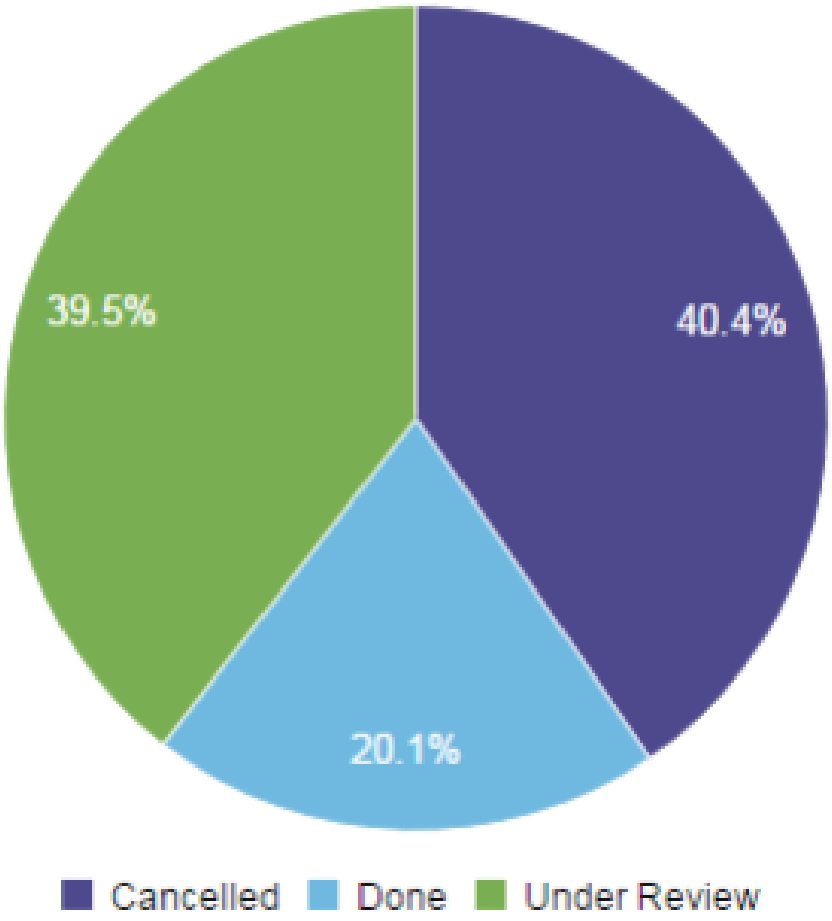
Issue chart

The issue chart shows the number of issues over time and you can filter by using JQL and group by time unit or (custom) field.

Chart preview



Parameters

Parameter		Type	Default Value
UI	Code		
JQL	`JQL`	JQL Autocomplete	
Value	`ValuePicker`	Value Field Picker	IssueCount
Time Period	`PeriodPicker`	Time Period Picker	Month
Group By	`GroupBy`	Group By Picker	Issue Type

Chart Type	ChartType	Chart Type Picker	Bar
Options • Accumulate Groups • Accumulate Values	AccumulateGroups	Boolean	True
	AccumulateValues	Boolean	True

Layout Script

```
c3.generate({
    data: chartData,
    axis: {
        x: {
            type: 'timeseries',
            label: {
                text: 'Months',
                position: 'outer-left'
            },
            tick: {
                format: '%B', // shows month as word, see link for more formatting examples: https://bl.ocks.
org/zanarmstrong/raw/ca0adb7e426c12c06a95/
                culling: {
                    max: 25
                },
                fit: true,
                multiline: false
            }
        },
        y: {
            label: 'Tickets'
        }
    }
})
```

Data Script

```
import java.lang.reflect.Field;
import java.math.BigDecimal;

import java.lang.reflect.Field;
import java.math.BigDecimal;

import org.apache.lucene.document.Document;

import com.atlassian.jira.component.ComponentAccessor;
import com.atlassian.jira.issue.DocumentIssueImpl;
import com.atlassian.jira.issue.Issue;
import com.atlassian.jira.jql.parser.JqlParseException;
import com.atlassian.jira.jql.parser.JqlQueryParser;
import com.atlassian.query.Query;
import com.decadis.jira.xchart.api.ChartParam;
import com.decadis.jira.xchart.api.model.Period;
import com.decadis.jira.xchart.api.util.DateUtils;

JqlQueryParser jqlQueryParser = ComponentAccessor.getComponent(JqlQueryParser.class);
Query query = null;
try
{
  query = jqlQueryParser.parseQuery(JQL);
} catch (JqlParseException e)
{
  throw new IllegalArgumentException("Bad JQL: " + JQL);
}
```

```
Period selectedPeriod = Period.fromString(PeriodPicker);
def countGroup = chartBuilder.newDataCollector();
def groupValueExtractor = chartBuilder.getGrouper(GroupBy);

def valueExtractor = chartBuilder.getValueExtractor(ValuePicker);
if ( valueExtractor == null )
{
  throw new IllegalArgumentException("Valuepicker for " + Value + " is not supported.");
}

Field documentField;
try
{
  documentField = DocumentIssueImpl.class.getDeclaredField("document");
  documentField.setAccessible(true);

  for ( Issue issue : chartBuilder.getFilterUtils().performSearch(query, user) )
  {
    def document = (Document) documentField.get(issue);
    BigDecimal value = valueExtractor.get(issue, document);

    for ( String group : groupValueExtractor.getGroups((Document) documentField.get(issue)) )
    {
      countGroup.addValue(value, group, dateUtils.getPeriodGroup(issue.getCreated(), selectedPeriod));
    }
  }
} catch (Exception e)
{ e.getStackTrace(); }

if ( AccumulateValues ) {
    countGroup.accumulateGroups();
}
else {
    countGroup.fillMissingValues();
}

def chartData = chartBuilder.newChartData("Issues");
for ( String grpKey : countGroup.keySet() )
{
  chartData.addGroupName(grpKey, groupValueExtractor.getResolvedValue(grpKey, null));
}

chartData.setxFormat(DateUtils.SimpleDateFormatD3);
chartData.setPeriod(selectedPeriod);
chartData.setType(ChartType);

chartBuilder.getChartUtil().transformResult(countGroup, chartData, AccumulateGroups);

return chartData;
```

If you still have questions, feel free to refer to our support team.