Estimation of Rainfall by Integrating a Satellite-Based Algorithm and Rain Gauge Networks

Qiao Ren

University College Twente

August, 2017

SUPERVISORS:

Dr.ir. R.L.G. (Rob) Lemmens

Dr.ir. C.M.M. (Chris) Mannaerts

Drs. B.J. (Barend) Köbben

CONTENTS

I. ABSTRACT 3
II. INTRODUCTION 3
III. STATE OF THE ART 4
IV. DATA DESCRIPTION 5
V. METHODOLOGY 6
VI. RESULTS AND DISCUSSION 11
VII. SUGGESTIONS FOR FUTURE WORK 16
VIII. CONCLUSION 18
REFERENCES 19
APPENDIX 21

I. ABSTRACT

The spatial and temporal distribution of precipitation is very important for scientific uses and human being's life [1]. The main goal of this study is to automate the process of producing precipitation estimates, based on gauge precipitation observation and satellite precipitation estimates. The scripts use ILWI-Python extensions (ilwis4). The algorithm applied in this study is Combined Scheme (CoSch) technique [1]. In order to generalize the script, two cases have been implemented: Latin America case and Africa case. The scripts and the precipitation estimation results produced in both cases are very useful for future work.

Keywords: combined scheme, additive bias correction, ratio bias correction

II. INTRODUCTION

The spatial and temporal distribution of precipitation at regional scale is widely needed for a variety of scientific uses such as water management [1] [3], precision agriculture [16], and forecasting floods and crops health [17]. A reliable estimation of spatial and temporal rainfall distribution, especially in the developing countries, is changing due to lack of dens rain gauge networks, satellite data spatial resolution, and the accuracy of the algorithm used to quantify the precipitation and derive the satellite based rainfall distribution [1]. The other main challenge is related to automation of the current modeling procedures for reparative implementation since daily or weekly rainfall distribution is required for monitoring floods, drought status or crops health [1]. In this study, we use ILWIS (The **I**ntegrated **L**and and **W**ater **I**nformation **S**ystem) [2] program to demonstrate the use the Hydroestimator" algorithm in combination with rain gauge networks of low density in order to retrieve precipitation and calculate rainfall statistics for any region of interest. The main objective is to provide to GEONETCast users with a Python script that automate data pre-prospecting, integration and post processing in order to generate near real-time information for daily rainfall distribution.

Currently, the Combined Scheme (CoSch) technique is implemented by the software ilwis 3 [3]. However, the efficiency of using ilwis 3 is not high enough. Users have to click the buttons or write code on their own to get their result, which takes them too much time. In order to solve this problem, a highly efficient way of implementing the Combined Scheme technique is expected. A newly developed software ilwis 4 (also called "ILWI-Python extension") is able to support this function. Different from ilwis 3, ilwis 4 is able to execute the python code.

The goal of this study is to develop scripts that automate and execute the Combined Scheme technique using ILWI-Python extensions.

It can be hypothesized the developed Python scripts (outcome of this study), will provide the efficiency and simplicity required for generating cost/benefit near real-time information for daily rainfall distribution.

III. STATE OF THE ART

Research review is based on two research questions: What are the advantages and disadvantage of gauge-based rainfall observation and satellite-based rainfall estimation? What is the best merging technique?

Merging satellite-based rainfall data and gauge-based rainfall observation provides higher accuracy than making rainfall estimation on either of them [3]. In all the reviewed studies, it is assumed that rain gauge observations have lower bias [3]. They will prevail over any satellite retrievals in those regions with dense networks [14]. However, over the ocean and non-well gauged areas, it is difficult to get accurate observed precipitation [3]. There are some methods to estimate the precipitation values over these areas. The most common methods are Arithmetic averaging method [4], Thiessen Method [5] and Isohyetal method [6]. They all perform errors to different extent [5] [9]. The errors are much larger than the merging technique [9]. Therefore, they are not used in this study.

The way of estimating precipitation by satellite also has both advantage and disadvantage [6] [13]. The first advantage is that satellites are able to capture a very large area, such as a continent [6]. The second advantage is that the satellite images are near-real-time; often within 15 to 30 minutes [6]. However, no satellite yet exists which can reliably identify rainfall and accurately estimate the rainfall rate in all circumstances [7] [12]. Firstly, clouds with “cold enough” temperature might do not produce rain, while clouds with “not cold enough” temperature might produce rain [13]. Secondly, rainy areas do not show up well, if the background is land [7] [8]. Thirdly, for a particular spot on the ground at a particular time, the images produced by the satellite are not much use [7] [9]. Therefore, satellite estimates need to be calibrated based on gauge-based precipitation observation [10] [12]. Table 1 shows the summary of the main characters of gauge-based rainfall observation and satellite-based rainfall estimation.

	Gauge-based rainfall observation	Satellite-based rainfall estimation
Accuracy	High	Low
Area Coverage	Small	Very Big

Table 1 comparison between gauge-based rainfall observation and satellite-based rainfall estimation on their reliability and area coverage

The second part of literature review is to find out the best merging technique to merge the precipitation estimation from gauges and from satellite. Combined scheme (CoSch) technique consistently presents the best performance, among all the five merging techniques [1]. Combined scheme technique combines two approaches (additive bias correction and ratio bias correction) into a single method to remove the bias of the satellite estimates [3]. If the precipitation value corrected by additive bias correction (ADD) is closer to the gauge-based observation than the value corrected by ratio bias correction (ADD), then the ADD-corrected rainfall value is chosen [1]. Vice versa. The tested merging techniques are additive bias correction; ratio bias correction; TRMM Multisatellite Precipitation Analysis, research version; and the combined scheme proposed in. These methodologies were tested for different months belonging to different seasons and for different network densities. The combined scheme technique appears to be a suitable tool to produce real-time, high-resolution, gauge- and satellite-based analyses of daily precipitation over land in regional domains [1] [7] [14] [15].

This study implemented the Combine Scheme (CoSch) technique in both Latin America case and Africa case.

IV. DATA DESCRIPTION

The main input data can be classified into three categories: RFS data, CPC data, and ROI data.

Category 1) RFS data: The rainfall satellite image is the result of transforming GOES East thermal images into precipitation by applying hydroestimator algorithm [3].

Category 2) CPC data: This is the data from NOAA’s Climate Prediction Center Unified Gauge-Based Analysis of Global Daily Precipitation [3]. For any given time period, 2 kinds of image are available:

-Gauge-based precipitation image: it gives the mean precipitation value of the gauges

-Gauge distribution image: it gives the number of rain gauges per pixel.

Category 3) ROI data: It means region of interest. It contains the major basins in a certain area.

The detailed information for the Latin America case and Africa is shown in table 2. The corresponding input images can be seen in appendix 1.

Data Description for Latin America case and Africa case				
Category of data			Latin America case	Africa case
Rainfall Satellite Data (RFS)		Temporal resolution	Daily	Pentad (or 5-day sums)
		Spatial resolution	4 km	Unknown
		Area Coverage	Latin America (exclude the southern parts of Chile and Argentina)	Africa
Data from NOAA’s Climate Prediction Center Unified Gauge-Based Analysis of Global Daily Precipitation (CPC)	CPC gauge-based precipitation image	Temporal resolution	Daily	Sum of 5 daily images
		Spatial resolution	0.5 degree	0.5 degree
		Area coverage	Global	Global
	CPC gauge distribution image	Spatial resolution	0.5 degree	0.5 degree
		Area coverage	Global	Global
Region Of Interest (ROI)		Area coverage	12 basins in Latin America	25 basins in Africa

Table 2 data description for Latin America case and Africa case

V. METHODOLOGY

The methodology of this study is based on the hydroestimator algorithm [10] and combined scheme technique [1]. Hydroestimator algorithm is used to derive the satellite based precipitation. Combined scheme technique combines two approaches into a single method to remove the bias of the satellite estimates [3]. The flowchart described in the study material [3] is shown in appendix 2. Because it is too brief, a series of detailed flowcharts are made. They elaborate the information that is omitted by the flowchart in the study material [3]. These elaborated flowcharts can be seen in figure 1,2,3,4,8 and appendix 6 (in case the figures are unclear to read).

The whole process is divided into four stages [figure 1].

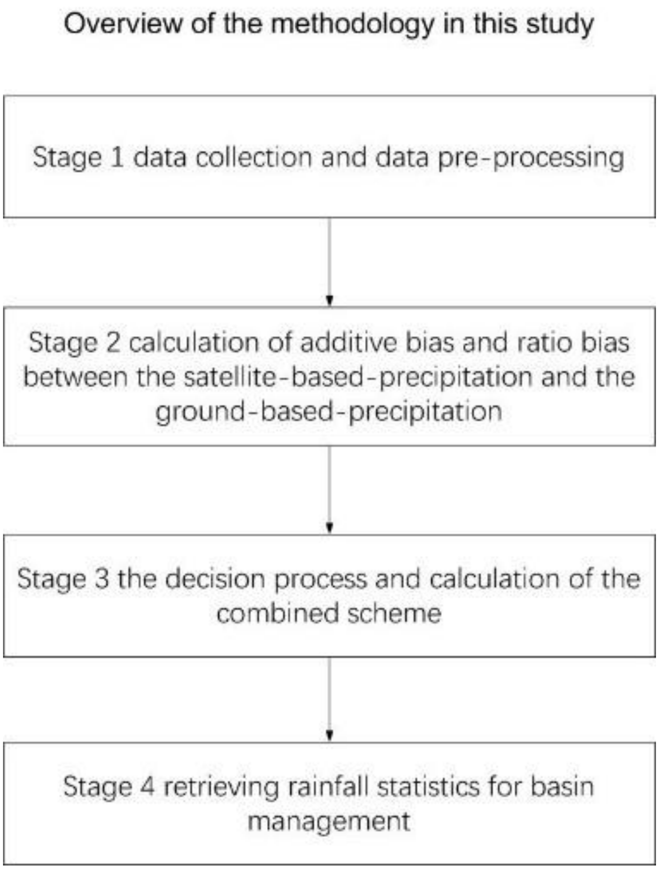


Figure 1 overview of the methodology in this study (based on the combined scheme algorithm)

The aim of stage 1 [figure 2] is data collection and data pre-processing. Three kinds of data (RFS, CPC and ROI) are collected as input. All maps have different boundaries and different resolution. The goal is to resample all the maps into the same resolution (In this case, 0.25 degree) and set the boundary as Latin America.

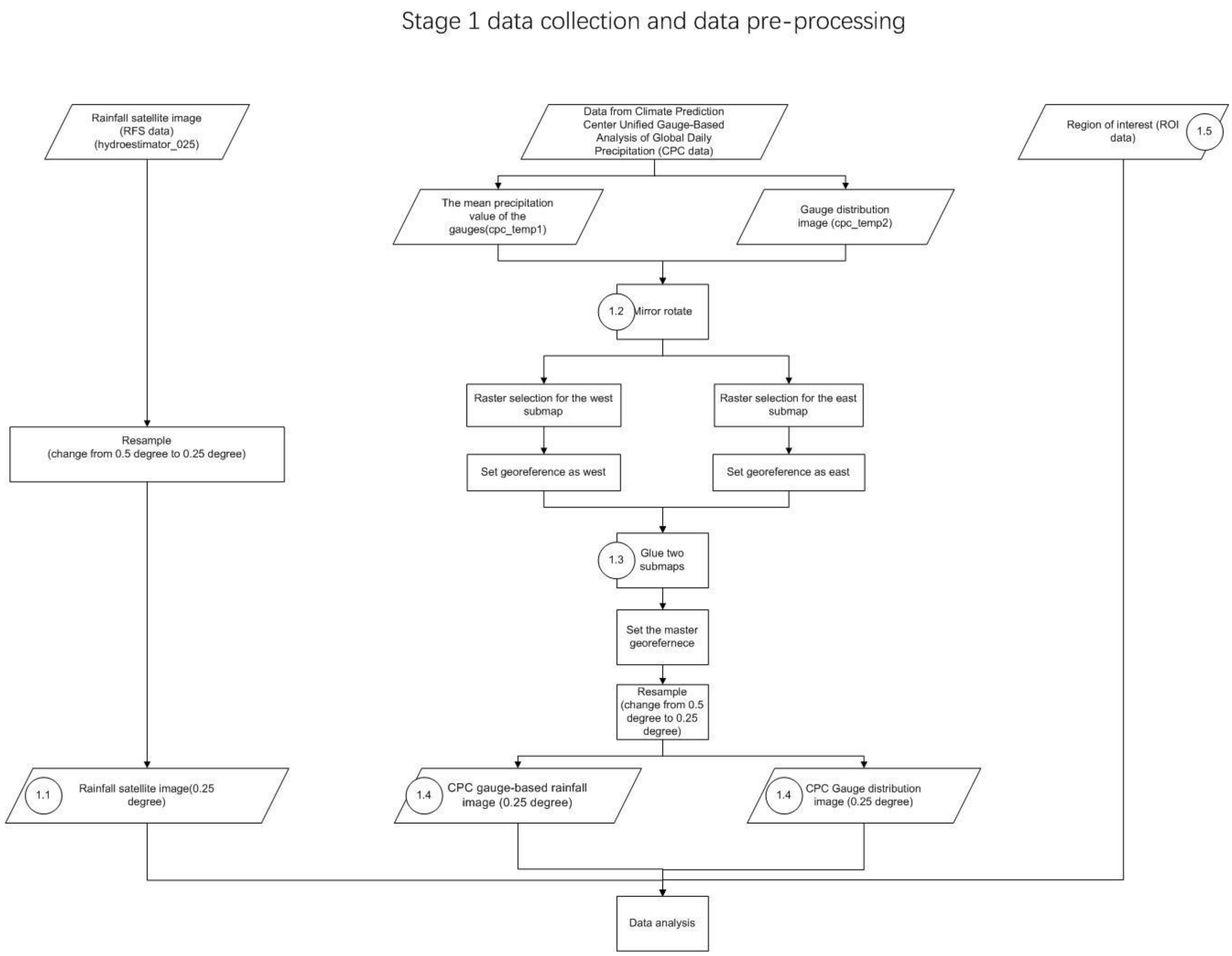


Figure 2 stage1 of the methodology in this study (based on the combined scheme algorithm)

In the stage 2 [figure 3], the pre-processed data (CPC and RFS) from stage 1 are in the input. The first step is to select and mask the precipitation values for the gauge-occupied pixels. This means that, in the CPC data, those pixels with no gauges are removed from analysis. Because no ground truth can be used to correct them [3]. The second step is to use two different mathematical methods separately to correct the satellite-based rainfall image. These two methods are additive bias correction (ADD) and ratio bias correction (RAT). Formulas of these two methods can be seen in the appendix 3. In this analysis, the masked gauge-based precipitation data is used as the observed rainfall.

Stage 2 calculation of additive bias and ratio bias between the satellite-based-precipitation and the ground-based-precipitation

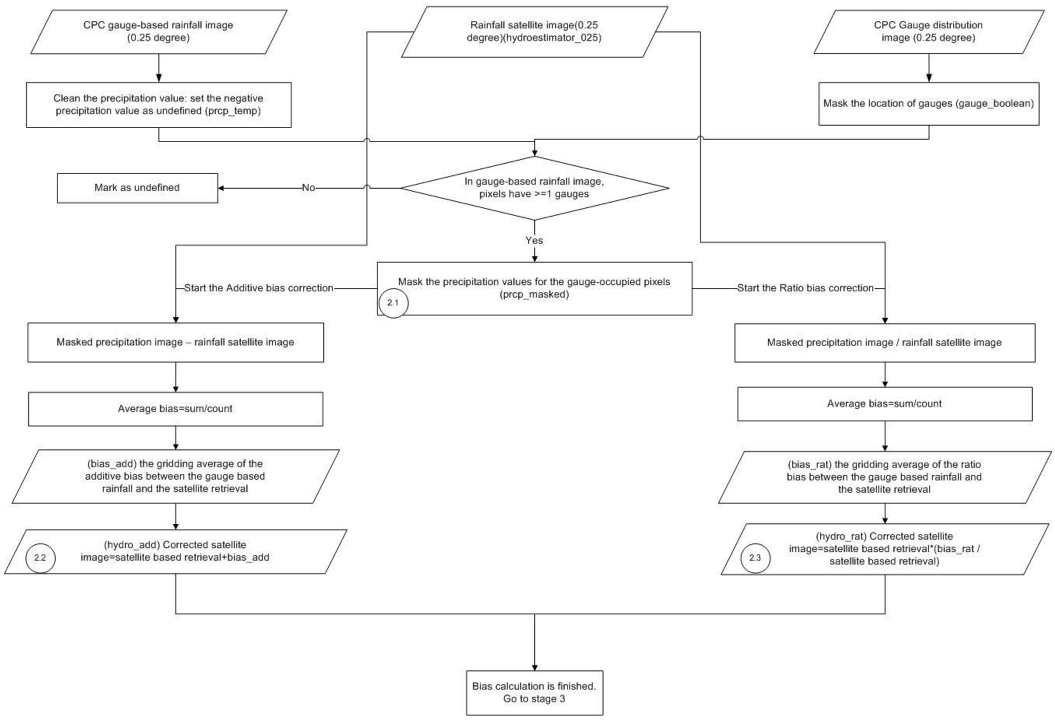


Figure 3 stage2 of the methodology in this study (based on the combined scheme algorithm)

The goal of stage 3 is to make a decision on that, among the ADD (additive bias correction)-corrected satellite precipitation value and the RAT (ratio bias correction)-corrected satellite precipitation value, which value is closer to the masked-gauge-based precipitation value. The closest value is the final precipitation value. This decision needs to be made for each pixel.

Stage 3 the decision process and calculation of the combined scheme

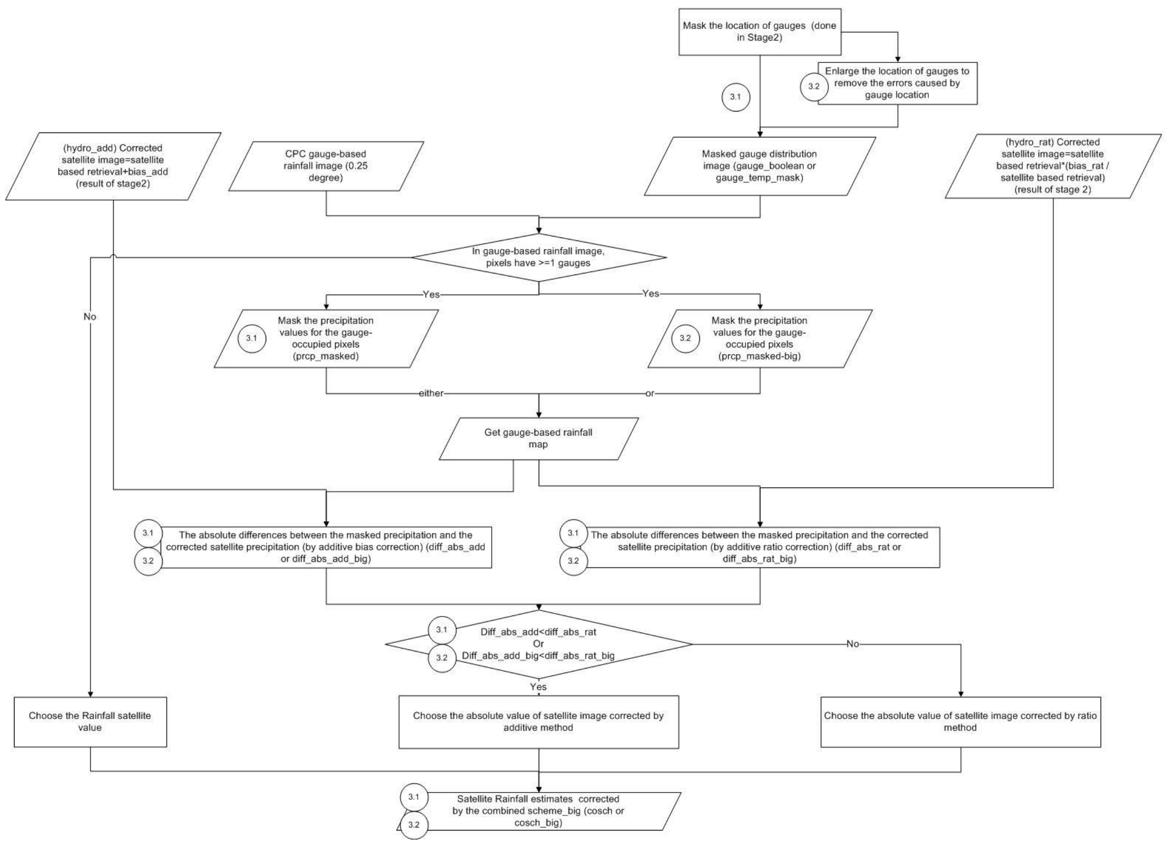


Figure 4 stage3 of the methodology in this study (based on the combined scheme algorithm)

The masked-gauge-based precipitation image is produced by two different ways. The first way has been done in stage 2. The second way need to be implemented in this stage. Because a gauge might be located on the edge of two or several pixels [figure 5 (b)], instead of luckily in the middle of a pixel [figure 5 (a)]. How large the resolution is has an influence on how to mark the gauge location [figure 5]. So errors could come out [figure 6]. In order to eliminate this kind of error, it is necessary to enlarge the masked gauge location [figure 7], which is implemented by a special filter. The combined-scheme-corrected precipitation map Therefore, two outputs ("cosch" and "cosch_big") are generated. "cosch" is denoted for the . "". They are produced after using different ways of masking gauge locations, separately. (however, it is unclear how cosch_big will be used)

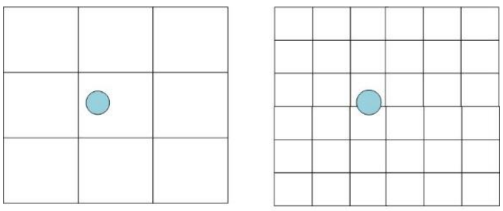


Figure (a) Figure (b)

Figure 5 how the resolution influences gauge-location detection

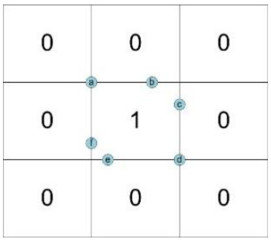


Figure 6 mask the gauge location without correction. (if there is a gauge in a pixel, the pixel is masked as “1”)

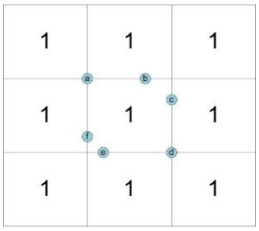


Figure 7 mask the gauge location with correction. (if there is a gauge in a pixel, the pixel is masked as “1”)

In the stage 4 [figure 5], the goal is to calculate rainfall statistics for the region of interest. This involves daily mean areal rainfall, maximum rainfall of the day, area where rainfall is larger than 1 mm, and daily mean areal rainfall considering only values are larger than 1 mm.

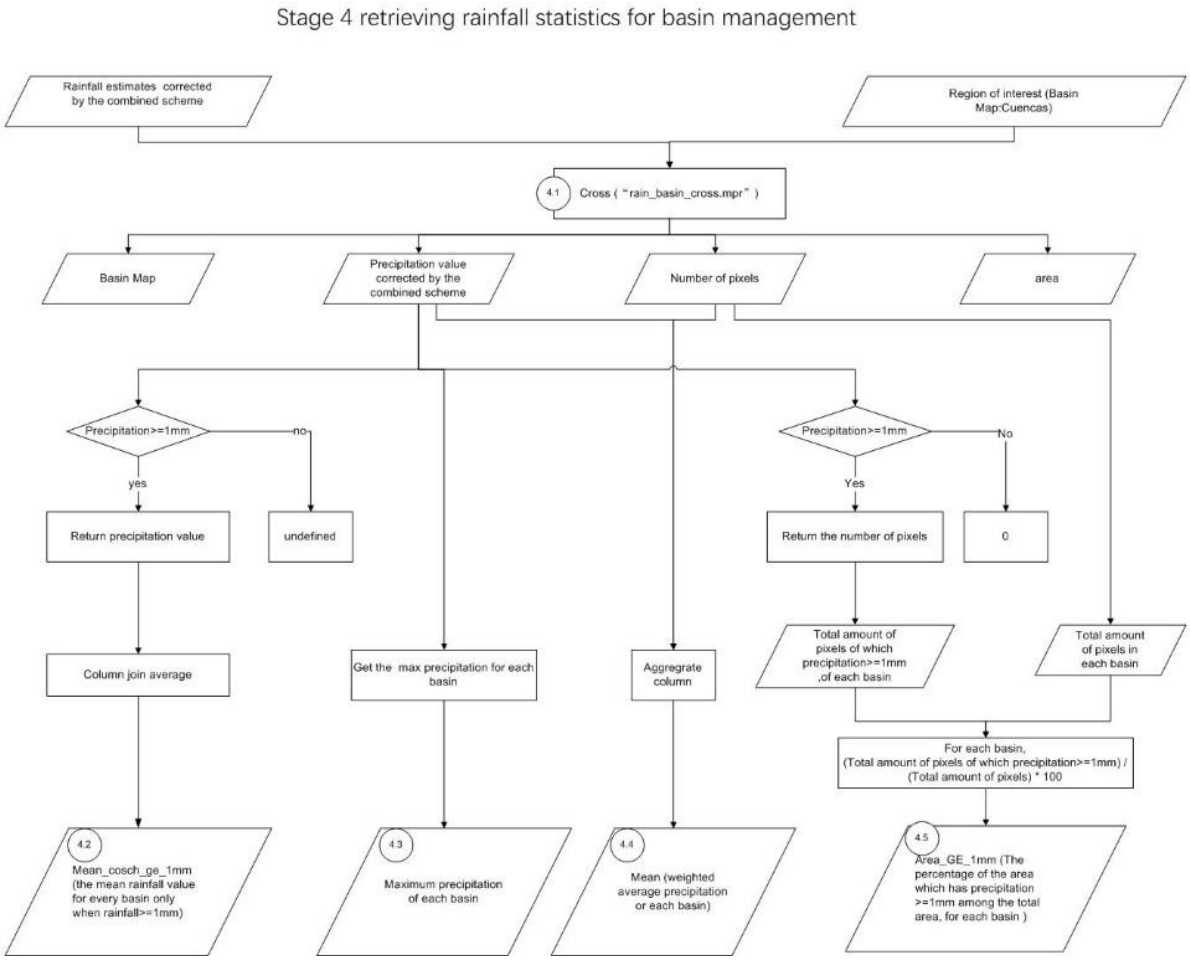


Figure 8 stage4 of the methodology in this study (based on the combined scheme algorithm)

The whole methodology is implemented for two cases: Latin America and Africa.

VI. RESULTS AND DISCUSSION

The result and evaluation of this research study is displayed in two parts, based on different cases. In each case, there are two kinds of result. The first part of the result is the satellite-based precipitation image which has been corrected by the combined scheme. The second part of the result is the statistics value for each basin.

A. Result and discussion in the Latin America case

The Latin America case is implemented by coding in Ilwis 4 python 3.5. This case comes from the Devcocast project. Figure 9 and 10 shows the intermediate results: satellite-based precipitation corrected by ADD and by RAT, respectively. Figure 11 and 12 show the final results: satellite rainfall estimates which has been corrected by the combined scheme without filter applied and with filter applied, respectively.

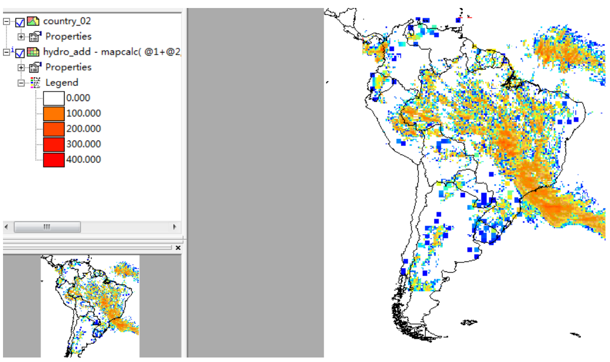
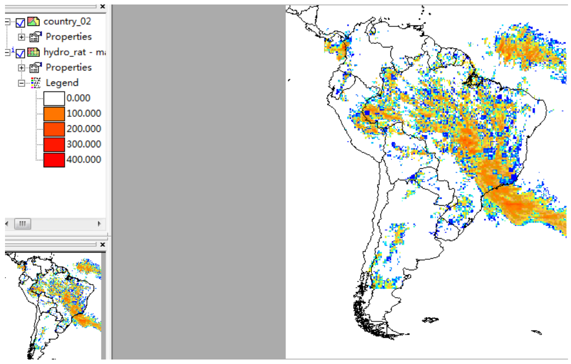
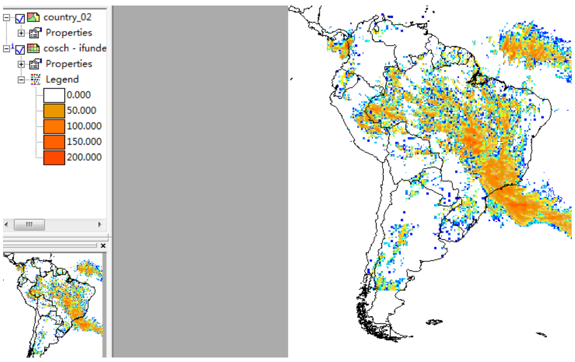
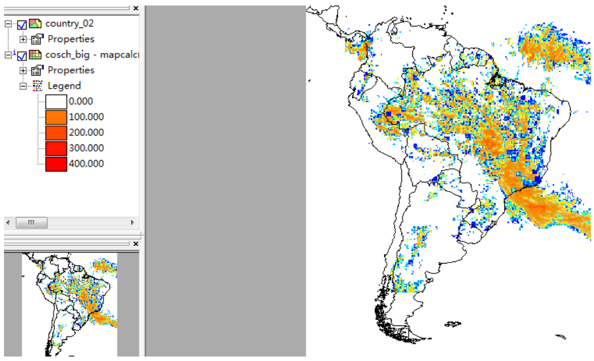
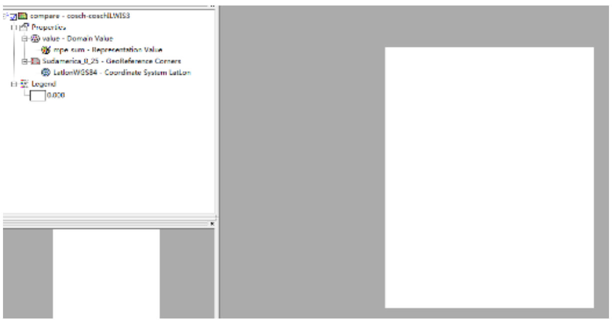
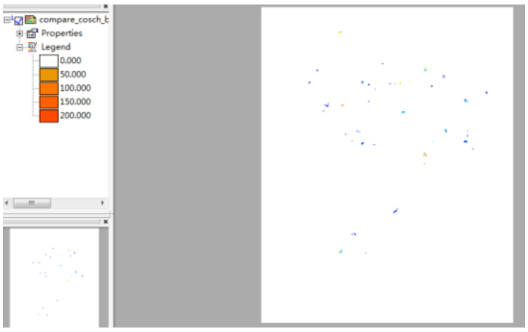
Figure 9 satellite-based precipitation corrected by ADD in Latin America case	Figure 10 satellite-based precipitation corrected by RAT in Latin America case
	

Figure 11 satellite rainfall estimates which has been corrected by the combined scheme without filter applied (cosch), in Latin America case	Figure 12 satellite rainfall estimates which has been corrected by the combined scheme with filter applied (cosch_big), in Latin America case
	

In order to evaluate the correctness of the results, it is necessary to compare them with the images which are produced in ilwis 3. The way of evaluation is subtraction. Figure 13 shows that the image "satellite rainfall estimates which has been corrected by the combined scheme without filter applied" is totally correct. Figure 12 shows that the image "satellite rainfall estimates which has been corrected by the combined scheme with filter applied" is slightly different from the image produced in ilwis 3. The reason is that ilwis 4 is not able to execute multiplication and division, neither separately nor at the same time. It has an influence on masking the gauge-occupied precipitation with filter, which is the following python command.

```
rc29=ilwis.Engine.do("mapcalc",'"@1*(@2/10)"', rc28, rc3) # multiplication and division does not work
rc29.store("prcp_masked_big", "map", "ilwis3")
```

Figure 13 the result of “cosch_ilwis4 – cosch_ilwis3”in Latin America case	Figure 14 the result of “cosch_big_ilwis4 – cosch_big_ilwis3” in Latin America case
	

The statistics value for each basin could not be produced by the current version of ilwis 4, because of the problems in ilwis 4. Fortunately, some intermediate process succeeded. The intermediate results are shown as bellow. Table 3 shows two columns that are produces by python. The column “cosch_GE_1mm” is an intermediate result in step 4.2 in the flowchart (Figure 8). The column “NPix_GE_1mm” is an intermediate result in step 4.5 in the flowchart (Figure 8). They are correct.

Table "tableCross_aaaaaaaaa" - TableCross(Cuencas.mpr,cosch.mpr,IgnoreUndefs) - ILWIS

File Edit Columns Records View Help

	NPix	cosch	Area	rain_basin_cross_forpython	coschGE1mm	NPixGE1mm
1	293	0.000	18.3	Magdalena * 0.0	?	0.000
2	1279	0.000	79.9	Orinoco * 0.0	?	0.000
3	1	1.772	0.1	Magdalena * 1.8	1.772	1.000
4	2	2.060	0.1	Orinoco * 2.1	2.060	2.000
5	1	2.576	0.1	Orinoco * 2.6	2.576	1.000
6	3	1.980	0.2	Orinoco * 2.0	1.980	3.000
7	1	2.444	0.1	Orinoco * 2.4 #4332	2.444	1.000
8	1	1.660	0.1	Orinoco * 1.7	1.660	1.000
9	1	2.362	0.1	Orinoco * 2.4	2.362	1.000
10	1	4.238	0.1	Orinoco * 4.2	4.238	1.000
11	2	5.580	0.1	Orinoco * 5.6	5.580	2.000
12	1	4.796	0.1	Orinoco * 4.8	4.796	1.000
13	4	0.420	0.3	Orinoco * 0.4	?	0.000
14	1	6.104	0.1	Orinoco * 6.1 #3563	6.104	1.000
15	3	8.200	0.2	Orinoco * 8.2	8.200	3.000
16	1	7.676	0.1	Orinoco * 7.7	7.676	1.000
17	1	2.771	0.1	Magdalena * 2.8	2.771	1.000
18	1	6.227	0.1	Orinoco * 6.2	6.227	1.000
19	1	8.211	0.1	Orinoco * 8.2 #3837	8.211	1.000
20	1	2.324	0.1	Orinoco * 2.3	2.324	1.000
21	1	4.809	0.1	Magdalena * 4.8	4.809	1.000
22	1	10.800	0.1	Magdalena * 10.8	10.800	1.000
23	1	12.930	0.1	Orinoco * 12.9	12.930	1.000
24	1	3.778	0.1	Magdalena * 3.8 #1191	3.778	1.000
25	2	3.770	0.1	Magdalena * 3.8	3.770	2.000
26	2	4.517	0.1	Magdalena * 4.5	4.517	2.000
27	2	5.263	0.1	Magdalena * 5.3	5.263	2.000
28	2	6.010	0.1	Magdalena * 6.0 #1691	6.010	2.000
29	4	1.300	0.3	Orinoco * 1.3	1.300	4.000
30	4	0.070	0.3	Magdalena * 0.1	?	0.000
Min	1	0.000	0.1		1.002	0.000
Max	3720	185.752	232.5		185.752	5.000
Avg	2	19.420	0.2		20.731	0.979
StD	55	21.032	3.5		21.127	0.363
Sum	17576	*******	1354.0		141176.712	7128.000

Table 3 column “cosch_GE_1mm” and column “NPix_GE_1mm” that are produced by ilwis4-python

The problems that hinder the python code execution are listed bellow. Detailed explanation is in the appendix 4. In stage 1, CPC data cannot be splitted or glued, due to the following two problems in ilwis 4. 1) Raster selection: Three different ways have been tried. None of them works because the ilwis-python has problems. 2) Glue two submaps.

In stage 3, ilwis 4 fails to mask the gauge-occupied precipitation value, due to the failure in image multiplication and image division. This function worked in the ilwis 4 which was released before June 2017. But in the ilwis 4 which released after June 2017, this function does not work anymore. According to the python guide, the syntax that should have been worked is: rc29=ilwis.Engine.do("mapcalc","'@1*(@2/10)'", rc28, rc3)

Most of the functional failures in ilwis 4 are in stage 4. They are listed below.

1) Table cross: table_cross = ilwis.Engine.do("cross", rc100 , rc33, "ignoreundef") the resulting table does not have columns: Cuencas, NPix, area, and cosch

2) Group by: tableStatistics=ilwis.Engine.do('groupby', tableCross, "Cuencas", max) The error message says “Please check the parameters provided”. Ilwis 4 is not able to create a table according to domain names.

3) Checking whether the value in a column equals to the value in the other column or not: iff(columnCuencas[rowIndex]==basinList[basinIndex]): The error message says “Please check the parameters provided.” This is a very basic operation. It is very necessary for quite a lot steps.

4) Store a cross table. tableCross.store("tableCross_aaaaaaaaa", "table", "ilwis3") this command works. but the orginal column "Cuencas" is missing. Secondly, the first column should be "basin name * value", instead of row number.

B. Results and discussion of the Africa Case

The original idea to do the Africa case is to test the code in the Latin America case and generalize it, so the code is able to work for all different kinds of cases. However, there are too much language problems in ilwis 4. Solving the language problems in ilwis 4 should be prior to generalizing the code. Therefore, a decision is made: Africa case should be implemented in ilwis 3, instead of ilwis 4.

The implementation of Africa case follows the same algorithm and the same process as in the Latin America case. The only difference is the time resolution. Arica case is in pentad (5-day sum) while Latin America case is in per day.

The results of the Africa case are a big achievement. They could be used in the Afrialliance project. Figure 15 shows the satellite-based precipitation which is corrected by ADD (additive bias correction). Figure 16 shows the satellite-based precipitation which is corrected by RAT (ratio bias correction). They are the intermediate result in the Africa case.

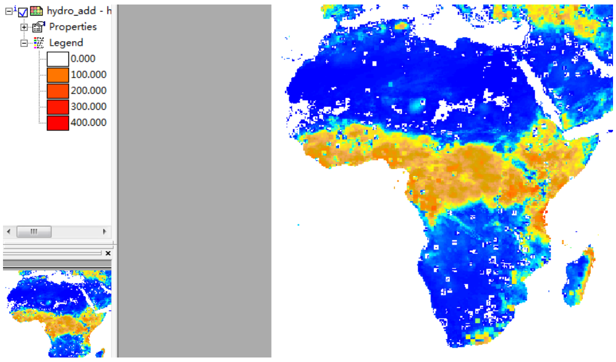
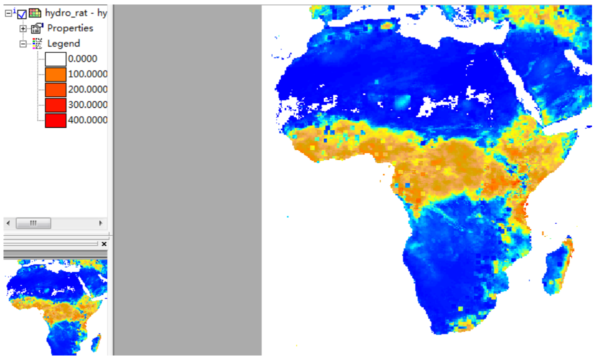
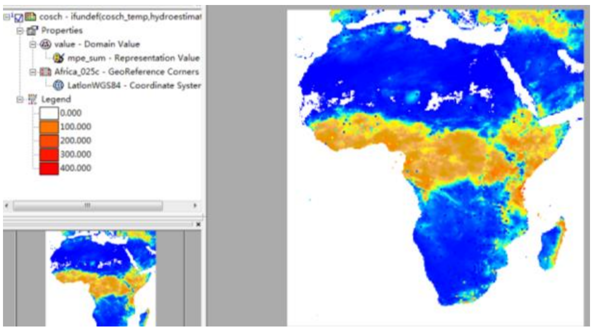
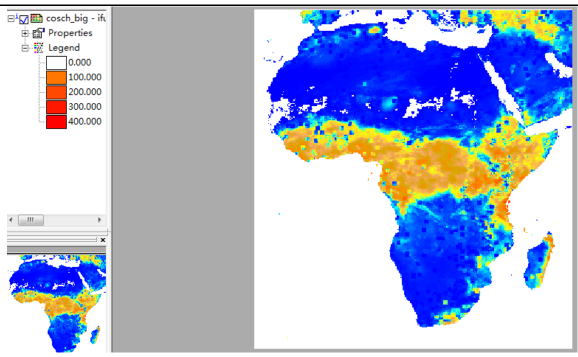
Satellite-based precipitation corrected by ADD, in the Africa case	Satellite-based precipitation corrected by RAT, in the Africa case
 Figure 15	 Figure 16

Figure 17 shows the Satellite rainfall estimates which has been corrected by the combined scheme without filter applied. Figure 18 shows the Satellite rainfall estimates which has been corrected by the combined scheme with filter applied.

Satellite rainfall estimates which has been corrected by the combined scheme without filter applied (cosch), in Africa case	Satellite rainfall estimates which has been corrected by the combined scheme with filter applied (cosch_big), in Africa case
 Figure 17	 Figure 18

The way to evaluate the correctness of cosch and cosch_big is unclear. Because it is impossible to compare the result from ilwis 3 and the result from ilwis 4, in this case. However, there are some findings when comparing the statistics value of “cosch” and “cosch_big”. Table 4, to some extents, indicates that the image “cosch” and the image “cosch_big” are reasonable. Because they have almost the same values in “Mean”, “Std.Dev” and “Median”.

Statistics comparison between Cosch (without filter) and Cosch_big (with filter) in the Africa Case		
	Cosch	Cosch_big
Mean	7.920	7.740
Std.Dev	14.350	14.640
Median	1.370	1.193

Table 4 Statistics comparison between “cosch” (without filter) and “cosch_big” (with filter) in Africa case

The statistics values for each basin are successfully calculated. The result is displayed in table 5.

Statistics of the Africa Case

MAJ_NAME	MEAN	MEANGE1MM	MAXIMUM	AREA_GE_1MM
Africa, East Central Coast	17.472045	18.007334	244.718063	96.94
Africa, Indian Ocean Coast	4.082019	4.479101	16.297643	89.80
Africa, North Interior	0.413799	1.934405	24.888346	10.27
Africa, North West Coast	0.647076	2.124052	6.542029	25.92
Africa, Red Sea - Gulf of Aden Coast	9.045799	11.521371	94.471497	77.96
Africa, South Interior	0.848201	1.549539	3.897462	30.11
Africa, West Coast	26.967134	30.997785	100.236694	86.94
Angola, Coast	2.854775	3.178310	34.690468	87.67
Congo	18.359983	19.292260	146.860594	94.97
Gulf of Guinea	32.046546	32.046546	103.812454	100.00
Lake Chad	3.166867	8.546683	55.569080	34.51
Limpopo	1.364725	2.092676	6.536041	51.55
Madasgacar	8.366974	9.271424	156.412903	89.55
Mediterranean South Coast	1.375005	3.233033	26.668756	33.07
Namibia, Coast	0.487304	1.336515	2.671052	4.96
Niger	10.902757	17.484648	73.396217	61.48
Nile	10.493799	17.884820	145.848456	58.01
Orange	0.719639	2.451876	8.974089	12.67
Rift Valley	17.101591	17.610655	106.059006	96.98
Senegal	2.109591	7.086404	33.709650	28.24
Shebelli - Juba	27.236863	27.313543	138.652486	99.71
South Africa, South Coast	3.952605	4.848354	41.797702	79.11
South Africa, West Coast	0.590634	1.727000	2.660919	11.26
Volta	20.664070	20.664070	60.829803	100.00
Zambezi	1.760678	2.516631	57.997898	59.40

Table 5 statistics of the Africa case. The first column contains all the basin names in Africa. The 2nd, 3rd, 4th and 5th columns are “MEAN”, “MEANGE1mm”, “MAXIMUM”, and “AREA_GE_1mm”

VII. SUGGESTIONS FOR FUTURE WORK

The first suggestion for future work is to solve the python problems in ilwis 4. This is very crucial and fundamental. Because of this problem, the speed of this study is relatively slow. A lot of time was spent on detecting the language problems in ilwis 4. There were a lot of communication with the programmers, since these problems are beyond my responsibility. Some basic language problems have been solved during this study, such as the “if-else” operation, the “undefined-setting” operation, the “georeference-setting” operation, and mirror rotate. However, due to the time limitation, it is impossible to debug all the language problems. If the problems described in section XII can be solved, this study will make another big achievement. Secondly, it would be better to complete the syntax description (http://ilwis.itc.utwente.nl/wiki/index.php/Main_Page:_Operations). Currently, most of the operations only has a name. Their syntax descriptions are missing. It is very important to publish operation descriptions to the public so people can easily access to the syntax.

The second suggestion is to make a user interface in ilwis 4. The interface could be designed as Figure 23. Text with blue background tells the user what input should be put for what purposes. The green button “execute” is used for executing the python code. Text with red background represents the output results that the user will get. A good user interface is much more convenient than copy-paste-and-customize the python code to the consoler. The user interface clearly tells the user what to do. It is highly-efficient.

The third suggestion is to add an unzip function in the python code, in stage 1 “data collection and preprocessing”. Because the CPC data is very likely a series of images, instead of a single image. This can be seen in the Africa case. One advantage in implementing the Africa case is to unzip the CPC data automatically. It saves a lot of time for the user.

The fourth suggestion is to design a time-test, which means to test how much time a customer will need to produce their final results (cosch, cosch_big, and the statistics table). The goal of this test is to find out whether or not the proposed automated mode in this study indeed improves the efficiency, compared with manual mode. A customer will be required to implement the same case on ilwis 3 and ilwis 4, respectively. The time will be counted. If the time spent in ilwis 4 is shorter than the time spent in ilwis 3, then the hypotheses of this study will be positive.

The fifth suggestion is to increase the amount of cases, especially some extreme cases. This is beneficial to test the quality of python code.

Finally, it is unclear in this study that how the image “Cosch_big” is used. This need to be done in future works.

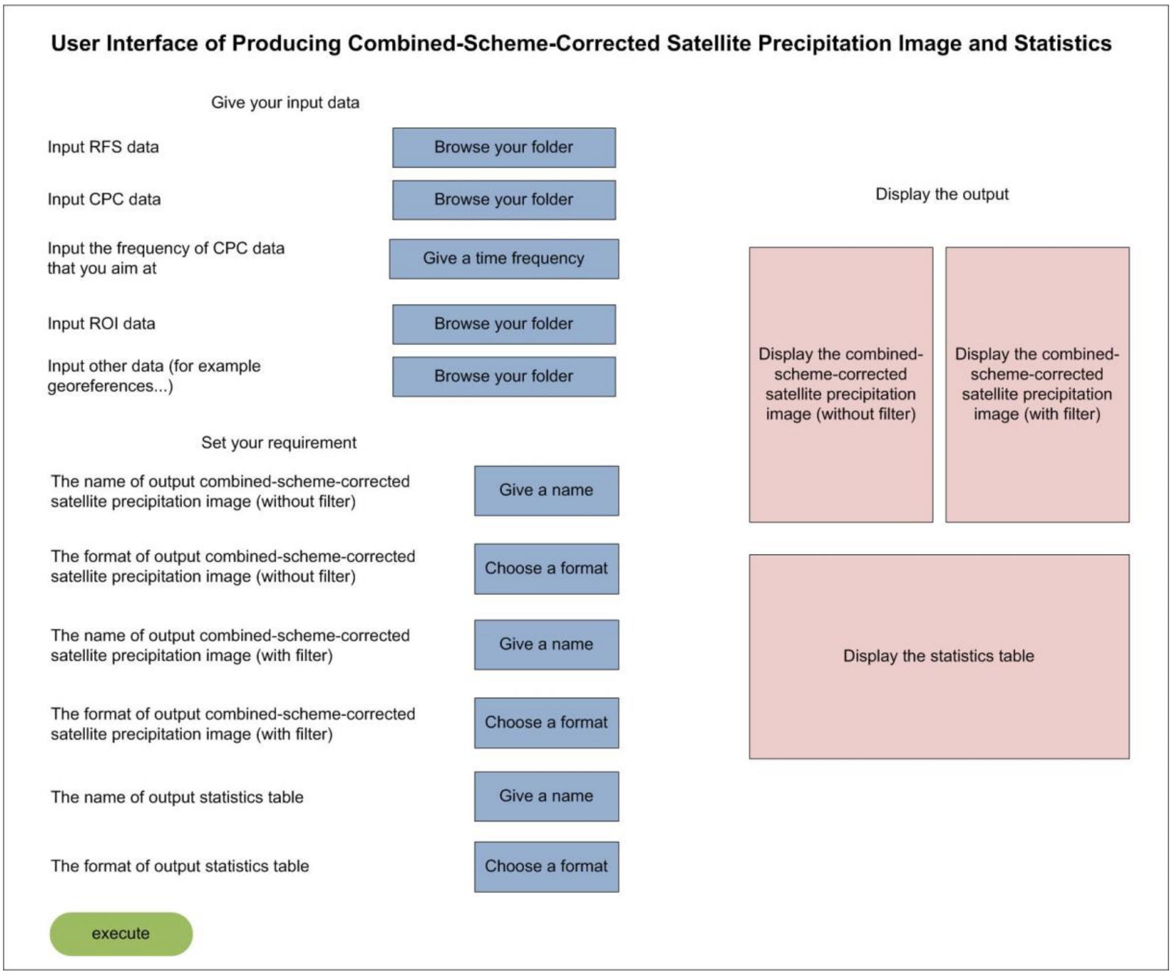


Figure 19 User Interface of Producing Combined-Scheme-Corrected Satellite Precipitation Image and Statistics

VIII. CONCLUSION

The main goal of this study is to develop scripts that automate and execute the Combined Scheme technique using ILWI-Python extensions (ilwis4). In order to generalize the script, the algorithm combined scheme is applied in two cases: Latin America case and Africa case. Latin America case is implemented by ilwis 4. Africa case should have been implemented by ilwis 4. Due to the language problems in ilwis 4, it is implemented by ilwis 3. In Latin America case, python script is able to correctly produce the combined-scheme-corrected satellite-based precipitation. However, the statistics cannot be completely generated. In the Africa case, both the combined-scheme-corrected satellite-based precipitation and the statistics has been produced successfully. To sum up, the scripts and the precipitation estimation images produced in both cases are very useful for future work. However, a lot of language problems in ilwis 4 bring limitations in this study.

REFERENCES

[1]. D. A. Vila, L. G. G. de Goncalves, D. L. Toll and J. R. Rozante (2009). Statistical Evaluation of Combined Daily Gauge Observations and Rainfall Satellite Estimates over Continental South America. American Meteorological Society, DOI: 10.1175/2008JHM1048.1 Available: http://cics.umd.edu/~dvila/web/CoSch/pdfs/CoSch.pdf

[2]. (2007). ITC’s GIS software ILWIS migrates to open source. [online] Available: https://web.archive.org/web/20070621134905/http://www.itc.nl/news_events/archive/research/0011.asp

[3]. D. Vila, C.L.Garcia. Chapter 11. Application of a combined daily rain gauges and rainfall satellite estimates scheme for basin management [Online] Available: https://www.itc.nl/Pub/WRS/WRS-GEONETCast/Application-manual.html

[4]. D.G. Simpson (2012). The Arithmetic-Geometric Mean. Department of Physical Sciences and Engineering Prince George’s Community College

[5]. (2013). Computation of Average Rainfall over a Basin. [Online] Available: http://www.yemenwater.org/wp-content/uploads/2013/03/computation-of-average-rainfall-over-a-basin.pdf

[6]. V.A. Tolpekin and A. Stein (2013). The core of GIScience: a systems-based approach. University of Twente Faculty of Geo-Information and Earth Observation (ITC). 524 p. ISBN 978-90-3653-719-3.

[7]. P. Ceccato, T. Dinku (2010). Introduction to Remote Sensing for Monitoring Rainfall, Temperature, Vegetation and Water Bodies. International Research Institute for Climate and Society Earth Institute at Columbia University

[8]. Scofield, R. A (1987). The NESDIS operational convective precipitation estimation technique. Mon. Wea. Rev., 115, 1773-1792.

[9]. Y. Jun Xu, R. Guo, Y. Liu (2016). Evaluation of Satellite Precipitation Products with Rain Gauge Data at Different Scales: Implications for Hydrological Applications. Laboratory of Watershed Geographic Sciences, Nanjing Institute of Geography and Limnology, Chinese Academy of Sciences, University of Chinese Academy of Sciences, No. 19

[10]. Vicente, G. A., R. A. Scofield, and W. P. Menzel (1998). The operational GOES infrared rainfall estimation technique. Bull. Amer. Meteor. Soc., **79**, 1883-1898.

[11]. Scofield, R. A., and R. J. Kuligowski (2003). Status and outlook of operational satellite precipitation algorithms for extreme-precipitation events. Wea.Forecasting, **18**, 1037-1051.

[12]. E. Ruprecht and W.M.Gary (1975) Analysis of satellite-observed tropical cloud clusters. ChapterII. Thermal, moisture and precipitation. Dept. of Atmospheric Science. Colorado State University, Fort Collins, Colorado, USA.

[13]. V. Levizzani, J. Schmetz, H. J. Lutz, J. Kerkmann, P. P. Alberoni and M. Cervino (2000). Precipitation estimations from geostationary orbit and prospects for METEOSAT Second Generation. Institute of Atmospheric and Oceanic Sciences-CNR, via Gobetti 101, I-40129 Bologna, Italy and Visiting Scientist at EUMETSAT, Am Kavalleriesand 31, D-64295 Darmstadt, Germany. ARPA-Regional Meteorological Service, viale Silvani 6, I-40122 Bologna, Italy and Institute

[14]. Dinku, T., Chidzambwa, S., Ceccato, P., Connor, S.J., Ropelewski, C.F. (2008). Validation of HighResolution Satellite Rainfall Products over Complex Terrain in Africa. *International Journal of Remote Sensing*, 29(14): 4097-4110

[15]. Dinku, T., Ceccato, P., Grover-Kopec, E.K., Lemma, M., Connor, S.J., Ropelewski, C.F. (2007). Validation of satellite rainfall products over East Africa's complex topography. *International Journal of Remote Sensing*, 28(7): 1503-1526

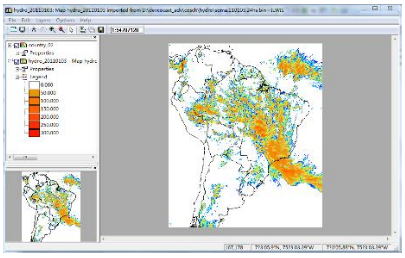
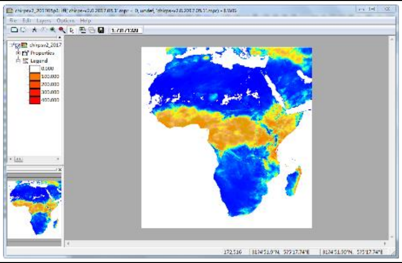
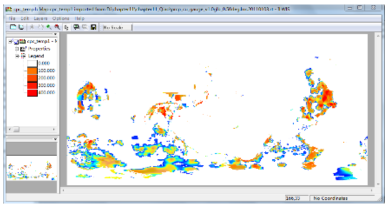
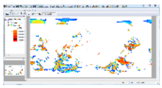
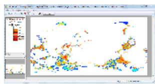
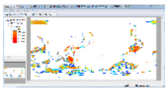
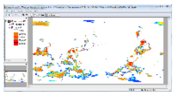
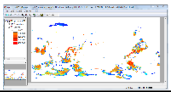
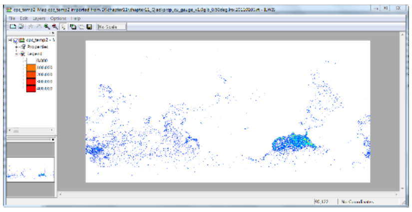
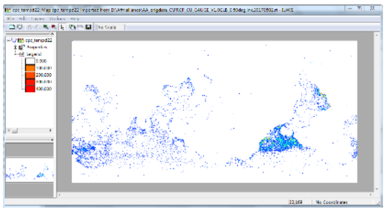
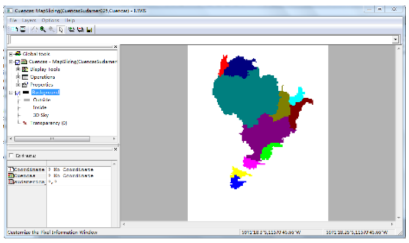
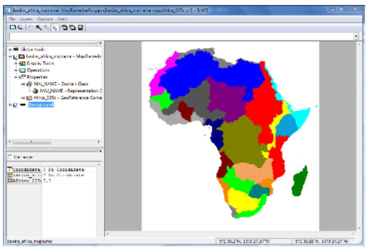
[16]. Ceccato, P. (2005). Operational Early Warning System Using SPOT-VGT and TERRA-MODIS to Predict Desert Locust Outbreaks. In Proceedings of the 2nd VEGETATION International Users Conference, 24-26 March 2004, Antwerpen. Editors: Veroustraete, F., Bartholome, E., Verstraeten, W.W. Luxembourg: Luxembourg: Office for Official Publication of the European Communities, ISBN 92-894-9004-7, EUR 21552 EN, 475 p

[17]. Ceccato, P., Flasse, S., and Gregoire, J.-M., (2002). Designing a spectral index to estimate vegetation water content from remote sensing data: Part 2: Validation and applications. *Remote Sensing of Environment* 82 (2-3): 198-207

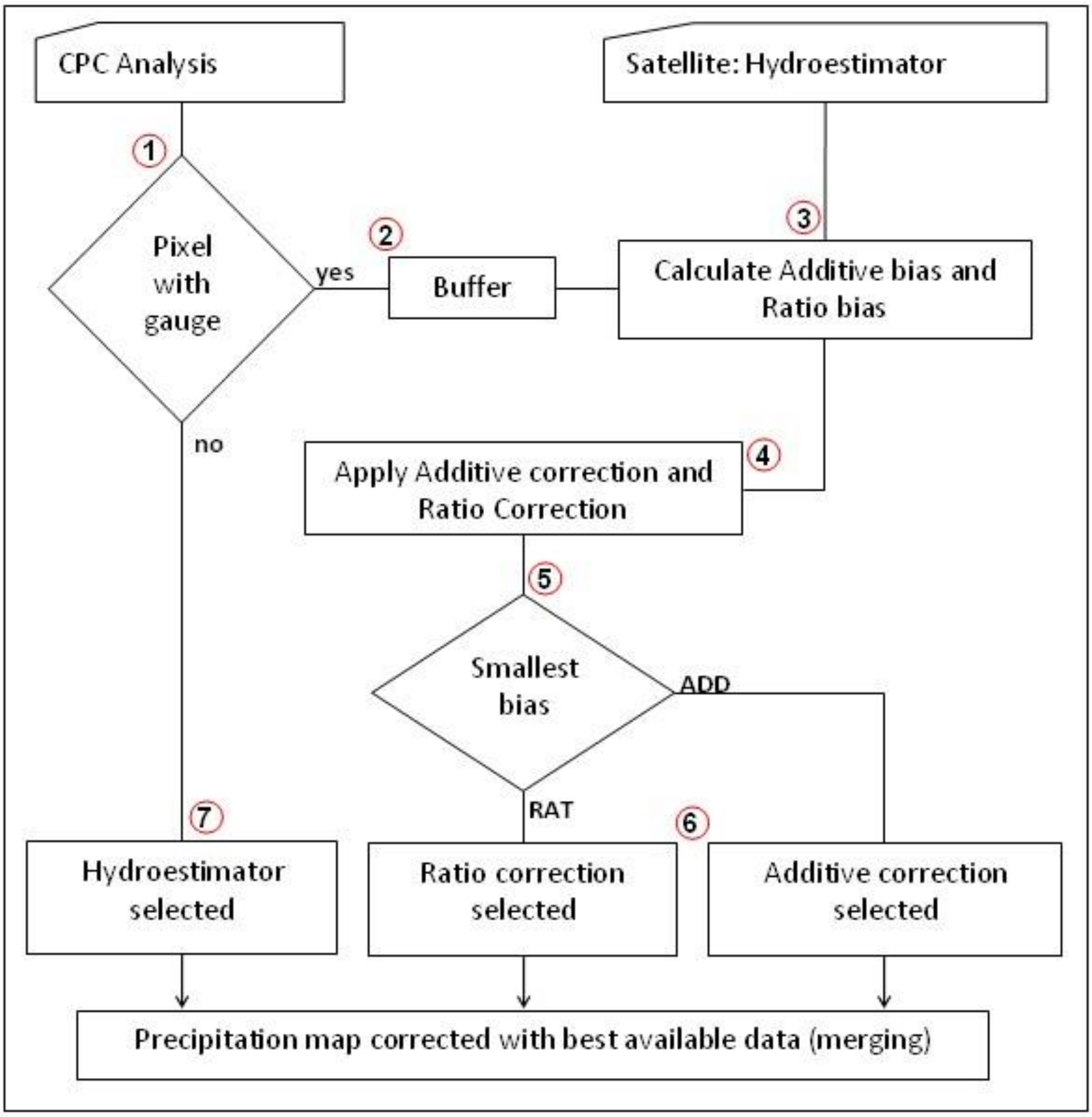
.

APPENDIX

Appendix 1. Input Data in Latin America Case and Africa Case

	Latin America case	Africa case
RFS data		
CPC gauge-based precipitation		5 images (put all of them here?)     
CPC gauge distribution image		
ROI data		

Appendix 2. Flowchart of Combined Scheme (CoSch) Algorithm described in the Study Material



Appendix 3. Formulas of Additive Bias Correction and Ratio Bias Correction

The additive bias correction (ADD) is defined as follows:

$$ADD = rr\_sat + \overline{(rr\_obs - rr\_sat)} \qquad \text{Eq. 1}$$

Where rr_sat is the satellite based retrieval and the second term represents the gridding average of the (additive) bias between the observed rainfall (rr_obs, CPC analysis) and the satellite retrieval (Hydroestimator, in this case) for each pixel group.

The ratio bias correction (RAT) is defined as follows:

$$RAT = rr\_sat * \overline{\left(\frac{rr\_obs}{rr\_sat}\right)} \qquad \text{Eq. 2}$$

Where the same conventions as those described for the additive bias correction are used.

Appendix 4. Python Script in ilwis v.4 in Latin America case

Stage 1 Data collection and pre-processing

Preprocess the rainfall-satellite data and rainfall-gauge data. Read region of interest data.

1.1 Resample the rainfall-satellite data

All the input data should have the same resolution. The geo-reference of south America is given, which is called “sudamerica_0_25.grf”

```
import ilwis
ilwis.Engine.setWorkingCatalog ("file:///D:/chapter11/chapter11_Qiao_ilwis4")
# read an input image
rcSatellite=ilwis.RasterCoverage("hydro_20110103.mpr")
#to check whether reading the map is successful or not
print(rcSatellite.name())
print(rcSatellite.geoReference().name())
#read an existing geo-refference
georef025 = ilwis.GeoReference("sudamerica_0_25.grf")
#resample: change the resolution to 0.25 degree
rc1 = ilwis.Engine.do("resample", rcSatellite, georef025, "nearestneighbour") #there are multiple ways to resample an image. In this project, the way "nearestneighbour" has been chosen.
```

```
rc1.store("hydroestimator_025Copy", "map", "ilwis3")
#check whether storing is successful or not
print(rc1.size().xsize, rc1.size().ysize )
```

1.2 horizontally rotate the two maps: the mean precipitation value of gauges (cpc_temp1) and the gauge-distribution image (cpc_temp2)

```
#read the mean precipitation value of gauges (cpc_temp1)
rc101=ilwis.RasterCoverage("cpc_temp1.mpr")
print(rc101.name())
print(rc101.size().xsize, rc101.size().ysize)
#read the gauge-distribution image (cpc_temp2)
rc102=ilwis.RasterCoverage("cpc_temp2.mpr")
print(rc102.name())
print(rc102.size().xsize, rc101.size().ysize)

#before rotating the images, it is necessary to set a correct georeference to the image. Otherwise, the compiler does not know how to rotate the image.
initialGeoref = ilwis.GeoReference("code=georef:type=corners,csy=epsg:4326,envelope= 0.0 -90.0 360.0 90.0,  gridsize=720 360, cornerofcorners=yes, name=initialGeoref")
rc101.setGeoReference(initialGeoref)
rc102.setGeoReference(initialGeoref)

#rotate the image the mean precipitation value of gauges (cpc_temp1)
rc104 = ilwis.Engine.do("mirrorrotateraster", rc101, "mirrhor")
rc104.setGeoReference(initialGeoref)
rc104.store("cpc_temp1_mirror", "map", "ilwis3")
print(rc104.name())
print(rc104.size().xsize, rc104.size().ysize)
print(rc104.geoReference().name())

# rotate the gauge-distribution image (cpc_temp2)
rc105 = ilwis.Engine.do("mirrorrotateraster", rc102, "mirrhor")
rc105.setGeoReference(initialGeoref)
rc105.store("cpc_temp2_mirror", "map", "ilwis3")
print(rc105.name())
print(rc105.size().xsize, rc105.size().ysize)
print(rc105.geoReference().name())
```

1.3 In order to set the correct geo-reference to the two gauge images (the mean precipitation value of gauges and the gauge-distribution image), it is necessary to split each of the image into two sub-maps: east sub-map and west sub-map. After that, glue them into one big map.

```
#create georeference: east
```

```
georefEast = ilwis.GeoReference("code=georef:type=corners,csy=epsg:4618,envelope= 0.0 90.0 180.0 90.0,    gridsize=360 360, cornerofcorners=yes, name=east")
# about the georeference definition, normally, epsg:4326 should be able to include the envelope. it does not make sense to define epsg and envelope at the same time
georefEast.store("east", "georeference", "stream")
print(georefEast.name())#succeed
print(georefEast.pixelSize())
print(georefEast.size().xsize, georefEast.size().ysize)

#create georeference: west
georefWest = ilwis.GeoReference("code=georef:type=corners,csy=epsg:4618,envelope= 180.0 90.0 0.0 90.0,    gridsize=360 360, cornerofcorners=yes, name=west")
georefWest.store("west", "georeference", "stream")
print(georefWest.name())#succeed
print(georefWest.pixelSize())
print(georefWest.size().xsize, georefWest.size().ysize)
```

Process the mean precipitation value of gauges (cpc_temp1): raster selection

```
# use raster selection to select which part in the map need to be split
# I made 3 tries.
# the syntax is (min x, min y, max x, max y). in a global map, does x goes from -180 to 180? or from 0 to 360? Or this is not fixed and I can define this in my own way?
# east to the Greenwich is negative, and south to the equator is negative. This is not written in the github tutorial
# the example from github is the following. It does not solve this problem:
#	create a new georeference
#	the following georeference is for a 1800x1380 grid, and the provided lat/lon bounds are assigned to the corners of this grid
#	georef = ilwis.GeoReference("code=georef:type=corners,csy=epsg:4326,envelope=32.991677775048 14.900003906339 47.991678557359 3.400003306568,gridsize=1800 1380,cornerofcorners=yes,name=ethnew")

# first try:
#rc_cpc_temp1_SelectedForEast = ilwis.Engine.do("selection", rc104, "boundingbox(361,-90,720,90)")
#rc_cpc_temp1_SelectedForWest = ilwis.Engine.do("selection", rc104, "boundingbox(0,-90,360,90)")
# both of the above two commands have the error: obj = Engine__do(str(out),str(operation),str(arg1),str(arg2),str(arg3),str(arg4),str(arg5),str(arg6),str(arg7)) Please check the parameters provided.

# second try:
# rc_cpc_temp1_SelectedForEast = ilwis.Engine.do("selection", rc104, "boundingbox(0,-90,180,90)")
```

```
# rc_cpc_temp1_SelectedForWest = ilwis.Engine.do("selection", rc104, "boundingbox(-180,-90,0,90)")
# both of the above two commands have the error: obj = Engine__do(str(out),str(operation),str(arg1),str(arg2),str(arg3),str(arg4),str(arg5),str(arg6),str(arg7))
Please check the parameters provided.

# third try:
# rc_cpc_temp1_SelectedForEast = ilwis.Engine.do("selection", rc104, "boundingbox(180,-90,360,90)")
# rc_cpc_temp1_SelectedForWest = ilwis.Engine.do("selection", rc104, "boundingbox(0,-90,180,90)")
# both of the above two commands have the error: obj = Engine__do(str(out),str(operation),str(arg1),str(arg2),str(arg3),str(arg4),str(arg5),str(arg6),str(arg7))
Please check the parameters provided.
```

Process the mean precipitation value of gauges (cpc_temp1): east georeferenced to the selected east and west submaps.

```
# set the east georeference to the east part of the map
rc_cpc_temp1_SelectedForEast.setGeoReference(georefEast)
rc_cpc_temp1_SelectedForEast.store("cpc_temp1_mirror_prepareForEast", "map", "ilwis3") #fail: readonly
print(rc_cpc_temp1_SelectedForEast.name())
print(rc_cpc_temp1_SelectedForEast.size().xsize, rc_cpc_temp1_SelectedForEast.size().ysize)
print(rc104.geoReference().name())

# set the west georeference to the west part of the map.
rc_cpc_temp1_SelectedForWest.setGeoReference(georefWest)
rc_cpc_temp1_SelectedForWest.store("cpc_temp1_mirror_prepareForWest", "map", "ilwis3")
print(rc_cpc_temp1_SelectedForWest.name())
print(rc_cpc_temp1_SelectedForWest.size().xsize, rc_cpc_temp1_SelectedForWest.size().ysize)
print(rc_cpc_temp1_SelectedForWest.geoReference().name())
```

Glue the east sub-map and the west sub-map into one big map.

```
The syntax of gluing is unknown.
glued_gauge_rainfall=…
```

Same process for the gauge-distribution image (cpc_temp2): raster selection, setting east and west geo-reference to two submaps, and finally glue.

1.4 Set the full geo-refference and resample

```
# read the existing georeference "full_WtoE.grf"
georefFullWtoE = ilwis.GeoReference("full_WtoE.grf")

# Set the full geo-reference on the mean precipitation value of gauges:
glued_gauge_rainfall.setGeoReference(georefFullWtoE)
# resample the mean precipitation value of gauges:
```

```
rc3 = ilwis.Engine.do("resample", glued_gauge_rainfall, georef025, "nearestneighbour")

# Set the full geo-reference on the gauge-distribution image
glued_gauge_distribution.setGeoReference(georefFullWtoE)
# resample the gauge-distribution image
rc2 = ilwis.Engine.do("resample", glued_gauge_distribution, georef025, "nearestneighbour")
```

Stage 2 calculation of additive bias and ratio bias between the satellite rainfall data and the ground based rainfall data

2.1 Mask the precipitation values for the gauge-occupied pixels

```
import ilwis
ilwis.Engine.setWorkingCatalog ("file:///D:/chapter11/chapter11_Qiao_ilwis4")

#Because, in stage 1, the raster selection does not work, and I need the result, I use the data made by
ilwis 3.7.2.

rc1=ilwis.RasterCoverage("hydroestimator_025.mpr")
print (rc1.name())

rc2=ilwis.RasterCoverage("gauges_20110103_025.mpr")
print (rc2.name())

rc3=ilwis.RasterCoverage("prcp_20110103_025.mpr")
print (rc3.name())

#Mask the location of gauges
rc2_1=ilwis.Engine.do("mapcalc","'iff(@1>=1,1,?)'",rc2)
#first problem:   fail in ilwis 4, invalid syntax. only works in python
#second problem: when I open the resulting map, the result is totally black.
#it shows both 0 and 1 as black. in its attribute table, the column "value" is missing.
#the undefined value should be shown as "?". But in the resulting map, it shows undefined value as 0

rc2_1.store("gauge_boolean","map","ilwis3")
print (rc2_1.name())
```

```
#Clean the precipitation value: set the negative precipitation value as undefined (prcp_temp)
rc3_1=ilwis.Engine.do("mapcalc","'iff(@1>=0,@1,?)'",rc3)
rc3_1.store("prcp_temp", "map", "ilwis3")
print (rc3_1.name())

#Mask the precipitation values for the gauge-occupied pixels
rc4=ilwis.Engine.do("mapcalc","'@1*(@2/10)'",rc2_1,rc3_1)
# succeded in the May-version of ilwis 4, but fail in the July-version of ilwis 4. I installed the updated version of ilwis 4 in June 13th.
rc4.store("prcp_masked", "map", "ilwis3")
print (rc4.name())
```

2.2 the additive bias correction

```
rc5=ilwis.Engine.do("mapcalc","'@2-@1'",rc1,rc4)
rc5.store("bias_add_temp", "map", "ilwis3")
print (rc5.name())

rc6=ilwis.Engine.do("mapcalc","'iff(@1==?,0,@1)'",rc5)
rc6.store("bias_add_for_sum", "map", "ilwis3")
print (rc6.name())

rc7 = ilwis.Engine.do('linearrasterfilter', rc6,"'code=1 1 1 1 1 1 1 1 1, 0.1111111'")
rc7.store("add_sum", "map", "ilwis3")
print (rc7.name())

rc8=ilwis.Engine.do("mapcalc","'iff(@1==0,0,1)'",rc6)
rc8.store("bias_add_for_count", "map", "ilwis3")
print (rc8.name())

rc9 = ilwis.Engine.do('linearrasterfilter', rc8,"'code=1 1 1 1 1 1 1 1 1, 0.1111111'")
rc9.store("add_count", "map", "ilwis3")
print (rc9.name())

rc10=ilwis.Engine.do("mapcalc","'@1/@2'",rc7, rc9)
rc10.store("bias_add_for_count", "map", "ilwis3")
print (rc10.name())

rc11=ilwis.Engine.do("mapcalc","'iff(@1==?,0,@1)'",rc10)
rc11.store("bias_add", "map", "ilwis3")
print (rc11.name())

rc12=ilwis.Engine.do("mapcalc","'@1+@2'",rc1, rc11)
```

```
rc12.store("hydro_add", "map", "ilwis3")
print (rc12.name())
```

2.3 the ratio bias correction

```
#to make "bias_rat_temp", the result has a problem, some points are missing!
# I want to find out the reason. But this piece of code got stuck the previous step: ("mapcalc","'@1*(@2/10)'",rc2_1,rc3_1)
# method 1
rc14=ilwis.Engine.do("mapcalc","'iff(@2>0,@2/@1,1)'",rc1,rc4)
rc14.store("bias_rat_temp", "map", "ilwis3")
print (rc14.name())
# method 2
#rc30=ilwis.Engine.do("mapcalc","'@2/@1'",rc1,rc4)
#rc14=ilwis.Engine.do("mapcalc","'iff(@1>0,@2,1)'",rc4,rc30)
#rc14.store("bias_rat_temp", "map", "ilwis3")


rc15=ilwis.Engine.do("mapcalc","'iff(@1==?,0,@1)'",rc14)
rc15.store("bias_rat_for_sum", "map", "ilwis3")
print (rc15.name())

rc16=ilwis.Engine.do('linearrasterfilter', rc15,'''code=1 1 1 1 1 1 1 1 1, 0.1111111''')
rc16.store("rat_sum", "map", "ilwis3")
print (rc16.name())

rc17=ilwis.Engine.do("mapcalc","'iff(@1==0,0,1)'",rc14)
rc17.store("bias_rat_for_count", "map", "ilwis3")
print (rc17.name())

rc18=ilwis.Engine.do('linearrasterfilter', rc17,'''code=1 1 1 1 1 1 1 1 1, 0.1111111''')
rc18.store("rat_count", "map", "ilwis3")
print (rc18.name())

rc19=ilwis.Engine.do("mapcalc","'@1/@2'",rc16,rc18)
rc19.store("average_rat", "map", "ilwis3")
print (rc19.name())

rc20=ilwis.Engine.do("mapcalc","'iff(@1==?,0,@1)'",rc19)
rc20.store("bias_rat", "map", "ilwis3")
print (rc20.name())

rc21=ilwis.Engine.do("mapcalc","'iff(@1>0,@1*@2,@1)'",rc1, rc20)
```

```
rc21.store("hydro_rat", "map", "ilwis3")
print (rc21.name())
```

Stage 3. the decision process and calculation of the combined scheme

3.1

```
# #calculate the difference between the masked precipitation and the corrected satellite precipitation (by additive bias correction)
rc13=ilwis.Engine.do("mapcalc",'''abs(@1-@2)''',rc4,rc12)
rc13.store("diff_abs_add", "map", "ilwis3")
print (rc13.name())

#calculate the difference between the masked precipitation and the corrected satellite precipitation (by ratio bias correction)
rc22=ilwis.Engine.do("mapcalc",'''abs(@1-@2)''',rc4,rc21)
rc22.store("diff_abs_rat", "map", "ilwis3")
print (rc22.name())

rc24=ilwis.Engine.do("mapcalc",'''abs(@1)''',rc12)
rc24.store("rc24", "map", "ilwis3")

rc25=ilwis.Engine.do("mapcalc",'''abs(@1)''',rc21)
rc25.store("rc25", "map", "ilwis3")

# if additive bias correction provides smaller error, choose the additive bias corrected rainfall. Vice versa.
rc26=ilwis.Engine.do("mapcalc",'''iff(@2<@4,@1,@3)''',rc24, rc13, rc25, rc22)
rc26.store("cosch_temp", "map", "ilwis3")
print (rc26.name())
rc13=ilwis.Engine.do("mapcalc",'''abs(@1-@2)''',rc4,rc12)
rc13.store("diff_abs_add", "map", "ilwis3")
print (rc13.name())

#calculate the difference between the masked precipitation and the corrected satellite precipitation (by ratio bias correction)
rc22=ilwis.Engine.do("mapcalc",'''abs(@1-@2)''',rc4,rc21)
rc22.store("diff_abs_rat", "map", "ilwis3")
print (rc22.name())

rc24=ilwis.Engine.do("mapcalc",'''abs(@1)''',rc12)
```

```
rc24.store("rc24", "map", "ilwis3")

rc25=ilwis.Engine.do("mapcalc",'''abs(@1)''',rc21)
rc25.store("rc25", "map", "ilwis3")

# if additive bias correction provides smaller error, choose the additive bias corrected rainfall. Vice versa.
rc26=ilwis.Engine.do("mapcalc",'''iff(@2<@4,@1,@3)''',rc24, rc13, rc25, rc22)
rc26.store("cosch_temp", "map", "ilwis3")
print (rc26.name())

#rc23=ilwis.Engine.do("mapcalc",'''iff(@2<@4,abs(@1),abs(@3))''',rc12, rc13, rc21, rc22)
#rc23.store("cosch_temp", "map", "ilwis3")
#print (rc21.name())

#for pixels on which no gauges are located (undefined pixels), choose the uncorrected(original) satellite data for them
rc27=ilwis.Engine.do("mapcalc",'''iff(@1==?,@2,@1)''',rc26, rc1)
rc27.store("cosch", "map", "ilwis3")
print (rc27.name())
```

3.2 Enlarge the location of gauges to remove the errors caused by gauge location big

```
rc28=ilwis.Engine.do('linearrasterfilter', rc2_1,'''code=9 9 9 9 9 9 9 9 9, 0.1111111''')
rc28.store("gauge_temp_mask", "map", "ilwis3")
print (rc28.name())


rc29=ilwis.Engine.do("mapcalc",'''@1*(@2/10)''', rc28, rc3)
rc29.store("prcp_masked_big", "map", "ilwis3")
print (rc29.name())


#calculate the difference between the masked precipitation and the corrected satellite precipitation (by additive bias correction)
rc30=ilwis.Engine.do("mapcalc",'''abs(@1-@2)''',rc29,rc12)
rc30.store("diff_abs_add_big", "map", "ilwis3")
print (rc30.name())


#calculate the difference between the masked precipitation and the corrected satellite precipitation (by ratio bias correction)
rc31=ilwis.Engine.do("mapcalc",'''abs(@1-@2)''',rc29,rc21)
```

```
rc31.store("diff_abs_rat_big", "map", "ilwis3")
print (rc31.name())

# if additive bias correction provides smaller error, choose the additive bias corrected rainfall. Vice versa.
rc32=ilwis.Engine.do("mapcalc","'iff(@2<@4,@1,@3)'",rc24, rc30, rc25, rc31)
rc32.store("cosch_temp_big", "map", "ilwis3")
print (rc32.name())

#for pixels on which no gauges are located (undefined pixels), choose the uncorrected(original) satellite data for them
rc33=ilwis.Engine.do("mapcalc","'iff(@1==?,@2,@1)'",rc32, rc1)
rc33.store("cosch_big", "map", "ilwis3")
print (rc33.name())
```

Stage 4 retrieving rainfall statistics for basin management

4.1 cross two maps

```
#read two maps
rc100=ilwis.RasterCoverage("Cuencas.mpr")
rc33=ilwis.RasterCoverage("cosch.mpr")
print(rc100.size().xsize, rc100.size().ysize)
print(rc33.size().xsize, rc33.size().ysize)
#cross two maps
table_cross = ilwis.Engine.do( "cross", rc100 , rc33, "ignoreundef")# syntax is correct
table_cross.store("cross_table_DoubleCheck", "table", "ilwis3")# the resulting table does not have columns: Cuencas, NPix, area, and cosch
# after executing cross, the Cuencas is changed to only one domain. but it should has 12 domains
# when I run the same code after one day, the resulting table cannot be produced.
# The error says "Failed to execute command "cross(_ILWISOBJECT_1004821,_ILWISOBJECT_1004824,ignoreundef)"; Please check the parameters provided."
```

Create a new table

```
#i want to create a new empty table and its first column should contain all the basin names. There are three ways.
# way 1) use 'groupby'
# tableStatistics=ilwis.Engine.do('groupby', tableCross, "Cuencas", max)
# fail. error: Please check the parameters provided.

# way 2) create a table by domain names
# i was told that ilwis4 does not support this function. but normally, this function should be able to work

# way 3)
tableStatistics=ilwis.Table("statisticsPython.tbt") #create a new table
print(tableStatistics.columnCount())
print(tableStatistics.name())
tableStatistics.addColumn("Cuencas", "String")



basinList=('Amazonas', 'Del Plata', 'Orinoco', 'Tocantins','San Francisco','Colorado','Uruguay','Parnaiba',
'Salado','Chubut','Negro','Magdalena')
basinTotalNumber=len(basinList)
print(basinList)
print(basinList[1])
print(basinTotalNumber)

for basinIndex in range (basinTotalNumber): #put the basin names into the Cuencas column of the table
    tableStatistics.setCell("Cuencas", basinIndex, basinList[1])
print(tableStatistics.columns())
```

4.2 calculate the mean rainfall value for every basin only when rainfall>=1mm (Cosch_ge_1mm)

```
#___________step2. add a new column to the table_cross:    value domain
tableCross=ilwis.Table("Rain_basin_cross_forPython.tbt")
print(tableCross.columnCount(), tableCross.recordCount())
print(tableCross.columns())
tableCross=ilwis.Engine.do("tabcalc","'iff(@1>=1,@1,?)'",tableCross,"coschGE1mm","cosch",False)
```

```
print(tableCross.columnCount(), tableCross.recordCount())
print(tableCross.columns())
```

4.3

```
# ___________step3. aggregrate by weighted average
totalRows = tableCross.recordCount()
print(totalRows)
columnCuencas = tableCross.column("Cuencas")
print(columnCuencas)
columnCosch = tableCross.column("cosch")
columnNPix = tableCross.column("NPix")
```

4.4 calculate the Mean (weighted average precipitation or each basin)

```
tableStatistics.addColumn("mean", "value")
Sum=0
for basinIndex in range (basinTotalNumber): # basin Index goes from the first basin name to the last basin name
    for rowIndex in range (totalRows): # row Index goes from the first row to the last row of the table
        iff(columnCuencas[rowIndex]==basinList[basinIndex])
        # fail: Please check the parameters provided. This is a basic operation. I also need it for the other steps
        Sum=Sum+columnCuencas[rowIndex]*cosch[rowIndex]
        totalWeight=totalWeight+cosch[rowIndex]
    meanForBasin = Sum/totalWeight
    tableStatistics.setCell("mean", basinIndex, meanForBasin)
```

4.5 Area_GE_1mm (The percentage of the area which has precipitation >=1mm among the total area, for each basin)

```
#_________________step 6    tabcalc rain_basin_cross.tbt NPix_GE_1mm:=iff(cosch ge 1,NPix,0)
tableCross.addColumn("NPixGE1mm", "value")
tableCross=ilwis.Engine.do("tabcalc","'iff(@1>=1,@2,0)'",tableCross,"NPixGE1mm","cosch","NPix",False)
tableCross.store("tableCross_aaaaaaaaa", "table", "ilwis3")
```

```
# this command works. but the original column "Cuencas" is missing. Secondly, the first column should be "basin name * value", instead of row number.

#________________step 9 and 8
#I have not created a tableTablaTemp yet
tableTablaTemp.addColumn("SumNPixGE1mm", "value")
tableTablaTemp.addColumn("SumNPix", "value")

#________________step 10
tableCross.addColumn("areaGE1mm", "value")
tableCross=ilwis.Engine.do("tabcalc","'(@1*100)/@2'",tableTablaTemp,"areaGE1mm","SumNPixGE1mm","SumNPix",False)
```

Appendix 5. ILWIS v.3 Script and Explanation in Africa case

rem copy scripts and rawdata and unzip
!cmd /c copy .\scripts*.zip"
!cmd /c copy .\rawdata*.*"
rem needs unzip program (here 7z)
!cmd /c d:\programs\7z\7z e Afri_biascorr_scripts.zip"
!cmd /c d:\programs\7z\7z e africa_hydrobasins.zip"
!cmd /c d:\programs\7z\7z e CPC_PRCP201705_5day.zip"
!cmd /c d:\programs\7z\7z e chirps-v2.0.2017.05.1.tif.gz"
del *.zip
del *.gz

closeall

Stage 1
rem Rainfall Bias Correction method script according CoSCH (Vila et al, 2009)
rem part 1- before please run copyandunzip.script to get data from rawdata storage
rem we use the same file names convention (as South America example) - for ease

rem autoimport Geotiff and reset nodata flag (-9999) to undef
rem this chirps data are pentad or 5-day sums (first pentad of May 2017)
open 'chirps-v2.0.2017.05.1'.tif -output='chirps-v2.0.2017.05.1'.mpr -method=GDAL -import
rem insert pause 2 seconds if ilwis on your laptop is fast (like on mine); the script continues and import is not ready

```
rem this method is implemented in ilwis so ilwis processes can continue on open (other) data
pause 8
chirpsv2_201705p1.mpr:=iff('chirps-v2.0.2017.05.1'.mpr < 0, undef, 'chirps-v2.0.2017.05.1'.mpr);

hydroestimator_025.mpr{dom=value.dom;vr=0.000:1493.670:0.000} :=
MapResample(chirpsv2_201705p1,Africa_025c.grf,nearest);

rem we need to import 5 days (here example of 01-05 May 2017) of cpc gauge files
cpc_tempd1 :=
maplist('PRCP_CU_GAUGE_V1.0GLB_0.50deg.lnx.20170501.RT',genras,Convert,720,2,0,BSQ,Real,4,
NoSwap,CreateMpr);
cpc_tempd2 :=
maplist('PRCP_CU_GAUGE_V1.0GLB_0.50deg.lnx.20170502.RT',genras,Convert,720,2,0,BSQ,Real,4,
NoSwap,CreateMpr);
cpc_tempd3 :=
maplist('PRCP_CU_GAUGE_V1.0GLB_0.50deg.lnx.20170503.RT',genras,Convert,720,2,0,BSQ,Real,4,
NoSwap,CreateMpr);
cpc_tempd4 :=
maplist('PRCP_CU_GAUGE_V1.0GLB_0.50deg.lnx.20170504.RT',genras,Convert,720,2,0,BSQ,Real,4,
NoSwap,CreateMpr);
cpc_tempd5 :=
maplist('PRCP_CU_GAUGE_V1.0GLB_0.50deg.lnx.20170505.RT',genras,Convert,720,2,0,BSQ,Real,4,
NoSwap,CreateMpr);

rem now total(sum) of the 5 days to   cpc_temp1 map value and cpc_temp2 gauge 5day sums
cpc_temp1.mpr:=cpc_tempd11+cpc_tempd21+cpc_tempd31+cpc_tempd41+cpc_tempd51
cpc_temp2.mpr:=cpc_tempd12+cpc_tempd22+cpc_tempd32+cpc_tempd42+cpc_tempd52
rem Mirror Rotate maps
cpc_temp1_mirror.mpr:= MapMirrorRotate(cpc_temp1,MirrHor)
cpc_temp2_mirror.mpr:= MapMirrorRotate(cpc_temp2,MirrHor)

cpc_temp1_east.mpr:= MapSubMap(cpc_temp1_mirror,1,1,360,360)
cpc_temp1_west.mpr:= MapSubMap(cpc_temp1_mirror,1,361,360,360)
cpc_temp2_east.mpr:= MapSubMap(cpc_temp2_mirror,1,1,360,360)
cpc_temp2_west.mpr:= MapSubMap(cpc_temp2_mirror,1,361,360,360)

crgrf west 360 360 -crdsys=latlonwgs84 -lowleft=(-180,-90) -upright=(0,90)
crgrf east 360 360 -crdsys=latlonwgs84 -lowleft=(0,-90) -upright=(180,90)

setgrf cpc_temp1_east.mpr east.grf
setgrf cpc_temp1_west.mpr west.grf
setgrf cpc_temp2_east.mpr east.grf
```

```
setgrf cpc_temp2_west.mpr west.grf

prcp_201705p1.mpr := MapGlue(full_WtoE.grf,cpc_temp1_east,cpc_temp1_west,replace)
gauges_201705p1.mpr := MapGlue(full_WtoE.grf,cpc_temp2_east,cpc_temp2_west,replace)

prcp_201705p1_025:=MapResample(prcp_201705p1,Africa_025c.grf,nearest)
gauges_201705p1_025:=MapResample(gauges_201705p1,Africa_025c.grf,nearest)

prcp_temp:=iff(prcp_201705p1_025.mpr ge 0,prcp_201705p1_025.mpr,?)

gauge_temp_boolean:=iff(gauges_201705p1_025 gt 0,1,?)

prcp_masked:=gauge_temp_boolean * (prcp_temp/10)

rem end of part 1

closeall
```

Stage 2

```
rem bias correction script part 2
rem compute additive and ratio bias

bias_add_temp:= prcp_masked-hydroestimator_025
bias_add_for_sum:=ifnotundef(bias_add_temp,bias_add_temp,0)
bias_add_for_count:=iff(bias_add_for_sum eq 0,0,1)

add_sum.mpr{dom=value;vr=::0.000000}:=MapFilter(bias_add_for_sum,sum3x3.fil,value)
add_count{dom=value;vr=::0.000000}:=MapFilter(bias_add_for_count,sum3x3.fil,value)

bias_add_mean_temp:=add_sum/add_count
bias_add:=ifnotundef(bias_add_mean_temp,bias_add_mean_temp,0)
hydro_add:=hydroestimator_025+bias_add


bias_rat_temp:=iff(hydroestimator_025 gt 0,prcp_masked/hydroestimator_025,1)
bias_rat_for_sum:=ifnotundef(bias_rat_temp,bias_rat_temp,0)
bias_rat_for_count:=iff(bias_rat_for_sum eq 0,0,1)

rat_sum.mpr{dom=value;vr=::0.000000}:=MapFilter(bias_rat_for_sum,sum3x3.fil,value)
rat_count{dom=value;vr=::0.000000}:=MapFilter(bias_rat_for_count,sum3x3.fil,value)

bias_rat_mean_temp:=rat_sum/rat_count
```

bias_rat:=ifnotundef(bias_rat_mean_temp,bias_rat_mean_temp,1)
hydro_rat:=hydroestimator_025*bias_rat

closeall

Stage 3
rem Rainfall Bias Correction Part 3
rem compute Combined Scheme (CoSCH) usinng 2 methods (2x2 0.5D station window or 3x3 px filter)

rem TWO OPTIONS, ACTIVATE ONLY ONE AT A TIME, OR KEEP THEM BOTH.
rem by default both will be activated: Cosh and Cosh_big

rem USE THIS IF YOU ONLY WANT TO CORRECT ON THOSE PIXELS WHERE THERE WAS A GAUGE 2X2 0,5deg

diff_abs_add:=abs(prcp_masked-hydro_add)
diff_abs_rat:=abs(prcp_masked-hydro_rat)

rem USE THIS IF YOU WANT TO KEEP THE 3X3 AREA FOR THE CORRECTION

gauge_temp_mask:=MapFilter(gauge_temp_boolean,RankOrder(3,3,9),value)
prcp_masked_big:=gauge_temp_mask * (prcp_temp/10)
diff_abs_add_big:=abs(prcp_masked_big-hydro_add)
diff_abs_rat_big:=abs(prcp_masked_big-hydro_rat)

rem *** cosh

cosch_temp:=iff(diff_abs_add ge diff_abs_rat,abs(hydro_rat),abs(hydro_add))

cosch:=ifundef(cosch_temp,hydroestimator_025)

rem *** cosh_big

cosch_temp_big:=iff(diff_abs_add_big ge diff_abs_rat_big,abs(hydro_rat),abs(hydro_add))

cosch_big:=ifundef(cosch_temp_big,hydroestimator_025)

rem Evaluate results (compare methods)

rem which_is_selected_1add_3rat:=iff(diff_abs_add ge diff_abs_rat, 1,3) rem diference_1eq_3dif:=iff(cosch eq hydroestimator_025, 1,3)

rem cosch_minus_hydro:=cosch - hydroestimator_025

closeall

Stage 4
rem Rain Bias Correction part 4
rem Apply CoSCH to Major African River basins and compute basin rainfall statistics
rem uses HydroSheds basin of Africa download e.g. ref. (c) FAO Geodataportal
rem Hydrosheds ESRI shape map was imported to Ilwis first, and set to attribute major basin name
rem copy basins_africa_majname.mpr + Maj_Name.dom and Maj_Name.rpr to this dir
rem two options: this uses CoSCH, for CoSCH_big (see other script)

rem Create a crosstabulation table of the Major river Basin map and Rainfall CoSch

Rain_basin_cross.tbt:= TableCross('basins_africa_majname',cosch,IgnoreUndefs)

rem create a new column containing the values of rain pixels >= 1mm (Greater or Equal).

tabcalc rain_basin_cross.tbt cosch_GE_1mm:=iff(cosch ge 1,cosch,?)

rem obtain the mean rainfall for every basin

crtbl statistics MAJ_NAME.dom

tabcalc statistics mean:=ColumnJoinAvg(Rain_basin_cross.tbt,cosch,'basins_africa_majname',NPix)

rem obtain the mean ranifall for every basin only when rainfall is GE than 1mm.

tabcalc statistics meanGE1mm:=ColumnJoinAvg(Rain_basin_cross.tbt,cosch_GE_1mm,'basins_africa_majname',NPix)

rem obtain maximun rainfall per basin

tabcalc statistics maximum:=ColumnJoinMax(Rain_basin_cross.tbt,cosch,'basins_africa_majname',1)

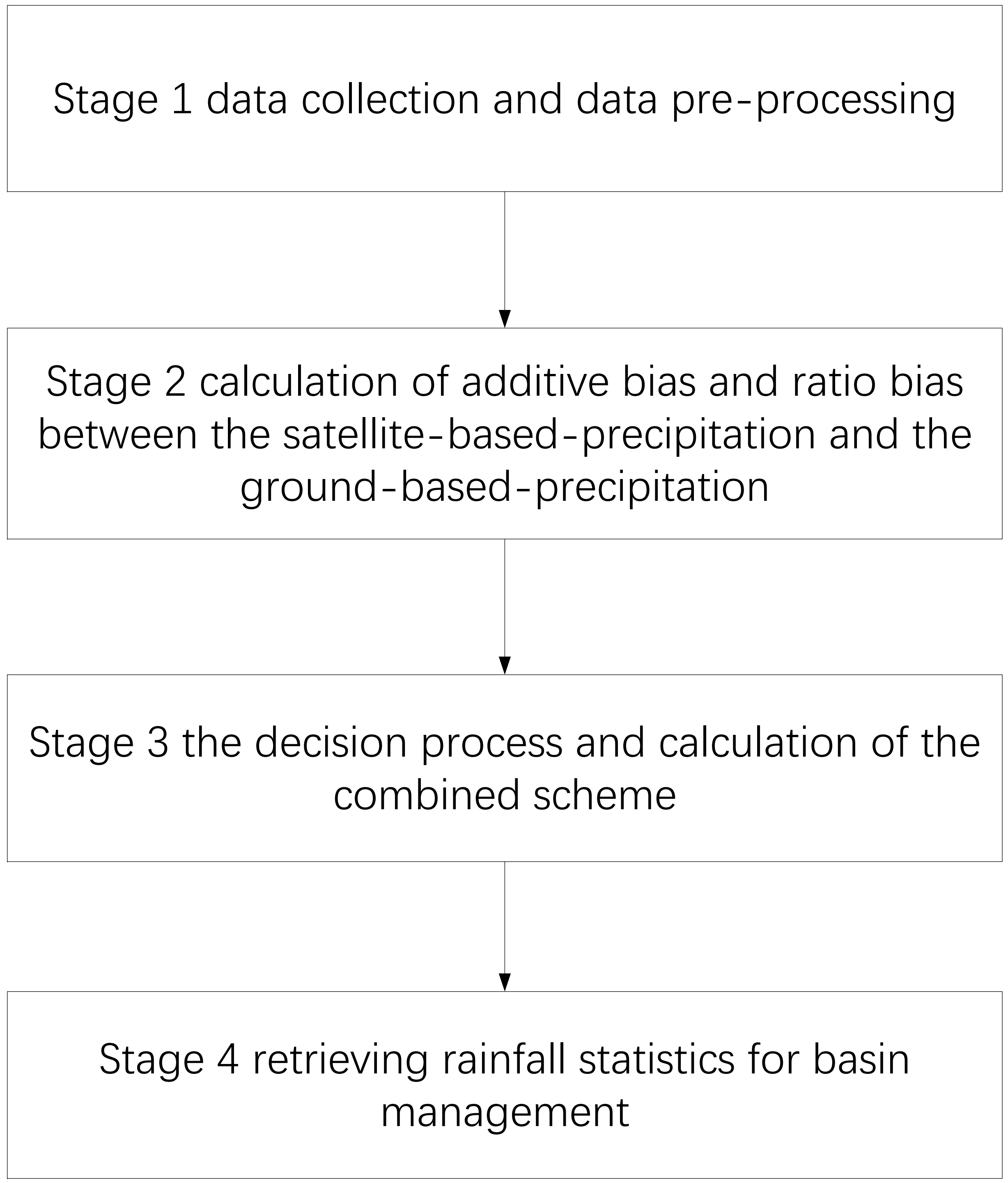
rem area with rainfall GE 1mm per Basin normalized

tabcalc rain_basin_cross.tbt NPix_GE_1mm:=iff(cosch ge 1,NPix,0)

```
crtbl table_temp MAJ_NAME.dom

tabcalc table_temp sum_NPix:=ColumnJoinSum(Rain_basin_cross.tbt,NPix,'basins_africa_majname',1)
tabcalc table_temp sum_NPix_GE_1mm:=ColumnJoinSum(Rain_basin_cross.tbt,NPix_GE_1mm,'basins_africa_majname',1)
tabcalc statistics area_GE_1mm:=table_temp.tbt.sum_NPix_GE_1mm *100 / table_temp.tbt.sum_NPix

# show Mean basin rainfall maps
Cosch_meanbasin_PCP_201705p1.mpr{dom=value.dom;vr=0.39:30.14:0.0} = MapAttribute(basins_africa_majname,statistics.tbt.mean)
show Cosch_meanbasin_PCP_201705p1.mpr -noask
pause

# show basin rainfall statistics table in excel
export dBase(statistics.tbt,statistics)
!'C:\Program Files\Microsoft Office\Office16\Excel.exe' statistics.dbf

closeall
```

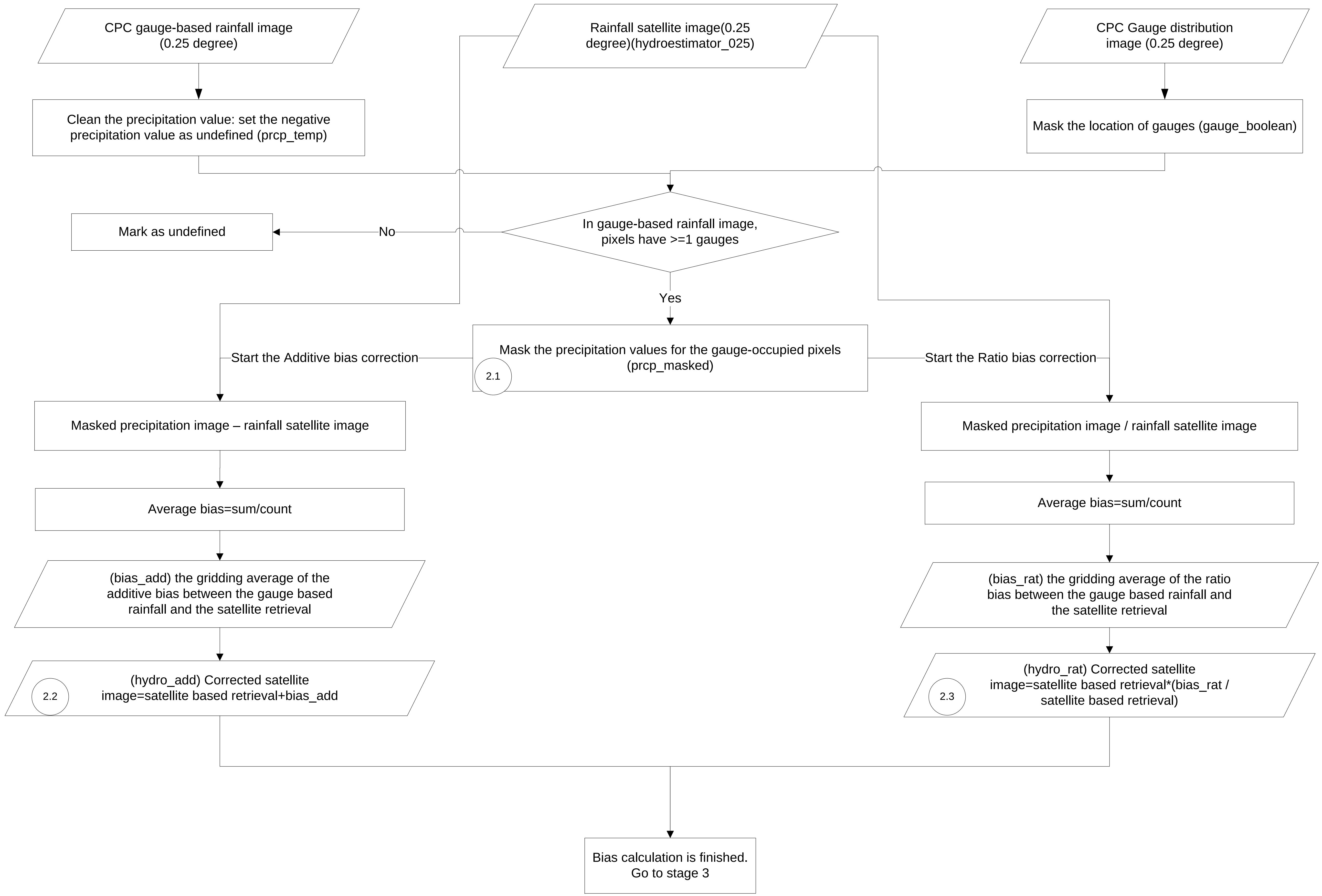
Appendix 6. the elaborated flowcharts in this study (based on the combined scheme algorithm)

Overview of the methodology in this study



Stage 1 data collection and data pre-processing

Stage 2 calculation of additive bias and ratio bias between the satellite-based-precipitation and the ground-based-precipitation



Stage 3 the decision process and calculation of the combined scheme

Stage 4 retrieving rainfall statistics for basin management