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

import com.twitter.algebird.Monoid

import com.twitter.summingbird.\_

object SummerWithSumValues {

/\*

A common pattern in heron is to use .sumByKeys to aggregate a value in a store, and then continue

processing with the aggregated value. Unfortunately, .sumByKeys returns the existing value from the

store and the delta separately, leaving you to manually combine them.

Example without sumValues:

someKeyedProducer

.sumByKeys(score)(monoid)

.map {

case (key, (existingValueOpt, delta)) =>

// if you want the value that was actually written to the store, you have to combine

// existingValueOpt and delta yourself

}

Example with sumValues:

someKeyedProducer

.sumByKeys(score)(monoid)

.sumValues(monoid)

.map {

case (key, value) =>

// `value` is the same as what was written to the store

}

\*/

implicit class SummerWithSumValues[P <: Platform[P], K, V](

summer: Summer[P, K, V]) {

def sumValues(monoid: Monoid[V]): KeyedProducer[P, K, V] =

summer.mapValues {

case (Some(oldV), deltaV) => monoid.plus(oldV, deltaV)

case (None, deltaV) => deltaV

}

}

}