**Material and Methods**

We have analysed more than 35000 time series of taxa from the gut microbiome of 97 individuals (sampling from three up to 332 time points), obtained from publicly available high throughput sequencing data on: healthy individuals over a long term span1, people with various degrees of obesity2, twin pairs discordant for kwashiorkor3, response to diet changes4 or to antibiotic perturbation5, and subjects diagnosed with irritable bowel syndrome (IBS)6. We engineered a complete software framework and a web platform, ComplexCruncher, ready to be implemented by other users. We summarize our dataset in ST1-6 (Tables 1-6 in supplemental material). The bacteria and archaea taxonomic assignations were obtained by analyzing 16S rRNA sequences, which were clustered into operational taxonomic units (OTUs) sharing 97 % sequence identity using QIIME7. WGS data3 were analyzed and assigned at strain level by the Livermore Metagenomic Analysis Toolkit (LMAT) 8, according to their default quality threshold. Genus, with best balance between error assignment and number of taxa, was chosen as our reference taxonomic level. We have verified that our conclusions are not significantly affected by selecting family or species as the reference taxonomic level (see Figure 1 in supplemental material).

### specify, in each study treated, the nature of the samples (conditions, timespan between timepoints, subjects). Specify, and it is very important, what we consider ‘healthy’ in each study (for example: pre-antibiotics is healthy)

**1. Model**

We model the microbial abundances across time along the lines of Blumm et al. (1). The dynamics of taxon relative abundances is described by the Langevin equation:

x ̇i=Fi·xαi +V·xβiξi(t)−φ(t)·xi, (1)

where Fi captures the fitness of the taxon i, V corresponds to the noise amplitude and ξi(t) is a Gaussian random noise with zero mean < ξi(t) > = 0 and variance uncorrelated in time, < ξi(t)ξi(t′) > = δ(t′ −t), . The function φ(t) ensures the normalization at all times, xi(t)=1,andcorrespondstoφ(t)= Fixαi + Vxβiξi(t). Thetemporalevolution of the probability that a taxon i has a relative abundance xi(t), P(xi,t), is determined by the Fokker-Planck equation

∂t =−∂x[(Fi·xi −φ(t)·xi)·P]+2∂x2(V ·xi ·P). (2) ii

The microbiota evolves towards a steady-state with a time-independent probability de- pending on the values of α, β, Fi and V. For α < 1 (otherwise, systems are always unsta- ble), the steady-state probability may be localized in a region around a preferred value or broadly distributed over a wide range, depending on whether the fitness Fi dominates or is overwhelmed by the noise amplitude V. The steady-state solution of the Fokker-Planck equation is given by:

equations

equations where φ0 = ( F1/(1−α))1−α and Cne and Ce are integrals that should be solved numerically for the parameters of interest. The ordered phase happens when the solution has a maximum in the physical interval (0 < xi < 1). For larger V, the transition to a dis- ordered phase happens when the maximum shifts to the unphysical region xi < 0, which sets the phase transition region V(α, β, Fi). The phase transition region can be calculated analytically in particular cases:

equation

where the first case, simplifies to F = 3V 2 if β = 0.75 and the fitness of this taxon dominates in φ0. In many physical systems (Brownian motion is the classical example), the two terms of the Langevin equation are related. The fluctuation–dissipation theorem states a general relationship between the response to an external disturbance and the internal fluctuations of the system (2). The theorem can be used as the basic formula to derive the fitness from the analysis of fluctuations of the microbiota, assuming that it is in equilibrium (the ordered phase).

### explain better the fluctuation-dissipation theorem

**2.1 Sample selection**

We have chosen studies about relevant pathologies containing metagenomic sequencing time data series of bacterial populations from humans in different healthy and non-healthy states. We have selected only those individuals who had three or more time points of data available in databases. Metadata of each study is provided in Supplementary Tables 1 to 6. All used 16S rRNA gene sequencing except for the study of the discordant kwashiorkor twins (3) (see Supplementary Tables 4 and 5) where shotgun metagenomic sequencing (SMS) and 16S rRNA were used. In the latter case we selected to work with SMS data to show that our method is valid regardless of the source of taxonomic information. Each one of the datasets was treated as follows:

**2.2 16rRNA sequences processing**

Reads from the selected studies were first quality filtered using the FastX toolkit (4), allowing only those reads which had more than 25 of quality along the 75% of the complete sequence. 16S rRNA reads were then clustered at 97% nucleotide sequence identity (97% ID) into operational taxonomic units (OTUs) using QIIME package software (5) (version 1.8.0) We followed open reference OTU picking workflow in all cases. The clustering method used was uclust, and the OTUs were matched against Silva database (6) (version 111, July 2012) and were assigned to taxonomy with an uclust-based consensus taxonomy assigner. The parameters used in this step were: similarity 0.97, prefilter percent id 0.6, max accepts 20, max rejects 500.

**2.3 Metagenomic sequences processing**

Metagenomic shotgun (and 16S too) sequences were analyzed with LMAT (Livermore Metagenomics Analysis Toolkit) software package (7) (version 1.2.4, with Feb’15 release of data base LMAT-Grand). LMAT was run using a Bull shared-memory node belonging to the team’s HPC (high performance computing) cluster. It is equipped with 32 cores (64 threads available using Intel Hyper-threading technology) as it has 2 Haswell-based Xeons, the E5-2698v3@2.3 GHz, sharing half a tebibyte (0.5 TiB, that is, 512 gibibytes) of DRAM memory. This node is also provided with a card PCIe SSD as NVRAM, the P420m HHHL, with 1.4 TB, and 750000 reading IOPS, 4 KB, achieving 3.3 GB/s, which Micron kindly issued free of charge, as a sample for testing purposes. The computing node was supplied with a RAID-0 (striping) scratch disk area. We used the “Grand” databaseI, release Feb’15, provided by the LMAT team. Previously to any calculation, the full database was loaded in the NVRAM. With this configuration the observed LMAT sustained sequence classification rate was 20 kpb/s/core. Finally, it is worth mentioning that a complete set of Python scripts have been developed as back-end and front-end of the LMAT pipeline in order to manage the added complexity of time series analysis.

**2.4 Taxa level selection**

We selected genus as taxonomic level for the subsequent steps of our work. In order to ensure that, between adjacent taxonomic levels, there were not crucial differences which could still be of relevance after standardization (see Section 12), we tested two different data sets. In the former, the antibiotics study (8) with 16S data, we tested the differences between genus and family levels. The latter dataset tested was the kwashiorkor discordant twins study (3) for both genus and species taxonomic levels. The Supplementary Figures 1 (overview) and 2 (detail) plot the comparison between studies (and so, 16S and SMS) and between adjacent taxonomical levels.

**4 Engineered Software**

A complete software framework, named ’ComplexCruncher’, has been engineered to support the analysis of the dynamics of ranking processes in complex systems. Although the software was devised with a clear bias towards metagenomics, it is general enough to be able to cope with a ranking process in any complex system. Implemented in Python using well-known open-source community software, the software solution is composed of two parts that can be used together or apart: a web-based graphic front-end connected to a database, and a computing kernel. Used together, this software enables other users to reproduce our results easily and, furthermore, upload and analyse their own data or experiment with the preloaded metagenomics data sets. The next sections of this supplementary material deal with these two software pieces in detail.

**7 Un-weighted power-law fit**

Fitting the best model As mentioned above, to choose between fitting power laws (y = V xβ ) using linear regression on log-transformed (LLR) data versus non-linear regression (NLR), we mainly follow *General Guidelines for the Analysis of Biological Power Laws*13. It consists of the following three steps:

1. Determining the appropriate error structure by likelihood analysis.

(a) Fit the Non-Linear Regression (NLR) model and obtain V , β and σ2 . NLR NLR NLR

(b) Calculate the loglikelihood that the data (n is sample size) are generated from a normal distribution with additive error:

- The likelihood of a normal distribution is:

equation

* So, the loglikelihood of a normal distribution is:

Equation

(c) Calculate the *corrected Akaike’s Information Criterion* for the NLR model:

equation

(d)  Fit the Log-transformed Linear Regression (LLR) model and obtain VLLR, βLLR and σ2 .

(e)  Calculate the loglikelihood that the data (n is sample size) are generated from a log- normal distribution with multiplicative error:  • The likelihood of a lognormal distribution is:

equation

So, the loglikelihood of a lognormal distribution is:

Equation

(f) Calculate the *corrected Akaike’s Information Criterion* for the LR model:

2. CompareAICcNLR withAICcLLR:

* If AICcNLR − AICcLLR < −2, the assumption of normal error is favoured compared to  lognormal error, so proceed with the results obtained from the NLR fit.
* If AICcNLR − AICcLLR > 2, the assumption of lognormal error is favoured compared to normal error, so proceed with the results obtained from the LLR fit.
* • If |AICcNLR − AICcLLR | ≤ 2, no model is favoured, so proceed with model averaging:

equation

equation

where:

equation

which are obtained to fulfill the next condition: wNLR + wLLR = 1. The CIs for Bav and βav are to be generated by ordinary bootstrappingIV.

3. Assess the validity of the underlying statistical assumptions with diagnostic plots because while it is rare for all the assumptions to be fully satisfied by real-life data sets, major viola- tions indicate the lack of appropriateness of the model and, thus, the potential invalidity of the results.

Calculating the coefficient of determination We think the best approach in this situation is to apply the generalized R2 that, for continuous models, was defined as14:

Equation

where L(θ) and L(0) denote the likelihoods of the fitted and the “null” model, respectively, and

n is the sample size. In terms of the loglikelihoods, the generalized coefficient of determination would be:

equation

We have the likelihoods calculated from the previous section, but what about the “null” models? We understand that they are the models with only the intercept. So for the Gaussian additive error model:

Equation

So:

Equation

since σ2 we have:

equation

recovering the traditional expression for R2. Using the same approach for calculating R2LLR, then:

equation

So:

equationsince σLLR0 = n log |yi| − log |y| = n TSSlogn. Again, recalling the expression for the

generalized coefficient of determination, we have:

**8 X-weighted power-law fit**

When fitting the power-law of std vs. mean, we can take into account that every mean has uncer- tainty and estimate it for a sample size n by the SEM (*Standard Error of the Mean*):

equation

where s is the sample standard deviation. So, the vector of weights is computed with:

equation

Here, the uncertainties affect the independent variable, so the fit is not so trivial as a Y- weighted fit, where the uncertainties affect the dependent variable. A standard approach to do this fit is: a) invert your variables before applying the weights, b) then perform the weighted fit, and finally, c) revert the inversion. This method is deterministic, but the approximate solution worsens with smaller R2. For comparison, we develop a stochastic method by using a bootstrapping-like strategy that avoids the inversion and is applicable regardless of R2. Both methods, detailed below, are implemented in *cmplxcruncher*.

Method 1: By inverting the data In the case of the log-LR model, we have:

Equation

where m determines the slope or gradient of the fitted line, and b determines the point at which the

line crosses the y-axis, otherwise known as the y-intercept. Once the model is fitted, the original parameters can be retrieved easily:

equation

Their respective uncertainties are to be obtained using *error propagation*:

Equation

Method 2: Bootstrapping-like strategy The basic idea of bootstrapping is that inference about a population from sample data (sample → population) can be modeled by resampling the sample data and performing inference on (resample → sample). To adapt this general idea to our problem, we resample the x-data array using its errors array. That is, for each replicate, a new x-data array is computed based on:

equation

where vi is a Gaussian random variable with mean μi = 0 and standard deviation σi = SEMi, as defined previously in this supplementary material. For each replicate a complete un-weighted power-law fit is performed, as described in the previous section. It is worth mentioning that each replicate is filtered to avoid values of x∗i under *eps*(obtained bynp.finfo(np.double).eps) in order to keep away from the error of getting log of negatives or zero during the fit.

We devised and implemented a multi-step algorithm to estimate the fit parameters that fin- ishes when a relative error of less than 10−4 is achieved. It also ends if the number of steps reaches 100 to avoid too much time lapse, to prevent any pathologic numeric case which, in fact, we still have not detected in all the data sets analyzed.

In the previous version of the algorithm, for each step, the method generated 10 replicates for each x-data point, in other words, it was computing the fit for 10 times the length of the x-data array replicates, with a maximum of 10000 fits per step. Nevertheless, we found that such an approach depending on the length of the x-data array did not perform better, so we decided to simplify the method and fix the number of fits per step in 100. This latter approach improved the performance.

The parameters of the X-weighted fit are then estimated by averaging through all the replicate fits performed, and their errors are estimated by computing the standard deviation also for all the fits. At the end of each step, the relative error is calculated by comparing the fit parameters estimation in the last step with the previous one.

Finally, both the coefficient of determination of the fit and the coefficient of correlation be- tween the fit parameters are estimated by averaging.

**9 Rank Stability Index (RSI)**

The Rank Stability Index is shown as a percentage in a separate bar on the right of the rank matrix plot provided by *cmplxcruncher*. The RSI is strictly 1 for an element whose range never changes over time, and is strictly 0 for an element whose rank oscillates between the extremes from time to time. So, RSI is calculated, per element, as 1 less the quotient of the number of true rank hops taken between the number of maximum possible rank hops, all powered to p:

Equation

where D is the total of rank hops taken by the studied element, N is the number of elements that have been ranked, and t is the number of time samples. The power index p is arbitrarily chosen to increase the resolution in the stable region; the value in the current version of the code is p = 4.

As an example of this “zooming” effect in the stable region, to match a linear (p = 1) RSI of 0.9 to a powered one of 0.1, we should select p = 21.8543. An alternative way to obtain this effect and exactly map a linear RSI of 0.9 to a non-linear RSI (RSI′) of 0.1, is by applying the following function:

Equation

where the approximation is valid because 1010 ≫ 1 but, the small price to pay for it is that, in the

worst instability case, the RSI′ would not be strictly 0 but 10−10.

The colour code of the RSI percentage text in the rank plot of *cmplxcruncher* is chosen following the first condition satisfied from those shown in Table 8 (see page 33).

Equation

**12 Standardization**

In order to show all the studies properly under common axes, we decided to standardize the Taylor parameters using the group of healthy individuals for each study. With this approach, all the studies can be visualized in a shared plot with units of Taylor-parameters standard-deviation on their axes.

For a Taylor parameter, e.g. V , the estimate of the mean (V ) for the healthy subpopula- tion, composed of h individuals, is:

[equation]

as [equation], since ωi are normalized weights calculated as: 1

being σVi the estimation of the uncertainty in Vi obtained together with Vi from the X-weighted power-law fit described in Section 8, for healthy individuals.

Likewise, the estimation of the standard deviation for the healthy population (σ V ) is: [equation]

[equation]

being W2 = hiωi2, which finally yields to:

[equation]