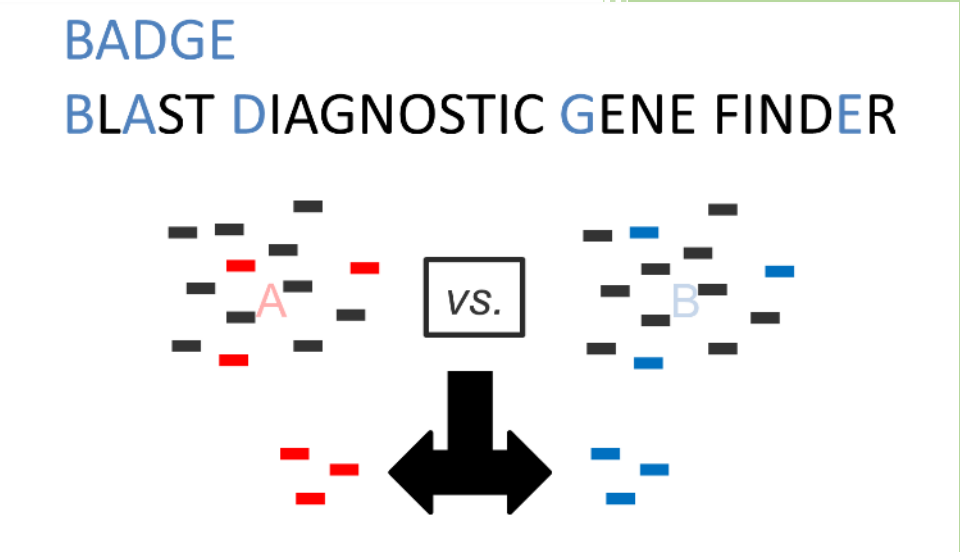
Juergen Behr & Andreas Geissler

2016

BADGE - MANUAL



Installation and execution instructions

Description

BlAst Diagnostic Gene findEr (BADGE) is an open source tool for the rapid identification of diagnostic marker genes (DMGs) for the differentiation of two or more bacterial groups of the same or different species. BADGE is programmed for Unix-based operating systems (OS) like Linux (Ubuntu, Mint, etc.) and MAC OS X. It can be easily installed also on Windows computers using a virtual Linux guest system. This manual describes the installation and execution of BADGE, RAST2BADGE and annotation_equalizer.

Licence - GPL

Third party binaries were obtained from NCBIs BLAST+ toolkit (Public Domain) and the EBI exonerate toolkit (GPL). Both are free software. For licenses see the corresponding toolkits and documentations. They are also included in the *BADGE/bin* directory. In case of the MAC version GNU grep is also included into the binaries and is GPL3 licensed.

BLAST+:
http://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastDocs&DOC_TYPE=Download

Exonerate: https://www.ebi.ac.uk/~guy/exonerate/

Quick Start – for Linux Systems

1. Install BADGE on a Linux OS and check installation

NOTE: NAMING CONVENTIONS → avoid the usage of **special characters** and **spaces** when naming files!!!!!

a. Download Linux version of BADGE from https://github.com/TMW-bioinformatics/BADGE and extract it to your home directory (*home/*your-home-directory*/BADGE*)
b. Start *check_BADGE.sh* within the *home/your-home-directory/BADGE/t-data* directory from command line using a terminal (command: *./check_BADGE.sh*)
c. If your installation was successful proceed to item 2 of the quick guide
d. If you need a more detailed instruction → see *Step-by-Step guide* starting with *Step 3* (for Linux)
e. NOTE: You can copy your BADGE folder (and rename to BADGE2, 3 etc.) as often as desired and run multiple instances in parallel.

2. Running BADGE

NOTE: It is all about naming of files and folder - First see the graphical illustration (figure 1 - 3) below! Also have a look at the data structure of the example from the publication within *BADGE/t-data/Pdam_set* → *genomes* and *orfs*.

a. Create 2 folders (genomes & orfs) within *home/your-home-directory/BADGE* → *home/your-home-directory/BADGE/genomes* and *home/your-home-directory/BADGE/orfs*
b. Create folders (cf. figure 1; at least 2) corresponding to your groups (e.g. A & B) (identically named folders) within *home/your-home-directory/BADGE/genomes* and *home/your-home-directory/BADGE/orfs* → e.g. *home/your-home-directory/BADGE/genomes/A*, *home/your-home-directory/BADGE/orfs/A home/your-home-directory/BADGE/genomes/B, home/your-home-directory/BADGE/orfs/B*
c. Provide *fasta*-files necessary → for each strain 1 file containing the genome (contigs) and 1 file containing the open reading frames (ORFs) → move them to the respective directory (genomes / orfs and group A / B) and rename them *identically*, always ending with the file-ending *.fasta* → e.g. *home/your-home-directory/BADGE/genomes/A/A1.fasta, home/your-home-directory/BADGE/orfs/A/A1.fasta*
d. Start *BADGE.sh* (figure 2) within the *home/your-home-directory/BADGE* directory from command line using a *Terminal* (command: *./BADGE.sh*)

e. NOTE: In case no DMGs are found with current settings for *minimum DMG occurrence*, BADGE can be interactive, waiting for your input –→ BADGE will ask you if you want to set a new value for the parameter or proceed with no changes

f. Key results will be found in a file like *BADGE_A_vs_B_final_out.tsv* (A vs B will be replaced by your group names) using a spreadsheet application, sequences can be found in the corresponding *fasta*-file (*BADGE_A_vs_B_DMGs.fasta*) and opened using a text editor (c.f. figure 3)

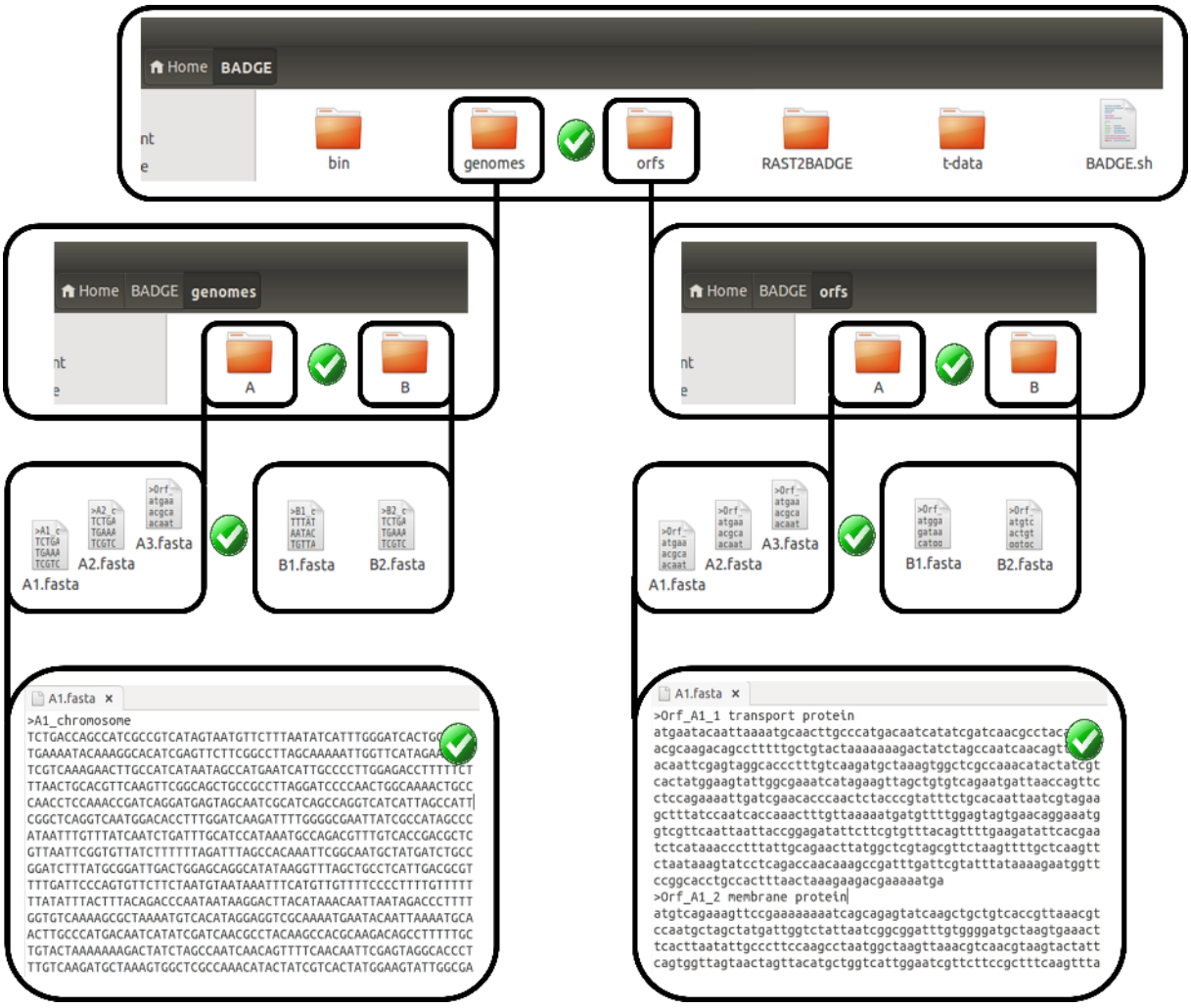


Figure 1: Data requirements for BADGE. Naming and structure are important for BADGE to work. Corresponding folders and files within *genomes* and *orfs* directory have to be named identically. Avoid any spaces or special characters.

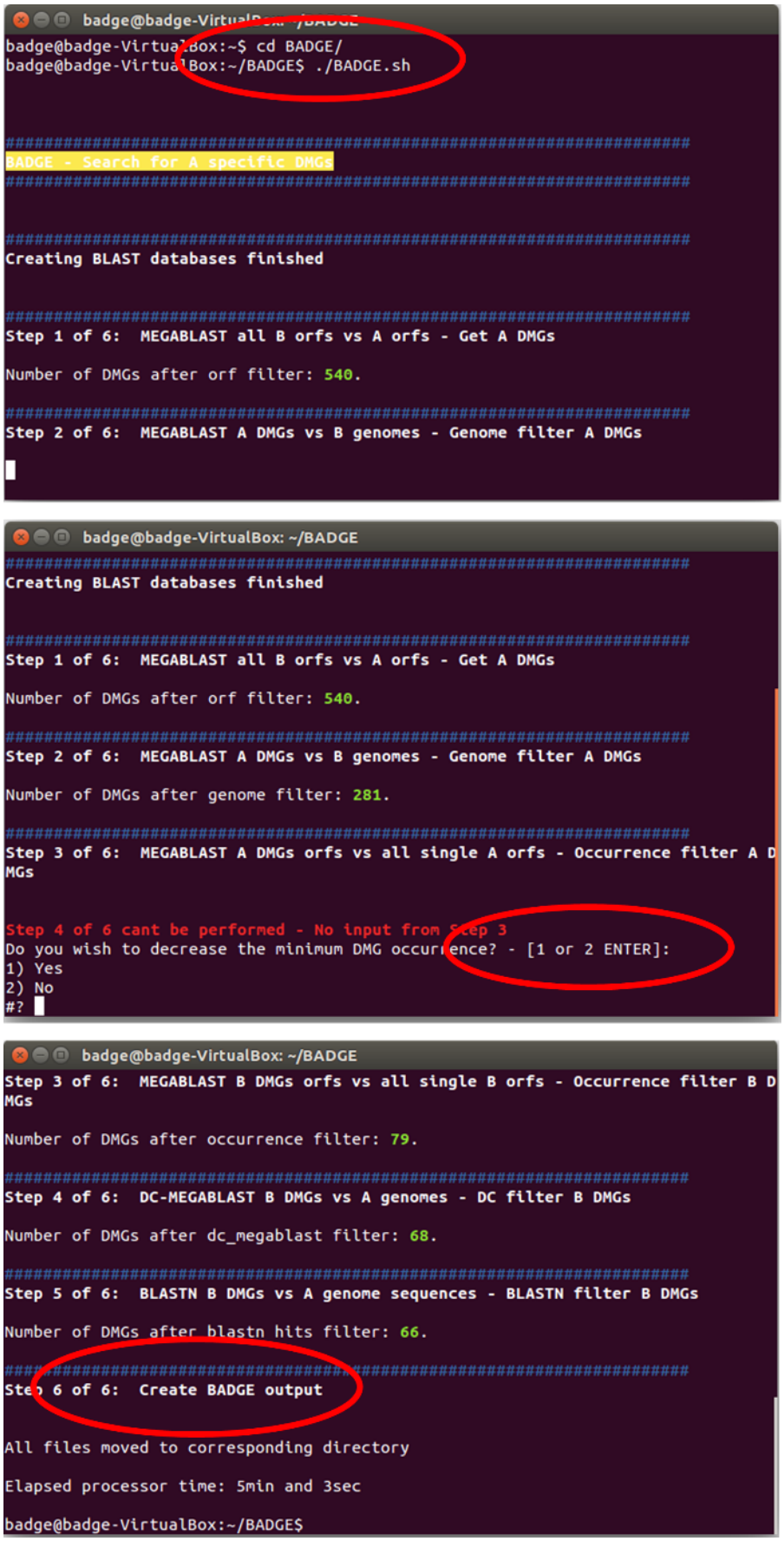


Figure 2: Running BADGE. Move to BADGE directory (*cd* command) and execute *BADGE.sh* (*./BADGE.sh*). Interact with script in case your chosen minimum DMG occurrence results in no DMGs

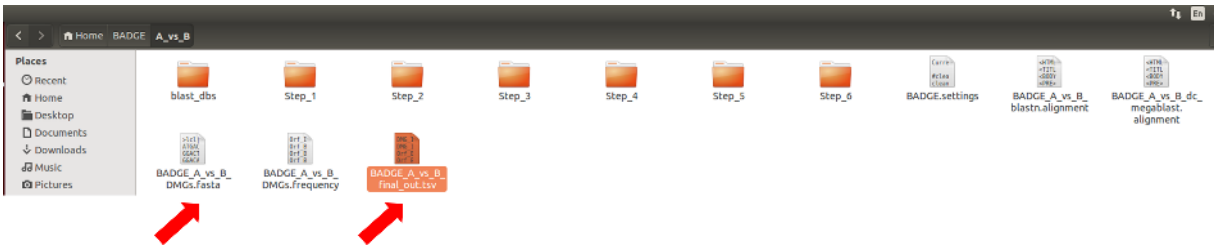


Figure 3: Output of BADGE. Key results can be found within a directory named as the comparison calculated (here A_vs_B). The central file, *BADGE_A_vs_B_final_out.tsv* contains the most important data about the identified DMGs and can be viewed and modified using a spreadsheet application.

3. Uninstall BADGE

BADGE can be uninstalled by simply deleting the BADGE folder. There are no temporary directories or files created elsewhere, nor any links to other directories.

Hardware requirements

BADGE was installed and successfully tested on an average personal computer (AMD Athlon™ X2 250 processor 3.00 GHz, 4.00 GB RAM) with Windows 8 (Microsoft) as host system (64 bit), using VirtualBox (www.virtualbox.org) and Ubuntu 14 (14.04.2 LTS, 64 bit) as guest system. The guest system was supplied with 1536 MB memory. It is not recommended to use less. For a better orientation you will find some performance data below:

The identification of 66 DMGs for the discrimination of 2 beer spoiling strains from 3 non-spoilers of *P. damnosus*, all having an approximate genome size of 2.3 Mbp took with:

- 1536 MB - Ubuntu 14 (64 bit) with VirtualBox on a Windows 8 host: 3 min 14 sec
- 4 GB – Centos 7 (64 bit): 2 min 23 sec
- 10 GB – Ubuntu 12 (32 bit) with VirtualBox on a Linux host: 2 min 26 sec
- 10 GB – Ubuntu 14, Debian 8, openSUSE, Fedora 21, Linux Mint 17 (all 64 bit) with VirtualBox on a Linux host: 1 min 30 sec
- 16 GB – Centos 7 (64 bit): 1 min 8 sec
- 16 GB – OS X Yoesmite (64 bit): 2 min 18 sec

Software requirements

BADGE was tested on 7 Unix-distributions, with 32 and 64 bit and in 2 OS languages (German / English):

- 64 bit: Ubuntu 14.04 LTS, CentOS Linux release 7.1.1503, Debian GNU/Linux 8, openSUSE 13.2, Fedora release 21, Linux Mint 17.1, OS X Yosemite 10.10.3
- 32 bit: Ubuntu 12.04.5 LTS

BADGE was also tested on Ubuntu 14 (14.04.2 LTS, 64 bit) as guest system on a Windows 8 (Microsoft, 64 bit) as host system using VirtualBox.

BADGE is using only binaries from blast+ (NCBI) and exonerate (EBI), which do not have to be compiled nor have any dependencies, which are not fulfilled by a normal Linux distribution. The rest of the script relies on BASH commands, preinstalled on every Linux distribution tested. In case of the MAC version GNU grep is also included into the binaries as a precompiled binary.

Installation on a MAC OS

1. Install BADGE and check installation

a. Download MAC OS X version of BADGE from https://github.com/TMW-bioinformatics/BADGE
b. Double-click the DMG (disk image) file and a window will appear with the BADGE folder and a link to your applications directory
c. Simply drag the BADGE folder to the appliactions directory link (as indicated by an arrow). This will copy the BADGE folder to your applications directory. If you want to run BADGE from any other location (e.g. home), simply drag it there instead.
d. After BADGE is installed, eject the BADGE installer by moving it to the Trash. You can now delete the BADGE disk image.
e. Open the *Terminal.app* (*Applications/Utilities/Terminal.app*) and change the working directory to *BADGE/t_data* (on the command prompt enter *cd* followed by a space character and drag the *t_data* folder on the *Terminal.app*, press enter)

f. Start *check_BADGE.sh* within the command line using a terminal (enter command: *./check_BADGE.sh,* press enter)
g. If your installation was successful you can use BADGE on your own data

Step-by-Step guide: Installation on Windows using Ubuntu as a guest system via VirtualBox

1. Installation of VirtualBox and creation of a virtual machine

a. Download latest version of VirtualBox for Windows hosts (https://www.virtualbox.org/wiki/Downloads) and install with default settings – simply proceed from window to window, also allow the installation of drivers by VirtualBox

b. Start VirtualBox and create a new virtual machine – click on *New* and proceed from window (1) to window (6) (c.f. figure 4)

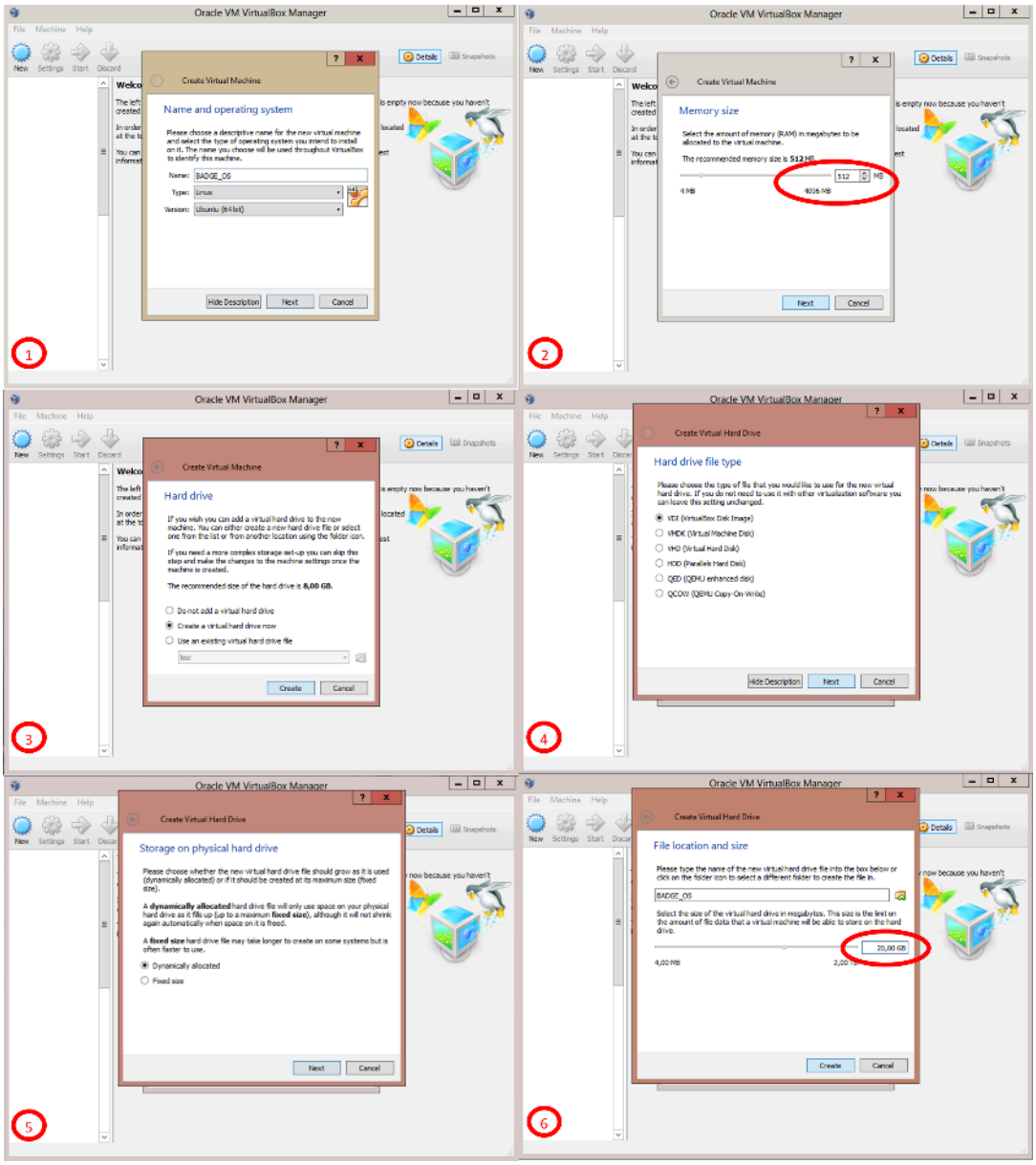


Figure 4: Create a new virtual machine. Chose *Linux* & *Ubuntu 64 bit* (1) and increase memory size to 50% of available (2) and file size to 20 GB (6).

(1) Name it *BADGE_OS*, select *Linux* as *Type* and *Ubuntu (64bit)* as *Version*

(2) If possible increase the amount of memory from 512 Mb (to maximum 50% of your host system – depends on your memory, the more the better) in order to improve the performance of your guest system. You don't have to change any of the following 3 settings – just click *Next* (3-5).

(6) Change the size of the virtual hard drive to at least 20 GB. You will end up with the new virtual machine *BADGE_OS* displayed in the *VirtualBox Manager*.

c. Change settings of your virtual machine by selecting *BADGE_OS* and click on the *Settings* icon (c.f. figure 5)

(1) Allow bidirectional file transfer – *Settings* → *General* → *Advanced* → set *Shared Clipboard* and *Drag'n Drop* to *Bidirectional*

(2) Create a shared folder (a folder the host and the guest have access to) – *Settings* → *Shared Folders* → "add folder" icon → chose a folder (you might create a separate – e.g. *shared_folder* for this purpose before) and select *Auto-mount* → stay in *Settings* for **Step 2**

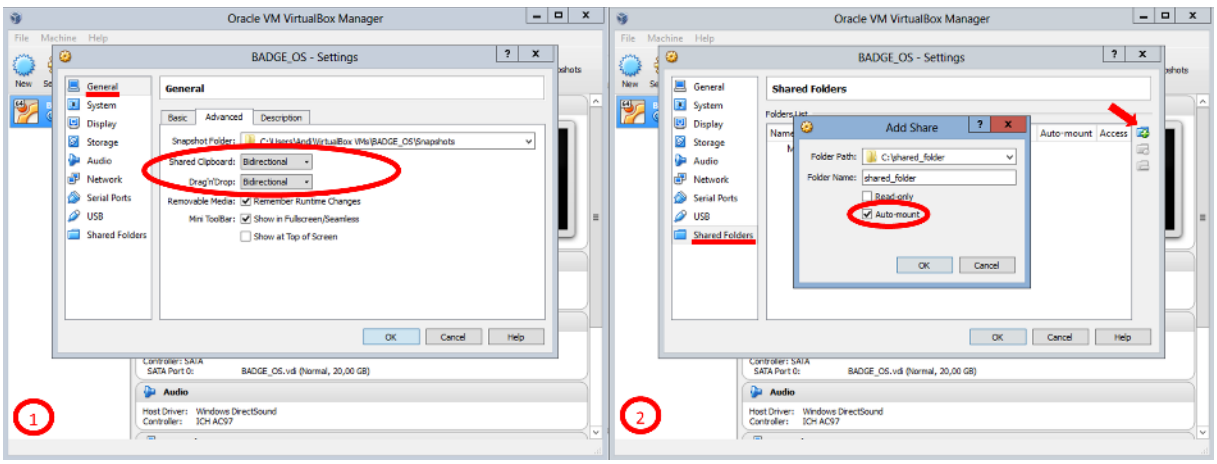


Figure 5: Change setting of virtual machine before installation of Ubuntu. Allow bidirectional file transfer (1) and create a shared folder (2).

2. Installation of Ubuntu as guest system

a. Download Ubuntu Desktop (image iso-file) from http://www.ubuntu.com/download and start installation (you don't need a CD)

(1) Insert image *iso*-file (figure 6): *Settings* → click on the *CD icon* in the *Storage Tree*, afterwards on the *CD icon* on the right side of the window and chose your virtual CD/DVD image file you have downloaded → Click *OK* and then click *Start* when you are back in the *VirtualBox Manager*

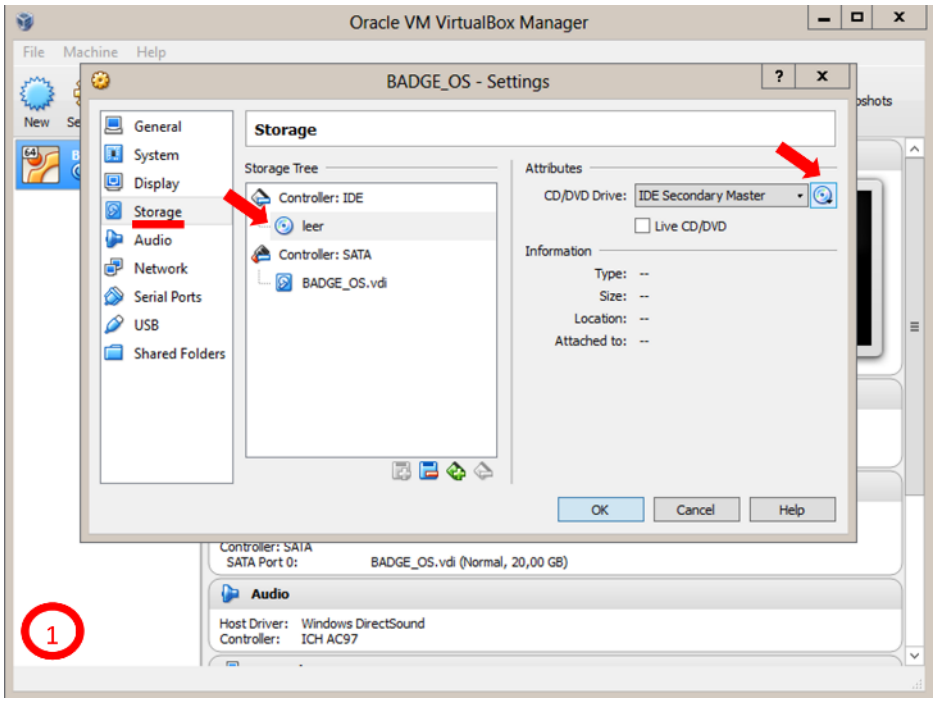


Figure 6: Insert Ubuntu image before installation.

b. Install Ubuntu – simply proceed from window to window

(1) Select your language and click *Install Ubuntu*

(2) Select *Download updates while installing* and click *Continue*

(3) Select *Erase disk and Install Ubuntu* and click *Install Now*

(4) Click Continue when you are asked to "*Write the changes to disk?*"

(5) Select your *Time Zone, Keyboard language* and enter your user data *(name)* – we recommend to use the name *BADGE* so you can better follow the further instructions

(6) Set your password as desired

(7) Wait until installation is done and restart Ubuntu when you are asked to

(8) When restarting you are asked to remove the CD – just press *enter* and wait until Ubuntu has started (you don't have to remove a CD as you did not insert one!)

c. Install Guest Additions – necessary for full screen, smooth performance and data sharing between host and guest system

(1) Click on *Devices* and select *Insert Guest Additions CD image* (figure 7) → Click *OK*, enter password when you are asked to and Click *Authenticate* – wait until guest additions are installed → Restart the virtual Ubuntu system and you will end up with a full screen version of Ubuntu

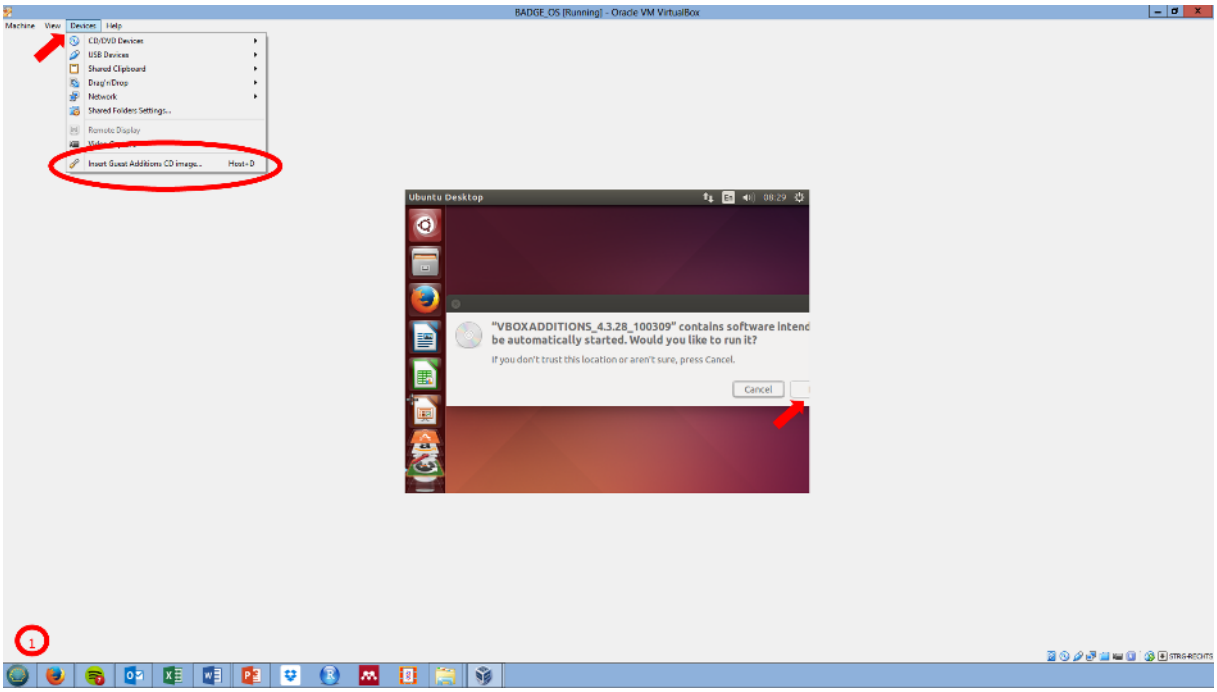


Figure 7: Install Guest additions. Select *Devices*, *Insert Guest Additions CD image* and proceed. Enter password when asked to.

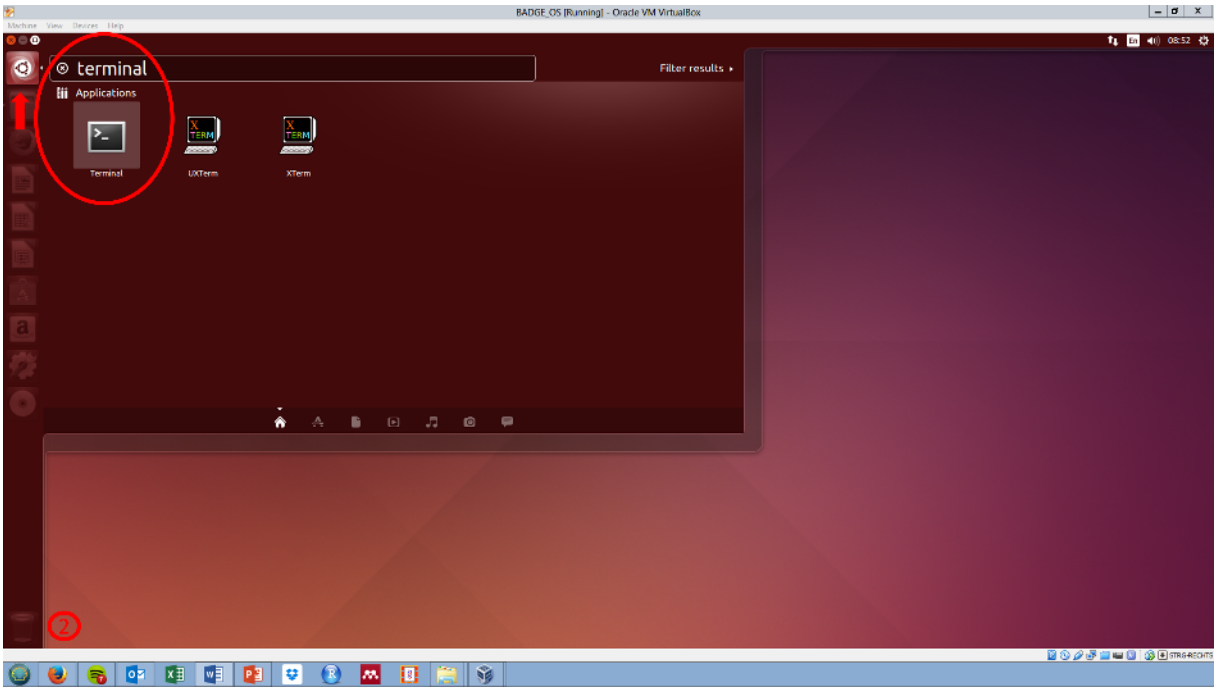


Figure 8: Start Terminal.

(2) Click on the *find-icon* and type *Terminal* – Start *Terminal* by clicking (c.f. figure 8)

(3) Enter your user to group with access to shared folders (figure 9) → type: *sudo adduser badge vboxsf* and press enter (replace *badge* (user) if you have chosen another username with your username) – enter your password and press enter – exit the *Terminal*

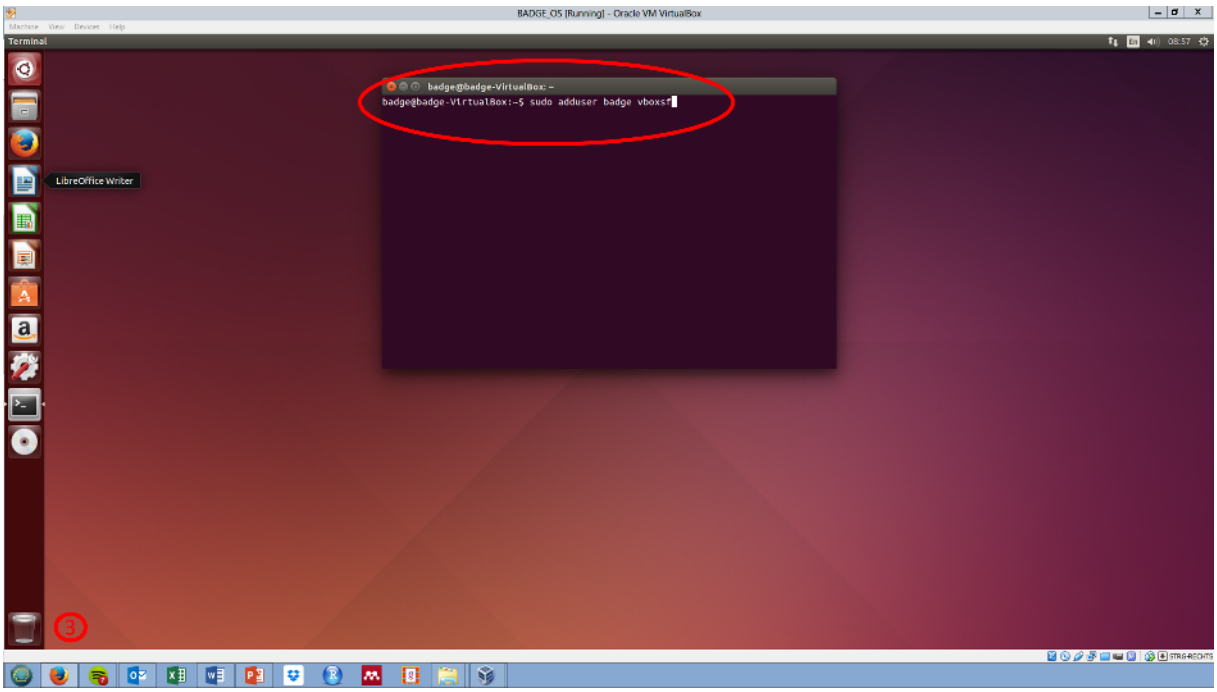


Figure 9: Add user to group with access to shared folder. Enter command and enter password when asked.

(4) Log out and log back in to Ubuntu

(5) You will find your automatically mounted shared folder by clicking on the files-icon → category *Devices* / *Computer*/*media*/*sf_shared_folder* (if you named your shared folder different the name of the *sf_* will be different) – you can test your shared folder by adding a file to the directory from the guest or the host system – it has to appear in both

3. Installation of BADGE on a Linux system – Ubuntu 64 bit

This guide will also work for other Linux distributions.

a. Download the Linux version (32 or 64 bit, depends on your system) of BADGE from https://github.com/TMW-bioinformatics/BADGE and extract it to your *Home* directory using either the archive manager or the command line (if desired) – the folder should look like (figure 10 (1))

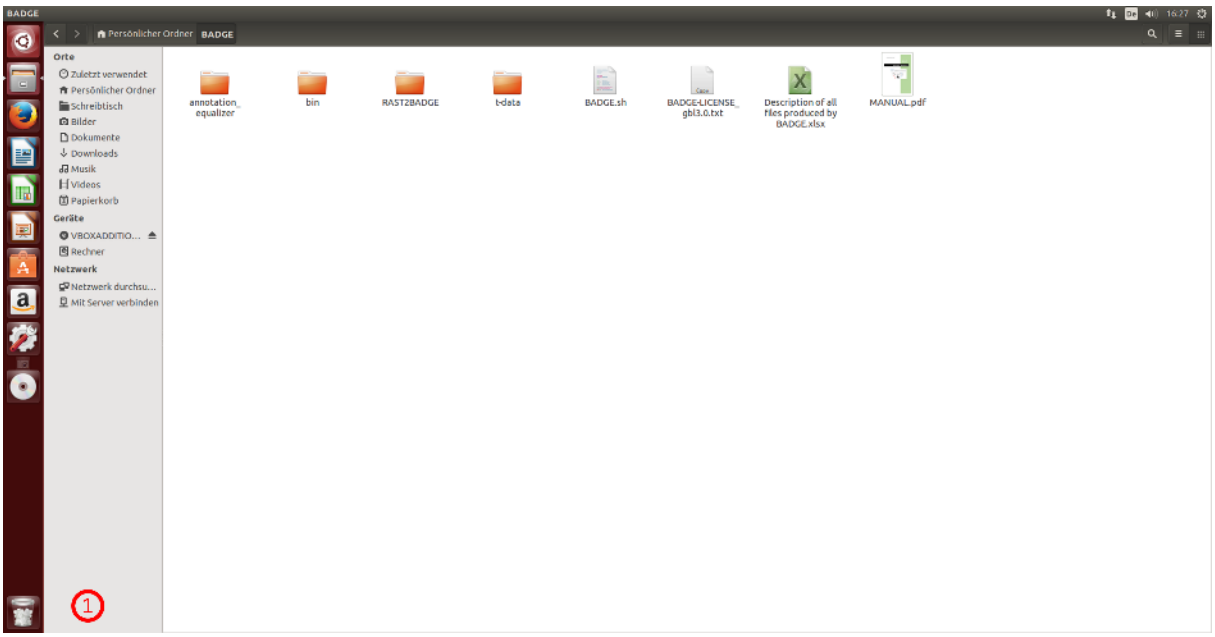


Figure 10: BADGE directory. After extraction of BADGE it should look like 1.

b. Check your BADGE installation

(1) Open a *Terminal*

(2) Navigate to *t-data* directory in *Terminal* by using the *change directory* command – type *cd BADGE/t-data/* and press enter (c.f. figure 11)

(3) Start check_BADGE by typing *./check_BADGE.sh* and press enter (c.f. figure 13) – NOTE: DO NOT CHANGE BADGE SETTINGS BEFORE YOU USE *check_BADGE.sh* - IT NEEDS DEFAULT SETTINGS TO CHECK YOUR INSTALLATION

(4) BADGE is now running on the supplied test data

(5) If the installation was correct the following message will be printed to the *Terminal* at the end of the run: '*The output produced by your BADGE installation is correct - BADGE installation was successful*'

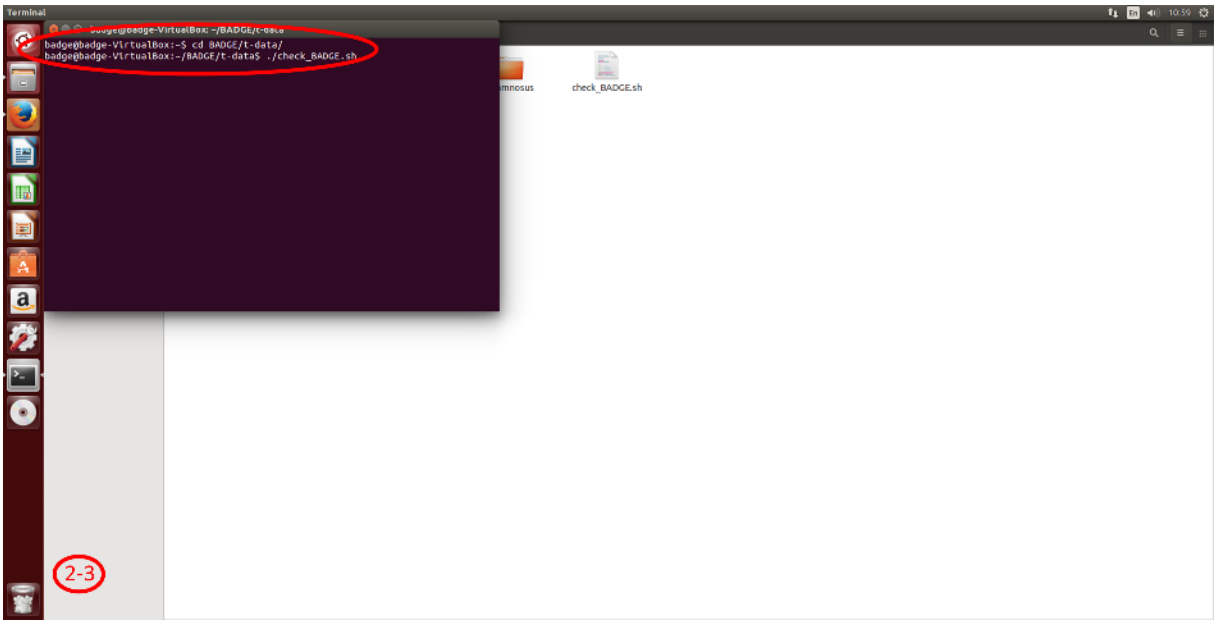


Figure 11: Check your BADGE installation. Open a *Terminal*, move to *BADGE/t-data* directory within *Terminal* using the change directory (*cd*) command and start *check_BADGE.sh* (*./check_BADGE.sh*).

Usage of BADGE

1. Data requirements and preparations

Briefly said, BADGE needs *fasta*-files, ending with '.fasta' and no alternative file ending, 1 containing the genome and 1 containing the open reading frames (ORFs) for each strain. These associated files than have to be named identically for each strain and placed into folders corresponding to their group membership, genome-*fasta*-files within the *BADGE/genomes* directory and ORFs-*fasta*-files within the *BADGE/orfs* directory. A graphical illustration can be found within the Quick Start and a more detailed description can be found below. If you did a RAST annotation with your data also see *4. RAST2BADGE*.

- File-ending: '.fasta' (DO NOT use '.fna' or '.fsa')
- NCBI blast is incompatible with some special characters (#,~,etc.). BADGE contains a function (switched on in default) to remove any (common) special characters, replacing them with '_'. Also avoid spaces.
- For each strain the user has to provide the following files:
    - genome-*fasta*-file: 1 *fasta* file is needed, containing the genome sequences (all contigs, chromosomes and plasmids) having unique identifiers – genomes (contigs) downloaded from NCBI or EBI already fulfil these requirements, assembly scripts/programs (e.g. SMRT analysis) also provide you with appropriate *fasta* files
    - orfs-*fasta*-file: 1 *fasta* file is needed, containing all open reading frames (ORFs) / coding regions (all genes predicted) having unique identifiers – again they can be downloaded from NCBI or EBI, or created by automatic annotation pipelines as RAST – NOTE: We included RAST2BADGE, a tool modifying RAST annotated *fasta* files to become more 'human-readable' and informative (the identifier is complemented with information about the function, contig and position on contig) – it is also based on BASH only
    - For each strain, both files have to be present and they have to be named exactly the same – e.g. file with open reading frames A1.fasta located in the orfs folder and file with genome sequence A1.fasta located in the genomes folder (you have to save them in different directories in order to name them identically)
    - Create 2 folders in the BADGE directory:
        - genomes (home/your-home-directory/BADGE/genomes)
        - orfs (home/your-home-directory/BADGE/orfs)

- Create folder corresponding to your groups – identically named folders have to be in genomes and orfs directories – avoid spaces!
    - Example: You aim to identify DMGs for the differentiation of A and B strains
        - *home/your-home-directory/BADGE/genomes/A, home/your-home-directory/BADGE/orfs/A*
        - *home/your-home-directory/BADGE/genomes/B, home/your-home-directory/BADGE/orfs/B*
- Place genome-*fasta*-files (corresponding to their group belonging) in the respective folder in the genomes directory
- Place ORFs-*fasta*-files (corresponding to their group belonging) in the respective folder in the orfs directory

2. Running BADGE

Start BADGE from command line:

- open a *Terminal*
- change directory to BADGE directory (e.g. *home/badge/BADGE)* → type: *cd home/your-home-directory/BADGE*
- Start BADGE by typing: *./BADGE.sh* – BADGE will start
    - NOTE: you can also add the BADGE directory persistently to your $*PATH* environment variable → then BADGE can be started from anywhere (*Terminal* without changing the directory) by typing *BADGE.sh* within the *Terminal*
- If BADGE will not find any DMGs with the selected minimum DMG occurrence, the user is interactively asked to proceed or to decrease the minimum DMG occurrence for the current run. This might help especially if you have wrongly group-assigned strains ('outlier').
    - The script asks you to if you want to decrease the minimum target occurrence – *1=yes / 2=no*
    - If you chose 'yes' → you will be asked to enter a new value from *0 – 1*
    - If you chose 'no' → BADGE will abort this calculation and proceed with the next combination (if available) or end.

BADGE will tell you after it is done!

NOTE: Besides the possibility to change all available settings, BADGE has switches for 2 additional modes, modifying the behaviour of the script explicitly

1. Protein level mode: BADGE will translate all orfs and perform Step1 and Step3 of the genome comparison with amino acid sequences – like normal BADGE but on protein level – interesting for comparison of more distant related bacteria
2. Mut(ation) level mode (2 sub-modes, nucleotide or amino acid level): BADGE will identify all DMGs, which are exactly identical within one group and have ANY difference to members in the other group – interesting for the identification of mutations

3. Evaluation of BADGE results

BADGE will create a folder for every group comparison performed, which will be named e.g. A_vs_B and B_vs_A and so on.

The most important output files are described below (examples refer to the A_vs_B scenario):

- *BADGE_A_vs_B_final_out.tsv* → contains most important information about each DMG – open with spreadsheet application (NOTE: tab as only delimiter!)
    - percent_occurrence: proportion (0-100 %) of group members a DMG is present in
    - dc_blast_hit: yes / no, yes if a DMG has a remaining dc-megablast hit (of course with quality values below those selected in BADGE) within the 'opposed' group (group B)
    - max_blastn: the maximum length of a blastn hit produced by a DMG in the 'opposed' group (group B)
    - ORF_ID: Identifiers of all ORFs which are part of the DMG (ORF IDs for each strains, if frequency of DMG within a genome is higher than 1, more than 1 ID per strain is possible)
    - ORF_lenght: length of ORFs
    - annotation: should contain information about the predicted function etc. (depends on your annotation)
    - contig: names the contig where the ORF is located (plasmid, chromosome)
    - start / stop: coordinates on contig in base pairs
    - overlapping DMGs: DMG variants which overlap (as consequence of annotation_equalizer)
- *BADGE_A_vs_B_DMGs.fasta* → *fasta*-file containing a representative sequence of each DMG identified – open with text editor
- *BADGE.settings* → contains the settings used for the particular run – open with text editor

- *BADGE_A_vs_B_blastn.alignment* & *BADGE_A_vs_B_dc_megablast.alignment* → *html* files containing the alignments / remaining blastn & dc-megablast hits of the calculated DMGs within the opposed group (in group B) – open with web browser
- *BADGE_A_vs_B_DMGs.frequency* → contains information about frequency of DMGs within group members – open with spreadsheet application
- *BADGE_A_vs_B_DMGs.distribution* → contains information about distribution of DMGs within group members – open with spreadsheet application
- Within the subdirectory *BADGE_DMGs_fasta_for_alignment* (e.g. *A_vs_B/ BADGE_DMGS_fasta_for_alignment*) you will find additional fasta files for every DMG
    - *Step_6_4_DMG_xx_seq_align.fasta* → contains all ORF sequences with membership to a given DMG – open with text editor or use for sequence alignment and consensus creation (e.g. clustalw)

In case you chose Protein level mode additional *fasta* files containing the protein sequences will appear.

4. RAST2BADGE

RAST2BADGE.sh can be used to modify RAST annotated genomes in order to become more 'human-readable' and informative (the identifier is complemented with information about the function, contig and position on contig).

Running RAST2BADGE:

1. Download the *fna* and the *gff*-files of your genome(s) from http://rast.nmpdr.org/
2. Place them all into the directory *RAST2BADGE* within your *BADGE* folder
3. Rename *fna* and the *gff*-files as you like (e.g. E_coli_K12.gff / E_coli_K12.fna) – the name of the files will be included into the annotation
4. Start *RAST2BADGE.sh* from command line – start Terminal, move to RAST2BADGE directory using the *cd* command and start the tool by typing *./RAST2BADGE.sh*
5. A file with the same basename (e.g. E_coli_K12.fasta) as your *gff* and the *fna* file will be created, ending with *".fasta"* – it can be used for BADGE now

5. annotation_equalizer

annotation_equalizer.sh can be used to find and equalize inconsistent annotations within a set of genomes. Non- or differently annotated orfs, as a consequence of frame shifts or different gene calling approaches (e.g. RAST, glimmer, NCBI prokaryotic annotation pipeline)

will be identified and added as additional orf variants to the orf *fasta* files of each particular genome. E.g. sequences which are identical within all genomes of a group, but not or differently annotated in single members of the group, will be extracted from the genome sequence of the corresponding member genome sequence and added as a possible gene variant to the orfs file. The *XTRA* tag in the identifier can be used to track these genes. The equalization enables BADGE to predict DMGs, even if they are only annotated within one member of a group and reduces the negative effect of inconsistent annotations.

Running annotation_equalizer:

1. Create directories *BADGE/genomes* & *BADGE/orfs* and place / rename your files exactly as for BADGE
2. Start *annotation_equalizer.sh* from command line – start Terminal, move to *annotation_equalizer* directory using the *cd* command and start the tool by typing *./annotation_equalizer.sh*
3. You will find your equalized *orf* files in *BADGE/annotation_equalizer/Xtra*, the original files in *BADGE/annotation_equalizer/raw*
4. Move the *genomes* and *orfs* folders from *BADGE/annotation_equalizer/orfs/Xtra* in the BADGE directory
5. Use them for a BADGE run (see point 2)

6. BADGE settings

Changing BADGE settings can be easily done using any text editor. Open *BADGE.sh* with your text editor and change settings as desired. Save your changes before you start a new BADGE run. We recommend to save a copy of *BADGE.sh* somewhere else, in case something goes wrong. If your version does not work anymore, just replace it by the original *BADGE.sh* script

Table 1: BADGE settings with type, options, default value und description.

Setting name in BADGE	type	options	default value	description and notes
clean_up	boolean	true/false	true	if switched on (true), only the most important output will be kept – all other files are deleted
min_DMG_occurrence	real	0 - 1	1	minimum occurrence of a DMG in group A to be reported, a value of 0.5 would mean, that the DMGs to identify only have to be in 50% of all members of group A in order to be reported
special_character	boolean	true/false	true	replaces incompatible special characters with '_'
megablast_perc_identity_cut	real	0 - 100	95	% identity threshold value – minimum identity of hit to be kept
megablast_e_value	real	> 0	1×10^{-15}	e-value threshold – maximum value of blast hit to be kept
megablast_within_group_qscov	real	0 - 1	0.95	Query / subject coverage (length of query divided by length of subject and vice versa), threshold within a group – minimum value of blast hit to be kept (relevant for core determination – Step 3)
megablast_between_group_qscov	real	0 - 1	0.50	Query / subject coverage threshold between groups – minimum

dc_mode	boolean	true/false	false	if switched on (true), basic DMG identification is done using dc-megablast instead of megablast, more sensitive but slower
dc_filter	boolean	true/false	true	if switched on (true), DC-megablast filter will be applied to data – potential DMGs with a long (> 50 % qscov) but 'low' identity (default between 70 – 90 % identity) hit in other group will be discarded
dc_perc_identity_cut	real	0 - 100	70	c.f. megablast_perc_identity_cut
dc_blast_e_value	real	> 0	10	c.f. megablast_e_value
dc_between_group_qscov	real	0 - 1	0.50	c.f. megablast_between_group_qscov
blastn_filter	boolean	true/false	true	if switched on (true), blastn filter will be applied to data – potential DMGs with a short (> 25 % qscov) but 'high' identity (> 95 % identity) hit in other group will be discarded
blastn_perc_identity_cut	real	0 - 100	95	c.f. megablast_perc_identity_cut
blastn_e_value	real	> 0	10	c.f. megablast_e_value
blastn_between_group_qscov	real	0 - 1	0.25	c.f. megablast_between_group_qscov
protein_level	boolean	true/false	false	if switched on (true), ORFs will be translated and used to perform BADGE (Steps 1 and 3) using blastp – NOTE: if Protein-Level is active dc-megablast filter and blastn filter are disabled automatically
blastp_perc_identity_cut	real	0 - 100	50	c.f. megablast_perc_identity_cut
blastp_e_value	real	> 0	10	c.f. megablast_e_value
blastp_within_group_qscov	real	0 - 1	0.50	c.f. megablast_within_group_qscov
blastp_between_group_qscov	real	0 - 1	0.50	c.f. megablast_between_group_qscov

fastatranslate_geneticcode	real	1 - 25	11	genetic code to be used for translation – 11 corresponds to bacterial, archaeal and plant plastid code
fastatranslate_frame	real	1 - 3	1	reading frame to be translated
protein_level_clean_up	boolean	true/false	true	if switched on (true), translated orf files will be removed after BADGE is done
identify_overlapping	boolean	true/false	false	if switched on (true) BADGE will label overlapping DMGs
mut_level_nt	boolean	true/false	false	if switched on (true) BADGE will look for DMGs with ANY (even single nucleotide changes) differences to the opposite group
mut_level_aa	boolean	true/false	false	if switched on (true) BADGE will look for DMGs with ANY (even single amino acid changes) differences to the opposite group in protein_level mode