**WEKA**
of Waikato

🏠  Wiki Home

🕐  Recent Changes

📄  Pages and Files

👤  Members

[ Search ]

Home

All pages

All tags

All files

Packages

FAQ

Not So FAQ

Troubleshooting

Learning Resources

Mailing List

# Creating an ARFF file (/Creating+an+ARFF+file)

| ✏ Edit | 💬 0 (/Creating+an+ARFF+file#discussion) | 🕐 6 (/page/history/Creating+an+ARFF+file) |

| … (/page/menu/Creating+an+ARFF+file) |

Table of Contents
[Missing values](#)

[Downloads](#)

[See also](#)

The following code generates an Instances object and outputs it to stdout as [ARFF](#) file.

It generates the following types of attributes:

- numeric
- nominal
- string
- date
- relational

Example class **AttTest**:

```
import weka.core.Attribute;
import weka.core.FastVector;
import weka.core.Instance;
import weka.core.Instances;
```

```java
/**
 * Generates a little ARFF file with different attribute types.
 *
 * @author FracPete
 */
public class AttTest {
  public static void main(String[] args) throws Exception {
    FastVector      atts;
    FastVector      attsRel;
    FastVector      attVals;
    FastVector      attValsRel;
    Instances       data;
    Instances       dataRel;
    double[]        vals;
    double[]        valsRel;
    int             i;

    // 1. set up attributes
    atts = new FastVector();
    // - numeric
    atts.addElement(new Attribute("att1"));
    // - nominal
    attVals = new FastVector();
    for (i = 0; i < 5; i++)
      attVals.addElement("val" + (i+1));
    atts.addElement(new Attribute("att2", attVals));
    // - string
    atts.addElement(new Attribute("att3", (FastVector) null));
    // - date
    atts.addElement(new Attribute("att4", "yyyy-MM-dd"));
    // - relational
    attsRel = new FastVector();
    // -- numeric
    attsRel.addElement(new Attribute("att5.1"));
    // -- nominal
    attValsRel = new FastVector();
```

```
      for (i = 0; i < 5; i++)
        attVaisRei.addElement("val5." + (i+1));
      attsRel.addElement(new Attribute("att5.2", attValsRel));
      dataRel = new Instances("att5", attsRel, 0);
      atts.addElement(new Attribute("att5", dataRel, 0));

      // 2. create Instances object
      data = new Instances("MyRelation", atts, 0);

      // 3. fill with data
      // first instance
      vals = new double[data.numAttributes()];
      // - numeric
      vals[0] = Math.PI;
      // - nominal
      vals[1] = attVals.indexOf("val3");
      // - string
      vals[2] = data.attribute(2).addStringValue("This is a string!");
      // - date
      vals[3] = data.attribute(3).parseDate("2001-11-09");
      // - relational
      dataRel = new Instances(data.attribute(4).relation(), 0);
      // -- first instance
      valsRel = new double[2];
      valsRel[0] = Math.PI + 1;
      valsRel[1] = attValsRel.indexOf("val5.3");
      dataRel.add(new Instance(1.0, valsRel));
      // -- second instance
      valsRel = new double[2];
      valsRel[0] = Math.PI + 2;
      valsRel[1] = attValsRel.indexOf("val5.2");
      dataRel.add(new Instance(1.0, valsRel));
      vals[4] = data.attribute(4).addRelation(dataRel);
      // add
      data.add(new Instance(1.0, vals));

      // second instance
```

```
    vals = new double[data.numAttributes()]; // important: needs NEW array!
    // - numeric
    vals[0] = Math.E;
    // - nominal
    vals[1] = attVals.indexOf("val1");
    // - string
    vals[2] = data.attribute(2).addStringValue("And another one!");
    // - date
    vals[3] = data.attribute(3).parseDate("2000-12-01");
    // - relational
    dataRel = new Instances(data.attribute(4).relation(), 0);
    // -- first instance
    valsRel = new double[2];
    valsRel[0] = Math.E + 1;
    valsRel[1] = attValsRel.indexOf("val5.4");
    dataRel.add(new Instance(1.0, valsRel));
    // -- second instance
    valsRel = new double[2];
    valsRel[0] = Math.E + 2;
    valsRel[1] = attValsRel.indexOf("val5.1");
    dataRel.add(new Instance(1.0, valsRel));
    vals[4] = data.attribute(4).addRelation(dataRel);
    // add
    data.add(new Instance(1.0, vals));

    // 4. output data
    System.out.println(data);
  }
 }
```

## Missing values

By default, a new double array will be initialized with 0s. In case you want to be a value missing at a certain position, you have to explicitly set the *missing value* via the missingValue() method of the weka.core.Instance class. In Weka > 3.7.1 Instance is an interface, so missingValue() moved into weka.core.Utils. In case you already have an existing

weka.core.Instance object, then you use its setMissing(int) method which sets a missing value at the given position. Here are examples, which set the **third** attribute to missing:

- double array:

```
doubie[] vais = ...  // from somewhere, e.g., from AttTest.java exampie
vals[2] = Instance.missingValue(); // or ... = Utils.missingValue() for Weka > 3.7.1
```

- weka.core.Instance object:

```
double[] vals = ... // from somewhere, e.g., from AttTest.java example
Instance inst = new Instance(1.0, vals);
inst.setMissing(2);
```

## Downloads

- ☐ **AttTest.java**
  Details  Download  3 KB
  the above class
  (book , stable-3.6 , developer )

## See also

- Save Instances to an ARFF File - if you want to save the data to a file instead of printing it to stdout
- Adding attributes to a dataset - shows how to add attributes to an existing dataset
- ARFF format

Help · About · Pricing · Privacy · Terms · **Support** · **Upgrade**