MSG400-TMS150

Stochastic data processing and simulation 2025

The Bootstrap for simulation-based uncertainty quantification

Exercise 1 constitute the assignment A2. Notice the "recommended deadline" on the course webpage. The A2 assignment must be handed in as a LaTeX report and be submitted to Canvas together with the corresponding Matlab code.
Please use the recommended report template provided on the course page. You must also upload separately from the report all the code you produced. And you must also embed the code inside the report, as illustrated in the report template found on the course webpage. Recall that each submitted report and the code therein is INDIVIDUAL not group work.
Finally, since you have to write a proper report: as from the provided template, you are also asked to produce some background on the methodology you use. So do not just write answers to the exercise questions. See `https://chalmers.instructure.com/courses/31060/pages/guidelines-for-report-writing` for guidelines on report writing.

The report should not be longer than 10 pages including figures, but excluding appendices. Figures and axes labels should be big enough to be readable if printed. It is OK to use colors. For a given project report, 0.5 points will be deducted if the report is not clearly structured or is otherwise hard to understand. Likewise, 0.5 points will be deducted if the code attached to the report is not properly structured and commented.
Full details on grading are on the course webpage.

I know most of you have had some experience with Matlab, but regardless at the end of the document you find a short Matlab primer. Also see the file `demo_matlab.m` on Canvas.

Some of the sections that follow are denoted with an asterisk * when the topic can be skipped (those are left for the interested reader).

1 Introduction

In the previous two lectures we have focussed on the selection of appropriate linear regression models, their parameters estimation and the assessment of the estimates uncertainty. However, the estimation of parameters in general models and the assessment of the estimates' variability, is not a problem exclusively pertaining regression models (linear or nonlinear). In fact, the estimation of model parameters is a central problem in statistics that one tends to encounter already during the very first course on the subject. More generally, we are interested in inferences for *unknowns*: these are not just model parameters, for example when we predicted new

observations, clearly those are unobserved quantities. Along with the estimate of some unobserved quantity we are (we should be!) also interested in its accuracy, which can be described in terms of the bias and the variance of the estimator, as well as confidence intervals around it. Sometimes, such measures of accuracy can be derived analytically. Often, they can not. The *bootstrap* is a technique that can be used to estimate them numerically[1].

2 The general idea

Let $X_1, ..., X_n$ be a i.i.d. sample from distribution F, that is $\mathbf{P}(X_i \leq x) = F(x)$, and let $X_{(1)}, ..., X_{(n)}$ be the corresponding ordered sample. For the purpose of this introduction, suppose we are interested in some *scalar* parameter θ which is associated with this distribution (mean, median, variance etc), the treatment of a multivariate θ being analogous. There is also an estimator $\hat{\theta} = t(\{X_1, ..., X_n\})$, with t denoting some function, that we can use to estimate θ from data. In this setting, it is the deviation of $\hat{\theta}$ from θ that is most interesting.

Ideally, to get an approximation of the estimator distribution we would like to repeat the data-generating experiment, say, B times, calculating $\hat{\theta}$ for each of the B data sets. That is, we would draw B samples of size n from the true distribution F (with replacement, if F is discrete). In practice, this is impossible.

The bootstrap method is based on the following simple idea: *Even if we do not know F we can approximate it from data and use this approximation, $\hat{F}$, instead of F itself.*

This idea leads to several flavours of bootstrap that deviate in how, exactly, the approximation $\hat{F}$ is obtained. Two broad areas are the *parametric* and *non-parametric* bootstrap.

The non-parametric estimate is the so called empirical distribution (you will see the corresponding pdf if you simply do a histogram of the data), that can be formally defined as follows:

Definition 2.1. *With # denoting the number of members of a set, the* empirical distribution function $\hat{F}$ *is given by*

$$\hat{F}(x) = \frac{\#\{i : X_i \leq x\}}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\{X_i \leq x\},$$

$$\mathbb{I}\{X_i \leq x\} = \begin{cases} 1 & \text{if } X_i \leq x \\ 0 & \text{otherwise} \end{cases}$$

That is, it is a discrete distribution that puts mass $1/n$ on each data point in your sample.

The parametric estimate assumes that the data comes from a certain distribution family (Normal, Gamma etc). That is, we say that we know the general functional form of the pdf, but not the exact parameters. Those parameters can then be estimated from the data (typically with Maximum Likelihood) and plugged in the pdf to get $\hat{F}$. This estimation method leads to more accurate inference if we guessed the distribution family correctly but, on the other hand, $\hat{F}$ may be quite far from F if the family assumption is wrong.

[1] More generally, the bootstrap is a method of approximating the distribution of functions of the data (statistics), which can serve different purposes, among others the construction of CI and hypothesis testing.

2.1 Algorithms

The two algorithms below describe how the bootstrap can be implemented.

Non-parametric bootstrap
Assuming a data set $x = (x_1, ..., x_n)$ is available.

1. Fix the number of bootstrap re-samples B. Often $B \in [1000, 2000]$.
2. Sample a new data set x^* set of size n from x *with replacement* (this is equivalent to sampling from the empirical cdf $\hat{F}$).
3. Estimate θ from x^*. Call the estimate $\hat{\theta}_1^*$. Store.
4. Repeat step 2 and 3 further $B - 1$ times.
5. Consider the empirical distribution of $(\hat{\theta}_1^*, ..., \hat{\theta}_B^*)$ as an approximation of the true distribution of $\hat{\theta}$.

You may produce the histogram of $(\hat{\theta}_1^*, ..., \hat{\theta}_B^*)$. This histogram represents the pdf of $\hat{F}$.

Parametric bootstrap
Assuming a data set $x = (x_1, ..., x_n)$ is available.

1. Assume that the data comes from a known distribution family F_ψ described by a set of parameters ψ (for a Normal distribution $\psi = (\mu, \sigma)$ with μ being the expected value and σ the standard deviation).
2. Estimate ψ with, for example, Maximum likelihood, obtaining the estimate $\hat{\psi}$.
3. Fix the number of bootstrap samples B. Often $B \in [1000, 2000]$.
4. Sample a new data set x^* set of size n from $F_{\hat{\psi}}$.
5. Estimate θ from x^*. Call the estimate $\hat{\theta}_1^*$. Store.
6. Repeat 4 and 5 further $B - 1$ times.
7. Consider the empirical distribution of $(\hat{\theta}_1^*, ..., \hat{\theta}_B^*)$ as an approximation of the true distribution of $\hat{\theta}$.

Again, you may produce the histogram of $(\hat{\theta}_1^*, ..., \hat{\theta}_B^*)$, this representing the pdf of $\hat{F}$.

Concretely, let us say that $X_i \sim N(0, 1)$, θ is the median and it is estimated by $\hat{\theta} = X_{(n/2)}$, the $n/2$-th element in the ordered sequence. In Figure 1 the distribution of $\hat{\theta}$ approximated with the non-parametric and parametric bootstrap is plotted. Note that the parametric distribution is smoother than the non-parametric one, since the samples were drawn from a continuous distribution.

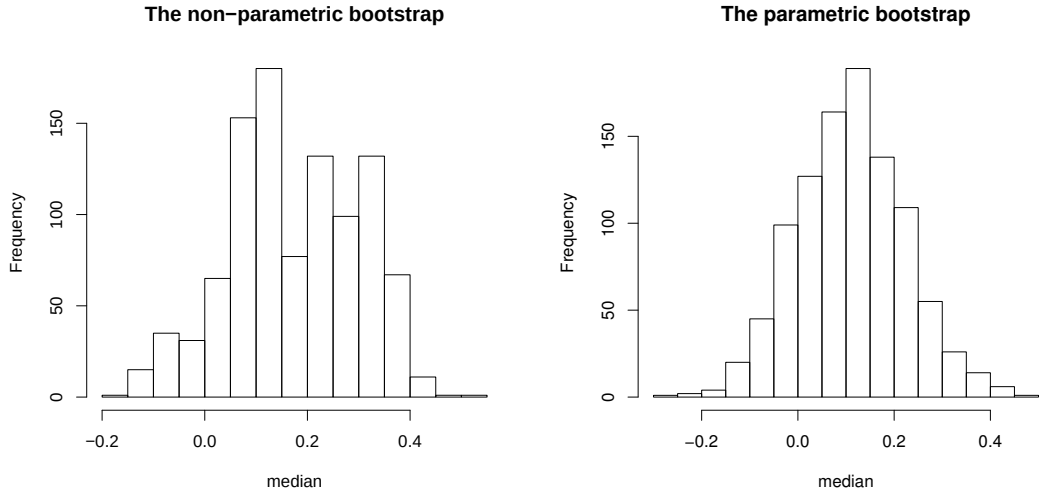


Figure 1: The non-parametric and the parametric bootstrap distribution of the median, $B = 1000$.

3 Bias and variance estimation (* can be skipped, only for those interested)

The theoretical bias and variance of an estimator $\hat{\theta}$ are defined as

$$\mathbf{Bias}(\hat{\theta}) = \mathbf{E}[\hat{\theta} - \theta] = \mathbf{E}[\hat{\theta}] - \theta$$
$$\mathbf{Var}(\hat{\theta}) = \mathbf{E}\left[(\hat{\theta} - \mathbf{E}[\hat{\theta}])^2\right]$$

In words, the bias is a measure of a systematic error ($\hat{\theta}$ tends to be either smaller of larger than θ) while the variance is a measure of random error.

In order to obtain the bootstrap estimates of bias and variance we plug in the original estimate $\hat{\theta}$ (which is a constant given data) in place of θ and $\hat{\theta}^*$ (the distribution of which we get from bootstrap) in place of $\hat{\theta}$. This leads us to the following approximations:

$$\mathbf{Bias}(\hat{\theta}) \approx \frac{1}{B}\sum_i \hat{\theta}_i^* - \hat{\theta} = \bar{\hat{\theta}}^* - \hat{\theta}$$

$$\mathbf{Var}(\hat{\theta}) \approx \frac{1}{B-1}\sum_i (\hat{\theta}_i^* - \bar{\hat{\theta}}^*)^2$$

That is, the variance is, as usual, estimated by the sample variance (but for the bootstrap sample of $\hat{\theta}$) and bias is estimated by how much the original $\hat{\theta}$ deviates from the average of the bootstrap sample denoted $\bar{\hat{\theta}}^*$.

4 Confidence intervals

There are several methods for CI construction with Bootstrap, the most popular being "normal", "basic" and "percentile". Let $\hat{\theta}^*_{(i)}$ be the ordered bootstrap estimates, with $i = 1, ..., B$ indicating the different samples. Let α be the significance level. In all that follows, $\hat{\theta}^*_\alpha$ will denote the

α-quantile of the distribution of $\hat{\theta}^*$. You can approximate this quantile with $\hat{\theta}^*_{([(B+1)\alpha])}$ with the $[x]$ indicating some interpolation or rounding procedure[2].

Basic	$[2\hat{\theta} - \hat{\theta}^*_{1-\alpha/2}, 2\hat{\theta} - \hat{\theta}^*_{\alpha/2}]$
Normal	$[\hat{\theta} - z_{1-\alpha/2}\hat{se}, \hat{\theta} - z_{\alpha/2}\hat{se}]$
Percentile	$[\hat{\theta}^*_{\alpha/2}, \hat{\theta}^*_{1-\alpha/2}]$

with z_α denoting an α quantile from a Normal distribution and $\hat{se}$ the estimated standard deviation of $\hat{\theta}$ calculated from the bootstrap sample.

Basic CI (* can be skipped, only for those interested)

To obtain this confidence interval we start with $W = \hat{\theta} - \theta$ (compare to the classic CI that correspond to a t-test). If the distribution of W was known, then a two-sided CI could be obtained by considering $\mathbf{P}(w_{\alpha/2} \leq W \leq w_{1-\alpha/2}) = 1 - \alpha$, which leads to CI $= [l_{low}, l_{up}] = [\hat{\theta} - w_{1-\alpha/2}, \hat{\theta} - w_{\alpha/2}]$. However, the distribution of W is not known, and is approximated with the distribution for $W^* = \hat{\theta}^* - \hat{\theta}$, with w^*_α denoting the corresponding α-quantile. The CI becomes $[\hat{\theta} - w^*_{1-\alpha/2}, \hat{\theta} - w^*_{\alpha/2}]$. Noting that $w^*_\alpha = \hat{\theta}^*_\alpha - \hat{\theta}$ and substituting this expression in the CI formulation leads to the definition given in the box.

This interval construction relies on an assumption about the distribution of $\hat{\theta} - \theta$, namely that it is independent of θ. This assumption, called pivotality, does not necessarily hold in most cases. However, the interval gives acceptable results even if $\hat{\theta} - \theta$ is close to pivotal.

Normal CI (* can be skipped, only for those interested)

This confidence interval probably looks familiar since it is almost an exact replica of the commonly used confidence interval for a population mean. Indeed, similarly to that familiar CI, we can get it if we have reasons to believe that $Z = (\hat{\theta} - \theta)/\hat{se} \sim N(0, 1)$ (often true asymptotically as the data size $n \to \infty$). Alternatively, we can again consider $W = \hat{\theta} - \theta$, place the restriction that it should be normal and estimate its variance with bootstrap. Observe that the normal CI also implicitly assumes pivotality.

Percentile CI

These type of CI may seem natural and obvious but actually requires a quite convoluted argument to motivate. Without going into details, you get this CI by assuming that there exists a transformation h such that the distribution of $h(\hat{\theta}) - h(\theta)$ is pivotal, symmetric and centered around 0. You then construct a Basic CI for $h(\theta)$ rather than θ itself, get the α quantiles and transform those back to the original scale by applying h^{-1}. Observe that although we do not need to know h explicitly, the existence of such a transformation is a must, which is not always the case.

[2]There is no general agreement on the rounding scheme to use. Some sources use ceiling (integer closest to x from above) and others use flooring (integer closest to x from below).

5 Using the bootstrap to approximate probabilities

We can use the bootstrap as a tool to approximate quantities via random simulations. Say that we wish to use it to approximate the probability of an event. Let's see a practical example of how we could do this.

Say that we have two theoretical populations $\mathcal{P}_1$ and $\mathcal{P}_2$ for which we want to learn some features. As an example $\mathcal{P}_1$ could represent subjects treated with drug A while $\mathcal{P}_2$ is subjects treated with drug B, and say that we wish to learn whether drug B is more effective than A for some pathology, where the mean efficacy rate for subject using A is θ_A and is θ_B for subjects using B. Or we have that $\mathcal{P}_1$ is the (theoretical) population of units of a product produced by a global company during dayshifts, while $\mathcal{P}_2$ is the theoretical population of units produced during nightshift in the same company, and the goal is to compare the mean number of daily faulty products produced in the two shifts (populations are "theoretical" because they include all potential units ever produced in the past, present and future). Say that here the mean number of faulty products is θ_1 for $\mathcal{P}_1$ and is θ_2 for $\mathcal{P}_2$. Both θ_1 and θ_2 are unknown to us.

In both cases above we want to understand the direction of the relationship $\Delta = \theta_A - \theta_B$ (positive, negative or zero?) and of $\Delta = \theta_1 - \theta_2$ (positive, negative or zero?), as of course having $\Delta > 0$ would mean that, in the first case, the efficacy rate for drug A is larger than drug B, etc.

Now, we need statistics and probability, as Δ itself cannot be observed *in the populations*, as the θ's are completely unknown to us. But we have some data. Now let's consider the first example and assume that we have n_A subjects treated with A and n_B subject treated with B. We also know how many of the n_A subjects have improved their condition due to assuming A, and we know how many of the n_B subjects have improved their condition due to assuming B. This way, we take $\bar{X}_A$ and $\bar{X}_B$ as *estimators* for (the unknown) θ_A and θ_B, respectively, where $\bar{X}_A = \#(\text{improved using A})/n_A$ and $\bar{X}_B = \#(\text{improved using B})/n_B$.

While we could just compare $\bar{X}_A$ and $\bar{X}_B$ and declare that, if $\bar{X}_A > \bar{X}_B$ (that is if $\hat{\Delta} = \bar{X}_A - \bar{X}_B > 0$), then "drug A is more effective than B", this would be actually hazardous as only based on the specific dataset we have happened to observe, it is not statistical *inference*. We would like to know more, as for example if we were given different datasets maybe the conclusions would be different. **What we want is to find an approximation to the distribution of $\hat{\Delta}$ (reflecting its variability as we pick different datasets and have different estimated efficacy rates for A and B) so that we can compute an approximation of the probability $Pr(\hat{\Delta} > 0)$.**

A non-parametric bootstrap procedure is as follows, and a parametric bootstrap can be also used in a very similar way. Here R is the number of bootstrap samples (we used B in previous sections):

- Set $r = 1$;
- From the set of n_A subject treated with A, sample subjects (with replacement) n_A times, to obtain a bootstrap sample, and in the latter count how many subjects have obtained an improvement due to using drug A. The ratio between such number and n_A is denoted $\bar{X}^*_A = \#(\text{improved using A})/n_A$.
- do the same as above, but for B: sample with replacement n_B times, ultimately to obtain $\bar{X}^*_B = \#(\text{improved using B})/n_B$.
- Compute and store $\hat{\Delta}^*_r = \bar{X}^*_A - \bar{X}^*_B$.
- increase r to $r + 1$, if $r = R$ stop.

In the end you have a collection of R values $(\hat{\Delta}_1^*, ..., \hat{\Delta}_R^*)$. You can now do several things with these value, such as producing an histogram for it, and you can also check how many bootstrap values are larger than a certain number, see below. For example, you can use $(\hat{\Delta}_1^*, ..., \hat{\Delta}_R^*)$ to approximate the probability

$$Pr(\bar{X}_A - \bar{X}_B > 0) \approx \frac{\sum_{r=1}^{R} \mathbb{I}(\hat{\Delta}_r^* > 0)}{R}$$

where $\mathbb{I}$ is the indicator function such that $\mathbb{I}(x) = 1$ if x is true and 0 otherwise. That is $\sum_{r=1}^{R} \mathbb{I}(\hat{\Delta}_r^* > 0)$ is the number of bootstrap samples for which $\bar{X}_A^* > \bar{X}_B^*$. Therefore, the previous ratio gives an approximation to the probability[3] we were interested in, namely "what is the probability that the mean efficacy rate for drug A is larger than the mean efficacy rate for drug B"?

6 Limitations of the bootstrap

Yes, those do exist. Bootstrap tends to give results easily (too much so, in fact), but it is possible that those results are completely wrong. More than that, they can be completely wrong without being obvious about it. The following are some such situations.

6.1 Infinite variance

In the "general idea" of bootstrapping we plugged in an estimate $\hat{F}$ of F, and then sampled from it. This works only if $\hat{F}$ actually is a good estimate of F, that is it captures the essential features of F despite being based on a finite sample. This may not be the case if F is very heavy tailed, i.e. has infinite variance. The intuition is that in this case extremely large or small values can occur, and when they do they have a great effect on the bootstrap estimate of the distribution of $\hat{\theta}$, making it unstable. As a consequence, the measures of accuracy of $\hat{\theta}$, such as CI, will be unreliable.

The classic example of this is the mean estimator $\hat{\theta} = \bar{X}$ with the data generated from a Cauchy distribution. For it, both the first and the second moments (e.g. mean and variance) are infinite, leading to nonsensical confidence intervals even for large sample sizes. A less obvious example is a non-central Student t distribution with 2 degrees of freedom. This distribution has pdf

$$f_X(x) = \frac{1}{2\,(1 + (x - \mu)^2)^{3/2}} \quad \text{for} \quad x \in \mathbb{R}.$$

where μ is the location parameter, defined as $\mathbf{E}[X]$. So, the first moment is finite and its estimator $\bar{X}$ is consistent. The second moment, however, is infinite, and the right tail of the distribution grows heavier with increasing μ. This leads to the 95% CI coverage probabilities that are quite far from the supposed 95% even when the sample size n is as large as 500 (that is the percentage of confidence intervals that are supposed to contain the true parameter value would be far from the expected 95%).

6.2 Parameter on the boundary

The classical example here is the $U(0, \theta)$ distribution. The maximum likelihood estimate $\hat{\theta}$ is simply $\max(x_1, ..., x_n)$, which will always be biased ($\hat{\theta} < \theta$). In this case the non-parametric

[3] For those who wonder: this is not equivalent to a "p-value". We haven't discussed p-values nor hypothesis testing in this course, but if we did, such p-value could be obtained via bootstrap using a slightly different procedure.

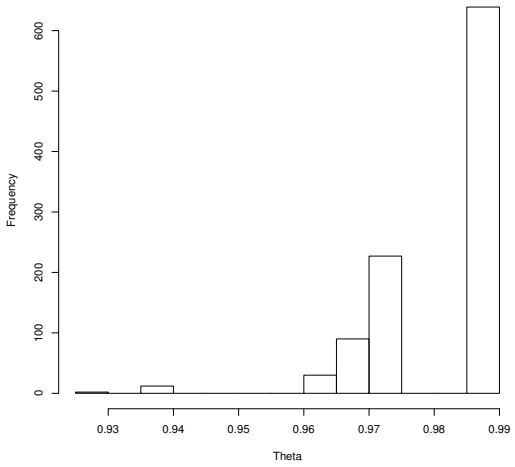


Figure 2: Histogram of 1000 non-parametric bootstrap samples of $\hat{\theta}^*$, the data from the uniform $U(0, \theta)$ distribution. $\theta = 1$.

bootstrap leads to a very discrete distribution, with more than half of the bootstrap estimates $\hat{\theta}^*$ equal to the original $\hat{\theta}$ (Figure 2). Clearly, if the quantiles used in CIs are taken from this distribution the results will be far from accurate. However, the parametric bootstrap will give a much smoother distribution and more reliable results.

6.3 Lack of pivotality (* can be skipped, only for those interested)

In all the CI descriptions above the word "pivotality" shows up. So we can guess that it is a bad thing not to have. To circumvent this, something called "studentized bootstrap" can be used.

The idea behind the method is simple and can be seen as an extrapolation of the basic bootstrap CI. There, we looked at $W = \hat{\theta} - \theta$. Now, we instead consider the standardized version $W = (\hat{\theta} - \theta)/\sigma$, with σ denoting the standard deviation of $\hat{\theta}$. The confidence interval will then be calculated through

$$\mathbf{P}(w_{\alpha/2} \leq W \leq w_{1-\alpha/2}) = \mathbf{P}(\hat{\theta} - w_{1-\alpha/2}\sigma \leq \theta \leq \hat{\theta} - w_{\alpha/2}\sigma) = 1 - \alpha$$

As with the basic CI, the distribution of W is not known and has to be approximated, this time with the distribution of $W^* = (\hat{\theta}^* - \hat{\theta})/\hat{se}^*$. Here, $\hat{se}^*$ denotes the standard deviation corresponding to *each bootstrap sample*.

This CI is considered to be more reliable than the ones described earlier. However, there is a catch, and this catch is the estimate of the standard deviation of $\hat{\theta}^*$, $\hat{se}^*$. This estimate is not easily obtained. You can get it either parametrically (but then you need a model which you probably don't have) or through re-sampling. This means that you have to do an extra bootstrap for each of the original bootstrap samples. That is, you will have a loop within a loop, and it can become very heavy computationally.

6.4 Further reading (* can be skipped, only for those interested)

For a wider and deeper treatment of the bootstrap see chapters 10 and 11 in *Computer Age Statistical Inference* by Bradley Efron and Trevor Hastie. The PDF has been made freely

available from the publisher at `https://web.stanford.edu/~hastie/CASI/`.

6.5 Primer on Gaussian mixture models

Gaussian mixture models are used in the exercises, so here are some notions and how to use them with Matlab.

A **mixture model** is a probabilistic model assuming that data are generated from a combination of several underlying probability distributions. You may think of each "sub-distribution" as representing a different subgroup or component in a hypothetical population. For example, Figure 3 shows the distribution (in black) of heights in a hypothetical population of humans. Then, this distribution is assumed to result as the weighted sum of two distributions each representing a subpopulation, the heights for the male and the heights for the female subpopulations. This makes sense intuitively, and the male and female subpopulations will have their specific means and standard deviations. Of course we could have also split the population according to other criteria, such as age groups or nationality, but for simplicity with stick to a categorization by gender.

Formally, the density (pdf) or probability mass function (pmf) of a mixture model for a random variable X (height, in the example) is defined as:

$$f(x) = \sum_{k=1}^{K} \pi_k f_k(x)$$

where:

- K is the number of components (subgroups), eg $K = 2$ if we have male and female subpopulations,
- π_k are the **mixing proportions** (or mixing probabilities), satisfying $\pi_k \geq 0$ and $\sum_{k=1}^{K} \pi_k = 1$. These values indicate how much each subgroup contributes to the overall distribution. If we randomly select an individual from the overall population, π_k is the probability that the individual belongs to component k ($k = 1, ..., K$). For example if $k = 2$ indicates the female subpopulation, then π_2 is the probability of sampling a female subject in the population.
- $f_k(x)$ are the component probability density functions (pdf) if X is continuous, and if X takes discrete values they are probability mass functions (pmf).
- Suppose each f_k is a pdf and hence integrates to 1, then also $\int f(x)dx = \sum_{k=1}^{K} \pi_k \int f_k(x)dx = \sum_{k=1}^{K} \pi_k = 1$, as expected. An analogous result holds if we we deal with pmfs.

Example: Gaussian Mixture Model for Human Heights

Suppose we want to model the distribution of adult human heights using a Gaussian mixture model with two components ($K = 2$): the first component represents heights of men, the second represents heights of women. For this example, we assume that both groups have 1-dimensional normal distributions, with different means and standard deviations. However, here we assume that a subject from the population has the same chance to belong to the male or the female group ($\pi_1 = \pi_2 = 0.5$).

Let:

- mean height $\mu_1 = 178$ cm and standard deviation $\sigma_1 = 7$ cm for men,
- mean heights $\mu_2 = 165$ cm and standard deviation $\sigma_2 = 6$ cm for women,

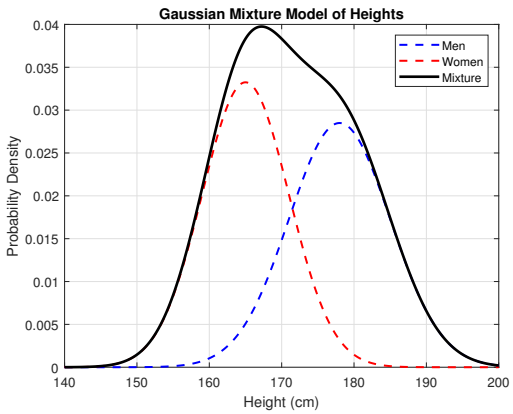


Figure 3: Pdf of the mixture $f(x)$ (solid black) and the pdfs of the two components f_k (dashed). Here $\mu_1 = 178$ cm and $\sigma_1 = 7$ cm for men, and $\mu_2 = 165$ cm and $\sigma_2 = 6$ cm for women, and $\pi_1 = \pi_2 = 0.5$.

· Mixing proportions: $\pi_1 = 0.5$, $\pi_2 = 0.5$.

This model reflects the idea that the overall population is composed of two subgroups with different average heights and different spreads. The probability density function of the mixture is:

$$f(x) = \pi_1 \cdot \phi_1(x; \mu_1, \sigma_1^2) + \pi_2 \cdot \phi_2(x; \mu_2, \sigma_2^2)$$

where, for this example, $\phi_k(x; \mu_k, \sigma_k^2)$ is the 1-dimensional normal density pdf:

$$\phi_k(x; \mu_k, \sigma_k^2) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x-\mu_k)^2}{2\sigma_k^2}\right). \tag{1}$$

Substituting the values:

$$f(x) = 0.5 \cdot \frac{1}{\sqrt{2\pi \cdot 7^2}} \exp\left(-\frac{(x-178)^2}{2 \cdot 7^2}\right) + 0.5 \cdot \frac{1}{\sqrt{2\pi \cdot 6^2}} \exp\left(-\frac{(x-165)^2}{2 \cdot 6^2}\right)$$

and we have the plot of $f(x)$ in Figure 3.

We can make the two components more separated by shrinking the standard deviations. Let's take $\sigma_1 = 5$ (men) and $\sigma_2 = 3$ (women), and then we have Figure 4. Notice how the men's curve in blue has a lower peak: that's because its mass is more spread due to its standard deviation σ_1 being larger than σ_2, and to preserve the fact that it must still be $\int f_1(x)dx = 1$ by definition, then it is necessary that the height of f_1 should be lower than f_2. Finally, (only for illustration purposes) if we give more importance[4] to the women's component by setting $\pi_2 > \pi_1$, for example with $\pi_1 = 0.3$ and $\pi_2 = 0.7$, then we have Figure 5.

How to estimate parameters of Gaussian mixtures in Matlab

Suppose you are given a vector of `data` and want to estimate $\theta = (\mu_k, \sigma_k, \pi_k)_{k=1}^K$ by maximum likelihood. This is possible thanks to the expectation-maximization (EM) algorithm, which we do not illustrate but it is the standard way to fit mixture models to data. In Matlab, this can be done with `GM = fitgmdist(data,K)` or via `GM = fitgmdist(data,K,'Start',startpar)`, the

[4]It makes sense only of we knew that there are more women than men otherwise the proportion should be set nearly equal

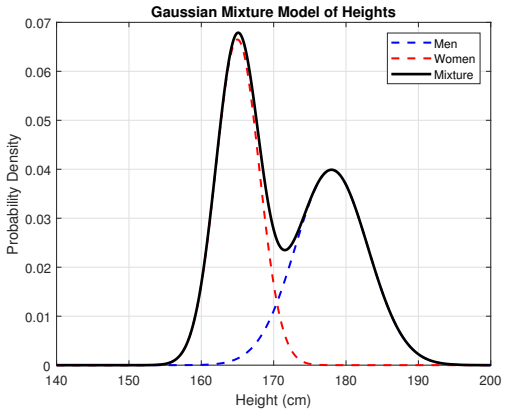


Figure 4: Same as Figure 3 but with $\sigma_1 = 5$ and $\sigma_2 = 3$.

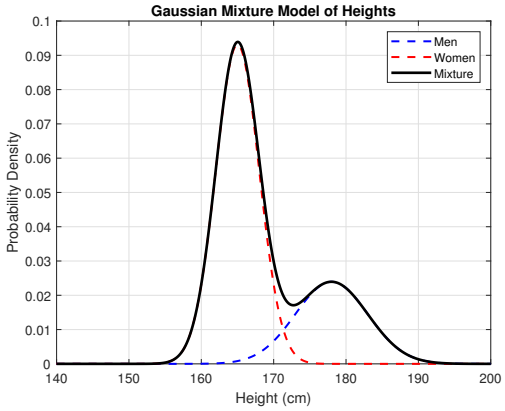


Figure 5: Same as Figure 4 but with $\pi_1 = 0.3$ and $\pi_2 = 0.7$.

latter form is to help the algorithm to converge to appropriate estimates by providing suitable starting values for θ via a vector `startpar` (specified by you).

However Matlab's sintax will more generally assume that X is a multivariate random variable, and hence $f(x)$ is assumed encoded as a mixture of K *multivariate* Gaussian distributions. Therefore in full generality, if $x \in \mathbb{R}^d$, we have the mixture of K d-dimensional ($d \geq 1$) multivariate Gaussian pdfs written as

$$f(x) = \sum_{k=1}^{K} \pi_k \phi_k(x; \mu_k, \Sigma_k)$$

and in such case

$$\phi_k(x) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k)\right) \tag{2}$$

where $\mu_k \in \mathbb{R}^d$, $\Sigma_k \in \mathbb{R}^{d\times d}$ and $|\cdot|$ is the determinant of a matrix. Recall that Σ_k is the variance-covariance matrix

$$\Sigma_k = \begin{pmatrix} \sigma_1^2(k) & \sigma_{12}(k) & \cdots & \sigma_{1d}(k) \\ \sigma_{12}(k) & \sigma_2^2(k) & \cdots & \sigma_{2d}(k) \\ \vdots & \vdots & & \vdots \\ \sigma_{1d}^2(k) & \sigma_{2d}(k) & \cdots & \sigma_d^2(k) \end{pmatrix}$$

where we used that covariances σ_{ij} are symmetric (ie $\sigma_{ij} = \sigma_{ji}$, $i \neq j$, for every k), where the diagonal elements are variances (hence positive) while off-diagonal elements are covariances. Obviously, if $d = 1$ (x one-dimensional), mathematically (2) becomes (1).

For the exercise you are asked to solve, we do not assume to deal with mixtures of multivariate Gaussians, only mixtures of K one-dimensional Gaussians ($d = 1$), so $\Sigma_k = \sigma_k^2$. **Therefore, for the exercise,** `fitgmdist` **will return you maximum likelihood estimates of the means μ_k, mixing proportions π_k and the $\Sigma_k = \sigma_k^2$ (not σ_k) ($k = 1, ..., K$),** again this is because the function encodes mixtures for multivariate Gaussians generally having $d \leq 1$. This is why in examples below when I write, say, `startpar.Sigma` this is meant to represent a covariance matrix even if the problem has $d = 1$.

Here follow some illustrative code with $d = 1$, where the data is a vector.

```
% fitgmdist fits some data with a K-dimensional mixture using maximum likelihood
GM = fitgmdist(data,K)  % generates a "structure" GM
GM.mu  % extract the vector of fitted means (mu1,...,mu_K)
GM.Sigma  % extract the fitted variance-covariance matrices (S_1,...,S_K)
GM.ComponentProportion  % extract the mixing proportions pi_1,...,pi_K
```

We now consider an hypothetical data vector `X`. It is not rigorously generated from a mixture model (to discover how to properly generate data from a generic mixture model, see the section on page 15), but it will work ok for this example, and we fit X with a mixture consisting of two 1-dimensional Gaussians.

```
% we simulate data and call this X. Notice X is a column vector of length n=2000.
mu = [-3 ; 1];       % means to generate data. mu(1) is for k=1, mu(2) is for k=2
```

```
sigma = [sqrt(2);  sqrt(1)];  % standard deviations to generate data
rng(1); % For reproducibility (similar to set.seed() in R)
X = [normrnd(mu(1),sigma(1),[1000,1]);normrnd(mu(2),sigma(2),[1000,1])];
% fit X with a 2-components mixture
>> GM = fitgmdist(X,2);
>> GM.mu

ans =

    -3.0229
     1.0865
% that's a good estimate of mu, as true values are [-3;1]

>> GM.Sigma
ans(:,:,1) =

    1.9550   % recall for Matlab this is a VARIANCE for component 1
% so the standard deviation is sqrt(1.9550)=1.3982

ans(:,:,2) =

    1.0979   % recall for Matlab this is the VARIANCE for component 2
% so the standard deviation is sqrt(1.0979)=1.0478
>> GM.ComponentProportion

ans =

    0.5003    0.4997  % these are the fitted \pi_1 and \pi_2 (and sum to 1)

 % However if we repeat the fitting again we get
>> GM = fitgmdist(X,2);
>> GM.mu

ans =

    1.0864
   -3.0231
```

The last result shows that occasionally the estimated values flip positions (the negative mean is now in the second slot instead of the first). This is because each call to fitgmdist uses some random starting values in the optimization, and converges to approximately the same optima but with positions switched. This is because mixture models written like

$$f(x) = \pi_1 \cdot \phi(x; \mu_1, \Sigma_1) + \pi_2 \cdot \phi(x; \mu_2, \Sigma_2)$$

or (notice we flip the order of components)

$$f(x) = \pi_2 \cdot \phi(x; \mu_2, \Sigma_2) + \pi_1 \cdot \phi(x; \mu_1, \Sigma_1)$$

give the same value of $f(x)$ for any x, regardless of how we order the components. This means that the components ordering may be *unidentifiable*, and the estimates "switch" position, which

is a serious problem whenever the components have a specific interpretation (eg (male,female) rather than (female, male)), and the switch of course make the components ordering meaningless with consequences on results interpretation, as we will think that the first estimated mean is for, say, male, while it should be for female.

This issue can be solved by providing suitably ordered starting values for the optimization parameters, as done below.

```
histogram(X)
% By looking at the histogram I guess some starting values for the means (-3.2,1.2),
% and I first write the smallest mean (-3.2) then the larger one (1.2).
startpar.mu = [-3.2; 1.2]; % notice it's a COLUMN vector
startpar.Sigma(:,:,1) = 2.2; % some guessed variance for component 1
startpar.Sigma(:,:,2) = 0.8; % some guessed variance for component 2
startpar.ComponentProportion = [0.5,0.5];

% let's refit using the guessed starting parameters
rng(123)  % so we get the same estimates
>> GM = fitgmdist(X,2,'Start',startpar);  # we provide starting values explicitly
>> GM.mu

ans =

    -3.0052
     1.0978
% GREAT! The right ordering is preserved

Retry as many times as you wish and you'll get very similar solutions over and over
>> GM = fitgmdist(X,2,'Start',startpar);
GM.mu

ans =

    -3.0052
     1.0978
% again, correct ordering
```

A comment on how I specified the starting variances for the optimization: for example for the first component I wrote `startpar.Sigma(:,:,1) = 2.2` and this means "for mixture component 1 the covariance matrix is actually a scalar (variance) equal to 2.2".

How to simulate draws from a Gaussian mixture

We have learned how to fit parameters of a Gaussian mixture. But what if we wish to simulate many random numbers of the type $x^* \sim f(x)$ where $f(x)$ is the pdf of a mixture model? **This is useful for parametric bootstrap** where we need to be able to simulate from the model. We just need to learn how to sample once from $f(x)$ and then you repeat the procedure as many times as you wish.

The procedure is easy[5]. First we randomly draw a component $k^* \in \{1, ..., K\}$ (from a *multinomial distribution* with probabilities $\pi_1, ..., \pi_K$), and then we draw $x^* \sim f_{k^*}(x)$ from the distribution of the selected component. The latter step, you should know how to do it (but see the last page of this document), as it is simply about sampling from $\phi_{k^*}(x)$ a Gaussian distribution. The first step is perhaps less obvious to you, but it is like tossing a biased "dice" with K sides, where each side of the dice has a corresponding probability π_k. See below.

Say that the goal is to sample a sequence of three values (x_1^*, x_2^*, x_3^*) i.i.d from a mixture $f(x)$ having $K = 4$ one-dimensional Gaussian components. Below I do not provide the $(\mu_k, \sigma_k, \pi_k)_{k=1}^K$ values, you can decide these yourself.

Generally we can sample a single k^* as follows (it samples k from a *multinomial distribution*):

```
weights = [pi_1, pi_2,...,pi_K];  % collect all the estimated mixing probabilities
cum_weights = cumsum(weights);  % cumulative sum of all probabilities
r = rand(1);  % a single uniform draw in (0,1)
k_star = find(r <= cum_weights, 1);  % this is a single randomly sampled component
```

Example with $K = 4$, and we generate a sequence of three values (x_1^*, x_2^*, x_3^*) from the mixture:

```
weights = [0.05,0.05,0.6,0.3];  % must sum to 1
cum_weights = cumsum(weights);  % cumulative sum of all probabilities
r = rand(1);  % a single uniform draw in (0,1)
k_star = find(r <= cum_weights, 1);  % this is a randomly sampled component

k_star =

     3
% we sampled the third component

% so let's simulate accordingly from the third Gaussian:
x_star = normrnd(mu_3,sigma_3);  % here sigma_3 is a standard deviation
% Great so we got our first sample. We want two more.


>> r = rand(1);  % a single uniform draw in (0,1)
k_star = find(r <= cum_weights, 1)

k_star =

     4
% we sampled the fourth component.

% so let's simulate accordingly from the fourth Gaussian:
```

[5]The probabilistic justification is: if you want a sample $x^* \sim f(x)$ it is equivalent (and simpler) to sample both $(x^*, k^*) \sim f(x, k) = f(x|k)f(k)$ where $f(k) = \pi_k$ is the probability to pick a $k \in 1, ..., K$ while $f(x|k)$ is the conditional density of x while knowing that the specific component k has been sampled. So if you have first sampled $k = 2$ from $f(k)$ then you sample $x^* \sim f(x|k = 2) = \phi_2(\mu_2|\Sigma_2)$.

```
x_star = normrnd(mu_4,sigma_4);


>> r = rand(1);  % a single uniform draw in (0,1)
k_star = find(r <= cum_weights, 1)

k_star =

     3
% once more, we sampled the third component
% so let's simulate accordingly from the third Gaussian:
x_star = normrnd(mu_3,sigma_3);
```

In summary, we obtained independently three `x_star` values, the (x_1^*, x_2^*, x_3^*) we wanted:. Notice that since the mixture components are 1-dimensional Gaussians we used `normrnd()`, which wants a standard deviation as second argument (see the last page of this document).

Exercise 1 (to be solved with MATLAB), 6 points

You are given simulated data as a file `diabetes` on Canvas. This synthetic dataset consists of 500 blood glucose concentrations (mg/dL) each obtained from 500 hypothetical subjects.

```
% load data
load('diabetes.txt');
```

We assume that measurements have been obtained on three groups of subjects, healthy, prediabetics and diabetics, where "healthy" subjects have smaller blood glucose measurements while the diabetic ones have the largest measurements. You can easily see the data distribution and notice the three groups by looking at an histogram[6] of the data.

(i) (a)[0.5 point] Fit a Gaussian mixture model with three components on the given data, and report the obtained maximum likelihood estimates for all the parameters $(\mu_k, \sigma_k, \pi_k)_{k=1}^K$ (notice I wrote σ_k and not σ_k^2). It is important to obtain stable estimates by providing suitable starting values for the parameters so not to obtain the "switching" problem.
(b) [0 points] What is the probability that a subject from the population is prediabetic?

(ii) [1 points] Simulate a vector of $n = 500$ observations from a three-components Gaussian mixture using the estimated parameters you obtained in (i). Compare the histogram of the simulated data with the histogram of the original `diabetics` dataset. Do you observe a good agreement? *[Place rng(123) in the code before simulating data so we get the same results.]*

(iii) [2 points] Run a nonparametric bootstrap procedure to infer the distribution of all parameters $(\mu_k, \sigma_k, \pi_k)_{k=1}^K$, using $B = 2,000$. For each parameter, report the histogram for the bootstrap distribution, and the 95% confidence intervals based on the percentile method. *[Place rng(123) in the code before simulating data so we get the same results.]*

[6] By the way, `histogram()` and `hist()` both work, the outputs are slightly different due to different algorithms to bin the data. Whatever you use is ok, but the former one is nicer and recommended.

(iv) [2 points] The same as in (iii) but using the parametric bootstrap. This time though, report the histograms only (not the percentiles). Do you see any appreciable difference or is the method giving similar results to the nonparametric one? And if you were told that the diabetes.txt data were indeed generated by a Gaussian mixture model with three components, and not by some other type of model, what would it be your appreciation of the performance of the nonparametric bootstrap at this point?

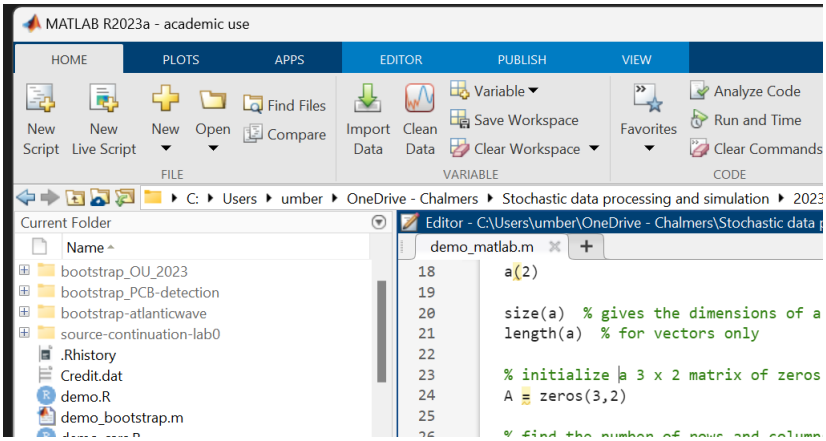
(v) [0.5 points] Using the results from the nonparametric bootstrap, what is the probability that the estimated mean glucose concentration is larger than 85 for healthy subjects? Also, what is the probability that the estimated mean glucose concentration is larger than 85 for prediabetic subjects?

7 Quick Matlab primer

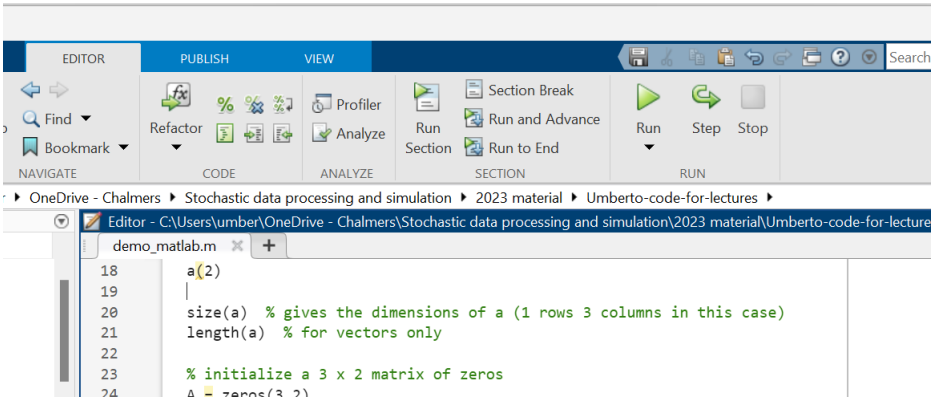
Just a few words on Matlab: if you got used to R and have never used Matlab, there is not an awful lot to learn, you can check cheat-sheets:

- MATLAB basic functions https://www.mathworks.com/content/dam/mathworks/fact-sheet/matlab-basic-functions-reference.pdf
- a MATLAB/R cheat sheet is at http://mathesaurus.sourceforge.net/octave-r.html
- a MATLAB/Python/R cheat sheet is http://mathesaurus.sourceforge.net/matlab-python-xref.pdf

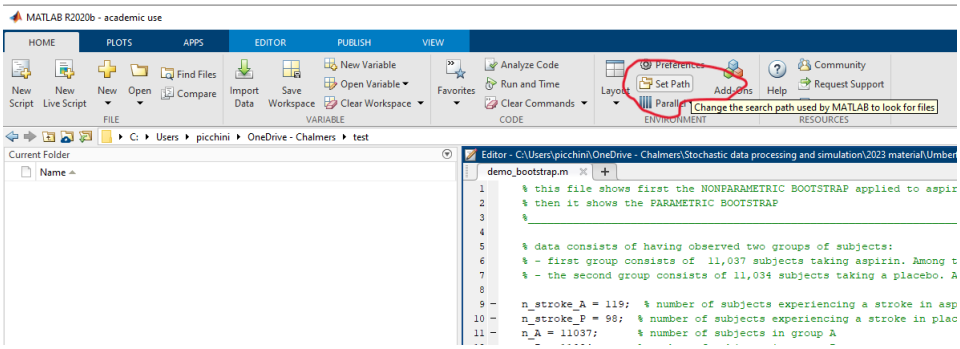
Create a script: Click on the HOME tab and click on New Script to the left:

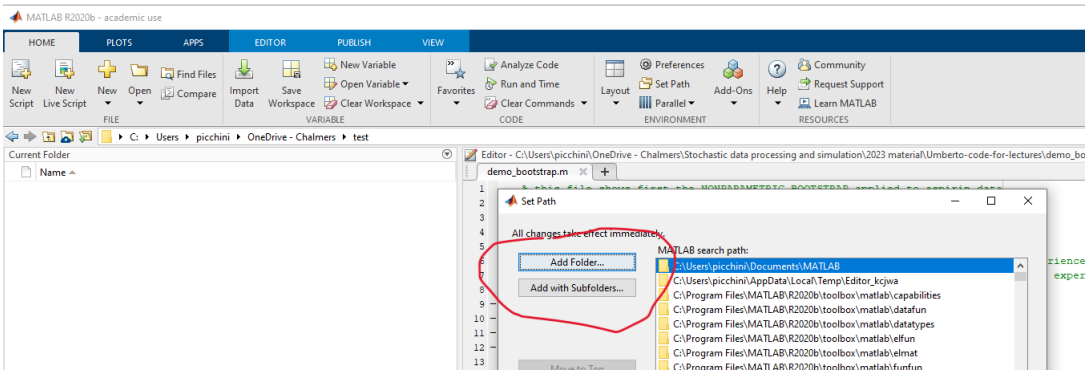


Run an entire script: First click o the script itself, so a new tab called EDITOR will appear (otherwise you won't see it if no script is open). Then click on the large green arrow "Run".



Another relevant thing is to know how to change your work directory:
click on the HOME tab on the top then select "Set Path", then click on either "Add folder" or even "Add with subfolders".





Basically you can use the previous procedure to add as many folders as you need, if your codes are scattered across different directories, so Matlab is aware of where to look to find them (I recommend to keep things tidy though).

Examples of things to know if you are new to Matlab: also check the file `demo_matlab.m`

- vectors are defined with square parentheses, eg

```
>> a=[1,2,3]
a =
     1     2     3
```

And you access vector's elements using round parenthesis, eg `a(3)`

```
>> a(3)
ans =
     3
```

- use semicolons ";" a lot or your scripts will print lots of things on screen , e.g.

```
>> a=[1,2,3]
a =
```

```
     1     2     3
% whereas

>> a=[1,2,3];  % this will not print anything on-screen
```

- Comments can be written via %
- Adding a plot to an already existing figure. Use `hold on` and `hold off`.

```
plot(randn(1,10),randn(1,10),'o')
hold on  % this will allow superimposing a plot on top of the existing plot
plot([1:0.01:10],[[-10:0.05:35]],'m--')  % added a dashed magenta line
hold off  % stop superimposing
```

Help/Documentation: for info on MATLAB's commands you may use both `help` and `doc` at the prompt, followed by an appropriate keyword, e.g. `help histogram` or `doc histogram`.

Setting the pseudorandom numbers seed: this can be done via `rng(123)`, where 123 is just an example (similarly to R's `set.seed()`).

Be careful when specifying dimensions for pseudorandom numbers: unlike in R, the function to produce a vector of uniform random numbers expects **two** dimensions to be provided. That is `rand(10000,1)` generates a 10000×1 column vector, while `rand(1,10000)` generates a row vector. However, if you were to write `rand(10000)`, this would create a $10,000 \times 10,000$ matrix.

Producing Gaussian pseudorandom numbers: you can use `randn` to produce a single draw from $N(0,1)$ (that is from the *standard Gaussian* aka *standard normal* distribution). To sample from a generic $N(\mu, \sigma^2)$ you can type $\mu + \sigma$`*randn` or use `normrnd(`μ`,`σ`)`. Notice σ^2 in $N(\mu, \sigma^2)$ is the variance, but software actually requires you to type in the standard deviation σ.

How about dimensions? You have to be careful: if you wish to simulate a column vector of 10 elements from $N(0,1)$ then you can just type `randn(10,1)` *however* if you want to simulate a column vector of 10 elements from $N(\mu = 3, \sigma^2 = 4)$ then you have to write `normrnd(3,2,[10,1])` (notice the dimensions are specified between square brackets and the standard deviation is specified as 2, not the variance).

More examples in `demo_matlab.m`.