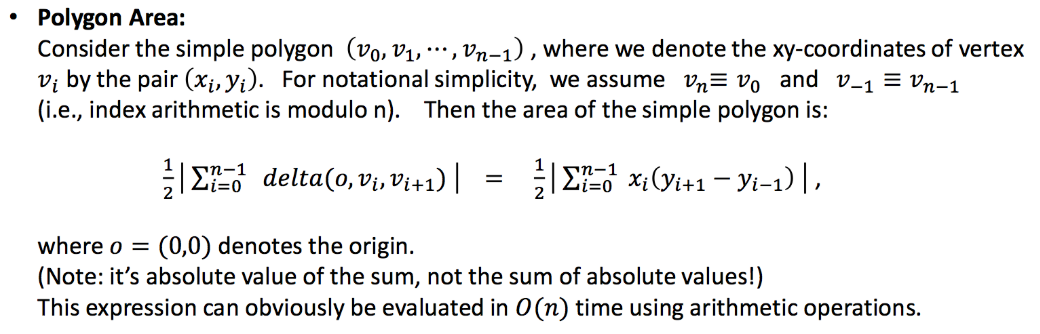
**1. Introduction**

Given the GPS records, time, and weather information, The program calculates the distance traveled, path sinuosity, woodland preference index, convex hull points, convex hull area, and the time slots of the activities for each cow.

1. **Logic**

* All the GPS data are read from the data file, and are stored and **sorted by cowid, date, time**.
* The program is coded by Aditay calculates the distance traveled, path sinuosity, woodland preference index. The logic of this part is hard to follow. Time is needed to understand it.
* The code of the convex hull is calculate following [Andrew's monotone chain convex hull algorithm](https://en.wikibooks.org/wiki/Algorithm_Implementation/Geometry/Convex_hull/Monotone_chain). The function accept the GPS locations of one cow, **the vertices of the convex hull are returned**. The complexity for calculation on **two-dimensional space** is **O(nlog(n))**.
* **The returned points are used to calculate the area of convex hull** by using following formula

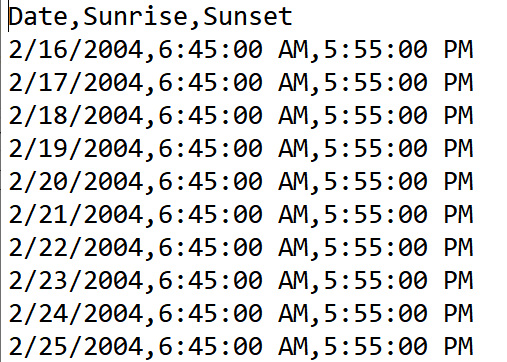


* Given two velocity parameters, one for walking **v\_walking**, one for running **v\_running**. The velocity of a cow is computed by two consecutive GPS records. If the cow's velocity is **[0,v\_walking]**, we assume **the cow was resting**. Similarly, if the cow's velocity is **(v\_walking, v\_running]**, **the cow is assumed that it was walking**. If the speed of the cow is **greater that v\_running**, we think **the cow is running**.
* The function **movementParitionForCow** counts the percentage of each cow that was resting, walking and running during the whole day, **the pre-day, the daytime, and the post-day** according given time date.
* The function **time\_slot** **identifies the activity time slots of the each cow in each day**. The logic need some time to figure out.

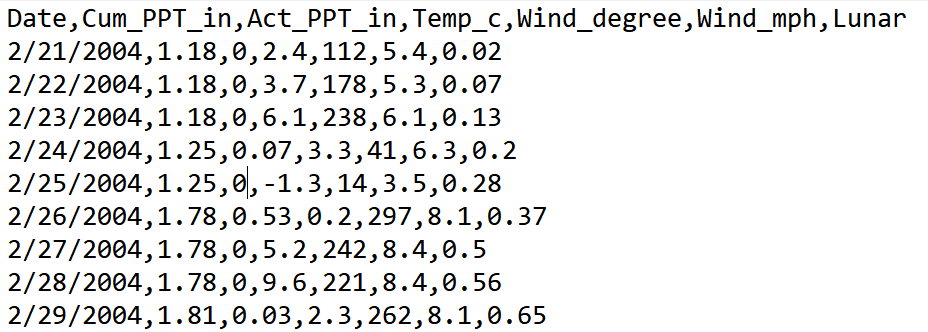
1. **Put the files into the folder**
2. Create a new folder in your Desktop or where you desire to put the Grazepix program.
3. Name the created folder to ”**data/speed**”.
4. Put the jar file ”range project.jar” into the folder.
5. Put files into the folder, which **default names are DataWithPosition.csv, Weather.csv, and Time.csv respectively.**

The format of the file is showed as following:

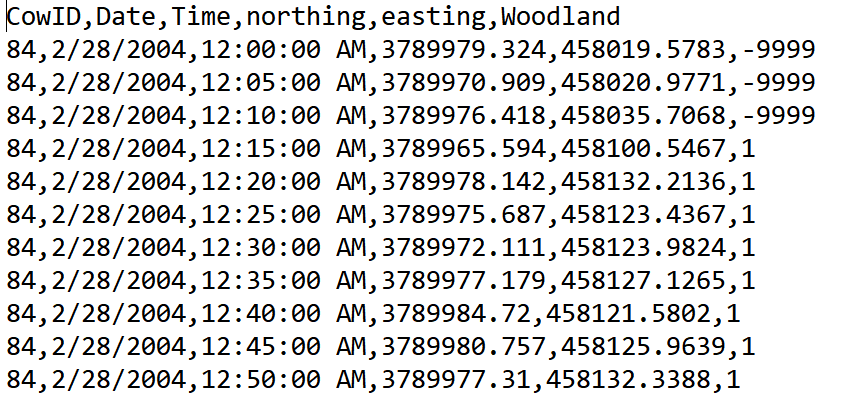
1. Convert your **time data(Time.csv)** into the CSV format and use the comma as the delimiter. In total, the CSV file should have 3 columns of basic information. They are: **Date, Sunrise, Sunset.** The format of Date is **yyyy-MM-dd**. The format of Sunrise and Sunset is **HH:mm:dd AM/PM**.



1. Convert your **weather data(Weather.csv)** into the CSV format and use the comma as the delimiter. In total, the CSV file should have 7 columns of basic information. They are: **Date, Cum\_PPT\_in, Act\_PPT\_in, Temp\_c, Wind\_degree, Wind\_mph, Lunar.** The format of Date is **MM/dd/yyyy**. The Cum\_PPT\_in, Act\_PPT\_in, Temp\_c, Wind\_degree, Wind\_mph, Lunar are **float numbers**.



1. Convert your **position data(DataWithPosition.csv)** into the CSV format and use the comma as the delimiter. In total, the CSV file should have 6 columns of basic information. They are: **CowID, Date, Time, northing, easting, Woodland.** The CowID is the unique id of each cow. It’s could be an integer or a string of text. The format of Date is **MM/dd/yyyy**. The format of time is **HH:mm:dd AM/PM**. The northing and easting are **float numbers** that represents the location of the cow at the time. The **Woodland** are **integer numbers.**



1. **Run The Code**

You could run the code from the Command line on Windows/Linux/MacOS. The results will put in the same folder as the input files and the java program

**The parameters:**

* Input the **position file name** (Default: DataWithPosition.csv)
* Input the **weather file name** (Default: Weather.csv)
* Input the **time file Name** (Default: Time.csv)
* Input the **upper bound of the speed (m/min) of the rest movement** (Default 5 m/min)
* Input the **upper bound of the speed (m/min) of the grazing movement** (Default 15 m/min)
* Input the **threshold of the speed (m/min)** to determine whether one GPS record is active or not (Default: 4 m/min)
* Input **the number of the lag** (number of GPS records need to be considered in previous and following, default is 5)

calculates the distance traveled, woodland preference index, convex hull points, convex hull area

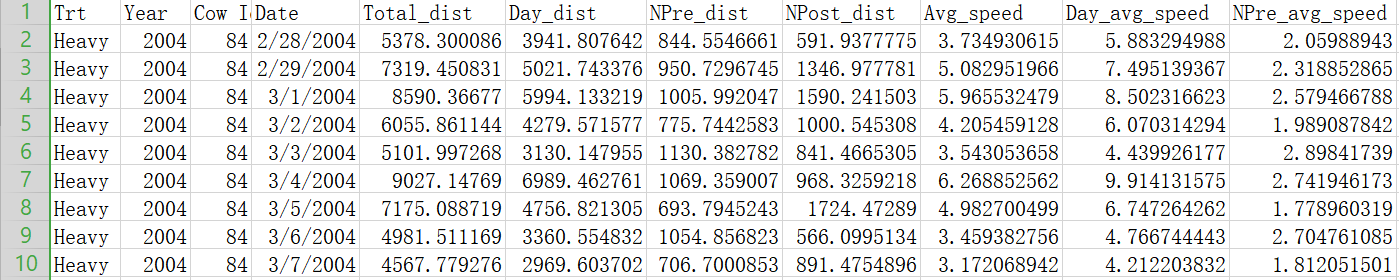
1. **Results of the code**

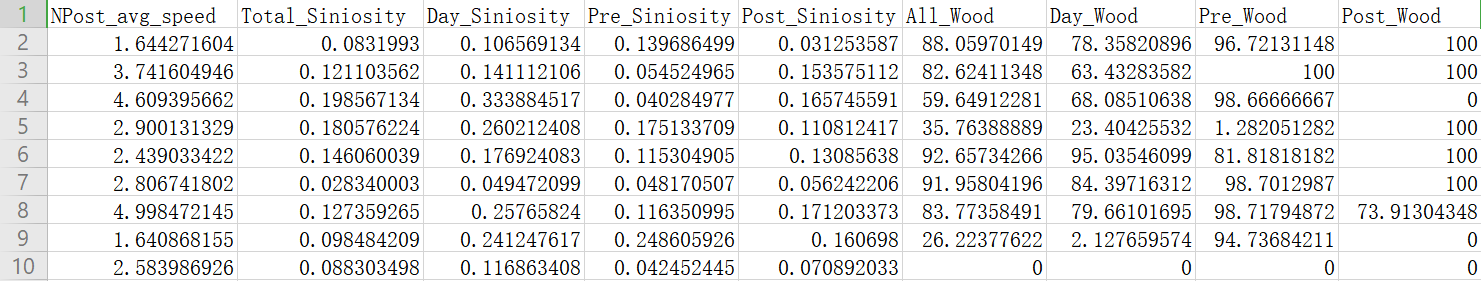
**New animal function:**

* **time\_slots.csv** (the time slots of the activities for each cow, which sorted by CowID)
  + CowID, date, activity\_time\_slots\_list
* **movement\_partition.csv**
  + Cow\_id,Date,r\_pre,g\_pre,t\_pre,r\_day,g\_day,t\_day,r\_post,g\_post,t\_post,r\_total,g\_total,t\_total
    - r\_pre: rest\_pre\_percentage
    - g\_pre: graze\_pre\_percentage
    - t\_pre: travel\_pre\_percentage
    - r\_day: rest\_day\_percentage
    - g\_day: graze\_day\_percentage
    - t\_day: travel\_day\_percentage
    - r\_post: rest\_post\_percentage
    - g\_post: graze\_post\_percentage
    - t\_post: travel\_post\_percentage
    - r\_total: rest\_total\_percentage
    - g\_total: graze\_total\_percentage
    - t\_total: travel\_total\_percentage
* **MCP\_Results.csv**(sorted by Cow\_id, then by Date)
  + Cow\_id,Date,Daily\_period,Easting,Northing
* **MCP\_Area\_Results.csv**(sorted by Cow\_id, then by Date)
  + Cow\_id,Date,Pre\_MCP\_area,Day\_MCP\_Area,Post\_MCP\_Area,All\_Day\_area

**Animal project:**

* **DistanceandSiniosity.csv(**calculates the path sinuosity**)**
  + Trt,Year,CowId,Date,Total\_dist,Day\_dist,NPre\_dist,NPost\_dist,Avg\_speed,Day\_avg\_speed,NPre\_avg\_speed,NPost\_avg\_speed,Total\_Siniosity,Day\_Siniosity,Pre\_Siniosity,Post\_Siniosity,All\_Wood,Day\_Wood,Pre\_Wood,Post\_Wood





* **CompleteProcessedData.csv**
  + Trt,Year,CowId,Date,Total\_dist,Day\_dist,NPre\_dist,NPost\_dist,Avg\_speed,Day\_avg\_speed,NPre\_avg\_speed,NPost\_avg\_speed,Total\_Siniosity,Day\_Siniosity,Pre\_Siniosity,Post\_Siniosity,All\_Wood,Day\_Wood,Pre\_Wood,Post\_Wood,Cum\_PPT\_in,Act\_PPT\_in,Temp\_c,Wind\_Degree,Wind\_mph,Lunar

