1 Supplement

1.1 Checklist Items

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] **Sec. 7**
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] **Supplement**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] **Code, instructions, and MSR data are included in the supplement**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] **Sec: 4, 6 and supplement**
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] **Supplement**
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] **Sec. 6**
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] **Supplement**
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] **Supplement**
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

1.1.1 Impact Considerations

Since an inherent goal in our method is to identify and characterize human activity, there are privacy considerations could be brought up. These concerns apply to both the potential for unwanted monitoring of activity, as well as potential exposure in identifying information with regard to the nature of, and sequence of tasks performed.

1.1.2 Data Acquisition

The Microsoft Research Human Activity (MSR) dataset[1] (Morris et al., 2014) was obtained under a CDLA-Permissive license. The time series included in this submission have been truncated to the first 5 minutes of activity.

[1] `https://msropendata.com/datasets/799c1167-2c8f-44c4-929c-227bf04e2b9a`

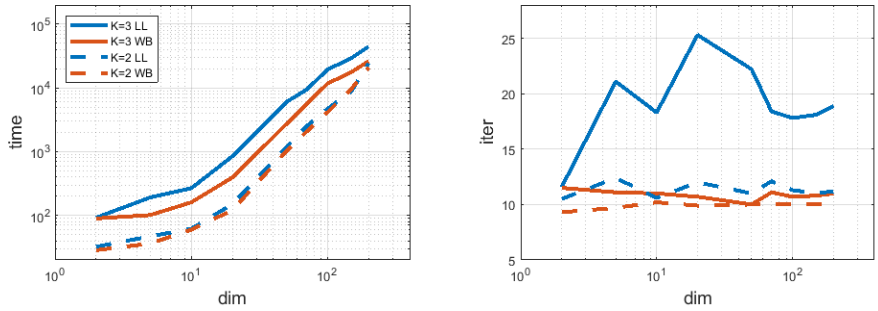

Figure 1: (left) Total time required for convergence optimizing simulated data outlined in Sec. 1.2 across data dimensions ranging from 2 to 200. Optimization with respect to Wasserstein-Bures geometry (red) for PSD matrices converges much faster than using Euclidean geometry (blue) parameterized by the Cholesky (LL) decomposition for both 2-state (dotted), and 3-state (solid) problems. (right) total number of line-search iterations required to reach convergence. Both methods converge to similar results.

The Beep Test (BT) dataset is a proprietary dataset and thus is not included in this submission and is unfortunately not available to share with the public. It was collected under approval by an affiliated institution's Institutional Review Board (IRB). All data has been deidentified before it was shared with study authors. All study authors are approved by their institutional IRB to use this deidentified wearable sensor data for research purposes.

1.1.3 Compute Time and Resources

Results were obtained using 4 CPU cores with 16 GB RAM, replicated for each time series on an internal cluster using slurm. On average convergence was reached in 6 hours.

1.2 Optimization Simulations

Algorithm 1: Riemannian Manifold Line Search (Absil et al., 2008)

Given : $(\mathcal{M}, g_p^{\mathcal{M}}(\cdot,\cdot))$: Riemannian manifold
$f(p)$: Smooth scalar function on the manifold
$\exp_p^{\mathcal{M}}$: Exponential map
α_0: Initial step
β: Sufficient decrease
c: Contraction factor
η: Convergence threshold

while $f\left(p^{(n)}\right) - f\left(p^{(n+1)}\right) > \eta$ **do**
  $\alpha = \alpha_0$
  while
    $\left(f\left(p^{(n)}\right) - f\left(exp_{p^{(n)}}^{\mathcal{M}}\left(\alpha\, gradf\left(p^{(n)}\right)\right)\right)\right) > \alpha\beta g_{p^{(n)}}^{\mathcal{M}}\left(gradf\left(p^{(n)}\right), gradf\left(p^{(n)}\right)\right)$ **do**
      $\alpha = c\alpha$
  end
  $p^{(n+1)} = \exp_{p^{(n)}}^{\mathcal{M}}\left(\alpha\, \mathrm{grad} f\left(p^{(n)}\right)\right)$
end

To motivate the choice of the Wasserstein-Riemannian manifold for optimization, we compare the optimization speeds in terms of wall-clock time and number of line search iterations with the more standard Euclidean geometry for covariance matrices (Arsigny et al., 2007) parameterized by the Cholesky decomposition. We evaluate performance on a set of simulated time series following our data model similar to that shown in Fig. 1 from the main paper where the system state linearly interpolates between two pure states over the course of 100 time steps. We vary the dimensionality of

the data ranging from $d = 2$ to 200, and number of states for $K = 2, 3$, repeating the experiment 10 times for each case.

To construct the data, we start by creating a random Gaussian distribution by generating a random mean vector $\boldsymbol{m}_1 \sim \mathcal{N}(0, \boldsymbol{I})$, and a random PSD matrix $\boldsymbol{S}_1$ generated by the method described in (Davies and Higham, 2000) with eigenvalues given by $\Lambda = \text{diag}([\lambda, ...\lambda_d]$ where $\lambda_i \sim U[0.5, 1.5]$. A second Gaussian distribution is generated a set distance away such that $\mathcal{W}_2^2((\boldsymbol{m}_1, \boldsymbol{S}_1), (\boldsymbol{m}_2, \boldsymbol{S}_2)) = 5$ where $\mathcal{E}^2(\boldsymbol{m}_1, \boldsymbol{m}_2) = 1$, and $\mathcal{B}^2(\boldsymbol{S}_1, \boldsymbol{S}_2) = 4$. This is achieved by generating a random vector in the tangent space and traveling the specified distance, along the geodesic on the manifold in the tangent direction. Specifically, for a random tangent vector to the mean $\boldsymbol{v} \in R^d$, we set $\boldsymbol{m}_2 = \boldsymbol{m}_1 + \frac{1}{\|\boldsymbol{v}\|_2}\boldsymbol{v}$, and for a random symmetric matrix $\boldsymbol{V}$, $\boldsymbol{S}_2 = \exp_{\boldsymbol{S}_1}^{\mathcal{B}}\left(\frac{2}{g_{\boldsymbol{S}_1}^{\mathcal{B}}(\boldsymbol{V},\boldsymbol{V})}\boldsymbol{V}\right)$, where $\exp_{\boldsymbol{S}}^{\mathcal{B}}$ and $g_{\boldsymbol{S}}^{\mathcal{B}}$ are the corresponding exponential map and Riemannian metric on the Wasserstein-Bures Manifold (Takatsu, 2011). For the three-state example, this process is repeated to generate a third Gaussian distribution such that $\mathcal{W}_2^2((\boldsymbol{m}_2, \boldsymbol{S}_2), (\boldsymbol{m}_3, \boldsymbol{S}_3)) = 5$.

For the simulated data, the state interpolates between $\boldsymbol{e}_1$ to $\boldsymbol{e}_2$ over 100 equi-spaced steps. The three-state problem, continues on and interpolates from $\boldsymbol{e}_2$ to $\boldsymbol{e}_3$ over an additional 100 steps. The empirical Gaussian distributions at each of these intermediate points is generated from $20d$ samples drawn from ρ_{B_t} according to the Wasserstein barycentric model described in the main paper. Since this optimization choice only pertains to $\boldsymbol{\Theta}$, for the purposes of this experiment we fix $\boldsymbol{X}$ as the ground truth thus eliminating the need to estimate $\boldsymbol{\Gamma}, \boldsymbol{H}$.

Fig. 1 shows that estimating the for $\boldsymbol{\Theta}$ using Wasserstein Riemannian geometry for Gaussians has improved performance in terms of converge speed in terms of both wall-clock and number of line-search iterations needed to converge. We found no significant difference in the solutions to which the two methods converged.

The python files used to generate the simulated data as well as run the simulated experiments are included in the supplemental code with instructions provided in Sec. 1.5.

1.3 Parameter Initialization and Optimization Parameters

Model Params	Initialization	Constraint	Description
$\boldsymbol{\gamma}_t[k]$	$1e-6$	$[0, 1]$	State innovation
$\boldsymbol{x}_0[k]$	$\frac{1}{K}$	$\sum_k \boldsymbol{x}_0[k] = 1$	Initial state
$(\boldsymbol{a}_1[k], \boldsymbol{b}_1[k])$	$(10, 20)$	$\boldsymbol{a}_1[k] > 1.1$ $\frac{\boldsymbol{a}_1[k]}{\boldsymbol{a}_1[k]+\boldsymbol{b}_1[k]} > 0.15$	Beta prior for transition dynamics
w	0.5	[0.01, 0.99]	Weight for Beta mixture prior
$(\mu_k, \boldsymbol{S}_k)$	Time series clustering given (Cheng et al., 2020b)	$\mu_k \in \mathbb{R}^d$ $\boldsymbol{S}_k \in Sym_+^d$	Pure state distribution

Table 1: Initialization and constraints of learned model parameters. Time index, $t = 1, ..., T$ and pure-state index $k = 1, ..., K$

The initialization and constraints for the learned model parameters are provided in Table 1. Other fixed parameters for the algorithm are included in Table 2.

For the optimization parameters, we use a learning rate of $2e-3$ for the ADAM optimization (Kingma and Ba, 2017) of $\boldsymbol{\Gamma}, \boldsymbol{H}$ and a convergence criterion of $\eta = 0.05$.

The Riemannian line search used to estimate $\boldsymbol{\Theta}$ is given in Alg. 1. Our implementation sets $\alpha_0 = 1e-1$, $\beta = 1e-10$ $c = 0.5$, $\eta = 0.05$

1.4 Additional Results

1.4.1 MSR Results with Ground Truth Initialization

Model Params	Value	Description
$\boldsymbol{m}_0$	$\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{y}_t$	Used as reference distribution mean for $\boldsymbol{\Theta}$
σ_0	average eigenvalue of covariance matrices of K-component GMM fit to $\boldsymbol{y}_t$ (Pedregosa et al., 2011)	$\sigma_0\mathbf{I}$ used as reference distribution covariance for $\boldsymbol{\Theta}$
η	0.05	Convergence threshold
$(\boldsymbol{a}_0, \boldsymbol{b}_0)$	(1.1, 20)	Beta prior for stationary dynamics
	10	# fixed point iterations for covariance Wasserstein barycenter computation (Álvarez Esteban et al., 2016)
	5000	# Monte Carlo samples used to compute Wasserstein distance between GMM and Gaussian (Sriperumbudur et al., 2010)

Table 2: Value of fixed algorithm parameters.

In order to stay true to the unsupervised nature of our problem, in our real-world experiments, we initialize the pure-state Gaussian model parameters using the time-series segmentation algorithm using the unsupervised methods described in (Cheng et al., 2020a). Since our problem is non-convex, poor initialization could lead to local minimum. To ensure that our reported results are not a result of biased initialization, we also run the same experiments initializing $\boldsymbol{\Theta}$ according to the sample mean and covariance matrices of each activity given the ground truth discrete labels of the MSR data. As shown in Fig. 2, the results given by this ground truth initialization do not deviate much from Fig. 6 from the main paper where the average absolute difference between the two initialization methods for $e_W(DWB)$ and $e_W(GMM)$ are 0.011 and 0.022 respectively.

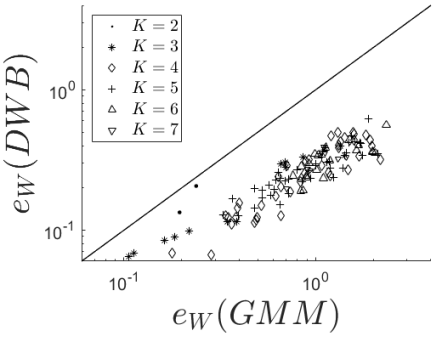

Figure 2: Evaluation comparison between the GMM and Wasserstein barycenter model for the MSR data when using the ground truth discrete labels to initialize the pure-state distribution parameters. Results shown here are close to that shown in the main paper using an unsupervised approach for parameter initialization.

1.4.2 Innovation Prior Ablation Study

Here Fig. 4 shows the comparison when using fixed $\boldsymbol{a}, \boldsymbol{b}$ parameters for a single-modal Beta distribution for the prior on $\boldsymbol{\gamma}_t$ versus using the two-component Beta mixture with learnable $\boldsymbol{a}, \boldsymbol{b}, w$ parameters for the transition component as specified in the main paper in Sec. 4.1.

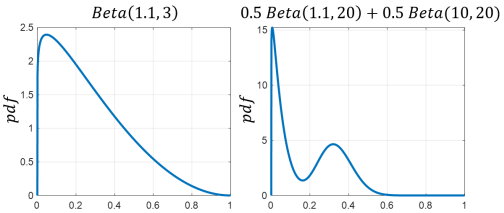

Figure 3: Sample pdfs of single a single component and two-component Beta mixture. Beta distributions are defined on the domain $[0, 1]$ and are uni-modal for parameters $a, b > 1$, The Beta mixture allows us to separately model the stationary and transition dynamics.

When using a fixed single component Beta prior, there is a single mode to learn the system dynamics in both transition regions (where one expects more rapid changes in state) and stationary regions (where changes are much smaller). Because of this compromise, the transitions between states are more sluggish while compared to the two-component Beta model. The learned pure-state distribution parameters do not differ much but the overall model fit is improved in the two-component model due to better tracking in the transition regions. For the two-component Beta prior, the average across all MSR datasets is $e_W = 0.50$ compared to the average $e_W = 0.27$ for the single-component fixed Beta model.

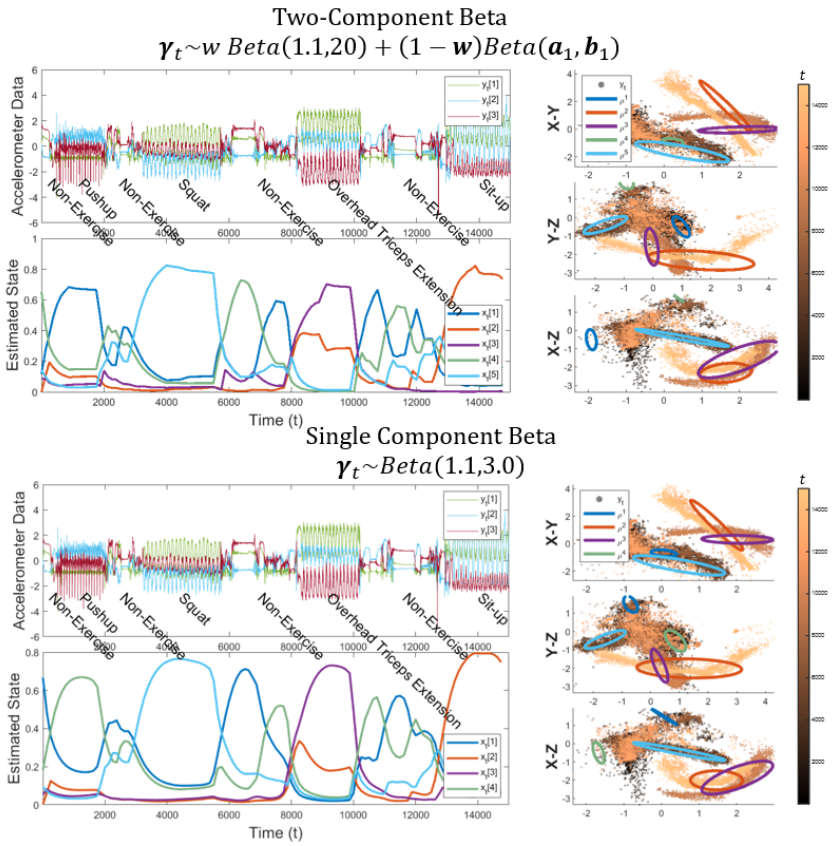

Figure 4: Comparison of model output using a single-component fixed prior versus a two-component learnable prior for the innovations γ_t. While the pure state results in the right column are comparable between the two approaches, the state-variable is more sluggish in adapting to faster transitions in the single-component model.

1.4.3 Deep State Space Benchmark

We emphasize that in addition to the characterization of the underlying distribution shown in the above evaluation, our DWB approach identifies a set of discrete pure states in the system and yields a directly-interpretable per-sample state representation; the Wasserstein barycentric mixing weights for each pure-state is directly given by the corresponding component of the simplex-state vector. In contrast, there is no direct interpretation of the latent state and no equivalent comparison to the learned pure-state in the DSS model. Additional ad hoc processing (e.g. clustering) of the DSS latent state space would be required to achieve this, which is beyond the scope of our present paper. Tab. 3 highlights additional similarities and differences between the two models.

As mentioned, we run the DSS with two parameter settings. The first uses 2 hidden layers each with 5 neurons for a total of 94 learned parameters for each transmission and emission networks. The second uses the default parameters included with the code[2] which contains 3 hidden layers for the transition and emission networks with a hidden state dimension of 200 for a total of for each neural network. In both cases, the latent space has dimension (K-1) (to match the dimensionality of the K-simplex), an RNN of 2 layers and 600 nodes each is used as the variational approximation network, and training occurs over 1000 epochs with a learning rate of 0.008. Plots of the variational lower bound show convergence under these conditions

[2] `https://github.com/clinicalml/dmm`

	DWB	DSS
State Space	K-simplex	R^n
State Transition Dynamics	Beta mixture	Gaussian distribution
State Transition Parameters	Learnable Beta parameters	Neural network parameterizing mean and diagonal covariance of Gaussian
Emission distribution model	Gaussian with full covariance	Gaussian with diagonal covariance
Number of learned parameters	$O(d^2K)$ d =data dimension $K = \#$ clusters	$O(m^2)$ m =NN hidden layer size

Table 3: Comparison between DWB and DSS models

1.4.4 Additional Results for MSR Data

We provide the results for learned model parameters for the entire MSR dataset. All plots follow the same format as Fig. 4 from the main paper, where the raw accelerometer data is included in the top right, the learned state in the bottom left, and the scatter plot of the time series overlayed with the estimated pure states projected onto each of the pair-wise axes in the right column.

The plots produced through initialization using change point detection and time series clustering (CPD) discussed in (Cheng et al., 2020a) are in the "MSR_Plots" folder, and those produced through ground truth discrete labels (GT) are in "MSR_GT_Plots".

1.5 Code Instructions

1.5.1 MSR

To run the code [3] and replicate the results reported in our paper,

```
# usage: DynamicalWassersteinBarycenters.py dataSet dataFile debugFolder
↪  interpModel [--ParamTest PARAMTEST] [--lambda LAM] [--s S]

# Sample run on MSR data
>> python DynamicalWassersteinBarycenters.py MSR_Batch
↪  ../Data/MSR_Data/subj090_1.mat ../debug/MSR/subj001_1.mat Wass

# Sample run for parameter test
>> python DynamicalWassersteinBarycenters.py MSR_Batch
↪  ../Data/MSR_Data/subj090_1.mat ../debug/ParamTest/subj001_1.mat Wass
↪  --ParamTest 1 --lambda 100 --s 1.0
```

The "interpMethod" is either "Wass" for the Wasserstein barycentric model or "GMM" for the linear interpolation model.

1.5.2 Optimization Simulations

The simulated data and experiment included in this supplement can be replicated using using the following commands.

```
# Generate 2 and 3 state simulated data
>> python GenerateOptimizationExperimentData.py
>> python GenerateOptimizationExperimentData_3K.py

# usage: OptimizationExperiment.py FileIn Mode File
```

[3] Code available at `https://github.com/kevin-c-cheng/DynamicalWassBarycenters_Gaussian`

```
# Sample run for optimization experiment
>> python OptimizationExperiment.py
↪  ../data/SimulatedOptimizationData_2K/dim_5_5.mat/ WB
↪  ../debug/SimulatedData/dim_5_5_out.mat
```

The "Mode" is either "WB" for Wasserstein-Bures geometry and "Euc" for Euclidean geometry using Cholesky decomposition parameterization.

1.5.3 Computation Environment

We use Python 3.8 with the package requirements included in "requirements.txt" and copied below.

```
_libgcc_mutex=0.1=conda_forge
_openmp_mutex=4.5=1_llvm
_pytorch_select=0.2=gpu_0
blas=2.17=openblas
ca-certificates=2020.12.5=ha878542_0
certifi=2020.12.5=py38h578d9bd_1
cffi=1.14.4=py38h261ae71_0
cudatoolkit=8.0=3
cudnn=7.1.3=cuda8.0_0
cycler=0.10.0=py_2
freetype=2.10.4=h7ca028e_0
future=0.18.2=py38h578d9bd_3
immutables=0.15=py38h497a2fe_0
intel-openmp=2020.2=254
joblib=1.0.0=pyhd8ed1ab_0
jpeg=9d=h36c2ea0_0
kiwisolver=1.3.1=py38h82cb98a_0
lcms2=2.11=hcbb858e_1
ld_impl_linux-64=2.33.1=h53a641e_7
libblas=3.8.0=17_openblas
libcblas=3.8.0=17_openblas
libedit=3.1.20191231=h14c3975_1
libffi=3.3=he6710b0_2
libgcc-ng=9.3.0=h5dbcf3e_17
libgfortran-ng=7.3.0=hdf63c60_0
libgomp=9.3.0=h5dbcf3e_17
liblapack=3.8.0=17_openblas
liblapacke=3.8.0=17_openblas
libopenblas=0.3.10=pthreads_hb3c22a3_4
libpng=1.6.37=h21135ba_2
libstdcxx-ng=9.3.0=h6de172a_18
libtiff=4.1.0=h4f3a223_6
libwebp-base=1.1.0=h36c2ea0_3
llvm-openmp=11.0.0=hfc4b9b4_1
lz4-c=1.9.2=he1b5a44_3
matplotlib-base=3.3.3=py38h5c7f4ab_0
mkl=2020.4=h726a3e6_304
mkl-service=2.3.0=py38he904b0f_0
mkl_fft=1.3.0=py38h5c078b8_1
mkl_random=1.2.0=py38hc5bc63f_1
ncurses=6.2=he6710b0_1
ninja=1.10.2=py38hff7bd54_0
numpy=1.19.5=py38h18fd61f_1
numpy-base=1.18.5=py38h2f8d375_0
olefile=0.46=pyh9f0ad1d_1
openssl=1.1.1k=h7f98852_0
pillow=8.1.0=py38h357d4e7_1
pip=20.3.3=py38h06a4308_0
```

```
pot=0.7.0=py38h950e882_0
pycparser=2.20=py_2
pyparsing=2.4.7=pyh9f0ad1d_0
python=3.8.5=h7579374_1
python-dateutil=2.8.1=py_0
python_abi=3.8=1_cp38
pytorch=1.7.1=cpu_py38h36eccb8_1
readline=8.0=h7b6447c_0
scikit-learn=0.24.1=py38h658cfdd_0
scipy=1.5.2=py38h8c5af15_0
setuptools=51.1.2=py38h06a4308_4
six=1.15.0=py38h06a4308_0
sqlite=3.33.0=h62c20be_0
threadpoolctl=2.1.0=pyh5ca1d4c_0
tk=8.6.10=hbc83047_0
tornado=6.1=py38h497a2fe_1
wheel=0.36.2=pyhd3eb1b0_0
xz=5.2.5=h7b6447c_0
zlib=1.2.11=h7b6447c_3
zstd=1.4.5=h6597ccf_2
```

References

P.-A. Absil, R. Mahony, and R. Sepulchre. 2008. *Optimization algorithms on matrix manifolds*. Princeton University Press, Princeton, N.J. ; Woodstock. OCLC: ocn174129993.

Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. 2007. Geometric Means in a Novel Vector Space Structure on Symmetric Positive-Definite Matrices. *SIAM J. Matrix Anal. Appl.* 29, 1 (Jan. 2007), 328–347. https://doi.org/10.1137/050637996

Kevin C. Cheng, Shuchin Aeron, Michael C. Hughes, Erika Hussey, and Eric L. Miller. 2020a. Optimal Transport Based Change Point Detection and Time Series Segment Clustering. *arXiv:1911.01325 [cs, eess]* (Feb. 2020). http://arxiv.org/abs/1911.01325 arXiv: 1911.01325.

Kevin C. Cheng, Eric L. Miller, Michael C. Hughes, and Shuchin Aeron. 2020b. On Matched Filtering for Statistical Change Point Detection. *IEEE Open Journal of Signal Processing* 1 (2020), 159–176. https://doi.org/10.1109/OJSP.2020.3035070 Conference Name: IEEE Open Journal of Signal Processing.

Philip I. Davies and Nicholas J. Higham. 2000. Numerically Stable Generation of Correlation Matrices and Their Factors. *BIT Numerical Mathematics* 40, 4 (Dec. 2000), 640–651. https://doi.org/10.1023/A:1022384216930

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]* (Jan. 2017). http://arxiv.org/abs/1412.6980 arXiv: 1412.6980.

Dan Morris, T. Scott Saponas, Andrew Guillory, and Ilya Kelner. 2014. RecoFit: using a wearable sensor to find, recognize, and count repetitive exercises. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Toronto Ontario Canada, 3225–3234. https://doi.org/10.1145/2556288.2557116

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, 85 (2011), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Bharath K. Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert R. G. Lanckriet. 2010. Hilbert Space Embeddings and Metrics on Probability Measures. *Journal of Machine Learning Research* 11, 50 (2010), 1517–1561. http://jmlr.org/papers/v11/sriperumbudur10a.html

Asuka Takatsu. 2011. Wasserstein geometry of Gaussian measures. *Osaka Journal of Mathematics* 48, 4 (Dec. 2011), 1005–1026. https://doi.org/10.18910/4973 Publisher: Osaka University and Osaka City University, Departments of Mathematics.

Pedro C. Álvarez Esteban, E. del Barrio, J. A. Cuesta-Albertos, and C. Matrán. 2016. A fixed-point approach to barycenters in Wasserstein space. *arXiv:1511.05355 [math, stat]* (April 2016). http://arxiv.org/abs/1511.05355 arXiv: 1511.05355.