package com.twitter.simclusters\_v2.scalding.common

import com.twitter.algebird.\_

import com.twitter.scalding.typed.TypedPipe

import com.twitter.scalding.{Execution, Stat, UniqueID}

/\*\*

\* A richer version of TypedPipe.

\*/

class TypedRichPipe[V](pipe: TypedPipe[V]) {

def count(counterName: String)(implicit uniqueID: UniqueID): TypedPipe[V] = {

val stat = Stat(counterName)

pipe.map { v =>

stat.inc()

v

}

}

/\*\*

\* Print a summary of the TypedPipe with total size and some randomly selected records

\*/

def getSummary(numRecords: Int = 100): Execution[Option[(Long, String)]] = {

val randomSample = Aggregator.reservoirSample[V](numRecords)

// more aggregator can be added here

pipe

.aggregate(randomSample.join(Aggregator.size))

.map {

case (randomSamples, size) =>

val samplesStr = randomSamples

.map { sample =>

Util.prettyJsonMapper

.writeValueAsString(sample)

.replaceAll("\n", " ")

}

.mkString("\n\t")

(size, samplesStr)

}

.toOptionExecution

}

def getSummaryString(name: String, numRecords: Int = 100): Execution[String] = {

getSummary(numRecords)

.map {

case Some((size, string)) =>

s"TypedPipeName: $name \nTotal size: $size. \nSample records: \n$string"

case None => s"TypedPipeName: $name is empty"

}

}

/\*\*

\* Print a summary of the TypedPipe with total size and some randomly selected records

\*/

def printSummary(name: String, numRecords: Int = 100): Execution[Unit] = {

getSummaryString(name, numRecords).map { s => println(s) }

}

}

object TypedRichPipe extends java.io.Serializable {

import scala.language.implicitConversions

implicit def typedPipeToRichPipe[V](

pipe: TypedPipe[V]

)(

implicit uniqueID: UniqueID

): TypedRichPipe[V] = {

new TypedRichPipe(pipe)

}

}