Impact of Yield Curves on Equity Sectors

Contents

Introduction and Motivation

For over 30 years, long-run interest rates have been on a slow, steady decline. Now, possibly, they have finally turned a corner. The implications for the American economy are substantial. Whole sectors of the economy have flourished only in a declining rate environment, and there is no empirical evidence of how they will function when long-run rates rise. It has become the received wisdom that asset classes which typically perform well when rates are declining, such as houses, are a safe and low-risk investment, even after the 2008 financial crisis.

The first place to look for the impending sea-change in the rate environment is the yield curve, which shows the current state of market expectations for interest rates over different time periods. Shifts in the yield curve have the most direct impact on the US economy by changing the cost and character of corporate financing.

As different industries rely on a different financing mix, and corporate bond issuance is hardly the only mechanism by which the yield curve can impact US corporations, we will investigate the relationship between changes in the yield curve and different corporate sectors, as defined by Kenneth French of Fama/French Factor Model fame.

We hope to provide a resource to aid in decision-making for portfolio managers, as well as for scenario analyis for corporate decision-makers.

The full source code of this project is available online at github.

First, we should examine the raw data.

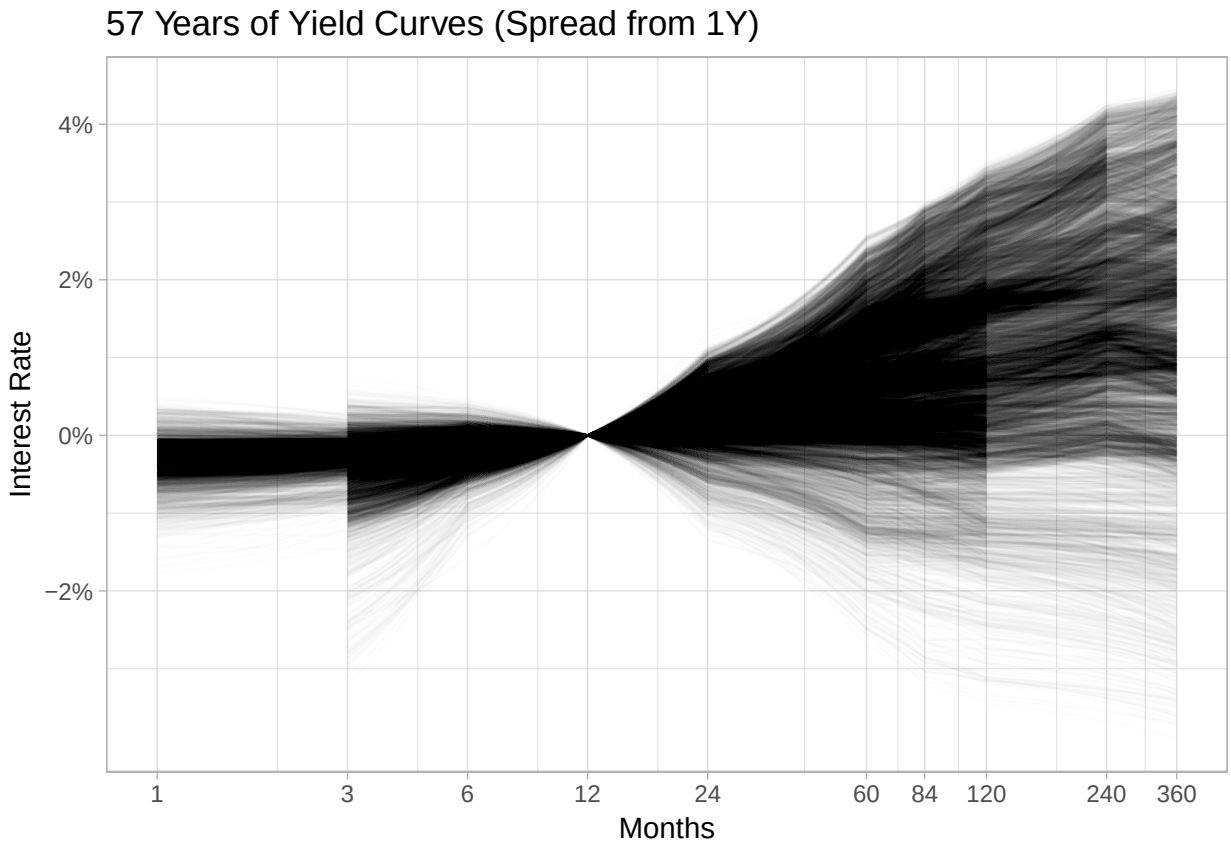
Yield Curve

We will characterize the yield curve using nominal, constant maturity bond rate data provided by the (St. Louis Fed)[https://fred.stlouisfed.org/].

It is typical to look at only one or two spreads between time periods to describe the yield curve, with 10Y-2Y spreads being perhaps the most popular. However, the true yield curve is more complex than that.

```
yc %>%#this is faster than moving from long-to-wide, subtracting the column, then moving back
  group_by(duration) %>%
  arrange(date) %>%
  fill() %>%
```

```
  ungroup() %>%
  left_join(yc %>%
              filter(duration==1) %>%
              transmute(date=date,`1`=value),
            by="date") %>%
  mutate(value=value-`1`) %>%
  select(-`1`) %>%
  mutate(duration=ceiling(12*as.numeric(duration))) %>%
  ggplot(aes(x=duration, y=value, group=date)) +
  geom_line(alpha=0.01) +
  coord_trans(x="log") +
  scale_x_continuous(breaks=unique(ceiling(12*yc$duration)))+
  scale_y_continuous(labels=scales::percent_format(accuracy=1)) +
  xlab("Months") + ylab("Interest Rate") +
  ggtitle("57 Years of Yield Curves (Spread from 1Y)")
```

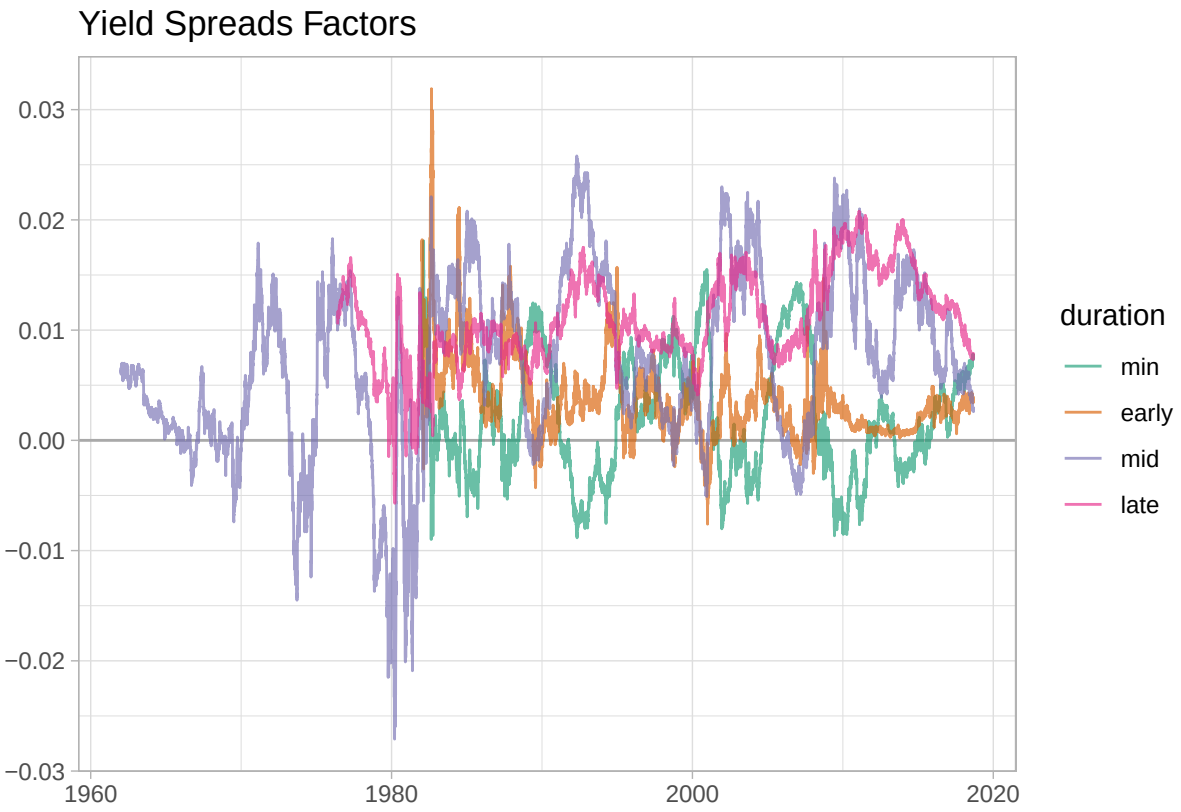


We can see that it is highly unusual for the long end of the yield curve to be inverted. It is *not* unusual for the short end of the curve to contain "kinks", in which the 3 or 6 month rates are higher than the 1, 2, or even 5 year rates, before the time-value of money supercedes expectations of future conditions.

We propose to characterize the Yield Curve in any given period as sequence of four spreads. We could fit a cubix spline model to admit early kinks inside a generally positive-sloping curve, but the coefficients of that fit would not be directly interpretable. By choosing several spreads, we will need to account for collinearity in any predictive regressions.

```
yc %>%
  group_by(date) %>%
  mutate(value=value-mean(value, na.rm=T)) %>%
```

```
  ungroup() %>%
  mutate(duration=ceiling(12*duration)) %>%
  spread(duration, value) %>%
  transmute(date=date,
         min=`6` - mean(c(`1`,`3`), na.rm=T),
         early=`12` - `3`,
         mid=`60`-`12`,
         late=mean(c(`120`,`240`,`360`), na.rm=T)-`24`) %>%
  gather(duration, value, -date) %>%
  mutate(duration=fct_relevel(duration, c("min","early","mid","late"))) %>%
  drop_na() ->
  yc_spreads
yc_spreads %>%
  ggplot(aes(x=date, y=value, col=duration)) +
  geom_hline(yintercept = 0, col="darkgrey") +
  geom_line(alpha=0.65) +
  scale_color_brewer(palette="Dark2") +
  xlab("") + ylab("") +
  ggtitle("Yield Spreads Factors")
```



The first date for which all of these spreads are available is 1982-01-04. We don't want to jettison the late 70's oil crisis, however, as that period represents a substantial portfolio of our body of mid-range negative interest rates.

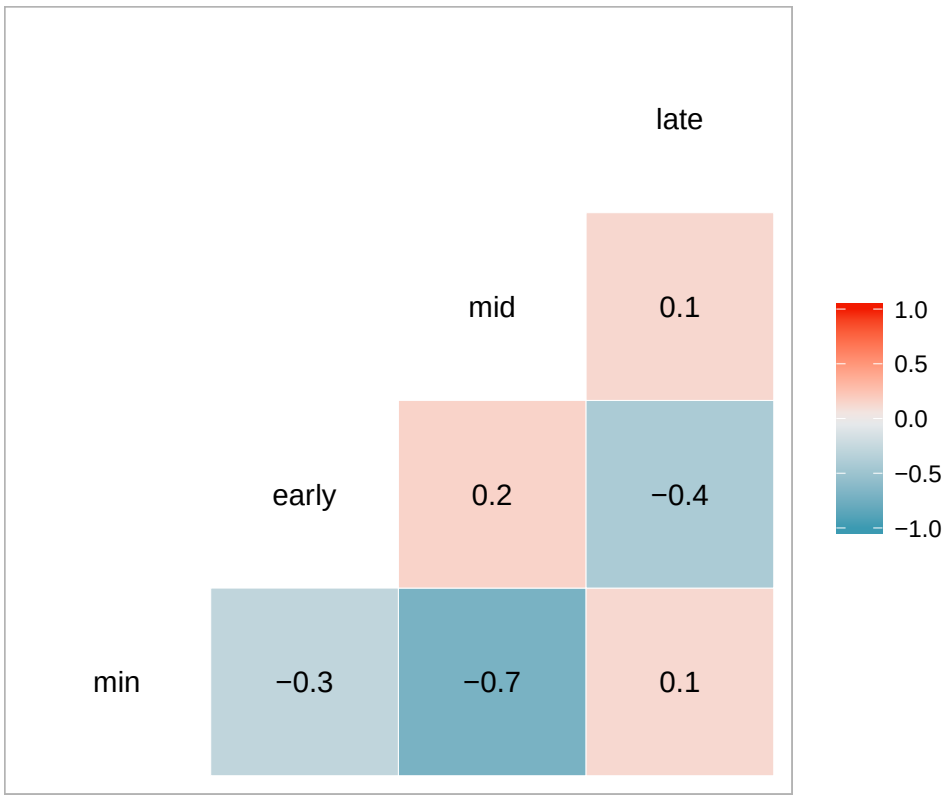
As we want to fit the impact of *changes* in the yield spread against equity prices, we will need to take the difference of the above factors. We'll also take the opportunity to scale at this point, as the MCMC models we'll be fitting later do better when the standard deviation is close to 1. We can multiply by 10,000 and refer to rate movements in terms of basis points.

```
library(GGally)
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```
yc_spreads %>%
  group_by(duration) %>%
  arrange(date) %>%
  mutate(value=c(NA,diff(value)*1e4)) %>% #multiplying by 10000 because the values are getting kind of
  ungroup() ->
  yc_spread_diff

yc_spread_diff %>%
  spread(duration, value) %>%
  select(-date) %>%
  ggcorr(label=T) +
  ggtitle("Collinearity in differences of yield curve spreads")
```



The highly negative correlation between changes in spreads in the half-year end ("min") and the 5Y - 2Y ("mid")

What are the typical values for daily changes in these yield curve indexes?

```
yc_spread_diff %>%
  group_by(duration) %>%
```

```
  drop_na() %>%
  summarize(min=min(value),
            `0.05`=quantile(value, 0.05),
            `0.10`=quantile(value, 0.10),
            `0.32`=quantile(value, 0.32),
            `0.5`=quantile(value, 0.5),
            `0.67`=quantile(value, 0.67),
            `0.90`=quantile(value, 0.9),
            `0.95`=quantile(value, 0.95),
            kurtosis=kurtosis(value),
            sd=sd(value)
            ) %>%
  knitr::kable(digits=2)
```

duration	min	0.05	0.10	0.32	0.5	0.67	0.90	0.95	kurtosis	sd
min	-39.87	-5.75	-4	-1.25	0.1	1.30	3.9	5.37	25.73	3.96
early	-67.00	-7.00	-5	-1.00	0.0	1.00	5.0	7.00	22.75	5.31
mid	-50.00	-7.00	-5	-1.00	0.0	1.00	5.0	7.00	13.01	5.20
late	-39.46	-4.37	-3	-1.00	0.0	0.89	3.0	4.24	13.03	3.08

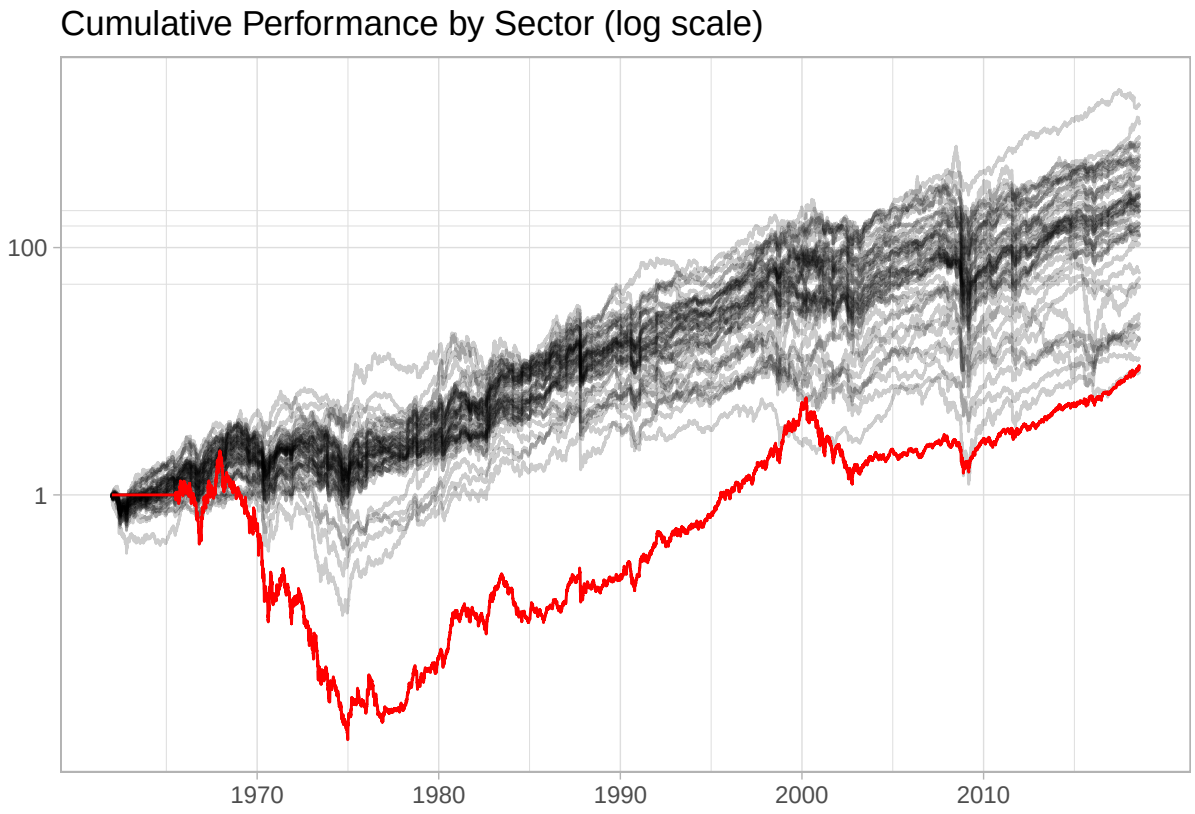
They are leptokurtic, as befits a time series indirectly set by committee, but surprisingly centered at 0. We would have expected a negative median given the last 30 years.

Sector Indexes

Kenneth French has undertaken the Herculean task of assembling economic sector return indexes for US stocks, with data going back to before the Great Depression. We have 49 sectors, which may become unwieldy in the final model. We can use Bayesian shrinkage via multilevel modeling to mitigate the prevalence of Type I errors that would arise from fitting separate regressions for each sector.

First, let's take a look at the cumulative returns of the sectors, starting from 1962, where our Yield Curve data starts.

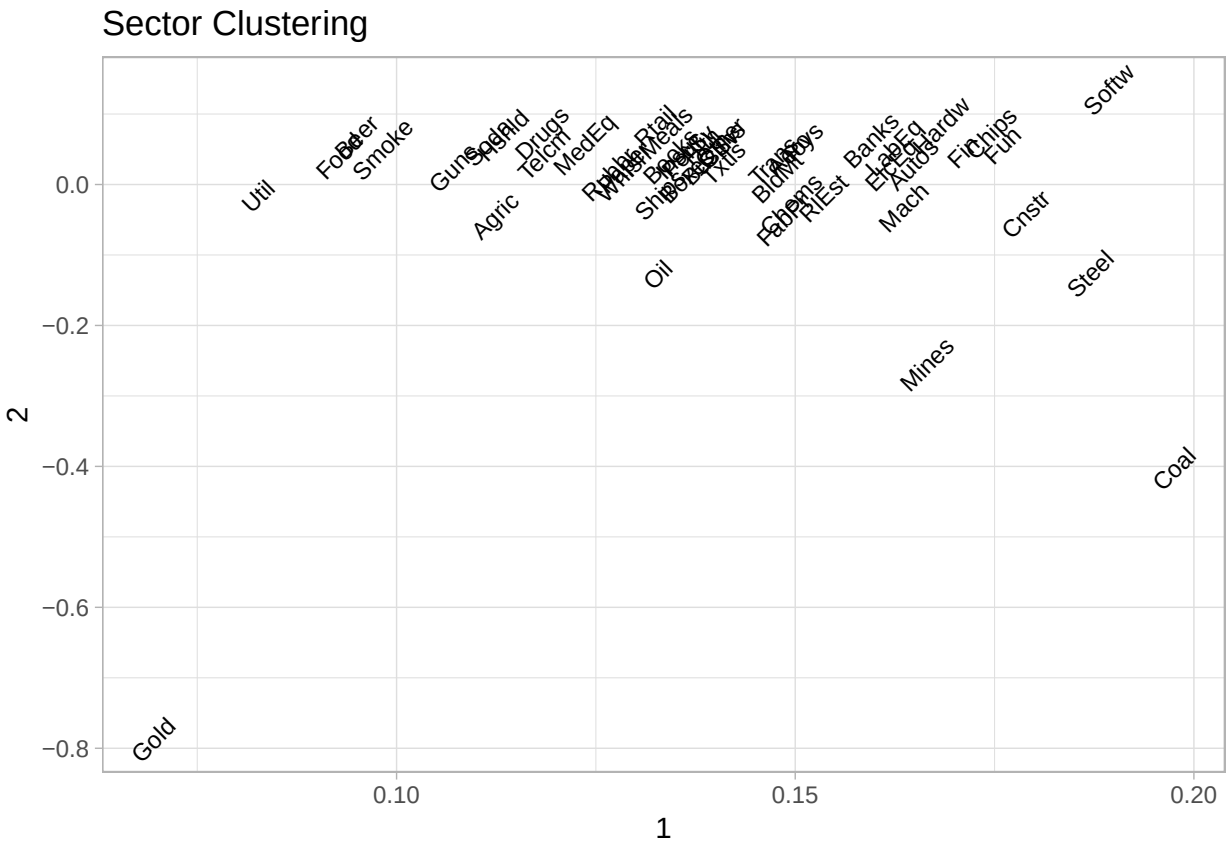
```
sector_returns %>%
  group_by(sector) %>%
  arrange(Date) %>%
  replace_na(list(avg_return=0)) %>%
  mutate(cum_return=cumprod(1+avg_return)) %>%
  ungroup() ->
  cum_sr

cum_sr %>%
  ggplot(aes(x=Date,y=cum_return, group=sector)) +
  geom_line(alpha=0.2) +
  coord_trans(y="log") +
  scale_y_continuous(breaks=c(0,1,1e2,1e4,1e6,1e8)) +
  geom_line(data=filter(cum_sr, sector=="Softw"), col="red") +
  theme_light() +
  ggtitle("Cumulative Performance by Sector (log scale)") + xlab("") + ylab("")
```



A rising tide lifts all boats, apparently. The long-term view also serves to emphasize the importance of starting points. The red-lined sector, the worst performer for nearly the entire history shown, is Software. We started the cumulative series shortly before the late 60's tech bubble collapsed, and an investment of $1 took until the mid-90's to recover.

Are some sectors more related than others? Let's look at the first two principal components.

```
sector_returns %>%
  spread(sector, avg_return) %>%
  drop_na() %>% # we need a contiguous dataset to perform PCA.
  select(-Date) %>%
  prcomp() %>%
  broom::tidy(matrix="v") %>%
  filter(PC<3) %>%
  spread(PC, value) %>%
  ggplot(aes(x=`1`, y=`2`)) +
  geom_text(aes(label=column), angle=45, size=3) +
  ggtitle("Sector Clustering")
```



It looks like the first principal component is approximately dividing cyclical from non-cyclical sectors, while the second might be picking out heavy industry, i.e. mining, commodities, agriculture, and construction, versus services and manufacturing higher up the value chain.

Models

We will fit a model of the impact of the state of the yield curve on equity returns. In a world with an efficient market, we might expect equity prices to reflect changes in the yield curve immediately, even though the impact of changes in rates on the cost of capital, or some other relevant effect might be expected to take longer to actually hit the firms' bottom lines. Hence we are comfortable regressing the contemporaneous change in the yield curve against the returns.

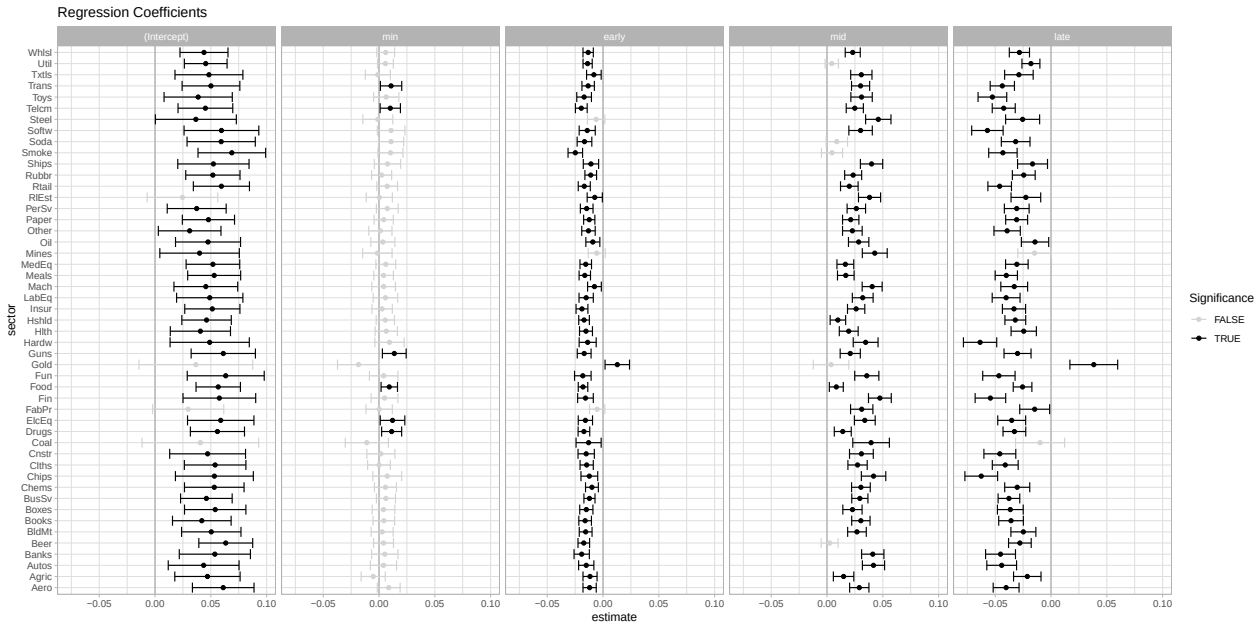
Regression on Sectors Separately

First, let's look at a basic multivariate regression on each sector independently. To keep things simple, we will only look at the contiguous data set, which starts in 1982.

N.B. We will also scale the sector returns by 100 at this point, again because later models will fit more easily if the standard deviation is close to 1. So we will be regressing basis points against percents.

```
yc_spread_diff %>%
  spread(duration, value) %>%
  left_join(sector_returns %>%
              mutate(avg_return=avg_return*100), #scale sector returns
            by=c("date"="Date")) ->
model_data
```

```
model_data %>%
  drop_na() %>%
  split(.$sector) %>%
  map_dfr(~lm(avg_return ~ min + early + mid + late, data=.x) %>%
            broom::tidy(conf.int=T),
          .id="sector") %>%
  mutate(term=fct_relevel(term, c("(Intercept)", "min","early","mid","late"))) ->
  fit_lm

fit_lm %>%
  arrange(sector) %>% # Tufte would say I should sort by some value, but I think in this case it's more
  mutate(significant=p.value<0.05) %>%
  ggplot(aes(x=estimate, y=sector, col=significant)) +
  facet_grid(~term) +
  geom_vline(aes(xintercept=0), col="darkgrey") +
  geom_errorbarh(aes(xmin=conf.low, xmax=conf.high)) +
  geom_point() +
  guides(col=guide_legend(title="Significance")) +
  scale_color_manual(values=c("TRUE"="black", "FALSE"="lightgrey")) +
  ggtitle("Regression Coefficients")
```



While the intercept looks significant, the other coefficients include a difference of 2 orders of magnitude between the change in yield curve and the equity returns, so it is actually basically zero. The very short end of the curve seems to be insignificant. Therefore, we can drop these two from later models without losing much information.

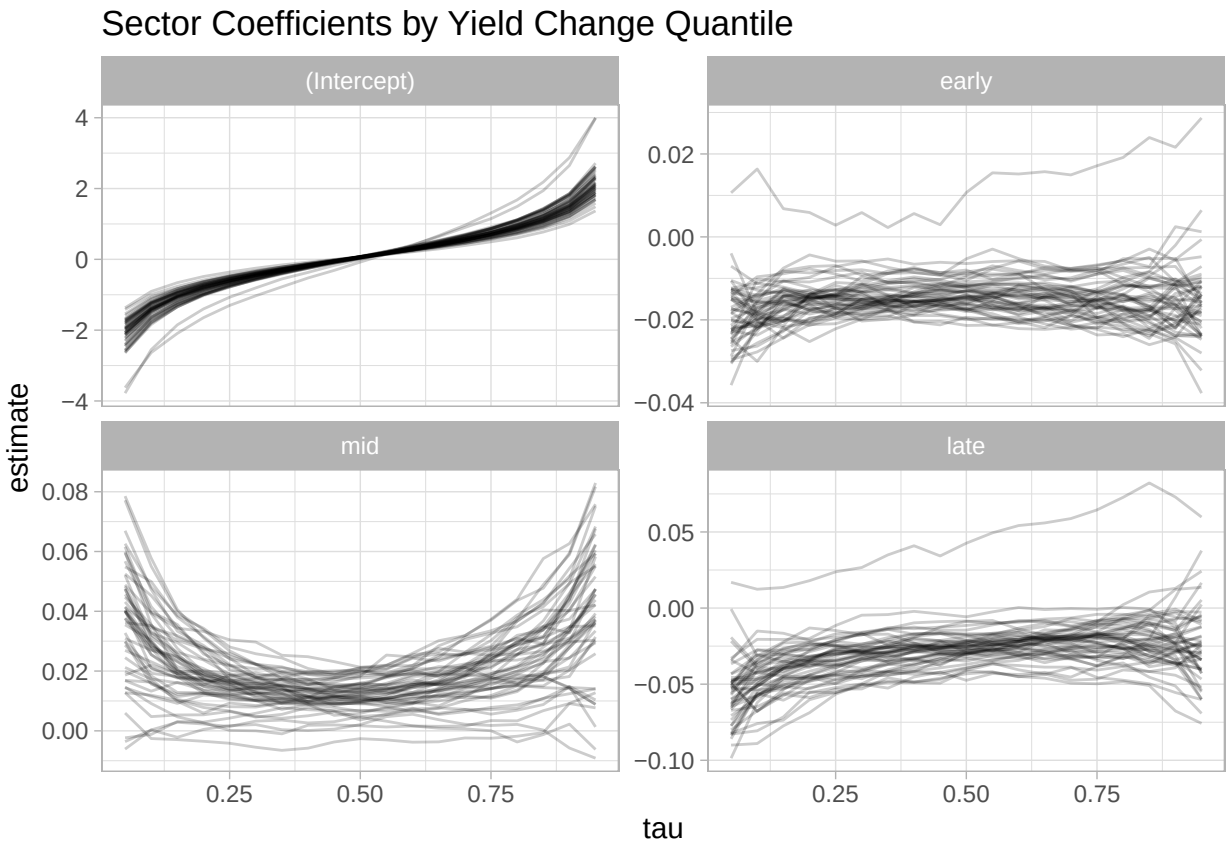
We can see confirmation of the Principal Components Analysis that placed gold miners as an extreme outlier; Gold alone benefits from positive changes in the long end of the yield curve, possibly because these signify inflation. It's also worth noting that the average day for all of these sectors has been positive.

Quantile Regression

Is the effect of spread changes on equity returns linear, or are there significant differences depending on the severity of the change?

```
yc_spread_diff %>%
  spread(duration, value) %>%
  left_join(sector_returns %>%
              mutate(avg_return=avg_return*100), #scale sector returns
            by=c("date"="Date")) ->
model_data

model_data %>%
  drop_na() %>%
  split(.$sector) %>%
  map_dfr(~rq(avg_return ~ early + mid + late, data=.x,
              tau=1:19/20) %>%
            broom::tidy(conf.int=T),
          .id="sector") %>%
  mutate(term=fct_relevel(term, c("(Intercept)", "early","mid","late"))) ->
  fit_rq
```

```
fit_rq %>%
  ggplot(aes(x=tau, y=estimate, group=sector)) +
  facet_wrap(~term, scale="free_y") +
  geom_line(alpha=0.2) +
  ggtitle("Sector Coefficients by Yield Change Quantile")
```



We can see a clear sensitivity to tail values for the mid-level (10Y - 2Y) changes in the yield curve, at least for some sectors. Coefficients in this model are swamped by the intercept, which is two orders of magnitude more important than the effects we're measuring. We generally do not expect that yield curve changes will be the primary drivers of equity returns, so perhaps this is not surprising.

HMC Model Version 1: Full pooling, missing data imputation

We have much more data for the mid curve factor than for the others. It would be a pity to throw that away. With a generative model, however, we can accomodate the ragged array. There are two options: either we draw the missing factor data from a distribution approximating the full extant data, or we fit a preliminary model regressing unknown portions of the yield curve against what data we do have, and fill missing periods on the basis of that fit. In the first case, we will have made the incorrect assumption that the yield curve changes are i.i.d with respect to time, and in the second, unless we are careful to account for the leptokurtic structure of the yield curve differentials, we will have imposed normality on the missing distributions. It's debateable, but I think the first option is more conservative. First, we'll need to fit a Student T distribution to the factors so we know what to work with.

```
// Fit the change in yield curve spreads to a student t distribution
// Following Stan Manual 2.17, pp 231-232.
data {
  int<lower=0> N; // # total observations
  int<lower=0> K; // # of factors
  vector[N] x; // observations
  int s[K]; // group sizes
}
parameters {
  vector[K] mu;
  vector<lower=0>[K] sigma;
  vector<lower=0>[K] nu; // degrees of freedom
}
model {
  int pos;
  nu ~ gamma(2,0.1);
  mu ~ normal(0, 0.01);
  sigma ~ cauchy(0, 2);
  pos = 1;
  for (k in 1:K) {
    segment(x, pos, s[k]) ~ student_t(nu[k], mu[k], sigma[k]);
    pos = pos + s[k];
  }
}
```

Now we've compiled our generative model, we can fit it to the data.

```
yc_spread_diff %>%
  drop_na() %>%
  arrange(duration) %>%
  mutate(duration=c(0, diff(as.numeric(duration)))) %>%
  select(-date) %>%
  rename(x=value, s=duration) %>%
  as.list() ->
  stan_data
stan_data <- within(stan_data,{
  s=diff(c(1,which(s==1))) #find indexes of change points, then calc differences to get variable length
```

```
  N=length(x) # # of observations
  K=length(s) # # of groups
})

yc_t_fit <- sampling(yc_diff_model, data=stan_data)
```

Check the distribution of the fit parameters.

```
library(bayesplot)
```

```
## This is bayesplot version 1.5.0
```

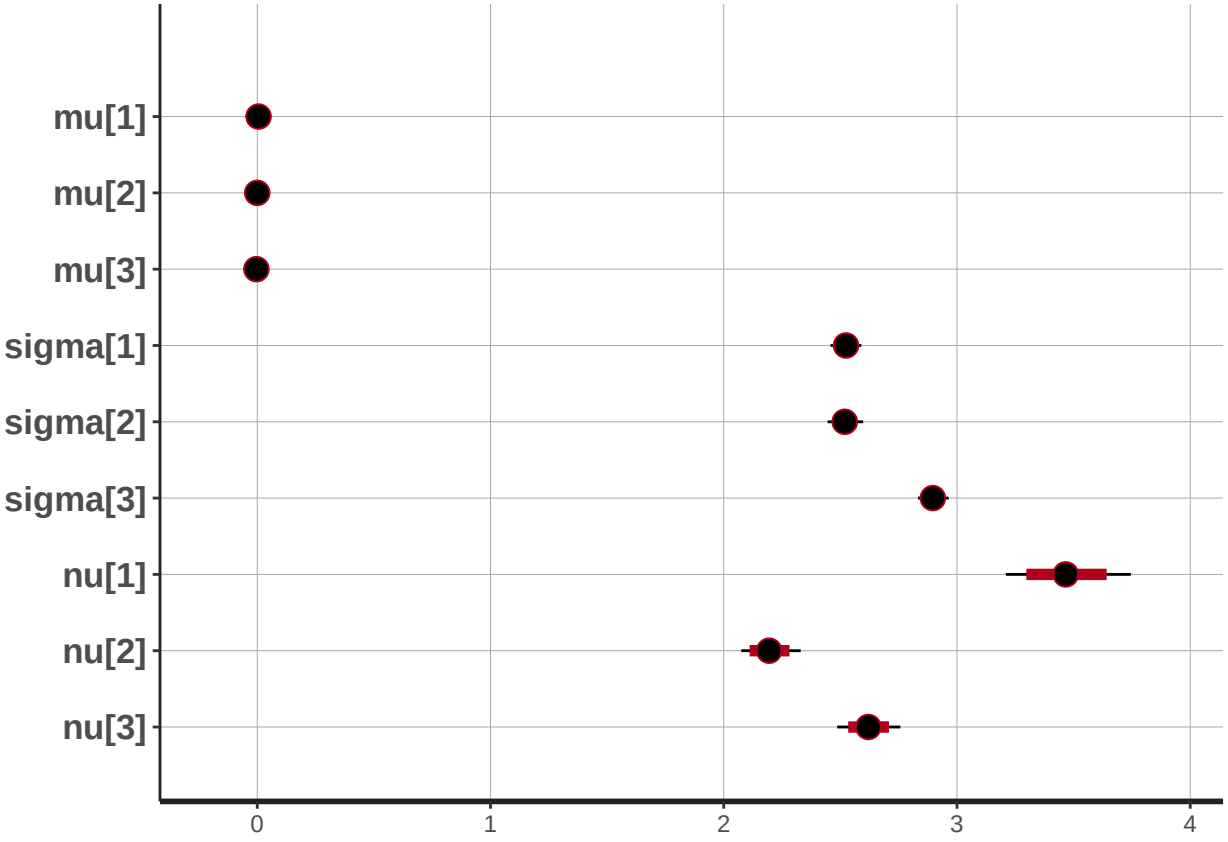
```
## - Plotting theme set to bayesplot::theme_default()
```

```
## - Online documentation at mc-stan.org/bayesplot
```

```
t_params <- get_posterior_mean(yc_t_fit)[,"mean-all chains"]
plot(yc_t_fit)
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```



These are fairly precise; we could have just run this model past the MLE optimizer rather than fitting by simulation. We'll go ahead and take the median estimates as priors in the regression model.

We'll start simple and look at a regression model on a single equity sector first.

```
// adapted from stan manual p182
data {
  int<lower=0> N; // total # of days
```

```
  int<lower=0> K; // total # of factors
  int<lower=0> N_mis[K]; // number of missing points for each factor

  // parameters for student_ts of missing factor data
  real param_mu[K];
  real<lower=0> param_nu[K];
  real<lower=0> param_sigma[K];

  vector[N-N_mis[1]] obs_early; // observed early factor
  vector[N-N_mis[2]] obs_mid; // observed mid factor
  vector[N-N_mis[3]] obs_late; // observed late factor
  real y[N]; // equity sector returns
}
parameters {
  vector[N_mis[1]] mis_early;
  vector[N_mis[2]] mis_mid;
  vector[N_mis[3]] mis_late;
  vector[K] beta; // OLS coefficients
  real<lower=0> sigma; // error term for y
}
transformed parameters {
  matrix[N,K] x;
  x[:N_mis[1],1] = mis_early;
  x[N_mis[1]+1:,1] = obs_early;
  x[:N_mis[2],2] = mis_mid;
  x[N_mis[2]+1:,2] = obs_mid;
  x[:N_mis[3],3] = mis_late;
  x[N_mis[3]+1:,3] = obs_late;
}
model {
  for(k in 1:K) {
    x[,k] ~ student_t(param_nu[k], param_mu[k], param_sigma[k]);
  }
  beta ~ normal(0, 0.1);
  sigma ~ cauchy(0, 1);
  y ~ normal(x*beta, sigma);
}
```

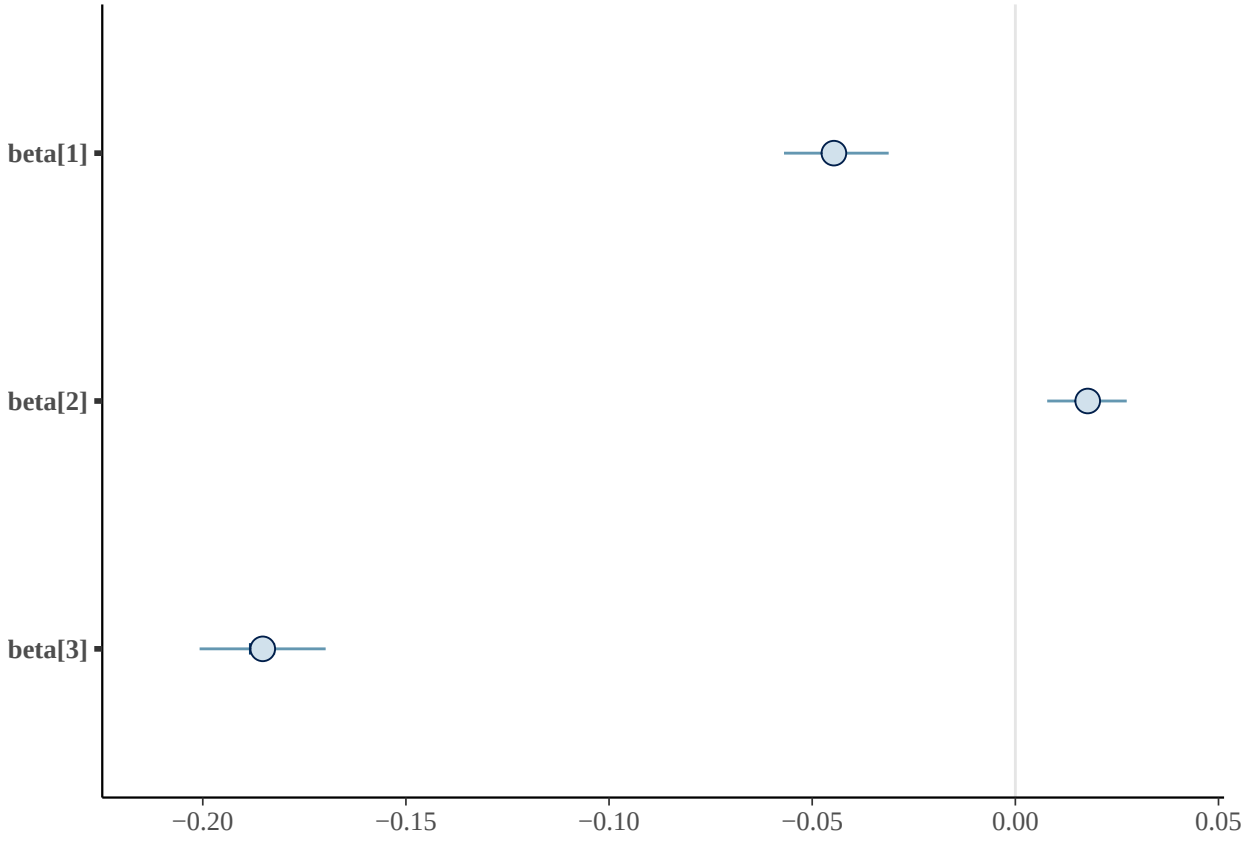
Let's look at Software:

```
sector_returns %>%
  filter(sector=="Softw") %>%
  drop_na() %>%
  left_join(yc_spread_diff %>% spread(duration, value),
            by=c("Date"="date")) %>%
  select(-Date,-sector,-min) ->
  tmp_data
singleY_mi_data <- list(N=nrow(tmp_data),
                        K=3,
                        N_mis=tmp_data %>% summarize_at(vars(early, mid, late), ~ sum(is.na(.))) %>% as
                        param_mu=t_params[grep("mu",names(t_params))],
                        param_nu=t_params[grep("nu",names(t_params))],
                        param_sigma=t_params[grep("sigma",names(t_params))],
                        obs_early=tmp_data %>% drop_na(early) %>% pull("early"),
                        obs_mid=tmp_data %>% drop_na(mid) %>% pull("mid"),
```

```
                        obs_late=tmp_data %>% drop_na(late) %>% pull("late"),
                        y=pull(tmp_data,"avg_return")
                        )
singleY_mi_fit <- sampling(singleY_mi_model, singleY_mi_data)
```

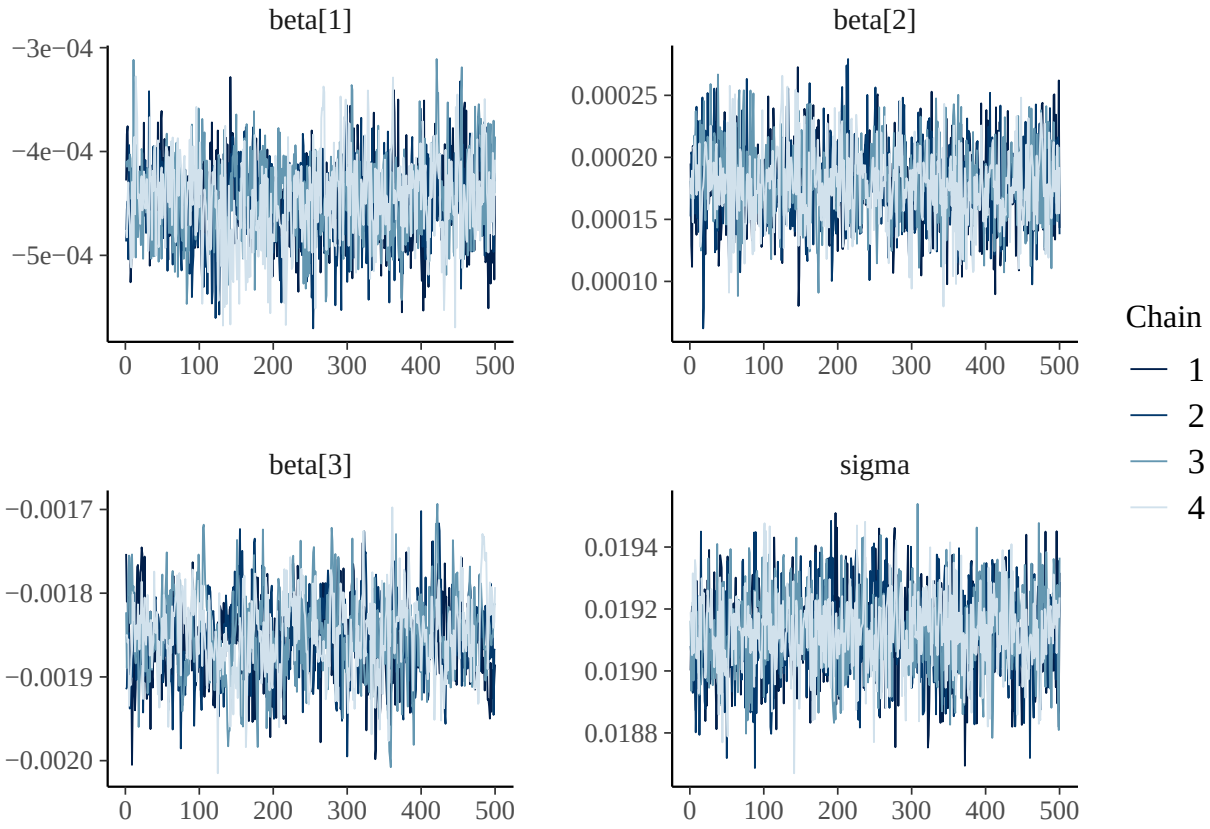
This single fit took a little over 5 hours to complete. Due to time constraints, we will have to do the full model on a subset of the data, although it may prove amenable to MLE optimization.

```
#singleY_mi_fit <- read_stan_csv(dir("~/src/cmdstan-2.18.0/", pattern="samples.*.csv", full.names=TRUE)
#singleY_mi_pars <- as.array(singleY_mi_fit, pars=c("beta", "sigma")) #We loaded this from .RData. The
mcmc_intervals(100*singleY_mi_pars[,,1:3], prob_outer=0.999) # times 100 b/c I forgot to scale sector_r
```



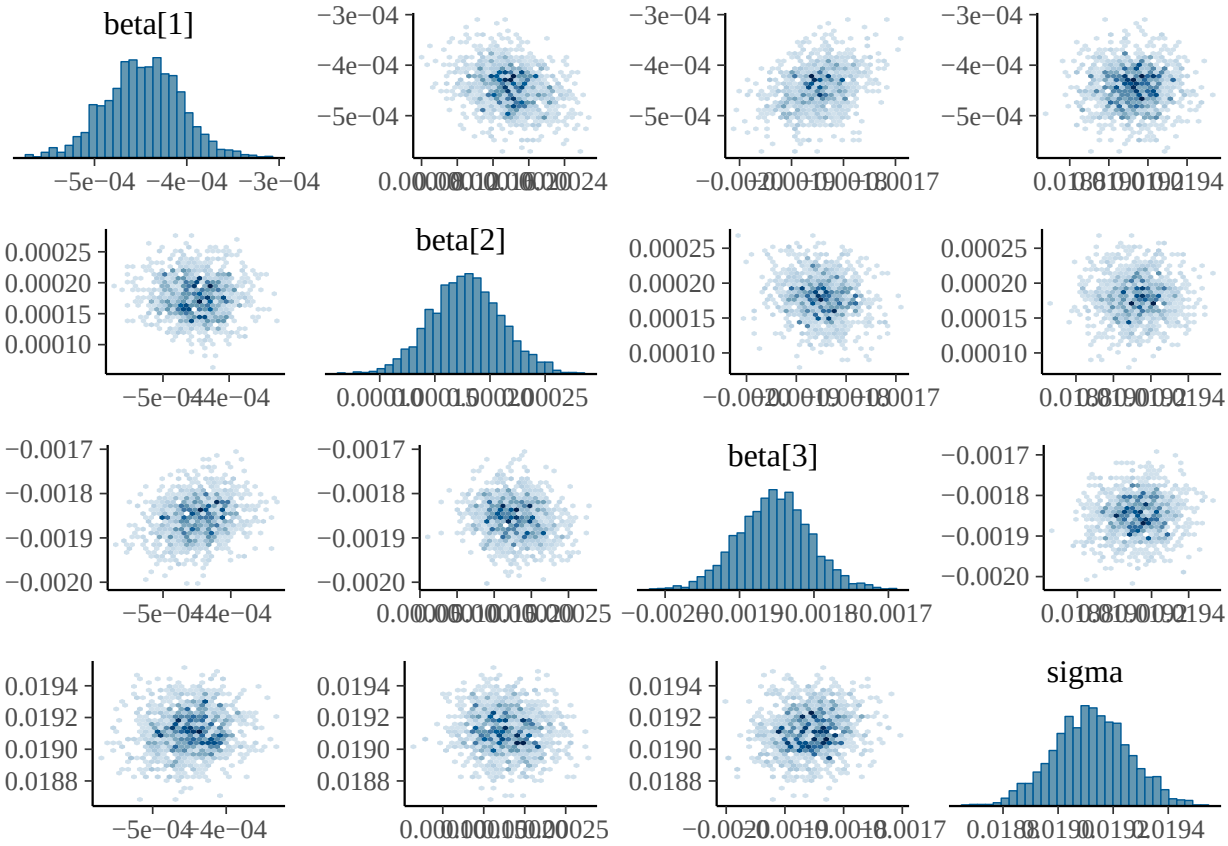
Those are pretty tight, although changes in yield curve factors on software are clearly not a primary driver for returns. Let's look at a quick diagnostic to make sure the markov chains are well-mixed. The worst one is likely to be beta[1] (beta for 'early'), which had 417 effective draws out of a potential 4000.

```
mcmc_trace(singleY_mi_pars)
```



Very well-mixed. We could have gotten away with fewer iterations. Let's take a quick look at implicit collinearity.

```
mcmc_pairs(singleY_mi_pars, off_diag_fun = "hex")
```



The interactions of the betas with sigma are of particular interest, but they look homoskedastic in this case. We can see some relationships between the betas, however. `early` and `mid` look to have a negative correlation, as we saw in the OLS regression.

HMC Model Version 2: Partial pooling

The full partial pooling model will be 2 orders of magnitude more complicated than the previous one, and so cannot be fit via sampling in the same fashion. However, fitting by MLE has been shown to be troublesome for hierarchical models. We will restrict our model to 5 sectors:

- Software: since we used that before
- Gold: since it is so different to the others
- Banks
- Utilities
- Healthcare

```
// adapted from stan manual p182, pp37-40
data {
  int<lower=0> N; // total # of observations
  int<lower=0> K; // total # of factors
  int<lower=0> J; // total # of sectors
  int<lower=0> N_mis[K]; // number of missing points for each factor
  int<lower=1, upper=N> ii_missing_early[N_mis[1]]; //index of missing points for each factor
  int<lower=1, upper=N> ii_obs_early[(N-N_mis[1])];
  int<lower=1, upper=N> ii_missing_mid[N_mis[2]];
  int<lower=1, upper=N> ii_obs_mid[(N-N_mis[2])];
  int<lower=1, upper=N> ii_missing_late[N_mis[3]];
```

```
  int<lower=1, upper=N> ii_obs_late[(N-N_mis[3])];

  // parameters for student_ts of missing factor data
  real param_mu[K];
  real<lower=0> param_nu[K];
  real<lower=0> param_sigma[K];

  vector[N-N_mis[1]] obs_early; // observed early factor
  vector[N-N_mis[2]] obs_mid; // observed mid factor
  vector[N-N_mis[3]] obs_late; // observed late factor

  // endogenous data
  real y[N]; // equity sector returns
  int<lower=1, upper=J> sector[N]; //sector

  //Flag for simulation
  int<lower=0, upper=1> simulate;
}
parameters {
  vector[N_mis[1]] mis_early;
  vector[N_mis[2]] mis_mid;
  vector[N_mis[3]] mis_late;
  vector[K] theta; // global OLS coefficients
  real<lower=0> sigma; // error term for y
  cholesky_factor_corr[K] L_Omega; //correlation of factors
  vector<lower=0>[K] tau; //scale for betas
  matrix[K, J] z; //intermediate group beta

}
transformed parameters {
  matrix[J,K] beta; // local OLS coefficients
  beta = diag_pre_multiply(tau, L_Omega) * z)';
}
model {
  matrix[N,K] x;
  x[ii_missing_early,1] = mis_early;
  x[ii_obs_early,1] = obs_early;
  x[ii_missing_mid,2] = mis_mid;
  x[ii_obs_mid,2] = obs_mid;
  x[ii_missing_late,3] = mis_late;
  x[ii_obs_late,3] = obs_late;
  for(k in 1:K) {
    x[,k] ~ student_t(param_nu[k], param_mu[k], param_sigma[k]);
  }
  to_vector(z) ~ std_normal();
  L_Omega ~ lkj_corr_cholesky(2);
  sigma ~ normal(0,5);
  tau ~ normal(0, 0.5); // Michael betancourt says use half-normal, not half-cauchy

  if(simulate==0)
    y ~ normal(rows_dot_product(beta[sector], x), sigma);
}
```

In the new model, equity sector returns are estimated using local coefficients, which are estimated in turn as

draws from a multivariate normal distribution:

$$\hat{y_{i,j}} \sim N(x_i\beta_j, \sigma)\beta \sim MN(0, \tau'\Omega\tau)\sigma \sim N(0,5); \sigma > 0$$

We will be passing tidy data with an indexer to denote to which sector each row belongs. This necessitates some changes to the missing data parameterization, as we can no longer assume that all the missing data occurs at the beginning of the vector. We make use of an additional set of indexers to denote missing data. This means the data list will be substantially more complicated.

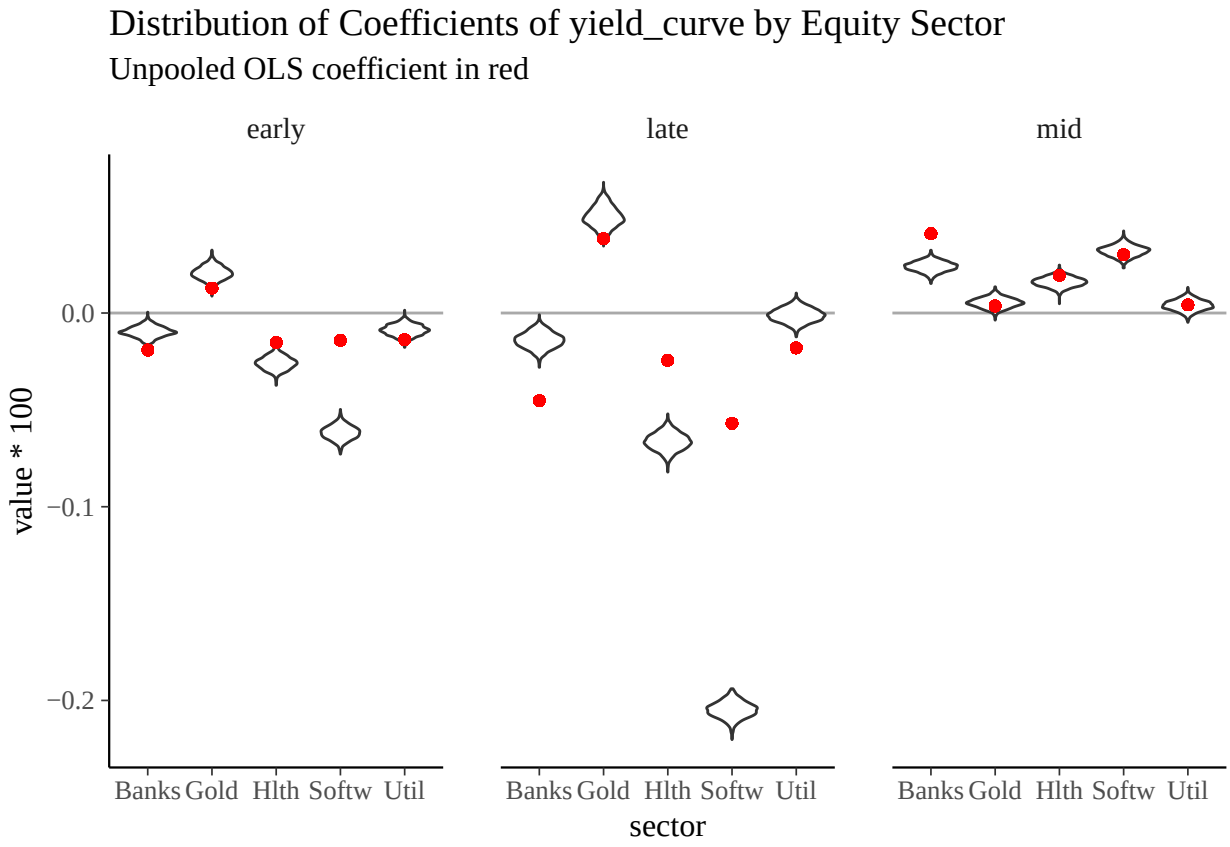
```
sector_returns %>%
  filter(sector %in% c("Banks", "Gold", "Hlth", "Softw", "Util")) %>%
  drop_na() %>%
  left_join(yc_spread_diff %>%
              spread(duration, value) %>%
              select(date, early, mid, late),
            by=c("Date"="date")) %>%
  mutate(sector=factor(sector)) ->
  tmp_data
lme_data <- list(
  N=nrow(tmp_data),
  K=3,
  J=length(unique(tmp_data$sector)),
  N_mis=tmp_data %>% summarize_at(vars(early, mid, late), ~ sum(is.na(.))) %>% as.numeric,
  ii_missing_early=which(is.na(pull(tmp_data, "early"))),
  ii_obs_early=which(!is.na(pull(tmp_data, "early"))),
  ii_missing_mid=which(is.na(pull(tmp_data, "mid"))),
  ii_obs_mid=which(!is.na(pull(tmp_data, "mid"))),
  ii_missing_late=which(is.na(pull(tmp_data, "late"))),
  ii_obs_late=which(!is.na(pull(tmp_data, "late"))),
  param_mu=t_params[grep("mu",names(t_params))],
  param_nu=t_params[grep("nu",names(t_params))],
  param_sigma=t_params[grep("sigma",names(t_params))],
  obs_early=tmp_data %>% drop_na(early) %>% pull("early"),
  obs_mid=tmp_data %>% drop_na(mid) %>% pull("mid"),
  obs_late=tmp_data %>% drop_na(late) %>% pull("late"),
  y=pull(tmp_data,"avg_return"),
  sector=as.numeric(pull(tmp_data, "sector"))
)
with(lme_data, rstan::stan_rdump(names(lme_data), file="lme_yc.data.R")) #data file for command line St
```

RStudio has some memory problems related to calling Stan's C++ code for big models, so we will fit outside of this notebook.

```
dimnames(lme_pars$beta) <- list(iteration=NULL,
                                sector=c("Banks", "Gold", "Hlth", "Softw", "Util"),
                                yield_curve=c("early", "mid", "late"))

as.tibble(lme_pars$beta) %>%
  gather(x, value) %>%
  separate(x, c("sector", "term")) %>%
  left_join(fit_lm %>% select(sector, term, estimate), by=c("sector", "term")) %>%
  ggplot(aes(y=value*100, x=sector)) + #recall we scaled terms for the OLS fit
  facet_wrap(~term) +
  geom_hline(yintercept=0, col="darkgrey") +
  geom_violin() +
```

```
  geom_point(aes(y=estimate), col="red") +
  ggtitle("Distribution of Coefficients of yield_curve by Equity Sector",
          subtitle = "Unpooled OLS coefficient in red")
```

```
## Warning: Column `term` joining character vector and factor, coercing into
## character vector
```



Surprisingly, partial pooling has created stronger betas than the separate models.

MLE optimization

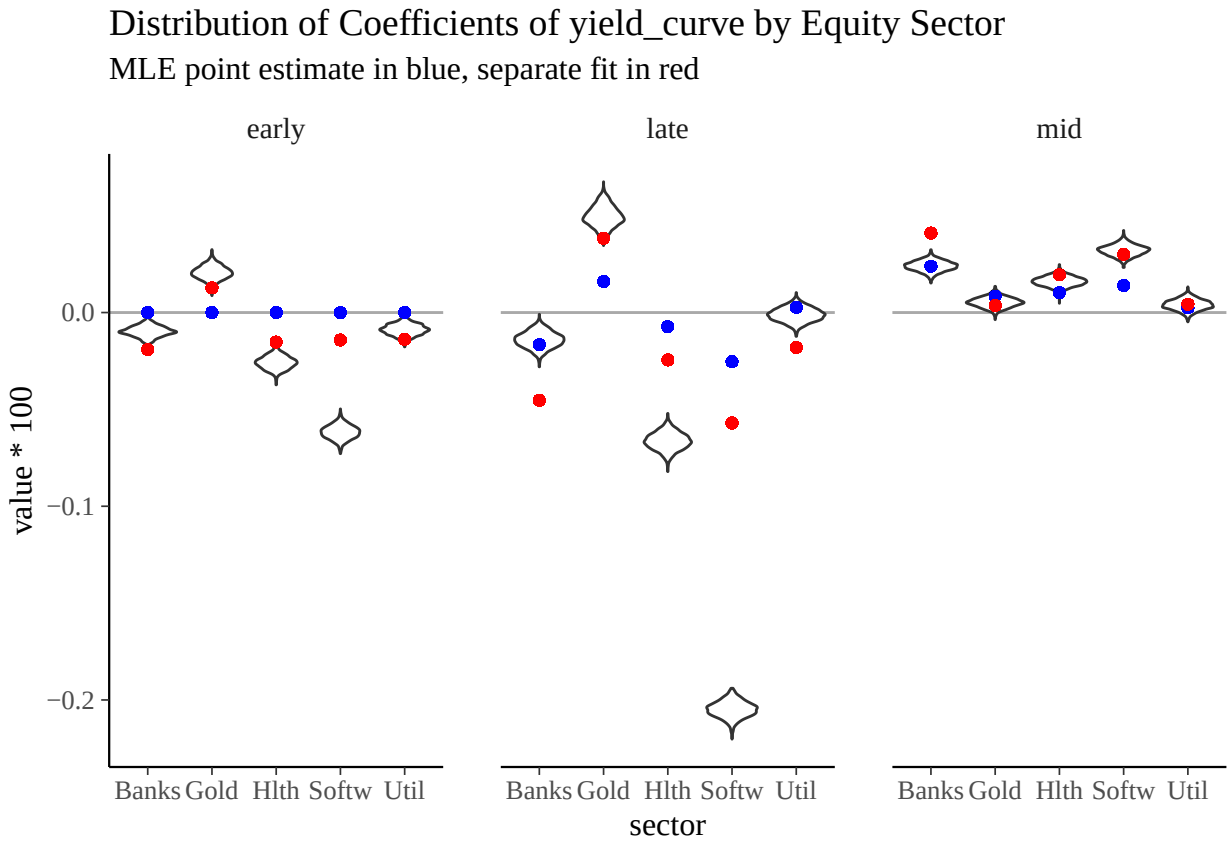
Does a point estimate fit by MLE get in the same ball-park? I'm not sure it will behave well with the simulated missing data.

```
lme_mle_fit <- optimizing(lme_model, data=lme_data)
```

```
mle_beta <- matrix(lme_mle_fit$par[grep("beta", names(lme_mle_fit$par))], 5,3,
                   dimnames=dimnames(lme_pars$beta)[2:3])

as.tibble(lme_pars$beta) %>%
  gather(x, value) %>%
  separate(x, c("sector", "term")) %>%
  left_join(
    as.tibble(as_tibble(mle_beta, rownames="sector")) %>%
      gather(term, mle, -sector), by=c("sector", "term")
  ) %>%
  left_join(fit_lm %>% select(sector, term, estimate), by=c("sector", "term")) %>%
  ggplot(aes(y=value*100, x=sector)) +
```

```
  facet_wrap(~term) +
  geom_hline(yintercept=0, col="darkgrey") +
  geom_violin() +
  geom_point(aes(y=mle*100), col="blue") +
  geom_point(aes(y=estimate), col="red") +
  ggtitle("Distribution of Coefficients of yield_curve by Equity Sector",
          subtitle = "MLE point estimate in blue, separate fit in red")
```

```
## Warning: Column `term` joining character vector and factor, coercing into
## character vector
```



These are not what we might have hoped, and we had convergence. It looks like running MLE on all 49 sectors will not be adequate. We'll have to stick with the 5 sector model in the report.

Implied Beta Correlations

As part of the hierarchical model, we fit a covariance matrix for the yield curve coefficients. How did that look? We actually fit the Cholesky decomposition of the correlation matrix, plus a scaling vector separately, as by that method we can guarantee positive-definite covariance matrices.

```
dimnames(lme_pars$L_Omega) <- list(iterations=NULL, dimnames(lme_pars$beta)[[3]], dimnames(lme_pars$beta
cor_matrices <- apply(lme_pars$L_Omega,1,tcrossprod)
expand.grid(dimnames(lme_pars$beta)[[3]],dimnames(lme_pars$beta)[[3]], stringsAsFactors = F) %>%
  transmute(nm=paste(Var1, Var2)) %>%
  pull() ->
  rownames(cor_matrices)
```

```r
cor_matrices %>%
  t() %>%
  as.data.frame() %>%
  gather(combo, rho) %>%
  filter(rho<0.99) %>%
  separate(combo, c("row", "col")) %>%
  ggplot(aes(x=rho)) +
  geom_density() +
  facet_grid(row~col) +
  ggtitle("Estimated Coefficient Correlations")
```