WATERLOO CENTRE FOR ASTROPHYSICS

Two-Point Statistics Final Project

Phys 788 Statistical Tools for Astronomers

Student
James Morawetz
M.Sc. Physics Thesis Option
University of Waterloo

Course Instructors
Dr. Marco Bonici, Dr. Pierre Burger,
Dr. Jack Elvin Poole, Dr. Ana Ennis,
Dr. Simone Paradiso, Dr. Liza Sazonova

Written report accompanying the final presentation for completion of the graduate course Phys 788 (Statistical Tools for Astronomers)

Apr. 22, 2024

1 Course Methods

- **Numerical Covariance Matrices** – In the absence of an analytical model, e.g. for the non-linear regime where much constraining power is contained, one must compute covariance matrices numerically:

$$\boldsymbol{C} = \frac{1}{N_r - 1}\sum_{i=1}^{N_r}(\boldsymbol{d_i} - \bar{\boldsymbol{d}})(\boldsymbol{d_i} - \bar{\boldsymbol{d}})^T, \quad \bar{\boldsymbol{d}} = \frac{1}{N_r}\sum_{i=1}^{N_r}\boldsymbol{d_i}$$

  where $\boldsymbol{C}$ is the covariance matrix, N_r is the number of realizations and $\boldsymbol{d_i}$ is the ith realization of the data vector. Inverting this expression gives a biased estimate of the inverse covariance matrix. To account for this we multiply by the Hartlap factor: $h = \frac{N_r - N_d - 2}{N_r - 1}, \boldsymbol{C^{-1}} \to h\boldsymbol{C^{-1}}$ where N_d is the number of entries in the data vector. While numerical covariance matrices are advantageous for probing small scales, the number of simulations must be sufficiently large compared to the number of entries in the data vector to be considered reliable making it a very computationally expensive endeavour.

- **Neural-Network Emulators** – Many late-time measurements such as from galaxy surveys (in particular on small, non-linear scales) lack precise analytical models, necessitating numerical simulations. It is not feasible to generate simulations for all possible cosmological models. Instead one can design an emulator, based on neural networks, to output summary statistics for any cosmological parameters based on a finite training set of statistics computed on simulations spanning the parameter space. While emulators reduce the computational cost significantly, they are prone to issues such as overfitting which can result from a poor choice of hyperparameters. We briefly describe the relevant hyperparameters and how they are chosen in our assignments:
  - Number of Training/Testing samples: A large training sample allows the model to be more generalized. But more training samples means fewer test samples to verify accuracy. A reasonable compromise is an 80%-20% balance between training/test sets.
  - Number of Layers/Nodes per Layer: Using more layers/nodes per layer allows for recognizing more complicated patterns and thus giving more accurate predictions. However too many layers/nodes per layer leads to overfitting and thus poor generalization beyond the training set. Excessive layers/nodes also leads to a larger computation time. A good balance in our case was 512 nodes per layer and 4 layers.
  - Learning Rates: The learning rates represent the step sizes taken in parameter space as the network learns. Greater step sizes reduce computation time but also increase the risk of not converging and oscillating around the local minima indefinitely, resulting in weaker accuracy. To account for this we include the following learning rates: $10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$.
  - Batch Sizes: The batch size is the number of samples from the training set used per iteration (update of the weights/biases) during training. A larger batch size is more representative of the full training set and thus gives more accurate gradients. Though higher batch sizes increases the computation time. A reasonable compromise in our case was a batch size of 500 (for a training set with 8000 samples).
  - Patience Values: The patience value is the number of epochs to wait before switching to a lower learning rate if the loss stops improving. A patience value which is too small is not ideal because the network may not be ready to move on to the lower learning rate. But an excessively large patience value is inefficient as time is wasted when stuck at a constant learning rate despite the loss not improving. We should choose a value that is large enough that it is not reached ideally. An optimal patience value in our case was 100.
  - Max Epochs: This is the maximum number of epochs before terminating the training process altogether. Setting this quantity too small is problematic as the network will not be sufficiently trained. Choosing a value too large can unnecessarily prolong the runtime if further improvement is negligible. We found that setting the maximum epochs to 1000 worked well.

- **Fisher Information** – The Fisher information matrix is defined as:

$$F_{ij} = \frac{\partial \boldsymbol{d^T}}{\partial \theta_i}\boldsymbol{C^{-1}}\frac{\partial \boldsymbol{d}}{\partial \theta_j}$$

  where (i, j) are the parameter indices, $\boldsymbol{d}$ is the mean summary statistic as a function of parameters and $\boldsymbol{C^{-1}}$ is the inverse covariance matrix. The Cramér–Rao bound states that the inverse of the Fisher matrix provides a lower bound on parameter variances (equality holds when the estimator is optimal), namely: $\sigma(\theta_i) \geq F_{ii}^{-1}$. Fisher Information provides a rough estimate for the achievable level of precision when estimating cosmological parameters with a particular statistic on a given sample volume and tracer. One of the flaws is that it relies on numerical estimates for the covariance matrix and derivatives which produce overly optimistic constraints.

- **Principle Component Analysis** – Principle Component Analysis (PCA) is a data compression method. The data vector is projected into a new optimized basis where the components are uncorrelated and sorted in order of decreasing variance. As a result the elements near the end of the vector carry the least information allowing us to compress the data to lower dimensions, i.e. reduce the number of elements in the data vector, while minimizing information loss. The new optimized basis is determined from the covariance matrix – the eigenvectors of the covariance matrix are the axes of the new PCA basis and the eigenvalues represent the associated variance in those given directions. Let $\boldsymbol{R}$ represent the (normalized) matrix of eigenvectors of the covariance matrix. The data vector is then rotated in the direction to align with these new axis: $\boldsymbol{d'} = \boldsymbol{Rd}$. Likewise we obtain the covariance matrix in the new basis (where it will become diagonal) as follows: $\boldsymbol{C'} = \boldsymbol{R^T C R}$. It is ideal for situations when data must be compressed for storage or computing constraints. A disadvantage is that it only reduces information loss if the existing data are significantly correlated; if the original basis elements are already uncorrelated, it is already in the PCA basis and no further reduction in information loss is possible!

- **Markov Chain Monte Carlo (MCMC)** – By Bayes' theorem the posterior distribution is proportional to the likelihood function times the prior distribution: $p(\boldsymbol{\theta}|\boldsymbol{d}) \propto p(\boldsymbol{d}|\boldsymbol{\theta})p(\boldsymbol{\theta})$. Normalization is performed to ensure probability sums to unity. Furthermore the log likelihood function is proportional to the chi-squared function (through factor $-1/2$). Specifically:

$$\ln L = -\frac{1}{2}(\boldsymbol{d} - \boldsymbol{t(\theta)})^T \boldsymbol{C^{-1}}(\boldsymbol{d} - \boldsymbol{t(\theta)})$$

  Although we can evaluate the posterior distribution for any data vector, evaluating the moments of the distribution, i.e. parameter means/covariances, etc., involves integrating over parameter space which is computationally prohibitive in higher dimensions. To work around this we can randomly sample from the posterior distribution to obtain estimates of the parameter means/covariances. To do this we can apply algorithms such as Metropolis-Hastings where we start at some random position in parameter space and evaluate the likelihood function. A random step is then taken in parameter space and the likelihood is re-evaluated at this new location. And the probability of making the movement is based on the ratio of the likelihoods at those two separate locations. When enough steps from a number of different initial starting positions are combined, we can generate contours of the likelihood in order to extract best fit parameters and their uncertainties. While MCMC is beneficial for this reason, it is also prone to problems such as getting stuck in local minima or choosing poor starting positions and not sampling near the peak of the distribution. Avoiding this requires carefully chosen hyperparameters, which we will now list off briefly and describe how they are chosen:

  - Total Steps: An MCMC chain is initiated with a certain number of walkers each starting at a random location within the specified prior range. Each walker then takes random steps throughout the parameter space and moves/rejects based on the Metropolis-Hastings algorithm. Using more steps is beneficial as a greater number of samples means there is less noise in the final parameter estimates/constraints, but also means the chains will take longer to run. A reasonable compromise in our case was 2000 steps.
  - Burning Steps: Each walker in the MCMC chain also discards a specific number of steps when the walker is starting out to eliminate the samples before the chain has settled. The number of burning steps must be sufficiently large to eliminate all the bad samples but also not unnecessarily large that we have to perform too many steps to compensate for this. We found that 100 burning steps was sufficient to account for this.
  - Number of Walkers: Using multiple walkers allows us to start at different positions within the parameter space without worrying about our results getting distorted based on having started at one particular location. The total number of samples within the chain is the product of the number of walkers and the number of steps per walker (barring the burning steps). However the particular choice for number of walkers vs number of steps does indeed matter. To obtain better coverage of the parameter space, we found that using more walkers with fewer steps generally worked best. But we also needed to be careful to take sufficient steps to ensure the distribution is converged for each walker. We struck a balance by choosing 100 walkers where each performed 2000 steps (minus the 100 burning steps) as specified above.
  - Prior Range: We do not assume any existing knowledge of the parameters before performing the MCMC chain. But we do have to set a prior range, i.e. a uniform distribution outside of which the probability is zero, to ensure that the MCMC chain is prohibited from walking too far away from the peak likelihood. To ensure that the MCMC chain spends as much time near the peak as possible it is ideal to set the prior range to tightly encompass the peak of the likelihood but still wide enough to enclose the confidence intervals of interest. This avoids the chain getting stuck in local minima far away from the peak. We chose these priors by first running a chain using a wider prior range and then eyeballing a prior range for each parameter that comfortably encompasses the full 95% contours but not much larger.

2 Summary of Assignments in Project

- **Assignment 1 (Covariance Matrices)** – The first assignment focused on the stability of covariance matrices, chi-squared distributions and the Hartlap correction. The data vector had 900 entries. When using the analytic covariance matrix the chi-squared distribution had 900 degrees of freedom leading to the plot in figure 1. We then computed numerical covariance matrices using 500, 1000, 5000 and 10000 samples. The covariance matrix with only 500 samples was not invertible due to having fewer samples than number of entries (characterized by negative eigenvalues). While the remaining covariance matrices were invertible, the resulting chi-squared distributions did not match what was expected for 900 degrees of freedom. Figure 2 shows the chi-squared distribution for 10000 realizations which is skewed to values larger than 900. However after applying the Hartlap factor the chi-squared distributions were no longer biased and had means close to 900 as shown in figure 3. Another issue we studied was that one should never measure chi-squared values on the samples that were used to make the covariance matrix in the first place. Figure 4 shows the chi-squared distribution for 10000 realizations with the Hartlap correction applied, but measured on the same samples used to generate the covariance matrix. We see that the distribution is still biased in spite of the correction!

- **Assignment 2 (Emulators)** – The second assignment focused on emulators, Fisher information and PCA. We were provided a large set of data vectors and their associated parameters. We split this sample into two subsets: a training set and a testing set. The training set was passed to COSMOPOWER and the emulator was trained; the percentile discrepancy between the test samples and their predictions from the emulator are shown in figure 5. The hyperparameters discussed in the previous section were adjusted to minimize these fluctuations. Once the emulator was trained we obtained estimates of the derivatives by computing the predicted features with each cosmological parameter varied slightly above/below their reference value. Using these derivatives and the inverse covariance matrix from the previous assignment, we computed the Fisher matrix and inverted to obtain the matrix of parameter covariances. Finally, we performed PCA data compression and measured the resulting Fisher constraints as a function of the number of PCA elements used, shown in figure 6. As can be seen a relatively small number of elements are needed to capture most information, i.e. come within 10%/5% of the maximum constraining power!

- **Assignment 3 (MCMC)** – The third assignment focused on running MCMC chains to obtain posterior distributions for cosmological parameters. We were provided the pre-trained emulator from the second assignment and a noisy realization of the data, from which we ran an MCMC chain (using the hyperparameters discussed in the previous section) to make contour plots for the confidence intervals. Figures 7 and 8 show the posterior and chi-squared distributions when the analytic covariance matrix was used – the chi-squared distribution peaks near 900 as expected given the number of degrees of freedom. Figures 9, 10, 11 show the posterior and chi-squared distributions when the numerical covariance matrices with/without the Hartlap correction were used – the chi-squared distributions without Hartlap correction deviate noticeably from 900 but after applying the correction they fall much closer to the expected value of 900. Additionally, the posteriors do not peak at exactly the same location which can be attributed to the covariance matrices being noisy and thus skewing the results slightly. Even though they peak at the same location regardless of whether Hartlap is applied or not, the constraints get artificially small (particularly for the covariance matrix with the fewest samples) which is attributed to not applying the Hartlap correction, an issue that goes away once it is applied. Figures 12, 13, 14 show the posterior and chi-squared distributions when PCA compression is applied using either the analytical or 1500 realizations covariance matrix with Hartlap correction. Similar explanations exist for these results as the previous figures. The contours when using the numerical covariance disagree slightly from the analytic version due to their noise. Notably though the constraints get tighter the more PCA elements are added which intuitively makes sense since less information is lost. But the improvement largely levels off beyond a certain number of elements. We also notice as expected that the chi-squared distributions peak near the number of PCA elements since those are the number of degrees of freedom when the data is compressed. Figures 15, 16, 17 show the same results as figures 12-14 but instead when using a noiseless data vector. In this last example, we observe similar results but since the data is noiseless we find the posteriors peak at almost the exact same location without their peaks shifted. We also notice the chi-squared distributions instead peak close to 0. Both of these results can be attributed to the fact that the data is noiseless and thus there are effectively no degrees of freedom compared to the previous plots where the data was noisy and thus each entry contributed to the degrees of freedom. To summarize, each task in this assignment incorporated running MCMC chains and linking results to tasks performed in first two assignments related to covariance matrices and emulators/PCA compression.

3 Application to my Research

- **Fisher Information Forecasts** – The main focus of my project is to obtain Fisher information forecasts for primordial non-Gaussianity marginalized over the ΛCDM parameters, using a new clustering method. This process invokes concepts from the first two assignments. First, we generate the covariance matrix for the summary statistic using a large number of realizations of the fiducial cosmology. The inverse covariance matrix is biased (given the non-negligible size of the data vector relative to the number of simulations), meaning the Hartlap factor must be applied. The second ingredient in Fisher information are derivative estimates – we measure how the summary statistic varies under simulations where the cosmological parameters are varied slightly above/below the fiducial values. We then compute the Fisher matrix which is inverted to obtain a lower bound on the cosmological parameter uncertainties (via the Cramér–Rao bound).

- **DESI Mock Challenge** – Alternative Clustering Methods (ACM) is a topical group in the DESI collaboration studying statistics beyond the 2-point correlation function/power spectrum. I am participating in a mock challenge where each group member computes their summary statistic on a mock galaxy catalog at which point the accuracy and constraining power are compared and analyzed. This process invokes concepts from each of the three assignments throughout the course. First, we generate the covariance matrix for the summary statistic numerically using a large number of realizations of the fiducial cosmology; the computation of the inverse covariance matrix involves the Hartlap factor. Second, we train a emulator to predict the summary statistic as a function of cosmological parameters using a training set of summary statistics computed on simulations with different cosmological parameters. Finally, we run an MCMC chain on the mock galaxy catalog (the 'noisy' data vector) where our log likelihood includes the inverse covariance matrix and the difference between the noisy data vector and the emulator's prediction based on the location in parameter space.

4 Figures

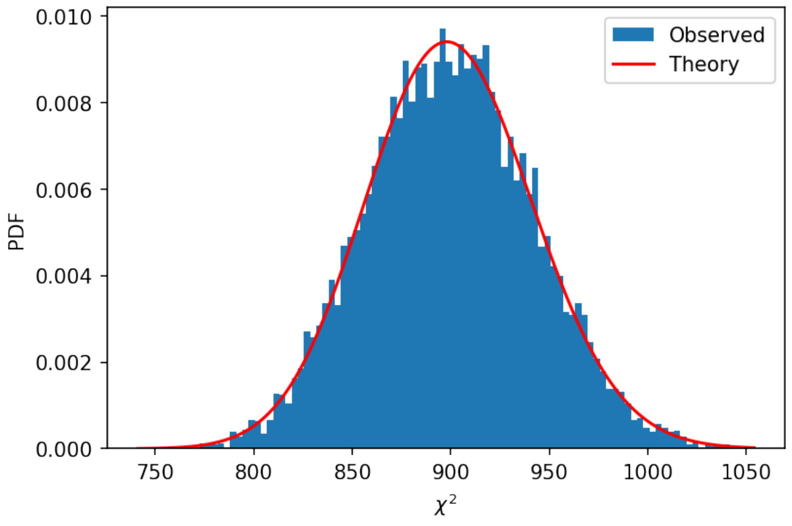

Figure 1: Chi-squared distribution when using the analytic covariance matrix (assignment 1).

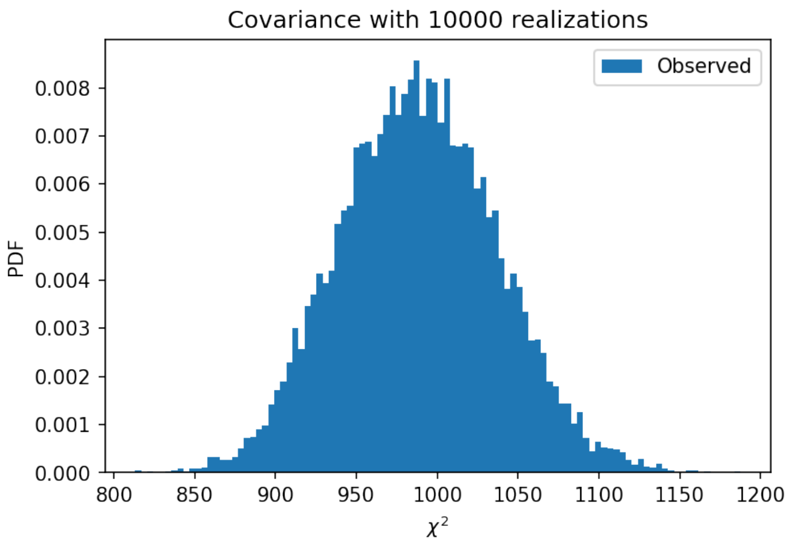

Figure 2: Chi-squared distribution when using the numerical covariance matrix with 10000 realizations and no Hartlap correction (assignment 1).

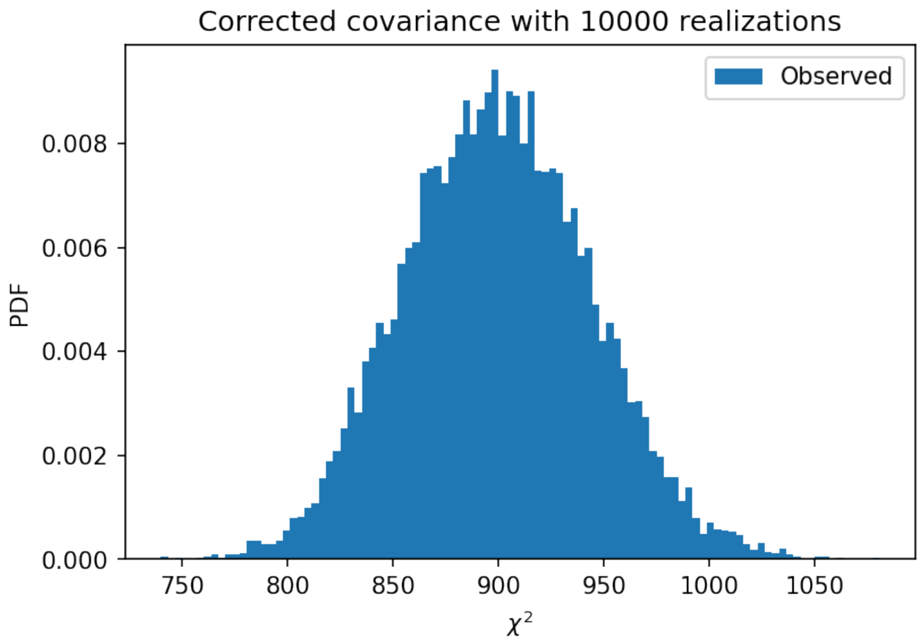

Figure 3: Chi-squared distribution for the numerical covariance matrix with 10000 realizations and Hartlap correction (assignment 1).

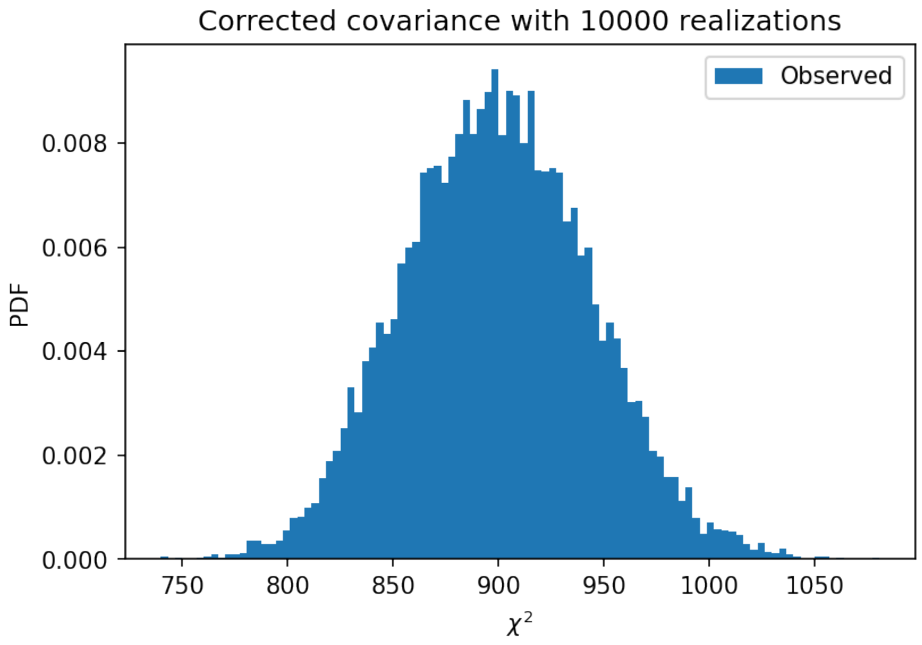

Figure 4: Chi-squared distribution when using the numerical covariance matrix with 10000 realizations and Hartlap correction, on the same samples used to generate the covariance matrix (assignment 1).

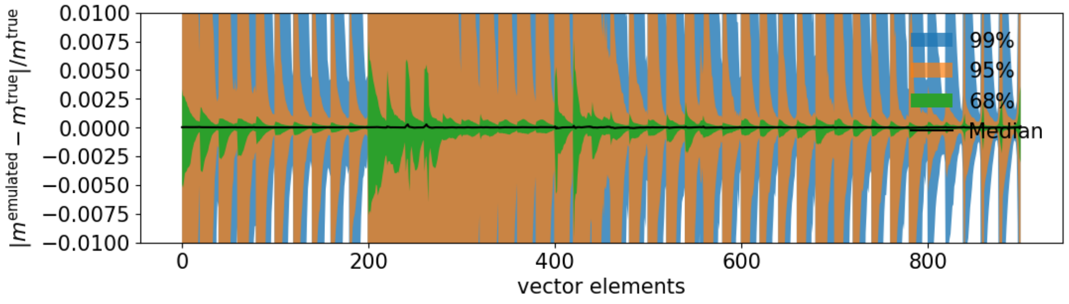

Figure 5: Percentiles for the discrepancy between the emulator predictions and the observed test set samples (assignment 2).

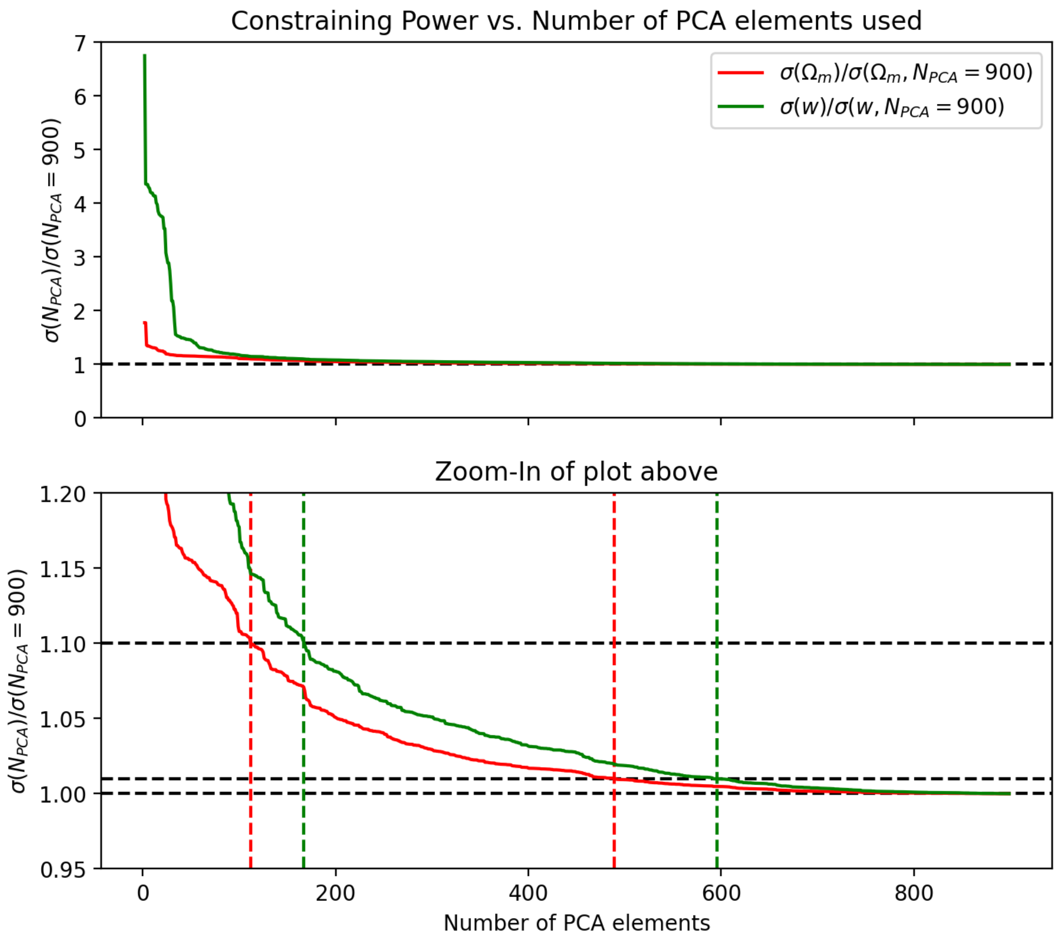

Figure 6: Fisher information constraints as a function of the number of PCA elements utilized, normalized relative to the constraining power when all the elements are used. The dashed horizontal lines correspond to the number of PCA elements required to fall within 10% or 5% of the maximal constraining power (assignment 2).

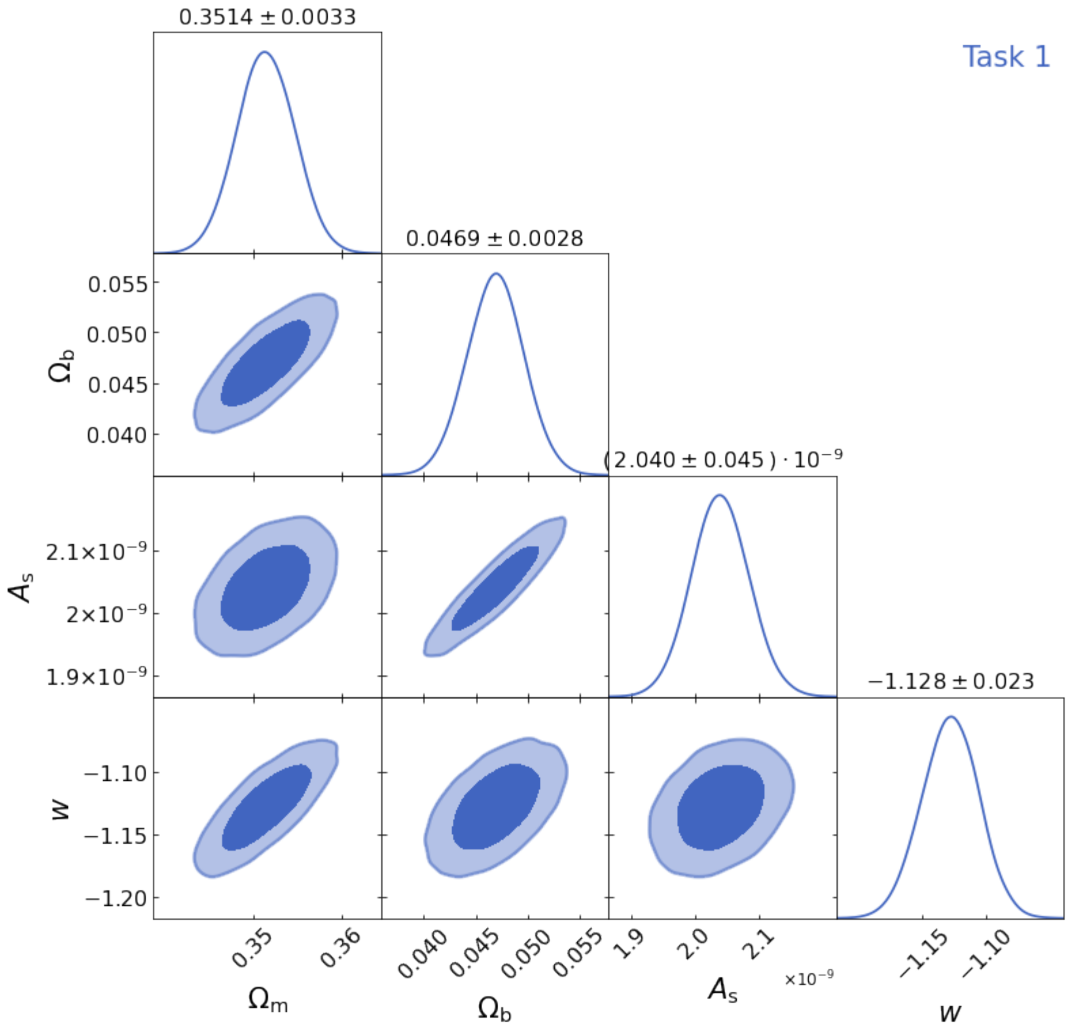

Figure 7: Posterior distribution for the analytic covariance matrix (assignment 3).

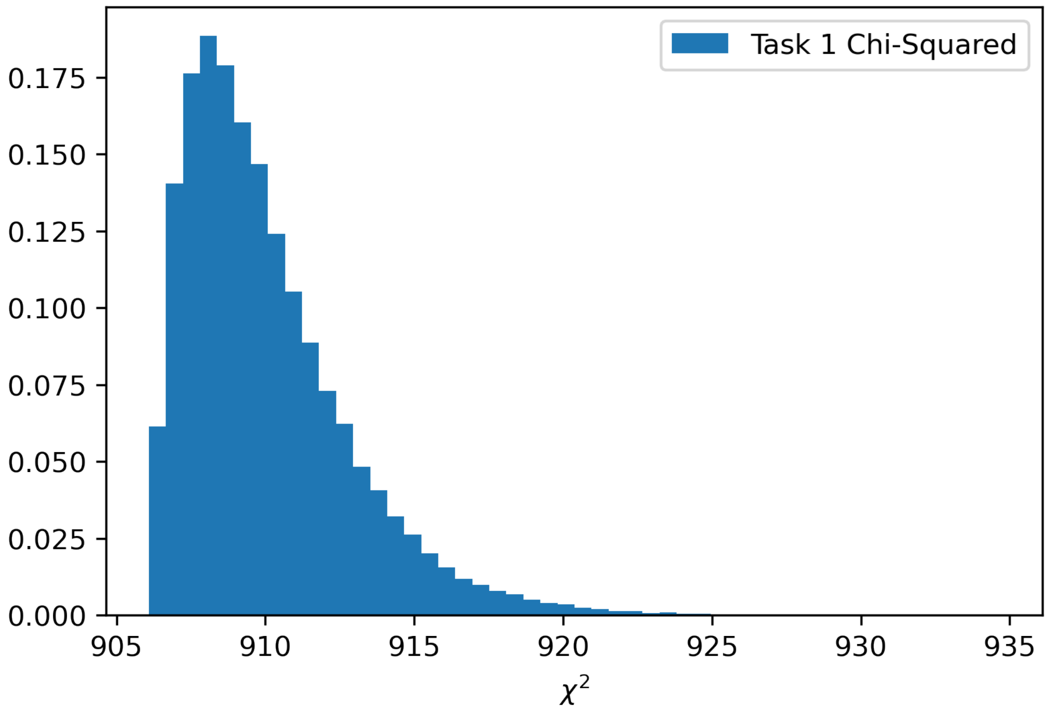

Figure 8: Chi-squared distribution for the analytic covariance matrix (assignment 3).

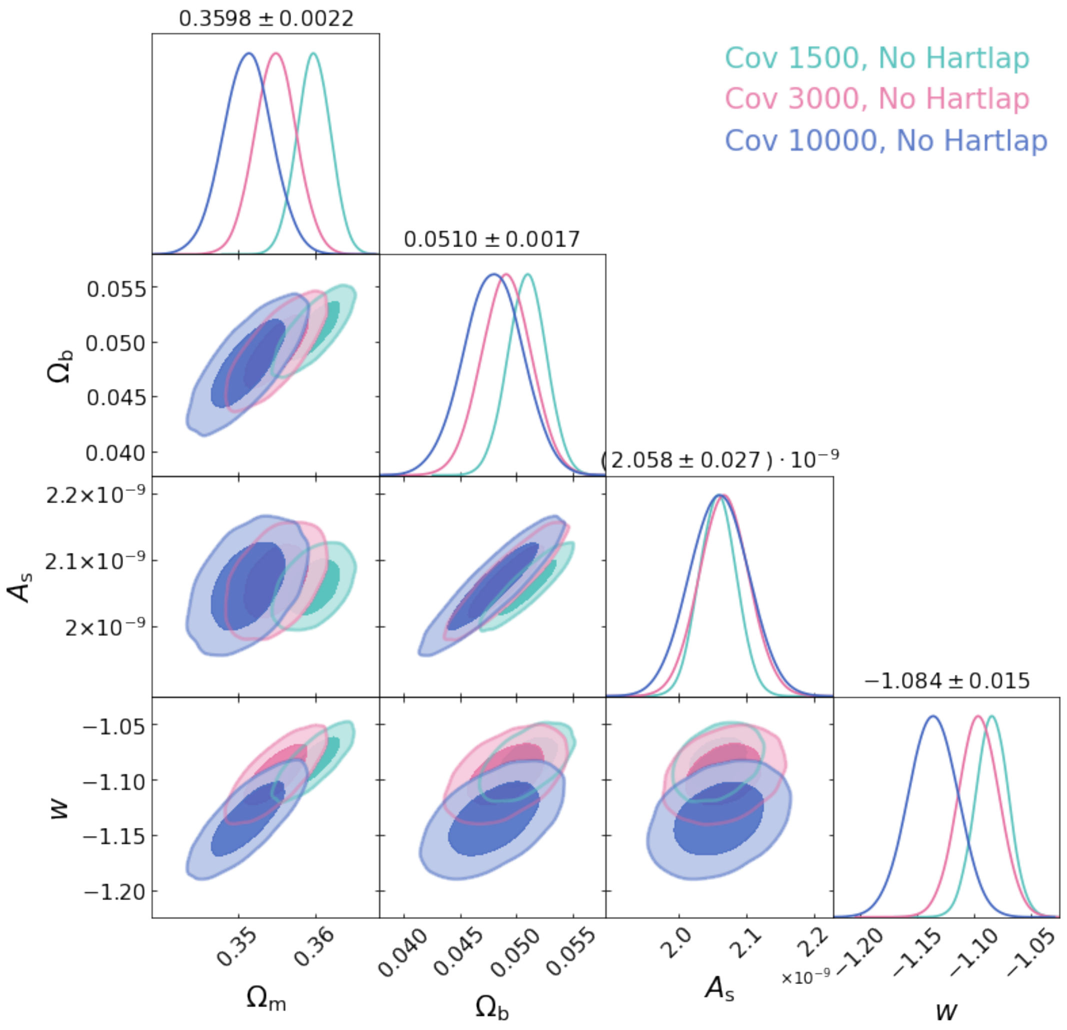

Figure 9: Posterior distribution for the numerical covariance matrix and no Hartlap correction (assignment 3).

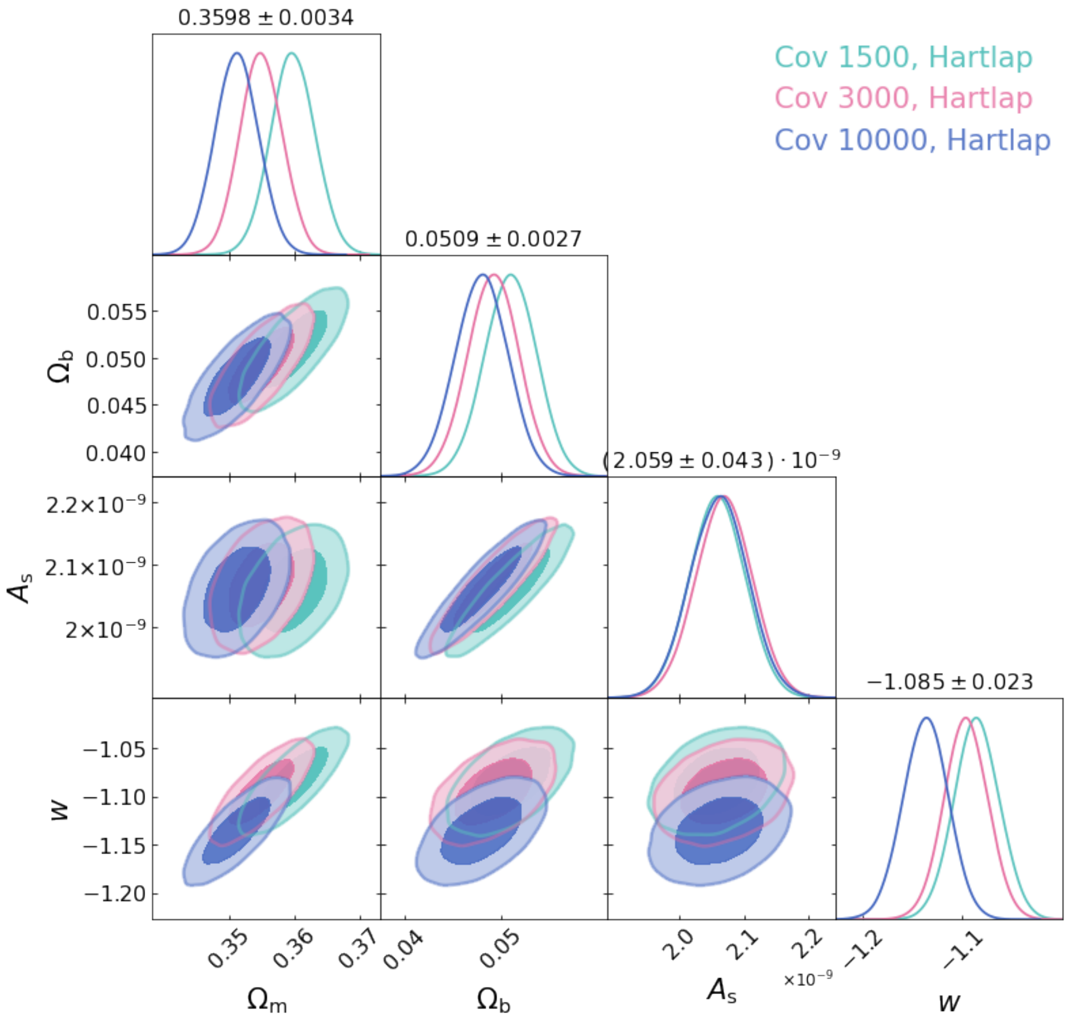

Figure 10: Posterior distribution for the numerical covariance matrix and Hartlap correction (assignment 3).

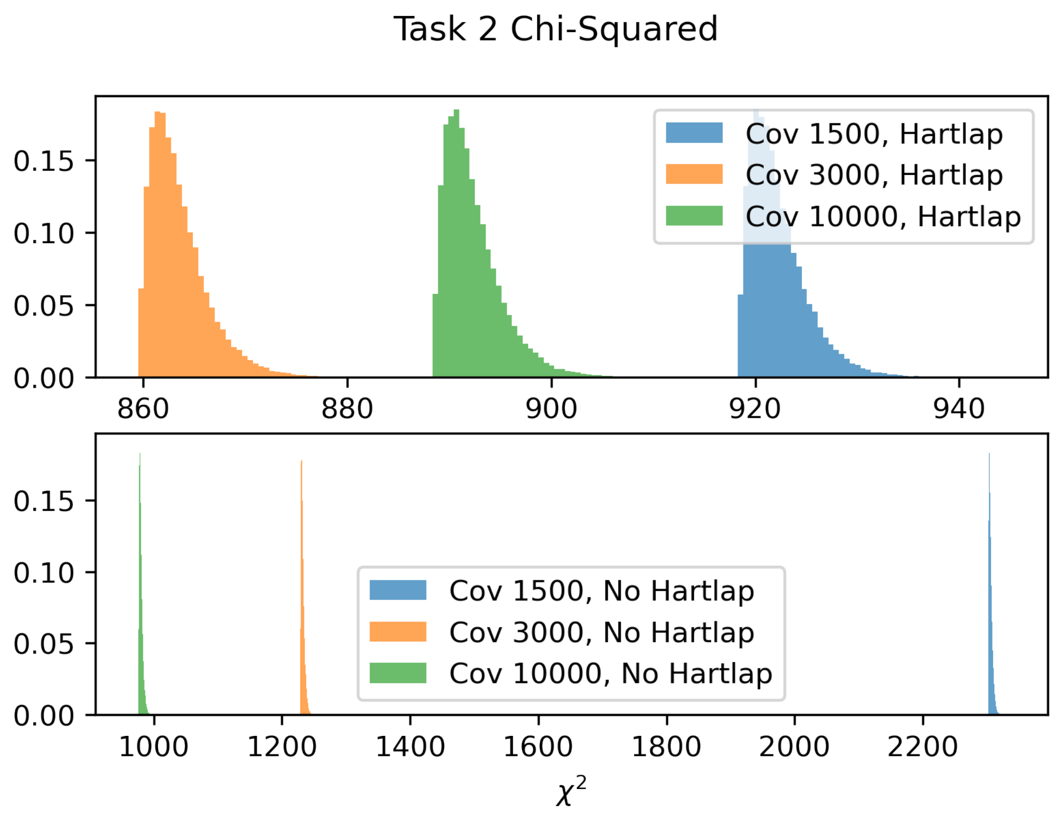

Figure 11: Chi-squared distributions for the numerical covariance matrix with/without Hartlap correction (assignment 3).

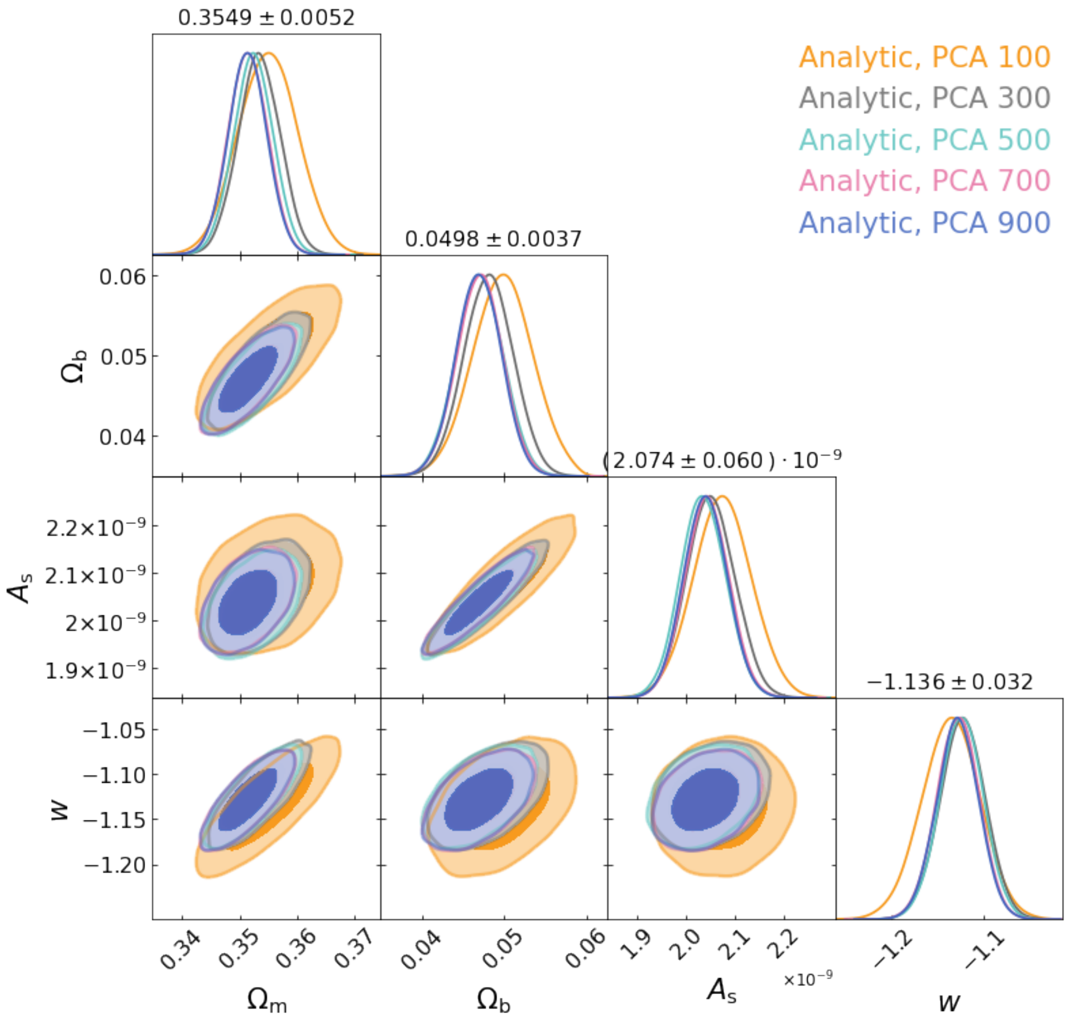

Figure 12: Posterior distribution for the PCA analytic covariance matrix (assignment 3).

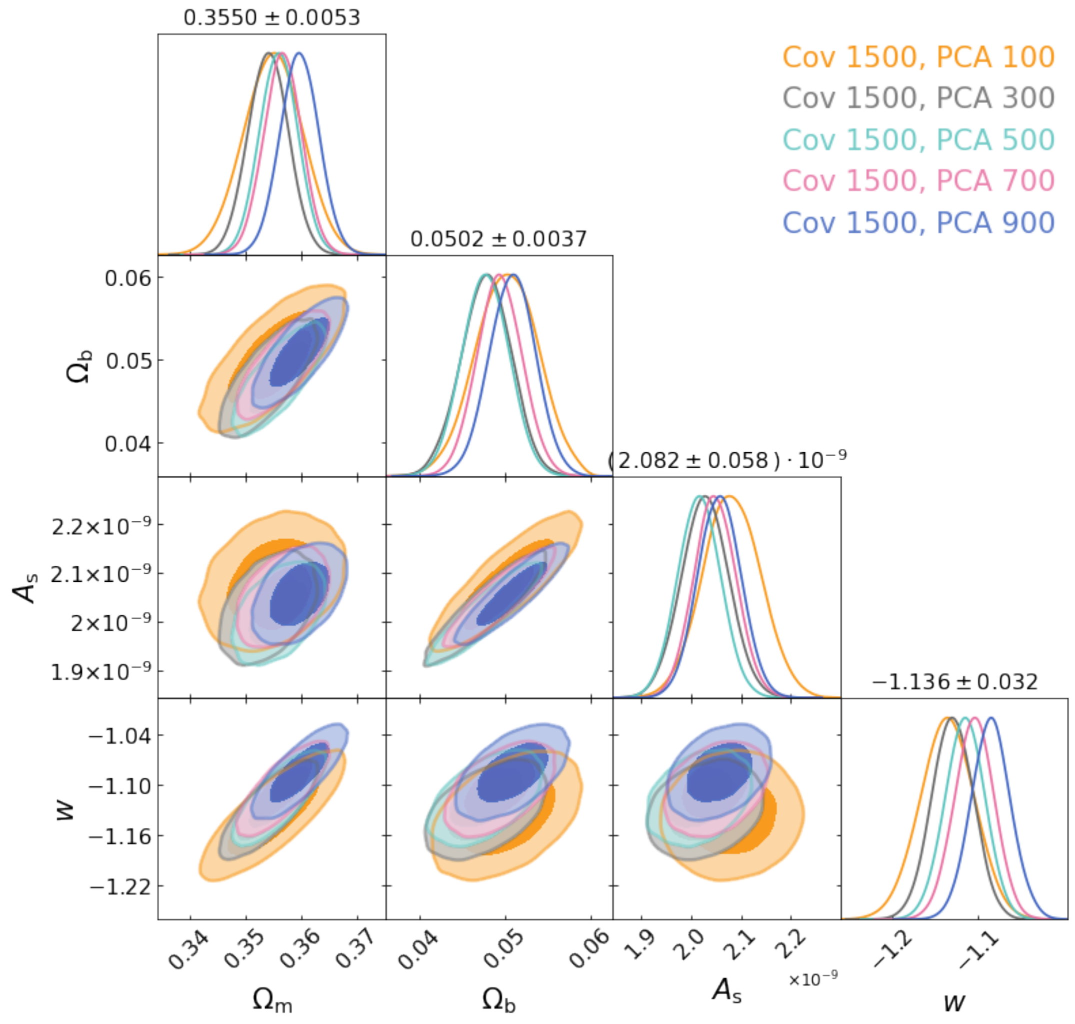

Figure 13: Posterior distribution for the PCA covariance matrix with 1500 realizations and Hartlap correction (assignment 3).

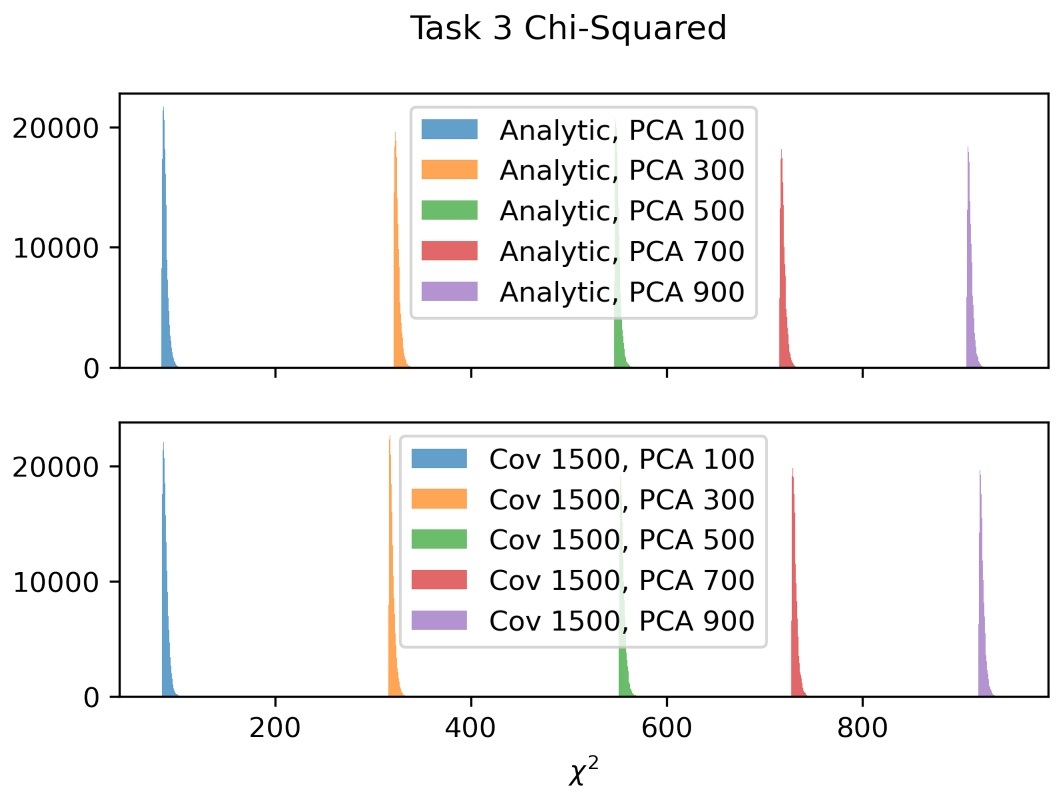

Figure 14: Chi-squared distribution for the PCA covariance matrix both analytic and 1500 realizations (assignment 3).

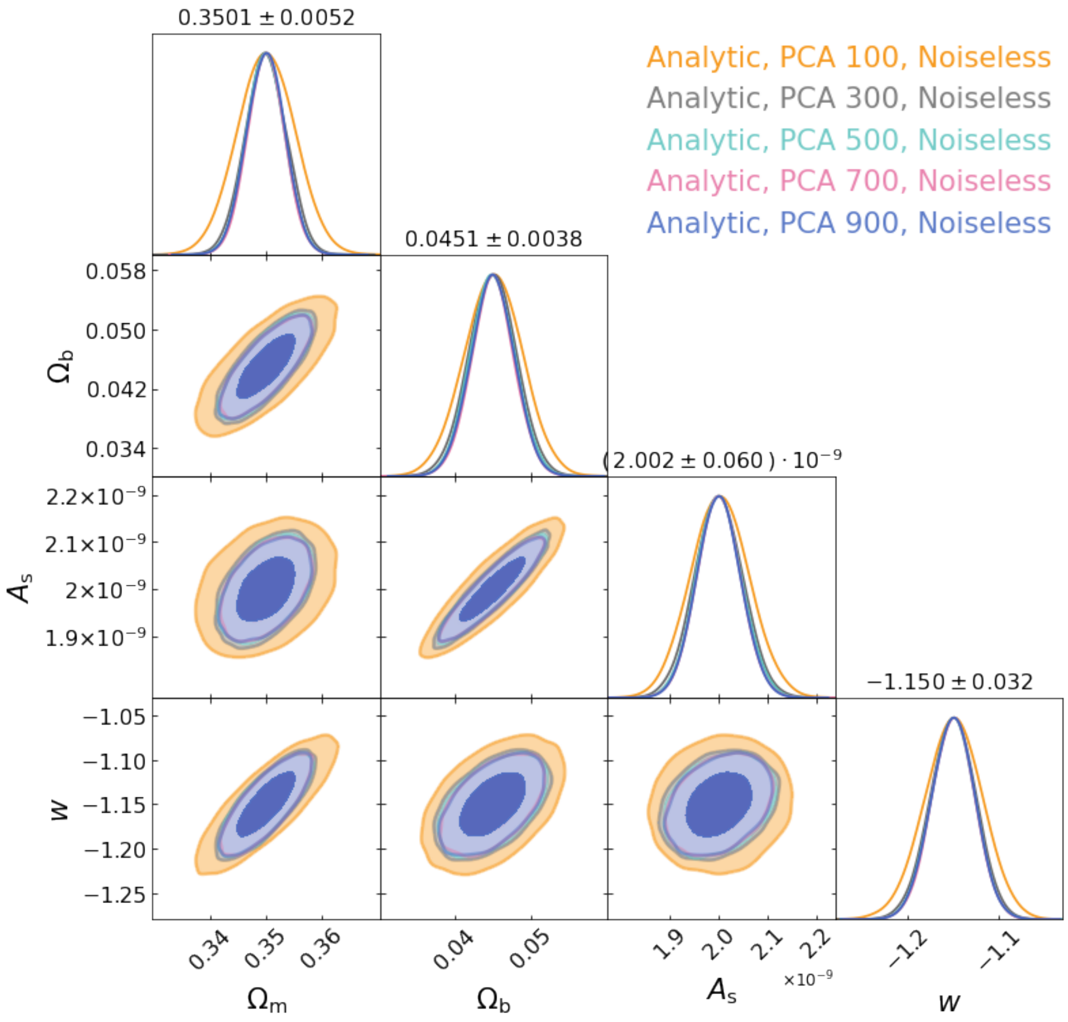

Figure 15: Posterior distribution for the PCA analytic covariance matrix, with noiseless data (assignment 3).

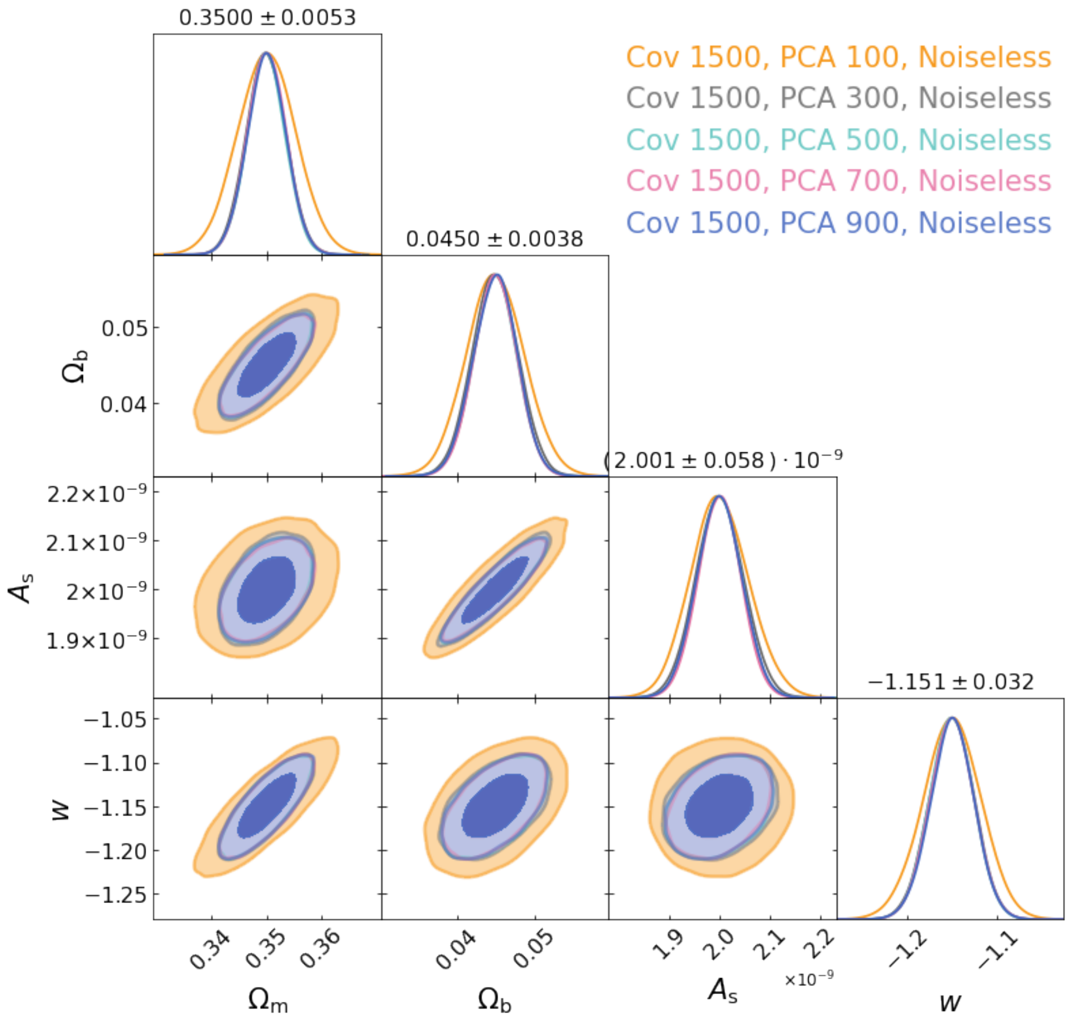

Figure 16: Posterior distribution for the PCA covariance matrix with 1500 realizations and Hartlap correction, with noiseless data (assignment 3).

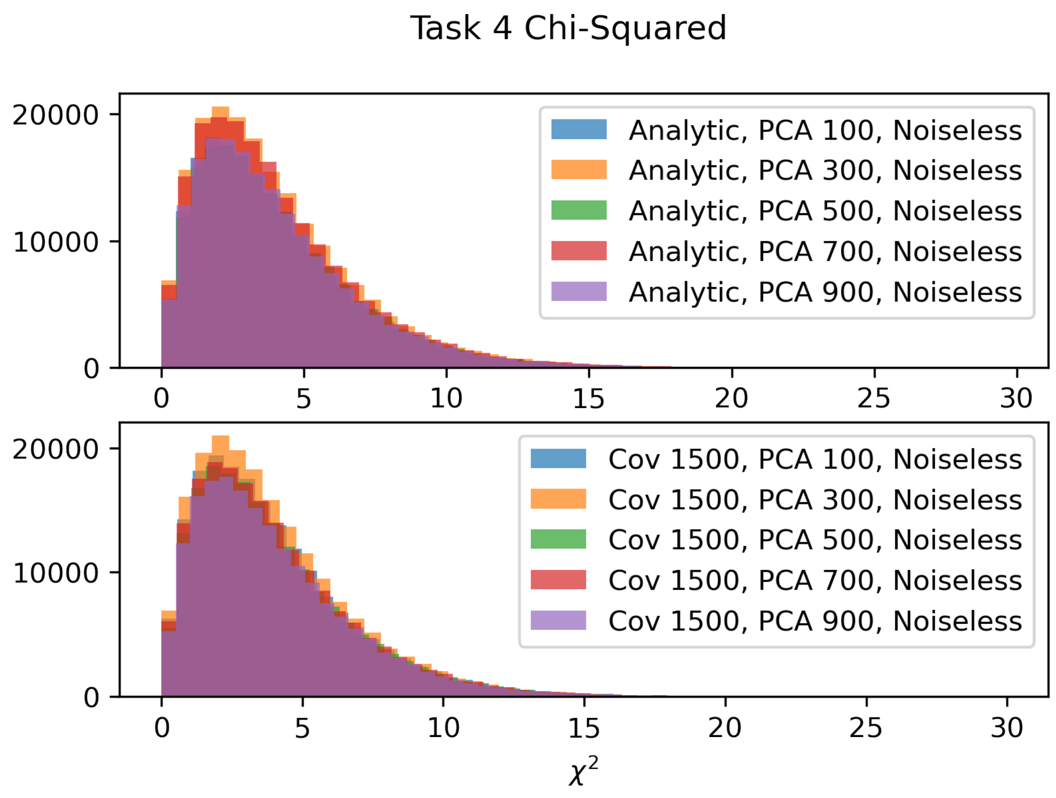

Figure 17: Chi-squared distribution for the PCA covariance matrix both analytic/1500 realizations, with noiseless data (assignment 3).