TABLE OF CONTENTS

PhysIO Wiki Main Page

This Wiki contains the most up-to date User Guide to the PhysIO Toolbox. It is most conveniently viewed online with gitlab-formatting. Its content can also be retrieved offline after downloading the toolbox, in folder `physio/wikidocs` as

- plain text `.md` markdown files
- as single HTML and PDF file: `documentation.{html,pdf}`

List of current Wiki files

- HOME: This page. Landing Page of PhysIO Wiki. Navigation to all other files and this explanation.
- FAQ: Frequently asked questions (for users) (also frequently updated!)
- QUICKSTART: Example script and how to use it on test data, Intro to Batch Editor GUI
- EXAMPLES: List and explanation of all examples available for download
- Manual: Detailed overview of the toolbox functionality, sub-pages following its modular structure
  - Read-In of Logfiles
  - *The following pages are under construction*
  - **TODO** Preprocessing Physiological Data
  - **TODO** Physiological Noise Modeling
  - **TODO** Performance Assessment
  - **TODO** Technical Documentation For developers, list of all functions, see header of `.m` files for now

Other sources of Documentation

Documentation for this toolbox is also provided in the following forms:

1. Overview and guide to further documentation: README.md and CHANGELOG.md
   - README.md: purpose, installation, getting started, pointer to more help
   - CHANGELOG.md: List of all toolbox versions and the respective release notes, i.e. major changes in functionality, bugfixes etc.

2. Within SPM: All toolbox parameters and their settings are explained in the Help Window of the SPM Batch Editor
3. Within Matlab: Extensive header at the start of each `tapas_physio_*` function and commenting
    - accessible via `help` and `doc` commands from Matlab command line
    - starting point for all parameters (comments within file): `edit tapas_physio_new`
    - also useful for developers (technical documentation)
4. Scientific Documentation: Our paper on the PhysIO Toolbox explains both the scientific background on physiological noise modeling, as well as the modular structure of the toolbox as comprehensivce yet succinct as we (and the reviewers) could.

Quickstart

Quickstart Manual

Purpose

This page provides simple walk-throughs of the SPM Batch Editor GUI, the scripts to run the main examples, and the most common output plots of the PhysIO Toolbox.

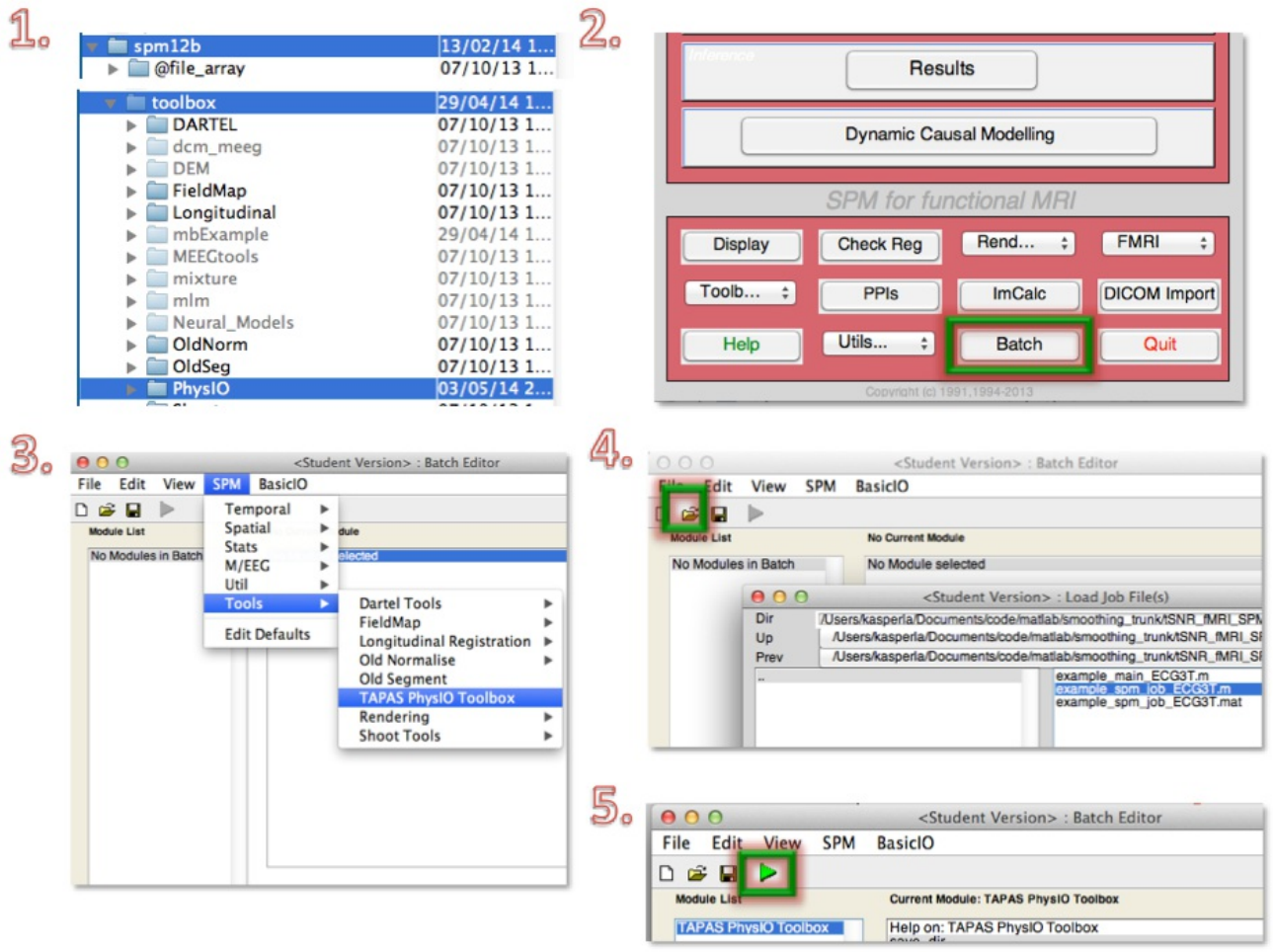
Requirements

- Download the latest PhysIO Toolbox version from the TAPAS software release page on GitHub.
- Download the example data by running `tapas_download_example_data()` in the `misc` subfolder of the TAPAS software download.

One-Page-Quickstart (with SPM)

...of SPM Batch Editor GUI for PhysIO Toolbox

1. Go to the `tapas/PhysIO` folder and run `tapas_physio_init()`
    - This should check whether PhysIO and SPM are properly installed, whether paths are set correctly, and whether the PhysIO code folder is properly linked (or copied, for Windows) to the sub-folder `spm/toolbox`, where SPM expects its Batch Editor toolboxes to reside.
2. (Re-)Start SPM (`spm fmri`) and open the Batch editor (Button Batch in SPM GUI).
3. The PhysIO Toolbox should now show up under SPM -> Tools -> TAPAS PhysIO Toolbox
4. Change directory (!) to `examples/Philips/ECG3T`-folder and open an example spm-job file with the Batch Editor, e.g : `example_philips_ecg3t_spm_job.m`
    - If you click on any of the various parameters, you will see a detailed description in the Help Box in the lower part of the Batch Editor window.
5. Press Play (in the Batch Editor)!
6. Feel free to try out the other batches in other vendor/device subfolders, which always end in `*_spm_job.m` or `*spm_job.mat`



One-Page-Quickstart (Matlab only, no SPM)

1. Go to the `tapas/PhysIO` folder and run `tapas_physio_init()`
2. Change Matlab directory to `examples/Philips/ECG3T`-folder
3. Open an example matlab script, e.g., `example_philips_ecg3t_matlab_script.m`
4. Press Play (in Matlab, or F5)!
5. Feel free to try out the other matlab scrupts in other vendor/device subfolders, which always end in `*_matlab_script.m`
    - A detailed documentation of all parameters is found in `tapas_physio_new.m`, next to each parameter, just open this file with Matlab or any editor.

Troubleshoot

1. If the PhysIO Toolbox does not show up in the SPM Batch Editor, the necessary matlab code-files cannot be found by SPM.
    - Manually copy the PhysIO Toolbox `PhysIO/code` folder to `spm/toolbox/PhysIO` (see Figure 1).
    - Note that this is only the `code` subfolder of PhysIO, i.e., the `tapas_physio*.m` files should be located directly in `spm/toolbox/PhysIO` (**not** `spm/toolbox/PhysIO/code`).

Readme

TAPAS PhysIO Toolbox

Current version: Release 2024a, v9.0.0

> lars.kasper@alumni.ethz.ch

Translational Neuromodeling Unit (TNU)
Institute for Biomedical Engineering
University of Zurich and ETH Zurich

Download

- Please download the latest stable versions of the PhysIO Toolbox on GitHub as part of the TAPAS software releases of the TNU.
- Older versions are available on the TNU website.
- The latest bugfixes can be found in the development branch of TAPAS and are announced in the GitHub Issue Forum.
- Changes between all versions are documented in the CHANGELOG.

Purpose

The general purpose of this Matlab toolbox is the model-based physiological noise correction of fMRI data using peripheral measures of respiration and cardiac pulsation. It incorporates noise models of cardiac/respiratory phase (RETROICOR, Glover et al. 2000), as well as heart rate variability and respiratory volume per time (cardiac response function, Chang et. al, 2009, respiratory response function, Birn et al. 2006), and extended motion models. While the toolbox is particularly well integrated with SPM via the Batch Editor GUI, its simple output nuisance regressor text files can be incorporated into any major neuroimaging analysis package.

Core design goals for the toolbox were: *flexibility*, *robustness*, and *quality assurance* to enable physiological noise correction for large-scale and multi-center studies.

Some highlights:

1. Robust automatic preprocessing of peripheral recordings via iterative peak detection and Hilbert transforms, validated in noisy data and patients.
2. Flexible support of peripheral data formats (Siemens, Philips, HCP, GE, Biopac, ...) and noise models (RETROICOR, RVHRCOR).
3. Fully automated noise correction and performance assessment for group studies.
4. Integration in fMRI pre-processing pipelines as SPM Toolbox (Batch Editor GUI).

The accompanying technical paper about the toolbox concept and methodology can be found at:
https://doi.org/10.1016/j.jneumeth.2016.10.019

Details on the peak-detection-free computation of respiratory volume and rate are outlined in our publication on the Hilbert transform: https://doi.org/10.1016/j.neuroimage.2021.117787

The PhysIO Toolbox is part of the TAPAS software collection of *Translational Algorithms for Psychiatry-Advancing Science*, whose design principles are described in the following paper: https://doi.org/10.3389/fpsyt.2021.680811.

There is also a video introduction into PhysIO and the TAPAS philosophy, given as part of the MRITogether23 Workshop on open science in MRI: *PhysIO, UniQC and a TAPAStry of Tools* (20 min YouTube Video).

Installation

Matlab

1. Unzip the TAPAS archive in your folder of choice
2. Open Matlab
3. Go to `/your/path/to/tapas/physio/code`
4. Run `tapas_physio_init()` in Matlab

Note: Step (4) executes the following steps, which you could do manually as well.

- Adds the `physio/code/` folder to your Matlab path

- Adds SPM to your Matlab path (you can enter it manually, if not found)
- Links the folder (Linux/Max) or copies the folder (Windows) `physio/code/` to `/your/path/to/SPM/toolbox/PhysIO`, if the PhysIO code is not already found there

Only the first point is necessary for using PhysIO standalone with Matlab. The other two points enable PhysIO's SPM integration, i.e., certain functionality (Batch Editor GUI, pipeline dependencies, model assessment via F-contrasts).

Getting Started

...following the installation, you can try out an example:

1. Download the TAPAS examples via running `tapas_download_example_data()` (found in `misc`-subfolder of TAPAS)
    - The PhysIO Example files will be downloaded to `tapas/examples/<tapas-version>/PhysIO`
2. Run `siemens_vb_ppu3t_sync_first_matlab_script.m` in subdirectory `Siemens_VB/PPU3T_Sync_First`
3. See subdirectory `physio/docs` and the next two section of this document for help.

You may try any of the examples in the other vendor folders as well.

Contact/Support

We are very happy to provide support on how to use the PhysIO Toolbox. However, as every researcher, we only have a limited amount of time. So please excuse, if we might not provide a detailed answer to your request, but just some general pointers and templates. Before you contact us, please try the following:

1. A first look at the FAQ (which is frequently extended) might already answer your questions.
2. A lot of questions (before 2018) have also been discussed on our mailinglist tapas@sympa.ethz.ch, which has a searchable archive.
3. For new requests, we would like to ask you to submit them as issues on our github release page for TAPAS, which is also an up-to-date resource to user-driven questions (since 2018).

Documentation

Documentation for this toolbox is provided in the following forms

1. Overview and guide to further documentation: README.md and CHANGELOG.md
    - README.md: this file, purpose, installation, getting started, pointer to more help
    - CHANGELOG.md: List of all toolbox versions and the respective release notes, i.e. major changes in functionality, bugfixes etc.
2. User Guide: The markdown-based GitLab Wiki, including an FAQ
    - online (and frequently updated) at http://gitlab.ethz.ch/physio/physio-doc/-/wikis/home.
    - offline (with stables releases) as part of the toolbox in folder `physio/wikidocs`:
        - plain text `.md` markdown files
        - as single HTML and PDF file: `documentation.{html,pdf}`
3. Within SPM: All toolbox parameters and their settings are explained in the Help Window of the SPM Batch Editor
4. Within Matlab: Extensive header at the start of each `tapas_physio_*` function and commenting
    - accessible via `help` and `doc` commands from Matlab command line
    - starting point for all parameters (comments within file): `edit tapas_physio_new`
    - also useful for developers (technical documentation)

Background

The PhysIO Toolbox provides physiological noise correction for fMRI-data from peripheral measures (ECG/pulse oximetry, breathing belt). It is model-based, i.e. creates nuisance regressors from the physiological monitoring that can enter a General Linear Model (GLM) analysis, e.g. SPM8/12. Furthermore, for scanner vendor logfiles (PHILIPS, GE, Siemens), it provides means to statistically assess peripheral data (e.g. heart rate variability) and recover imperfect measures (e.g. distorted R-peaks of the ECG).

Facts about physiological noise in fMRI:

- Physiological noise can explain 20-60 % of variance in fMRI voxel time series (Birn2006, Hutton2011, Harvey2008)
  - Physiological noise affects a lot of brain regions (s. figure, e.g. brainstem or OFC), especially next to CSF, arteries (Hutton2011).
  - If not accounted for, this is a key factor limiting sensitivity for effects of interest.
- Physiological noise contributions increase with field strength; they become a particular concern at and above 3 Tesla (Kasper2009, Hutton2011).
- In resting state fMRI, disregarding physiological noise leads to wrong connectivity results (Birn2006).
- Uncorrected physiological noise introduces serial correlations into the residual voxel time series, that invalidate assumptions on noise correlations (e.g., AR(1)) used in data prewhitening by all major analysis packages. This issue is particularly aggravated at short TR (<1s), and most of its effects can be suitably addressed by physiological noise correction (Bollmann2018)

Therefore, some kind of physiological noise correction is highly recommended for every statistical fMRI analysis.

Model-based correction of physiological noise:

- Physiological noise can be decomposed into periodic time series following heart rate and breathing cycle.
- The Fourier expansion of cardiac and respiratory phases was introduced as RETROICOR (RETROspective Image CORrection, Glover2000, see also Josephs1997).
- These Fourier Terms can enter a General Linear Model (GLM) as nuisance regressors, analogous to movement parameters.
- As the physiological noise regressors augment the GLM and explain variance in the time series, they increase sensitivity in all contrasts of interest.

Features of this Toolbox

Flexible Read-in

The toolbox is dedicated to seamless integration into a clinical research setting and therefore offers correction methods to recover physiological data from imperfect peripheral measures. Read-in of the following formats is currently supported (alphabetic order):

- BioPac `.mat` and `.txt` export files
- Brain Imaging Data Structure files (BIDS) `*_physio.tsv[.gz]/.json` files
- Custom logfiles: should contain one amplitude value per line, one logfile per device. Sampling interval(s) are provided as a separate parameter to the toolbox.
- General Electric
- Philips SCANPHYSLOG files (all versions from release 2.6 to 5.3)
- Siemens Manual Recordings (acquired via `IdeaCmdTool`), named "Siemens" files in the PhysIO interface (files `.ecg`, `.resp`, `.puls`)
- Siemens Automatic Recordings (XA60 and above, or WIP Advanced Physiologging (AdvPhysio), or CMRR C2P BOLD sequences), named "Siemens Tics" in the PhysIO interface (files `*_ECG.log`, `*_RESP.log`, `*_PULS.log`, `*_Info.log`)

- Siemens Human Connectome Project (preprocessed files `*Physio_log.txt`)

See also the Wiki page on Read-In for a more detailed list and description of the supported formats.

Robust Preprocessing

The toolbox has been extensively tested on different recording devices, field strengths and (clinical) study populations, and can handle a wide range of data imperfections in peripheral traces. Highlights are:

- Tested on various systems and subject populations (e.g., HCP, Children, 7T, ADHD, Major Depression, Psychosis)
- Cardiac Data: Iterative peak detection via Bayesian updates of current heart rate estimates (Steffen Bollmann)
- Respiratory Data: Novel volume estimation via Hilbert transform, respects, e.g., sighs, yawns, deep breaths (Samuel J. Harrison)

Physiological Noise Modeling

- Modeling physiological noise regressors from peripheral data (breathing belt, ECG, pulse oximeter)
  - State of the art RETROICOR cardiac and respiratory phase expansion
  - Cardiac response function (Chang et al, 2009) and respiratory response function (Birn et al. 2006) modelling of heart-rate variability and respiratory volume per time influence on physiological noise
  - Flexible expansion orders to model different contributions of cardiac, respiratory and interaction terms (see Harvey2008, Hutton2011)
- Data-driven noise regressors
  - PCA extraction from nuisance ROIs (CSF, white matter), similar to aCompCor (Behzadi2007)

Automatization and Performance Assessment

- Automatic Quality Assurance Diagnostic Figures (Heart beat-to-beat interval curves, respiration amplitude histograms) and flagging of
- Automatic creation of nuisance regressors, full integration into standard GLMs, tested for SPM8/12 (`multiple_regressors.mat`)
  - Data export as text file available for usage in other analysis packages
- Integration in SPM Batch Editor: GUI for parameter input, dependencies to integrate physiological noise correction in preprocessing pipeline
- Performance Assessment: Automatic F-contrast and tSNR Map creation and display for groups of physiological noise regressors, using SPM GLM tools via `tapas_physio_report_contrasts()`.

Requirements

- All specific software requirements and their versions are in a separate file in this folder, `requirements.txt`.
- In brief:
  - Typically, PhysIO needs Matlab to run, and a few of its toolboxes
  - Some functionality requires SPM (GUI via the SPM Batch Editor, nuisance regression, contrast reporting, writing residual and SNR images).
  - There is a standalone version within Neurodesk that does not require Matlab

Contributors

- Lead Programmer:
  - Lars Kasper, TNU & MR-Technology Group, IBT, University of Zurich & ETH Zurich
- Project Team:
  - Johanna Bayer, Donders Institute for Brain, Cognition and Behaviour, Nijmegen, Netherlands

    - Saskia Bollmann, Centre for Advanced Imaging, University of Queensland, Australia

- Project Team Alumni:
    - Steffen Bollmann, Centre for Advanced Imaging, University of Queensland, Australia
    - Sam Harrison, TNU Zurich
- Contributors (Code):
    - Eduardo Aponte, TNU Zurich
    - Tobias U. Hauser, FIL London, UK
    - Jakob Heinzle, TNU Zurich
    - Chloe Hutton, FIL London, UK (previously)
    - Miriam Sebold, Charite Berlin, Germany
    - External TAPAS contributors are listed in the Contributor License Agreement
- Contributors (Examples):
    - listed in EXAMPLES.md

Acknowledgements

The PhysIO Toolbox ships with the following publicly available code from other open source projects and gratefully acknowledges their use.

- `utils\tapas_physio_propval.m`
    - `propval` function from Princeton MVPA toolbox (GPL) a nice wrapper function to create flexible propertyName/value optional parameters
- `utils\tapas_physio_fieldnamesr.m`
    - recursive parser for field names of a structure
    - Matlab file exchange, adam.tudorjones@pharm.ox.ac.uk

We further acknowledge the generous support from the MathWorks (MA, USA), who supported Johanna Bayer for the 2022 MATLAB Community Toolbox Training Program established in collaboration with the International Neuroinformatics Coordinating Facility (INCF).

Cite Me

Main Toolbox and TAPAS Reference

Please cite the following papers in all of your publications that utilized the PhysIO Toolbox.

1. Kasper, L., Bollmann, S., Diaconescu, A.O., Hutton, C., Heinzle, J., Iglesias, S., Hauser, T.U., Sebold, M., Manjaly, Z.-M., Pruessmann, K.P., Stephan, K.E., 2017. The PhysIO Toolbox for Modeling Physiological Noise in fMRI Data. Journal of Neuroscience Methods 276, 56-72. https://doi.org/10.1016/j.jneumeth.2016.10.019
    - *main PhysIO Toolbox reference*
2. Frässle, S., Aponte, E.A., Bollmann, S., Brodersen, K.H., Do, C.T., Harrison, O.K., Harrison, S.J., Heinzle, J., Iglesias, S., Kasper, L., Lomakina, E.I., Mathys, C., Müller-Schrader, M., Pereira, I., Petzschner, F.H., Raman, S., Schöbi, D., Toussaint, B., Weber, L.A., Yao, Y., Stephan, K.E., 2021. TAPAS: an open-source software package for Translational Neuromodeling and Computational Psychiatry. Frontiers in Psychiatry 12, 857. https://doi.org/10.3389/fpsyt.2021.680811
    - *main TAPAS software collection reference*

You can include the following snippet in your Methods section, along with a brief description of the physiological noise models used:

> The analysis was performed using the Matlab PhysIO Toolbox ([1], version x.y.z, open-source code available as part of the

> TAPAS software collection: [2], https://www.translationalneuromodeling.org/tapas)

If you process respiratory traces, for example, to compute respiratory volume per time (RVT) or RETROICOR regressors, please also cite:

3. Harrison, S.J., Bianchi, S., Heinzle, J., Stephan, K.E., Iglesias, S., Kasper L., 2021. A Hilbert-based method for processing respiratory timeseries. NeuroImage, 117787. https://doi.org/10.1016/j.neuroimage.2021.117787
    - *superior RVT computation, preprocessing of respiratory traces*

Our FAQ contains a suggestion for a more comprehensive RETROICOR-related methods paragraph. See the main TAPAS README for more details on citing TAPAS itself.

Related Papers (Implemented noise correction algorithms and optimal parameter choices)

The following sections list papers that

- first implemented specific noise correction algorithms
- determined optimal parameter choices for these algorithms, depending on the targeted application
- demonstrate the impact of physiological noise and the importance of its correction

It is loosely ordered by the dominant physiological noise model used in the paper. The list is by no means complete, and we are happy to add any relevant papers suggested to us.

RETROICOR

4. Glover, G.H., Li, T.Q. & Ress, D. Image-based method for retrospective correction of Physiological motion effects in fMRI: RETROICOR. Magn Reson Med 44, 162-7 (2000).
5. Hutton, C. et al. The impact of Physiological noise correction on fMRI at 7 T. NeuroImage 57, 101-112 (2011).
6. Harvey, A.K. et al. Brainstem functional magnetic resonance imaging: Disentangling signal from PhysIOlogical noise. Journal of Magnetic Resonance Imaging 28, 1337-1344 (2008).
7. Bollmann, S., Puckett, A.M., Cunnington, R., Barth, M., 2018. Serial correlations in single-subject fMRI with sub-second TR. NeuroImage 166, 152-166. https://doi.org/10.1016/j.neuroimage.2017.10.043

aCompCor / Noise ROIs

8. Behzadi, Y., Restom, K., Liau, J., Liu, T.T., 2007. A component based noise correction method (CompCor) for BOLD and perfusion based fMRI. NeuroImage 37, 90-101. https://doi.org/10.1016/j.neuroimage.2007.04.042

RVT

9. Birn, R.M., Smith, M.A., Jones, T.B., Bandettini, P.A., 2008. The respiration response function: The temporal dynamics of fMRI signal fluctuations related to changes in respiration. NeuroImage 40, 644-654. doi:10.1016/j.neuroimage.2007.11.059
10. Jo, H.J., Saad, Z.S., Simmons, W.K., Milbury, L.A., Cox, R.W., 2010. Mapping sources of correlation in resting state FMRI, with artifact detection and removal. NeuroImage 52, 571-582. https://doi.org/10.1016/j.neuroimage.2010.04.246
    - *regressor delay suggestions*

HRV

11. Chang, C., Cunningham, J.P., Glover, G.H., 2009. Influence of heart rate on the BOLD signal: The cardiac response function. NeuroImage 44, 857-869. doi:10.1016/j.neuroimage.2008.09.029
12. Shmueli, K., van Gelderen, P., de Zwart, J.A., Horovitz, S.G., Fukunaga, M., Jansma, J.M., Duyn, J.H., 2007. Low-frequency fluctuations in the cardiac rate as a source of variance in the resting-state fMRI BOLD signal. NeuroImage 38, 306-320. https://doi.org/10.1016/j.neuroimage.2007.07.037
    - *regressor delay suggestions*

Motion (Censoring, Framewise Displacement)

13. Siegel, J.S., Power, J.D., Dubis, J.W., Vogel, A.C., Church, J.A., Schlaggar, B.L., Petersen, S.E., 2014. Statistical

improvements in functional magnetic resonance imaging analyses produced by censoring high-motion data points. Hum. Brain Mapp. 35, 1981-1996. https://doi.org/10.1002/hbm.22307

14. Power, J.D., Barnes, K.A., Snyder, A.Z., Schlaggar, B.L., Petersen, S.E., 2012. Spurious but systematic correlations in functional connectivity MRI networks arise from subject motion. NeuroImage 59, 2142-2154. https://doi.org/10.1016/j.neuroimage.2011.10.018

    - *definition of framewise displacement*

Copying/License

FAQ

Frequently Asked Questions (FAQ)

1. What is the PhysIO Toolbox?

PhysIO is a toolbox for model-based physiological noise correction of fMRI data.

PhysIO stands for Physiological Input/Output toolbox, which summarizes its core purpose. A quote from our paper:

> In short, the toolbox transforms physiological input, i.e. peripheral recordings, into physiological output, i.e. regressors encoding components of physiological noise [...] A modular Matlab implementation supports command-line operation and is compatible with all major fMRI analysis packages via the export of regressor text-files. For the Statistical Parametric Mapping SPM software package in particular, PhysIO features a full integration as a Batch Editor Tool, which allows user-friendly, GUI-based setup and inclusion into existing preprocessing and modeling pipelines.

2. How does PhysIO differ from other toolboxes for physiological noise correction for fMRI using peripheral recordings?

Citing from the introduction of our paper again

> ### Highlights
>
> - A Toolbox to integrate preprocessing of physiological data and fMRI noise modeling.
> - Robust preprocessing via iterative peak detection, shown for noisy data and patients.
> - Flexible support of peripheral data formats and noise models (RETROICOR, RVHRCOR).
> - Fully automated noise correction and performance assessment for group studies.
> - Integration in fMRI pre-processing pipelines as SPM Toolbox (Batch Editor GUI).

3. How do I cite PhysIO?

The **core references for PhysIO** are:

1. Kasper, L., Bollmann, S., Diaconescu, A.O., Hutton, C., Heinzle, J., Iglesias, S., Hauser, T.U., Sebold, M., Manjaly, Z.-M., Pruessmann, K.P., Stephan, K.E., 2017. *The PhysIO Toolbox for Modeling Physiological Noise in fMRI Data*. Journal of Neuroscience Methods 276, 56-72. https://doi.org/10.1016/j.jneumeth.2016.10.019

    - *main PhysIO Toolbox reference, also a good starting point to learn about more about the methods in PhysIO (see next question)*

2. Frässle, S., Aponte, E.A., Bollmann, S., Brodersen, K.H., Do, C.T., Harrison, O.K., Harrison, S.J., Heinzle, J., Iglesias, S., Kasper, L., Lomakina, E.I., Mathys, C., Müller-Schrader, M., Pereira, I., Petzschner, F.H., Raman, S., Schöbi, D., Toussaint, B., Weber, L.A., Yao, Y., Stephan, K.E., 2021. *TAPAS: an open-source software package for Translational Neuromodeling and Computational Psychiatry*. Frontiers in Psychiatry 12, 857. https://doi.org/10.3389/fpsyt.2021.680811

    - *main TAPAS software collection reference, see main TAPAS README for more details on TAPAS itself*

Please cite these papers if you use PhysIO in your work. Here is a minimum example snippet:

> The analysis was performed using the Matlab PhysIO Toolbox ([1], version x.y.z, open-source code available as part of the TAPAS software collection: [2], https://www.translationalneuromodeling.org/tapas)

If you use respiratory volume per time (RVT) regressors or preprocess respiratory traces for RETROICOR, please also cite:

3. Harrison, S.J., Bianchi, S., Heinzle, J., Stephan, K.E., Iglesias, S., Kasper L., 2021. A Hilbert-based method for processing respiratory timeseries. NeuroImage, 117787. https://doi.org/10.1016/j.neuroimage.2021.117787

    - *superior RVT computation, preprocessing of respiratory traces*

A **standard comprehensive snippet to include** in your method section could look like the following, assuming you use our specific implementation of RETROICOR, which uses Fourier expansions of different order for the estimated phases of cardiac pulsation (3rd order), respiration (4th order) and cardio-respiratory interactions (1st order) following (Harvey et al., 2008), and include respiratory volume per time (RVT) as well as heart-rate variability (HRV) regressors.

> Physiological noise correction was performed using the Matlab PhysIO Toolbox ([1], version x.y.z, open-source code available as part of the TAPAS software collection: [2], https://www.translationalneuromodeling.org/tapas). A RETROICOR model [4,5]) was employed, using Fourier expansions of different order for the estimated phases of cardiac pulsation (3rd order), respiration (4th order) and cardio--respiratory interactions (1st order) [6]. Furthermore, respiratory volume per time (RVT, [3,7]) and heart rate variability (HRV, [8]) were modeled.

4. Glover, G.H., Li, T.Q. & Ress, D. Image--based method for retrospective correction of PhysIOlogical motion effects in fMRI: RETROICOR. Magn Reson Med 44, 162-- 7 (2000).
5. Hutton, C. et al. The impact of PhysIOlogical noise correction on fMRI at 7 T. NeuroImage 57, 101--112 (2011).
6. Harvey, A.K. et al. Brainstem functional magnetic resonance imaging: Disentangling signal from PhysIOlogical noise. Journal of Magnetic Resonance Imaging 28, 1337--1344 (2008).
7. Birn, R.M., Smith, M.A., Jones, T.B., Bandettini, P.A., 2008. The respiration response function: The temporal dynamics of fMRI s ignal fluctuations related to changes in respiration. NeuroImage 40, 644–654. doi:10.1016/j.neuroimage.2007.11.059 PhysIO Toolbox | Citing this work 20
8. Chang, C., Cunningham, J.P., Glover, G.H., 2009. Influence of heart rate on the BOLD signal: The cardiac response function. NeuroImage 44, 857–869. doi:10.1016/j.neuroimage.2008.09.029

If you use **noise ROIs (aCompCor) or 12/24 regressor motion modeling**, also include the respective references:

9. Behzadi, Y., Restom, K., Liau, J., Liu, T.T., 2007. A component based noise correction method (CompCor) for BOLD and perfusion based fMRI. NeuroImage 37, 90–101. doi:10.1016/j.neuroimage.2007.04.042

    - *aCompCor*

10. Siegel, J.S., Power, J.D., Dubis, J.W., Vogel, A.C., Church, J.A., Schlaggar, B.L., Petersen, S.E., 2014. Statistical improvements in functional magnetic resonance imaging analyses produced by censoring high-motion data points. Hum. Brain Mapp. 35, 1981–1996. doi:10.1002/hbm.22307

    - *Motion Regressors*

4. Where do I find more documentation for PhysIO?

- The paper describing its structure, objective and modules
- README.md in the main folder when downloading
    - For help on installation and getting started

- Quickstart
  - PDF (or markdown .md file)
  - Tutorial matlab-scripts
- Reference Manual (for developers)

5. I am using FSL, AFNI, BrainVoyager, etc., for my fMRI analyses. Do I need SPM for PhysIO to work?

No, the basic functionality of PhysIO, i.e. creating nuisance regressors for your GLM analysis, is available in plain Matlab. The following extra functionality related to automatizing and assessing noise correction, require the installation of SPM:

- GUI (SPM Batch Editor)
- Pipeline dependencies (automatic input of realignment parameters, feed-in of multiple regressors file to GLM)
- Model assessment via F-tests and automatic F-map/tSNR report
- Noise-ROIs model (read-in of nifti files via SPM)

6. I am using device X for physiological recordings. Does PhysIO support the physiological logfile format Y?

Currently, PhysIO natively supports the following physiological logfile types:

- Brain Imaging Data Structure (BIDS)
  - Standard for peripheral recordings
  - both raw physiological traces and pre-computed pulse events are supported
- BioPac formats
  - Biopac `.mat`-export
    - assuming the following variables (as columns): `data`, `isi`, `isi_units`, `labels`, `start_sample`, `units`
    - See `tapas_physio_read_physlogfiles_biopac_mat.m` for details
  - Biopac `.txt`-export
    - assuming the following 4 columns, with one sample per row: respiratory, skin conductance (GSR), cardiac (PPG), and trigger signal (on/off)
- General Electric
- Philips SCANPHYSLOG files (`SCANPHYSLOG<DateTime>.log`; all versions from release 2.6 to 5.3)
- Siemens formats
  - Siemens VB (files `.ecg`, `.resp`, `.puls`)
  - Siemens VD/VE (files `*_ECG.log`, `*_RESP.log`, `*_PULS.log`)
    - including CMRR-derived multiband-files
  - Siemens Human Connectome Project log files (preprocessed 3 column files `*_Physio_log.txt`)

See Read-In of Logfiles for a detailed description of the expected file formats.

Furthermore, physiological recordings can be entered via a *custom* data format, i.e., providing one text file per device. The files should contain one amplitude value per line. The corresponding sampling interval(s) are provided as a separate parameter in the toolbox.

If your favourite logfile format is not supported, please contact the developers. We try everything to accomodate the read-in flexibility of the toolbox to your needs.

7. I am running the toolbox for a lot of subjects / on a remote server

without graphics. Can I somehow reproduce the output figures relevant to assess the data quality?

Yes you can, using the toolbox function `tapas_physio_review`. This function takes the physio-structure as an input argument, which is per default saved as `physio.mat` in the specified output folder of your batch job.

8. How do I interpret the various output plots of the toolbox?

Have a look at our publication: *The PhysIO Toolbox for Modeling Physiological Noise in fMRI Data* (http://dx.doi.org/10.1016/j.jneumeth.2016.10.019)

The figures there give a good overview of the toolbox output figures, in particular:

- Fig. S1 (supplementary): Philips Scan Timing Sync from `gradient_log` (explanation of `thresh.zero`, `thresh.sli`, `thresh.vol`, `thresh.vol_spacing`
- Fig. 3: Diagnostic Raw Time Series (cardiac cycle length curve, respiration histogram)
- Fig. 8C: Single Subject F-contrast results (cardiac regressors)
- Fig. 9: Group results/typical activation sites for F-contrasts of RETROICOR regressors (cardiac/resp/interaction)

9. I want to access subject's physiological measures, e.g. heart rate or respiratory volume (per time), before they enter the regressors. Where can I do that?

All intermediate data processing steps (e.g. filtering, cropping) of the peripheral data, including the computation of physiologically meaningful time courses, such as heart rate and respiratory volume, are saved in the substructure `ons_secs` ("onsets in seconds) of the physio-structure mentioned in question 7. This structure is typically saved in a file `physio.mat`.

`physio.ons_secs` then contains the different time courses, cropped to the acquisition window synchronized to your fMRI scan (the same values before synchronization/cropping, is found in `physio.ons_secs.raw`). Here are the most important ones:

- `ons_secs.t` = []; % time vector corresponding to c and r
- `ons_secs.c` = []; % raw cardiac waveform (ECG or PPU)
- `ons_secs.r` = []; % raw respiration amplitude time course
- `ons_secs.cpulse` = []; % onset times of cardiac pulse events (e.g. R-peaks)
- `ons_secs.fr` = []; % filtered respiration amplitude time series
- `ons_secs.c_sample_phase` = []; % phase in heart-cycle when each slice of each volume was acquired
- `ons_secs.r_sample_phase` = []; % phase in respiratory cycle when each slice of each volume was acquired
- `ons_secs.hr` = []; % [nScans,1] estimated heart rate at each scan
- `ons_secs.rvt` = []; % [nScans,1] estimated respiratory volume per time at each scan
- `ons_secs.c_outliers_high` = []; % onset of too long heart beats
- `ons_secs.c_outliers_low` = []; % onsets of too short heart beats
- `ons_secs.r_hist` = []; % histogram of breathing amplitudes

For a detailed list of all properties and their documentation, read the source code of `tapas_physio_new.m`

10. What is the order of the regressor columns in the multiple regressors file?

This depends on the physiological models (and their order) specified in the `model`-submodule of physio (or in the batch editor). The general order is outlined in Fig. 7A of the Main PhysIO Toolbox Paper. The []-brackets indicate the number of regressors:

1. RETROICOR cardiac regressors [2 x nOrderCardiac]
2. RETROICOR respiratory regressors [2 x nOrderRespiratory]
3. RETROICOR cardXResp interaction regressors [4 x nOrderCardiacXRespiratory]
4. HRV [nDelaysHRV]
5. RVT [nDelaysRVT]
6. Noise ROIs (PCA signatures and mean of each region) [nNoiseROIs x (nComponents+1)]
7. Other (included other text file) [nColumnsOtherFile]
8. Motion [6 or 12 or 24, depending on motion model]

If any of the models was not specified, the number of regressors is reduced accordingly.

11. How do I know whether the physiological noise correction worked?

The best way to assess the quality of the correction is an F-test over the respective physiological noise model regressors in the design matrix. Luckily, if you use SPM, the toolbox can create these contrasts and corresponding output plots with overlays of your brain automatically via calling the following function in the Matlab command window:

```
args = tapas_physio_report_contrasts(...
                'fileReport', 'physio.ps', ...
                'fileSpm', 'analysisFolder/SPM.mat', ...
                'filePhysIO', 'analysisFolder/physio.mat', ...
                'fileStructural', 'anatomyFolder/warpedAnatomy.nii')
```

Of course, you will have to adapt all paths to your `SPM.mat`, `physio.mat` and `anatomy.nii` files. There are more parameters to set (e.g. F-contrast thresholds), type `help tapas_physio_report_contrasts` for a list of options.

There should be whole-brain multiple-comparison corrected "activation" in physiological noise sites (similar to Fig. 8C or 9 in our paper.

If your F-contrast results differ or are absent, have a look at the *Diagnostic raw physiological time series*-plot and check whether it resembles Fig. 3 in the paper or whether there are any suspicious spikes in the heart cycle length.

Other than that, scan timing synchronisation is a major source of error, so always check the *Cutout actual scans* plot, whether the curves and scan events, TR etc. make sense.

12. Philips: I would like to use the gradient log for timing synchronization, but how do I set the thresholds?

Have a look at the following figure:

1. SUPPLEMENTARY MATERIAL

1.1. Figure S1

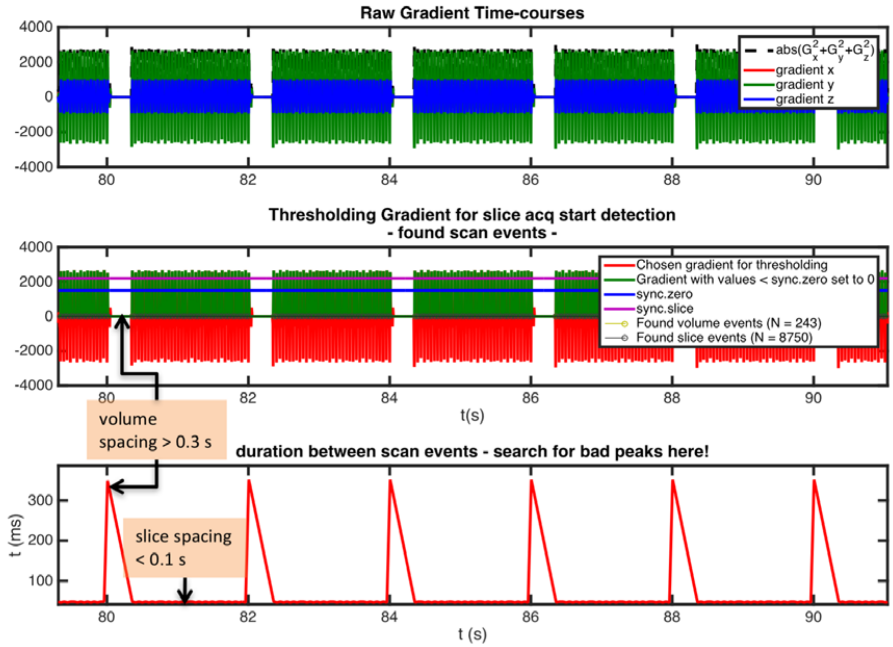


Figure S1: *Determining scan timing from gradient logging for Philips data (`sync.method = 'gradient_log'`). Besides cardiac and respiratory data, the Philips SCANPHYSLOG-file stores time courses of the x,y,z-gradients at a coarse temporal resolution (2 ms) (top). Their regularity, however, is sufficient to infer on slice and volume repetitions, and does not change over subjects. Thus, for each study, one only has to determine once which gradient shows the highest regularity (e.g. `sync.grad_direction = 'y'`, green curve), set lower thresholds for irrelevant and slice peaks (middle, `sync.zero = 1500`, blue line; `sync.slice = 2200`, magenta line), as well as volume peaks, if they differ from slice peaks (not shown, `sync.vol = []`). If volume peaks are not discernible, the non-equidistant slice spacing between volumes can be used to identify volume onsets (bottom, red curve, `sync.vol_spacing = 0.3` seconds).*

This figure can be found as figure S1 in the supplementary material of our paper.

The following heuristics might help with the threshold settings in the `sync` structure:

1. Note that these thresholds have to be set correctly only once for each functional sequence, i.e., usually once per study. Even small changes to scan geometry (e.g. slice tilt) between subjects shouldn't affect them significantly.

2. Setting the thresholds is an iterative procedure. You might start with the defaults, probably running into an error or warning

(`Warning: Invalid MinPeakHeight. There are no data points greater than MinPeakHeight.` or `Not enough volume/slice scan events found`). Then you inspect the figure output resembling the one above and adjust (usually lower) the thresholds in the order mentioned below.

3. There are three time courses in the upper of the three subplots shown in the figure. These time courses show the traces of the three gradient directions `x`, `y`, `z`. Choose the one as `sync.grad_direction` parameter that has the highest peaks and most regular features reflecting slice and/or volume scan events.

4. `sync.zero` has to be smaller than `sync.slice` and `sync.vol`. It should be about 4/5 of the typical peak height in the gradient trace. Note that you can set this thresholds (and all other) either in absolute values or relative to the maximum peak height. Set a value below 1, if you prefer the latter.

5. `sync.slice` should be about 9/10 of the typical peak height of a slice scan event.

6. `sync.vol`, if you set it, should be larger than `sync.slice`. It should be 9/10 of the peak height that stands out at the beginning of a volume, and is followed by some dozens of smaller peaks (for the slices) typically. It might be, however, that there is no such peak marking the start of a volume. If so, you might try `sync.vol_spacing` or leave it empty and rely on the slice thresholds exclusively

7. `sync.vol_spacing`, if set, should reflect the temporal spacing (in seconds), between the end of the previous volume and the start of the next one. The figure above gives some idea how to do that based on the bottom subplot that shows peak onset differences. If once every few seconds (your `TR`) you find an exposed peak, its height will give you the value for `sync.vol_spacing` (maybe reduce it by about 5-10ms to allow for timing inaccuracies).

13. How do I know which logfile type ('vendor') I have to choose?

- Typically, you will know your scanner manufacturer or the supplier of your peripheral recording device. The currently supported vendors can always be found in the SPM Batch Editor, as dropdown options for the vendor parameter in any PhysIO batch, and are also listed as cases in `tapas_physio_read_physlogfiles.m`.

- For Siemens, since there are a couple of formats, it is often helpful to check the extensions of the files (or the file name structure in general) see question 7.

- Sometimes you will have to look in the log files themselves and compare them to the examples provided on the Data Section of our homepage.

14. What does Parameter *XY* mean and what is its best setting?

Before you ask us directly, there are two simple ways to find out more about the parameters and options of the PhysIO toolbox:

- In SPM, you can use the Batch Editor as a Help GUI directly. If you open or create a TAPAS PhysIO Batch and click on any parameter, there will be useful information about its meaning and suitable values in the help window, located in the lower part of the Batch Editor.

- Within Matlab, type `edit tapas_physio_new`. This constructor function lists all parameters of the physio-structure with inline comments on their purpose and possible values.

15. I don't have Matlab, can I still use the PhysIO Toolbox?

Yes, our friends at NeuroDesk provide a docker container that contains the latest standalone version of SPM with inbuilt PhysIO. NeuroDesk runs in a browser and on any major operating system, and no Matlab license is required. For more details on PhysIO in NeuroDesk, check out their Tutorials on functional imaging.

Internally, this is using a compiled version of SPM with PhysIO. Contact us if you want to compile SPM with PhysIO yourself - for example, to deploy your own modifications of PhysIO to collaborators without Matlab.

Once SPM+PhysIO is compiled and you have the right Matlab Runtime Environment installed on your computer (determined by the Matlab version used for compilation), you can run a PhysIO SPM Batch file (`.m` or `.mat`) by calling

```
./run_spm12.sh ../MATLAB_Runtime/v99 batch physio_batch.mat
```

If you omit the batch file (but keep the keyword `batch`), this will just open the Batch Editor to interactively create and run a PhysIO batch.

16. PhysIO fails already when running your provided example data, e.g., with "Out of Memory" errors. What happened?

All PhysIO examples are thoroughly tested before each release using a unit/integration testing framework, but of course we cannot catch every bug in advance.

The most likely explanation, however, is that the setup of PhysIO on your machine was not successful. Here are a few things you can try:

- Did you run `tapas_physio_init()` and did it give any warnings?
- Do you have the subfolders of SPM in your Matlab path = e.g., you used `addpath(genpath('path/to/spm12'))`?
    - If yes, try `rmpath(genpath('path/to/spm12'));addpath('path/to/spm12')` and run the example again.
    - More details on this issue can be found in our GitHub Forum.

17. Which models do I have to include in my physiological regressor matrix? And which number of regressors (model order / delays) per model?

The question about how many and which regressors you need for successful noise removal depends a lot on your experimental design and your research questions. Here are a couple of rules of thumb:

1. To have sufficient degrees of freedom to estimate the parameters of the general linear model (GLM), it is often recommended to have a least 10 times as many datapoints as regressors in your model. That means, for 26 regressors in the GLM, it is advisable to have 260 volumes or more.
2. 26 regressors is the standard output if you include
    1. RETROICOR (3rd order cardiac = 6 regressors (1 cosine and one sine per order), 4th order respiratory = 8 regressors, and 1st order multiplicative terms (4: cos*cos, sin*sin, cos*sin, sin*cos),
    2. HRV (1 regressor),
    3. RVT (1) and
    4. motion realignment parameters (6)
    5. This list also gives an indication of the order of the regressors in the matrix, but see question 10 for details.
3. You can now reduce the number of regressors by the following observations:
    1. The RETROICOR 3/4/1 order is taken from a paper that optimized physiological noise removal in the brainstem [1]. You might not need the full model, if you are interested in other brain areas
    2. For example, in our PhysIO paper, figure 9 [2], where we evaluated this model for 35 subjects, you can see that the multiplicative terms explain noise mostly in the midbrain, acquaeduct and more inferior parts of the brainstem.
    3. Furthermore, also in that figure, you can see that the cardiac RETROICOR terms explain most of the variance (temporal SNR (tSNR) gains of up to 70%, compared to only motion correction), whereas the effect of both respiratory and multiplicative terms is one order of magnitude smaller (5% and 3% tSNR gains, respectively).
    4. So, you could probably leave out the multiplicative terms and reduce the number of regressors by 4, or reduced the respiratory terms to 2nd or 1st order, reducing them by 4 or 6, respectively. I have also seen 2nd cardiac order models, reducing by a further 2 regressors.
    5. In total, you would end up with 14 regressors for a RETROICOR 3/2/1, 10 regressors for RETROICOR 3/2/0 and 8 regressors for 2/2/0. In each case, you would still add the 8 regressors from HRV, RVT and motion.
4. To get a first idea which sets of regressors contribute to noise removal for your data, you can run F-Tests over the columns for the respective regressors only (see also question 11 and see where in the brain they explain significant variance (you can also use tapas_physio_compute_tsnr_gains to compute tSNR as in our figure).
    5. To formally compare whether it's warranted to include a set of regressors into your GLM, you would have to do model comparison (as in [1]). There is a toolbox for SPM that can do this automatically for the kind of regressor matrices that are output by PhysIO [3].

[1] Harvey, A.K., Pattinson, K.T.S., Brooks, J.C.W., Mayhew, S.D., Jenkinson, M., Wise, R.G., 2008. Brainstem functional magnetic resonance imaging: Disentangling signal from physiological noise. Journal of Magnetic Resonance Imaging 28, 1337–1344. https://doi.org/10.1002/jmri.21623

[2] https://www.sciencedirect.com/science/article/pii/S016502701630259X#fig0045 from: Kasper, L., Bollmann, S., Diaconescu, A.O., Hutton, C., Heinzle, J., Iglesias, S., Hauser, T.U., Sebold, M., Manjaly, Z.-M., Pruessmann, K.P., Stephan, K.E., 2017. The PhysIO Toolbox for Modeling Physiological Noise in fMRI Data. Journal of Neuroscience Methods 276, 56–72. https://doi.org/10.1016/j.jneumeth.2016.10.019

[3] Soch, J., Allefeld, C., 2018. MACS – a new SPM toolbox for model assessment, comparison and selection. Journal of Neuroscience Methods 306, 19–31. https://doi.org/10.1016/j.jneumeth.2018.05.017

18. I cannot find the answer to my question in the FAQ. Whom do I ask for help?

We are very happy to provide support on how to use the PhysIO Toolbox. However, as every researcher, we only have a limited amount of time. So please excuse, if we might not provide a detailed answer to your request, but just some general pointers and templates. Before you contact us, please try the following:

1. A first look at the FAQ (which is frequently extended) might already answer your questions.
2. For older versions of PhysIO, a lot of questions have been discussed on our now defunct mailinglist tapas@sympa.ethz.ch, but its archive is still searchable.
3. For new requests, we would like to ask you to submit them as issues on our GitHub page for TAPAS.

Examples

Example Datasets for PhysIO

The following datasets are available to explore the read-in and modeling capabilities of PhysIO. They can be downloaded by running the function `tapas_download_example_data()` in Matlab, which is located in the `misc` subfolder of the TAPAS software release you downloaded (probably here).

Afterwards, the examples can be found in `tapas/examples/<tapasVersion>/PhysIO` as different subfolders (`vendor/device`) and shall be run directly from within these individual folders.

Besides the raw physiological logfiles, each example contains example scripts to run PhysIO as

- SPM job (`\\\*spm_job.mat`)
- editable SPM job (`\\\*spm_job.m`)
- plain matlab script (`\\\*matlab_script.m`)

Brain Imaging Data Structure (BIDS)

CPULSE 3T

Courtesy of Hrvoje Stojic, Max Planck UCL Centre for Computational Psychiatry and Ageing Research, University College London

Vendor-computed (software: Spike2) cardiac pulse events from PPU (finger plethysmograph) data, Siemens 3T scanner, Multiband CMRR sequence

Description: This datasets contains the (compressed) tab-separated value (`.tsv.gz`) files as well as the meta-file (`.json`) holding sampling rate of the physiological recording, and its relative onset to scanning, in adherence with the BIDS standard for peripheral recordings files.

PPU 3T

Courtesy of Hrvoje Stojic, Max Planck UCL Centre for Computational Psychiatry and Ageing Research, University College

London

PPU (finger plethysmograph) and breathing belt, Siemens 3T scanner, Multiband CMRR sequence

Description: Similar to CPULSE3T (same acquisition system), but now with analog data instead of vendor-detected pulses, data from different subject

PPU 3T Separate Files

Courtesy of Alexandre Sayal CIBIT, University of Coimbra

PPU (finger plethysmograph) and breathing belt, Siemens 3T scanner, Multiband CMRR sequence

Description: Similar to PPU 3T, but two separate BIDS files for cardiac and respiratory recordings, because of differing sampling rate (5 vs 20 ms). Externally converted from `SIEMENS_VD/PPU3T_For_BIDS` example below.

General Electric

PPU 3T

Courtesy of Steffen Bollmann, Kinderspital Zurich and ETH Zurich

PPU (finger plethysmograph) and breathing belt, General Electric 3T scanner

Description: Similar to PPU, but acquired on a GE system with two separate output logfiles for pulse oximetry and breathing amplitude, sampled with 40 Hz. The quality of the signal is particularly challenging, stemming from a patient population.

Philips

ECG 3T

Courtesy of Sandra Iglesias, Translational Neuromodeling Unit, ETH & University of Zurich

4-electrode ECG and breathing belt, Philips 3T Achieva scanner

Description: Standard example; shows how to use volume counting either from beginning or end of run to synchronize physiological logfile with acquisition onsets of fMRI scans.

ECG 7T

Courtesy of Zina-Mary Manjaly, University Hospital Zurich

4-electrode ECG and breathing belt, Philips 7T Achieva scanner

Description: The ECG data for ultra-high field data is typically much noisier than at 3 Tesla. Therefore, R-wave peaks are frequently missed by prospective trigger detection and not marked correctly in the logfile. This example shows how to select typical R-wave-peaks manually and let the algorithm find the heartbeat events.

PPU 3T

Courtesy of Diana Wotruba, University and University Hospital of Zurich

PPU (finger plethysmograph) and breathing belt, Philips 3T Achieva scanner

Description: Similar to ECG3T, but a plethysmograph instead of an ECG was used to monitor the cardiac pulsation. Example shows how to extract heart and breathing rate.

PPU 7T

Courtesy of Jakob Heinzle and Lars Kasper, TNU, University Zurich and ETH Zurich

PPU (finger plethysmograph) and breathing belt, Philips 7T Achieva scanner

Description: Challenging cardiac data that requires bandpass-filtering during preprocessing, since it is compromised by both high frequency noise (from the scanner, modulated at every slice TR) and low frequency noise (breathing modulation).

Siemens - VB

Siemens has different physiological logfile formats, for which examples are provided here. A detailed description of these formats is on a different wiki page.

This is the older Siemens log file format (also available via *manual recording*), which is part of software release _VB_, and can be determined by the file extensions `.resp`, `.ecg`, `.puls`, in combination with an optional `.dcm` DICOM header file for the first (or last) acquired volume.

A lot of 7T scanners still use this format, but it is also the default on modern 3T systems, if you don't have C2P sequences for fMRI (e.g., from CMRR) or WIPs from Siemens (see below).

ECG 3T

Courtesy of Miriam Sebold, Charite Berlin, and Quentin Huys, TNU Zurich

4-electrode ECG data, Siemens 3T scanner, logfile version 1

Description: Similar to ECG 3T, but acquired on a Siemens system with only one logfile for ECG data. The quality of the signal is challenging, stemming from a patient population.

PPU3T (Sync First and Sync Last)

Courtesy of Alexander Ritter, University of Jena, Germany

Siemens 3T pulse oximetry and respiratory bellows data, logfile version 1 DICOM header file of first and last (382nd) volume of an fMRI run, respectively.

Description: This data covering a complete scan session of a healthy volunteer showcases scan timing synchronization using the DICOM timestamps in an intricate case, where the physiological logfile spans the whole scan session (and not only the fMRI run). See TAPAS github issue #55 for further details.

ECG 3T - Logversion 3

Courtesy of Shahin Safa, see TAPAS GitHub issue 204

4-electrode ECG data, Siemens scanner, logfile version 3 corresponding respiratory data: Resp 3T - Logversion 1

Description: This is an fMRI study on the auditory system of the brain, which explains the long TR (10 s), to put scanning gaps when presenting the sound to the subject.

Resp 3T - Logversion 1

Courtesy of Shahin Safa, see TAPAS GitHub issue 204

Respiratory bellows data, Siemens scanner, logfile version 1 corresponding cardiac data: ECG 3T - Logversion 3

Description: This is an fMRI study on the auditory system of the brain, which explains the long TR (10 s), to put scanning gaps when presenting the sound to the subject.

Resp 3T - Logversion 3

Courtesy of Lars Kasper, University Health Network Toronto, Canada

Respiratory bellows data, Siemens Prisma 3T, logfile version 3

Description: Short fingertapping run with logging automatically switched off after about 2 minutes (nominally 5) due to ECG channels not connected, but requested for recording. Biomatrix sensors were not available, but are logged as 4 extra channels with constant values here.

Siemens - HCP

The Human Connectome Project uses Siemens scanners, and the logfile format that comes with their published data seems to be pre-converted and custom (even though the documentation desribes the VB format). We have implemented an own reader for that and written a little tutorial for a single subject dataset of the HCP.

https://github.com/translationalneuromodeling/tapas/issues/6#issuecomment-361001716

If you download the whole dataset (including functional image files), this example with the additional batches mentioned below also demonstrates how to use the toolbox for model assessment using statistical maps (F-contrasts).

HCP (Subject 178748)

You will have to download the dataset from the HCP yourself, we just provide the matlab batches and the physiological logfile `tfMRI_MOTOR_LR_Physio_log.txt` here.

For consistency with the other example files, the batch files have been renamed compared to the blog entry:

- `batch_preproc.m` -> `batch_preproc.m`
- `batch_physio.m` -> `siemens_hcp_ppu3t_spm_job.m`
- `batch_glm.m` -> `batch_glm.m`

If you want to run the preproc and glm batch, place them on the same level as the subject folder `178748` for the downloaded data. The physio-batch shall reside in the same folder as the physiological logfile `tfMRI_MOTOR_LR_Physio_log.txt`.

Siemens - VD/VE Tics

This is the most recent logfile format of Siemens, included in Software releases _VD_, _VE_ and sometimes referred to as the *Tics* format, because all time stamps in all files refer to the same reference point (start of the day) and count in the same intervals or "*tics*" of 2.5 ms from there.

You will recognize this file format via the extensions `\\\_Info.log` (or `\\\_AcquisitionInfo.log`), `\\\_RESP.log`, `\\\_ECG.log` and `\\\_PULS.log`. Sometimes, it is also written into the DICOM header (`.dcm`) file of your functional data directly. In this case, use extractCMRRPhysio.m to convert it to the above separate files before using PhysIO.

Most modern Siemens scanners, such as the Prisma or 7T Terra, use this format.

There are a couple of variants for this format around (e.g., with the WIP Multiband Protocol that is distributed to multiple sites), and PhysIO tries to support all of them.

PPU 3T

Courtesy of Saskia Bollmann, Centre for Advanced Imaging, University of Queensland, Brisbane, Australia

Pulse oximetry and breathing belt data, Siemens Prisma 3T, logfile version `EJA_1`, multi-echo fMRI (3 echoes)

The UUID and date/time stamps were altered for anonymization.

PPU 3T Separate Files

Courtesy of Alexandre Sayal CIBIT, University of Coimbra

PPU (finger plethysmograph) and breathing belt, Siemens 3T scanner, Multiband CMRR sequence

Description: Raw data that was used to convert to two separate BIDS files above (`BIDS/PPU3T_Separate_Files`) for cardiac and respiratory recordings, because of differing sampling rate (5 vs 20 ms).

The UUID and date/time stamps were altered for anonymization.

Technical Documentation: Read-in Brain Imaging Data Structure (BIDS)

PhysIO supports physiological logfiles prepared according to the BIDS standard

- In brief, BIDS files are (optionally compressed) tab-separated values (`\*.tsv\[.gz\]`) files that contain raw traces of peripheral recordings from cardiac and respiratory sources, as well as scan trigger events
- The header of the columns of this `\*.tsv` file, as well as meta-information, such as sampling rate and relative onset of

physiological logging to MRI scan onset is described in an accompanying `\*.json` file

- It is assumed to have the that this `\*.json` file has the same name (apart from the extension) as the `\*.tsv` file
- If PhysIO does not find this file, you can manually enter the timing information in the `log_files` structure, and a default column order of (cardiac, respiratory, trigger) is assumed

- Example `\*.tsv` file (with cardiac, respiratory, trigger column :

```
-0.949402 -0.00610382 0
-0.949402 -0.00610382 0
-0.951233 -0.00915558 0
-0.951233 -0.00915558 0
-0.953064 -0.0122073  0
-0.953064 -0.0122073  0
-0.95459  -0.0076297  1
-0.95459  -0.0076297  0
```

- Example `\*.json` file:

```
{
    "SamplingFrequency": 50.0,
    "Columns": [
        "cardiac",
        "respiratory",
        "trigger"
    ],
    "StartTime": -255.45
}
```

- Note that `StartTime` refers to when the physiological recording started relative to the first scan volume of the fMRI run, which means that typically this value is negative, because one starts the recording before the onset of scan volumes.
- See `tapas_physio_read_physlogfiles_bids.m` for more details and technical documentation.

BioPac

Mat-file Export (`.mat`)

- assuming the following variables (as columns): `data`, `isi`, `isi_units`, `labels`, `start_sample`, `units`
- See `tapas_physio_read_physlogfiles_biopac_mat.m` for details

Single Text File Export (`.txt`)

```
RESP - RSP100C    GSR - EDA100C-MRI    PPG - PPG100C    Marker
-0.949402    -0.00610382    0.0134277    0
-0.949402    -0.00610382    0.0134277    0
...
-0.951233    -0.00915558    0.0204468    11
-0.951233    -0.00915558    0.0204468    11
-0.953064    -0.0122073    0.0259399    0
...
```

Custom

If you have logfile data from any other vendor than the ones specified below, you may still use it with PhysIO:

1. Export your traces from cardiac and breathing recording devices into 2 text files and select `log_files.vendor = 'Custom'`. The format is explained in `tapas_physio_new` or the help window of the Batch Editor:
    - 'Custom' expects the logfiles (separate files for cardiac and respiratory) to be plain text, with one cardiac (or respiratory) sample per row;

- If heartbeat (R-wave peak) events are recorded as well, they have to be put as a 2nd column in the cardiac logfile by specifying a 1; 0 in all other rows, e.g.

```
0.2  0
0.4  1 <- cardiac pulse event
0.2  0
-0.3 0
```

2. You have to specify the sampling intervals for these log files (in seconds), via `log_files.sampling_interval`, e.g. [0.01 0.02] if you have 10 ms (100 Hz) and 20 ms (50 Hz) sampling intervals (frequencies) for cardiac and respiratory data, respectively
3. You will probably have to change `log_files.relative_start_acquisition`, if logging of your physiological recording device does not start synchronized to the first fMRI volume.

General Electric (GE)

- Very similar to custom format
- One text file each for ECG, pulse oximetry and respiratory data, e.g., `ECGData_epiRT_phys_0921201215_38_08` or `RespData_epiRT_phys_0921201215_38_08`
- One amplitude entry per line, e.g.,

```
2626
2649
2673
2699
2727
2755
```

- sampling rate is determined as a setting beforehand, has to be noted manually (not in log file)

Philips

- Physiology automatically recorded into `SCANPHYSLOG\_<Date>\_<Time>.log` (one file per scan) as soon as ECG is connected to scanner, and scan is started
- tabular text (ascii) format, different columns for ECG, pulse oximetry and breathing data
  - additionally, trigger events and gradient timecourses are logged, and can be used for synchronization by the toolbox

```
## <YourScannerLocation>, Release r32 (SWID 77)
## Mon 01-01-2011 12:00:01
## 2628 1214 775 387 -1024 -323 -780 -274 0
## Dockable table = FALSE
# v1raw v2raw  v1 v2  ppu resp  gx gy gz mark
-458 325  -494 2  0 -762  0 0 0 0000
-497 284  -527 -32  0 -745  0 0 0 0000
-533 251  -560 -68  0 -745  0 0 0 0000
-571 219  -592 -104  0 -745  0 0 0 0000
-606 190  -623 -139  0 -745  0 0 0 0000
```

- fixed sampling rate (2 ms for cable connection, 1/496 ms for Wi-Fi devices)

Siemens

Manual Recording (`IdeaCmdTool`, `Siemens VB`)

Physiological data collection on the Siemens scanners uses the physiological monitoring unit (PMU). The initial sampling is performed at 400 Hz, but through the PMU buffer the effective sampling intervals are ECG: 2.5 ms, RESP: 20 ms, PULS: 20 ms and EXT: 5 ms.

There are several ways to control the physiological data collection. The 'manual' version is available on all platforms. It uses the

telnet `mpcu/ideacmdtool` to manually start and stop the log file acquisition. The log files (`logFileName.ecg`, `logFileName.resp`, `logFileName.puls`, `logFileName.ext`) are stored in `\\MedCom\\log`. More details on how to record these data can be found here or in the "Other Miscellaneous Topics" slides from the IDEA course.

This is the classical method to do physiological recordings on a Siemens scanner. According to the CMRR multi-band sequence manual (see this blogpost), it used to be accurate in terms of timing until Siemens software release VB (hence the name in PhysIO), but has since been superseded by more integrated methods (like CMRR Tics logging or the AdvPhysio Work-in-progress (WIP) by Siemens, which can write the physiological traces as part of the DICOM image header files, see below).

General Properties

An example of a `.puls` logfile is given below. The data are stored in one long line. The text between 5002 and 6002 forms the header, and the text between 5003 and 6003 the footer. Important information in the footer is the `LogStartMDHTime` and the `LogStopMDHTime` (in ms since midnight), which can be used to synchronize the logfiles with the DICOM images using the `AcquisitionTime` in the DICOM header (in hhmmss.ms). The values 5000 and 6000 are inserted into the signal trace and indicate trigger events. Note that only the modality which is selected to be displayed during the acquisition will have triggers.

We use the time stamp of the clock of the Measurement Data Header (MDH), i.e., computer that controls the scanner, to synchronize with the DICOMs, because this computer also controls the creation of the scan data, i.e., reconstructed DICOM images. This is in accordance to other packages reading Siemens physiological logfile data, e.g., Chris Rorden's PART, with a detailed explanation on the DICOM timestamp in `AcquisitionTime` found here.

```
1 2 40 280 5002 Logging PULSE signal: reduction factor = 1, PULS_SAMPLES_PER_SECOND = 50; PULS_SAMPLE_INTERVAL
= 20000 6002 1653 1593 1545 1510 1484 ...
ACQ FINISHED
 6002 3093 3096 3064 5000 3016 2926 5003
ECG  Freq Per: 0 0
PULS Freq Per: 66 906
RESP Freq Per: 18 3260
EXT  Freq Per: 0 0
ECG  Min Max Avg StdDiff: 0 0 0 0
PULS Min Max Avg StdDiff: 731 1113 914 1
RESP Min Max Avg StdDiff: 3080 4540 3779 73
EXT  Min Max Avg StdDiff: 0 0 0 0
NrTrig NrMP NrArr AcqWin: 0 0 0 0
LogStartMDHTime:  47029710
LogStopMDHTime:   47654452
LogStartMPCUTime: 47030087
LogStopMPCUTime:  47652240
6003
```

Updates Logversion 3

The above specification for physiological Siemens logfiles has been changed by the new logfile version 3 for both cardiac and respiratory data:

1. There may be multiple info regions between `5002` and `6002` tags, interleaved with the physiological trace, e.g., to log an update in the respiratory cushion gain.
2. The `.ecg` file has now 4 instead of 2 channels.
3. The `.resp` file now contains not only the respiratory bellows signal, but also 4 datapoints per sampling interval of the biomatrix signals (integrated sensors in patient bed for breathing detection, e.g., in Siemens Vida series).

A logfile in version 3 therefore adheres to the following template (note the logfile version indicated by keyword `LOGVERSION` at the start):

```
<Header> 5002 <LOGVERSION XX> 6002
<[optional] training trace data> 5002 uiHwRevisionPeru ... [optional] 6002
5002 <infoRegion3> 6002 5002 <infoRegion4> 6002 ... 5002 <infoRegionN> 6002
<trace data 1 (all channels, arbitrary number of samples, trigger markers 5000, 6000)> ...
5002 <infoRegionN+1> 6002
<trace data 2 (all channels, arbitrary number of samples, trigger markers 5000, 6000)> ...
5002 <infoRegionN+2> 6002 ...
<trace data M> ... 5003
```

PhysIO automatically reads the Siemens logfiles correctly for all modalities and logfile versions 1,2 and 3.

CMRR Sequence / WIP Advanced Physio Logging (`Siemens_VD`, `Siemens_Tics`)

The CMRR sequence on VD/VE/XA also allows the automatic recording of physiological log files (to be selected in the sequence special card). For more information have a look at the manual. The physiological traces are stored in logFileName_PULS.log, logFileName_RESP, logFileName_ECG.log or in the DICOM image file header (use readCMRRPhysio to extract it into separate files for PhysIO). Timing information is stored in logFileName_Info.log and external trigger events in logFileName_EXT.log.

An example of the current format (December 2017, Release 016a) for the logFileName_Info.log is given below:

```
UUID        = 7a16ea95-ac36-4ee3-9b76-bbb686ac07ca
ScanDate    = 20171206_150609
LogVersion  = EJA_1
LogDataType = ACQUISITION_INFO
NumSlices   = 48
NumVolumes  = 30
NumEchoes   = 3

VOLUME   SLICE   ACQ_START_TICS  ACQ_FINISH_TICS  ECHO

     0       0         21754755         21754762     0
     0      12         21754755         21754762     0
     0      24         21754755         21754762     0
     0      36         21754755         21754762     0
     0       0         21754763         21754770     1
     0      12         21754763         21754770     1
     0      24         21754763         21754770     1
     0      36         21754763         21754770     1
     0       0         21754771         21754779     2
     0      12         21754771         21754779     2
     0      24         21754771         21754779     2
     0      36         21754771         21754779     2
     0       5         21754787         21754795     0
     0      17         21754787         21754795     0
     0      29         21754787         21754795     0
     0      41         21754787         21754795     0
     0       5         21754795         21754803     1
     0      17         21754795         21754803     1
```

The accompanying logFileName_PULS.log looks like this:

```
UUID        = 7a16ea95-ac36-4ee3-9b76-bbb686ac07ca
ScanDate    = 20171206_150609
LogVersion  = EJA_1
LogDataType = PULS
SampleTime  = 2

ACQ_TIME_TICS  CHANNEL  VALUE  SIGNAL

     21747857     PULS   2086
     21747859     PULS   2076
     21747861     PULS   2071
     21747863     PULS   2057
     21747867     PULS   2038
     21747869     PULS   2024
     21747871     PULS   2010
     21747873     PULS   1991
     21747875     PULS   1976
     21747877     PULS   1962
```

PhysIO uses the logFileName_Info.log to synchronize the physiological traces with the data acquisition. Note that the reference slice does not yet take into account the multiband slice ordering, but just assumes an even distribution. Older version of the CMRR sequence produced slightly different output files, which might work. Please log an issue if you have a very different format that is not supported.

Human Connectome Project

Disclaimer: Most of the information below is a best guess from the developers, but without any guarantee of accuracy.

The physiological log file (`\*paradigm\*\_Physio_log.txt`) distributed with the Human Connectome Project data contains respiratory and puls-oximeter data in one file. The first column marks when data acquisition is performed, the second and third contain the respiratory and puls-oximeter traces, respectively. The files are written at a sampling rate of 400Hz and start and end with the scan. PhysIO does provide a reader, you just need to select the appropriate option in the file format tab. An example is provided below:

```
1    1904    1756
1    1904    1756
1    1907    1754
1    1904    1756
1    1907    1756
1    1907    1756
1    1907    1758
1    1907    1760
1    1910    1760
1    1907    1762
1    1907    1762
1    1904    1764
1    1907    1766
1    1904    1766
1    1907    1768
1    1907    1768
1    1904    1768
1    1904    1770
1    1904    1770
1    1904    1772
1    1904    1772
1    1904    1776
1    1904    1776
1    1904    1778
1    1904    1780
```

Version History (Changelog)

RELEASE INFORMATION

Current Release

Current version: PhysIO Toolbox Release R2024a, v9.0.0

January 29th, 2025

Major Release Notes (v9.0.0)

Added

- BIDS writer: write out BIDS-compatbile physiological logfiles (`.tsv.gz` and `.json`) from any vendor format
- Greatly expanded Review visualization (`tapas_physio_review`): Allow more detailed re-creation of figures from online execution, control visual verbosity retrospectively (e.g., for debugging)
- Read-In of field `AcquisitionTime` from BIDS `.json` side-car file for converted Siemens DICOMs after dcm2niix conversion for synchronization (Gitlab issue 109)

Changed

- Expanded usage of scan trigger trace (continuous, binary, on/off vs alternating levels)

Fixed

- Bugfix processing of short Siemens IdeaCmdTool logfiles (stop before scan ending)

- DICOM Read-in for files without extension

Minor Release Notes (v8.2.0)

Added

- Interface `tapas_physio_test` to TAPAS-generic `tapas_test` function
- Added suport for logfile version 3 of Siemens physio recordings
  - multi ECG/Resp channels and interleaved status messages
  - new integration test for Siemens VB Logversion 3
- Added support for ADInstruments/LabChart Txt-export format (see CUBRIC Seminar Example and gitlab branch #107)

Fixed

- Removed dependence on `nanmean` (Statistics Toolbox)
  - See GitHub issue #205
- Compatibility with multiple SPM toolbox locations for `lmod` (GitHub issue #211)
  - as listed in `spm_get_defaults('tbx')`
- Refactoring of Philips read-in to support novel 12-column logfile version, see GitHub issue #207
- Unit/Integration tests for filtered traces (cardiac and respiratory) switched to absolute tolerances (relative problematic for traces close to zero)

Minor Release Notes (v8.1.0)

Added

- Compatibility of whole code base with Matlab compiler in order to run `spm_make_standalone`
  - provides oppurtunity to run SPM Batch Editor GUI version of PhysIO without Matlab license requirement
  - compiled version readily available within Neurodesk
- BIDS Read-in for separate cardiac/respiratory trace files (e.g., due to different sampling frequencies)
  - see GitHub Issue #164 and pull request #167 by @alexsayal
  - Additional unit tests for new read-in and example data

Changed

- Switch for certain toolbox functions (e.g., `imtool`) to only run in non-compiled code

Fixed

- Documentation (function headers, see Github issue #149)
- Typos in unused function (spotted in compilation)
- Synchronization SIEMENS AcquisitionLog / Physiological files (see Github issue #172)
  - better visualization of sync, clearer error messages if dummy scans not found

Bugfix Release Notes (v8.0.1)

Changed

- Citation of novel TAPAS paper in README and CITATION

Major Release Notes (v8.0.0)

Added

- New method for computing respiratory volume per unit time (RVT) via the Hilbert transform.
  - Publication: Harrison et al., "A Hilbert-based method for processing respiratory timeseries", NeuroImage, 2021. https://doi.org/10.1016/j.neuroimage.2021.117787
  - This is now the default option, but the old method is available by setting `physio.model.rvt.method = 'peaks'` (or the equivalent within the SPM batch editor).
- Respiratory preprocessing now includes an optional de-spiking step based on median filtering.

Changed

- Now possible to change the frequencies of the respiratory filtering during preprocessing via `physio.preproc.respiratory.filter` (or the equivalent within the SPM batch editor).
- More robust detrending of raw respiratory timeseries via windowed padding before filtering.

Fixed

Bugfix Release Notes (v7.3.2)

Added

- version number `physio.version` in physio-struct (Gitlab issue #101)

Fixed

- Subfolder of SPM in path (fieldtrip) created ambiguous function calls to overloaded functions (Gitlab issue #102 and Gihub issue #110)
  - e.g., `filtfilt` in `tapas_physio_filter_respiratory` created flat regressors for v7.3 for Siemens VD example dataset
  - now subfolders of SPM will be removed when calling `tapas_physio_main_create_regressors`

Bugfix Release Notes (v7.3.1)

Fixed

- PPU read-in works for BioPac mat-file now
  - correct column labels and cardiac modality (Github Issue #103)
  - thank you to Manon Durand-Ruel @mdurandruel for reporting

Minor Release Notes (R2020a, v7.3.0)

Added

- Added descriptive names for the multiple regressors matrix.
  - Closes GitLab issue #82.
  - Now possible to straightforwardly inspect `physio.model.R` and the contents of `physio.model.output_multiple_regressors` using `physio.model.R_column_names`.
  - Added `tapas_physio_guess_regressor_names()` to maintain backwards compatibility.
- New example datasets Siemens VB PPU3T with DICOM Sync (courtesy of Alexander Ritter, Jena, Germany)
- More versatile control on figure visibility, saving and closing during `main` and `review` runs of PhysIO
  - feature provided by Stephan Heunis, TU Eindhoven, The Netherlands (github issue #89)
  - figures can now be plotted in the background without disturbing interactive Matlab sessions, and can be (more) selectively saved and closed during execution of `tapas_physio_review`

- more comprehensive support within `tapas_physio_main_create_regressors` to follow

Bugfix Release Notes (v7.2.8)

Fixed

- Bug(s) when checking SPM and PhysIO paths in `tapas_physio_init` under certain particular Matlab path environment settings (Gitlab merge request !37)
  - e.g., when add `physio-public/code` manually without subfolder
  - or if spm existed as a folder name without being added to path

Bugfix Release Notes (v7.2.7)

Changed

- Reimplemented `tapas_physio_filter_respiratory.m`.
  - Closes GitLab issue #98.
  - Reduces the cut-off frequency of the high-pass filter, so as to behave more like a detrending step. Previous value of 0.1 Hz could distort e.g. sigh breaths, which can easily last upwards of 10 s.
  - Reimplements the filters themselves, with a higher filter order, extra padding, and a two step high-pass and low-pass procedure (i.e. rather than band pass) to improve the overall stability.

Bugfix Release Notes (v7.2.6)

Fixed

- Meaningful error message for `auto_matched` peak detection, if time series is too short
  - at least 20 peaks (and pulse templates) are required to create a pulse template
  - this is now stated explicitly, if time series is too short
  - reported by Joy Schuurmans Stekhoven, TNU, as gitlab issue #92

Bugfix Release Notes (v7.2.5)

Fixed

- Corrected documentation for `preproc.cardiac.initial_cpulse_select.min` parameter
  - threshold for peak relative to z-scored time series, *not* correlation
  - reported by Sam Harrison, TNU, as gitlab issue #95

Bugfix Release Notes (v7.2.4)

Fixed

- Stop docked figure default throwing error with `-nodisplay`
  - allows generating saved figure without a display, e.g., on remote server
  - bugfix provided by Sam Harrison, TNU

Bugfix Release Notes (v7.2.3)

Fixed

- Bugfix manual peak selection (Github issue #85, Gitlab #90)
  - did not work because of figure handling

Bugfix Release Notes (v7.2.2)

Fixed

- Bugfix Siemens VB (`*.resp, *.puls, *.ecg`)
  - Synchronization to DICOM time stamp did not work for extended physiological recordings (not starting/ending with functional run) due to ignored absolute start time stamp
  - reported by Alexander Ritter, Jena, Germany (Github issue #55, Gitlab #87)
  - probably fixes Github issue #63 (Gitlab #86) as well

Bugfix Release Notes (v7.2.1)

Changed

- PhysIO: removed Matlab statistics toolbox dependency for PCA by SVD implementation (thanks to Benoît Beranger, pull request 64)
  - new function `tapas_physio_pca` allows for switch between stats and native SVD implementation of PCA
  - comes with unit tests checking equivalency

Minor Release Notes (R2019b, v7.2.0)

Added

- `requirements.txt` making dependencies on Matlab and specific toolboxes explicit
- `max_heart_rate_bpm` now a user parameter to adjust prior on max allowed heart rate for cardiac pulse detection (`method = 'auto_matched'`)
- bandpass-filtering of cardiac data during preprocessing now possible (`preproc.cardiac.filter`)
- Added integration tests for all examples in `tests/integration` for both SPM Batch Editor and Matlab-only configuration scripts. Reference data provided in `examples/TestReferenceResults/examples`

Changed

- Toned down and replaced irrelevant peak height and missing cardiac pulse warnings (github issue #51)
- Updated README to include external contributors and recent findings about impact of physiological noise for serial correlations (Bollmann2018)
- Added unit tests for convolution and moved all to `tests/unit`

Fixed

- Corrected half-width shift of response functions for HRV and RVT regressors by erroneous use of Matlab `conv`
  - For details on the bug, its impact and fix, see our specific Release Info on RVT/HRV Bugfix
  - other references: TNU gitlab issue #83, found and fixed by Sam Harrison, TNU, see `tapas_physio_conv`)
- Bugfix `tapas_physio_init()` not working, because dependent on functions in `utils` subfolder not in path; `utils` added to path
- `tapas_physio_review` for motion parameters (found and fixed by Sam Harrison, TNU)
- visualization error for regressor orthogonalization (github issue #57), when only `'RETROICOR'` set was chosen

Minor Release Notes (R2019a, v7.1.0)

Added

- Brain Imaging Data Structure (BIDS) reader and example for `*_physio.tsv[.gz]/.json` files
- Added BioPac txt-File read-in and example
- Template example with all physio-fields for matlab script and settings as in default SPM batch
- Started unit testing framework in folder `tests`
    - example functions for findpeaks and BIDS readin
    - reference data saved with example data in subfolder `TestReferenceResults`
    - reference data reflects physio structure after running example scripts with PhysIO R2019a

Changed

- put all functions in `code` into subfolders relating to different modules: `readin`, `sync`, `preproc`, `model`, `assess`, `utils` (gitlab-issue #58)
    - updated deployment `tapas_physio_init` because of that
    - updated figure names to reflect respective code module
- matlab-script examples now contain more comments
    - fixed internal bug that prepended absolute paths to input logfiles in automatic example generation
- `tapas_physio_create_noise_rois_regressors` with more flexible ROI reslicing options (speed-up) and uses `spm_erode` (no Matlab image processing toolbox needed), thanks to a contribution by Benoît Béranger
- introduced semantic version numbers for all previous releases, and changed Release numbering to R style
- extended documentation (FAQ, new read-in BIDS)
- several bugfixes (Sep 18 - Mar 19), see GitHub Issues

Removed

- `tapas_physio_findpeaks` now refers to current Matlab signal processing toolbox implementation, instead of copy of older version
- some Matlab toolbox dependencies by custom simplified functions (e.g., `suptitle`)

Bugfix Release Notes (R2018.1.3, v7.0.3)

Changed

- fixed bug for matching of Philips SCANPHYSLOG-files (Gitlab #62), if physlogs were acquired on different days, but similar times

Bugfix Release Notes (R2018.1.2, v7.0.2)

Added

- BioPac txt-file reader (for single file, resp/cardiac/trigger data in different columns)

Changed

- fixed bug for 3D nifti array read-in in tapas_physio_create_noise_rois_regressors (github issue #24, gitlab #52)

Bugfix Release Notes (R2018.1.1, v7.0.1)

Changed

- documentation.{html,pdf} export nicer with different FAQ numbering

Major Release Notes (R2018.1, v7.0.0)

Added

- initialization function `tapas_physio_init()` to check Matlab paths, including SPM for batch processing
- Extended motion diagnostics via Framewise displacement (Power et al., 2012)
  - Outlier motion models generate 'spike' regressors from FD outliers (gitlab issue #)
- Censoring of intervals with bad physiological recordings in RETROICOR regressors (github issue #11, gitlab #36)
- Added examples of Siemens VD (Tics Format, Prisma) and Human Connectome Project (HCP) format

Changed

- Updated read-in examples of all vendors (Siemens, Philips, GE) to latest PhysIO Toolbox version.
- Updated `README.md` to reflect changes to example download, new references
- Extended Wiki documentation, in particular examples and read-in formats

Minor Release Notes (R2017.3, v6.3.0)

- Included references to external ETH gitlab physio-doc repo and wiki
- New Human Connectome Project reader for preprocessed Siemens 3-column logfiles (`*Physio_log.txt`)
- Updated Siemens Reader for Multiband patches(CMRR), versions EJA_1
  - including multi-echo data (4,5 columns)
  - multi-channel ECG data
  - significant speed up of read-in
  - generalized framework for later changes to format
  - interpolation of different sampling rates RESP/CARDIAC
- updated README about documentation, new support policy and TAPAS on GitHub
- extended FAQ

Minor Release Notes (R2017.2, v6.2.0)

- Included Markdown-based documentation via Wiki (also CITATION, LICENSE, CHANGELOG.md)
- Included FAQ in Wiki
- Split git repositories into public, dev, examples, and added wiki, to disentangle development from deployed toolbox code and data
- Bugfix and Typo correction
- Philips SCANPYHSLOG for their software release 5.1.7.

Minor Release Notes (R2017.1, v6.1.0)

- Substantially improved Siemens interface, both for VB/VD and 3T/7T releases
  - several bugfixes
  - based on extensive user feedback from Berlin and Brisbane
- New functionality tapas_physio_overlay_contrasts.m to display non-physio contrasts automatically as well

Major Release Notes (r904 / R2016.1, v6.0.0)

- Software version for accepted PhysIO Toolbox Paper: doi:10.1016/j.jneumeth.2016.10.019
- Tested and expanded versions of examples

- Improved stability by bugfixes and compatibility to Matlab R2016
- Slice-wise regressor creation
- Detection of constant physiological time series (detachment, clipping)
- Refactoring of report_contrasts and compute_tsnr_gains as standalone functionality
- Improved Read-in capabilities (Siemens respiration data, BioPac .mat)
- Migration from svn (r904) to git (tnurepository) for version control

Major Release Notes (r835, v5.0.0)

- Software version for Toolbox Paper submission
- Noise ROIs modeling
- Extended motion models (24 parameters, Volterra expansion)
- HRV/RVT models with optional multiple delay regressors
- Report_contrasts with automatic contrast generation for all regressor groups
- compute_tsnr_gains for individual physiological regressor groups
- consistent module naming (scan_timing, preproc)
- Visualisation improvement (color schemes, legends)

Minor Release Notes (r666, v4.1.0)

- Compatibility tested for SPM12, small bugfixes Batch Dependencies
- Cleaner Batch Interface with grouped sub-menus (cfg_choice)
- new model: 'none' to just read out physiological raw data and preprocess, without noise modelling
- Philips: Scan-timing via gradient log now automatized (gradient_log_auto)
- Siemens: Tics-Logfile read-in (proprietary, needs Siemens-agreement)
- All peak detections (cardiac/respiratory) now via auto_matched algorithm
- Adapt plots/saving for Matlab R2014b

Major Release Notes (r534, v4.0.0)

- Read-in of Siemens plain text log files; new example dataset for Siemens
- Speed up and debugging of auto-detection method for noisy cardiac data => new method thresh.cardiac.initial_cpulse_select.method = 'auto_matched'
- Error handling for temporary breathing belt failures (Eduardo Aponte, TNU Zurich)
- slice-wise regressors can be created by setting sqpar.onset_slice to a index vector of slices

Major Release Notes (r497, v3.0.0)

- SPM matlabbatch GUI implemented (Call via Batch -> SPM -> Tools -> TAPAS PhysIO Toolbox)
- improved, automatic heartbeat detection for noisy ECG now standard for ECG and Pulse oximetry (courtesy of Steffen Bollmann)
- QuickStart-Manual and PhysIO-Background presentation expanded/updated
- job .m/.mat-files created for all example datasets

- bugfixes cpulse-initial-select method-handling (auto/manual/load)

Major Release Notes (r429, v2.0.0)

- Cardiac and Respiratory response function regressors integrated in workflow (heart rate and breathing volume computation)
- Handling of Cardiac and Respiratory Logfiles only
- expanded documentation (Quickstart.pdf and Handbook.pdf)
- read-in of custom log files, e.g. for BrainVoyager peripheral data
- more informative plots and commenting (especially in tapas_physio_new).

Minor Release Notes (r354, v1.1.0)

- computation of heart and breathing rate in Philips/PPU/main_PPU.m
- prefix of functions with tapas_*

Major Release Notes (r241, v1.0.0)

- complete modularization of reading/preprocessing/regressor creation for peripheral physiological data
- manual selection of missed heartbeats in ECG/pulse oximetry (courtesy of Jakob Heinzle)
- support for logfiles from GE scanners (courtesy of Steffen Bollmann, KiSpi Zuerich)
- improved detection of pulse oximetry peaks (courtesy of Steffen Bollmann)
- improved documentation
- consistent function names (prefixed by "physio_")

NOTE: Your main_ECG/PPU.m etc. scripts from previous versions (<=r159) will not work with this one any more. Please adapt one of the example scripts for your needs (~5 min of work). The main benefit of this version is a complete new variable structure that is more sustainable and makes the code more readable.

License

```
                GNU GENERAL PUBLIC LICENSE
                   Version 3, 29 June 2007
```



```
                        Preamble
```

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights.

Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

```
                    TERMS AND CONDITIONS
```

0. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section

7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

    A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

    Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

    In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

    If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not

available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS

IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

```
            END OF TERMS AND CONDITIONS

   How to Apply These Terms to Your New Programs
```

    If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

    To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.> Copyright (C)

    This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

    This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

    You should have received a copy of the GNU General Public License along with this program. If not, see http://www.gnu.org/licenses/.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and` show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read

http://www.gnu.org/philosophy/why-not-lgpl.html.