KNIME and R

The best of two worlds

For questions and suggestions please contact education@knime.com

Overview

Meet: R (Interactive)

The new nodes

Integration with KNIME

A real world application

Why use KNIME and R?

R

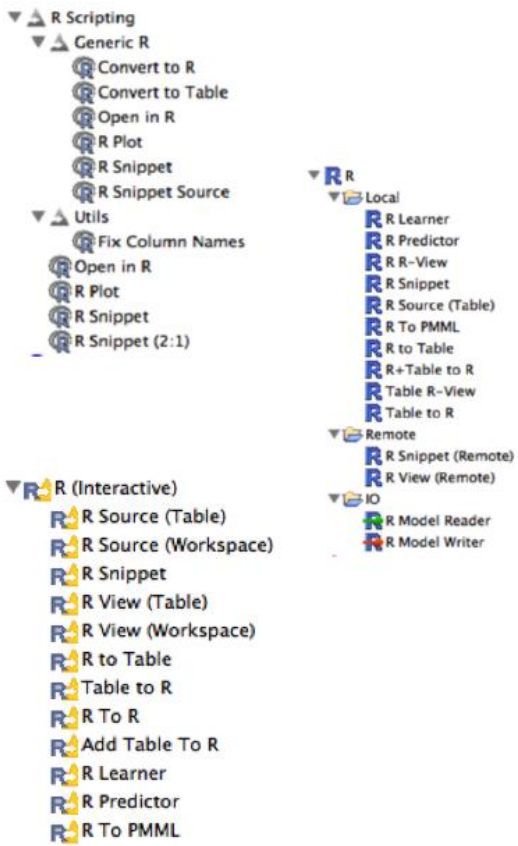
- Powerful statistics
- Leading edge algorithms
- Powerful/flexible graphics
- Widely accepted language

- Open source analytics
- Cross platform
- Vibrant communities

KNIME

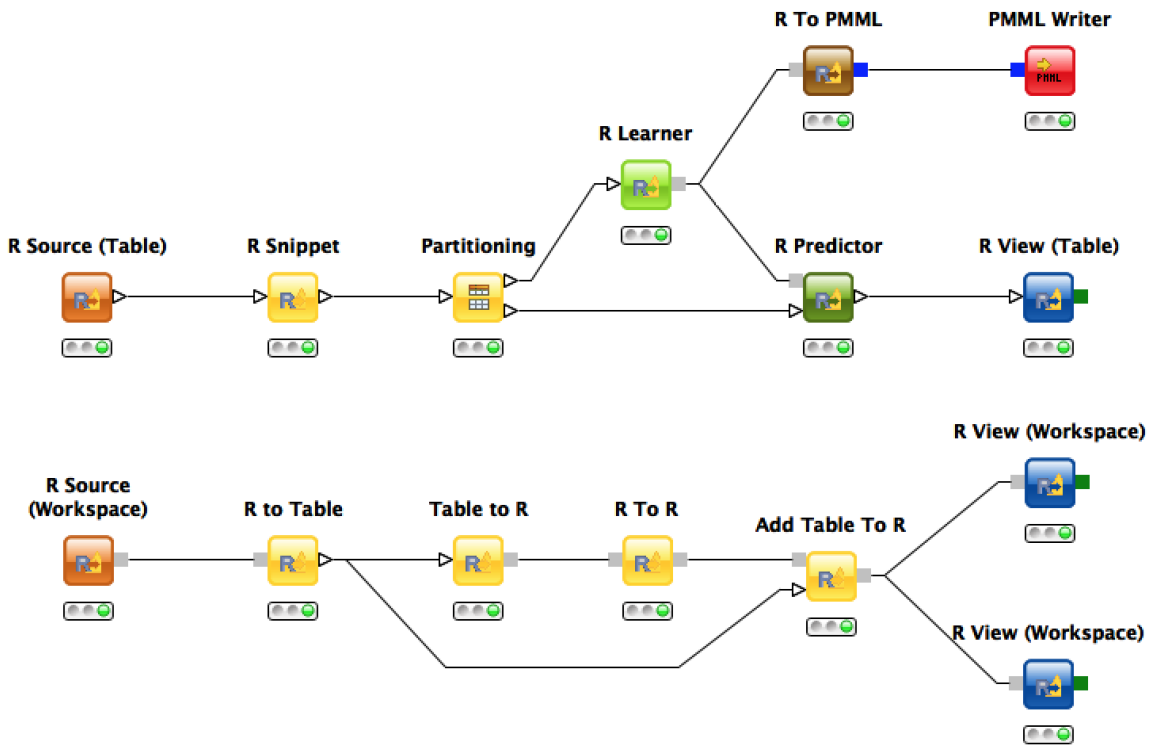
- Powerful user interface
- Designed for big data
- Integrates com and org tools
- Enterprise grade solutions

R in KNIME: 3 ways to play…



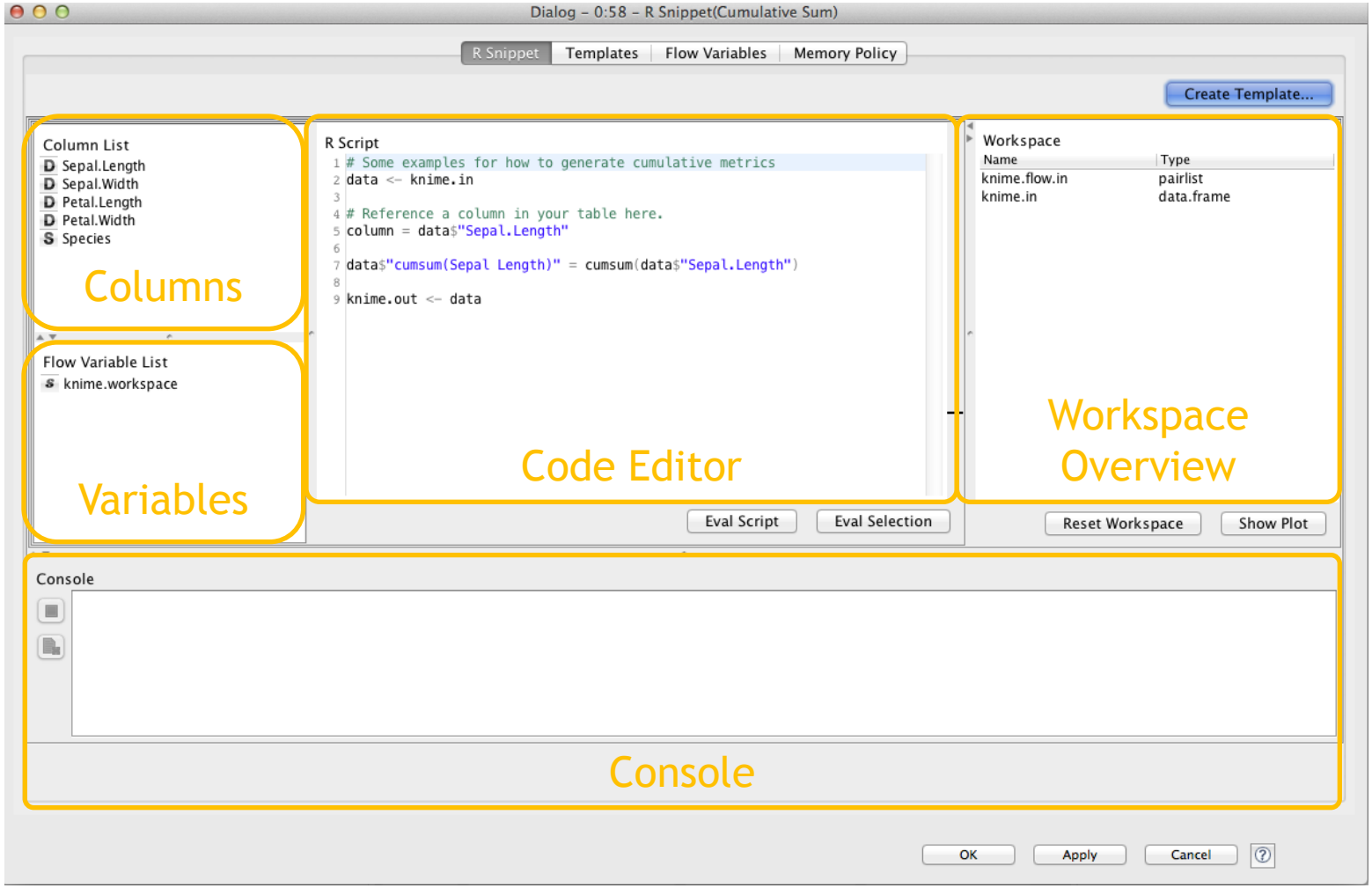
- Community (RServe Integration)
- Core (Deprecated soon)
- R Interactive (Today's topic)

Overview of R (Interactive)

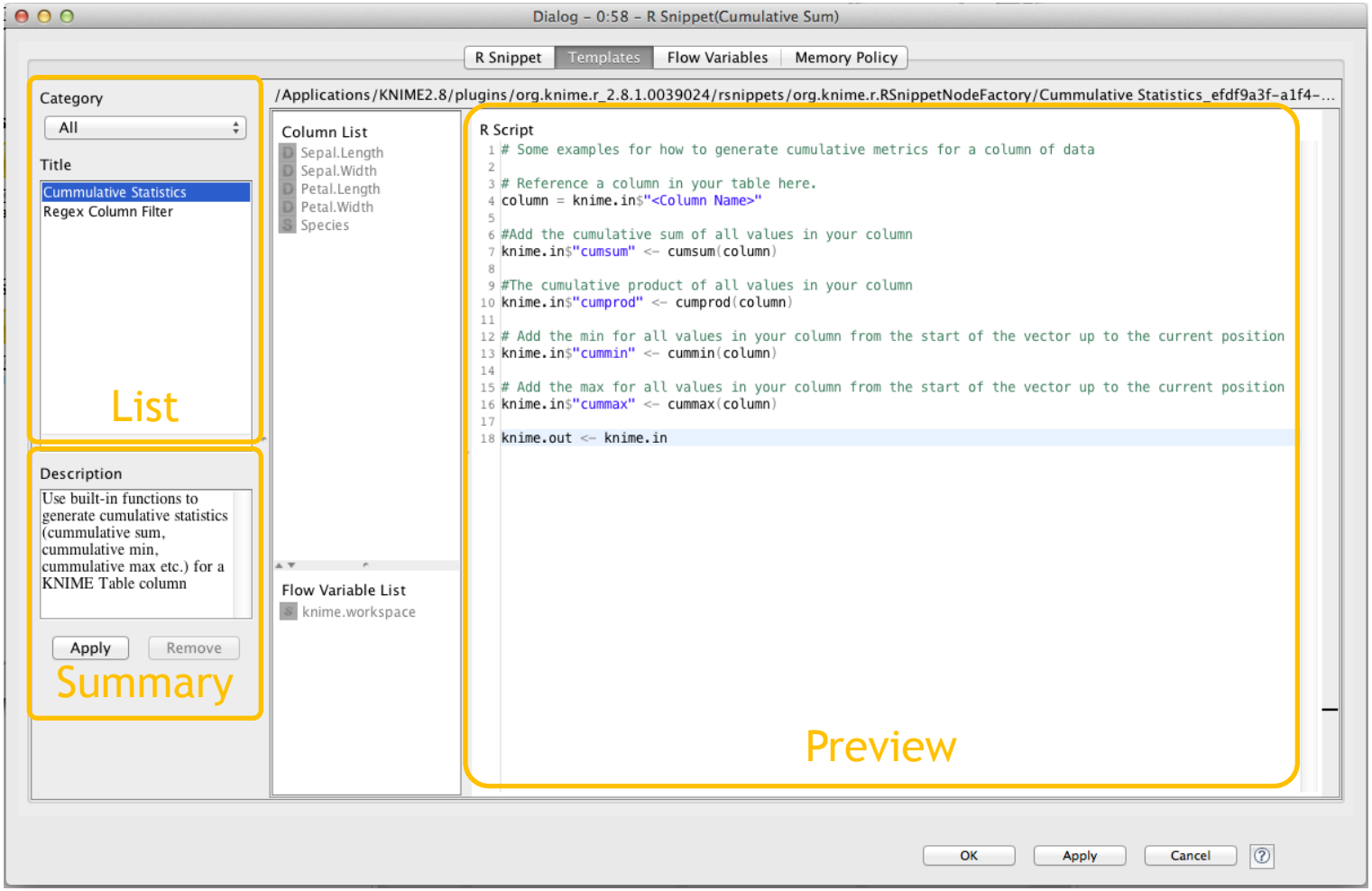


- Different input and output options
- Grey ports enable workspace branching

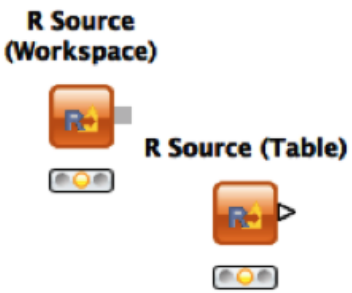
The Interactive Editor



Templates

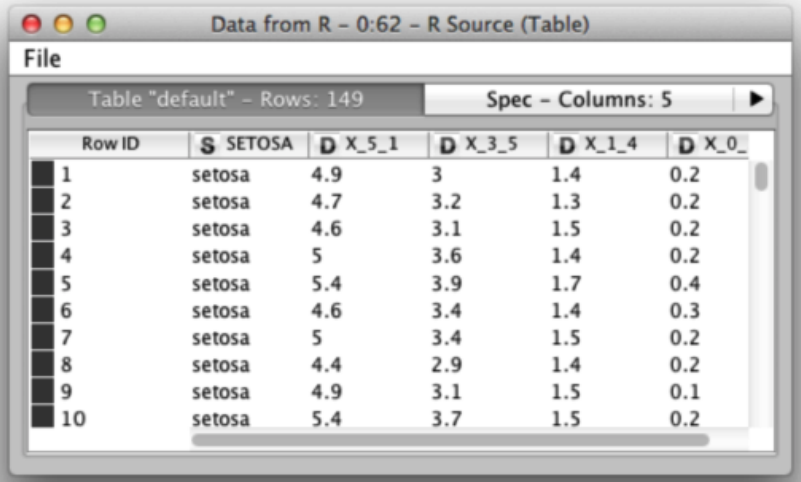


Node: R Source



R Script

```
# The foreign library provides access to many 3rd party data formats.
# Just a few examples are listed below, many others exist.
# More details cran.r-project.org/web/packages/foreign/foreign.pdf

library(foreign)

# map filepath from a flow variable here.
path = "/Users/knime/Desktop/iris.stx"

# Read SAS XPORT
data = read.xport(path)

knime.out <- data
```



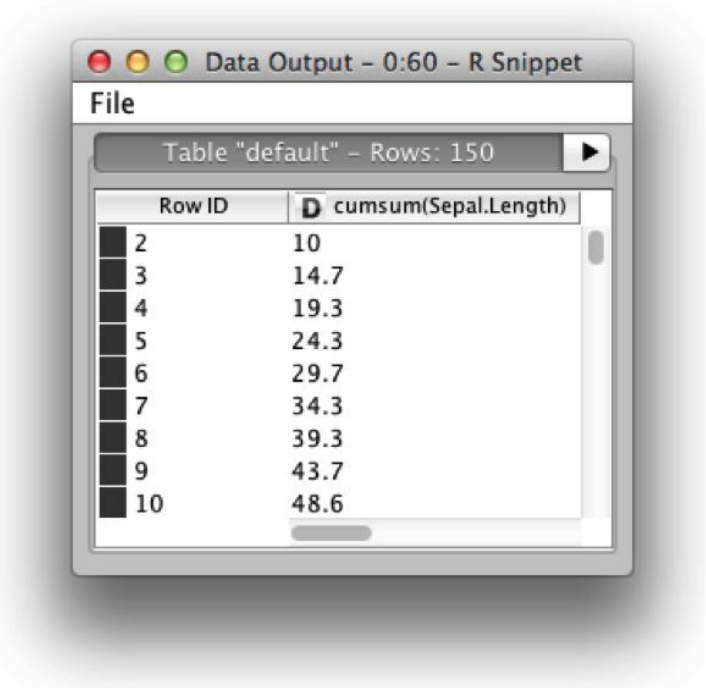
Row ID	S SETOSA	D X_5_1	D X_3_5	D X_1_4	D X_0_
1	setosa	4.9	3	1.4	0.2
2	setosa	4.7	3.2	1.3	0.2
3	setosa	4.6	3.1	1.5	0.2
4	setosa	5	3.6	1.4	0.2
5	setosa	5.4	3.9	1.7	0.4
6	setosa	4.6	3.4	1.4	0.3
7	setosa	5	3.4	1.5	0.2
8	setosa	4.4	2.9	1.4	0.2
9	setosa	4.9	3.1	1.5	0.1
10	setosa	5.4	3.7	1.5	0.2

- Get data from an R data frame
- Assign output to knime.out
- Use with foreign, RCurl, or ...

Node: R Snippet

R Script

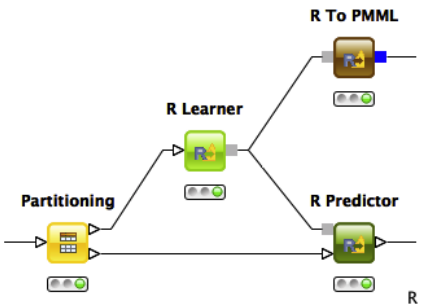
```
# Reference a column in your table here.
column = knime.in$"Sepal.Length"

data = knime.in
#Add the cumulative sum of all values in your column
data$"cumsum(Sepal.Length)" <- cumsum(column)

knime.out <- data
```



Row ID	cumsum(Sepal.Length)
2	10
3	14.7
4	19.3
5	24.3
6	29.7
7	34.3
8	39.3
9	43.7
10	48.6

- Generic data manipulation
- Derive knime.out from knime.in
- Use with grep(), plyr, or ...

Nodes: R Mining

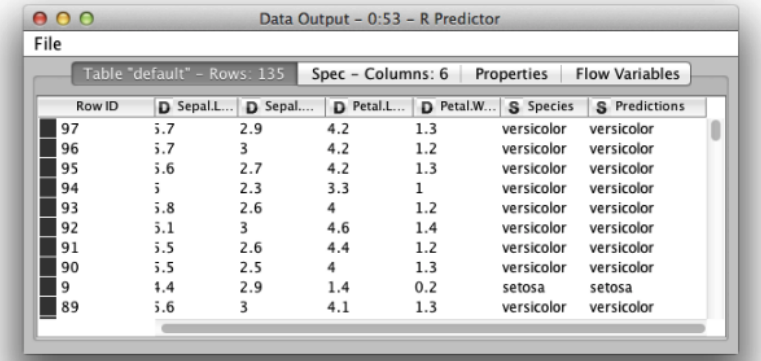


R Script

```
library(rpart)

# use knime.in here and fix class column
data <- iris

# Send model to knime.model
knime.model <- rpart(data$"Species" ~ ., data[1:(ncol(data) - 1)])
```

R Script

```

# Make predictions
Predictions <-  predict(knime.model, knime.in, type = "class")

# Append them to the original table
knime.out <- cbind(knime.in, Predictions)
```

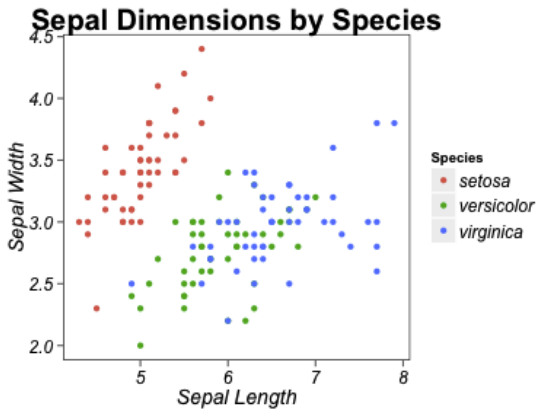


Row ID	Sepal.L...	Sepal....	Petal.L...	Petal.W...	Species	Predictions
97	5.7	2.9	4.2	1.3	versicolor	versicolor
96	5.7	3	4.2	1.2	versicolor	versicolor
95	5.6	2.7	4.2	1.3	versicolor	versicolor
94	5	2.3	3.3	1	versicolor	versicolor
93	5.8	2.6	4	1.2	versicolor	versicolor
92	5.1	3	4.6	1.4	versicolor	versicolor
91	5.5	2.6	4.4	1.2	versicolor	versicolor
90	5.5	2.5	4	1.3	versicolor	versicolor
9	4.4	2.9	1.4	0.2	setosa	setosa
89	5.6	3	4.1	1.3	versicolor	versicolor

- Use R models in KNIME
- Learner & Predictor motif
- PMML support for portability

Nodes: R View

R View (Table)

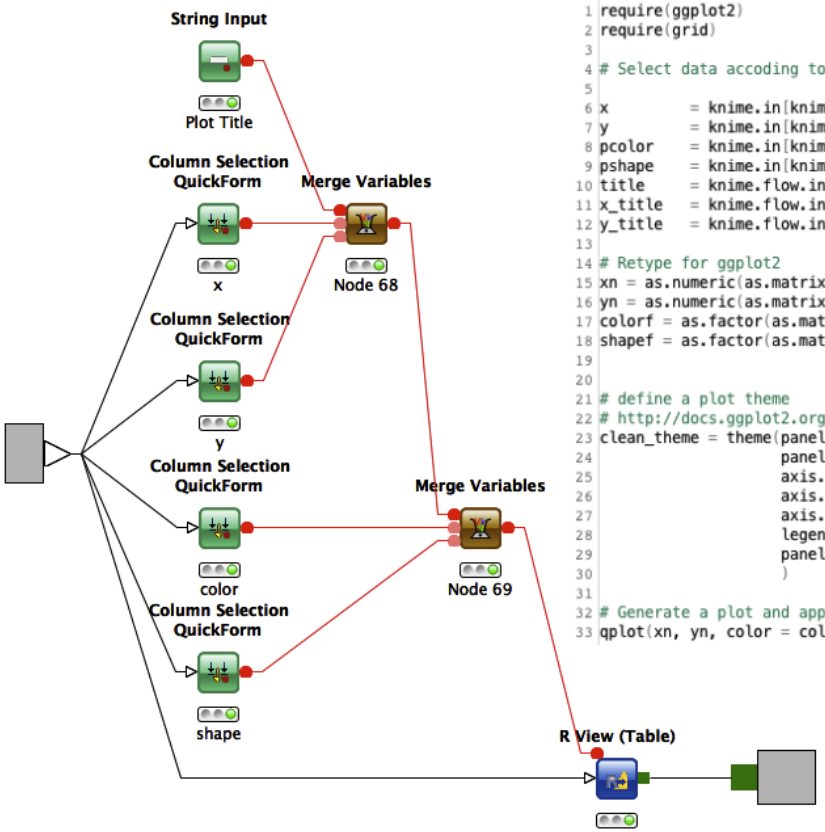


R Script

```
require(ggplot2)
require(grid)

# Insert data references here.
data = iris

x = data$Sepal.Length
x_label = "Sepal Length"

y = data$Sepal.Width
y_label = "Sepal Width"

#Column for coloring:
class = data$Species
legend_title = "Species"

# A plot title
title = "Sepal Dimensions by Species"

# define a plot theme: http://docs.ggplot2.org/0.9.2.1/theme.html for more options
clean_theme = theme(panel.background = element_blank(),
                    plot.title = element_text(size=20, face="bold", colour = "black"),
                    panel.border = element_rect(color = "black", linetype = "solid", fill = "transparent"),
                    axis.title.x = element_text(size=14, face="italic", colour = "black"),
                    axis.title = element_text(size=14, face="italic", colour = "black"),
                    axis.text = element_text(size=12, face="italic", colour = "black"),
                    legend.text = element_text(size=12, face="italic", colour = "black"),
                    panel.grid = element_blank()
                    )

#Define some labels
labels = labs(list(title = title, x = x_label, y = y_label, color = legend_title))

# Generate a plot
plot = qplot(x, y, color = class, main = title)

# Apply theme and labels
plot + labels + clean_theme
```

- Generic R plots
- Plot(knime.in)
- Use with many packages including ggplot2

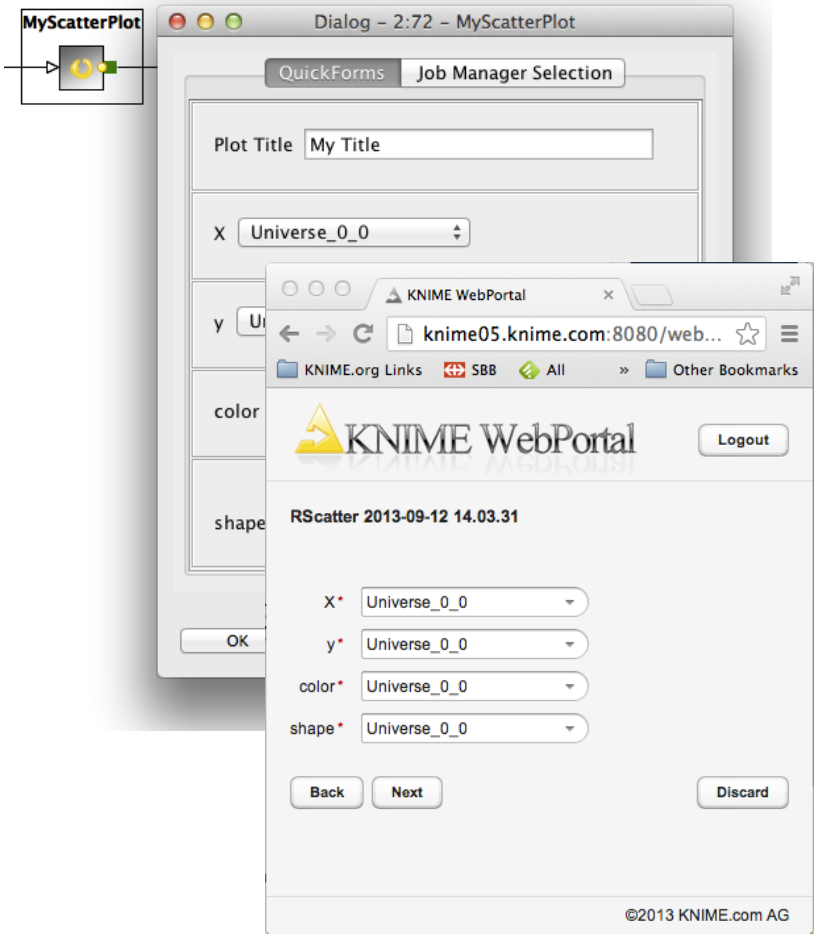
Metanodes and R: Quickforms



R Script

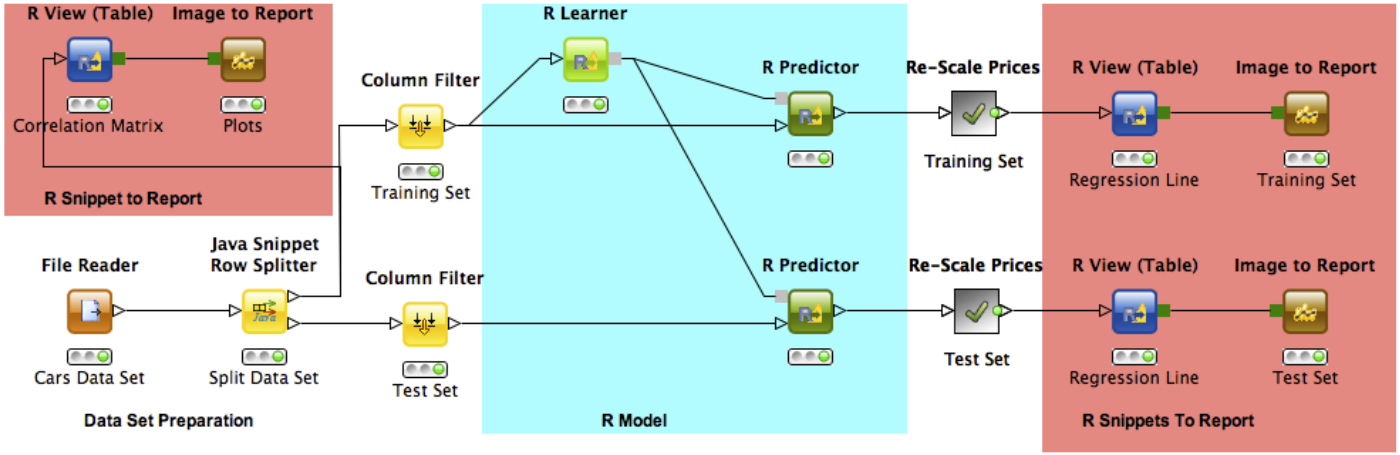
```
1  require(ggplot2)
2  require(grid)
3
4  # Select data accoding to quicforms here.
5
6  x         = knime.in[knime.flow.in[["x"]]]
7  y         = knime.in[knime.flow.in[["y"]]]
8  pcolor    = knime.in[knime.flow.in[["color"]]]
9  pshape    = knime.in[knime.flow.in[["shape"]]]
10 title     = knime.flow.in[["title"]]
11 x_title   = knime.flow.in[["x"]]
12 y_title   = knime.flow.in[["y"]]
13
14 # Retype for ggplot2
15 xn = as.numeric(as.matrix(x))
16 yn = as.numeric(as.matrix(y))
17 colorf = as.factor(as.matrix(pcolor))
18 shapef = as.factor(as.matrix(pshape))
19
20
21 # define a plot theme
22 # http://docs.ggplot2.org/0.9.2.1/theme.html for more options
23 clean_theme = theme(panel.background = element_blank(),
24                     panel.border = element_rect(color = "black", linetype = "solid", fill = "transparent"),
25                     axis.title.x = element_text(size=14, face="italic", colour = "black"),
26                     axis.title = element_text(size=14, face="italic", colour = "black"),
27                     axis.text = element_text(size=12, face="italic", colour = "black"),
28                     legend.text = element_text(size=12, face="italic", colour = "black"),
29                     panel.grid = element_blank()
30                     )
31
32 # Generate a plot and apply the theme
33 qplot(xn, yn, color = colorf, shape = shapef, xlab = x_title , ylab = y_title, main = title) + clean_theme
```

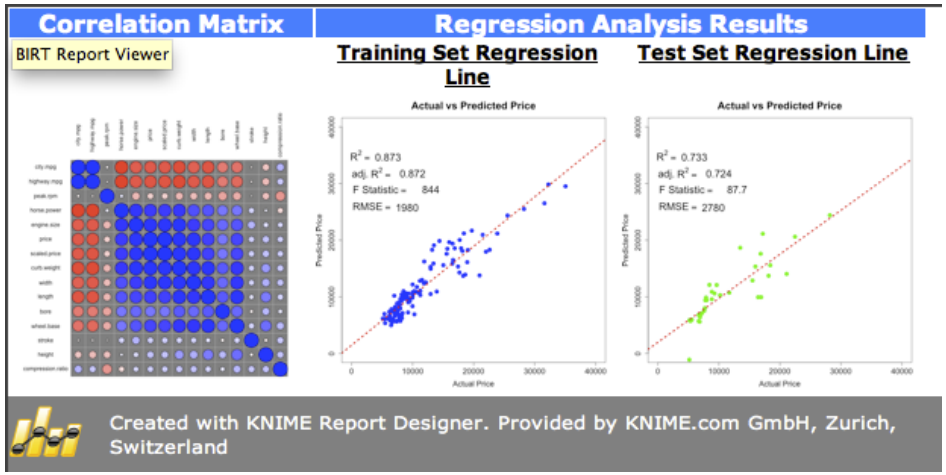
Metanodes and R: Deployment



- Configure yields simple dialog
- Share (Email/TeamSpace/Server)
- Deploy to web (KNIME Webportal)

Embedding plots in BIRT





- Genearate plots in R
- Send to BIRT

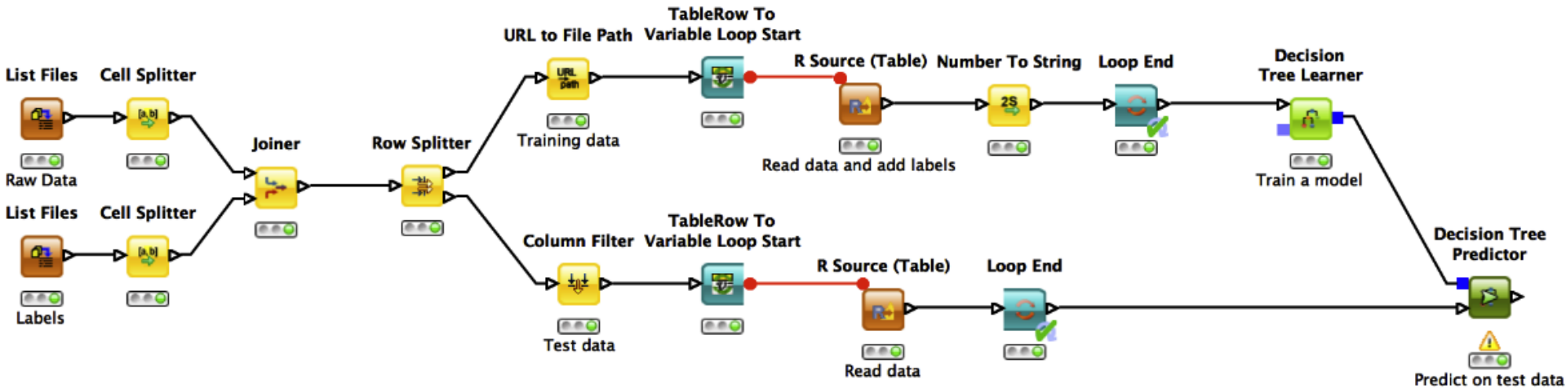
EQPOL Data with Bioconductor I

- External Quality Assurance Program Oversight Laboratory
- NIH, NIAID, DAIDS program for QA of HIV/AIDS research
- Can machine learning automate some manual analysis?
- Problem: Lots of real data (~100,000,000 rows)
- Bioconductor provides flowCore to make this easier

R Script

```
library(flowCore)

# Read data
fcs = read.FCS(knime.flow.in[["Location"]])
labels = read.csv(knime.flow.in[["File path"]], header = FALSE)

#An exotic transform:
# Estimate parameters
lgcl = estimateLogicle(fcs,c("FITC", "PE", "APC", "PerCP"))

#Apply
logiclefcs = transform(fcs,lgcl)

f = as.data.frame(exprs(fcs))
# Format and send data to knime.out
df <- cbind(f,labels)
knime.out <- df
```

EQPOL Data with Bioconductor II



Thank you

education@knime.com