Part of German Network for Bioinformatics Infrastructure

GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE

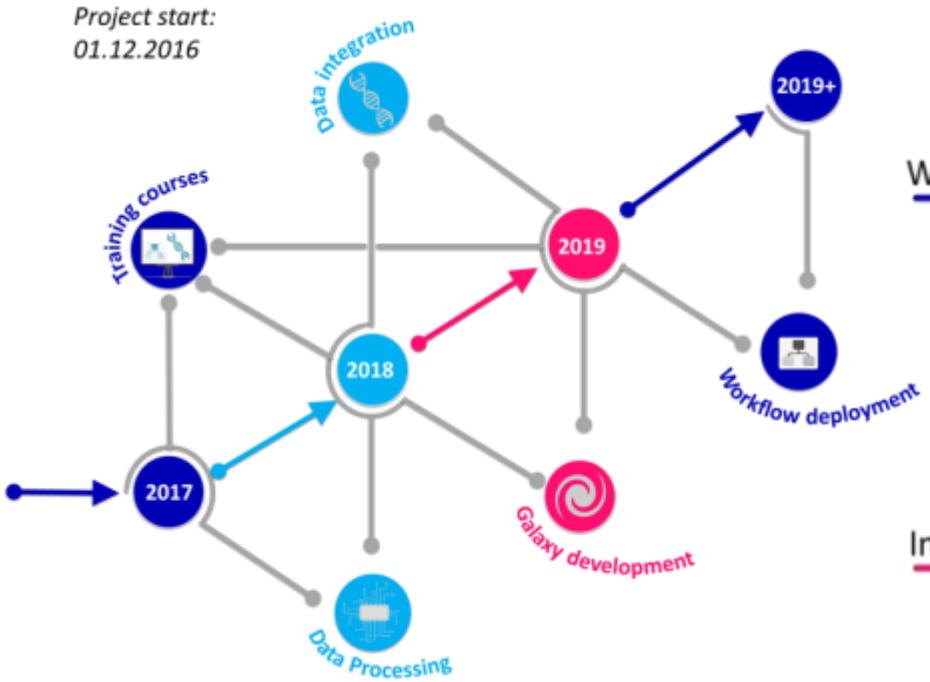
denbi.de

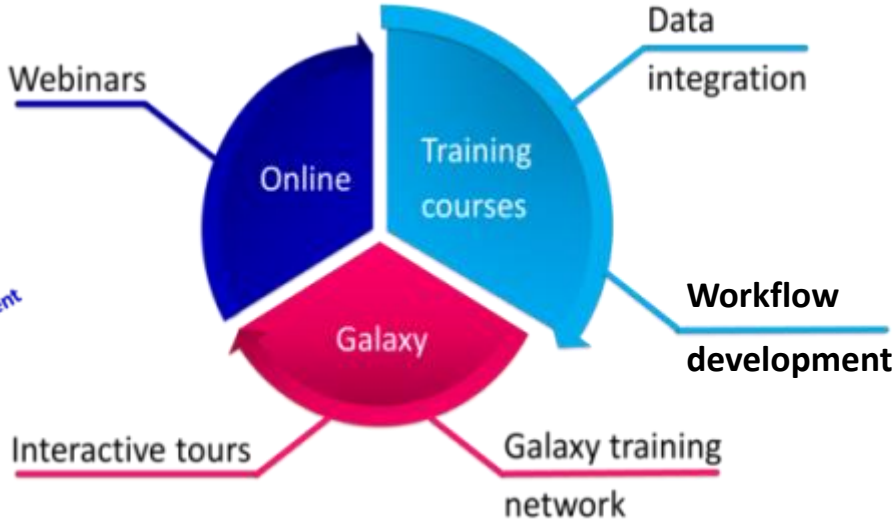
destair.bioinf.uni-leipzig.de

Structured Analysis and Integration of RNA-Seq experiments (de.STAIR)

Our aim is to enable a comprehensive **analysis of RNA-Seq experiments as a service.** To enable maximum usefulness, interconnectivity, and accessibility for the developed approaches and services, we will provide dedicated **workshops**, **training programs and screen casts** for bioinformaticians and other life scientists.

de.STAIR – RNA-Seq analysis and integration

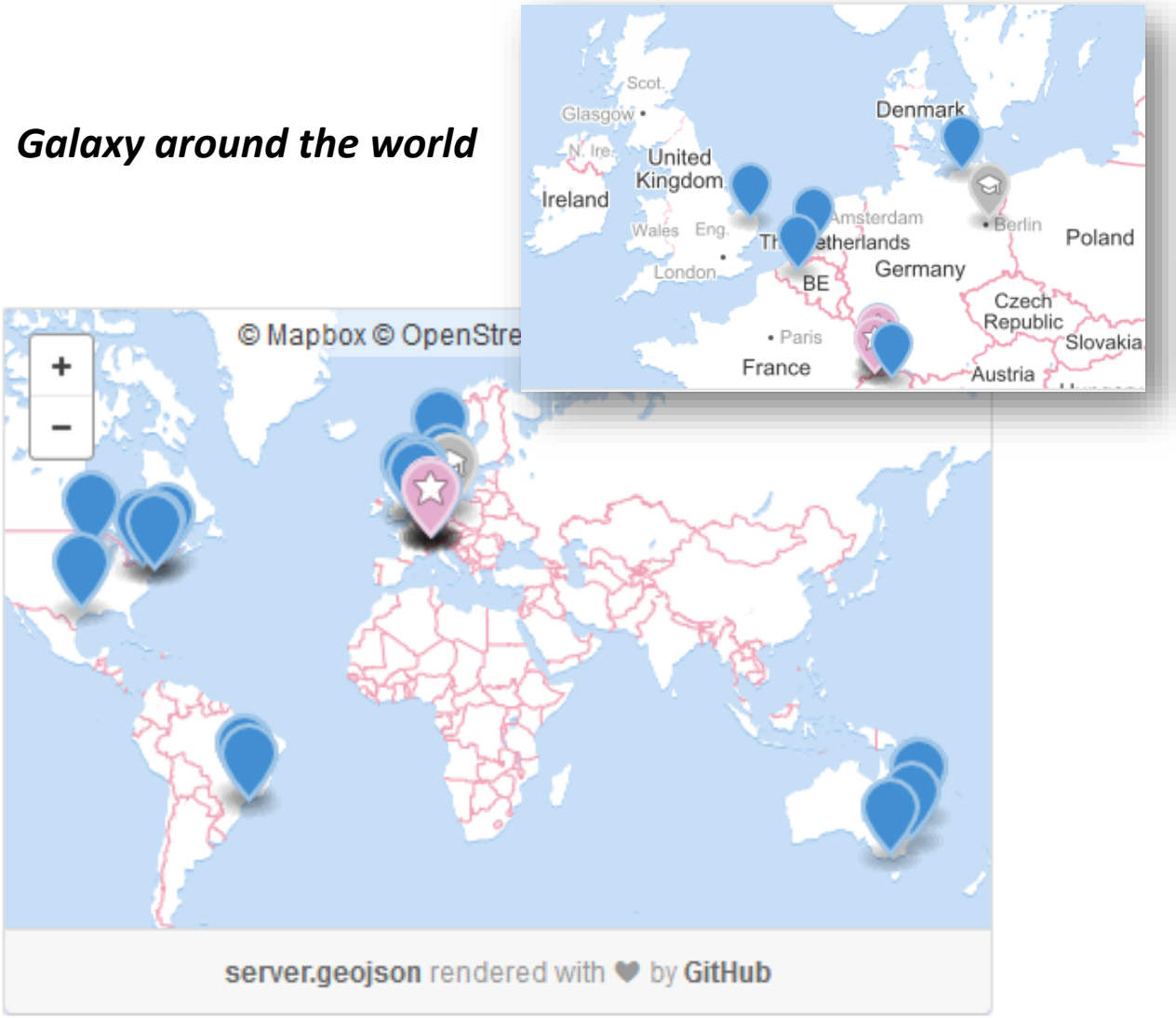




GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE

Supporting new data analysis approaches

Galaxy around the world



docker.com

usegalaxy.org

elixir-europe.org

denbi.de

Bioinformatics carpentry utilizing Galaxy

Day 2 - Data formatting

David Brauer

de.NBI
GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE

deSTAIR

SYSTEMS BIOLOGY
BIOINFORMATICS
ROSTOCK

Data formatting

- Why is data formatting necessary and important?
- Which data formatting tools does Galaxy offer?
- How can you use them for e.g.:
  - data preprocessing
  - data merging
  - data subsetting
- Regular expressions

Data formatting

- Why is data formatting necessary and important?
  - machine-readable format is mandatory
  - different tools = different formats/standards
  - subsetting for directed analyses

Types of data (1)

- computer-generated data, e.g.:
  - FASTA/FASTQ files
- manually recorded data, e.g.:
  - physiological parameters (weight, blood sugar, …)
  - cell counts
  - questionnaires

Types of data (2)

- computer-generated data
  - usually standardized
  - machine-readable by nature

Types of data (3)

- manually recorded data
  - usually your metadata
  - the “human” factor
  - data tidiness often only an afterthought

Problems with manually recorded data (1)

- way of recording the data
  - transcribing from a notebook
  - typing into Excel (or LibreOffice Calc etc) sheets
  - optimized for human readability/comfort

Tabelle 1

	sample_a	sample_b	sample_c	sample_d	sample_e	sample_f
feature_1	123	123	123	123	123	123
feature_2	456	456	456	456	456	456
feature_3	789	789	789	789	789	789
feature_4	1122	1122	1122	1122	1122	1122
feature_5	1455	1455	1455	1455	1455	1455
feature_6	1788	1788	1788	1788	1788	1788
feature_7	2121	2121	2121	2121	2121	2121
feature_8	2454	2454	2454	2454	2454	2454
feature_9	2787	2787	2787	2787	2787	2787
feature_10	3120	3120	3120	3120	3120	3120

Tabelle 2

	feature_15	feature_16	feature_17	feature_18	feature_19	feature_20
sample_w	A	4785	4785	AB	4785	4785
sample_y	B	5118	5118	HFG	5118	5118
sample_z	C	5451	5451	WU	5451	5451
sample_j	D	5784	5784	CJH	5784	5784
sample_q	E	6117	6117	WI	6117	6117
sample_i	F	6450	6450	CJH	6450	6450
sample_j	G	6783	6783	WI	6783	6783

Tabelle 3

	1h			5h			12h			24h		
	feature_1	feature_2	feature_3	feature_1	feature_2	feature_3	feature_1	feature_2	feature_3	feature_1	feature_2	feature_3
sample_a	123	456	789	124	457	790	125	458	791	126	459	792
sample_b	123	456	789	124	457	790	125	458	791	126	459	792
sample_c	123	456	789	124	457	790	125	458	791	126	459	792
sample_d	123	456	789	124	457	790	125	458	791	126	459	792
sample_e	123	456	789	124	457	790	125	458	791	126	459	792
sample_f	123	456	789	124	457	790	125	458	791	126	459	792
sample_g	123	456	789	124	457	790	125	458	791	126	459	792
sample_h	123	456	789	124	457	790	125	458	791	126	459	792

Problems with manually recorded data (2)

- common errors/inconsistencies
  - capitalization
  - varying use of . and , as decimal separators
- common problems
  - small helpers for human readability, e.g.:
    - empty columns to distinguish columns more easily
    - several tables on one sheet

Solving the problems

- takes time (a lot in many cases)
- can be difficult/tedious

- easiest solution:

getting it right at the source!

How to (generally) structure data (1)

- actually rather simple:
  - every single sample in its own row
  - every single parameter in its own column
- do not:
  - use subcategories
  - place separate tables on one sheet

How to (generally) structure data (2)

- helpful conventions (esp. for variable names)
  - do not use whitespace characters, rather use:
    - underscores (e.g.: max_velocity)
    - Camel case variations (e.g. maxVelocity or MaxVelocity)
  - do not include special characters, e.g. for units
    - bad: “Serum creatinine level [µmol/l]”
    - better: “serum_creatinine” and an additional plain text file with information about units
  - do not use umlauts etc. (ä, ö, ü, ß, …)

Other use cases

- merging data
  - by ID to add features
  - to add samples
- subsetting data
  - excluding samples below a certain threshold
  - excluding a certain experimental group

Tools in your toolbox

- many tools available
- context- and problem-dependent uses
- different ways to achieve the same results

What we’re going to cover

- transposing rows and columns
- find&replace operations using different tools
- merging two complementary data sets
- excluding samples based on their experimental group

What we're going to use

- existing "GUI" tools
- wrappers for the UNIX utilities *sed* and *awk*
- regular expressions

Use Case 1: Wrong orientation (1)

- first, we need to upload the data:
  - click “Choose local files”
  - select “transpose.csv” and “addFeatures.csv” (we’ll need it later)
  - click “Start”
- comma-separated, Galaxy mainly works on tab-separated files

Use Case 1: Wrong orientation (2)

- go to “General Text Tools” > “Text Manipulation” > “Convert delimiters to TAB”
- in the “Convert all” dropdown menu, select “Commas”
- select Multiple Datasets (select both files)
- uncheck “Condense consecutive delimiters in one TAB”
- click “Execute”

Use Case 1: Wrong orientation (3)

- (rename the files)
- go to “General Text Tools” > “Join, Subtract and Group” > “Transpose rows/columns in a tabular file”
- select the correct file
- click “Execute”

Use Case 2.1: Matching IDs (1)

- can be achieved using different tools in Galaxy
- for this we will need regular expressions (Regex)
- “language” to describe string patterns
- can be used to match/find said patterns in text

Digression: Regular Expressions (1)

- character sets:
  - [a-z], [A-Z]
  - [0-9]
  - [a-zA-Z0-9]
- special characters (extended regex syntax):

  . matches any character once (except newline)

  ^ signifies beginning of a line

  $ signifies end of a line

Digression: Regular Expressions (2)

- special characters (cont.):

*	previous pattern is matched zero or more times
+	prev. pattern is matched once or more times
?	prev. pattern is matched once at most
{…}	specifies an expected number of repetitions of the prev. pattern ({n}, {n,}, or {n,m})
()	group particular pattern for later reference
|	separate alternate possibilities

Use Case 2.1: Matching IDs (2)

- go to “General Text Tools” > “Text Manipulation” > “Column Regex Find And Replace”
- Select cells from “addFeatures_tabbed”
- using column “Column: 1”
- click “Insert Check”
- Find Regex: “([a-z])”
- Replacement: “sample_\1”

Use Case 2.2: Adding data (1)

- (rename the file)
- go to “General Text Tools” > “Text Manipulation” > “Join two files”
- 1st file: “transposed_tabbed”,
  Column to use: “Column: 1”
- 2nd file: “addFeatures_tabbed_regex”,
  Column to use: “Column: 1”

Use Case 2.2: Adding data (2)

- Output lines appearing in: “Both 1st & 2nd file.”
- Check: “First line is a header line”
- we can misuse the “Value to put in unpaired (empty) fields”:
  type in “id” to assign a new column name for column 1
- click “Execute”

Use Case 2.2: Adding data (3)

- (rename the file)
- go to “General Text Tools” > “Text Manipulation” > “Replace Text in entire line”
- select file: “merged_data”
- Find pattern: “Feat”
- Replace with: “feat”
- click “Execute”

Use Case 3: Selecting groups (1)

- upload: “addExpGrp.tsv”, then merge
- (rename the file)
- go to “General Text Tools” > “Filter and Sort” > “Filter data on any column using simple expressions”
- select file: “added_Groups”

Use Case 3: Selecting groups (2)

- With following condition: “c22!='expA’”
- Number of header lines to skip: “1”
- click “Execute”

Advanced alternatives: sed and awk

- originally UNIX utilities
- implemented into Galaxy with wrappers
- can be more complex than the “simpler”, mostly GUI-based functions we used so far
- have a lot of power and potential
- work row-wise

Text transformation with sed (1)

- stream editor
- usually used for substitutions with the command s
- general structure:

  sed OPTIONS… [SCRIPT] [INPUTFILE…]

- for us here, only the script part is interesting:

  s / [*regex to replace*] / [*replacement*] / [*flags*]

Text transformation with sed (2)

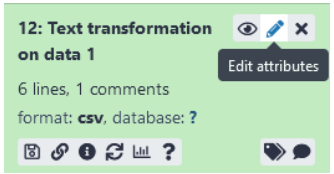
- go to “General Text Tools” > “Text Manipulation” > “Text transformation with sed”
- Select multiple datasets (select both start files)
- SED Program:

```
s/^([a-z]),/sample_\1,/      |prepends “sample_”
s/^(,)/id\1/                 |inserts “id” in first cell
s/Feat/feat/g                |lower case features
s/,/\t/g                     |replaces commas with tabs
```

- click “Execute”

Text transformation with sed (3)

- while theoretically now a tab-separated file, Galaxy doesn't convert it to tabular itself
- on the resulting files, go to "Edit attributes"
- select the tab "Convert"
- Name: "Convert CSV to tabular"
- click "Convert datatype"



Text reformatting with awk (1)

- named after the initials of the authors
- usually used to match a pattern (regex) and perform an action on a match
- general structure:

  pattern { action 1; action 2; action 3; }

- possible actions include printing only specified fields or arithmetic operations

Text reformatting with awk (2)

- go to “General Text Tools” > “Text Manipulation” > “Text reformatting with awk”
- select file “added_Groups”
- AWK Program:
  $2>124 && $22 ~ /exp[A-Z]/ {print $1, $2, $21-$20, $22}
- click “Execute”

Text reformatting with awk (3)

- AWK Program:

  $2>124 && $22 ~ /exp[A-Z]/ {print $1, $2, $21-$20, $22}

- pattern:

$2>124	field 2 needs to be bigger than 124
&&	and
$22 ~ /exp[A-Z]/	field needs to match the regex

Text reformatting with awk (4)

- AWK Program:
  $2>124 && $22 ~ /exp[A-Z]/ {print $1, $2, $21-$20, $22}
- action:
  on success, print the values of field 1, field 2, field 21 minus field 20, and field 22
- the commas cause the output to be separated (multiple table cells)

Quick recap

- lots of tools, plenty more available
- lots of use cases, plenty more imaginable
- no need to use anything, but:
  good to know that you can, should you need to!

Tools in your toolbox…

- learning by doing
- look through the tool list
- play around with the additional files
- play around with your own files

Thank you for listening!