Investment Analytics with R and SQL via the PL/R Extension

Dirk Hugen

June 1, 2018

Tail Estimation

Preparing to install funda on our local Postgres database, we locate the directory and read in the funda csv file, convert datadate to R date format, and examine.

```
> library(ggplot2)
> setwd("/Users/owner/Desktop/FA/plr3")
> funda<-read.csv("funda.csv")
> funda$datadate<-as.Date(funda$datadate,"%Y-%m-%d")
```

We now connect to the default postgres database and write the R dataframes to PostgreSQL tables using the `dbWriteTable()` function.

```
> library(RPostgreSQL)
> con<-dbConnect(PostgreSQL(),dbname="postgres",host="localhost")
> dbWriteTable(con, "funda", funda, overwrite=TRUE, row.names=FALSE)
```

```
[1] TRUE
```

Tail Estimation

```
> query<-"
+ WITH funda2 AS (SELECT *, prcc_f * csho AS mcap FROM funda)
+ SELECT fyear, tic, mcap/sale AS psr
+ FROM funda2
+ WHERE fyear = 2010"
> res<-dbGetQuery(con,query)
> head(res)
```

	fyear	tic	psr
1	2010	CMA	NA
2	2010	AAL	0.1171662
3	2010	PNW	1.3814359
4	2010	ABT	2.1075608
5	2010	AET	0.3424646
6	2010	AET	NA

Tail Estimation

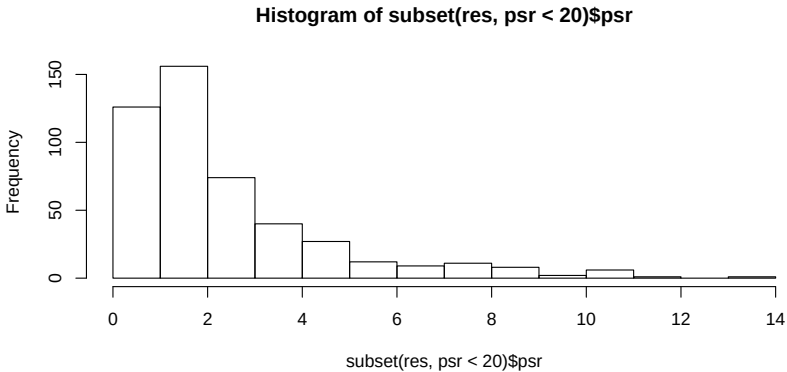
Long tails are evident. Closer look...

```
> hist(res$psr)
```

Tail Estimation

Examine shape of distribution. Even closer look...

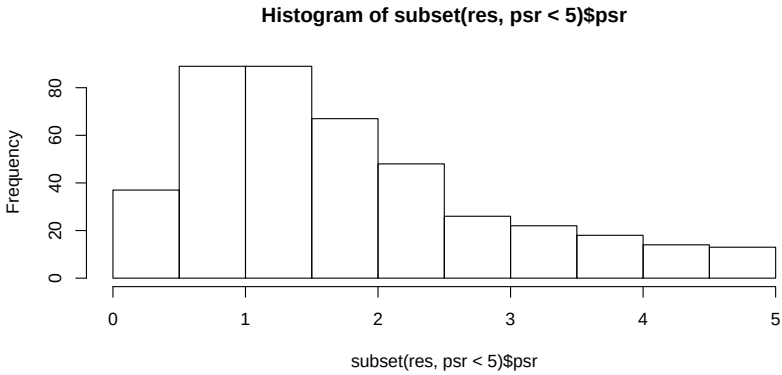
```
> hist(subset(res,psr<20)$psr)
```



Tail Estimation

Observe mode of distribution greater than minimum, so conclude Weibull.

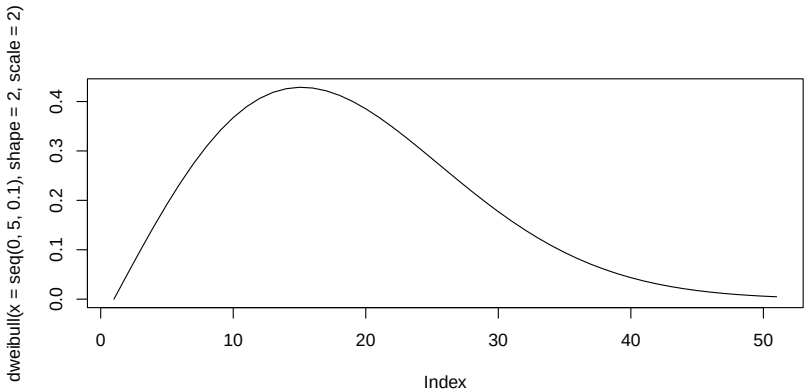
```
> hist(subset(res,psr<5)$psr)
```



Tail Estimation

Observe mode of distribution greater than minimum, so conclude Weibull.

```
> plot(dweibull(x=seq(0,5,.1),shape=2,scale=2),type='line')
```



Tail Estimation

We begin with a simple example.

```
> library(MASS)
> fitdistr(na.omit(res$psr),densfun=dweibull,start=list(scale=2,shape=5))

     scale         shape
  2.59377259    1.10988930
 (0.11377169) (0.03477376)

> fitdistr(na.omit(res$psr),densfun=dweibull,start=list(scale=2,shape=5))$estimate[1]

   scale
2.593773

> fitdistr(na.omit(res$psr),densfun=dweibull,start=list(scale=2,shape=5))$estimate[2]

   shape
1.109889
```

Tail Estimation

```
> query<-"
+ CREATE OR REPLACE FUNCTION r_fitweibull(float8[]) RETURNS float8 AS'
+ shape<-NA
+ library(MASS)
+ try(shape<-fitdistr(arg1,densfun=dweibull,start=list(scale=1,shape=2))$estimate[2])
+ return(shape)
+ ' LANGUAGE 'plr'"
> res<-dbSendQuery(con,query)
```

Tail Estimation

```
> query<-"
+ DROP AGGREGATE IF EXISTS fitweibull(double precision);
+ CREATE AGGREGATE fitweibull(
+ sfunc = plr_array_accum,
+ basetype = float8,
+ stype = float8[],
+ finalfunc = r_fitweibull);"
> res<-dbSendQuery(con,query)
```

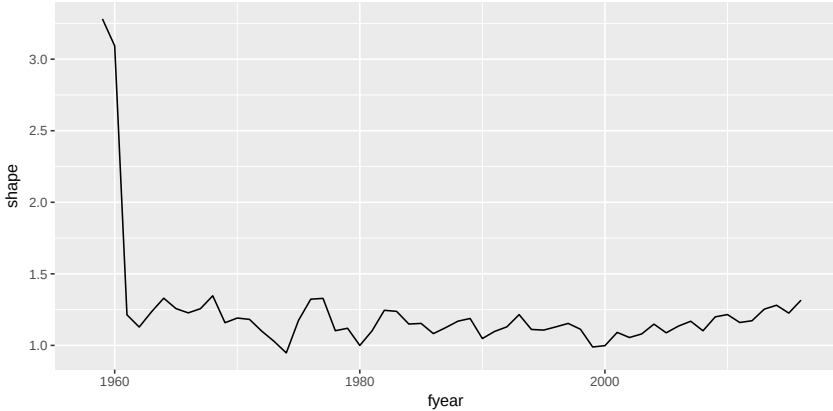
Tail Estimation

```
> query<-"
+ WITH funda2 AS (SELECT *, prcc_f*csho/sale AS psr FROM funda)
+ SELECT fyear, fitweibull(psr) AS shape
+ FROM funda2
+ WHERE psr < 20
+ GROUP BY fyear
+ ORDER BY fyear"
> res<-dbGetQuery(con,query)
> head(res)
```

	fyear	shape
1	1958	NA
2	1959	3.281179
3	1960	3.092712
4	1961	1.213482
5	1962	1.128314
6	1963	1.234134

Analysis of Quarterly Sales Predictions

```
> ggplot(res, aes(x=fyear)) + geom_line(aes(y=shape))
```



Tail Estimation

Once we have shape a we can calculate the method of moments estimate of scale b as:

$$b = \frac{E(X)}{\Gamma(1 + 1/a)}$$

However, for this we need the Gamma function. SQL doesn't have it, but R does. We define it using PL/R as follows:

```
> query<-"
+ CREATE OR REPLACE FUNCTION rgamma(float8) RETURNS float8 AS'
+ val<-NA
+ try(val<-gamma(arg1))
+ return(val)
+ ' LANGUAGE 'plr'"
> res<-dbSendQuery(con,query)
```

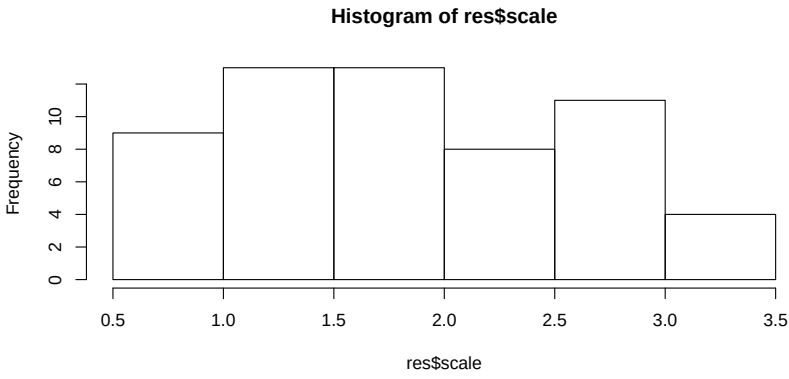
Tail Estimation

We can now use the Gamma function in a SQL query:

```
> query<-"
+ WITH funda2 AS (
+ SELECT *, prcc_f * csho/sale AS psr
+ FROM funda
+ ),
+ funda3 AS (
+ SELECT fyear, fitweibull(psr) AS shape, AVG(psr) AS mean
+ FROM funda2
+ WHERE psr < 20
+ GROUP BY fyear
+ ORDER BY fyear
+ )
+ SELECT fyear, shape, mean / RGAMMA(1 + 1 / shape) AS scale
+ FROM funda3
+ ORDER BY fyear"
> res<-dbGetQuery(con,query)
```

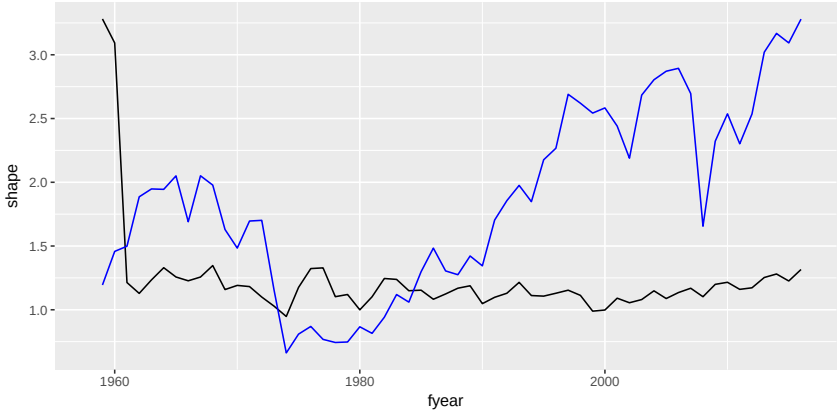
Tail Estimation

```
> hist(res$scale)
```



Analysis of Quarterly Sales Predictions

```
> ggplot(res, aes(x=fyear)) + geom_line(aes(y=shape)) + geom_line(aes(y=scale), colour='blue')
```



Tail Estimation

We might want to test if scale and shape parameters vary across industries. We divide the four digit `sic` to get a rough classification of industry segment.

```
> query<-"
+ WITH funda2 AS (
+ SELECT sic/1000::INTEGER AS sic, tic, fyear, prcc_f * csho/sale AS psr
+ FROM funda
+ ),
+ funda3 AS (
+ SELECT sic, tic, fitweibull(psr) AS shape, AVG(psr) AS mean
+ FROM funda2
+ WHERE psr < 20
+ GROUP BY sic, tic
+ ORDER BY sic, tic
+ )
+ SELECT sic, tic, shape, mean / RGAMMA(1 + 1 / shape) AS scale
+ FROM funda3
+ ORDER BY sic, tic"
> res<-dbGetQuery(con,query)
```

Tail Estimation

Carrying out an analysis of variance we see moderately significant dependence of shape on industry.

```
> summary(aov(shape~sic,data=res))

             Df Sum Sq Mean Sq F value Pr(>F)
sic           1     42   41.81   4.049 0.0447 *
Residuals   498   5142   10.33
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
3 observations deleted due to missingness
```

Carrying out an analysis of variance we see highly significant dependence of scale on industry.

```
> summary(aov(scale~sic,data=res))

             Df Sum Sq Mean Sq F value   Pr(>F)
sic           1   81.1   81.10   13.62 0.000248 ***
Residuals   498 2964.3    5.95
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
3 observations deleted due to missingness
```

Tail Estimation

```
> dbSendQuery(con,"DROP FUNCTION rgamma(float8)")
<PostgreSQLResult>
> dbSendQuery(con,"DROP AGGREGATE fitweibull(float8)")
<PostgreSQLResult>
```

Analysis of Quarterly Sales Predictions

Recall definition of the quarterly time series prediction window function we used.

```
> query<-"
+ CREATE OR REPLACE FUNCTION r_predict(float8)
+ RETURNS float8 AS
+ $BODY$
+ pred <- NA
+ x1 <- farg1
+ if (fnumrows==24) try (
+   mod<-arima(x = log(x1),order = c(0,1,1),
+   seasonal = list(order=c(0,1,1), period=4)))
+ if (fnumrows==24) try (
+   pred <- predict(mod, n.ahead = 1)$pred)
+ return(exp(pred))
+ $BODY$
+ LANGUAGE plr WINDOW"
> #res<-dbSendQuery(con,query)
```

Analysis of Quarterly Sales Predictions

We can carry out this prediction not only on a single company, but on the entire S&P 500.

```
> query<-"
+ WITH fundq2 AS(
+   SELECT *, r_predict(saleq) OVER w AS predq
+   FROM fundq
+   WINDOW w AS (PARTITION BY gvkey ORDER BY datadate
+     ROWS BETWEEN 24 PRECEDING AND 1 PRECEDING)
+ )
+ SELECT gvkey, tic, datadate, sic, saleq, predq
+ FROM fundq2
+ WHERE datadate >= '2000-01-01'
+ ORDER BY tic, datadate"
> #res<-dbGetQuery(con, query)
```

Analysis of Quarterly Sales Predictions

The query takes a while, so we load stored results, convert date format, examine, and then write to SQL table.

```
> setwd("/Users/owner/Desktop/FA/plr3")
> res<-read.csv("res.csv")
> res$datadate<-as.Date(res$datadate,"%Y-%m-%d")
> head(res)
```

```
   gvkey tic   datadate  sic saleq predq
1 126554   A 2000-01-31 3826  1851    NA
2 126554   A 2000-04-30 3826  2142    NA
3 126554   A 2000-07-31 3826  2351    NA
4 126554   A 2000-10-31 3826  3017    NA
5 126554   A 2001-01-31 3826  2565    NA
6 126554   A 2001-04-30 3826  2406    NA
```

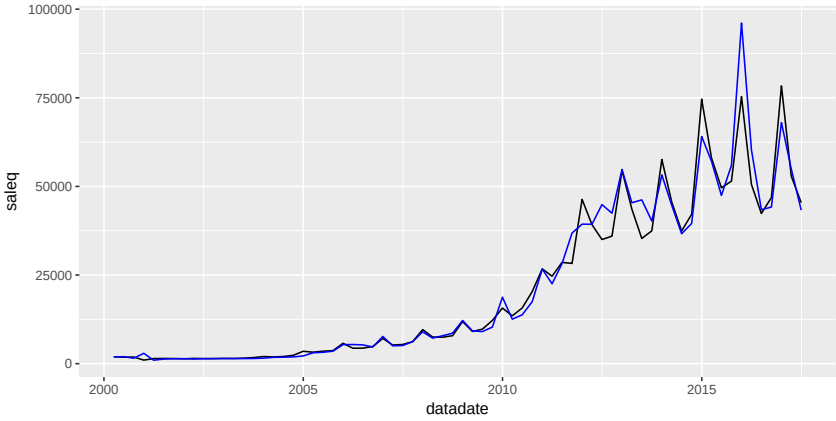
```
> dbWriteTable(con,"fundq2",res,overwrite=TRUE,row.names=FALSE)
```

```
[1] TRUE
```

Analysis of Quarterly Sales Predictions

Examine quarterly sales predictions for Apple. Exponential trend and quarterly cyclicality is evident.

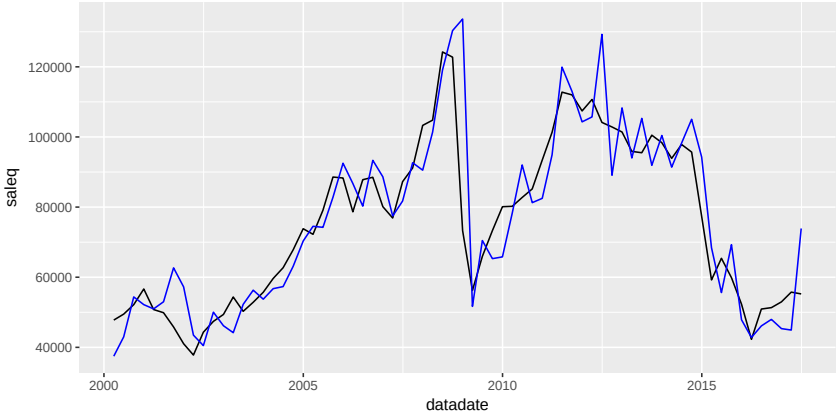
```
> temp<-subset(res, tic == 'AAPL')
> ggplot(temp, aes(x=datadate)) + geom_line(aes(y=saleq)) + geom_line(aes(y=predq),colour='blue')
```



Analysis of Quarterly Sales Predictions

Examine quarterly sales predictions for Exxon-Mobil. Challenge is structural break caused by financial crisis.

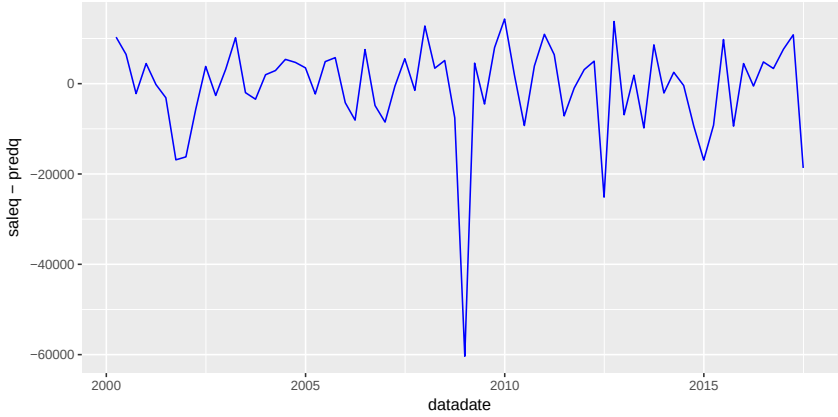
```
> temp<-subset(res, tic == 'XOM')
> ggplot(temp, aes(x=datadate)) + geom_line(aes(y=saleq)) + geom_line(aes(y=predq),colour='blue')
```



Analysis of Quarterly Sales Predictions

Plot prediction errors. Outlier corresponding to structural break is evident.

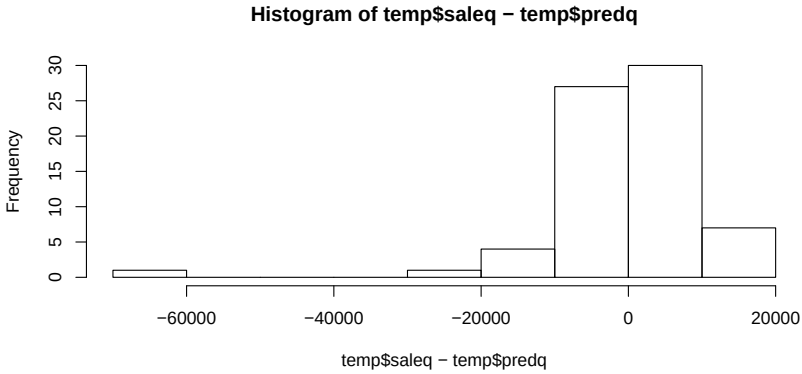
```
> temp<-subset(res, tic == 'XOM')
> ggplot(temp, aes(x=datadate)) + geom_line(aes(y=saleq-predq),colour='blue')
```



Analysis of Quarterly Sales Predictions

Examine the histogram of prediction error.

```
> temp<-subset(res, tic == 'XOM')
> hist(temp$saleq-temp$predq)
```



Analysis of Quarterly Sales Predictions

Test prediction errors for normality.

```
> shapiro.test(temp$saleq-temp$predq)

        Shapiro-Wilk normality test

data:  temp$saleq - temp$predq
W = 0.80082, p-value = 2.514e-08
```

Question is how to carry out a sensible analysis of prediction quality for entire S&P 500. Eye-balling and hand-waving is no longer sufficient. We need the abilty to systematically apply analytic tools. Begin by defining `shapiro.test()` for use in SQL query.

```
> query<-"
+ CREATE OR REPLACE FUNCTION r_shapiro(float8[]) RETURNS float8 AS'
+ pval<-NA
+ try(pval<-shapiro.test(arg1)$p.value)
+ return(pval)
+ 'LANGUAGE'plr'"
> dbSendQuery(con,query)

<PostgreSQLResult>
```

Analysis of Quarterly Sales Predictions

Now define SQL function `r_shapiro()` as SQL aggregate `shapiro()`.

```
> query<-"
+ DROP AGGREGATE IF EXISTS shapiro(double precision);
+ CREATE AGGREGATE shapiro(
+ sfunc = plr_array_accum,
+ basetype = float8,
+ stype = float8[],
+ finalfunc = r_shapiro);"
> dbSendQuery(con,query)
```

```
<PostgreSQLResult>
```

Analysis of Quarterly Sales Predictions

Apply SQL aggregate `shapiro()` to each company in table.

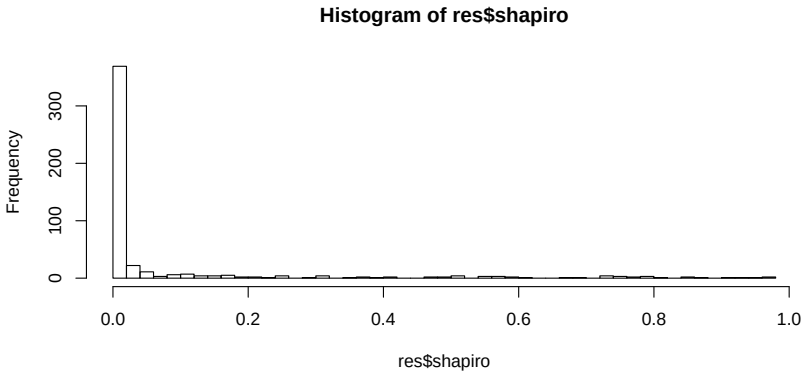
```
> query<-"
+ WITH fundq3 AS (SELECT *, saleq - predq AS errorq FROM fundq2)
+ SELECT tic, shapiro(errorq) AS shapiro
+ FROM fundq3
+ GROUP BY tic
+ ORDER BY tic"
> res<-dbGetQuery(con,query)
> head(res)
```

```
   tic      shapiro
1    A 1.373692e-04
2   AA           NA
3  AAL 1.129364e-06
4  AAP 6.280113e-13
5 AAPL 6.882298e-10
6 ABBV 8.504754e-01
```

Analysis of Quarterly Sales Predictions

Examine histogram of Shapiro test results. Observe that most companies fail test of normality.

```
> hist(res$shapiro,breaks=50)
```



Analysis of Quarterly Sales Predictions

Normality is broken by skew or kurtosis. We might ask which is more prevelant here. First step is to bring `skewness()` function of `moments` package into SQL.

```
> query<-"
+ CREATE OR REPLACE FUNCTION r_skewness(double precision[]) RETURNS float8 AS'
+ library(moments)
+ skew<-NA
+ try(skew<-skewness(arg1))
+ return(skew)
+ 'LANGUAGE'plr';"
> dbSendQuery(con,query)
```

```
<PostgreSQLResult>
```

Analysis of Quarterly Sales Predictions

```
> query<-"
+ DROP AGGREGATE IF EXISTS skewness(double precision);
+ CREATE AGGREGATE skewness(
+ sfunc = plr_array_accum,
+ basetype = float8,
+ stype = float8[],
+ finalfunc = r_skewness);"
> dbSendQuery(con,query)
```

```
<PostgreSQLResult>
```

Analysis of Quarterly Sales Predictions

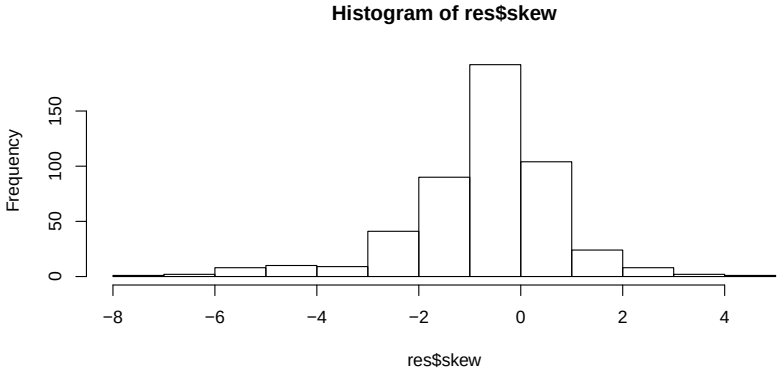
```
> query<-"
+ WITH fundq3 AS (SELECT *, saleq - predq AS errorq FROM fundq2)
+ SELECT tic, skewness(errorq) AS skew
+ FROM fundq3
+ GROUP BY tic
+ ORDER BY tic"
> res<-dbGetQuery(con,query)
> head(res)
```

```
   tic       skew
1    A -1.0239756
2   AA         NA
3  AAL -0.6354026
4  AAP -4.4060850
5 AAPL -1.7339168
6 ABBV -0.4335979
```

Analysis of Quarterly Sales Predictions

Examine histogram of prediction skew for all companies. Note that if errors were all perfectly normal, we would see only one bar in the histogram, at zero. Instead, we see a distribution.

```
> hist(res$skew)
```



Analysis of Quarterly Sales Predictions

Carry out the same for kurtosis.

```
> query<-"
+ CREATE OR REPLACE FUNCTION r_kurtosis(double precision[]) RETURNS float8 AS'
+ library(moments)
+ kurt<-NA
+ try(kurt<-kurtosis(arg1))
+ return(kurt)
+ 'LANGUAGE'plr';"
> dbSendQuery(con,query)
```

```
<PostgreSQLResult>
```

Analysis of Quarterly Sales Predictions

```
> query<-"
+ DROP AGGREGATE IF EXISTS kurtosis(double precision);
+ CREATE AGGREGATE kurtosis(
+ sfunc = plr_array_accum,
+ basetype = float8,
+ stype = float8[],
+ finalfunc = r_kurtosis);"
> dbSendQuery(con,query)
```

```
<PostgreSQLResult>
```

Analysis of Quarterly Sales Predictions

```
> query<-"
+ WITH fundq3 AS (SELECT *, saleq - predq AS errorq FROM fundq2)
+ SELECT tic, kurtosis(errorq) AS kurt
+ FROM fundq3
+ GROUP BY tic
+ ORDER BY tic"
> res<-dbGetQuery(con,query)
> head(res,10)
```

	tic	kurt
1	A	7.571920
2	AA	NA
3	AAL	8.938145
4	AAP	31.323258
5	AAPL	10.926605
6	ABBV	1.991714
7	ABC	9.101168
8	ABT	19.567190
9	ACN	6.270113
10	ADBE	5.849562

Analysis of Quarterly Sales Predictions

```
> hist(res$kurt)
```

Analysis of Quarterly Sales Predictions

So we obviously see variation in skew and kurtosis across companies. We might wonder if this is due in part to variation across industries. We classify using the `sic` code.

```
> query<-"
+ WITH fundq3 AS (SELECT *, saleq - predq AS errorq FROM fundq2)
+ SELECT sic/1000::INTEGER AS sic, tic, skewness(errorq) AS skew, kurtosis(errorq) AS kurt
+ FROM fundq3
+ GROUP BY sic, tic
+ ORDER BY sic, tic"
> res<-dbGetQuery(con,query)
> head(res)
```

	sic	tic	skew	kurt
1	0	MON	0.08628396	3.65665
2	1	APA	-1.72298339	9.41701
3	1	APC	-2.84558009	17.81858
4	1	BHI	-2.97151025	18.57178
5	1	CHK	-0.54875362	10.05669
6	1	COG	-0.15931485	10.40506

Analysis of Quarterly Sales Predictions

We carry out an analysis of variance and find that there is moderate significance of industry in skew, but no significance for kurtosis.

```
> summary(aov(skew~sic,data=res))

             Df Sum Sq Mean Sq F value Pr(>F)
sic           1   13.8  13.780   6.453 0.0114 *
Residuals   490 1046.4   2.136
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
7 observations deleted due to missingness

> summary(aov(kurt~sic,data=res))

             Df Sum Sq Mean Sq F value Pr(>F)
sic           1      0    0.26   0.004  0.951
Residuals   490  33190   67.73
7 observations deleted due to missingness
```

Gereralized Auto-Regressive Conditional Heteroskedasticity

As we have discussed previously, many financial time series have a variance that changes with time. Generalized Auto-Regressive Conditional Heteroskedasticity (GARCH) models give us a framework to model this phenomenon. While ARMA(p,q) processes have both auto-regressive AR(p) and moving average MA(q) components, GARCH processes have a similar structure.

$$\sigma^2_{t|t-1} = \omega + \beta_1 \sigma^2_{t-1|t-2} + \cdots + \beta_p \sigma^2_{t-p|t-p+1} + \alpha_1 r^2_{t-1} + \cdots + \alpha_q r^2_{t-q}$$

In the GARCH framework the series we are trying to model: σ^2_t is analogous to y_t in the ARMA framework, and squared returns r^2_t in GARCH are analogous to the innovations e_t in ARMA. The β coefficients determine how past values of volatility σ^2_t affect the present value of σ^2_t, and the α coefficients determine how past values of squared returns r^2_t affect the present value of σ^2_t.

Gereralized Auto-Regressive Conditional Heteroskedasticity

```
> library(TSA)
> data(google)
> plot(google,col='blue')
```

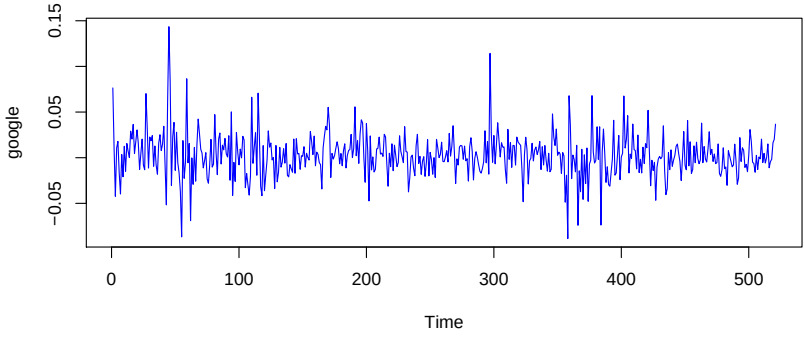
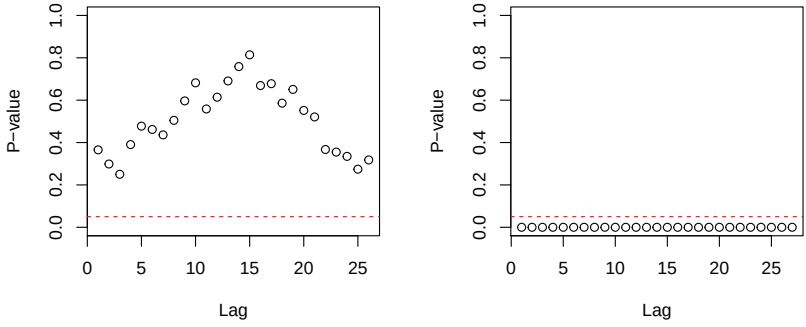


Figure: Daily returns of Google stock from 08/20/04 - 09/13/06

Gereralized Auto-Regressive Conditional Heteroskedasticity

The next step is to test for Auto-Regressive Conditional Heteroskedasticity (ARCH). The McLeod-Li test carries out this hypothesis test with a null hypothesis of constant variance, or homoskedasticity.

```
> par(mfrow=c(1,2))
> McLeod.Li.test(y=rnorm(500))
> McLeod.Li.test(y=google)
```



Gereralized Auto-Regressive Conditional Heteroskedasticity

Observe lag 1 significance, model a GARCH(1,1) to the series.

```
> par(mfrow=c(1,2))
> acf(google^2)
> pacf(google^2)
```

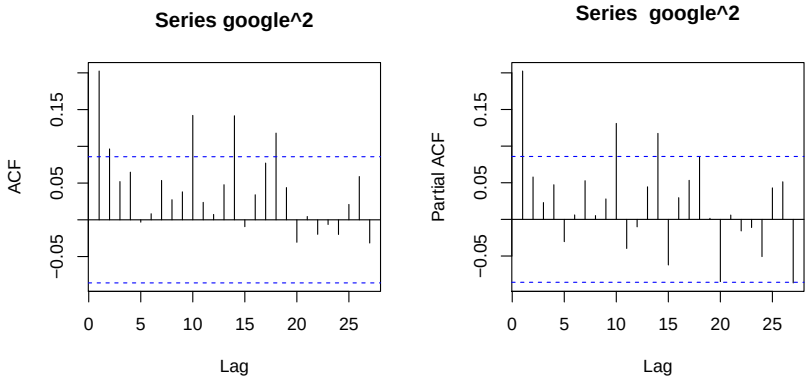


Figure: Autocorrelation functions of squared daily returns of Google stock from 08/20/04 - 09/13/06

Gereralized Auto-Regressive Conditional Heteroskedasticity

We fit the model to the data and examine the output. We see that the estimates of ω, α_1, and β_1 are all highly significant.

```
> m1 <- garch(x=google-mean(google),order=c(1,1),reltol=1e-6)
> summary(m1)
Call:
garch(x = google - mean(google), order = c(1, 1), reltol = 1e-06)

Model:
GARCH(1,1)

Residuals:
     Min       1Q   Median       3Q      Max
-3.60772 -0.59914 -0.04721  0.54559  5.56378

Coefficient(s):
    Estimate  Std. Error  t value Pr(>|t|)
a0 5.246e-05   1.276e-05    4.111 3.94e-05 ***
a1 1.397e-01   2.335e-02    5.984 2.17e-09 ***
b1 7.698e-01   3.722e-02   20.682  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Diagnostic Tests:
        Jarque Bera Test
```

Gereralized Auto-Regressive Conditional Heteroskedasticity

```
> par(mfrow=c(1,2))
> hist(residuals(m1))
> McLeod.Li.test(y=residuals(m1))
```

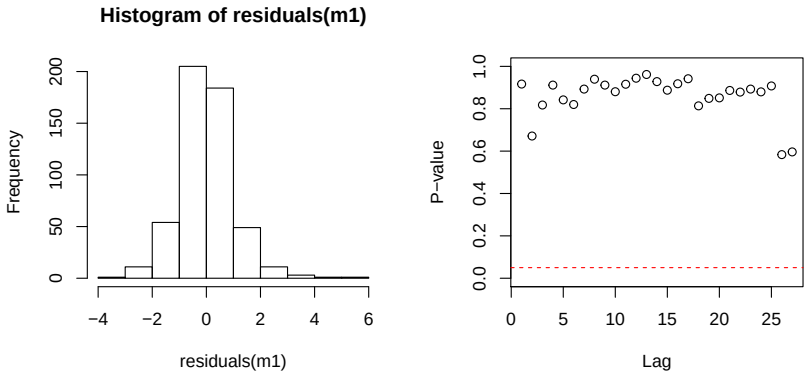


Figure: Histogram and McLeod.Li test of residuals

Gereralized Auto-Regressive Conditional Heteroskedasticity

We now turn to discussion of model diagnostics. We want to guage how well the model fits the data. First a Shapiro test to test for normality.

```
> shapiro.test(residuals(m1))

        Shapiro-Wilk normality test

data:  residuals(m1)
W = 0.96922, p-value = 5.534e-09
```

Assuming stationarity and noting that $\sigma_t^2 = E(r_t^2) - E^2(r_t) \approx E(r_t^2)$ since $E(r_t)$ is close to zero, we can take expectations of both sides of the equation $\sigma_{t|t-1}^2 = \omega + \beta_1 \sigma_{t-1|t-2}^2 + \alpha_1 r_{t-1}^2$ above, yielding:

$$\sigma^2 = \omega + \beta_1 \sigma^2 + \alpha_1 \sigma^2$$

which, when we solve for σ^2 gives us

$$\sigma^2 = \frac{\omega}{1 - \alpha_1 - \beta_1} = \frac{0.00005246}{1 - 0.1397 - 0.7698} = 0.00057967 \tag{1}$$

which is very close to the sample variance of the return process:

```
> var(google)

[1] 0.0005693958
```

Gereralized Auto-Regressive Conditional Heteroskedasticity

```
> query<-"
+ CREATE OR REPLACE FUNCTION r_alpha1(float8[]) RETURNS float8 AS'
+ alpha1<-NA
+ library(tseries)
+ try(alpha1<-garch(x=arg1-mean(arg1),order=c(1,1),reltol=1e-6)$coef[2])
+ return(alpha1)
+ ' LANGUAGE 'plr'"
> dbSendQuery(con,query)
```

```
<PostgreSQLResult>
```

Gereralized Auto-Regressive Conditional Heteroskedasticity

```
> query<-"
+ DROP AGGREGATE IF EXISTS alpha1(double precision);
+ CREATE AGGREGATE alpha1(
+ sfunc = plr_array_accum,
+ basetype = float8,
+ stype = float8[],
+ finalfunc = r_alpha1);"
> dbSendQuery(con,query)
```

```
<PostgreSQLResult>
```

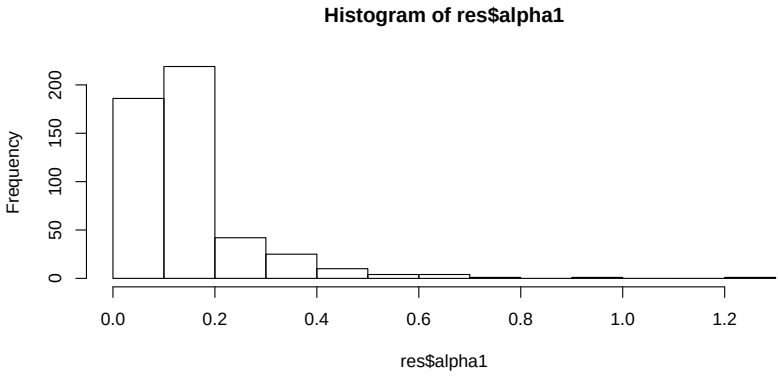
Gereralized Auto-Regressive Conditional Heteroskedasticity

```
> query<-"
+ SELECT ticker, alpha1(ret) AS alpha1
+ FROM msf
+ GROUP BY ticker
+ ORDER BY ticker"
> res<-dbGetQuery(con,query)
> head(res)
```

	ticker	alpha1
1	A	0.09938781
2	AA	0.07375769
3	AAL	0.13281225
4	AAP	0.10405016
5	AAPL	0.11279424
6	ABBV	0.19214361

Gereralized Auto-Regressive Conditional Heteroskedasticity

```
> hist(res$alpha1)
```



Skew

```
> query<-"
+ CREATE OR REPLACE FUNCTION r_beta1(float8[]) RETURNS float AS'
+ beta1<-NA
+ library(tseries)
+ try(beta1<-garch(x=arg1-mean(arg1),order=c(1,1),reltol=1e-6)$coef[3])
+ return(beta1)
+ ' LANGUAGE 'plr'"
> dbSendQuery(con,query)
```

```
<PostgreSQLResult>
```

Gereralized Auto-Regressive Conditional Heteroskedasticity

```
> query<-"
+ DROP AGGREGATE IF EXISTS beta1(double precision);
+ CREATE AGGREGATE beta1(
+ sfunc = plr_array_accum,
+ basetype = float8,
+ stype = float8[],
+ finalfunc = r_beta1);"
> dbSendQuery(con,query)
```

```
<PostgreSQLResult>
```

Gereralized Auto-Regressive Conditional Heteroskedasticity

```
> query<-"
+ SELECT ticker, beta1(ret) AS beta1
+ FROM msf
+ GROUP BY ticker
+ ORDER BY ticker"
> res<-dbGetQuery(con,query)
> head(res)
```

	ticker	beta1
1	A	9.006919e-01
2	AA	8.784991e-01
3	AAL	6.601286e-01
4	AAP	9.021751e-01
5	AAPL	8.526298e-01
6	ABBV	3.300991e-14

Gereralized Auto-Regressive Conditional Heteroskedasticity

```
> hist(res$beta1)
```

The End

Questions?