Prep Meeting 35

# Writing

**Trajectory of Chapter 2**

Did quite some writing, rewrote chapter 1 and chapter 2 based on your feedback. I think ready for first careful read? Biggest difficulty: when to stop Chapter 2. The trajectory now is: structure learning 🡪 density estimation of P(X) 🡪 assumptions on data, e.g. time-independent factorization where X\_i only depends on its parent set 🡪 assumption of a linear model (SEM) 🡪 continue back to time-dependent factorization 🡪 assumption of a linear model (VAR) 🡪 goal: learn fitting models such that they remain acyclic.

**When to stop Chapter 2**

From there, I stopped, as the learning part differs for each of the methods. E.g., score-based learning methods define a scoring function to be maximized, while explicitly adhering to the acyclicity constraint. Continuous optimization methods such as NO TEARS define a cost function that penalizes cyclicity, thereby enforcing acyclicity through the scoring / cost function. Constraint-based approaches use conditional independence tests to learn the edges. Therefore, the learning part differs too much, and I thought that the learning part should be done in Chapter 3, Related Work, where we can define:

* The methods of learning
* For each method, how some algorithms use this method.

**Formatting of Chapter 3.**

Quite difficult to form Chapter 3. We do not want to go into too much detail already and thereby make it too difficult for the reader to continue. We want to provide just enough information so that the reader understands it, but is tempted to continue to the relevant chapters to learn more about these.

The formatting of Chapter 3 *should be in line with the formatting of the developed methods*. I think there are two options:

Types of learning:

* Constraint-Based: PC-Algorithm.
* Score-Based Learning:
  + Order-Based Learning.
  + Continuous Optimization
* Noise Structure: LiNGAM.

Also, a question: Should we include methods that were interesting, yet their direction was not pursued? For example, constraint-based methods such as PC algorithm are interesting and very related, but require quite some causal lingo and arc orientation rules.

Furthermore, subjects like Granger Causality and methods that investigate the noise structure, such as LiNGAM are certainly interesting, but we did not investigate those.

So, should only methods be included with a *direct* relation to our methods, or can we devote some effort into writing up a nice selection of interesting algorithms? Is it worth putting in the effort there?

**Formatting of methods.**

Perhaps better to decompose the methods such that each *method type* has its own chapter, combined with its own model selection / regularization procedures. Several options:

*Go from exact to approximate:*

* First discuss the exhaustive order-based approach, then the random walks, then the metropolis-hastings algorithm. The first one is exact, but due to scalability, the others become approximate. However, they will also not scale very well. Regularization can be done just as NO TEAR by using an l\_1 penalty (LASSO) or an information criterion such as BIC, or by bootstrapping.
* After this, go to “faster” methods, such as continuous optimization methods such as NO TEARS. These do not require any regularization / model selection, etc.
* After this, go to the “fastest” methods such as greedy algorithms. Examples are the nine methods: forward / backward / backward-violations first, based on MSE / residual correlation / coefficient size. To regularize, we can do the same as for the order-based approaches: l1 regularization / BIC / bootstrapping. However, the greedy nature gives a nice order of importance of the edges, over which we can do our novel *cross-validation*. Also discuss there the AR(1) cross validation analysis.

*More on type of method*:

* Order-Based search methods consists of finding a suitable ordering. Acyclicty is implicitly enforced due to the permutation matrix and upper triangle U. Quite some sources here,
* Continuous optimization methods. Acyclicity is enforced in the cost function, e.g. h(W) = 0 or we only terminate when lambda is so large that we have a directed acyclic graph. Examples here are *NO TEARS, DYNOTEARS*, and graphical *LASSO*. Furthermore, we also have the *Birkhoff Polytope*.
* Iterative methods, where iteratively, one arc is removed or added (based on the direction) that maximizes some deletion / selection criterion. Examples of developed methods are *greedy equivalence search,* and *max-min hill-climbing*, and *greedy DAG search*.

Both approaches yield I think almost the same formatting, but the latter seems more nicely organized, and also links better to the Related Work.

**Evaluation of Methods**.

Now that the methods are a bit better organized, the question is how we want to evaluate them. We can evaluate on both models described in Chapter 2, SEMs and VAR(1) models. Subsequently, we can evaluate them on real (SEM / VAR(1)) data, or on simulated (SEM / VAR(1)) data. However, the question is to what extent we want to do an extensive evaluation? What would be the added gain?

**Conclusion**

Important chapter, I think a summary and consequent conclusion are good to make. Also want to extensively highlight the limitations of our model, and point into future directions.

# Assessment Committee

Is Jacques kind enough to be the third assessment committee member? If so, I can sent the e-mail to the assessment committee right away.

Mailed around, asked some friends, and had the following clarification:

“The assessment committee referred to in section 1 consists of at least three voting members, one of whom is the graduation supervisor. The voting members together determine the final mark for the master project. The members of the Assessment Committee are appointed as examiners by the Examination Committee according to article 7.12, section 3 of the Higher Education and Research Act.”

“At least two of the voting members are an assistant professor, associate professor or full professor within the Department of Mathematics and Computer Science according the division outlined in article 4.2 and are independent from each other, understood as belonging to a different cluster.”

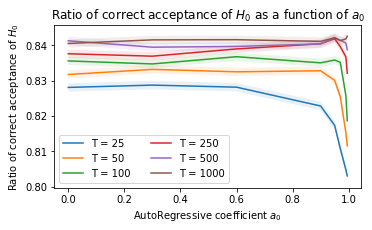
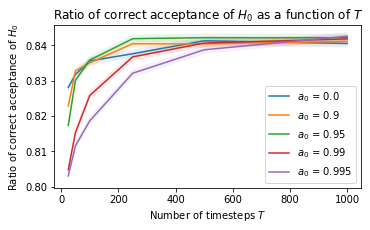
“So, hypothetically, I am doing a double degree project in IAM and CSE (track in DSiE). I have one graduation supervisor from Statistics. We invite another assistant professor from the Operations Research cluster within IAM and another Postdoc from Statistics. This would satisfy all requirements as I have read them in the .pdf, correct?”

“Yes you have interpreted this correctly!”

Conclusion: Even “just” the assessment committee consisting of Rui, Jacques, and Alex would suffice.

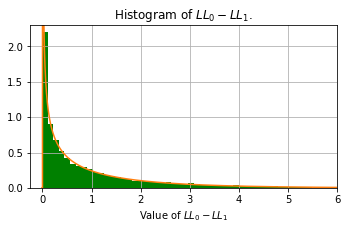
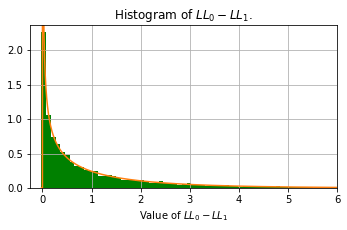
# Using Different Noise

Now using **Laplace noise**, with parameter *mu* *= 0* to get zero-mean noise, and parameter *lambda / b* = ½ sqrt(2) such that the variance is 2b^2 = 1. We first investigate the probability of correct acceptance. It seems not to differ significantly, also converging to the same value.

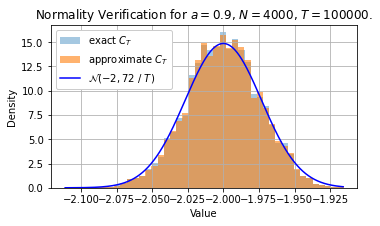
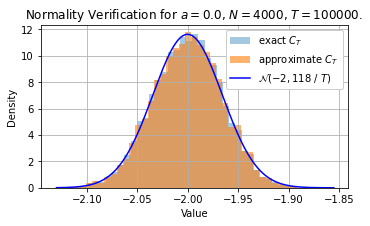


**Laplace noise with unit variance: Seems same as normal noise.**

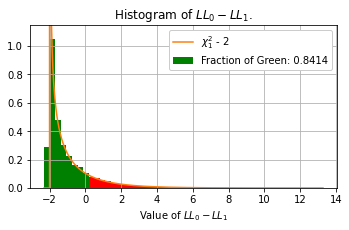
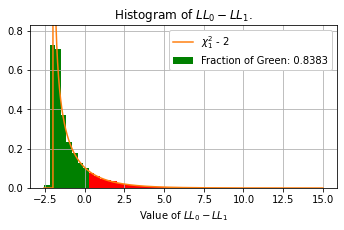
Let us first investigate LL\_0 – LL\_1. Previously, this had a chi squared distribution with one degree of freedom. Now, let us consider the *Laplace noise* version for *a = 0.0, T = N = 10,000 (left)*. Furthermore, let us also consider *a = 0.9, T = N = 10,000 (right)*.



We see that the chi-squared-ness of LL\_0 – LL\_1 still holds, so Wilk’s theorem still applies.

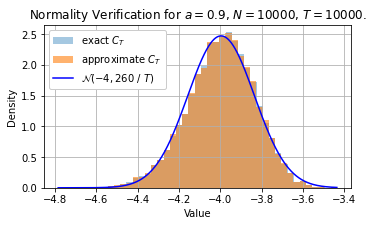
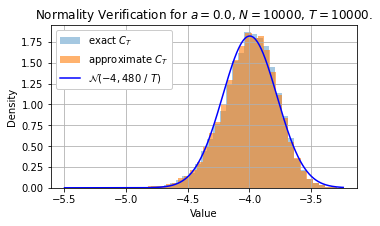
Now, let us consider the value for *CT* and its approximate value for *CT*

We see that the mean of -2 still applies. However, the variance differs slightly. It is slightly wider, but the variance still converges to zero as T tends to infinity. For *a* larger, we still see that this is still the case, and even the constant seems to be slightly smaller. Nevertheless, the main result still applies, as we see in the figures below (*left: a = 0.0*, *right*: *a = 0.9*) for *T = N = 10,000*.



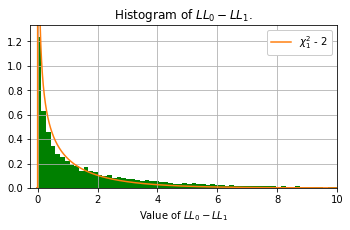
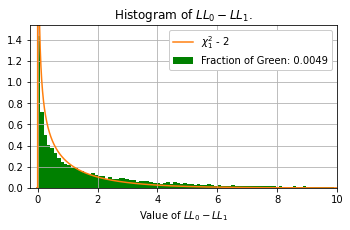
**In conclusion: It seems that different noise values do not change the scenario here, at least for when E[epsilon] = 0, E[epsilon^2] = 1.**

**Laplace noise with variance of two: Rescaling required.**

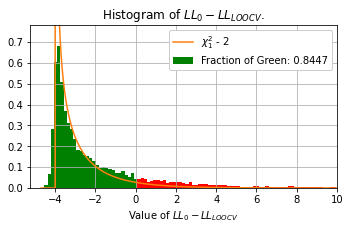
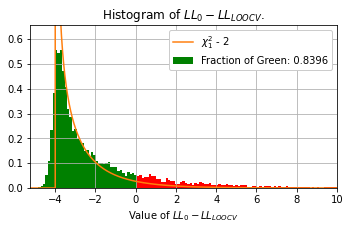
****

The mean has now doubled, and the variance has approximately quadrupled, as expected.

Interestingly, the LL\_0 – LL\_1 is now no longer chisquared. It seems as if there is not enough density on [0, 2], and too much density on [2, 6]. This applies for both *a = 0.0 (left)* and *a = 0.9* *(right)*.



Now the shifted plots are not that different, again too much weight on [-2, …], and not enough weight on [-4, 2]. Perhaps we need more samples than 10,000? Did also *50,000*, but that did not change anything.

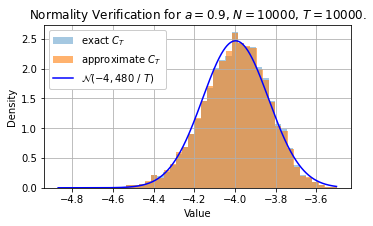
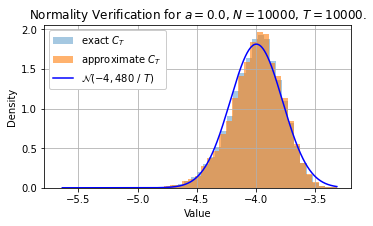


Interestingly, changing the Laplace parameter also affects the chi-squaredness of the distribution of LL\_0 – LL\_1. Interestingly, the fraction of green remains approximately 0.84 though.

**Laplace noise with variance 2: Rescaling.**

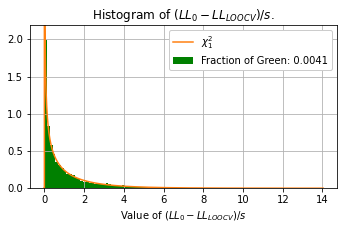
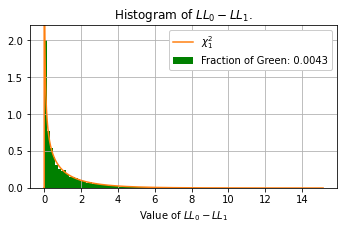
Interestingly, if we rescale, we **do** get the chi-squared distribution. If we zero-mean and unit variance the LL\_0 – LL\_1, we get a nice chi-squared distribution. Let us now consider a value for *b* such that the covariance is equal to 2, so b = (0.5 s)^(1/2).

The value for C\_T will approximately have mean -2 \* s = -4.



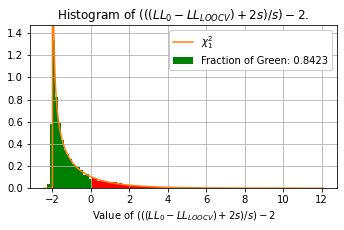
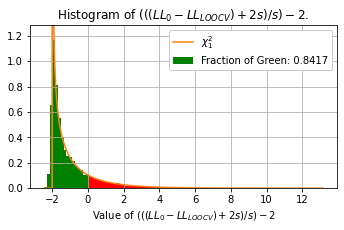
*a = 0.0 a = 0.9*

*Standardizing LL\_0 – LL\_1: (LL\_0 – LL\_1) / s*



*a = 0.0 a = 0.9*

*Standardizing LL\_0 – LL\_{LOOCV}:* ((LL\_0 – LL\_{LOOCV}) – 2 \* s) / s)

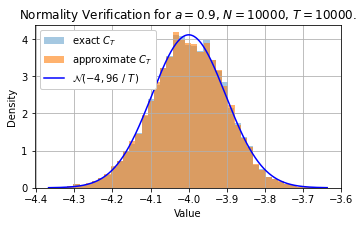
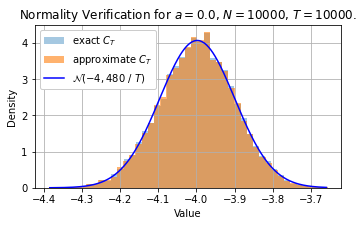


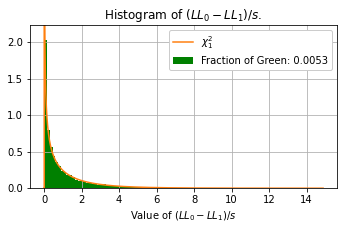
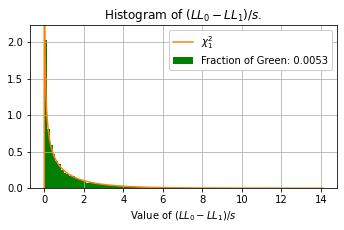
*a = 0.0 a = 0.9*

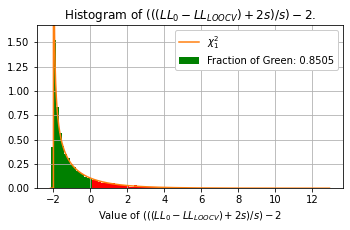
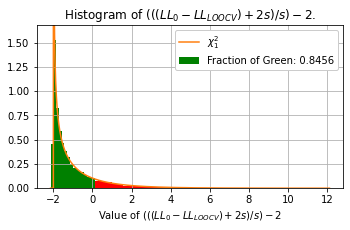
Interestingly enough, the probability is still 0.84.

**Change Normal distribution with variance of 2. Get same problem? Again, same solution.**

Let us now consider normal noise with a variance of 2, do we again need rescaling? Seems like we do, which is rather strange really.





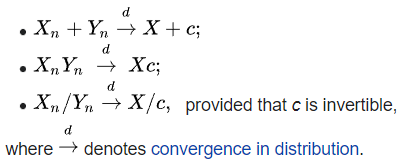


**Conclusion: For unit variance, nothing changes really. However, when the noise variance is no longer unitary, we need to rescale the log-likelihoods to get the chi squared distribution shape. Interestingly, this does not affect the probability of success, as the zero point does not change.**

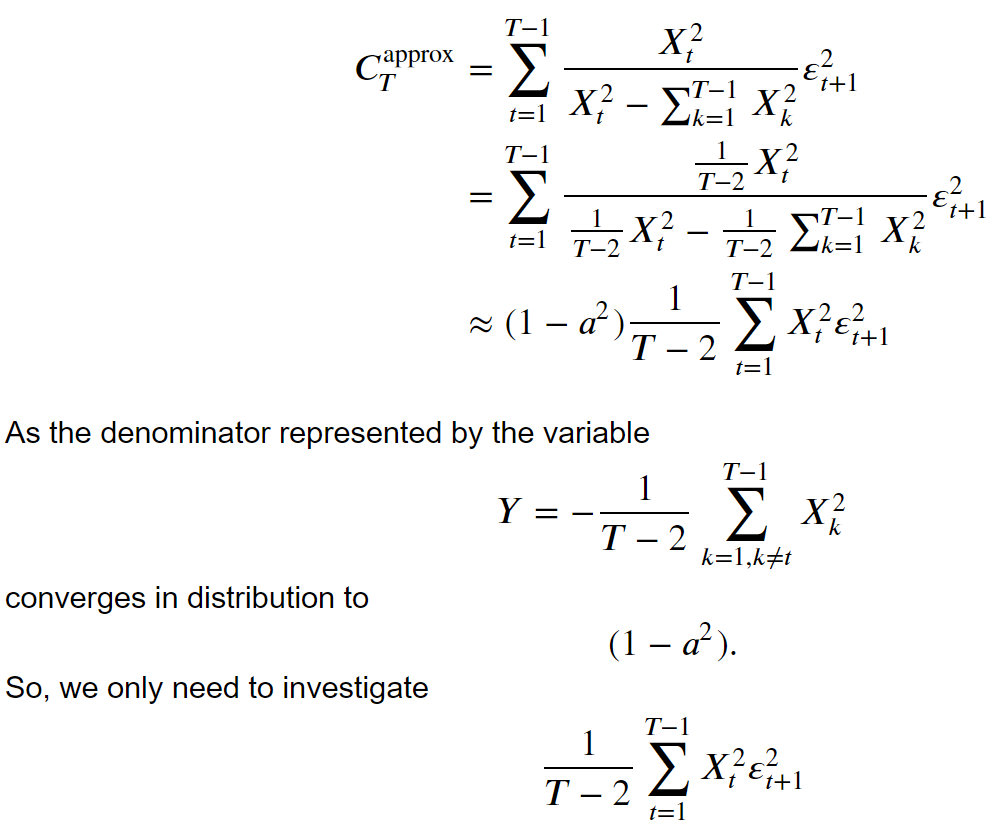
**However, it seems strange that for e.g. the normal distribution, this rescaling needs to be done; would Wilk’s theorem not have that this is chi squared distributed already? Why is this rescaling needed?**

# Further Derivation using Slutsky’s Theorem

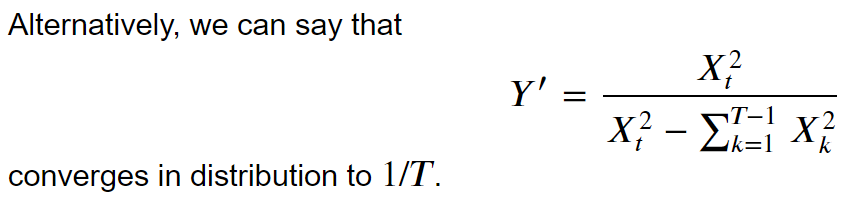
Put the 1 / (Xt^2 – sum X\_k^2) out of it, and sort of treat it as a constant. We have case 3:



**Slutsky v1**. Let us rewrite our approximate *CT* as

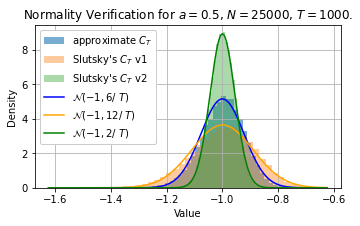
.

**Slutsky v2**



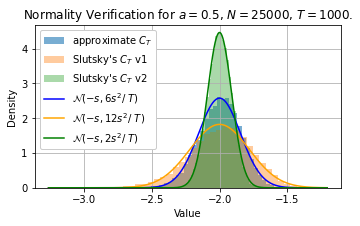
Slutsky’s v2 then corresponds to a sum of i.i.d. scaled random normal variables with mean V(epsilon).

Low, let is compare Slutsky’s to our covariance:



They are not the same, because for T tending to infinity, both converge to a point with mean -1 and covariance 0. Interestingly, Slutsky’s v1 is **wider** than the approximate C\_T, by a factor of 2 approximately. However, Slutsky’s v2 is **narrower** than the approximate C\_T, by a factor of 3 interestingly. Nevertheless, all have the same mean.

Also, for different values of the noise component, slutsky’s remains adequate, although there is a difference in covariance again. Here, we used **s = 2**, so that the variance of epsilon now is s^2 = 4, rather than 1^2 = 1.



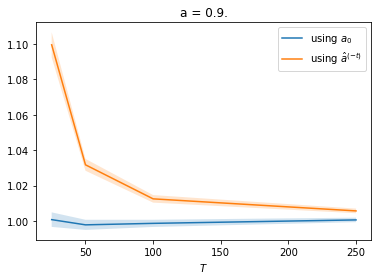
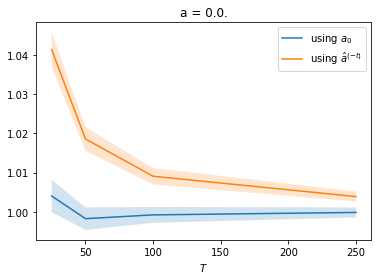
# Smaller probability of success when *a close to 1, is it problematic?*

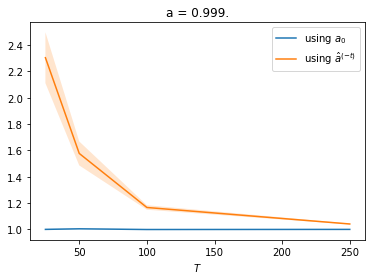
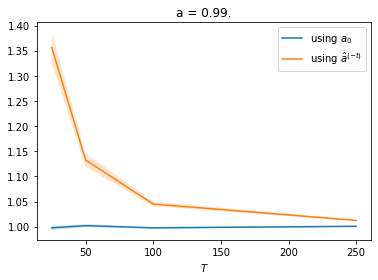
We saw that the probability of success was slightly lower when a was close to 1. However, Rui mentioned that this might not be problematic.

**Setting:**

We generate X using an AR(1) model with coefficient *a* for *T* timesteps. We compute a-t. Now, we generate X2 the same as X. We compare the score of using *a* to the average score of using *a-t*. We try this for varying values of *a* and *T*, and try each pair (*a, T*) 1,000 times and compare their means with accompanying standard errors.

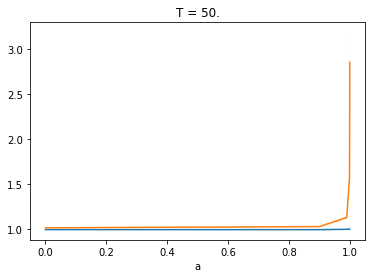
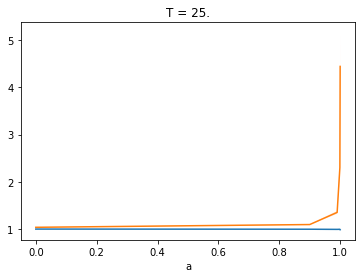
**Varying values of *a* as a value of *T* on *X2.***

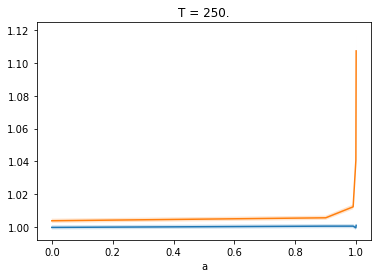
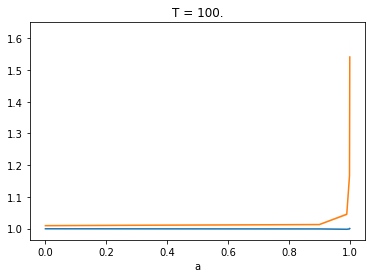




We see that all the time, using *a0* is by far the better option. Furthermore, this difference is greater when *T* is small, and when *a* is closer to one. If *T* is smaller / *a* is closer to one, then the difference between using the LOOCV estimate and the original *a* increases quite sharply. If we consider this for varying values of *T*, this is also the case. We see a huge increase as *a* gets closer to one.

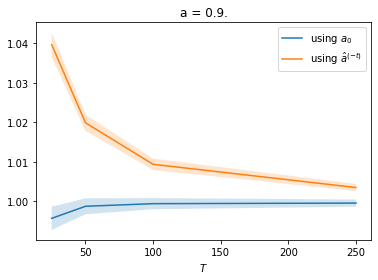
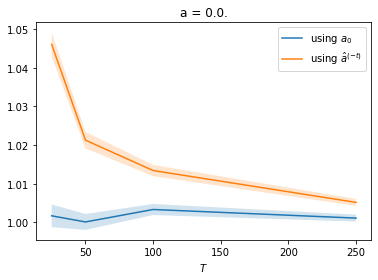
**Varying values of *T* as a function of *a* on *X2*.**

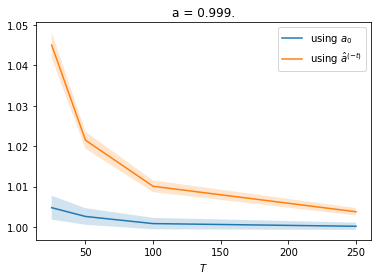
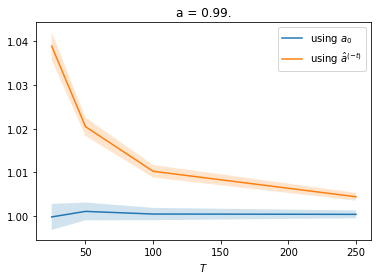




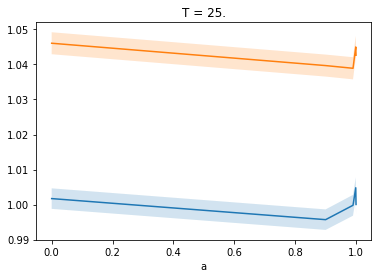
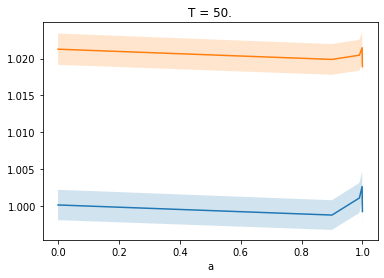
So, the problem seems to be even more dramatic for larger *a* and for smaller *T*. Now, let us consider the error on the original data. We now expect the LOOCV estimate to be better, as it “overfits” on the data.

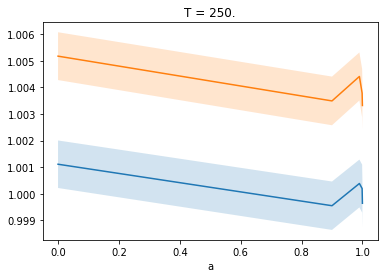
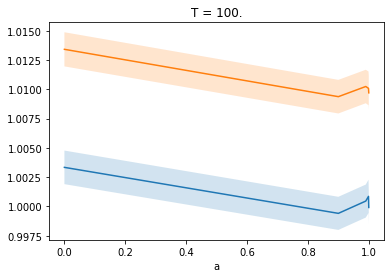
**Varying values of *a* as a function of *T* on *X1*.**





**Varying values of *T* as a function of *a* on *X1*.**

****



We can also increase the length of X2, say 10 \* T rather than T. This only increases the discrepancy between the two. We can also use the expected value, so for T2 -> infinity, which again only increases the discrepancy even further.

# Writing

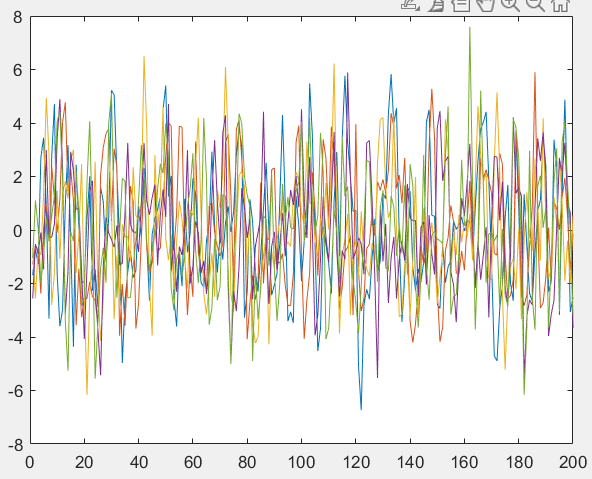
Continued quite a bit on the writing, many thanks for the feedback. First version of Introduction is done now, incorporated all your feedback.

At the moment, working on improving Chapter 2 to make it less dry. Also have some things in minds for the previous work section, but remains to be seen how extensive each part should be, e.g. write out algorithms all that explicitly, etc.

# FMRI Dataset

Recovered and downloaded the FMRI dataset of the Survey that Alex send. Note that it is simulated data, not real-life data, yet they claim it is realistic. However, they also state that “lag-based approaches perform very poorly.”

Example of one of the 28 time series:



# Twitter Dataset

After long searching, as there are many different datasets looking like this, I managed to find the greater 2016 presidential election dataset of Garvesh, and managed to extract the tweets of all presidential candidates. For democrats, there are:

* Bernie Sanders, Hilary Clinton, Tim Kaine, Martin O’Malley.

For the republicans, there are:

* Rick Santorum, John Kasich, Gov. Mike Huckabee, Ted Cruz, Rand Paul, Governor Mike Pence, Mike Pence, Governor Christie, Jeb Bush, Marco Rubio, Jim Gilmore, Carly Fiorina, Ben & Candy Carson, Chris Christie.

Interestingly Donald Trump does not seem be the there, due to the fact that his account has been deleted. Are ways to retrieve it without Twitter.