Geo-Localization of KNIME Downloads

as a static report and as a movie

Thorsten Meinl	Thorsten.Meinl@knime.com
Peter Ohl	Peter.Ohl@knime.com
Christian Dietz	Christian.Dietz@uni-konstanz.de
Martin Horn	Martin.Horn@uni-konstanz.de
Bernd Wiswedel	Bernd.Wiswedel@knime.com
Rosaria Silipo	Rosaria.Silipo@knime.com

Table of Contents

Geo-Localization of KNIME Downloads *as a static report and as a movie* 1

Summary 3

So much Information available in Web Log Files! 3

Monitoring the KNIME Downloads 4

- The Use Cases 4
- The Weblog Reader Node 4
- The Data 5

Geo-Localization of Downloads 6

- Data Preparation: the Rule based Row Filter Node 6
- Data Preparation: IP Addresses and World Regions 8
- The Open Street Map Package 9
- The interactive View with the OSM Map View Node 9
- The static Report with the OSM Map to Image Node 10

Monitoring Daily Downloads 13

- Data Preparation and Static Report 13
- The KNIME Image Processing Package 13

Workflow Remote Execution from the WebPortal 16

Conclusions 17

Summary

There is so much information hidden in the web log file of a company web site! In this particular study, we concentrate on the geolocalization of the IP addresses that download the KNIME open source data analytics platform. The goal is to get an idea of where most of the KNIME users are located in the world, to set up future community events.

First of all, we extract the download data from the Apache web log file of the KNIME web page. We restrict these data to the week around December 6th 2013, when the new version of KNIME was released. The hypothesis is that, during these days, frequent KNIME users would download or update to the latest version of the software. The data extracted from the web log file contain the IP addresses that connected to the KNIME web site for download or update.

IP addresses are correlated to geographical locations. After appending its latitude and longitude coordinates to each IP address, the KNIME Open Street Map integration is used to geolocalize the IP addresses on a world map.

The geo-localization of the IP addresses can be performed for all the 7 days or day by day on a map sequence. Such map sequence can then be translated into a movie by means of the KNIME Image Processing extension available from the KNIME Community.

Web log reading, geolocalization, and image processing are three very interesting and very common data analytics applications covered in this whitepaper. All workflows are available on the KNIME EXAMPLES public server in "008_WebAnalytics_and_OpenStreetMap" and the KNIME software can be downloaded from www.knime.com.

So much Information available in Web Log Files!

So much of the company work goes into building and maintain the company's web site, with news, reports, whitepapers, and all sorts of other contents. The basic web analytics, such as how many views, how many visitors, how many clicks, and similar numbers, is usually implemented and made available within the company by the analysts or the web site host.

However, so much more information is still hidden in the web site web log file! If we could only read the file and parse it, we could know so much more about our web site visitors! The web log file indeed contains information about every single click and every single request for every single visitor of the web site.

The same, of course, is true for the KNIME web site. In particular, since KNIME is open source and people are not forced to leave personal information when downloading the tool, we have very spotty knowledge about the KNIME users.

While we are not really interested in knowing much more about our users than what they are willing to share, it could still be interesting to know the geographical distribution of the KNIME user base. This could for example help in planning for KNIME community events around the world. Indeed, a full KNIME user day in a place where there are no KNIME users is clearly a waste of time and resources.

In order to get an idea about the geographical distribution of the KNIME users, we can turn again to the site web log file. There, together with time and type of request, the visitor IP address is also available. More specifically, we are interested in those visitors who have downloaded the KNIME installation file rather than in simple visitor of the web site.

IP addresses are just sequences of numbers. However, IP addresses can be converted into IP numbers and specific ranges of IP numbers have been assigned to specific geographical locations. The map associating IP number ranges and geographical locations can be easily found on the Internet. For example http://www.maxmind.com/en/city offers the location/IP address map both as a web service and as a csv file. We downloaded the csv file from this site and we use it in this work to get more insight about the KNIME user base geographical distribution.

Monitoring the KNIME Downloads

The analysis goal of this whitepaper is quite clear: to gather information about the geographical distribution of the KNIME user base. The next question is about the time representation of the KNIME downloads around the world.

The Use Cases

There is no doubt that we need to show the distribution of the KNIME tool downloads on a world geographical map. But, what about time? We envisioned three most likely use cases.

1. A static summary report with all download points on the world map for the selected period of time

2. A sequence of images, each one showing the download points on the world map for each minute/hour/day in the selected period of time. This rendering shows the evolution of the geography of the user base over time. Such evolution can be observed even better if, instead of using a sequence of images, we pack the same images one after the other into a movie.

3. Finally, we could visualize the geographical representation of the KNIME user base only for the last minute/hour/day. By refreshing the image at each minute/hour/day we can observe the evolution of the current geography over time.

Mainly for time reasons, we have decided to implement the first two use cases and to leave to the reader the challenge of extending this work to the third use case.

The Weblog Reader Node

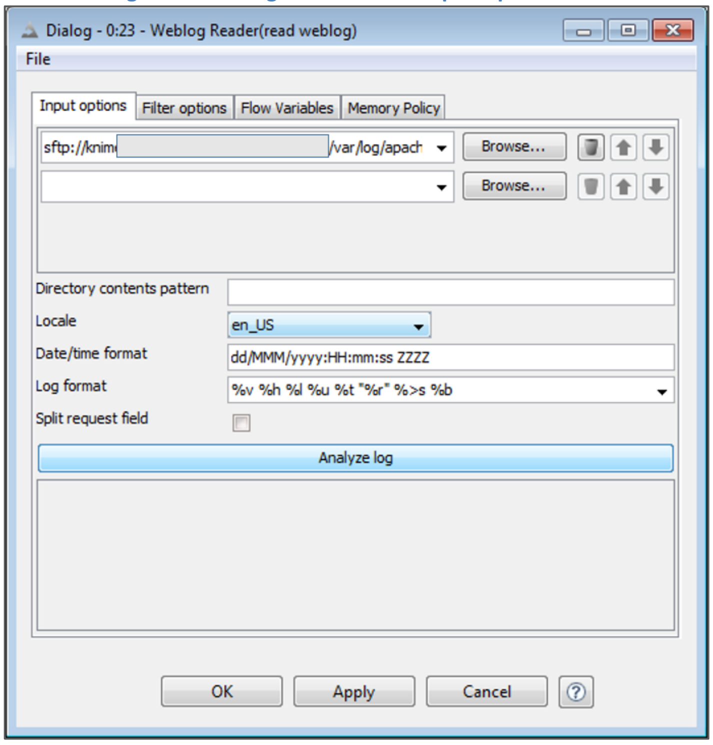
The IP addresses accessing a web page are usually contained in the web log file. The web log file though might be messy and hard to parse to extract the information of interest. For our good luck, KNIME offers a Weblog Reader node that reads Apache web log files.

The Weblog Reader node is part of the standard KNIME Extensions and can be found in the Misc/Web Analytics category in the Node Repository. The configuration window of the node hosts two tabs: Input Options and Filter Options. The Input Options tab sets the location and the format of the Apache log file(s). The Filter Options tab sets a possible time window to extract specific time constrained contents from the web log file(s).

We wrote file(s), and not file, because the Weblog Reader node can read one or more web log files. All targeted log files have to be specified in the first two text boxes (Fig. 1) as a file path/url or as a folder path/url in the Input Options tab. If you specify a folder rather than a single file, you can insert a regular expression in the “Directory contents pattern” box to read only some of the files in the folder. Wildcard expressions are not allowed.

After that, you need to define the formats used in your log file(s): the locale, the datetime, and the log format. The locale setting is necessary to parse dates correctly. The default locale is en_US, which works for most of the web servers. The date java format is also necessary to read dates and the default format provided in the configuration window again works for most of the web servers.

Figure 1. Weblog Reader node: Input Options tab



Dialog - 0:23 - Weblog Reader(read weblog)

File

Input options | Filter options | Flow Variables | Memory Policy

sftp://knim /var/log/apach

Directory contents pattern:

Locale: en_US

Date/time format: dd/MMM/yyyy:HH:mm:ss ZZZZ

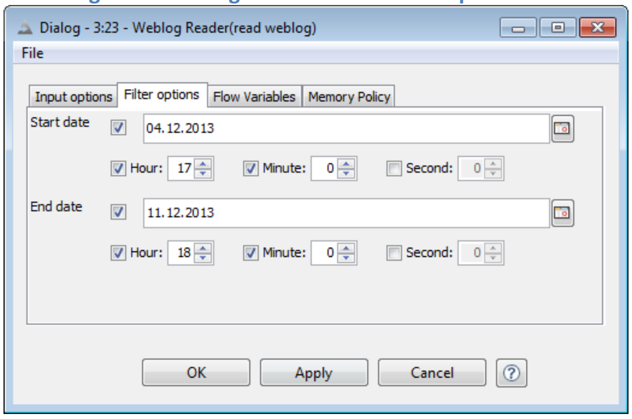
Log format: %v %h %l %u %t "%r" %>s %b

Split request field: ☐

Finally, you need to specify the format for a complete log line, as specified in the Apache configuration file. You can check the full syntax in the Weblog Reader node description or, even better, in the Apache documentation. Different log formats can be used alone or combined to read different log lines. If you click on "Analyze log" the first line of the first file is read and analyzed with the given format. If the format matches, you will get a preview of the columns and types. The types and columns names are hard-coded and cannot be changed.

In the second tab (Fig. 2) you can specify time ranges for requests you want to include in the output data table. All requests outside the specified time range are filtered out. The start date is inclusive, whereas the end date is exclusive.

Figure 2. Weblog Reader node: Filter Options tab



Dialog - 3:23 - Weblog Reader(read weblog)

File

Input options | Filter options | Flow Variables | Memory Policy

Start date ☑ 04.12.2013

☑ Hour: 17 ☑ Minute: 0 ☐ Second: 0

End date ☑ 11.12.2013

☑ Hour: 18 ☑ Minute: 0 ☐ Second: 0

The Data

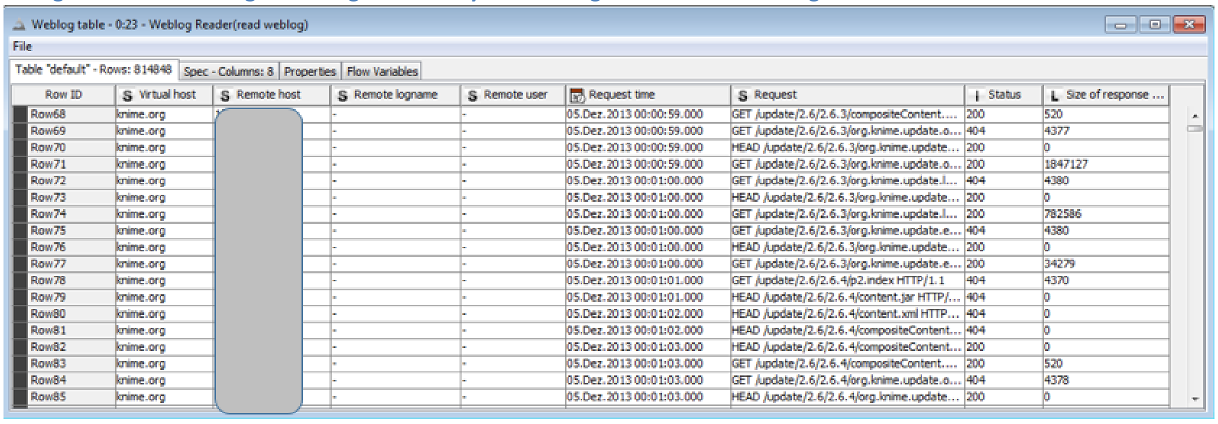
We set our Weblog Reader node to read the access.log file of our Apache server on a time window from December 5th (included) to December 12th (excluded) 2013. Notice that on December 6th 2013 the new KNIME 2.9 had been released. At this point in time and for a few following days, most of the active KNIME users have been updating their installation with the new version. We therefore considered this one-week time window as the most suitable to give us a representative picture of the geographical distribution of the KNIME user base around the world.

As log format we used “%v %h %l %u %t "%r %>s %b”, which means to extract:

- %v the virtual host the request was sent to
- %h the remote host, that is the client IP address
- %l the remote logname, if identification is required
- %u the remote user, if identification is required
- %t the request timestamp
- %r the request
- %>s the HTTP status code
- %b the response size in bytes

This log format, after the node execution, produces the data table in figure 3.

Figure 3. The Web Log Table as generated by the Weblog Reader node with log format %v %h %l %u %t "%r" %s %b



Weblog table - 0:23 - Weblog Reader(read weblog)

File

Table "default" - Rows: 814848 | Spec - Columns: 8 | Properties | Flow Variables

Row ID	Virtual host	Remote host	Remote logname	Remote user	Request time	Request	Status	Size of response ...
Row68	knime.org	[redacted]	-	-	05.Dez.2013 00:00:59.000	GET /update/2.6/2.6.3/compositeContent....	200	520
Row69	knime.org	[redacted]	-	-	05.Dez.2013 00:00:59.000	GET /update/2.6/2.6.3/org.knime.update.o...	404	4377
Row70	knime.org	[redacted]	-	-	05.Dez.2013 00:00:59.000	HEAD /update/2.6/2.6.3/org.knime.update...	200	0
Row71	knime.org	[redacted]	-	-	05.Dez.2013 00:00:59.000	GET /update/2.6/2.6.3/org.knime.update.o...	200	1847127
Row72	knime.org	[redacted]	-	-	05.Dez.2013 00:01:00.000	GET /update/2.6/2.6.3/org.knime.update.l...	404	4380
Row73	knime.org	[redacted]	-	-	05.Dez.2013 00:01:00.000	HEAD /update/2.6/2.6.3/org.knime.update...	200	0
Row74	knime.org	[redacted]	-	-	05.Dez.2013 00:01:00.000	GET /update/2.6/2.6.3/org.knime.update.l...	200	782586
Row75	knime.org	[redacted]	-	-	05.Dez.2013 00:01:00.000	GET /update/2.6/2.6.3/org.knime.update.e...	404	4380
Row76	knime.org	[redacted]	-	-	05.Dez.2013 00:01:00.000	HEAD /update/2.6/2.6.3/org.knime.update...	200	0
Row77	knime.org	[redacted]	-	-	05.Dez.2013 00:01:00.000	GET /update/2.6/2.6.3/org.knime.update.e...	200	34279
Row78	knime.org	[redacted]	-	-	05.Dez.2013 00:01:01.000	GET /update/2.6/2.6.4/p2.index HTTP/1.1	404	4370
Row79	knime.org	[redacted]	-	-	05.Dez.2013 00:01:01.000	HEAD /update/2.6/2.6.4/content.jar HTTP/...	404	0
Row80	knime.org	[redacted]	-	-	05.Dez.2013 00:01:02.000	HEAD /update/2.6/2.6.4/content.xml HTTP...	404	0
Row81	knime.org	[redacted]	-	-	05.Dez.2013 00:01:02.000	HEAD /update/2.6/2.6.4/compositeContent...	404	0
Row82	knime.org	[redacted]	-	-	05.Dez.2013 00:01:03.000	HEAD /update/2.6/2.6.4/compositeContent...	200	0
Row83	knime.org	[redacted]	-	-	05.Dez.2013 00:01:03.000	GET /update/2.6/2.6.4/compositeContent....	200	520
Row84	knime.org	[redacted]	-	-	05.Dez.2013 00:01:03.000	GET /update/2.6/2.6.4/org.knime.update.o...	404	4378
Row85	knime.org	[redacted]	-	-	05.Dez.2013 00:01:03.000	HEAD /update/2.6/2.6.4/org.knime.update...	200	0

%h extracts the remote host, i.e. its IP address. For privacy reasons, the IP addresses have been masked in figure 3 and only the first three digits of the IP addresses are stored in the system for future purposes. For the same reasons, the example workflows, that are downloadable from the EXAMPLES server, use a set of randomized fake IP addresses.

Geo-Localization of Downloads

From the Apache server we then retrieve the list of IP addresses that have accessed the web site in the defined time window. The next steps now are:

- to extract only those IP addresses who have downloaded or updated the KNIME data analytics platform
- to associate each IP address with a specific world region
- to display the IP address as a point on the corresponding region of the world map

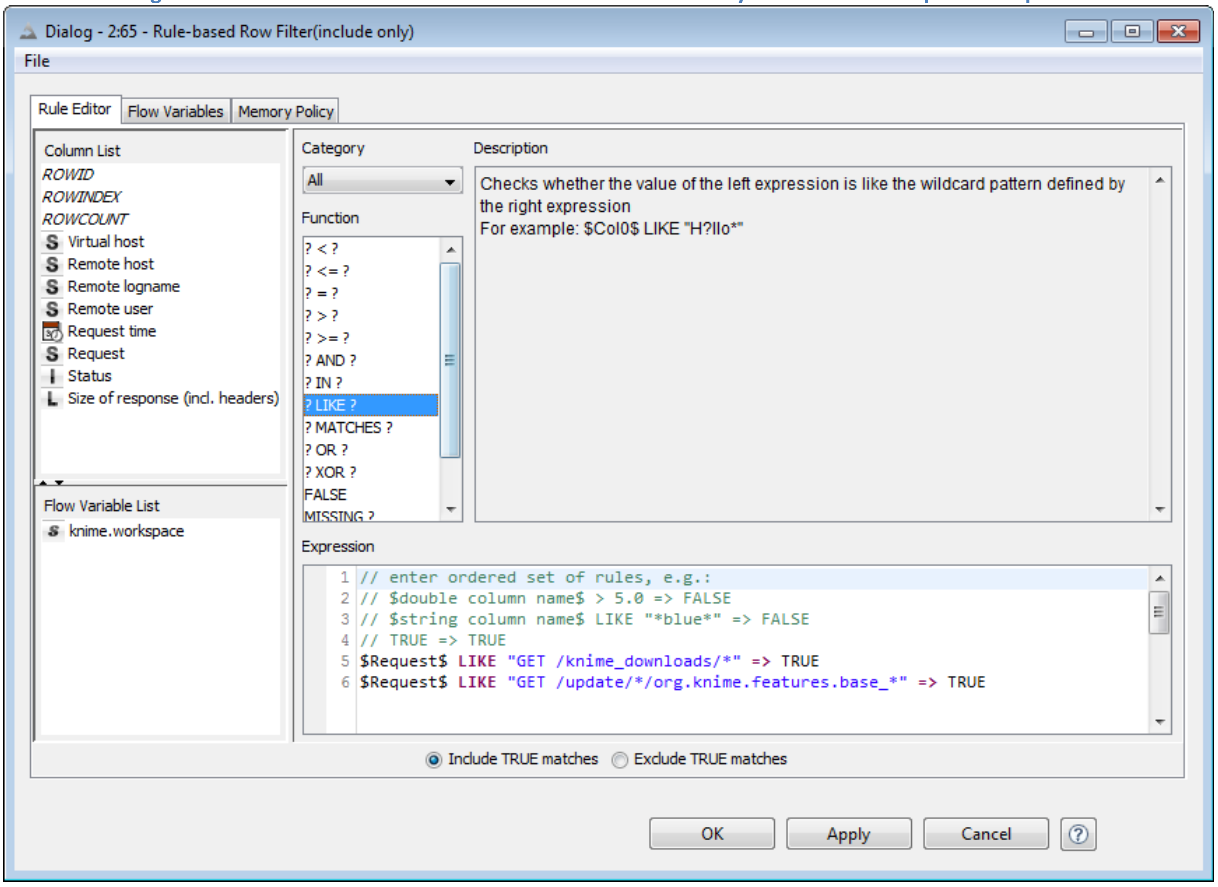
Data Preparation: the Rule based Row Filter Node

It is quite easy to isolate the requests pertaining to downloads and updates. Indeed, as we can see in figure 3, the data column named “Request” contains all the details of the web access type, including "GET /knime_downloads/…" for installation package downloads and "GET /update/…/org.knime.features.base_…." for the update requests.

In order to filter these two request types only, we just need to use two Row Filter nodes with the wild card flag enabled, followed by a Concatenate node. Alternatively, and probably more practically, we can use only one node, the Rule-based Row Filter node (Fig. 4), covering both row filtering rules, like:

$Request$ LIKE "GET /knime_downloads/" => TRUE*
$Request$ LIKE "GET /update//org.knime.features.base_*" => TRUE*

Figure 4. The Rule-based Row Filter node used to include only downloads and updates requests



Checks whether the value of the left expression is like the wildcard pattern defined by the right expression
For example: $Col0$ LIKE "H?llo*"

```
// enter ordered set of rules, e.g.:
// $double column name$ > 5.0 => FALSE
// $string column name$ LIKE "*blue*" => FALSE
// TRUE => TRUE
$Request$ LIKE "GET /knime_downloads/*" => TRUE
$Request$ LIKE "GET /update/*/org.knime.features.base_*" => TRUE
```

(●) Include TRUE matches ( ) Exclude TRUE matches

The Rule-based Row Filter node allows the user to define a list of rules in its editor space. Rules consist of a condition part (antecedent) - which must evaluate to TRUE or FALSE - and an outcome part (consequent) - which follows the => symbol and assumes value either TRUE or FALSE.

During execution, each rule is applied to each row of the input data table: if the first matching rule has outcome TRUE, the data row is included in the output data table; if the first matching rule yields to FALSE, the data row is excluded from the output data table. If no rule matches the data row, the data row is excluded. Inclusion and exclusion may be inverted, by switching the enabled radio button below the rule editor frame.

You can also write comments in the rule editor frame to explain the field of action and/or the reasoning behind each rule. A comment line starts with //. Anything after // is not interpreted by the rule engine. Logical expressions can be grouped within parentheses.

The Rule-based Row Filter node automatically accepts * as the wild card sign. In addition, ROWID, ROWINDEX, and ROWCOUNT can be used to build a rule, respectively to express the RowID, the row index and the number of table rows.

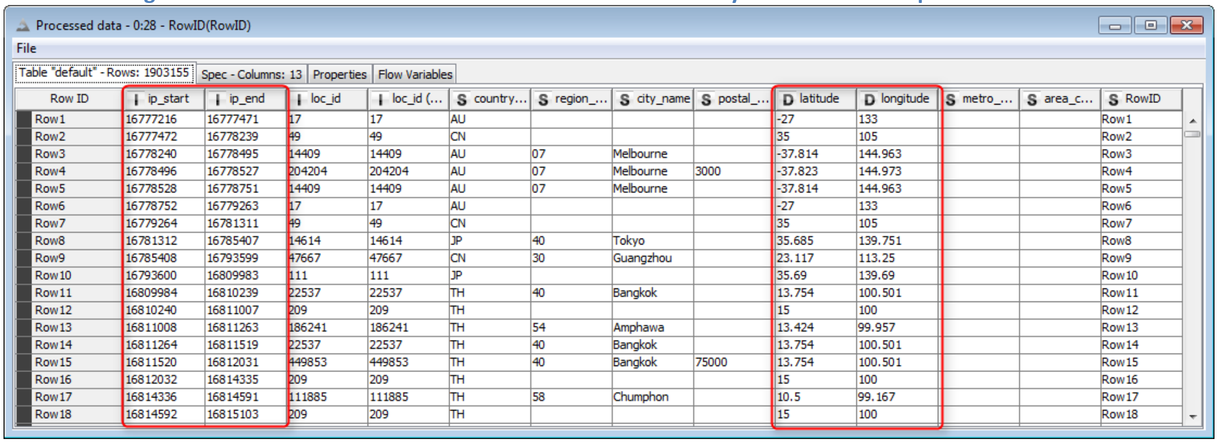
Data Preparation: IP Addresses and World Regions

Now that we have the IP addresses of KNIME downloads and updates, we extract the list of unique IP addresses using a GroupBy node.

IP numbers, calculated from IP addresses, have been assigned to specific regions of the world. We then need the map that associates ranges of IP numbers to geographical locations, in terms of latitude and longitude, in order to display IP addresses as points on a world map

The web site http://www.maxmind.com/en/city offers the location vs. IP address map both as a web service and as a csv file. We downloaded the csv file map and we imported it into an sqlite database. In the workflow then, after reading the map from the database, we get the data table displayed in figure 5, where we can see the IP number ranges (from ip_start to ip_end) and the corresponding latitude and longitude coordinates.

Figure 5. The Rule-based Row Filter node used to include only downloads and updates data rows


Processed data - 0:28 - RowID(RowID)

File

Table "default" - Rows: 1903155 | Spec - Columns: 13 | Properties | Flow Variables

Row ID	ip_start	ip_end	loc_id	loc_id (...	country...	region_...	city_name	postal_...	latitude	longitude	metro_...	area_c...	RowID
Row1	16777216	16777471	17	17	AU				-27	133			Row1
Row2	16777472	16778239	49	49	CN				35	105			Row2
Row3	16778240	16778495	14409	14409	AU	07	Melbourne		-37.814	144.963			Row3
Row4	16778496	16778527	204204	204204	AU	07	Melbourne	3000	-37.823	144.973			Row4
Row5	16778528	16778751	14409	14409	AU	07	Melbourne		-37.814	144.963			Row5
Row6	16778752	16779263	17	17	AU				-27	133			Row6
Row7	16779264	16781311	49	49	CN				35	105			Row7
Row8	16781312	16785407	14614	14614	JP	40	Tokyo		35.685	139.751			Row8
Row9	16785408	16793599	47667	47667	CN	30	Guangzhou		23.117	113.25			Row9
Row10	16793600	16809983	111	111	JP				35.69	139.69			Row10
Row11	16809984	16810239	22537	22537	TH	40	Bangkok		13.754	100.501			Row11
Row12	16810240	16811007	209	209	TH				15	100			Row12
Row13	16811008	16811263	186241	186241	TH	54	Amphawa		13.424	99.957			Row13
Row14	16811264	16811519	22537	22537	TH	40	Bangkok		13.754	100.501			Row14
Row15	16811520	16812031	449853	449853	TH	40	Bangkok	75000	13.754	100.501			Row15
Row16	16812032	16814335	209	209	TH				15	100			Row16
Row17	16814336	16814591	111885	111885	TH	58	Chumphon		10.5	99.167			Row17
Row18	16814592	16815103	209	209	TH				15	100			Row18

To convert an IP address, x.y.z.k, into the corresponding IP number, we use a Math Formula node implementing:

*IP number = x * 16777216 + y * 65536 + z * 256 + k*

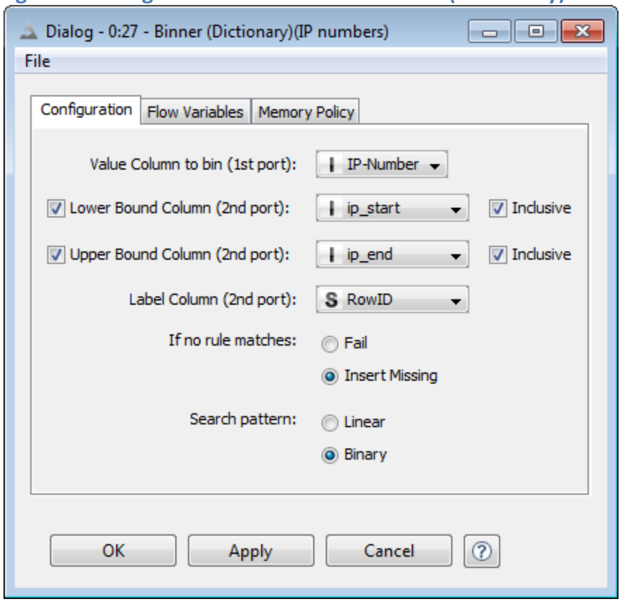
The resulting IP numbers are collected into bins, as defined in the map data table of figure 5. Each bin starts from *ip_start*, ends in *ip_end,* and it is labelled with the RowID of the map data table. That is if IP_number falls between ip_start and ip_end of row n, it will end up in the bucket labelled "Row n".

For this binning operation we use a Binner(Dictionary) node, which requires a dictionary table for the bin settings at the lower input port. We use the map data table as the dictionary table, thus, connecting it to the lower input port. We then connect the IP numbers coming from the Apache web log file to the upper input port. IP-Number is the data column to bin, ip_start and ip_end the bin boundaries, and RowID the bin label (Fig. 6).

We have now the IP address, the IP number, and the bin label. Each bin label corresponds to an IP number range [ip_start, ip_end), which also corresponds to a world region localized through its latitude and longitude coordinates. We need to pull in these geographical coordinates for each IP number range from the original map data table. In order to achieve this, we join the original map data table with the IP number bins using the RowID as the joining key.

We have now all we need for the geo-localization of each IP address: the IP address, its latitude and longitude.

Figure 6. Configuration window of the Binner(Dictionary) node



Dialog - 0:27 - Binner (Dictionary)(IP numbers)

Value Column to bin (1st port): IP-Number
[✓] Lower Bound Column (2nd port): ip_start [✓] Inclusive
[✓] Upper Bound Column (2nd port): ip_end [✓] Inclusive
Label Column (2nd port): RowID
If no rule matches: Fail [✓ Insert Missing]
Search pattern: Linear [✓ Binary]

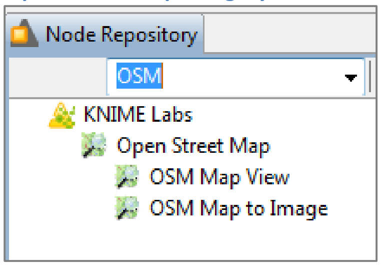
The Open Street Map Package

For the geo-localization task, KNIME offers an integration with the OpenStreetMap libraries.

The OpenStreetMap project (http://www.openstreetmap.org) creates and distributes free world geographic data. Connecting to the OpenStreetMap API you can localize and visualize a point on a world map through its latitude and longitude coordinates. The project wiki (http://wiki.openstreetmap.org/wiki/Main_Page) can teach you more about implementation and connection details.

KNIME has made available an OpenStreetMap integration extension under KNIME Labs Extensions / Open Street Map Integration. This package is installed in the KNIME Labs category under Open Street Map (Fig. 7) and, as of KNIME 2.9, contains two nodes only: OSM Map View and OSM Map to Image. OSM Map View provides an interactive view on OpenStreetMap. OSM Map to Image generates an image of a selected section of the world map. Both nodes require an Internet connection.

Figure 7. The Open Street Map category in the Node Repository

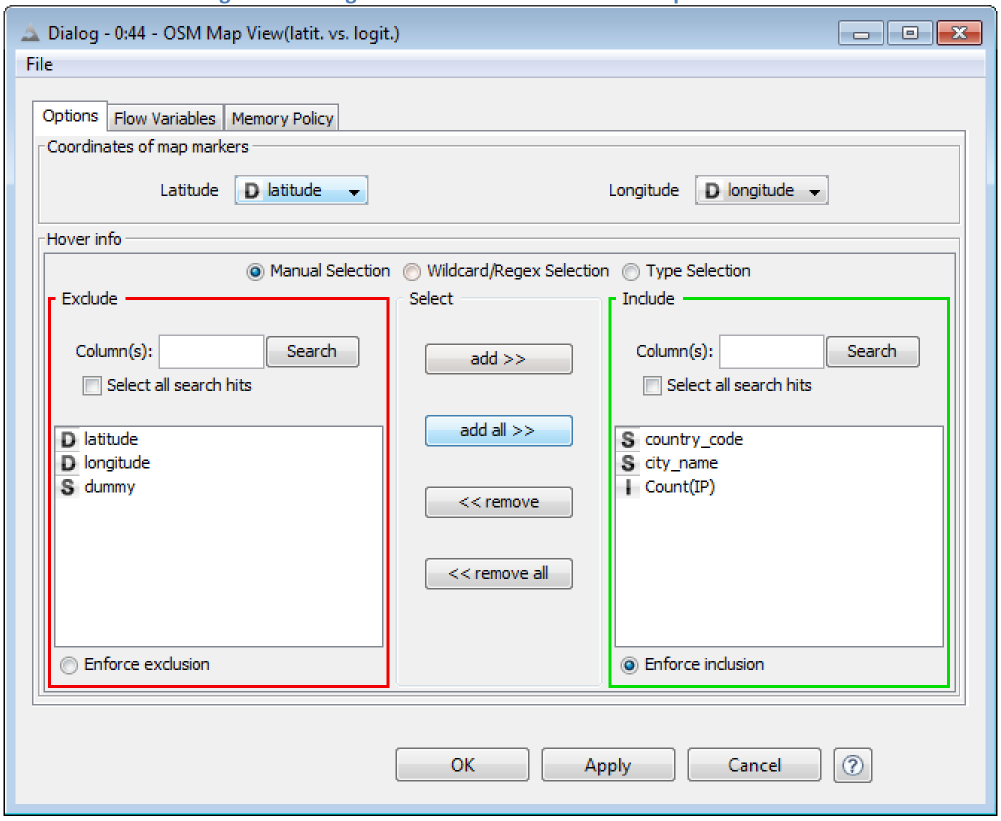


The interactive View with the OSM Map View Node

We are not really interested in localizing every single IP address, but rather in visualizing the number of downloads/updates for each world location. The first thing to do then is to count the number of IP addresses by location, i.e. by latitude and longitude with a GroupBy node.

The OSM Map View node reads the latitude and longitude coordinates for each input data row from two columns of the input data table. Additional information can be displayed when hovering over each point, like the country, the city, and the number of IP addresses found in this location. All this information needs to be provided in the node configuration window (Fig. 8).

Figure 8. Configuration window of the OSM Map View node


Dialog - 0:44 - OSM Map View(latit. vs. logit.)

File

Options | Flow Variables | Memory Policy

Coordinates of map markers

Latitude: D latitude

Longitude: D longitude

Hover info

[✓ Manual Selection] Wildcard/Regex Selection Type Selection

Exclude: D latitude, D longitude, S dummy

Include: S country_code, S city_name, I Count(IP)

Enforce exclusion [✓ Enforce inclusion]

Once executed, the OSM Map View node produces an interactive view of the number of downloads per location over the whole world (Fig. 9). This interactive view also includes hiliting, as you can see from the Hilite option in the top menu of the node view.

General Data Views properties apply to the OSM Map View node's view as well. In particular, the point shape is set to a circle, the color to a heatmap on the number of IP addresses (light blue for a low count, red for a high count), and the size as proportional to the number of IP addresses. Therefore, a large red circle indicates a very large number of IP addresses downloading KNIME in that location. A small light blue circle indicates one or two KNIME downloads only. Some of the IP ranges only have the country code and no city and therefore the biggest and reddest points on the map usually refer to a full country with no specific city.

From the interactive map reported in figure 9, you can see that, around the end of 2013, Europe hosts most of the KNIME users, followed by USA, India, and China.

The static Report with the OSM Map to Image Node

Now we want to incorporate the image in figure 9 into a report together with a pie chart displaying the percentages of unique downloads by country. The data for the pie chart is produced by counting the IP addresses by country with a GroupBy node.

The world map image in figure 9 is exported into the report by using an OSM Map to Image and an Image to Table node. The OSM Map to Image node works similarly to the OSM Map View node. It reads the point latitude and longitude from selected input data columns in the Map Marker tab (Fig. 11) and it displays the point on a world map as defined in the Options tab (Fig. 10) of the configuration window. Once executed, the node produces the data table and the world image at the output ports. Countries with less than 3% downloads are grouped together under the label "other".

Figure 9. Interactive view generated by the OSM Map View node after execution

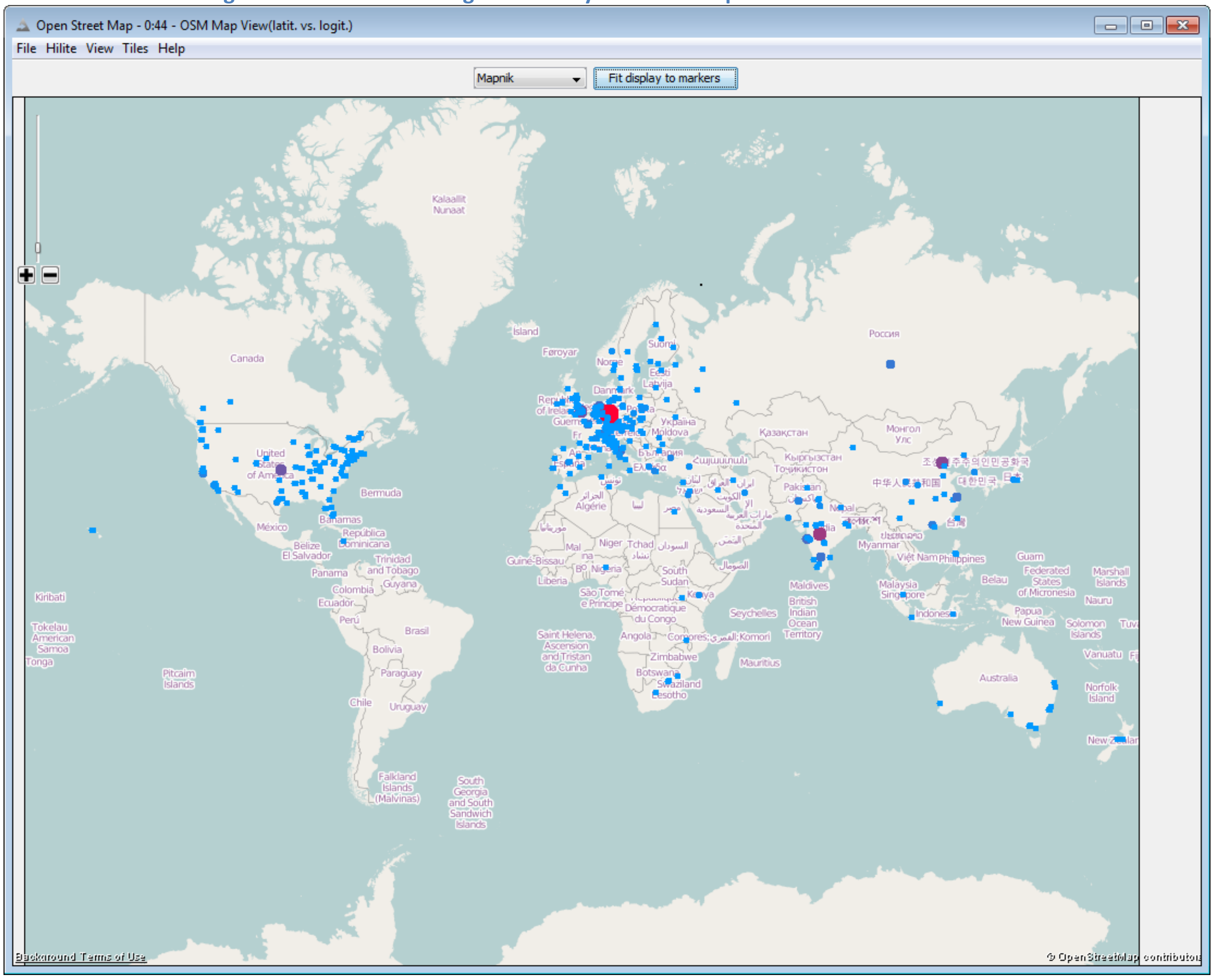
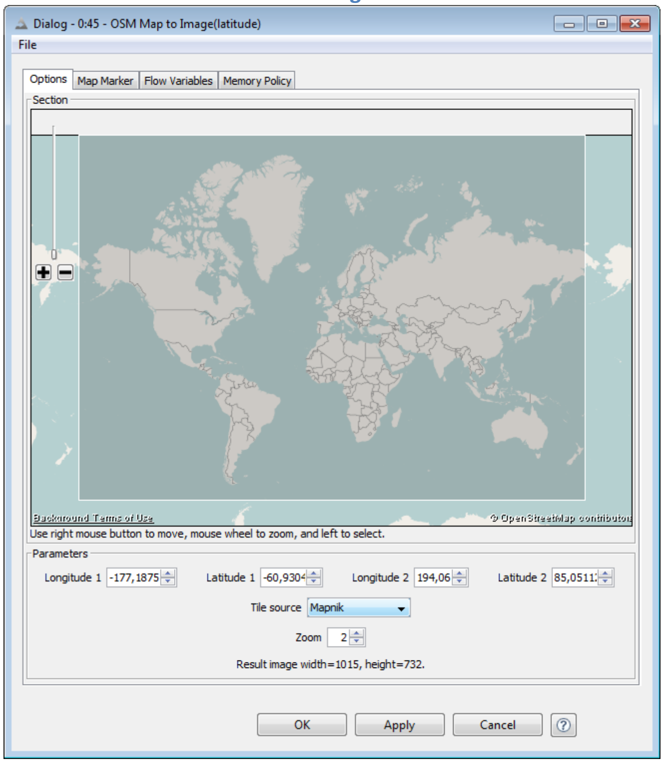

Open Street Map - 0:44 - OSM Map View(latit. vs. logit.)
File Hilite View Tiles Help
Mapnik
Fit display to markers
Background Terms of Use
© OpenStreetMap contributors

Figure 10. "Options" tab of the configuration window of the OSM Map to Image node



Use right mouse button to move, mouse wheel to zoom, and left to select.

Parameters

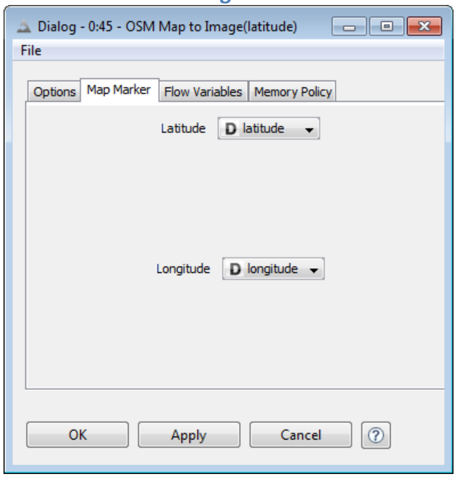
Longitude 1 -177,1875 Latitude 1 -60,930 Longitude 2 194,06 Latitude 2 85,0511

Tile source Mapnik

Zoom 2

Result image width=1015, height=732.

Figure 11. "Map Marker" tab of the configuration window of the OSM Map to Image node



Latitude D latitude

Longitude D longitude

Figure 12. Static report with the image generated by the OSM Map to Image node

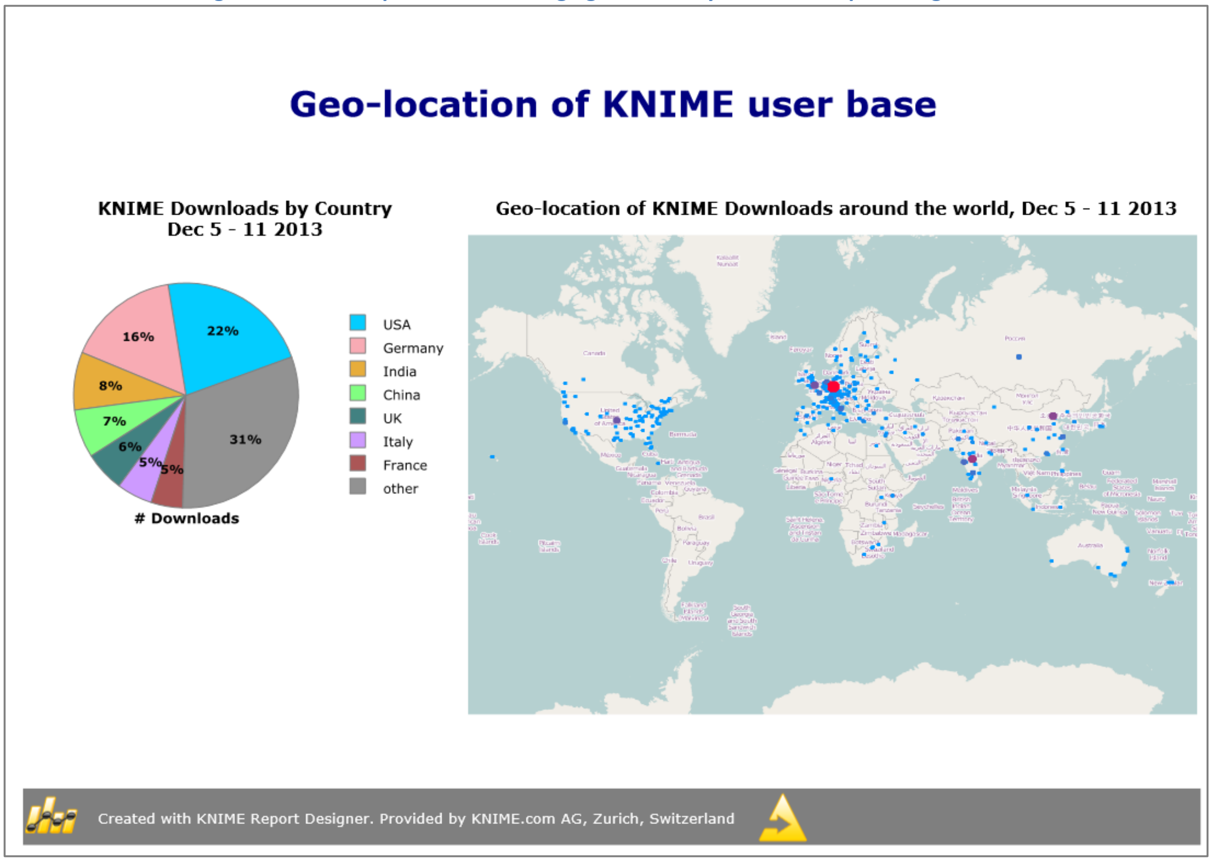
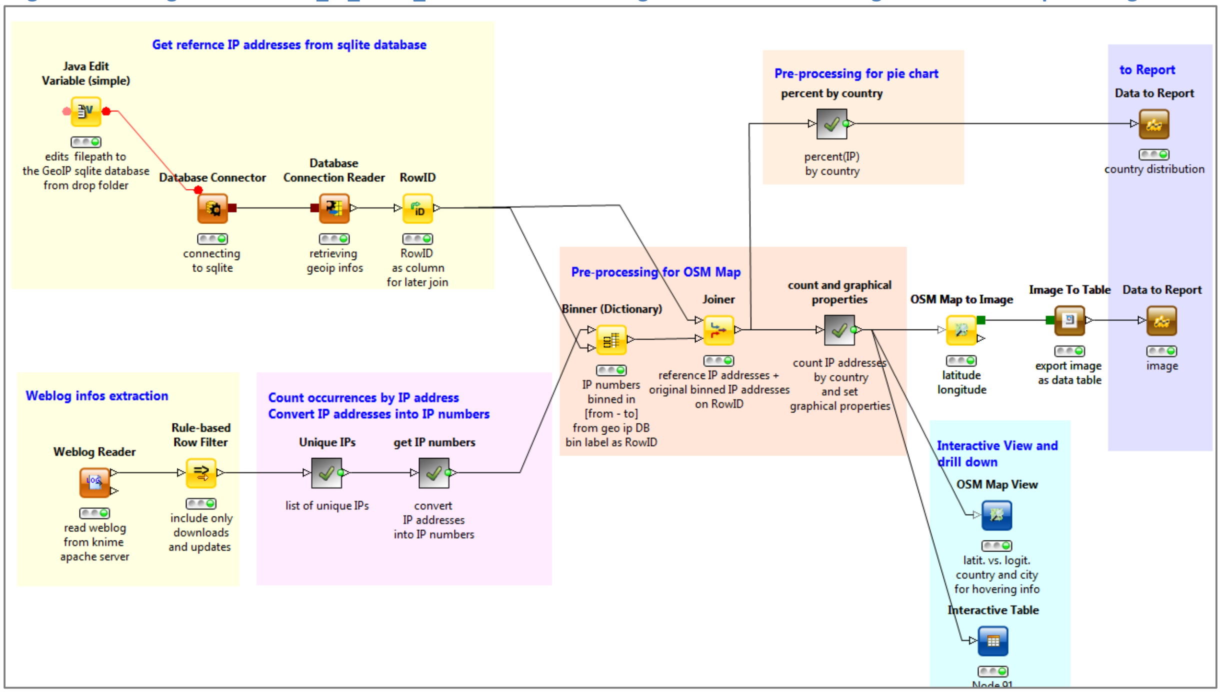


Figure 13. The "geolocalization_IP_static_2013" workflow that generates the view in figure 9 and the report in figure 12



Get refernce IP addresses from sqlite database
Java Edit Variable (simple)
edits filepath to the GeoIP sqlite database from drop folder
Database Connector
connecting to sqlite
Database Connection Reader
retrieving geoip infos
RowID
RowID as column for later join
Pre-processing for pie chart
percent by country
percent(IP) by country
to Report
Data to Report
country distribution
Pre-processing for OSM Map
Binner (Dictionary)
IP numbers binned in [from - to] from geo ip DB bin label as RowID
Joiner
reference IP addresses + original binned IP addresses on RowID
count and graphical properties
count IP addresses by country and set graphical properties
OSM Map to Image
latitude longitude
Image To Table
export image as data table
Data to Report
image
Weblog infos extraction
Weblog Reader
read weblog from knime apache server
Rule-based Row Filter
include only downloads and updates
Count occurrences by IP address
Convert IP addresses into IP numbers
Unique IPs
list of unique IPs
get IP numbers
convert IP addresses into IP numbers
Interactive View and drill down
OSM Map View
latit. vs. logit. country and city for hovering info
Interactive Table
Node 91

The final report page is shown in figure 12 and the underlying workflow, named "geolocalization_IP_static_2013" in figure 13. This workflow is available for download from the EXAMPLES server under:

"008_WebAnalytics_and_OpenStreetMap / 008004_geolocalization_IP_static_2013".

In order to protect the privacy of the KNIME users the original data has been changed with a fake Apache web log file containing a list of 100 000 randomly generated IP addresses. Notice that, since the IP address space is by now close to saturation, the randomly generated IP addresses might exist. This is pure coincidence and has nothing to do with the original IP addresses used for this study.

Monitoring Daily Downloads

The image generated in the previous report is a summary image, i.e. it covers the whole time between December 5th and December 11th 2013: there is no information about the download trend over time.

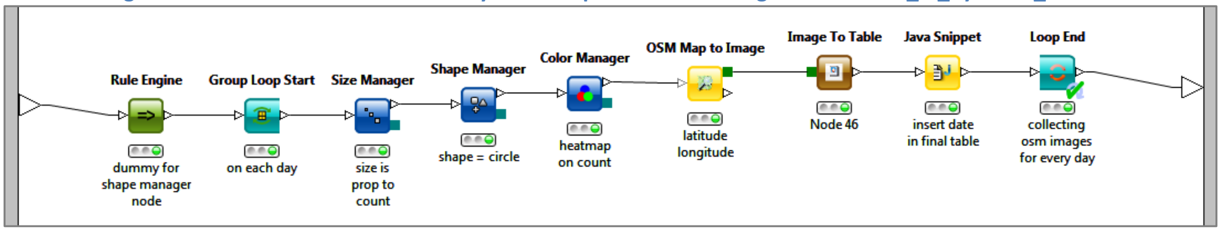
Let's suppose that we want to know more about the daily downloads. In this case, we need to modify the workflow "geolocalization_IP_static_2013", reported in figure 13, after the Joiner node, to loop over the days between December 5th and December 11th 2013 and to generate the world-wide map of KNIME downloads for every day.

Data Preparation and Static Report

In the Request Time field, we mask the time and we only keep the day information. Then we loop on the values in data column Request Time, using a Group Loop Start and a Loop End node.

The loop body consists of the Size Manager, the Color Manager, and the Shape Manager nodes (previously in the metanode "count and graphical properties"), the OSM Map to Image and the Image to Table nodes as in figure 13, and an additional Java Snippet node to append the current day to the generated image. This sequence of nodes is encapsulated in a new metanode named "Daily OSM Maps" (Fig. 14) in the new workflow named "geolocalization_IP_dynamic_2013".

Figure 14. Content of metanode "Daily OSM Maps" in workflow "geolocalization_IP_dynamic_2013"



Finally a Data to Report node exports the sequence of daily images into a report, obtaining the effect of a roughly movie-like report (Fig. 15).

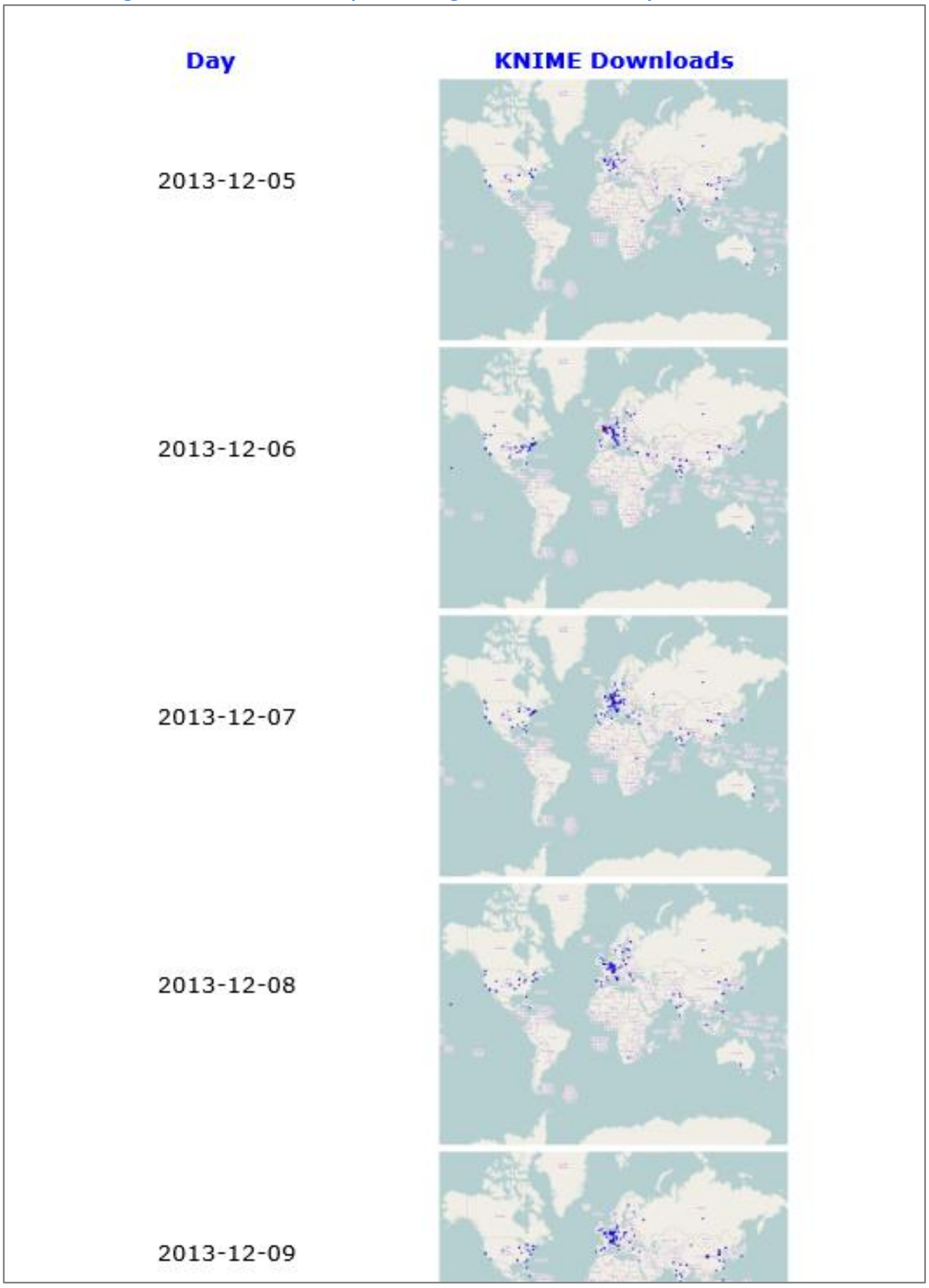
The KNIME Image Processing Package

The sequence of daily download snapshots in the previous report though does not give justice to the dynamical geographical landscape evolution of the KNIME downloads. It would be infinitely more informative to summarize such a sequence of images into a movie.

If we do not want to produce a movie by flipping the report pages, we must use the KNIME Image Processing extension (http://tech.knime.org/community/image-processing). In fact, this extension contains many nodes for highly sophisticated image processing, including the conversion of an image sequence into a movie file.

The KNIME Image Processing Plugin is a community extension and it has to be installed separately from the community extension packages. KNIME image reading nodes are based on Bio-Formats API and therefore can read more than 120 different kinds of images. In addition, the KNIME Image Processing package offers a large number of nodes for image processing, like filtering, segmentation, feature extraction, tracking and classification. All more than 100 nodes operate on multi-dimensional image data (e.g. videos, 3D images, multi-channel images, or even a combination of them).

Figure 15. The movie-like report of the geolocation of the daily KNIMEdownloads

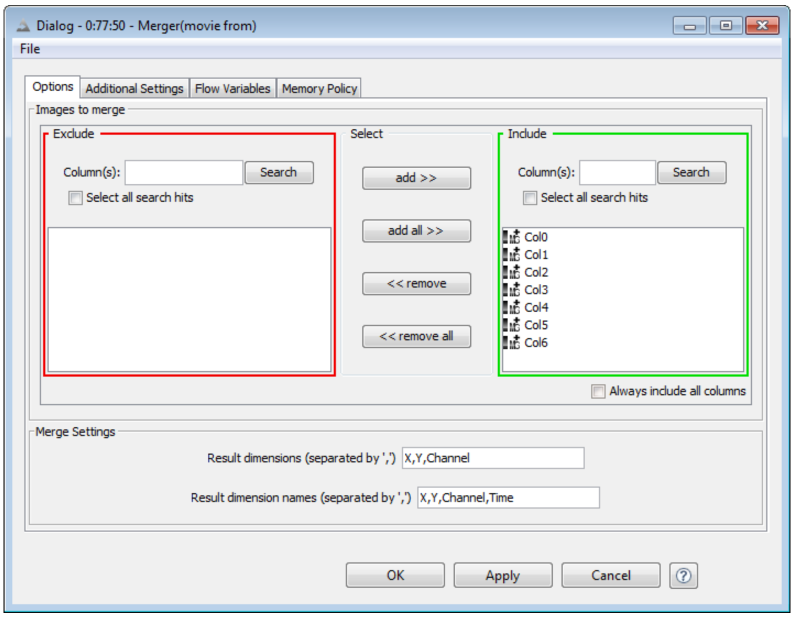


Day	KNIME Downloads
2013-12-05	
2013-12-06	
2013-12-07	
2013-12-08	
2013-12-09	

Of all nodes available in the Image Processing category, we are interested in the Merger node. This node merges corresponding pixels from images in different columns into one signle line of pixel in a new image. The pixel type in the resulting image can then be interpreted as a movie pixel.

The configuration window of the Merger node requires, in the Options tab (Fig. 16), the list of image columns to merge and the image size over all resulting coordinates, like for example 500x400x10, where 500x400 is the number of pixels along the x,y coordinates and 10 is the number of images collapsed into the resulting one. However, if you do not want to define fixed maximum sizes for your coordinates, you can always provide the coordinate labels instead of numbers. In this case, the coordinate size will be determined automatically during execution. Again on the example above, using “X, Y, Channel” instead of “500, 400, 10” allows for an automatic definition of the 3 coordinate maximum size. Labels of the image coordinates have also to be provided.

Figure 16. The Options tab in the configuration window of the Image Merger node



Dialog - 0:77:50 - Merger(movie from)

Options | Additional Settings | Flow Variables | Memory Policy

Images to merge

Exclude: Column(s): [ ] Search; Select all search hits

Select: add >>; add all >>; << remove; << remove all

Include: Column(s): [ ] Search; Select all search hits; Col0, Col1, Col2, Col3, Col4, Col5, Col6

Always include all columns

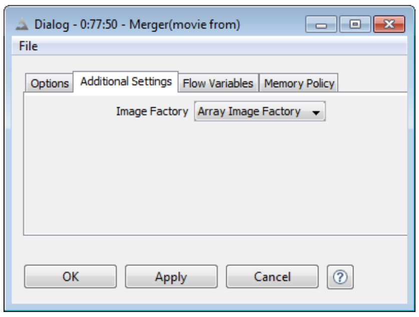
Merge Settings

Result dimensions (separated by ',') X,Y,Channel

Result dimension names (separated by ',') X,Y,Channel,Time

OK | Apply | Cancel

Figure 17. The Additional Settings tab in the configuration window of the Image Merger node



Dialog - 0:77:50 - Merger(movie from)

Options | Additional Settings | Flow Variables | Memory Policy

Image Factory: Array Image Factory

OK | Apply | Cancel

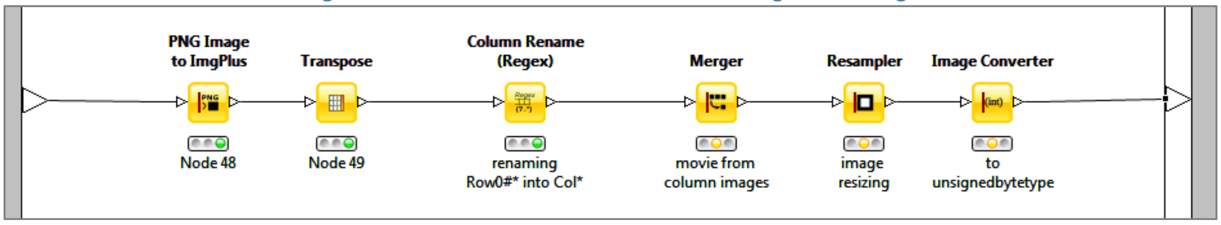
In the Additional Settings tab (Fig. 17), the Image Factory setting defines the way the new image is created and stored in memory: as an array of images or as a planar image.

Alternatively, you could also merge images with a GroupBy node using the Merge-Image aggregation operation.

There are two necessary caveats to using the Image Merger node: input images have to be of type ImgPlus (PNG Images will not work) and the image sequence has to be stored on a data row rather than on a data column. In order to solve the first issue, we use a PNG Image to ImgPlus node, while for the second issue we just transpose the input data table.

In order to smooth the image transition in the movie, after the Image Merger node, we linearly interpolate the image sequence and stretch the coordinates with a Resampler node. We then convert the resulting image pixels to unsignedbytetype, to fit most movie readers’ requirements. The resulting movie is finally written to a temporary directory with the Image Writer node.

Figure 18. Conent of the Metanode named Image Processing



This last image processing workflow is stored in a metanode named “Image Processing” and reported in figure 18. Below you can see the final short movie, named Image.avi.

Image.avi

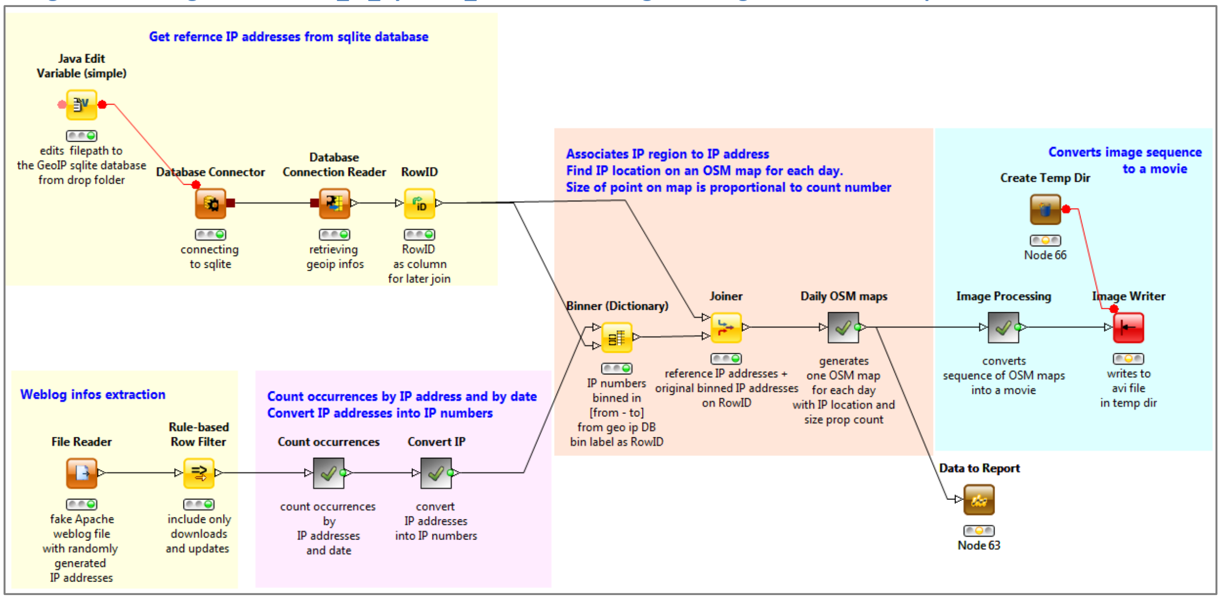
The workflow, named "geolocalization_IP_dynamic_2013" (Fig. 19), is available for download from the EXAMPLES server under:

"008_WebAnalytics_and_OpenStreetMap / 008005_geolocalization_IP_dynamic_2013".

Again, in order to protect the privacy of the KNIME users the original data has been changed with a fake Apache web log file containing a list of 100 000 randomly generated IP addresses. Notice that, since the IP address space is by now close to saturation, the randomly generated IP addresses might exist. This is pure coincidence and has nothing to do with the original IP addresses used for this study.

Also, to reduce the execution time, we introduced a Row Filter node in the Count Occurrences metanode, selecting only the month of December from the input fake web log file.

Figure 19. The "geolocalization_IP_dynamic_2013" workflow generating the movie-like report and the actual movie



Get refernce IP addresses from sqlite database
Java Edit Variable (simple)
edits filepath to the GeoIP sqlite database from drop folder
Database Connector
connecting to sqlite
Database Connection Reader
retrieving geoip infos
RowID
RowID as column for later join
Associates IP region to IP address
Find IP location on an OSM map for each day.
Size of point on map is proportional to count number
Binner (Dictionary)
IP numbers binned in [from - to] from geo ip DB bin label as RowID
Joiner
reference IP addresses + original binned IP addresses on RowID
Daily OSM maps
generates one OSM map for each day with IP location and size prop count
Converts image sequence to a movie
Create Temp Dir
Node 66
Image Processing
converts sequence of OSM maps into a movie
Image Writer
writes to avi file in temp dir
Data to Report
Node 63
Weblog infos extraction
File Reader
fake Apache weblog file with randomly generated IP addresses
Rule-based Row Filter
include only downloads and updates
Count occurrences by IP address and by date
Convert IP addresses into IP numbers
Count occurrences
count occurrences by IP addresses and date
Convert IP
convert IP addresses into IP numbers

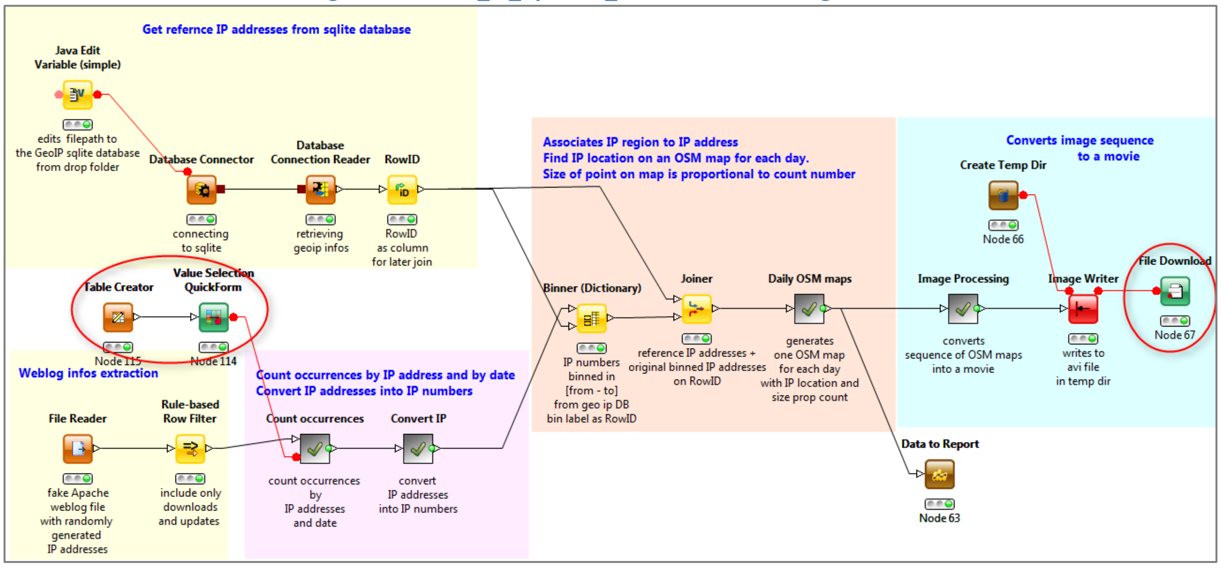
Workflow Remote Execution from the WebPortal

Now, if the head of the company would like to have a look at the movie of the geolocalization of KNIME downloads every month on demand from a web browser, we need:

1. to upload our "geolocalization_IP_dynamic_2013" workflow to the KNIME Server to be able to run a remote execution from the WebPortal on demand;
2. to download the final movie to the WebPortal through a File Download Quickform node;
3. and finally to control the month filtering from the WebPortal by using a Value Selection Quickform node fed by a Table Creator node with the months list.

These new additions to the "geolocalization_IP_dynamic_2013" workflow are marked in red in figure 20.

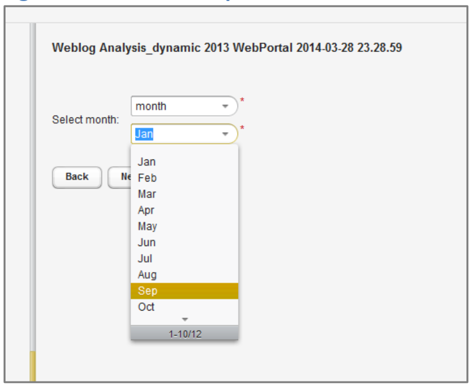
Figure 20. The "geolocalization_IP_dynamic_2013_WebPortal" workflow is a WebPortal adapted version of the "geolocalization_IP_dynamic_2013" workflow in figure 19



Get refernce IP addresses from sqlite database
Java Edit Variable (simple)
edits filepath to the GeoIP sqlite database from drop folder
Database Connector
connecting to sqlite
Database Connection Reader
retrieving geoip infos
RowID
RowID as column for later join
Table Creator
Node 115
Value Selection QuickForm
Node 114
Weblog infos extraction
File Reader
fake Apache weblog file with randomly generated IP addresses
Rule-based Row Filter
include only downloads and updates
Count occurrences by IP address and by date
Convert IP addresses into IP numbers
Count occurrences
count occurrences by IP addresses and date
Convert IP
convert IP addresses into IP numbers
Associates IP region to IP address
Find IP location on an OSM map for each day.
Size of point on map is proportional to count number
Binner (Dictionary)
IP numbers binned in [from - to] from geo ip DB bin label as RowID
Joiner
reference IP addresses + original binned IP addresses on RowID
Daily OSM maps
generates one OSM map for each day with IP location and size prop count
Converts image sequence to a movie
Create Temp Dir
Node 66
Image Processing
converts sequence of OSM maps into a movie
Image Writer
writes to avi file in temp dir
File Download
Node 67
Data to Report
Node 63

In figures 21 and 22 you can see the wizard sequence on the WebPortal: the month selection and the movie and report display.

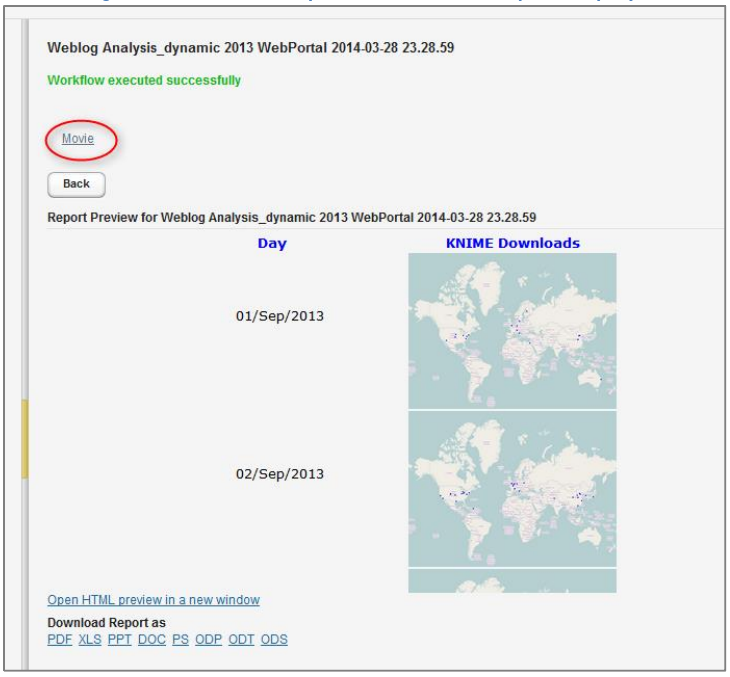
Figure 21. On the Webportal: Month Selection


Weblog Analysis_dynamic 2013 WebPortal 2014-03-28 23.28.59

Select month: month *

Jan *

Figure 22. On the Webportal: Movie and Report Display


Weblog Analysis_dynamic 2013 WebPortal 2014-03-28 23.28.59

Workflow executed successfully

Report Preview for Weblog Analysis_dynamic 2013 WebPortal 2014-03-28 23.28.59

Day	KNIME Downloads
01/Sep/2013	
02/Sep/2013	

Open HTML preview in a new window

Download Report as
PDF XLS PPT DOC PS ODP ODT ODS

Conclusions

In this whitepaper we visualize on a world map the IP addresses downloading or updating a KNIME installation.

We start from reading the Apache web log files using the Weblog Reader node. We then display the IP addresses as points on a geographical world map, by means of the KNIME Open Street Map integration extension. We also show how to produce a sequence of world map images and how to merge them into a movie stream. Finally, we use the chance to quickly describe the new Rule-based Row Filter node using the new rule engine framework introduced with KNIME 2.9.

A set of simulated data and the workflows used for this study can be downloaded from the KNIME EXAMPLES public server under “008_WebAnalytics_and_OpenStreetMap”, while the KNIME open source platform is downloadable from the KNIME site at www.knime.com.