Introduction

some stand in intro text

1 xAPI Formal Specification

The current formal specification only defines xAPI statements abstractly within the context of Z. A concrete definition for xAPI statements is outside the scope of this document.

1.1 Basic Types

$IFI ::= mbox \mid mbox\_sha1sum \mid openid \mid account$

- Type unique to Agents and Groups, The concrete definition of the listed values is outside the scope of this specification

$OBJECTTYPE ::= Agent \mid Group \mid SubStatement \mid StatementRef \mid Activity$

- A type which can be present in all activities as defined by the xAPI specification

$INTERACTIONTYPE ::= true-false \mid choice \mid fill-in \mid long-fill-in \mid matching \mid performance \mid sequencing \mid likert \mid numeric \mid other$

- A type which represents the possible interactionTypes as defined within the xAPI specification

$INTERACTIONCOMPONENT ::= choices \mid scale \mid source \mid target \mid steps$

- A type which represents the possible interaction components as defined within the xAPI specification

- the concrete definition of the listed values is outside the scope of this specification

$CONTEXTTYPES ::= parent \mid grouping \mid category \mid other$

- A type which represents the possible context types as defined within the xAPI specification

$[STATEMENT]$

- Basic type for an xAPI data point

$[AGENT, GROUP]$

- Basic types for Agents and collections of Agents

1.2 Id Schema

$$\begin{array}{|l}
\multicolumn{1}{l}{\_\_ Id \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
id : \mathbb{F}_1 \#1 \\
\hline
\end{array}$$

- the schema Id introduces the component id which is a non-empty, finite set of 1 value

1.3 Schemas for Agents, Groups and Actors

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Agent}} \\
\hline
agent : AGENT \\
objectType : OBJECTTYPE \\
name : \mathbb{F}_1 \, \#1 \\
ifi : IFI \\
\hline
objectType = Agent \\
agent = \{ifi\} \cup \mathbb{P}\{name, objectType\} \\
\hline
\end{array}$$

- The schema *Agent* introduces the component *agent* which is a set consisting of an ifi and optionally an *objectType* and/or *name*

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Member}} \\
\hline
Agent \\
member : \mathbb{F}_1 \\
\hline
member = \{a : AGENT \mid \forall a_n : a_i..a_j \bullet i \leq n \leq j \bullet a = agent\} \\
\hline
\end{array}$$

- The schema $Member$ introduces the component *member* which is a set of objects a, where for every a within $a_0..a_n$, a is an *agent*

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Group}} \\
\hline
Member \\
group : GROUP \\
objectType : OBJECTTYPE \\
ifi : IFI \\
name : \mathbb{F}_1 \, \#1 \\
\hline
objectType = Group \\
group = \{objectType, name, member\} \vee \{objectType, member\} \vee \\
\qquad\qquad \{objectType, ifi\} \cup \mathbb{P}\{name, member\} \\
\hline
\end{array}$$

- The schema *Group* introduces the component *group* which is of type *GROUP* and is a set of either *objectType* and *member* with optionaly *name* or *objectType* and ifi with optionally *name* and/or *member*

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Actor}} \\
\hline
Agent \\
Group \\
actor : AGENT \vee GROUP \\
\hline
actor = agent \vee group \\
\hline
\end{array}$$

- The schema *Actor* introduces the component *actor* which is either an *agent* or *group*

1.4 Verb Schema

$$\begin{array}{|l}
\multicolumn{1}{l}{Verb} \\ \hline
Id \\
display, verb : \mathbb{F}_1 \\ \hline
verb = \{id, display\} \lor \{id\} \\ \hline
\end{array}$$

- The schema $Verb$ introduces the component $verb$ which is a set that consists of either id and the non-empty, finite set $display$ or just id

1.5 Object Schema

$$\begin{array}{|l}
\multicolumn{1}{l}{Extensions} \\ \hline
extensions, extensionVal : \mathbb{F}_1 \\
extensionId : \mathbb{F}_1 \#1 \\ \hline
extensions = \{e : (extensionId, extensionVal)\ | \forall e_n : e_i..e_j \bullet i \leq n \leq j \bullet \\
\qquad (extensionId_i, extensionVal_i) \lor (extensionId_i, extensionVal_j) \land \\
\qquad (extensionId_j, extensionVal_i) \lor (extensionId_j, extensionVal_j) \land \\
\qquad extensionId_i \neq extensionId_j\} \\ \hline
\end{array}$$

- The schema $Extensions$ introduces the component $extensions$ which is a non-empty, finite set that consists of ordered pairs of $extensionId$ and $extensionVal$. Different $extensionId$s can have the same $extensionVal$ but there can not be two identical $extensionId$ values
- $extensionId$ is a non-empty, finite set with one value
- $extensionVal$ is a non-empty, finite set

$$\begin{array}{|l}
\multicolumn{1}{l}{InteractionActivity} \\ \hline
interactionType : INTERACTIONTYPE \\
correctResponsePattern : \mathrm{seq}_1 \\
interactionComponent : INTERACTIONCOMPONENT \\ \hline
interactionActivity = \{interactionType, correctReponsePattern, interactionComponent\} \lor \\
\qquad \{interactionType, correctResponsePattern\} \\ \hline
\end{array}$$

- The schema $InteractionActivity$ introduces the component $interactionActivity$ which is a set of either $interactionType$ and $correctResponsePattern$ or $interactionType$ and $correctResponsePattern$ and $interactionComponent$

$$
\begin{array}{|l}
\multicolumn{1}{l}{Definition} \\
\hline
InteractionActivity \\
Extensions \\
definition, name, description : \mathbb{F}_1 \\
type, moreInfo : \mathbb{F}_1 \#1 \\
\hline
definition = \mathbb{P}_1 \{name, description, type, moreInfo, extensions, interactionActivity\} \\
\hline
\end{array}
$$

- The schema $Definition$ introduces the component $definition$ which is the non-empty, finite power set of $name$, $description$, $type$, $moreInfo$ and $extensions$

$$
\begin{array}{|l}
\multicolumn{1}{l}{Object} \\
\hline
Id \\
Definition \\
Agent \\
Group \\
Statement \\
objectTypeA, objectTypeS, objectTypeSub, objectType : OBJECTTYPE \\
substatement : STATEMENT \\
object : \mathbb{F}_1 \\
\hline
substatement = statement \\
objectTypeA = Activity \\
objectTypeS = StatementRef \\
objectTypeSub = SubStatement \\
objectType = objectTypeA \vee objectTypeS \\
object = \{id\} \vee \{id, objectType\} \vee \{id, objectTypeA, definition\} \\
\qquad\quad \vee \{id, definition\} \vee \{agent\} \vee \{group\} \vee \{objectTypeSub, substatement\} \\
\qquad\quad \vee \{id, objectTypeA\} \\
\hline
\end{array}
$$

- The schema $Object$ introduces the component $object$ which is a non-empty, finite set of either id, id and $objectType$, id and $objectTypeA$, id and $objectTypeA$ and $definition$, $agent$, $group$, or $substatement$
- The schema $Statement$ and the corresponding component $statement$ will be defined later on in this specification

1.6 Result Schema

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Score}} \\
\hline
\mathit{score} : \mathbb{F}_1 \\
\mathit{scaled}, \mathit{min}, \mathit{max}, \mathit{raw} : \mathbb{Z} \\
\hline
\mathit{scaled} = \{n : \mathbb{Z} \mid -1.0 \leq n \leq 1.0\} \\
\mathit{min} = n < \mathit{max} \\
\mathit{max} = n > \mathit{min} \\
\mathit{raw} = \{n : \mathbb{Z} \mid \mathit{min} \leq n \leq \mathit{max}\} \\
\mathit{score} = \mathbb{P}_1\{\mathit{scaled}, \mathit{raw}, \mathit{min}, \mathit{max}\} \\
\hline
\end{array}$$

- The schema *Score* introduces the component *score* which is the non-empty powerset of *min*, *max*, *raw* and *scaled*

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Result}} \\
\hline
\mathit{Score} \\
\mathit{Extensions} \\
\mathit{success}, \mathit{completion}, \mathit{response}, \mathit{duration} : \mathbb{F}_1 \, \#1 \\
\mathit{result} : \mathbb{F}_1 \\
\hline
\mathit{success} = \{\mathit{true}\} \vee \{\mathit{false}\} \\
\mathit{completion} = \{\mathit{true}\} \vee \{\mathit{false}\} \\
\mathit{result} = \mathbb{P}_1\{\mathit{score}, \mathit{success}, \mathit{completion}, \mathit{response}, \mathit{duration}, \mathit{extensions}\} \\
\hline
\end{array}$$

- The schema *Result* introduces the component *result* which is the non-empty power set of *score*, *success*, *completion*, *response*, *duration* and *extensions*

1.7 Context Schema

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Instructor}} \\
\hline
\mathit{Agent} \\
\mathit{Group} \\
\mathit{instructor} : \mathit{AGENT} \vee \mathit{GROUP} \\
\hline
\mathit{instructor} = \mathit{agent} \vee \mathit{group} \\
\hline
\end{array}$$

- The schema *Instructor* introduces the component *instructor* which can be ether an *agent* or a *group*

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Team}} \\
\hline
\mathit{Group} \\
\mathit{team} : \mathit{GROUP} \\
\hline
\mathit{team} = \mathit{group} \\
\hline
\end{array}$$

- The schema $Team$ introduces the component $team$ which is a $group$

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Context}} \\
\hline
\mathit{Instructor} \\
\mathit{Team} \\
\mathit{Object} \\
\mathit{Extensions} \\
\mathit{registration}, \mathit{revision}, \mathit{platform}, \mathit{language} : \mathbb{F}_1 \#1 \\
\mathit{parentT}, \mathit{groupingT}, \mathit{categoryT}, \mathit{otherT} : \mathit{CONTEXTTYPES} \\
\mathit{contextActivities}, \mathit{statement} : \mathbb{F}_1 \\
\hline
\mathit{statement} = \mathit{object} \setminus (\mathit{id}, \mathit{objectType}, \mathit{agent}, \mathit{group}, \mathit{definition}) \\
\mathit{parentT} = \mathit{parent} \\
\mathit{groupingT} = \mathit{grouping} \\
\mathit{categoryT} = \mathit{category} \\
\mathit{otherT} = \mathit{other} \\
\mathit{contextActivity} = \{\mathit{ca} : \mathit{object} \setminus (\mathit{agent}, \mathit{group}, \mathit{objectType}, \mathit{objectTypeSub}, \mathit{substatement})\} \\
\mathit{contextActivityParent} = (\mathit{parentT}, \mathit{contextActivity}) \\
\mathit{contextActivityCategory} = (\mathit{categoryT}, \mathit{contextActivity}) \\
\mathit{contextActivityGrouping} = (\mathit{groupingT}, \mathit{contextActivity}) \\
\mathit{contextActivityOther} = (\mathit{otherT}, \mathit{contextActivity}) \\
\mathit{contextActivities} = \mathbb{P}_1\{\mathit{contextActivityParent}, \mathit{contextActivityCategory}, \\
\qquad\qquad\qquad\qquad \mathit{contextActivityGrouping}, \mathit{contextActivityOther}\} \\
\mathit{context} = \mathbb{P}_1\{\mathit{registration}, \mathit{instructor}, \mathit{team}, \mathit{contextActivities}, \mathit{revision}, \\
\qquad\qquad \mathit{platform}, \mathit{language}, \mathit{statement}, \mathit{extensions}\} \\
\hline
\end{array}$$

- The schema $Context$ introduces the component $context$ which is the non-empty powerset of $registration$, $instructor$, $team$, $contextActivities$, $revision$, $platform$, $language$, $statement$ and $extensions$

- The notation $object \setminus agent$ represents the component $object$ except for its subcomponent $agent$

1.8 Timestamp and Stored Schema

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Timestamp}} \\
\hline
\mathit{timestamp} : \mathbb{F}_1 \#1 \\
\hline
\end{array}$$

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{Stored}} \\
\hline
\mathit{stored} : \mathbb{F}_1 \#1 \\
\hline
\end{array}$$

- The schema $Timestamp$ and $stored$ introduce the components $timestamp$ and $stored$ respectively. Each are non-empty, finite sets containing one value

1.9 Attachements Schema

Attachments

$$\begin{array}{l}
display, description, attachment, attachments : \mathbb{F}_1 \\
usageType, sha2, fileUrl, contexntType : \mathbb{F}_1 \#1 \\
length : \mathbb{N} \\
\hline
attachment = \{usageType, display, contentType, length, sha2\} \cup \mathbb{P}\{description, fileUrl\} \\
attachments = \{a : attachment\}
\end{array}$$

- The schema *Attachements* introduces the componenet *attachements* which is a non-empty, finite set of the component *attachement*
- The component *attachment* is a non-empty, finite set of the componenets *usageType*, *display*, *contentType*, *length*, *sha*2 with optionally *description* and/or $fileUrl$

1.10 Statement and Statements Schema

Statement

$$\begin{array}{l}
Id \\
Actor \\
Verb \\
Object \\
Result \\
Context \\
Timestamp \\
Stored \\
Attachements \\
statement : STATEMENT \\
\hline
statement = \{actor, verb, object, stored\} \cup \\
\qquad\qquad \mathbb{P}\{\mathrm{id}, result, context, timestamp, attachments\}
\end{array}$$

- The schema *Statement* introduces the component *statement* which consists of the components *actor*, *verb*, *object* and *stored* and the optional components *id*, *result*, *context*, *timestamp*, and/or *attachments*
- The schema *Statement* allows for subcomponent of *statement* to refrenced via the . (selection) operator

Statements

$$\begin{array}{l}
Statement \\
IsoToUnix \\
statements : \mathbb{F}_1 \\
\hline
statements = \{s : statement \mid \forall s_n : s_i..s_j \bullet i \leq n \leq j \\
\qquad\qquad \bullet\ convert(s_i.timestamp) \leq convert(s_j.timestamp)\}
\end{array}$$

- The schema *Statements* introduces the component *statements* which is a non-empty, finite set of the component *statement* which are in chronological order.

Appendex A: Visualization Exemplars

Appendex A includes a typology of data visualizations which may be supported within DAVE workbooks. These visualizations can either be one to one or one to many in regards to the algorithms defined within this document. Future iterations of this document will increasingly include these typologies within the domain-question template exemplars.

Line Charts

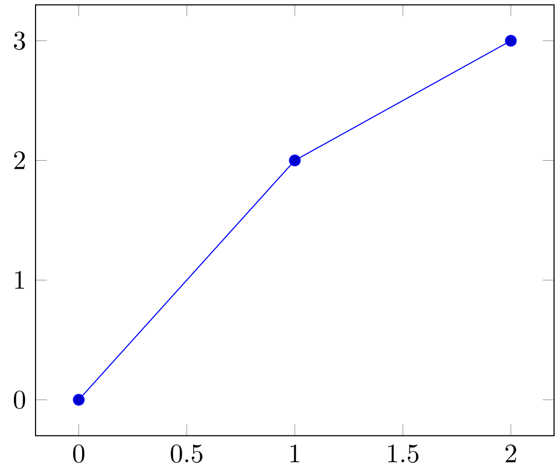

Figure 1: Line Chart

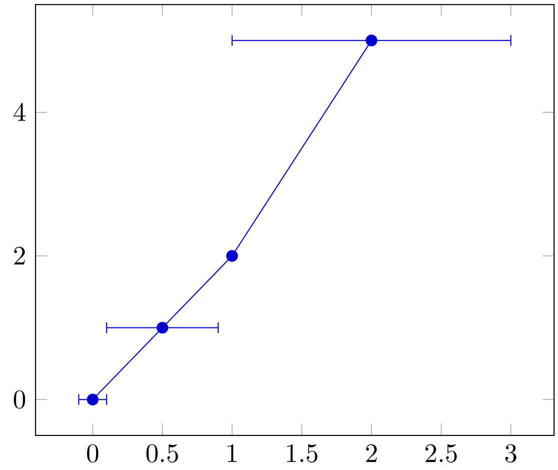

Figure 2: Line Chart with Error

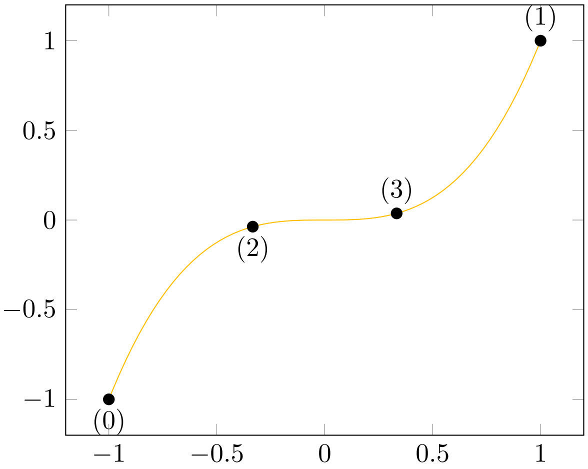

Figure 3: Spline Chart

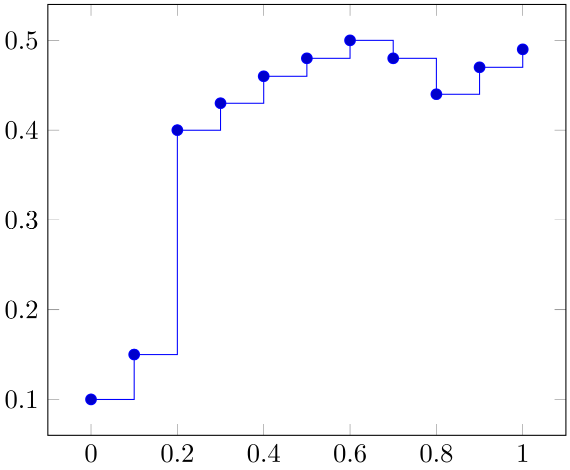

Figure 4: Quiver Chart

Grouping Charts

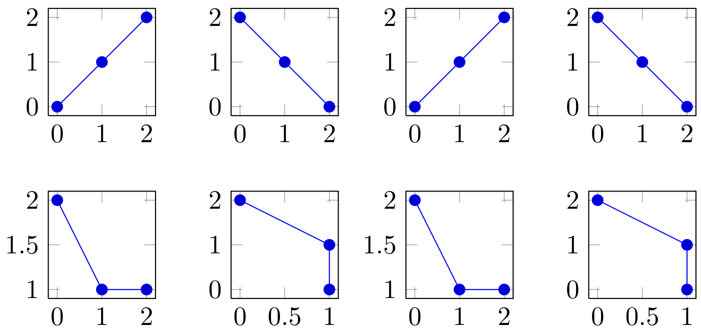

Figure 5: Grouped Line Charts

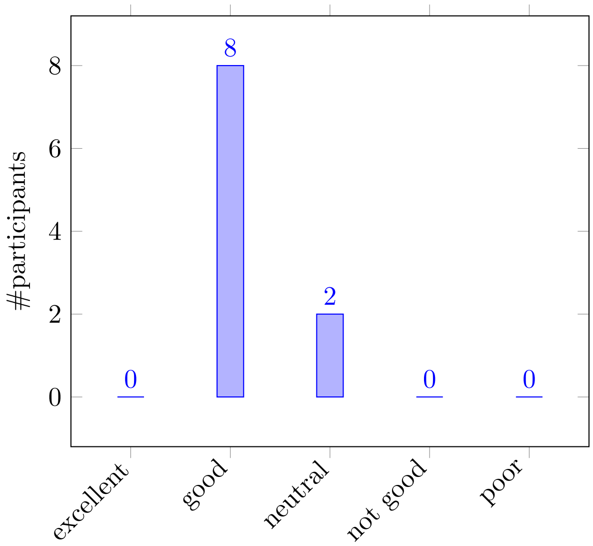

Figure 6: Histogram

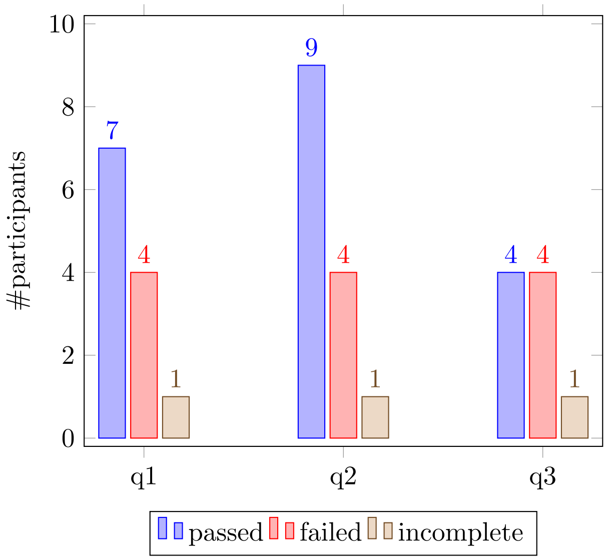

Figure 7: Bar Chart

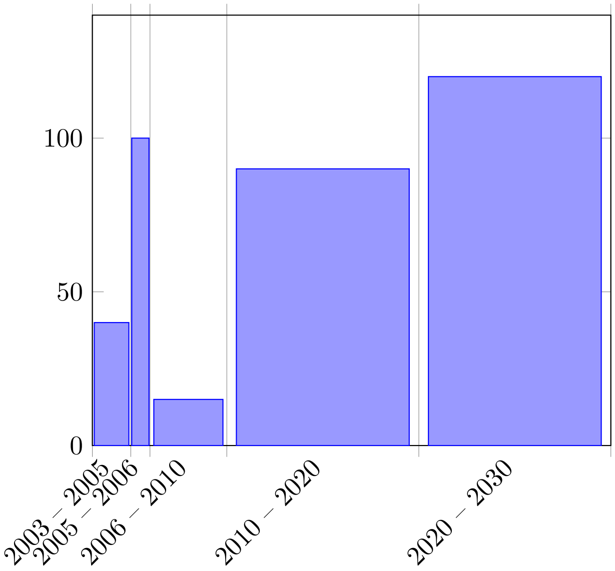

Figure 8: Bar Chart Grouped by Time Range

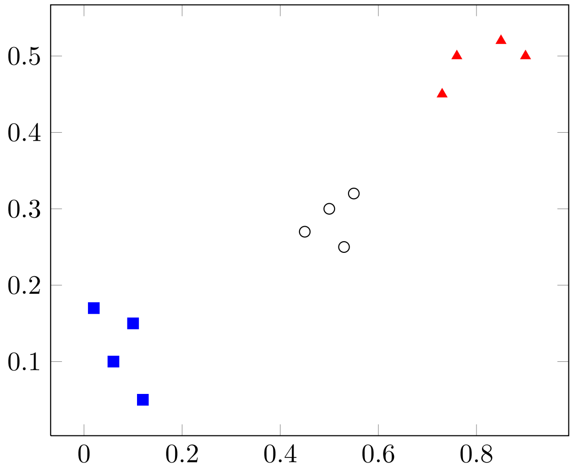

Figure 9: Scatter Plot

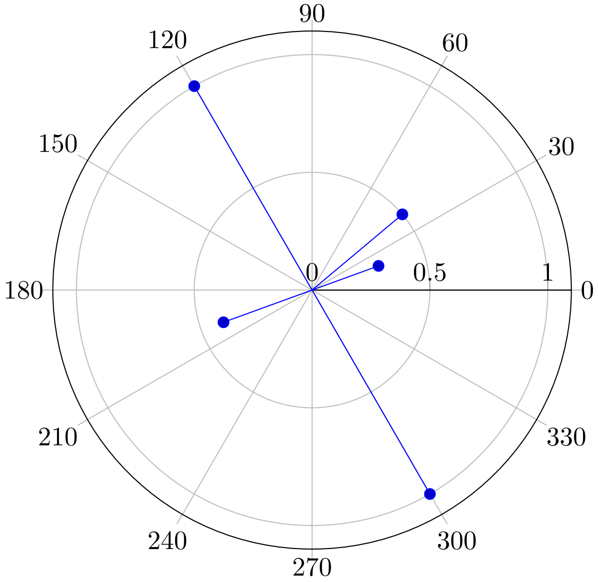

Figure 10: Polar Chart

Specialized Charts

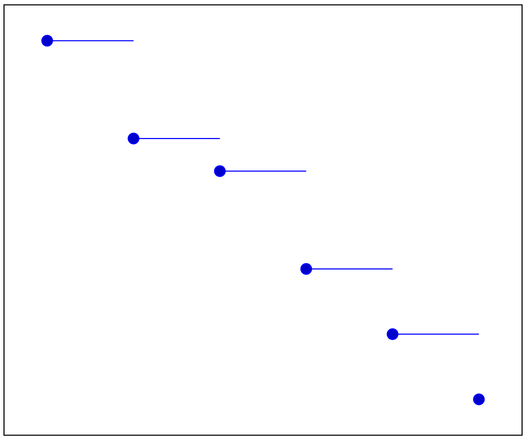

Figure 11: Gantt Chart

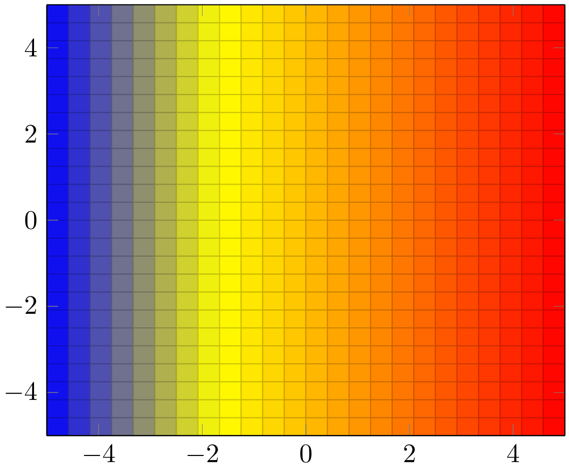

Figure 12: Heat Map

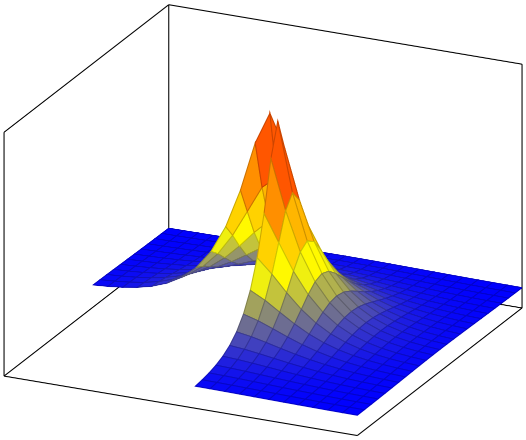

Figure 13: 3D Plot

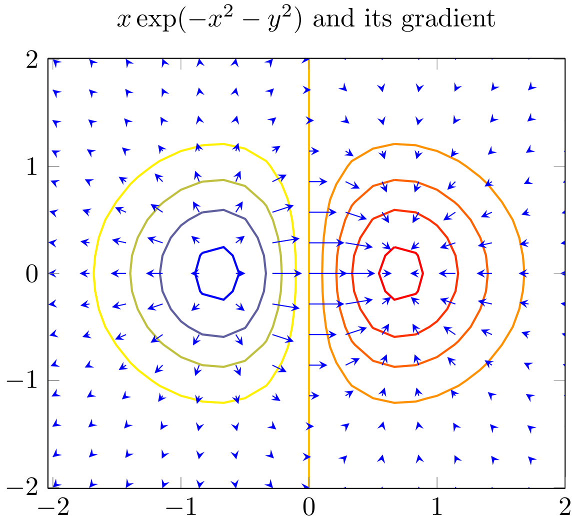

Figure 14: Gradient Plot