Research Comp: Paul

Paul Harmon

3/26/2019

Logistical details:

- Resources: For this exam you may use any textbooks or course materials. You are also free to use any publicly available materials that you might come across in a research setting; however, all writing must be original and all reference materials must be cited.
- Research Articles: PDF copies of all research articles mentioned in the questions will be provided.
- Length of answers: Suggested page lengths are provided for some questions, note these refer to default settings (12 point font with double spacing). These page lengths are approximate and for your guidance; deviations in answer length are acceptable.
- Computer Code and Reproducibility: Please turn in all relevant computer code to reproduce your results.
- Time: This part of the exam is meant to take two days - approximately 16 hours to complete.
- Advice: Be sure to adequately justify your answers and appropriately reference any sources used. Try to focus your answer less on technical details and more on showing your ability to think critically, put ideas together, and clearly articulate your thoughts in a limited amount of time. Contact Mark Greenwood (greenwood@montana.edu) with any questions.

Part I: Penalized EFA and SEM

1) **Describe "regularization" in general. Then discuss a situation or two where regularization might be useful and which types of regularization might be considered in those situations and why. [1 page]**

Consider the standard linear model $Y = \mathbf{X}\beta$. Under a certain set of assumptions in the Gauss-Markhov theorem, the Least-Squares estimator from such a model is a best-linear-unbiased-estimator, where "best" means that the OLS estimator has the lowest variance among unbiased estimators. A common theme in statistics is the variance-bias tradeoff, by which one might accept the use of a biased estimator to substantially reduce the variance in that estimate. An entire field of statistics: Bayesian statistics, is built on this concept – informative prior distributions on parameters typically "shrink" (i.e. bias) an estimate but tend can produce less model error.

Moreover, in any setting involving multiple features, there is likely to be shared information between those features. Multicollinearity, then is a problem that plagues even moderate-dimensional problems. As noted by Kuhn and Johnson (2013), "Combatting collinearity by using biased models may result in regression models where the overall MSE is competitive with" and, quite frankly, substantially better than, unbiased OLS estimates.

Regularization, then, is a technique by which a bias is imposed by adding a penalty to the objective function of a statistical model that controls for model complexity. In a regression setting, ridge and lasso techniques add a l2 or l1-norm penalty, respectively, to the SSE calculation to be minimized, biasing the OLS coefficients by shrinking them towards 0 (or, in the case of the lasso, all the way to 0). Hoff (2009) illustrates that a Bayesian model does this in an even more intuitive fashion – the prior distribution pulls the posterior parameter expectation away from the data value (sample statistic) and towards the prior parameter (like μ). However, the discussion of Yamamoto et al (2016) indicates that there are meaningful differences between the Bayesian treatment and regularization via non-convex optimizers such as MC+, so it may not be possible to directly compare a frequentist and Bayesian version of regularized models.

Several situations arise where regularization might be useful. Consider a clustering problem involving pea plants that have been measured at 300 genetic markers. It might be useful to consider regularizing the number of variables used in the clustering with a lasso penalty so that, say, only 15-20 features are used to create clusters. In a regression model to predict house prices, it may make sense to take advantage of the shared information between collinear features without explicitly removing them from the model, so a ridge penalty might make sense. Finally, in functional data analysis, regularization can be used to smooth an estimated spline by adding a roughness penalty that allows the smoother to not hit every data point (Ramsay and Silverman, 2005).

2)**Write out the typical EFA and CFA models using notation either in one of the following papers or that we used in STAT 537, defining the components in them. Discuss the differences between EFA and CFA. When would you use one versus the other? You can use an example from your own work or another example. Try to motivate this with a data set that you can use for later questions that both EFA and CFA could be considered even if one is more likely to be of interest than the other. The data can be simulated or real. [1 page plus figures/tables]**

The typical Exploratory Factor Analysis and Confirmatory Factor Analysis models are written out and described below. I technically use the notation of Hirose and Yamamoto (2013) but it is roughly the same as that found in Everitt and Hothorn (2010). The model is given as:

$$X_{px1} = \mu_{px1} + \Lambda_{pxm} F_{mx1} + \epsilon_{px1}$$

In the above, X is a px1 vector of values, μ is a px1 vector of means with some variance-covariance matrix Σ. F is a vector of loadings and Λ is a pxm matrix of factor loadings. To be clear, the loading $\lambda_{i,j}$ is the loading of the i^{th} observation onto the j^{th} latent factor. Finally, we assume that Φ_{mxm} and Ψ_{pxp} are the variance-covariance matrices of F and ϵ, respectively. Factor models are made clearer in matrix notation, so I'll show that below:

$$\begin{bmatrix} x_1 \\ \dots \\ x_p \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \dots \\ \mu_p \end{bmatrix} + \begin{bmatrix} \lambda_{1,1} & & \lambda_{1,m} \\ \dots & & \\ \lambda_{p,1} & & \lambda_{p,m} \end{bmatrix} * \begin{bmatrix} F_1 \\ \dots \\ F_m \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \dots \\ \epsilon_p \end{bmatrix}$$

Fundamentally, the differences between CFA and EFA are largely driven by the context in which they are used. If one were faced with a survey dataset (or some set of manifest variables) with the goal of developing latent scales to describe unmeasured factors, they might use an EFA model to assess the existence of factors. EFA would answer questions such as "how many latent factors are represented by these manifest variables?" and "how are the manifest variables related to each other and to each factor?"

On the contrary, properly applied Confirmatory Factor Analysis implies that a latent factor model has been specified before the data are examined. Typically, this sort of model makes sense in contexts such as psychometrics or econometrics, where theory-driven latent models are empirically tested via surveys, etc. These also make sense in iterative research, where previous work may inform the CFA model to be used in the next paper/research project. It also makes sense that one might create a latent scale in an exploratory analysis (EFA) and then apply CFA to a new dataset to test that scale.

I recently ran into a good example of CFA in working at the Office of Planning and Analysis, which helped organize the HERI Survey of Campus Climate. Basically, the survey asked questions about whether respondents felt safe/valued/etc. on campus, and whether they had witnessed things like discrimination,etc. HERI is analyzed each year, so their previous report contained a list of factors found in the previous survey, which contained different schools than in 2016 (the year MSU participated). In that case, it made sense to utilize the factor model that HERI had already established by using CFA to map the survey items to the latent factors and test model fit. I would use these data in this example, but given their sensitive nature, I instead opted to go with a more canonical example.

Hildreth (2013) explored an example from a paper by Mardia, Kent, and Bibby (1979) in her doctoral dissertation; these data provide an interesting application of CFA (and a nice extension to more general SEMs). The dataset is previewed in Table 1. The data refer to $n = 88$ student's test scores across five subjects where two tests were open book and the other three were closed book. Dr. Hildreth (2013) points out that these data are typically analyzed with a 2-factor model, one for open and the other for closed book. It makes sense to use CFA for this because the goal is to test whether or not the open and closed-book exams load differently onto separate factors that could be construed as latent functions of exam type. Fortunately, the data are available in the MVT package in R. Note EFA could also be used on this dataset, although it has been worked on enough that CFA probably makes more sense.

```r
library(MVT)
data(examScor)
set.caption("First rows of examScor dataset.")
pander(head(examScor))
```

Table 1: First rows of examScor dataset.

mechanics	vectors	algebra	analysis	statistics
77	82	67	67	81
63	78	80	70	81
75	73	71	66	81
55	72	63	70	68
63	63	65	70	63
53	61	72	64	73

3) **For this question, we will explore sparse exploratory factor analysis as discussed in Hirose and Yamamoto (2015), which I will call HY2015, and compare that to two papers we have discussed previously, regularized SEM as described in Jacobucci, Grimm, and McArdle (2016), JGM2016, and Jacobucci (2017 on ArXiv), J2017.**

a) **What are some reasons that regularization is of interest in EFA? In CFA? What is similar and what is different about penalized estimation in these two situations? [0.75 page]**

In any factor analysis model, there are several issues that can be solved with regularization. For one, in datasets that involve more predictors than observations ($p > n$ setting), methods that reduce the dimensionality of the data in a reasonable way are necessary to make factor analysis possible. This is discussed in a later question in more detail.

Consider a gene expression model much like the one examined in West (2003) or by Carvalho et al (2012). In either case, the goal of the factor analysis is to map only small (i.e. useful) number of observed genes to a given factor (i.e. a latent biological structure/phenomenon). In those two instances, regularization was achieved via careful choice of shrinkage priors on the factor loadings to induce sparsity, but in the same vein, a lasso could achieve much of the same goal.

In EFA, recall that the goal is to explore a series of factors and "investigate the relationship between manifest variables and factors without making any assumptions about which manifest variables are related to which factors" (Everitt and Hothorn, 2011). Much like a linear regression problem, an EFA might benefit from removal of some of the noisier features to map manifest variables to latent factors. Regularization would penalize the loadings of each variable to a factor, removing the less important ones entirely (assuming you are using something that penalizes all the way to 0, like lasso). This would result in a parsimonious mapping of manifest features to the latent traits with which they are most strongly tied.

CFA, on the other hand, is a special case of Structural Equation Modeling (Everitt and Hothorn, 2011). In Hildreth (2013), the CFA model is described as akin to the measurement model in SEM, the model by which latent factors are mapped to observed manifest variables. Regularization in this context would remove (or shrink) the loadings for features that did not play a role in measuring the latent factor. Since CFA models test known a priori theories, regularization may be construed as a tool to inform how tightly measured variables are related to unobserved latent theoretical constructs (i.e. the factors). Perhaps more succinctly, I think of regularization in CFA as something of a substitute for hypothesis tests for "significance" of a given loading, in the same way that lasso zeroing out a coefficient in a linear model might be taken as an alternative to significance tests for that β coefficient.

b) **In the two approaches, they report similar but slightly different versions of penalized functions to optimize (equation 7 in HY2015 and equation 6 in JGM2016). Report the two equations and explain the differences. I think that if you used the same penalty (something simple like lasso) for either EFA or CFA and appropriately tuned using either HY2015's eq 7 or JGM2016's eq 6 for a given method (EFA or CFA), that they would give you the same answer. Do you agree or disagree? Provide your thoughts in either direction. [0.75 page]**

The equations reported by the two papers are given below. In both, $P(\cdot)$ is a penalty function.

In Hirose and Yamamoto (2015), they describe the problem as being maximized by the penalized log-likelihood, where $\rho > 0$ is a regularization parameter.:

$$l_p(\Lambda, \Psi) = l(\Lambda, \Psi) - \sum_{i=1}^{N} \sum_{j=1}^{N} \rho P(|\lambda_{ij}|)$$

This is further expanded when considering the form of the likelihood given in equation 1. Note that I'm defining C to be the sample covariance matrix whereas these authors use S in the paper.

$$l_p(\Lambda, \Psi) = -\frac{N}{2}[plog(2\pi + log(\Sigma)) + tr(\Sigma^{-1}C)]$$

In equation 6 in Jacobucci et al (2016), the ML cost function is written slightly differently - I'm combining the form in Equation 5 into the Equation 6. In this case, the regularization parameter is λ.

$$F_{ML} = log(|\Sigma|) + tr(C * \Sigma^{-1}) - log(|C|) - p + \lambda P(\cdot)$$

Assuming that the same penalty were used, these do appear to give us a similar, if not exact, answer. Carrying out some math on the piece from Hirose and Yamamoto (2015), we can get to something that looks pretty similar to the loss function in the SEM framework. I'm dropping the λ_{ij} and re-writing the penalty parameter as λ to match notation types.

$$l_p(\Lambda, \Psi) = -\frac{N}{2}[plog(2\pi) + log(\Sigma) + tr(\Sigma^{-1}C)] - N\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda P(|\cdot|)$$

My sense of things is that the signs differ because Jacobucci is trying to minimize logloss, whereas Hirose and Yamamoto talk about maximizing the log-likelihood. Even if we flip the signs, there seem to be some minor differences between the formulas. However, the goal is to optimize with regard to the model parameters, so in that case, the terms that differ should have partial derivatives of 0; in both cases, the optimum value should be a function of the $log(\Sigma)$, the trace of the covariances, and the penalty term. So I think it is reasonable to expect that you would end up in the same place.

c) **The methods of HY2015 are available in the `fanc` R package (Hirose, Yamamoto, and Nagata, 2016; also discussed in Yamamoto, Hirose, and Nagata, 2017). It provides options for MC+ (type= "MC") and "prenet" (prenet is from a different paper and we will ignore that for now). What results do HY2015 use to support a focus on MC+? How do the results for MC+ from J2017 compare to these results? [0.75 page]**

The results used by HY2015 to justify the focus on the MC penalizer are presented in Example 3, where the lasso was directly compared to the non-convex MC penalizer. In their example, they plotted the solution paths of the lasso and the MC with $\gamma = 1.96$; note that a value of γ close to 1 provides a hard threshold operator rather than the soft threshold of the lasso (associated with $\gamma = \infty$). The solution paths show that for the lasso, no value of tuning parameter ρ can recover the true nonzero elements. For the MC penalizer, there is an entire range of ρ values for which the penalizer selects the correct model, meaning that the MC penalizer outperformed the lasso in this particular case.

Jacobucci (2017) compared different types of penalties in an example of a latent growth curve model, including standard ridge, lasso, and ML versions with other penalties including an adaptive lasso, non-convex SCAD, and MC-Plus penalizers. The data were simulated to have two large effects, two small effects, and 16 true 0 effects. Further, the γ parameter for MCP was fixed at 3.7. Interestingly, the MCP performed poorly relative to the lasso, although it only identified one false effect (lasso mis-identified 2). The MCP model did end up with a lower BIC than either the lasso or MLE, but the results for Jacobucci were not as effusive as those in the sparse factor analysis models of Hirose and Yamamoto (2014). It is possible that these results would have been improved if they had allowed γ to vary instead of fixing it.

d) **What is a relaxed lasso as discussed in `regsem` (Jacobucci, Grimm, Brandmaier, Serang, and Kievit, 2019)? In `fanc`, the MC+ penalty can provide the lasso penalty - when does that happen? But HY2015 do not discuss the relaxed lasso. Why can `regsem` easily implement the relaxed lasso but it is not a trivial extension of the methods for `fanc`? [0.5 page]**

The relaxed lasso model is discussed in Jacobucci (2017) as part of a comparison of different types of penalities available in the `regsem` package. He notes that "the lasso estimation of the true effects was attenuated in comparison to other…methods… necessitating the use of a two-step relaxed-lasso method"(2017). The original paper on relaxed lasso by Meinshausen (2007) describes the relaxed lasso as a method for separating out its two functions: 1) **Regularization** and 2)**Feature Selection**. In some sense, one could think of the relaxed lasso as a method by which lasso is performed twice, once to sort out the noisiest group of features and the second time to shrink the remaining estimates. In the second model, I would consider using ridge penalties if I felt that I had performed adequate feature selection in the first step.

The MC penalty that provides the lasso penalty in the `fanc` package is actually a more generalized penalizer than a pure lasso. Unlike the pure l1-penalization precure, the MC is a non-convex penalizer that involves two parameters, γ and rho. For values of $\rho > 0$, when γ approaches ∞, the MC penalty is equal to the lasso.

The MC penalty is discussed by Hirose and Yamamoto (2015) as being a flexible penalizer that, when properly tuned, can yield a soft-thresholding operator (lasso penalty) as well as a hard threshold operator. In relaxed lasso, the ϕ parameter controls the

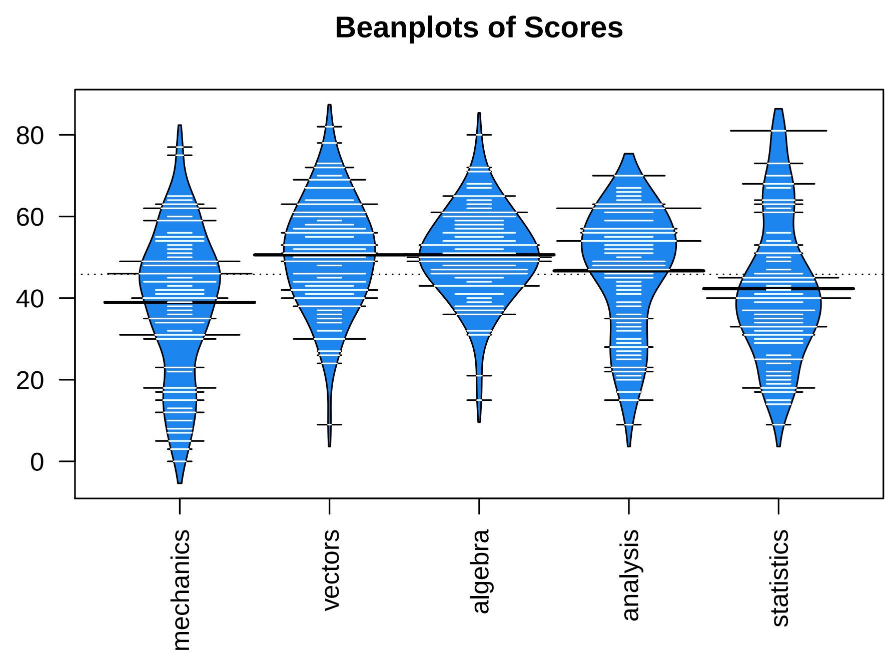


Figure 1: Beanplots of scores on each exam. Mechanics and Vectors were closed book tests.

'hardness' of the thresholding parameter (Meinshausen, 2006) - this indicates that in some sense, relaxed lasso may be a way to compete with the non-convex MC penalizer. Since `fanc` relies on a choice of two non-convex penalties (MC or SCAD), and does not explicitly allow a traditional lasso penalty (only via the special case of MC), it does not easily implement the additional penalty term to the objective function (ϕ) nor does it allow for easy two-step implementation of feature selection followed by regularization. Put more plainly, it seems that the methodology under the hood of `fanc` is designed to do what `regsem` does in a two-stage process in a single stage.

e) **For your example, perform regular EFA, raw ML and varimax rotated ML, and penalized EFA (tuned using a criterion that they suggest). Pick a reasonable number of factors for the situation and hold that constant across the methods (you do not need to spend too much time trying to find the "correct" number of factors). Compare the results and discuss which you might use here. Try to include diagrams of the models to make comparisons easier. (Note: the `plot` option in `fanc` maybe helpful for one of the models.) [1 page of text plus figures/tables]**

The data utilized by Hildreth (2013) are analyzed in several different ways. Prior to analysis, I conducted a little bit of exploratory analysis to get a sense for the different scores on each exam. The beanplots in Figure 1 show the mean scores (wide black ticks), individual scores (white ticks), and estimated distributions (blue shapes) of exam scores by each course exam. Recall that the two exams that were taken as closed-book tests were Mechanics and Vectors, while the other three courses were open-book exams. There are some slight differences in the mean and variability of each group, but nothing too dramatic.

```r
#head(examScor)
beanplot(examScor, las = 2, main = "Beanplots of Scores",
         col = c("dodgerblue2","white"), cut = TRUE)
```

Typically, a factor analysis on these data is performed with two factors, so I hold that value constant for all the versions of factor analyses fit below. Prior to running any of the analyses, I think the traditional method may be just as effective here because there are only 5 measured variables in the data - it may make more sense to utilize the penalized version of this in higher-dimensional problems. Note that these data do have some issues with influential points and outliers (particularly observation

81), as overviewed by Hildreth (2013) and outlined in Tanaka et al (1991). However, I did not remove them from the dataset and instead opt to keep these concerns top of mind as I interpret the results.

Interestingly, I found that the two factor solution did not appear to be a great fit for the data - even in the CFA I ran, most of the model metrics did not indicate great model fit. As it pertains to the different EFA methods, I tried the ML without rotation, the ML with varimax rotation, and a penalized version. I also include a principal-axis-based version for reference.

The penalized verion of the EFA uses a $\gamma = Inf$ so it performs the soft thresholding lasso penalizer; I chose the combination of tuning parameters that minimized Akaike's An Information Criterion (AIC), resulting in a $\rho = 0.129$.

Interestingly, most of the models do not fit loadings onto the second factor. The principal axis and raw ML versions assign large loadings from all the exam scores to the first factor. For the raw ML version, the "closed-book" exams load positively onto the 2nd factor whereas the "open-book" exams load negatively. The varimax-rotated version loads more evenly than either of the other versions, with large loadings for the in-class exams going towards the first factor and mapping the in-class exams to the second factor.

The penalized version was chosen based on a lasso penalizer and optimized based on AIC. I show both the path diagram and the heatmap of the loadings for the model (or, technically, one pretty close to the optimized version - the sliders are not granular enough for me to match the selected model exatly). The AIC-selected penalized version loads the manifest variables all to the first factor, with the in-class exams providing the most weight to the second factor. One open-book exam, Algebra, gets mapped to the second factor but it has a much smaller loading than the closed-book exams; the other two open-book manifest variables are zeroed out by the penalizer. Table 2 gives the loadings for each of the models; Figure 2 shows the path diagrams for the ML, rotated, and PA versions of the model. The regularized FA model is shown in two ways - a heat map in Figure 4 and a path diagram in Figure 3.

Given what I know about this dataset, the varimax-rotated FA appears to most closely match the results that I expected to get; however, the penalized version could be selected with a combination of ρ and γ that provides a similar solution. Given that the AIC-selected solution did not achieve this, I would stick with the two-stage ML and varimax-rotated solution here.

```
#Exploratory Factor Analysis: Raw, ML, Varimax ML, Penalized
library(psych)
#Raw principal axis version
fa_raw <- fa(examScor, nfactors = 2, rotate = "none", fm = "pa")
## Maximum Likelihood versions
fa_ml <- fa(examScor, nfactors = 2,rotate = "none") #EFA with ML
fa_vm <- fa(examScor,nfactors = 2, rotate = "varimax") #ML with varimax rotation

#Penalized Version
X <- examScor %>% as.matrix()
fa_pen <- fanc(X, factors = 2) #runs in about 10 seconds
out(fa_pen, rho = 0.1, gamma = Inf)
```

```
## $loadings
## 5 x 2 sparse Matrix of class "dgCMatrix"
##              Factor1 Factor2
## mechanics  0.4654632       .
## vectors    0.5324247       .
## algebra    0.7928909       .
## analysis   0.6389628       .
## statistics 0.5901570       .
##
## $uniquenesses
##  mechanics    vectors    algebra   analysis statistics
##  0.6540316  0.5670667  0.1522464  0.4116500  0.4853876
##
## $df
## [1] 10
##
## $criteria
##      AIC      BIC     CAIC     EBIC
## 271.8500 296.6234 306.6234 319.6493
##
## $goodness.of.fit
##        GFI       AGFI        CFI      RMSEA       SRMR
## 0.93891706 0.81675117 0.97732597 0.07042962 0.18297036
##
## $rho
## [1] 0.1023743
##
## $gamma
## [1] Inf
```

```
fa_select <- select(fa_pen, criterion = "AIC", gamma = Inf)
```

```
par(mfrow = c(3,1))
fa.diagram(fa_raw, main = "PA Version")
fa.diagram(fa_ml, main = "ML Not Rotated")
fa.diagram(fa_vm, main = "ML Varimax")
```

```
#code to create the static images below
plot(fa_pen, type = "heatmap")
plot(fa_pen, type = "path")
```

```
#table of loadings

fa_ml$loadings
```

```
##
## Loadings:
##            MR1    MR2
## mechanics   0.643  0.342
## vectors     0.708  0.287
## algebra     0.897
## analysis    0.771 -0.235
## statistics  0.718 -0.228
##
##                  MR1   MR2
## SS loadings    2.829 0.314
## Proportion Var 0.566 0.063
## Cumulative Var 0.566 0.629
```

```
fa_vm$loadings
```

```
##
## Loadings:
##            MR1   MR2
```

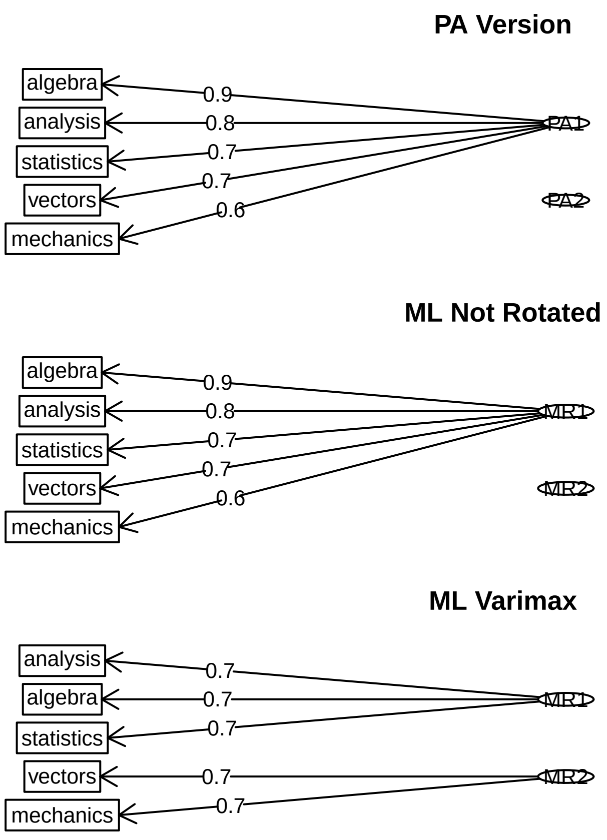
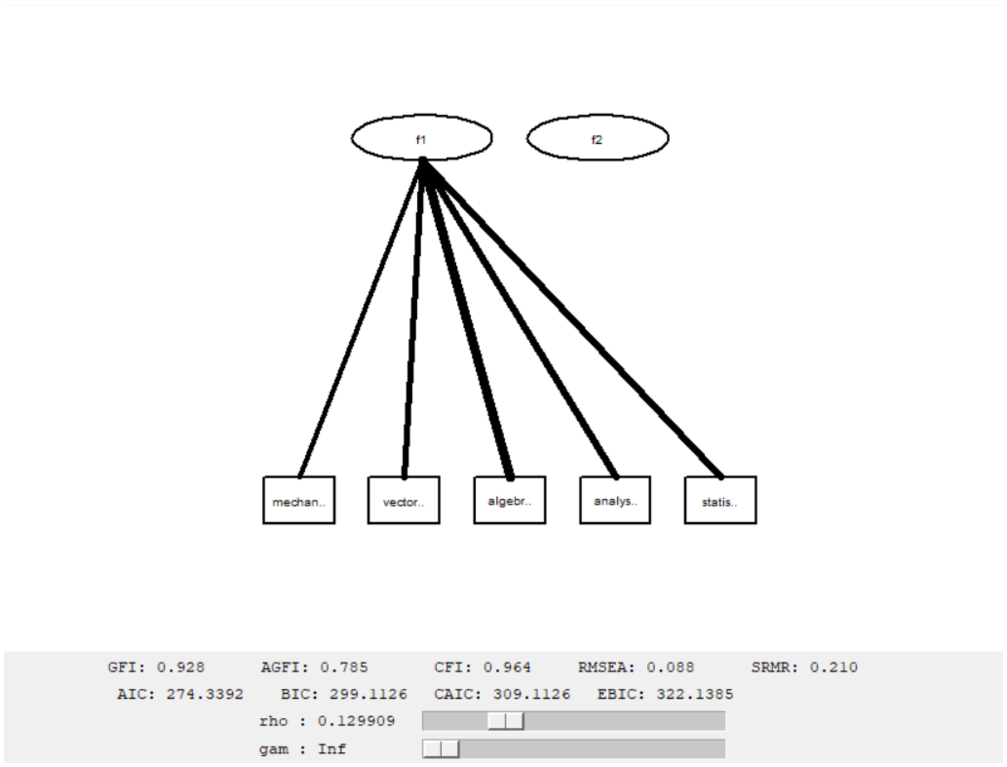
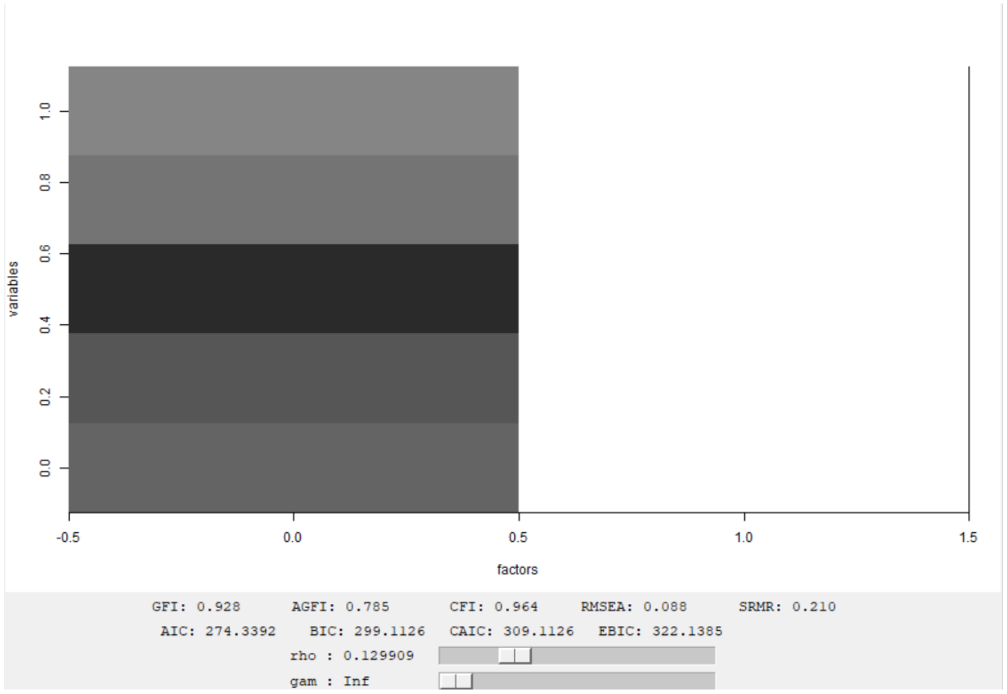


Figure 2: Diagrams created by the different factor analyses.



```
GFI: 0.928     AGFI: 0.785     CFI: 0.964     RMSEA: 0.088     SRMR: 0.210
 AIC: 274.3392    BIC: 299.1126    CAIC: 309.1126    EBIC: 322.1385
```

Figure 3: Path diagram from penalized EFA.



```
GFI: 0.928     AGFI: 0.785     CFI: 0.964     RMSEA: 0.088     SRMR: 0.210
 AIC: 274.3392    BIC: 299.1126    CAIC: 309.1126    EBIC: 322.1385
```

Figure 4: Heatmap from penalized lasso EFA.

```
## mechanics  0.267 0.677
## vectors    0.353 0.678
## algebra    0.739 0.515
## analysis   0.739 0.321
## statistics 0.695 0.292
##
##                  MR1   MR2
## SS loadings    1.771 1.372
## Proportion Var 0.354 0.274
## Cumulative Var 0.354 0.629
```

```r
fa_select$loadings
```

```
## 5 x 2 sparse Matrix of class "dgCMatrix"
##              Factor1   Factor2
## mechanics  0.5044027 0.4943428
## vectors    0.4614189 0.5739471
## algebra    0.1679969 0.8633724
## analysis   .         0.7836796
## statistics .         0.7303809
```

f) **Generate or find an example with more variables than observations ($N \ll p$). Perform a penalized EFA on this data set and discuss the results. What happens when you try to do a typical EFA with $N \ll p$? [1 page plus figures/tables]**

West(2003) explores a dataset first utilized by Osborne et al.(1984) and Brown et al.(2001). The data refer to 72 observations of biscuit dough made from recipes with varying levels of fat, water, sucrose and flour that were subjected to NIR spectroscopy to assess how well the Spectroscopy could measure the composition of the aforementioned ingredients in each batch of dough. The problem is high-dimensional in that there are substantially more NIR spectrum measurements (700) than observations (72). The data pertain to spectrum measured between 1100 and 2498 nm at 2nm wavelengths; there are 700 spectra and 72 observations on which the spectra are measured.

The data are plotted in Figure 5.

```r
suppressMessages(library(ppls))
data(cookie)
X <- scale(as.matrix(cookie[,1:700]),center = TRUE,
           scale = FALSE)#NIR spectra
Y <- as.matrix(cookie[,701:704]) #I will likely ignore these

#plot(X)
## Just produces a plot of scaled spectra
wavelength <- seq(1100, 2498, by = 2)
plot(wavelength, X[1,], type ="l", main = "Scaled Spectra",
     ylim = c(-.3,.34))
for(j in 2:nrow(X)){
  lines(wavelength,X[j,], col = j)
}
```

West notes that his Bayesian factor analysis model identifies 16 factors in the model, so I chose 16 as a the baseline to work with in this analysis. Below, Penalized EFA is performed on the dataset.

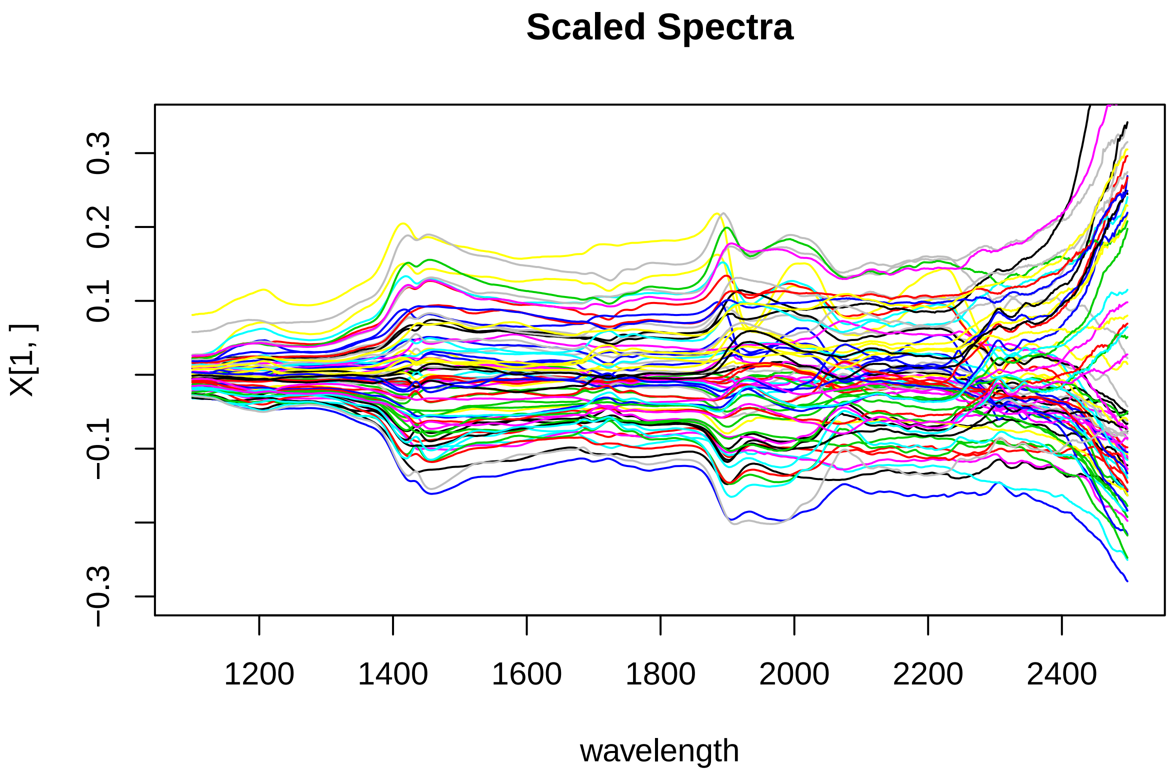


Figure 5: Scaled Spectra at each wavelength.

```r
fa_pen <- fanc(X,factors = 16, type = 'MC')
#this code takes about 3 hours to run
```

```r
#output for fixed tuning parameters
out.1 <- out(fa_pen, rho=0.1, gamma=Inf)
#select a model via model selection criterion
model_select <- fanc::select(fa_pen, criterion="BIC", gamma=Inf)
model_select$loadings[1:15,1:6]
```

```
## 15 x 6 sparse Matrix of class "dgCMatrix"
##          Factor1      Factor2   Factor3       Factor4      Factor5
## 1  -3.652523e-03 .            0.1024064  1.722644e-03 -0.004722665
## 2  -2.218307e-03 .            0.1027062  1.180630e-03 -0.005131344
## 3  -1.714164e-03 2.319089e-04 0.1031929  1.408060e-03 -0.004375329
## 4  -1.532889e-03 5.737682e-04 0.1034688  1.505570e-03 -0.004189485
## 5  -6.853019e-04 3.299554e-04 0.1036873  8.499138e-04 -0.004471487
## 6  -6.693151e-05 5.546554e-05 0.1035929  5.857019e-04 -0.005174041
## 7   .            .            0.1036768  6.262277e-04 -0.004879535
## 8   .            .            0.1036760  5.898472e-04 -0.004985082
## 9   .            .            0.1042062  8.773588e-04 -0.003997937
## 10  1.040712e-03 .            0.1047068  2.114747e-05 -0.003802506
## 11  1.343086e-03 4.283851e-06 0.1056140  .            -0.002480577
## 12  1.945602e-03 .            0.1063243  .            -0.002309862
## 13  1.870780e-03 .            0.1070358  .            -0.002398041
## 14  2.144889e-03 .            0.1079697 -8.433588e-05 -0.002232159
## 15  2.768826e-03 .            0.1086996 -1.499412e-03 -0.003035121
##         Factor6
## 1  -0.008444366
## 2  -0.008482873
## 3  -0.008938074
## 4  -0.008532906
## 5  -0.008858830
## 6  -0.008388550
## 7  -0.009080263
## 8  -0.009199019
## 9  -0.009507416
## 10 -0.009188087
## 11 -0.009210169
## 12 -0.008054734
## 13 -0.006893450
## 14 -0.005850334
## 15 -0.004070346
```

The loadings matrix is quite large in this case, so I did not include the entire thing. Instead, I printed out the first 15 rows and 4 columns, and show a heatmap of the loadings matrix in Figure 6. The effect of the lasso penalizer is obvious in that the loadings for some facator 1 for observations 7 to 10 are shrunk to 0. Similarly, most of the loadings are shrunk to 0 for the second factor. Moreover, the the loadings are all quite small; this is likely an effect of the regularization. Recall that the data are scaled spectra at a 2mm distance, so it makes some sense that the regularization should occur in chunks. You can see that the majority of the loadings are taken all the way down to zero, especially for Factors 9 - 16.

```r
suppressMessages(library(plsgenomics))
matrix.heatmap(model_select$loadings, main = "Loadings from biscuit data")
```

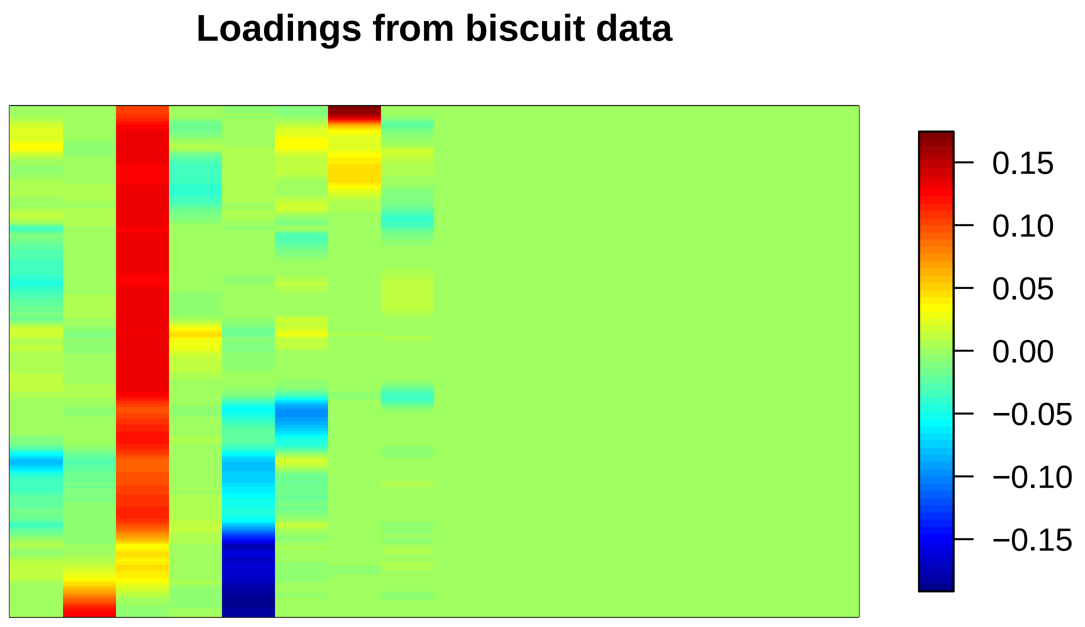


Figure 6: Heatmap of the loadings. Green values are zeroed out by the lasso.

I suspect that the 16 factor solution is probably a more complex model than necessary. Since all the loadings for the 9-16th factors were zeroed out, this may indicate evidence that somemthing like an 8 factor model would be sufficient. While there are goodness of fit statistics that can be calculated, these would make more sense to assess in a CFA framework.

When regular EFA is attempted on the dataset, it returns an error that the "system is computationally singular". It will not run in a $p > n$ setting - note that `factanal` runs on a subset of the data as long as the dimensionality of that subset is less than $n = 72$.

```r
fa_16 <- factanal(X, 16, rotation="varimax")
```

```
## Error in solve.default(cv): system is computationally singular: reciprocal condition number = 1.59919
```

```r
#for reference
fa_sample <- factanal(X[,1:71], 16, rotation = "varimax") #runs
fa_sample <- factanal(X[,1:72],16,rotation = "varimax") #throws the error
```

```
## Error in solve.default(cv): system is computationally singular: reciprocal condition number = 1.82685
```

Part II: Mapping high dimensional data

4) **Focus on van der Maaten's papers related to t-SNE and Witten and Tibshirani's 2010 paper related to `sparcl` (Witten and Tibshirani, 2018) to answer the following questions:**

a) **Explain the purpose of t-SNE and how it is used, including an example to illustrate its use and interpretation. [1 page plus figure(s)]**:

In statistics, high-dimensional data problems often involve situations where there are more features in the data than there are observations (the $p > n$ problem), or even when there are enough observations, most of the variables contribute noise rather than signal to the process. Dimension-reduction techniques focus on identification of a "smaller fundamental set of independent variables... which determine the values of the original p variables"(Hotelling, 1933). This concept that a narrower set of $k < p$ features that may contain the majority of the information in the original p features drove the development of PCA, MDS, Sammon Mapping, and other multivariate mapping techniques.

High-dimensional data visualization, on the other hand, seeks to find ways to visualize more than 2 or 3 dimensions of data without fundamentally changing the underlying scales of interest; a good example of this idea can be found in Chernoff faces (Chernoff, 1973).

T-distributed Stochastic Neighbor Embedding (t-SNE) seeks to solve both problems simultaneously by identifying a reduction of the data to 2 (or 3) dimensions and producing a 2 (or 3)-dimensional map, with meaningful groupings from the high-dimensional space preserved in the low dimension. In plain English, if observations are similar when using all the features, they should be mapped close together in the resulting t-SNE map.

More technically, t-SNE minimizes the symmetricized Kullback-Leibler divergence between two joint probability distributions, P_{ij} and Q_{ij}. Without getting too technical here, P can be interpreted as a joint probability (using the full feature set) that a given observation (i) would choose another observation (j) as a neighbor, under the assumption that P is distributed normally and centered at i. This is illustrated in Figure 7. The Q similarity operates in the same way, except that it uses a heavier-tailed t-distribution with 1 degree of freedom to estimate the joint probability using only a subset of the features. If the lower-dimensional mapping is faithful to the high-dimensional setting (the actual data), the KL-divergence between P and Q will be small (van der Maaten, 2008), so t-SNE seeks to minimize this for all the pairs of datapoints.

T-SNE is surprisingly easy to implement, with only the choice of a perplexity parameter necessary to run. Perplexity can be thought of as a rough estimate of the expected number of neighbors in each group. Van der Maaten (2008) notes that t-SNE is fairly robust to changes in perplexity values; however, I have not found this to be the case. Below, I implement t-SNE with the package Rtsne on a dataset of NBA Rookies drafted in 2017, with a perplexity of 10. I chose 10 because the dataset is relatively small and I expected to see only two groups based on previous research with PCA.

It is useful to scale the data prior to running t-SNE (in part because the default setting in Rtsne is to run PCA in the background and then t-SNE the already dimension-reduced scores). The plot of the 2-D t-SNE map is shown below in Figure 8. Care should be taken when interpreting these maps; groups of observations have meaning but positions of groups do not. Indeed, based on the random seed chosen, positions of groups may differ substantially across runs.

T-SNE does not offer much in the way of interpretation or creation of new latent scales to assess performance – the axes of variation (X and Y) do not have meaning in an interpret-able sense. That being said, the groupings do have meaning. In this case, t-SNE grouped the players who contributed meaningful statistics to their teams in their rookie year in the bottom left corner (note Donovan Mitchell, Jayson Tatum, and Ben Simmons were all in the running for rookie of the year in 2018).

```r
nba <- read.csv('Data/na.csv', header = TRUE)
suppressMessages(library(Rtsne))#;library(tsne)

nba.dat <- nba[,-c(1:3)]

#using Rtsne
set.seed(32219)
rt1 <- Rtsne(nba.dat, dims = 2, initial_dims = 20, perplexity = 10)
```

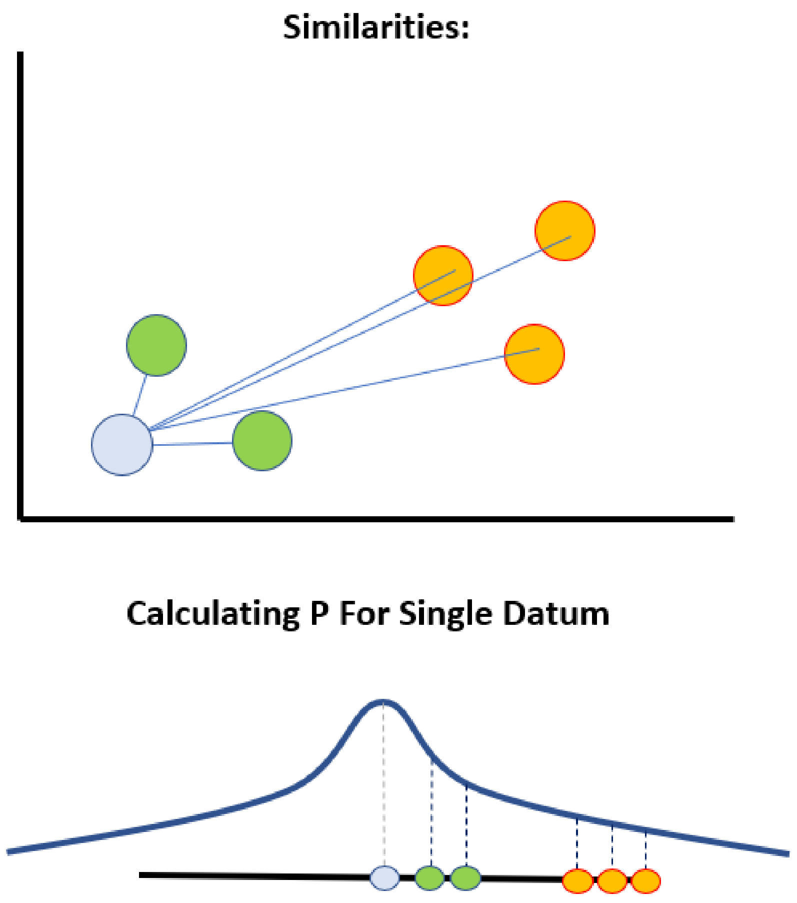


Figure 7: T-SNE similarity diagram, assuming a gaussian distribution.

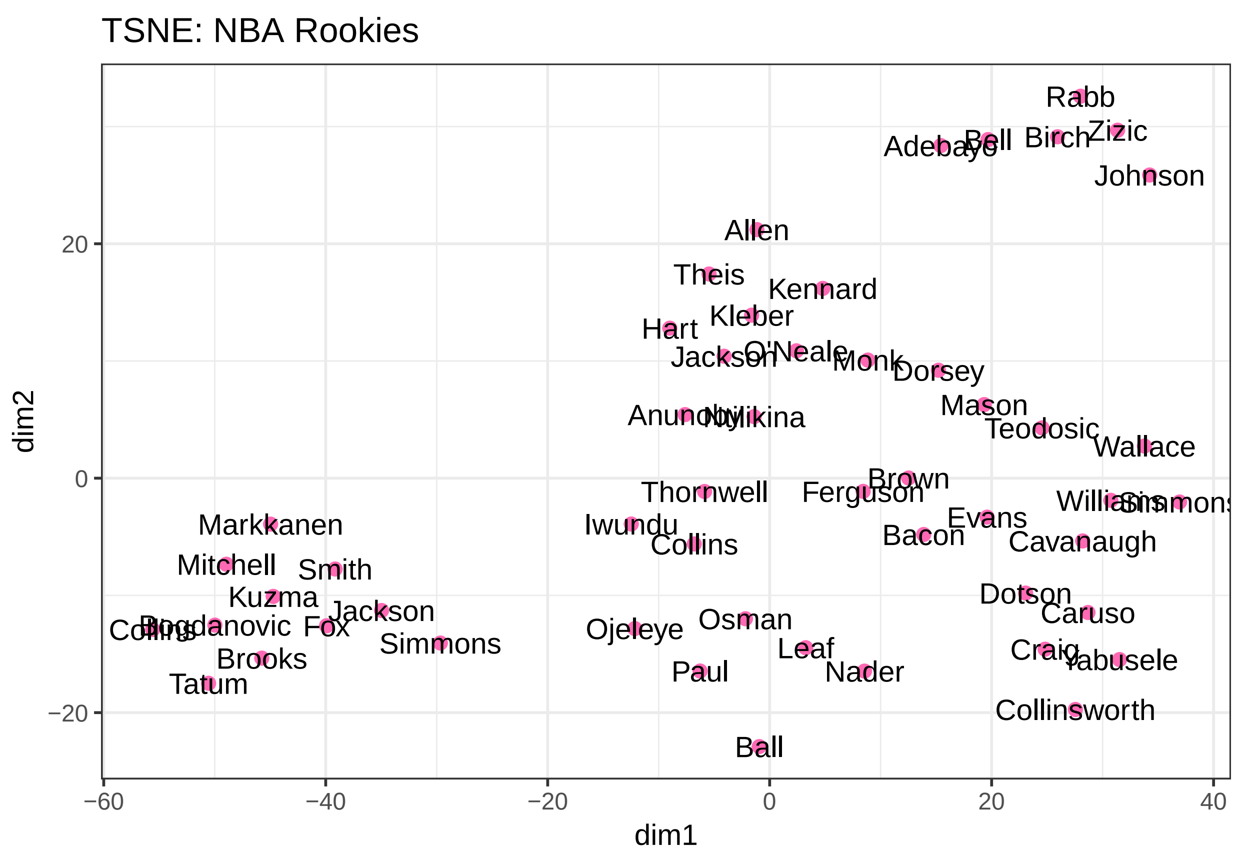


Figure 8: t-SNE plot of NBA rookies from 2017 season.

```
makeplot <- function(TSNEobject, names){
  Y <- data.frame(TSNEobject$Y)
  names(Y) <- c("dim1","dim2")
  p <- ggplot(Y) + geom_point(aes(x=dim1, y = dim2),
                              color = "hotpink", size = 2) +
    theme_bw()
  p + ggtitle("TSNE: NBA Rookies") +
    geom_text(aes(dim1,dim2, label = names))
}

makeplot(rt1, nba[,3])
```

Because of the difficulty of interpretation and black-boxed nature of t-SNE, I would not use this method as a replacement for PCA or a latent-variable model. I like t-SNE for exploratory analysis primarily.

Additionally, I came across a use case where I had the ability to add additional features to a logistic regression model but due to software constraints, the model had a hard cap on the number of features that could be used.

I could only add a single feature to the model (I had 722 to choose from) – the goal in this case was to increase predictive ability of the model without necessarily creating interpretable features to be used. One use of t-SNE might have been to take the 722 feature set and create a 2-D mapping, and then apply model-based clustering to the map. Those clusters could then be added to the model as an additional single feature that contains as much information (perhaps more than would be available if PCA were used) as possible from the original 722 feature set.

b) **Report and explain result 15 (p 719) in Witten and Tibshirani (2010). Make sure to explain how this provides "sparsity" and what that means in terms of the original variables and the distance matrix used for clustering. [0.75 page]**

Result 15 refers to the application of sparse clustering to hierarchical clustering, which segregates the data into groups in a different way from the main focus of the paper, K-means clustering. Interestingly, most of the more recent work in this field has been done on the K-means version of sparse clustering rather than in sparse hierarchical clustering. As noted by Kondo, et al(2016), sparse K-means "cleverly exploits the fact that commonly used dissimilarity measures can be additively decomposed into p terms, each depending on a single variable." This is evident in results 4 and 5 in Witten and Tibshirani (2010), which note that many clustering methods can be expressed as a form:

$$\underset{\Theta \in D}{Max} \sum_{j=1}^{p} f_j(\mathbf{X}_j, \Theta)$$

As noted, based on this ability to factorize the objective function by each feature, they are able to additively implement weights that apply to each factor, w_j. The new optimization problem in sparse clustering algorithms takes a general form, with w subject to an $l_1 < s$ constraint (s is a tuning parameter):

$$\underset{w,\Theta \in D}{Max} \sum_{j=1}^{p} w_j f_j(\mathbf{X}_j, \Theta)$$

The above forms are a general form for the optimization, but this leads us to the framework of hierarchical clustering, which agglomerates observations into groups based on producing dendrograms that must be pruned at a given location. Witten and Tibshirani note that hierarchical clustering requires as input a dissimilarity matrix $\mathbf{U}_{nxn}$, resulting in an optimization problem that seeks to minimize **U** subject to specific constraints ($\sum_{i,i'} \mathbf{U}^2 \leq 1$), as shown below:

$$\underset{\mathbf{U}}{Max} \sum_{j} \sum_{i,i'} d_{i,i',j} \mathbf{U}_{i,i'}$$

Results 14 and 15 follow those of 4 and 5 in that they allow for the distance matrix to be penalized by down-weighting the distance matrix for features that do not contribute to the cluster solution. Result 15, the optimization for sparse hierarchical clustering, results naturally from this implementation of the penalty, with w_j subject to the same constraints as above:

$$\underset{\mathbf{U}}{Max} \sum_{j} w_j \sum_{i,i'} d_{i,i',j} \mathbf{U}_{i,i'}$$

Sparsity is induced because the weights are allowed to vary for each feature in such a way that the weights for "noise" features can be shrunk to 0, and the weights for "signal" features can be retained (and kept relatively large). Looking at their algorithm for doing sparse hierarchical clustering, the original distance matrix is repeatedly updated based on this re-weighting process and then is used as an input to the hierarchical clustering method.

c) **In t-SNE, discuss the role of the distance matrix among the original observations and how it is compared to the distances of the lower-dimensional points. [0.75 page]**

In t-SNE, the distance matrix among the original observations (typically Euclidean) can be input instead of the raw data (however, this requires the Matlab implementation and the `tsne_d.m` function.) More importantly, the distance matrix of the original observations gives the t-SNE algorithm what it needs to calculate the P_{ij} terms.

Recall that t-SNE seeks to calculate two metrics of similarity, P (in the original data space, or high-dimension), and Q, in the low-dimensional space. They are defined below:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

n the high-dimensional space P, the similarity metric still relies on a normal distribution, as in tradition SNE. Typically the individual $p_{i|j}$ is defined as the probability that x_i would choose x_j as a neighbor, with σ_i as the variance of the Gaussian distribution (in the high-dimensional space) centered at the i^{th} datapoint. The distance matrix of the original data is used to calculate the distances between each x_i, x_j, and x_k.

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2/2\sigma_i^2)}$$

Recall that if the lower-dimensional representation closely matches the original data structure, Q should be a reasonable approximation of P. In t-SNE, the joint distribution of the Q's is calculated with a t-distribution, and the symmetricized Kullback-Leibler Divergence between P (which utilizes the original distance matrix) and Q (in the lower-dimensional space) is minimized over all the data points. Put simply, the algorithm is designed to take observations that have large distances in P and map them far apart from each other in the lower-dimensional mapping; conversely, for observations that are close together (i.e. small values in original distance matrix), they should remain close in the lower-dimensional mapping.

d) **Propose a general framework for regularization in the t-SNE algorithm, discussing where the regularization could happen, leveraging the previous discussions, and why this could be useful. Do not program this as this may or may not actually work and certainly would be complicated to optimize. Assume that minimizing the sum of KL-divergences optimizes a given t-SNE map. [0.75 page]**

In t-SNE, there may be two different ways that regularization could enter the algorithm. The first, and probably easiest way, would be to do the procedure in a two-step algorithm not unlike some of the implementations of the relaxed lasso as discussed by Meinshausen (2006). In such a method, the first step would be to do some sort of feature selection via lasso-selection. In a regression setting, one could optimize the tuning parameter λ for a regression problem and choose only the features with non-zero coefficients.

This might be akin to replacing the "whitening" process within t-SNE, in which PCA is run on the data, with a feature selection driven by the lasso. Alternatively, regularized PCA methodology (such as sparse PCA discussed in Zou,et al.(2006)) could be used to combine the w. Application of sparse PCA instead of the traditional PCA (or none), or other sparse methods such as Regularized SEM/sparse FA in this step, might allow for the t-SNE algorithm to operate on a subset of less noisy features. Some options for dealing with regularization in t-SNE are overviewed in Figure 9.

However, these are two-step procedures that, while interesting and probably easier to implement, would not be as optimal as finding a way to penalize the objective function of the t-SNE algorithm itself. The method of penalizing the distance matrix directly utilized by Witten and Tibshirani (2010) may provide our best bet for a way to add regularization to the t-SNE algorithm. First, regularization makes sense really in the high-dimensional space P. Since the similarity metric, P_i, is really just a conditional joint probability distribution given datapoint x_i, this conditional distribution is just a (high-dimensional multivariate) normal distribution centered at x_i. My sense of things is that if we penalized certain features in a similar manner as to Witten and Tibshirani's method in sparse hierarchical clustering, we may be able to downweight the use of certain features in the calculation of the conditional probability in P. This might - perhaps only slightly - mitigate some of the problems associated with the Curse of Dimensionality that affect the Gaussian approximation in high-dimensional settings.

From there, I think the optimization would be quite similar. You would still attempt to make the lower-dimensional mapping Q look like the (penalized) high-dimensional similarity P, so minimizing the symmetric KL divergence would be the goal. If the regularization does a reasonable job of removing noise from the higher-dimensional space, this might actually speed up the optimzation process. That being said, I'm not sure that it's clear how reducing the number of features would affect the crowding problem in the lower-dimensional space. It may be possible that by reducing the number of features in the data, you actually remove some of the map separation between observations that are far apart.

e) **Explore the potential for a sparse version of t-SNE. Do this by working with a data set with relatively clear structure on some variables and some that are not structured or related to that structure, performing t-SNE on it and then performing t-SNE just using the variables with clear structure. You can use your knowledge of which variables should be used to demonstrate this possibility. Note: I don't know if this will work, so use this as an opportunity to assess the potential for pursuing this as a research opportunity and it is OK to conclude that this may not be worth pursuing based on these initial results. [1.5 pages plus figures]**

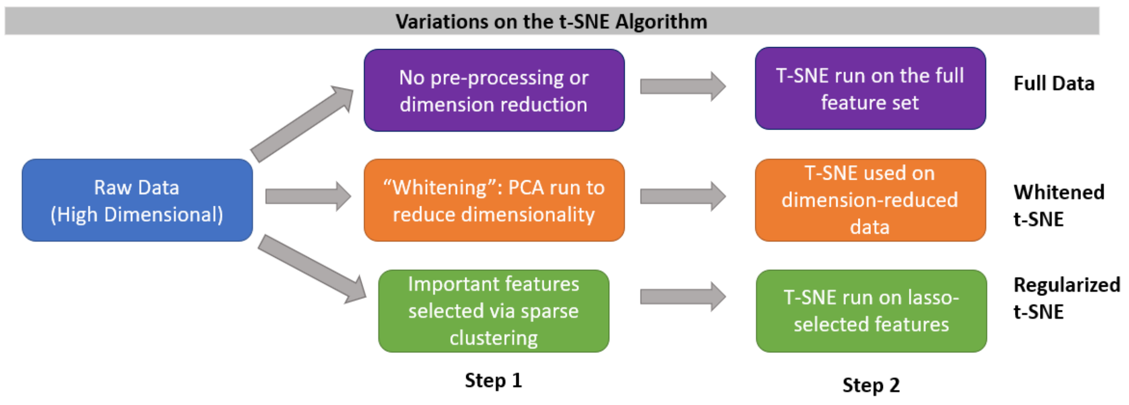

Variations on the t-SNE Algorithm
Raw Data (High Dimensional)
No pre-processing or dimension reduction
"Whitening": PCA run to reduce dimensionality
Important features selected via sparse clustering
T-SNE run on the full feature set
T-SNE used on dimension-reduced data
T-SNE run on lasso-selected features
Full Data
Whitened t-SNE
Regularized t-SNE
Step 1
Step 2

Figure 9: Proposed methods for proceeding through the t-SNE algorithm, with regularized version shown on bottom path.

For exploration of a sparse version of t-SNE, I examined the UCI's repository of Wine characteristics. While many of these types of datasets are much more high-dimensional, this particular dataset has 13 features to consider. There are 3 classes of wine, and although I am not sure what they each class refers to, they are relatively distinct from each other. Figure 10 shows the pearson correlations between the features in the data.

```r
wine <- read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data",
                 col_types = cols())

colnames(wine) <- c('Class','Alcohol', 'Malic.acid', 'Ash',
                    'Alcalinity.of.ash' ,'Magnesium', 'Total phenols',
                    'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins',
                    'Color intensity','Hue'     ,'OD280/OD315','Proline')

pander(head(wine)) #prints first few observations out
```

Table 2: Table continues below

Class	Alcohol	Malic.acid	Ash	Alcalinity.of.ash	Magnesium
1	13.2	1.78	2.14	11.2	100
1	13.16	2.36	2.67	18.6	101
1	14.37	1.95	2.5	16.8	113
1	13.24	2.59	2.87	21	118
1	14.2	1.76	2.45	15.2	112
1	14.39	1.87	2.45	14.6	96

Table 3: Table continues below

Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins
2.65	2.76	0.26	1.28
2.8	3.24	0.3	2.81
3.85	3.49	0.24	2.18
2.8	2.69	0.39	1.82
3.27	3.39	0.34	1.97
2.5	2.52	0.3	1.98

Color intensity	Hue	OD280/OD315	Proline

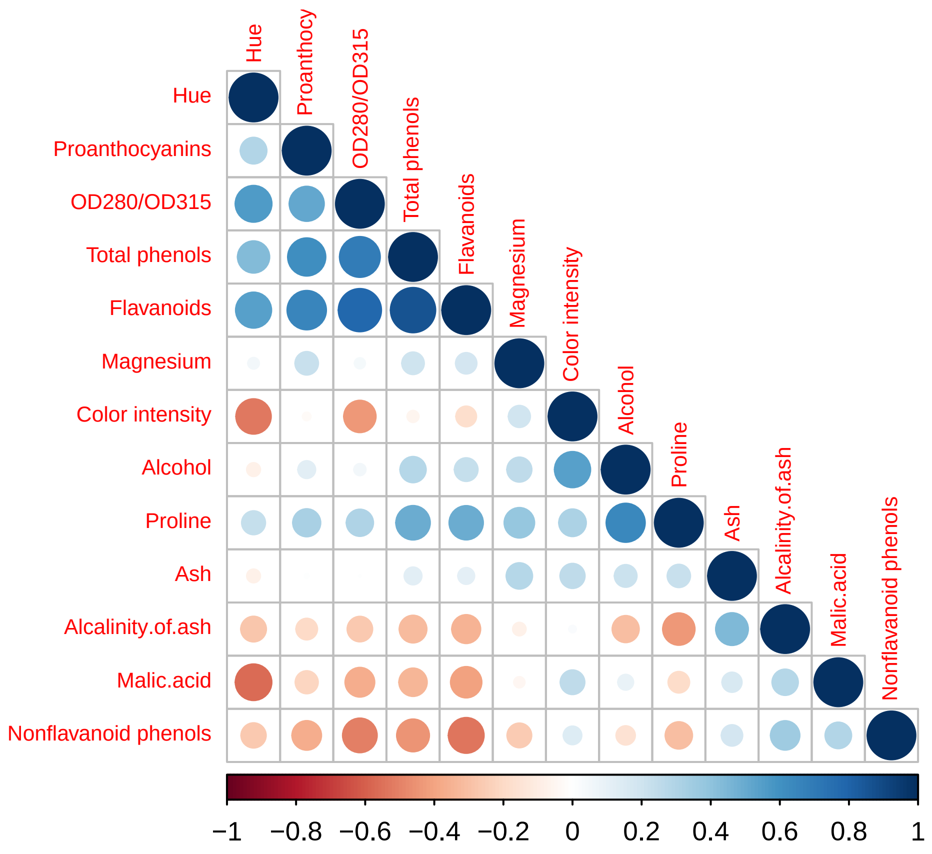


Figure 10: Correlations of variables in the wine dataset.

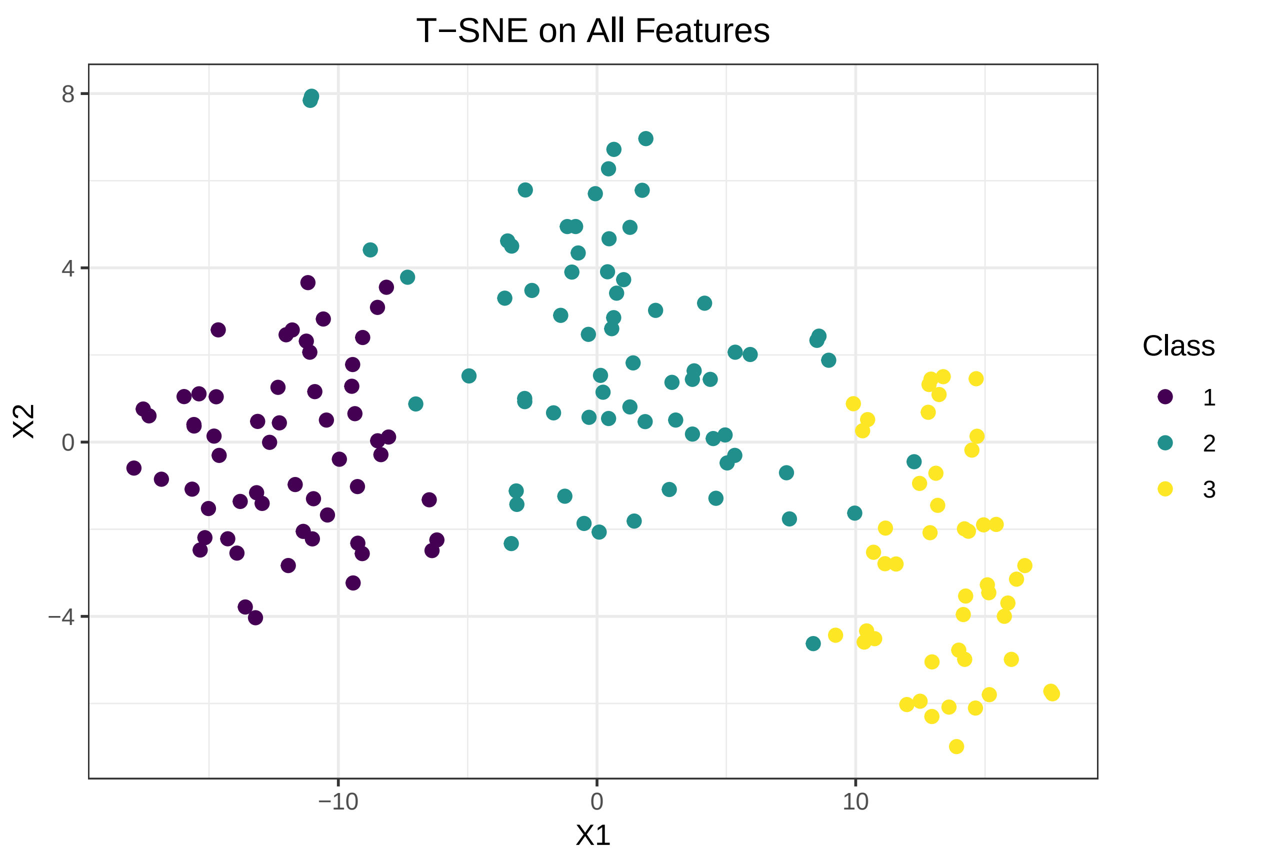


Figure 11: Weights of each coefficient from the RSKC applied to the wine data.

Figure 11 gives a t-SNE run on the entire wine dataset with all 13 features. The three wine types are pretty easy to distinguish, and t-SNE has very little difficulty parsing them out from each other.

The next step was to find a subset of features that contain the majority of the structure in the data. I used sparse k-means (as implemented in the RSKC package, not sparcl) to do the feature selection - a hacky method, I admit, but an easy way to do the lasso selection I need to do. I plotted the indices of the nonzero features in Figure 12. For the purposes of this study, I don't really care that feature 7 (Total phenols) has the largest weight, rather, I'm only interested in using it as input to the t-SNE solution, shown in Figure 13.

```
wine.scale <- wine %>% lapply(.,scale) %>% as_tibble()

ts1 <- Rtsne(wine.scale[,2:14], perplexity = 20, dims = 2, pca = FALSE)
tsdat1 <- as_tibble(ts1$Y);colnames(tsdat1) <- c("X1","X2")
```

```
## Warning: `as_tibble.matrix()` requires a matrix with column names or a `.name_repair` argument. Usin
## This warning is displayed once per session.
```

```
tsdat1$Class <- factor(wine$Class)
ggplot(tsdat1, aes(X1,X2)) + geom_point(aes(color = Class), size = 2) +
  ggtitle("T-SNE on All Features") + theme_bw() + scale_color_viridis_d() +
  theme(plot.title = element_text(hjust = 0.5))
```

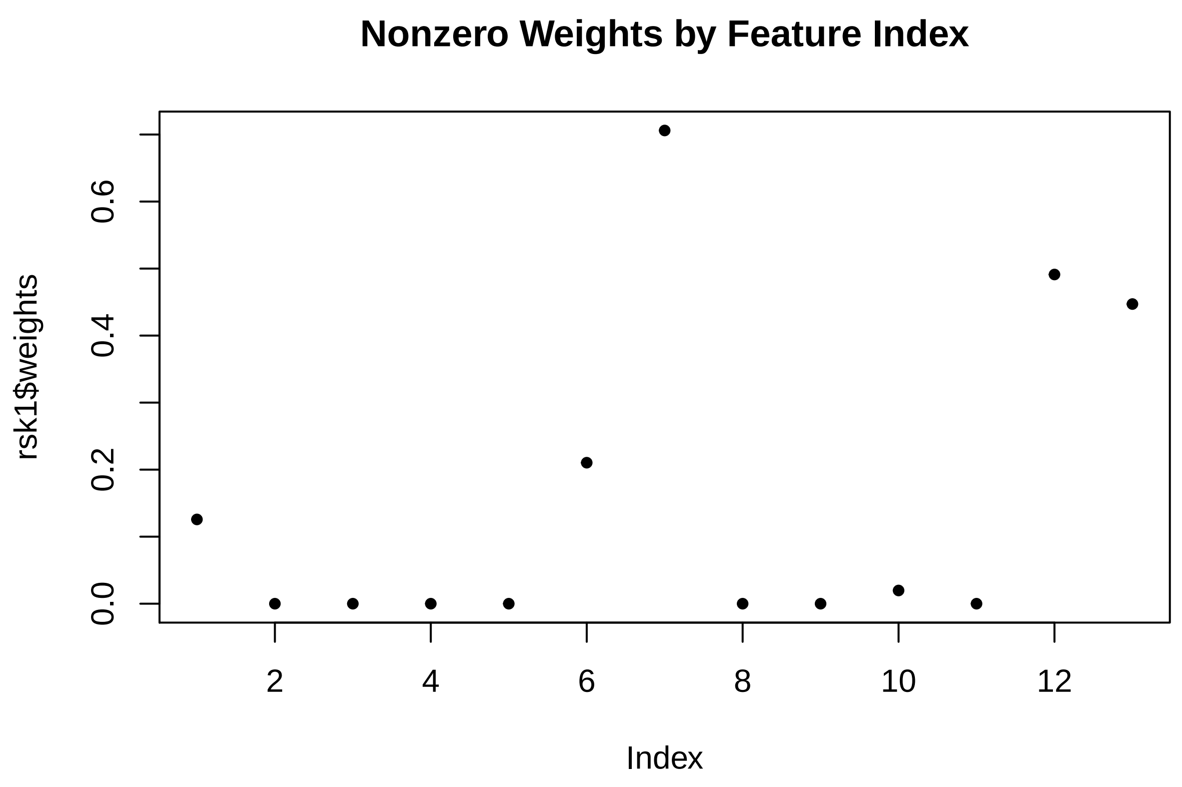


Figure 12: Weights of each coefficient from the RSKC applied to the wine data.

```r
suppressMessages(library(RSKC))
```

```
## Warning: package 'RSKC' was built under R version 3.5.3

## Warning: package 'flexclust' was built under R version 3.5.3
```

```r
rsk1 <- RSKC(wine.scale[,2:14], ncl = 3, alpha = 0.3, L1 = 2)
plot(rsk1$weights, pch = 20, main = "Nonzero Weights by Feature Index")
```

```r
index.use <- which(rsk1$weights != 0)
print(index.use) #only uses 6 of the features
```

```
##         Alcohol   Total phenols      Flavanoids Color intensity
##               1               6               7              10
##       OD280/OD315         Proline
##              12              13
```

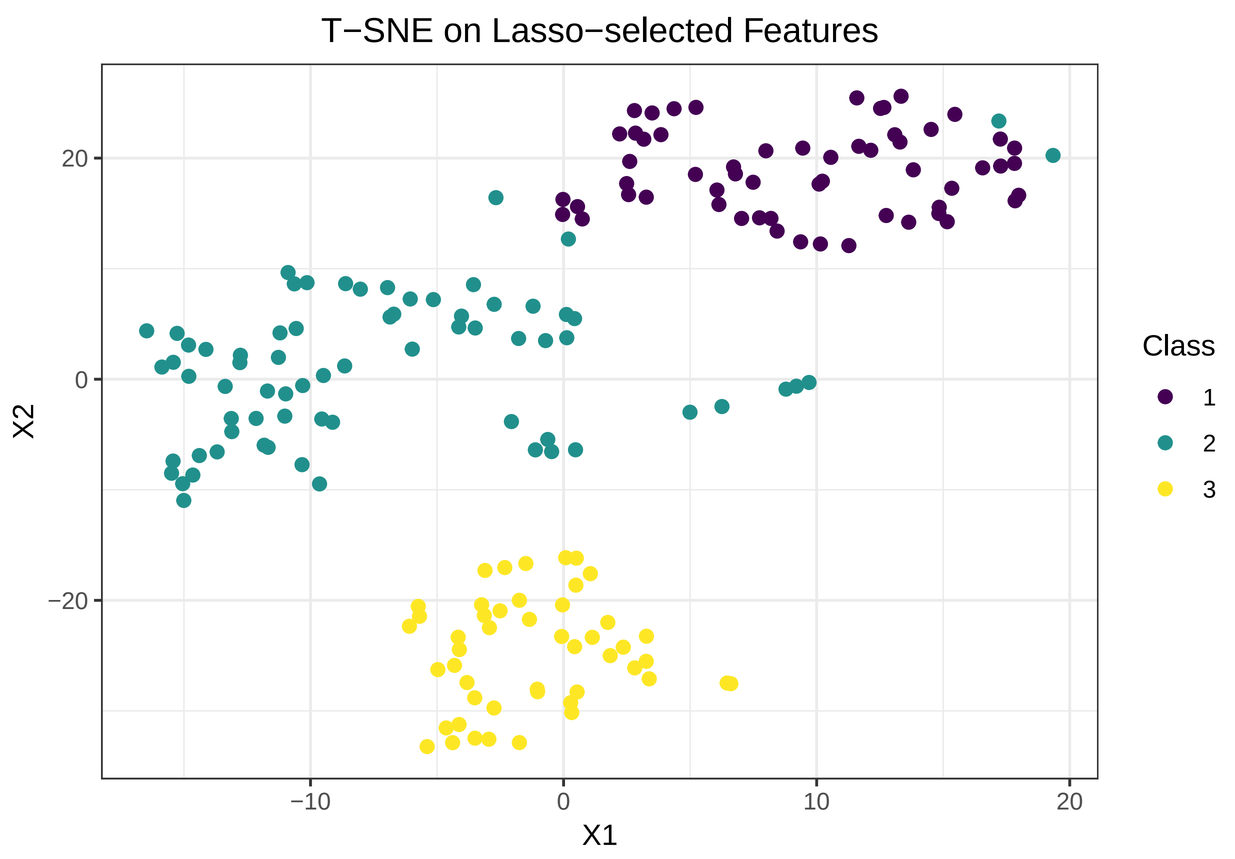


Figure 13: t-SNE on the lasso selected features from the RSKC function.

```r
#now apply the t-SNE version of this
suppressMessages(library(Rtsne))
ts2 <- Rtsne(wine.scale[,index.use], perplexity = 10, dims = 2, pca = FALSE)
tsdat2 <- as_tibble(ts2$Y);colnames(tsdat2) <- c("X1","X2")
tsdat2$Class <- factor(wine$Class)
ggplot(tsdat2, aes(X1,X2)) + geom_point(aes(color = Class), size = 2) +
  ggtitle("T-SNE on Lasso-selected Features") + theme_bw() + scale_color_viridis_d() +
  theme(plot.title = element_text(hjust = 0.5))
```

Admittedly, I am not familiar enough with the data to know if, cannonically, any of the features contain a great deal of structure from which the t-SNE can learn. I can, however, modify the dataset by adding a number of additional fields that are completely highly uncorrelated with the feature set. Since there were six fields that were used in the lasso-selected clustering, I'll use those as a baseline and try several simulations.

My set of "structure variables" refers to those six features. Then, I added additional features to the dataset, all simulated from Normal distributions with mean 0 and standard deviation 1, since the data were previously scaled.

Since increasingly higher-dimensional problems may provide more noise (and thus more difficulty) for the t-SNE algorithm, I compared those solutions with 20 additional noise features, 100 additional noise features, and 1000 additional noise features to the t-SNE solution with only those 6 structure-features utilized. The results are interesting: the t-SNE algorithm fails to recognize that the new features are uncorrelated and has a great deal trouble distinguishing those from those that do provide signal. It is unable to identify the groupings in this new dataset. Notice that in the 20-feature version, it is still able to distinguish some of the class 3 wines, but by the time the data are swamped with 100 noise features, t-SNE is unable to parse the groups out. The resulting maps are given in Figure 14.

```r
wine_baseline <- dplyr::select(wine.scale,index.use)
wine_baseline$Class <- wine$Class


#simulates data (standardized so that it matches scale)
makeFakedata <- function(mean = 0,sd = 1){
   fake.dat <- rnorm(nrow(wine),mean, sd)
return(fake.dat)}


### BUILDS A 20-variable feature set
for(j in 1:20){
  top <- ncol(wine_baseline)
  wine_baseline[,top + 1]<- makeFakedata()
    names(wine_baseline)[top + 1] <- paste0("Fake ",j)
}
## runs the t-sne and saves the plot
ts3 <- Rtsne(wine_baseline[,-1], perplexity = 10, dims = 2, pca = FALSE)
tsdat3 <- as_tibble(ts3$Y);colnames(tsdat3) <- c("X1","X2")
tsdat3$Class <- factor(wine$Class)
A <- ggplot(tsdat3, aes(X1,X2)) + geom_point(aes(color = Class), size = 2) +
  ggtitle("T-SNE with 20 Noise Features") + theme_bw() +
  scale_color_viridis_d() + theme(plot.title = element_text(hjust = 0.5))


#BUILDS A 100-variable feature set
### BUILDS A 20-variable feature set
for(j in 1:100){
  top <- ncol(wine_baseline)
  wine_baseline[,top + 1]<- makeFakedata()
    names(wine_baseline)[top + 1] <- paste0("Fake ",j)
}
## runs the t-sne and saves the plot
ts3 <- Rtsne(wine_baseline[,-1], perplexity = 10, dims = 2, pca = FALSE)
tsdat3 <- as_tibble(ts3$Y);colnames(tsdat3) <- c("X1","X2")
tsdat3$Class <- factor(wine$Class)
B <- ggplot(tsdat3, aes(X1,X2)) + geom_point(aes(color = Class), size = 2) +
  ggtitle("T-SNE with 100 Noise Features") + theme_bw() +
  scale_color_viridis_d() + theme(plot.title = element_text(hjust = 0.5))


#Builds a 1000-variable feature set
for(j in 1:1000){
  top <- ncol(wine_baseline)
  wine_baseline[,top + 1]<- makeFakedata()
    names(wine_baseline)[top + 1] <- paste0("Fake ",j)
}
## runs the t-sne and saves the plot
ts3 <- Rtsne(wine_baseline[,-1], perplexity = 10, dims = 2, pca = FALSE)
tsdat3 <- as_tibble(ts3$Y);colnames(tsdat3) <- c("X1","X2")
tsdat3$Class <- factor(wine$Class)
```

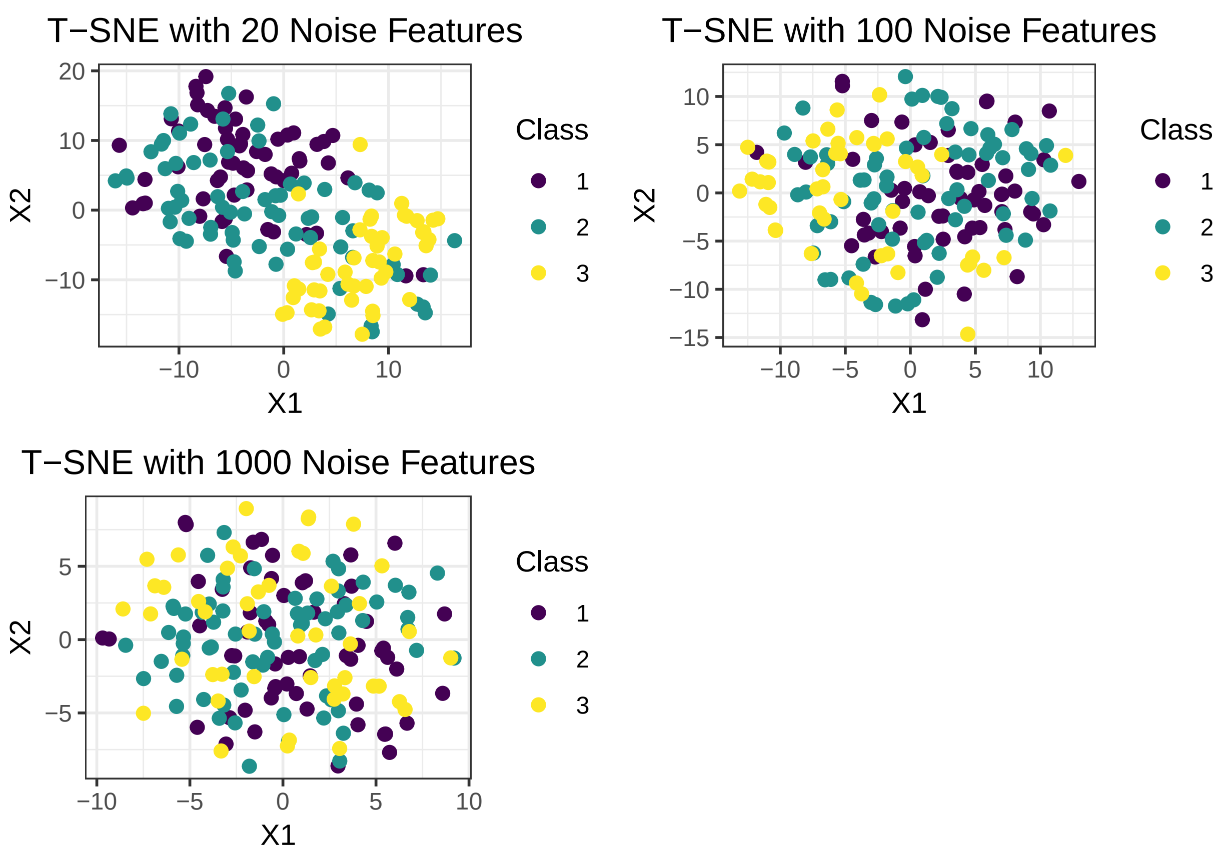


Figure 14: t-SNE runs with more noise features added in.

```r
C <- ggplot(tsdat3, aes(X1,X2)) + geom_point(aes(color = Class), size = 2) +
  ggtitle("T-SNE with 1000 Noise Features") + theme_bw() +
  scale_color_viridis_d() + theme(plot.title = element_text(hjust = 0.5))

ggarrange(A,B,C)
```

Alternatively, we might try simulating data with some weak, but non-zero, signal. Below, I simulate data with varying means (one for each Class of wine) with different (random) variances. A correlation plot shows that these 20 factors contain some signal, but it is weak relative to the structure in the known features, as noted in Figure 15. Figure 16 gives the resulting t-SNE map.

```r
#simulate means in each group (will be normalized later)
mu.1 <- 4
mu.2 <- 9
mu.3 <- 6
muvec <- rep(0, nrow(wine))
muvec <- ifelse(wine_baseline$Class == 1, mu.1,
       ifelse(wine_baseline$Class == 2,mu.2,
       mu.3))
fake_weak_sig <- matrix(0,nrow = nrow(wine), ncol =20 )
#samples the data with varying variance terms (so not all the same, but all still pretty big)
```

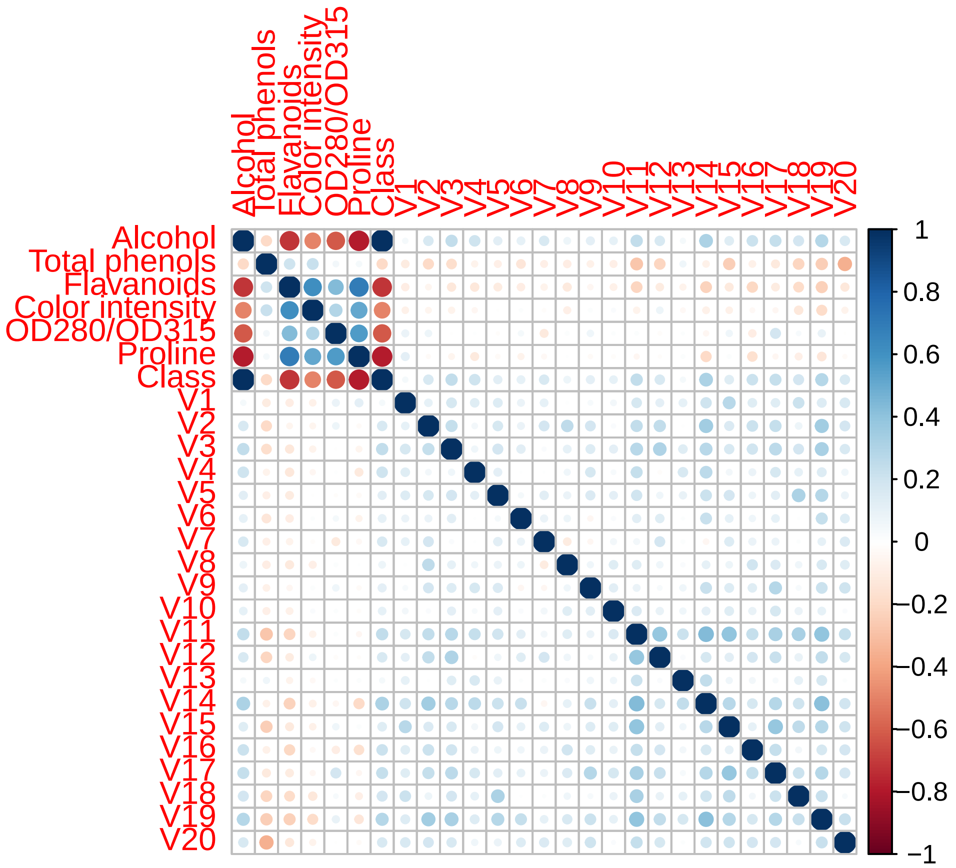


Figure 15: Correlation plot of 20 noise features with weak signal added in.

```r
for(j in 1:20){
  fake_weak_sig[,j] <- rnorm(nrow(wine),muvec, sd = sample(3:10,1))
}
fake_scale <- scale(fake_weak_sig) %>% as_tibble()

#rewrite the wine baseline object
wine_baseline <- dplyr::select(wine.scale,index.use)
wine_baseline$Class <- wine$Class

#combine the fake data with some signal to the new data
wine_baseline <- dplyr::select(wine.scale,index.use)
wine_baseline$Class <- wine$Class

new_wine <- cbind(wine_baseline, fake_scale)

#shows the "weak" signal in the simulated variables
corrplot(cor(new_wine))
```

```r
ts4 <- Rtsne(new_wine[,-1], perplexity = 10, dims = 2, pca = FALSE)
tsdat4 <- as_tibble(ts4$Y);colnames(tsdat4) <- c("X1","X2")
tsdat4$Class <- factor(wine$Class)
```

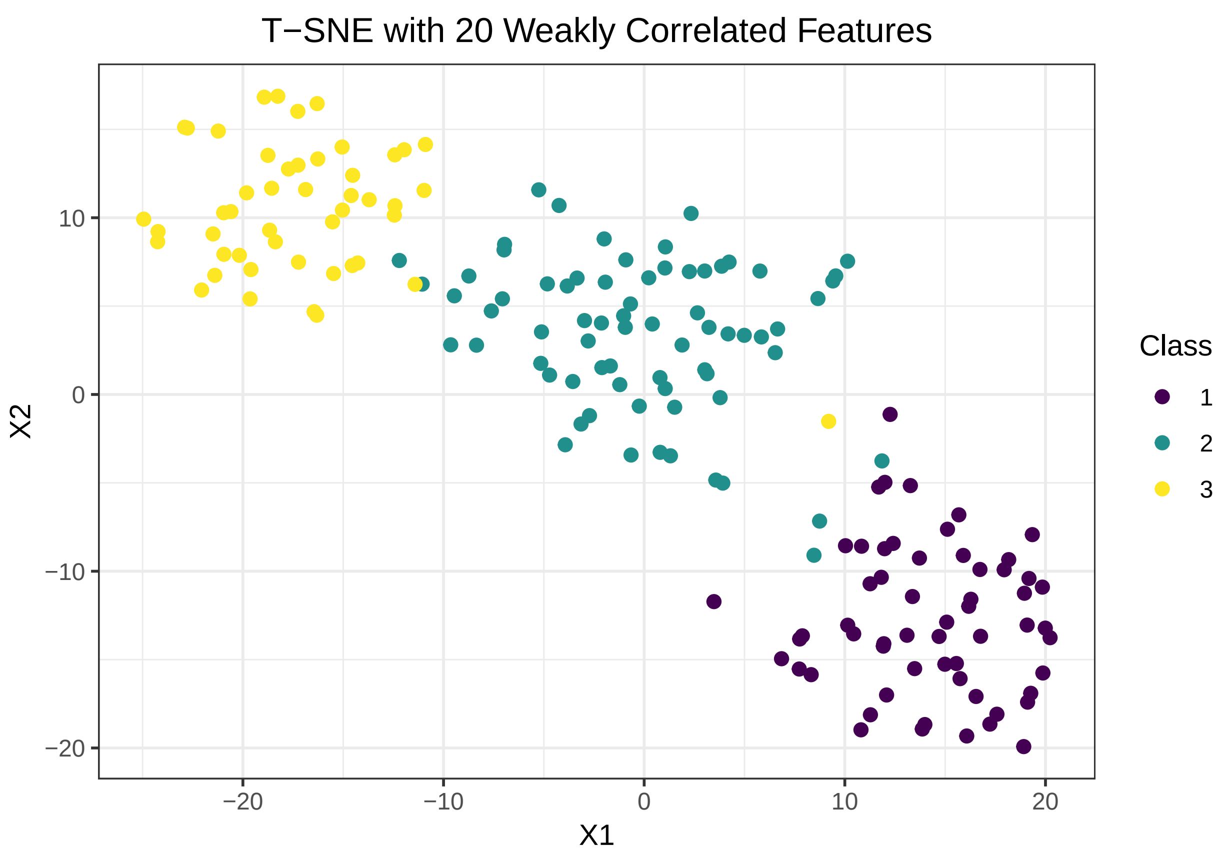


Figure 16: t-SNE with the weak signal features added in.

```r
ggplot(tsdat4, aes(X1,X2)) + geom_point(aes(color = Class), size = 2) +
  ggtitle("T-SNE with 20 Weakly Correlated Features") + theme_bw() +
  scale_color_viridis_d() + theme(plot.title = element_text(hjust = 0.5))
```

A sparse version of t-SNE may also be worth testing in a less-supervised setting. Returning to the dataset used by West(2003), we can attempt to see how t-SNE would operate in a P>n setting with a lasso-selected group of features vs. t-SNE on the entire dataset.

In some sense, I think this dataset contains an interesting analogue to some biological situations, as well as genetics-type problems. We do not know exactly what the differences in each observation are (the recipes were varied but not systematically), but we suspect that there are groups in the underlying data. A method by which features were selected using RSKC according to a lasso penalty and then those features were used in t-SNE is utilized below - the resulting map is given in Figure 17.

Strikingly, the solutions for both are quite similar. Figure 18 shows the values of the cost functions are shown for each iteration. I decided to investigate the cost functions a bit more, so I ran repeated t-SNE runs for the entire dataset and the lasso-selected group. In single runs it appears that the lasso-selected cost values are more stable than those of the full dataset; over multiple runs, it still seems to be the case. In running the code many times, the biggest spikes have come from the model run on the full dataset rather than the lasso-selected subgroup.

```r
red.alpha = rgb(10,0,1,4, maxColorValue = 10)
blue.alpha = rgb(0,1,10,5, maxColorValue = 10)
```

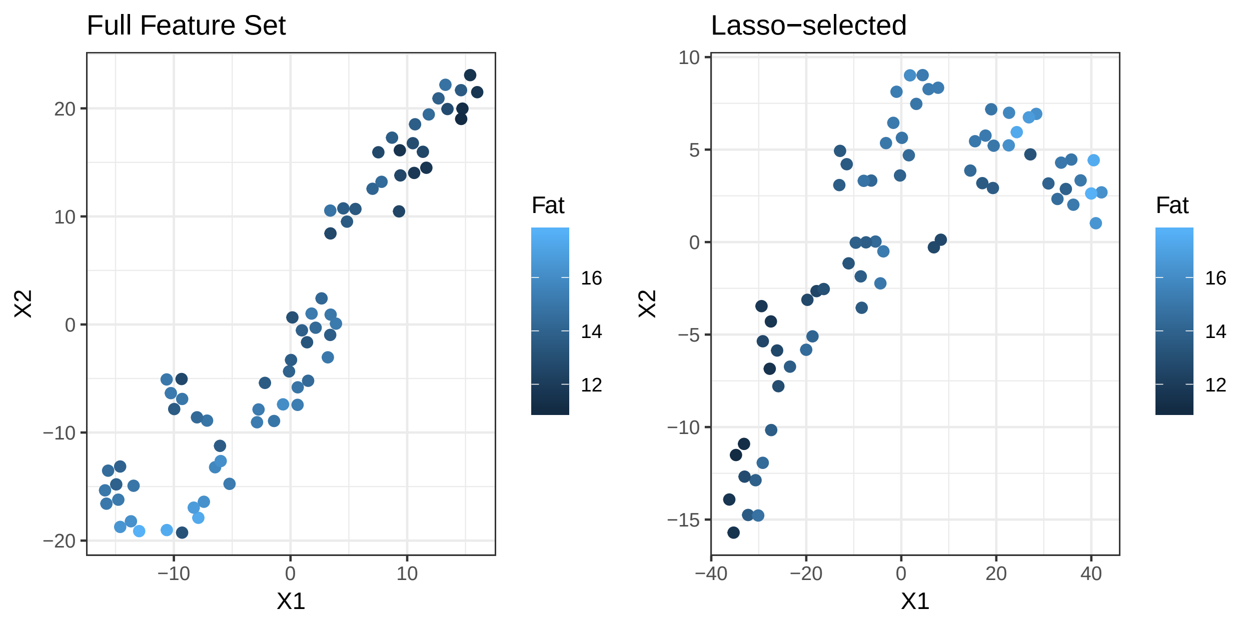


Figure 17: t-SNE maps for biscuit dough data.

```r
library(ppls)
data(cookie)
X <- scale(as.matrix(cookie[,1:700]))#NIR spectra
Y <- as.matrix(cookie[,701:704]) #I will likely ignore these

ts1 <- Rtsne(X, perplexity = 10, dims = 2, pca = FALSE)
tsdat <- as_tibble(ts1$Y);colnames(tsdat) <- c("X1","X2")
tsdat$Fat <- Y[,4]

p1 <- ggplot(tsdat, aes(X1,X2)) + geom_point(aes(color = Fat), size = 2) +
  ggtitle("Full Feature Set") + theme_bw()


## Now try doing some sort of regularization on this
rsk1 <- RSKC(X, ncl = 3, alpha = 0.3, L1 = 6)
#plot(rsk1$weights)
index.use <- which(rsk1$weights != 0)

X.lasso <- X[,index.use]
ts2 <- Rtsne(X.lasso, perplexity = 10, dims = 2, pca = FALSE)
tsdat2 <- as_tibble(ts2$Y);colnames(tsdat2) <- c("X1","X2")
tsdat2$Fat <- Y[,4]
p2 <- ggplot(tsdat2, aes(X1,X2)) + geom_point(aes(color = Fat), size = 2) +
  ggtitle("Lasso-selected") + theme_bw()

ggarrange(p1,p2)
```

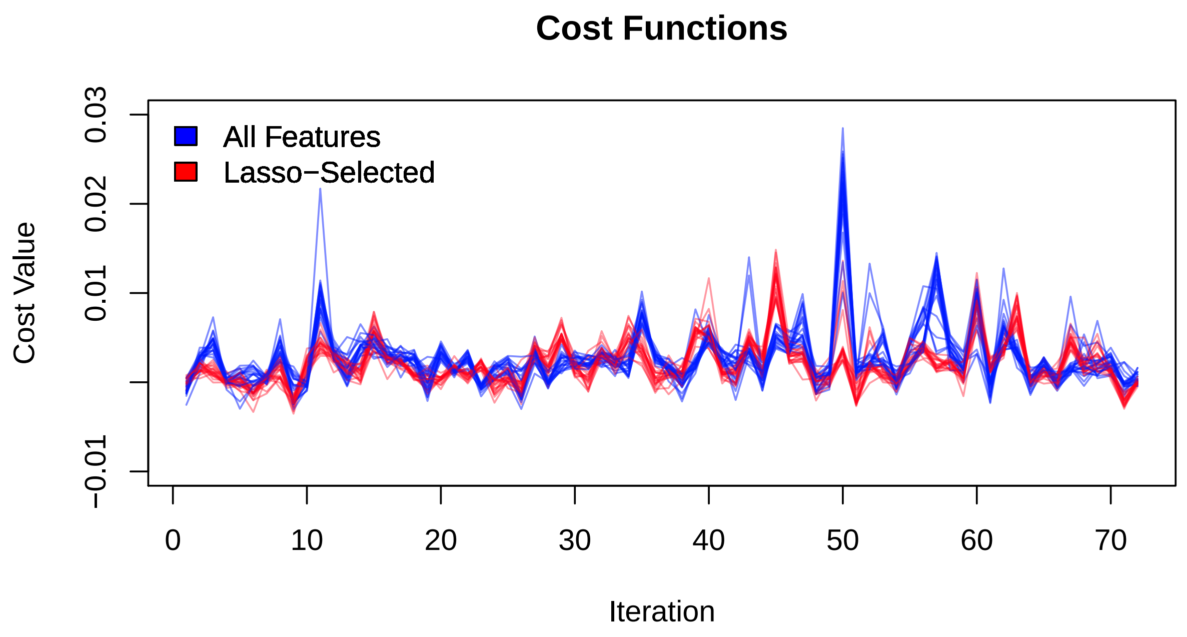


Figure 18: Cost functions for t-SNE runs on biscuit dough.

```
#looking at the KL cost function for both
plot(1:length(ts1$costs), ts1$costs,main = "Cost Functions", ylab = "Cost Value",
     xlab = "Iteration", type = "n", ylim = c(-0.01,.03))
for(j in 1:20){
ts1 <- Rtsne(X, perplexity = 10, dims = 2, pca = FALSE)
ts2 <- Rtsne(X.lasso, perplexity = 10, dims = 2, pca = FALSE)
lines(1:length(ts1$costs), ts1$costs, type = "l", col = blue.alpha)
lines(1:length(ts2$costs), ts2$costs, type = "l", col = red.alpha)
legend('topleft', fill = c('blue','red'),
       legend = c("All Features","Lasso-Selected"),bty = 'n')
}
```

In conclusion, it seems like we have a few interesting things to think about regarding t-SNE and regularization. This method only explores a simple use of regularization to inform t-SNE, and the simulations are a bit contrived because I did not re-regularize after adding the noise features to the dataset (I'm sure the lasso would have selected a few of the noise features as well). However, it seems that t-SNE breaks down in the presence of increasing numbers of uncorrelated features, and that it is possible to swamp a simulated dataset with enough noise that even t-SNE cannot reliably distinguish different observations based on features that drive data structure. Moreover, after returning to the dataset used by West(2003), I think it is possible that the cost function is less variable when additional noise features are removed. More work is needed to verify this, and with the t-SNE, I'm never entirely sure if I'm looking at something real or an artifact of the random seed selected for a given run.

Session Information

```
sessionInfo()
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18356)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
##  [1] stats4    grid      splines   stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] RSKC_2.4.2        flexclust_1.4-0   modeltools_0.2-22
##  [4] lattice_0.20-38   Rtsne_0.13        plsgenomics_1.5-2
##  [7] ppls_1.6-1.1      MASS_7.3-51.1     MVT_0.3
## [10] psych_1.8.12      fanc_2.2          effects_4.1-0
## [13] ggpubr_0.1.9      car_3.0-2         carData_3.0-2
## [16] emmeans_1.3.3     lmerTest_3.1-0    lme4_1.1-20
## [19] Matrix_1.2-15     beanplot_1.2      readxl_1.3.1
## [22] corrplot_0.84     knitr_1.22        tidyr_0.8.2
## [25] pander_0.6.3      tibble_2.1.1      magrittr_1.5
## [28] dplyr_0.7.8       ggplot2_3.1.0     readr_1.1.1
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-137        numDeriv_2016.8-1   tools_3.5.2
##  [4] R6_2.3.0            lazyeval_0.2.1      colorspace_1.3-2
##  [7] nnet_7.3-12         withr_2.1.2         tidyselect_0.2.5
## [10] mnormt_1.5-5        curl_3.2            compiler_3.5.2
## [13] sandwich_2.5-0      labeling_0.3        scales_1.0.0
## [16] mvtnorm_1.0-8       stringr_1.3.1       digest_0.6.18
## [19] foreign_0.8-71      minqa_1.2.4         rmarkdown_1.11
## [22] rio_0.5.16          RhpcBLASctl_0.18-205 pkgconfig_2.0.2
## [25] htmltools_0.3.6     highr_0.7           maps_3.3.0
## [28] rlang_0.3.1         bindr_0.1.1         zoo_1.8-4
## [31] zip_2.0.0           dotCall64_1.0-0     Rcpp_1.0.0
## [34] munsell_0.5.0       abind_1.4-5         stringi_1.3.1
## [37] multcomp_1.4-8      yaml_2.2.0          plyr_1.8.4
## [40] parallel_3.5.2      forcats_0.4.0       crayon_1.3.4
## [43] cowplot_0.9.3       haven_2.1.0         hms_0.4.2
## [46] pillar_1.3.1        boot_1.3-20         estimability_1.3
## [49] reshape2_1.4.3      codetools_0.2-15    glue_1.3.0
## [52] evaluate_0.12       data.table_1.11.8   spam_2.2-0
## [55] nloptr_1.2.1        cellranger_1.1.0    gtable_0.2.0
## [58] purrr_0.2.5         assertthat_0.2.0    xfun_0.4
## [61] openxlsx_4.1.0      xtable_1.8-3        survey_3.35-1
## [64] coda_0.19-2         class_7.3-14        viridisLite_0.3.0
## [67] survival_2.43-3     fields_9.6          bindrcpp_0.2.2
## [70] TH.data_1.0-9
```

References:

PAUL REFERENCES

- Carvalho, C. M., Chang, J., Lucas, J. E., Nevins, J. R., Wang, Q., & West, M. (2008). High-Dimensional Sparse Factor Modeling: Applications in Gene Expression Genomics. Journal of the American Statistical Association, 103(484), 1438-1456.
- Everitt, B., & Hothorn, T. (2011). An introduction to applied multivariate analysis with R. New York: Springer.
- Hildreth, L. (2013). "Residual Analysis for Structural Equation Modeling." Iowa State University, lib.dr.iastate.edu/etd/13400/
- Hotelling, H. (1933). Analysis of a complex statistical variables into principal components. J. Educ. Psychol., 24, 414-441.
- Herman Chernoff (1973). "The Use of Faces to Represent Points in K-Dimensional Space Graphically" (PDF). Journal of the American Statistical Association. American Statistical Association. 68 (342): 361–368. doi:10.2307/2284077. JSTOR 2284077. Archived from the original (PDF) on 2012-04-15.
- Meinshausen, N. (2007). Relaxed Lasso. Computational Statistics & Data Analysis, 52(1), pp.374-393.
- P.J. Brown, T. Fearn, and M. Vannucci (2001) Bayesian Wavelet Regression on Curves with Applications to a Spectroscopic Calibration Problem. Journal of the American Statistical Association, 96, pp. 398-408.
- B.G. Osborne, T. Fearn, A.R. Miller, and S. Douglas (1984) Application of Near-Infrared Reflectance Spectroscopy to Compositional Analysis of Biscuits and Biscuit Dough. Journal of the Science of Food and Agriculture, 35, pp. 99 - 105.
- Kondo, Y., Salibian-Barrera, M., and Zamar, R. (2016). RSKC: Robust and Sparse K-Means Clustering in R. Journal of Statistical Software, 72, Issue 5. doi: 10.18637/jss.v072.i05.
- Kuhn, Max, and Kjell Johnson. Applied Predictive Modeling. Springer, 2016.
- Hui Zou, Trevor Hastie & Robert Tibshirani (2006) Sparse Principal Component Analysis, Journal of Computational and Graphical Statistics, 15:2, 265-286, DOI: 10.1198/106186006X113430
- Tanaka, Y., Watadani, S. and Moon, S. H. (1991). Influence in covariance structure analysis: with an application to confirmatory factor analysis. Communications in Statistics– Theory and Methods, 20(12), 3805–3821.

MARK REFERENCES

- Hirose, K., and Yamamoto, M.(2015) Sparse estimation via nonconcave penalized likelihood in factor analysis model, *Statistical Computing*, 25, 863-875.
- Hirose, K., Yamamoto, M., and Nagata, H. (2016) fanc: Penalized Likelihood Factor Analysis via Nonconvex Penalty. R package version 2.2.
- Jacobucci, R. (2017) regsem: Regularized Structural Equation Modeling, arXiv:1703.08489v2.
- Jacobucci, R. , Grimm, K., Brandmaier, A., Serang, S. and Kievit, R. (2019) regsem: Regularized Structural Equation Modeling. R package version 1.2.3.

- Jacobucci, R., Grimm, K., and McArdle, J. (2016) Regularized Structural Equation Modeling, *Structural Equation Modeling: A Multidisciplinary Journal*, 23, 555-566.
- Krijthe, J. (2015) Rtsne: T-Distributed Stochastic Neighbor Embedding using a Barnes-Hut Implementation, URL: https://github.com/jkrijthe/Rtsne
- van der Maaten, L. (2014) Accelerating t-SNE using Tree-Based Algorithms, *Journal of Machine Learning Research*, 15, 3221-3245.
- van der Maaten, L. and Hinton, G. (2008) Visualizing High-Dimensional Data Using t-SNE, *Journal of Machine Learning Research*, 9, 2579-2605.
- Witten, D. and Tibshirani, R. (2010) A Framework for Feature Selection in Clustering, *Journal of the American Statistical Association*, 105:490, 713-726.
- Witten, D. and Tibshirani, R. (2018) sparcl: Perform Sparse Hierarchical Clustering and Sparse K-Means Clustering. R package version 1.0.4.
- Yamamoto, M., Hirose, K., and Nagata, H. (2017) Graphical tool of sparse factor analysis, *Behaviormetrika*, 44, 229-250.