and s5.list may be modified and returned by the function, but test (and its parent object, s3.list), will remain unchanged while operations are taking place.

```
input=list('hist.pred' = hist.pred, 'hist.targ' = hist.targ, 'fut.pred' = fut.pred,
                        'att.list'=att.list)
```

hist.pred and fut.pred are named lists of variables carrying arrays of data; hist.targ is a single array of data consisting of the historical target.

Of these, a list consisting of input and postproc.output is returned to the main body of the function, to be iterated upon further.

```
adjusted.list <- list(input=list('hist.pred' = hist.pred, 'hist.targ' = hist.targ, 'fut.pred' = fut.pred,
                        'att.list'=att.list),
                    s5.list=s5.instructions)
```

General things to keep in mind:
- If the method being used is supposed to act upon a certain variable, it is possible to reference the name of that variable in the predictor datasets via names(hist.pred); if you need the name of the target variable, use names(input$att.list(hist.targ))
- There are currently few checks in place to keep s3 adjustments from being called on datasets that do not support them (such as precipitation adjustments being used on

non-precipitation datasets); this should change as more s3 methods are adapted and placed in the FUDGE workflow
- Be very careful with section 3 adjustment steps that result in the creation of extra missing values. If a s3 adjustment introduces extra missing values to a predictor dataset (such as the precipitation settings used to downscale only those days for which there was precipitation), then those missing values should be applied to all other predictor future predictor or historical predictor datasets. There are several subsequent places in the code that assume there are a predictable number of missing values in the code, such as assigning downscaled values by time window (see LoopByTimeWindow documentation for more details). In addition, some downscaling methods rely on having the same number of missing values present in the historical datasets in order to make sure that time series are coordinated (aka multi.lm); these methods can be thrown off by the addition of extra missing values.

4.Section 5: Post-downscaling

Section 5 is also the only place in the current FUDGE workflow where general QC takes place: flagging negative values and outliers, and adjusting those values if so desired. This is very important, since outlier correction can often have a significant effect on the downscaled result (such as the infamous section 2.5 paper).

Calls to callS5Adjustment are made with 6 arguments:

```
temp.postproc <- callS5Adjustment(post.ds,
                          data = ds$esd.final,
                          data.atts = list.target$att_table,
                          hist.pred = list.hist$clim.in,
                          hist.targ = list.target$clim.in,
                          fut.pred  = list.fut$clim.in)
```

Note the general similarity in structure to the section 3 arguments. Section 5 is also operating off of a list, though there are fewer assignment statements than there would be for the section 3, since only one dataset might get adjusted (though another qc dataset might be generated alongside the adjusted downscaled dataset). There is also no longer a need to add adjustment steps for later in the workflow; the instructions as they are should remain unchanged.

Internal to the file callS5Adjustment, there are two arguments provided to each function:
test : The name of the test to be performed.
input : The parameters that remain unchanged from adjustment to adjustment. Consists of the hist.pred, hist.targ and fut.pred datasets, plus the attribute table.
adjusted.output : The data being adjusted from section 5 adjustment to section 5 adjustment. Consists of two elements, **ds.out** (the downscaled data, plus any adjustments

made to it), and **qc.mask** (the mask of ones and NAs used to flag suspicious values in the downscaled output; if no mask is generated, is set to NULL)

In section 5, adjusted.output (consisting of the two elements outlined above) is the only thing returned from the section 5 adjustment function; it is also the only element returned from any individual section 5 function.