Manager Incentives

Setup and Create Functions

Setup

```
knitr::opts_chunk$set(echo = TRUE, message = FALSE, warning = FALSE)

suppressPackageStartupMessages({

library(tidyverse)
library(reshape2)
library(lubridate)
library(xts)
library(scales)
library(ReporteRs)
library(RollingWindow)
library(parallel)
library(pbapply)
library(dtplyr)
library(matrixStats)
library(data.table)
library(profvis)
library(ggplot2)
library(scales)
library(knitr)

})

#Read in data
returns <- read_csv("//sigisdev/Risk Share/Share/RV/Projects/overdiversification/returns.csv")

#Melt data and select only managers benchmarked to S&P500 and then repivot
returns_melt = melt(returns,id.vars = c("Firm Name","Product Name","Vehicle Name","Benchmark","VT","RM"


returns_pivot = returns_melt %>% dcast(variable ~ `Firm Name` + `Product Name` + `Vehicle Name`, value.

#Change no return months to NA for easier maniupulation
returns_pivot[returns_pivot =="---"] = NA

#Turn date column into date object
returns_pivot$variable = mdy( returns_pivot$variable)


#Only pick managers with full history between 2006/6/30 and 2016/06/30
returns_subset = returns_pivot %>% filter(variable > ymd("2006/06/30"))

#returns_subset = returns_pivot
#returns_subset = returns_subset[colSums(is.na(returns_subset))<nrow(returns_subset)]
returns_subset = returns_subset[colSums(is.na(returns_subset))<1]
```

```
returns_subset[,-1] = lapply(returns_subset[,-1],function(x) as.numeric(x)/100)

returns_subset_no_na = returns_subset
returns_subset_no_na[is.na(returns_subset)] = 0
rolling_alpha = RollingSum(returns_subset_no_na[,-1],window = 12,na_method = "ignore" ) %>% as.data.fra

returns_subset_no_dates = returns_subset[,-1] %>% as.data.frame()
rolling_alpha_df = rolling_alpha %>% as.data.frame()
dates = returns_subset$variable %>% as.data.frame()

managers = 1:ncol(rolling_alpha)

x = 11
returns = returns_subset_no_dates[,x]
alpha = rolling_alpha_df[,x]
allReturns = returns_subset_no_na
```

Function with buffer

Mean TE

```
get10Date_months = function(returns,alpha,dates,m,buffer,EOY = FALSE){



        #If buffer is true
        if(buffer) {b = 12} else {b = 0}


        returns = returns %>% as.data.frame()
        alpha = alpha %>% as.data.frame()
        dates = dates %>% as.data.frame()

        #mgr = colnames(alpha)

        colnames(alpha) = "returns"
        colnames(returns) = "returns"
        colnames(dates) = "dates"


        data_alpha = cbind(alpha,dates) %>% as.data.frame() %>%
                arrange(dates) %>%
                mutate(dates = as_date(dates))

        dates_alpha = data_alpha %>%
                dplyr::filter(returns>0.1 )#& month(dates)==12)


        if(EOY){
              dates_alpha = dates_alpha %>% dplyr::filter(month(dates)==12)
        }
```

```r
        if (nrow(dates_alpha) ==0) {

                te_dif = data.frame(0,as_date(0),0,0)
                colnames(te_dif) = c("returns","dates","te_before","te_after")


        } else{

                data_returns = cbind(returns,dates) %>% as.data.frame() %>%
                        arrange(dates) %>%
                        mutate(dates = as_date(dates))

                te_before = sapply(dates_alpha$dates,function(x) {
                        rets_sd =  data_returns %>% dplyr::filter(dates<= (x %m-% months(b)) & (x %m-%
                        stdev = sd(rets_sd$returns,na.rm = T)
                        return(stdev)
                }) %>% as.data.frame()

                te_after = sapply(dates_alpha$dates,function(x) {
                        rets_sd =  data_returns %>% dplyr::filter( dates > x & (dates <= ( x  %m+% mont

                        stdev = sd(rets_sd$returns,na.rm = T)

                        return(stdev)
                }) %>% as.data.frame()

                te_dif = data.frame(dates_alpha,te_before, te_after)
                colnames(te_dif) = c("returns","dates","te_before","te_after")
                #colnames(te_dif) = c("before","after")
                #te_dif$dif = te_dif[,1] - te_dif[,2]


        }
        return(te_dif)
}
```

Percentile

```r
get10Date_months_Perc = function(returns,alpha,dates,m,buffer = FALSE,allReturns,EOY = FALSE){

      allReturns = as.data.table(allReturns)

        #If buffer is true
        if(buffer) {b = 12} else {b = 0}


        returns = returns %>% as.data.frame()
        alpha = alpha %>% as.data.frame()
        dates = dates %>% as.data.frame()

        #mgr = colnames(alpha)
```

```
colnames(alpha) = "returns"
colnames(returns) = "returns"
colnames(dates) = "dates"



data_alpha = cbind(alpha,dates) %>% as.data.table() %>%
        arrange(dates) %>%
        mutate(dates = as_date(dates))


dates_alpha = data_alpha %>%
        dplyr::filter(returns>0.1 & dates >= min(dates) %m+% months(m)) #& month(dates)==12)


if(EOY){
      dates_alpha = dates_alpha %>% dplyr::filter(month(dates)==12)
}




if (nrow(dates_alpha) ==0) {

        te_dif = data.frame(0,as_date(0),0,0)
        colnames(te_dif) = c("returns","dates","te_before","te_after")


} else{

        data_returns = cbind(returns,dates) %>% as.data.frame() %>%
                arrange(dates) %>%
                mutate(dates = as_date(dates))  %>%
              as.data.table()




        te_before = sapply(dates_alpha$dates,function(x) {


                date_min = x %m-% months(m+b)
                date_max = x  %m+% months(m)


                allReturns_sub = allReturns %>%  dplyr::filter(variable >= date_min & variable <


               #allTE = allReturns_sub %>% select(-variable) %>% summarise_all(sd) %>% t()


              #allTE = apply(allReturns_sub %>% select(-variable),2,sd)
              allTE = colSds(allReturns_sub %>% select(-variable) %>% as.matrix())

                rets_sd =  data_returns %>% dplyr::filter(dates< (x %m-% months(b)) & (x %m-% mo

                stdev = sd(rets_sd$returns,na.rm = T)

                #Takes too long
                #perc_fun = ecdf(allTE)
                #perc = perc_fun(stdev)

                r = rank(c(stdev,allTE))

                perc = r[[1]]/length(allTE)

                return(perc)
        }) %>% as.data.frame()

        te_after = sapply(dates_alpha$dates,function(x) {
                print(x)
                date_min = x %m-% months(m+b)
                date_max = x  %m+% months(m)

                allReturns_sub = allReturns %>% dplyr::filter(variable > x & variable <= date_ma

                if (nrow(allReturns_sub)>0) {

                #allTE = allReturns_sub %>% select(-variable) %>% summarise_all(sd) %>% t()

                #allTE = apply(allReturns_sub %>% select(-variable),2,sd)
                allTE = colSds(allReturns_sub %>% select(-variable) %>% as.matrix())

                rets_sd =  data_returns %>% dplyr::filter( dates > x & (dates <= ( x  %m+% mont

                stdev = sd(rets_sd$returns,na.rm = T)

                #Takes too long
                #perc_fun = ecdf(allTE)
                #perc = perc_fun(stdev)

                r = rank(c(stdev,allTE))
                perc = r[[1]]/length(allTE)
                } else {perc = NA}


                return(perc)
        }) %>% as.data.frame()

        te_dif = data.frame(dates_alpha,te_before, te_after)
        colnames(te_dif) = c("returns","dates","te_before","te_after")
        #colnames(te_dif) = c("before","after")
        #te_dif$dif = te_dif[,1] - te_dif[,2]
```

```
        }
        return(te_dif)
}
```

Function with no overlapping periods

```
get10Date_months_no_overlap = function(returns,alpha,dates,m,buffer){


        #If buffer is true
        if(buffer) {b = 12} else {b = 0}


        returns = returns %>% as.data.frame()
        alpha = alpha %>% as.data.frame()
        dates = dates %>% as.data.frame()


        colnames(alpha) = "returns"
        colnames(returns) = "returns"
        colnames(dates) = "dates"


        data_alpha = cbind(alpha,dates) %>% as.data.frame() %>%
                arrange(dates) %>%
                mutate(dates = as_date(dates))

        dates_alpha = data_alpha %>%
                dplyr::filter(returns>0.1 )#& month(dates)==12)








        if (nrow(dates_alpha) ==0) {

                te_dif = data.frame(0,as_date(0),0,0)
                colnames(te_dif) = c("returns","dates","te_before","te_after")


        } else{

                #Remove Overlapping Periods
                for (i in 1:nrow(dates_alpha)) {

                        d = dates_alpha$dates[i]

                        if(is.na(d)){} else{

                                dates_alpha =  dates_alpha %>%
                                        filter(!(dates >d & dates < d %m+% months(12)))
                        }

                }

                #Create Return df
                data_returns = cbind(returns,dates) %>% as.data.frame() %>%
                        arrange(dates) %>%
                        mutate(dates = as_date(dates))


                #Calculate TE before and after

                te_before = sapply(dates_alpha$dates,function(x) {
                        rets_sd =  data_returns %>% dplyr::filter(dates<= (x %m-% months(b)) & (x %m-%
                        stdev = sd(rets_sd$returns,na.rm = T)
                        return(stdev)
                }) %>% as.data.frame()

                te_after = sapply(dates_alpha$dates,function(x) {
                        rets_sd =  data_returns %>% dplyr::filter( dates > x & (dates <= ( x  %m+% mont

                        stdev = sd(rets_sd$returns,na.rm = T)

                        return(stdev)
                }) %>% as.data.frame()

                #Calculate Average TE for the rolling period #TODO




                #Create Data Frame
                te_dif = data.frame(dates_alpha,te_before, te_after)
                colnames(te_dif) = c("returns","dates","te_before","te_after")



        }
        return(te_dif)
}
```

Vectorize Functions for Number of Months

```
get_Mean_TE_Results = function(cl,months,buffer,rolling_alpha,dates,returns_subset_no_dates){

      #Get number of managers
      managers = 1:ncol(rolling_alpha)

      results_months = parLapply(cl,managers, function(x){
              print(x)

              get10Date_months(
                      returns_subset_no_dates[,x],
                      rolling_alpha[,x],
                      dates,
                      months,
                      buffer
              )})


      #Name values
      names(results_months) = colnames(returns_subset_no_dates)

      #Bind into a dataframe
      results_months_df = bind_rows(results_months,.id = "manager")
      results_months_df$dif = results_months_df$te_before - results_months_df$te_after

      te_after_mean =  mean(results_months_df$te_after,na.rm = T)
      te_before_mean =  mean(results_months_df$te_before,na.rm = T)

      t_test_result = t.test(results_months_df$te_before,results_months_df$te_after,alternative = "grea


      te_table =  tribble(
            ~Months, ~Before_TE, ~After_TE, ~Mean_Difference, ~Mean_Difference_Percentage, ~T_Statistic
            months, te_before_mean, te_after_mean, te_after_mean-te_before_mean, (te_after_mean / te_be
      )


      return(te_table)


}

get_Percentile_TE_Results = function(cl,months,buffer,rolling_alpha,dates,returns_subset_no_dates,return

      #Get number of managers
      managers = 1:ncol(rolling_alpha)

      results_months_perc = parLapply(cl,managers, function(x){

              print(x)

              get10Date_months_Perc(
                      returns_subset_no_dates[,x],
                      rolling_alpha[,x],
                      dates,
                      months,
                      buffer,
                      returns_subset_no_na,
                      EOY = FALSE
              )})
      #Name values
      names(results_months_perc) = colnames(returns_subset_no_dates)

      #Bind into a dataframe???
      results_months_perc_df = bind_rows(results_months_perc,.id = "manager")
      results_months_perc_df$dif = results_months_perc_df$te_before - results_months_perc_df$te_after

      te_after_perc =  mean(results_months_perc_df$te_after,na.rm = T)
      te_before_perc =  mean(results_months_perc_df$te_before,na.rm = T)

      t_test_result = t.test(results_months_perc_df$te_before,results_months_perc_df$te_after,alternati



      te_table =  tribble(
            ~Months, ~Before_TE, ~After_TE, ~Mean_Difference, ~T_Statistic,
            months, te_before_perc, te_after_perc, te_after_perc-te_before_perc, t_test_result$statisti
      )


      return(te_table)


}
```

Create Parallel Cluster and Export Functions

```
suppressPackageStartupMessages({
#Create Parallel Cluster
cl = makeCluster(getOption("cl.cores",4))



#Export needed variables and cuntions
clusterExport(cl,varlist = c("get10Date_months","get10Date_months_Perc",'returns_subset_no_dates','rolli

clusterEvalQ(cl,library(tidyverse,quietly = T))
clusterEvalQ(cl,library(lubridate,quietly = T))
clusterEvalQ(cl,library(data.table,quietly = T))
clusterEvalQ(cl,library(matrixStats,quietly = T))
})
#
#
# results_months = pblapply(cl = cl,X = managers, FUN = function(x){
#
#         print(x)
#
#         get10Date_months(
#                 returns_subset_no_dates[,x],
#                 rolling_alpha[,x],
#                 dates,
#                  24,
#                  FALSE
#          )})
```

Results

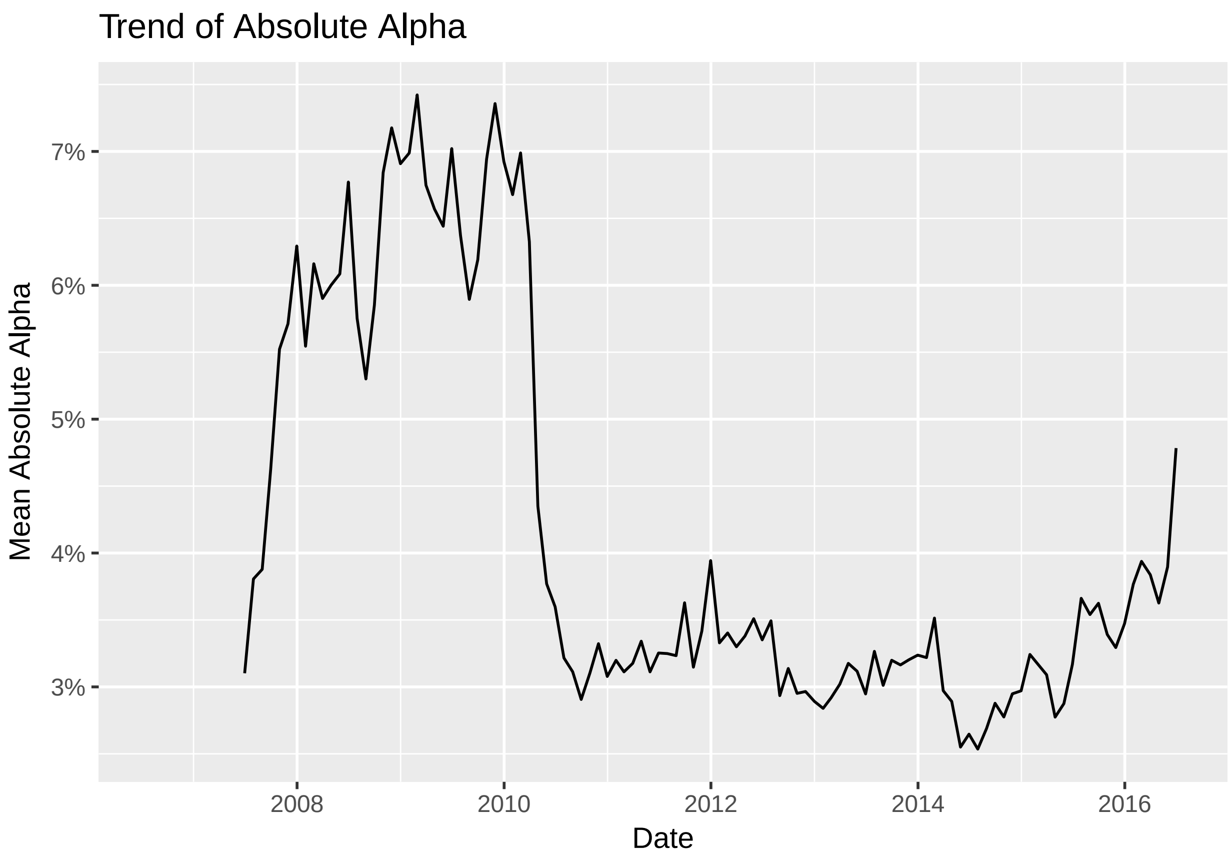
Run Mean TE Function for 6, 12, 24 Month Periods

```
mean_te_list = lapply(c(6,12,24), function(months){

      get_Mean_TE_Results(cl = cl,months = months,buffer = FALSE, rolling_alpha = rolling_alpha, dates =


})


mean_te_df = mean_te_list %>% bind_rows() %>%
      mutate_at(c("Before_TE","After_TE","Mean_Difference","Mean_Difference_Percentage"),function(x){pe
      mutate(T_Statistic = round(T_Statistic,2))

kable(mean_te_df)
```

Months	Before_TE	After_TE	Mean_Difference	Mean_Difference_Percentage	T_Statistic
6	2.13%	2.03%	-0.10%	-4.61%	7.23
12	2.20%	2.06%	-0.14%	-6.55%	10.46
24	2.14%	1.93%	-0.21%	-9.73%	17.80

Mean TE over Time

```
rolling_alpha_df_abs = abs(rolling_alpha_df)

abs_alpha = rowMeans(rolling_alpha_df_abs,na.rm = T) %>% as.data.frame()

abs_alpha$dates = dates[,1]

colnames(abs_alpha) = c("Mean_Absolute_Alpha","Date")


ggplot(abs_alpha,aes(x = Date, y = Mean_Absolute_Alpha)) + geom_line() +
      ggtitle("Trend of Absolute Alpha") + scale_y_continuous(labels = percent) +
      ylab("Mean Absolute Alpha")
```



Run Mean TE Percentile Function for 6, 12, 24 Month Periods

```
perc_te_list = lapply(c(6,12,24), function(months){

      get_Percentile_TE_Results(
            cl = cl,
            months = months,
            buffer = FALSE,
            rolling_alpha = rolling_alpha,
            dates = dates,
            returns_subset_no_dates = returns_subset_no_dates,
            returns_subset_no_na = returns_subset_no_na)


})


perc_te_df = perc_te_list %>% bind_rows() %>%
      mutate_at(c("Before_TE","After_TE","Mean_Difference"),function(x){percent(round(x,4))}) %>%
      mutate(T_Statistic = round(T_Statistic,2))

kable(perc_te_df)
```

Months	Before_TE	After_TE	Mean_Difference	T_Statistic
6	64.98%	63.60%	-1.38%	5.02
12	67.63%	65.38%	-2.25%	9.67
24	65.77%	63.06%	-2.72%	11.45

Run Mean TE Function With Buffer for 6, 12, 24 Month Periods

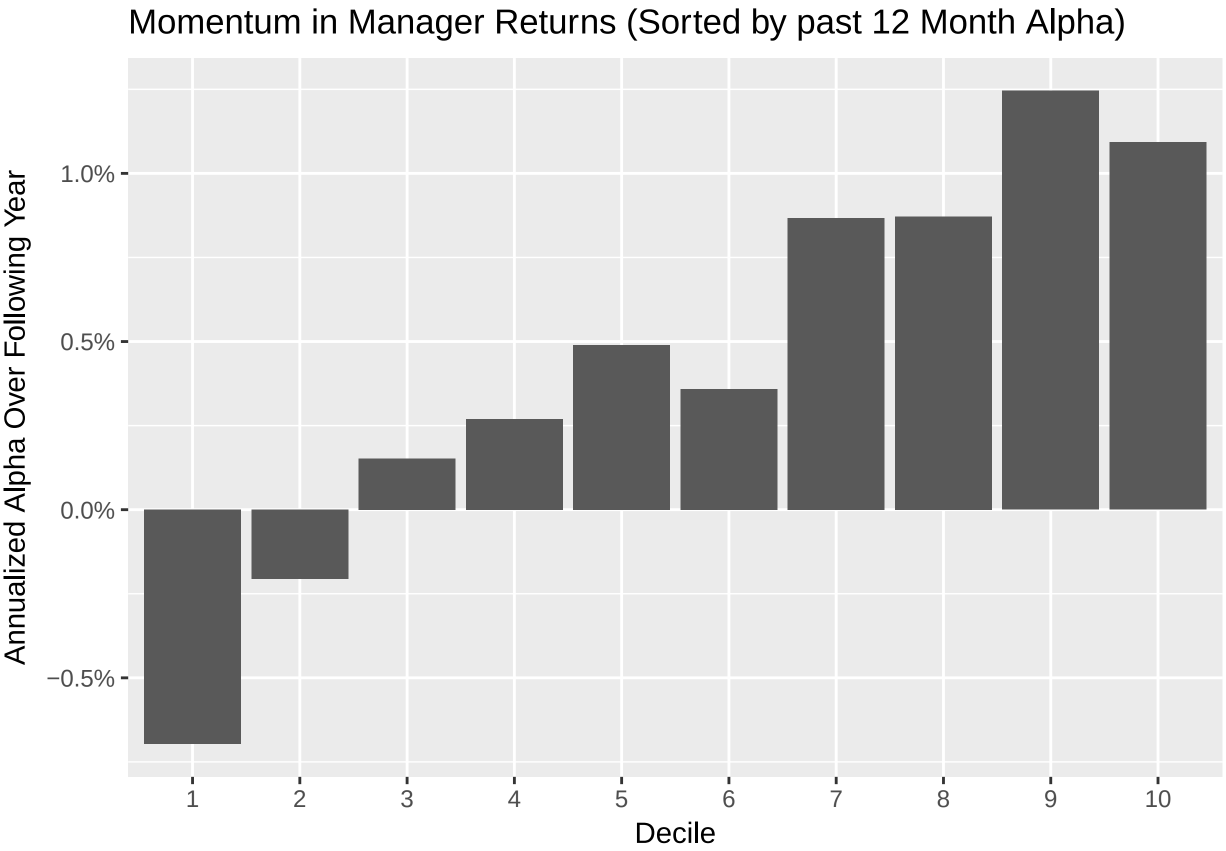
```
mean_te_list_buffer = lapply(c(6,12,24), function(months){

      get_Mean_TE_Results(cl = cl,months = months,buffer = TRUE, rolling_alpha = rolling_alpha, dates =


})


mean_te_df_buffer = mean_te_list_buffer %>% bind_rows() %>%
      mutate_at(c("Before_TE","After_TE","Mean_Difference","Mean_Difference_Percentage"),function(x){pe
      mutate(T_Statistic = round(T_Statistic,2))

kable(mean_te_df_buffer)
```

Months	Before_TE	After_TE	Mean_Difference	Mean_Difference_Percentage	T_Statistic
6	1.82%	2.03%	0.21%	11.41%	-12.15
12	1.79%	2.06%	0.26%	14.73%	-17.33
24	1.71%	1.93%	0.23%	13.39%	-18.06

Momentum in Manager Returns

```
monthly_ranks = apply(rolling_alpha %>% na.omit(),1,rank)
monthly_perc = monthly_ranks/nrow(monthly_ranks)

#rolling_alpha_melt = rolling_alpha %>% mutate(n = as.integer(rownames(rolling_alpha))) %>% melt(id.var

returns_subset_melt = returns_subset %>% select(-variable) %>% mutate(n = as.integer(rownames(returns_s
        melt(id.vars = "n")

top_decile = monthly_perc %>% melt() %>% filter(value>0.9) %>%
        left_join(returns_subset_melt, by = c("Var2" = "n","Var1" = "variable")) %>%
        group_by(Var2) %>% summarise(ret= mean(value.y))

top_decile_sd = sd(top_decile$ret)


deciles = ( seq(0.0,0.9,0.1))


getDecileSD = function(dec){

        decile_ret = monthly_perc %>% melt() %>% filter(value>dec & value < (dec+0.1)) %>%
```

```r
                mutate(np1 = Var2+1) %>%
        left_join(returns_subset_melt, by = c("np1" = "n","Var1" = "variable")) %>%
        group_by(Var2) %>% summarise(ret= mean(value.y,na.rm = T))

decile_sd = mean(decile_ret$ret,na.rm = T)

return(decile_sd)

        }




decile_sd_list = sapply(deciles,function(x) getDecileSD(x)) %>% as.data.frame() %>%
        mutate(decile = (deciles+0.1)*10)
colnames(decile_sd_list) = c("TE","Decile")

decile_sd_list = decile_sd_list %>%
      arrange(Decile) %>%
      mutate(Decile = factor (Decile,levels = Decile))

ggplot(decile_sd_list,aes(x = Decile,y= (1+TE)^12 -1)) + geom_col() +
        ylab("Annualized Alpha Over Following Year") +
      scale_x_discrete() +
      scale_y_continuous(label = percent) +
      ggtitle("Momentum in Manager Returns (Sorted by past 12 Month Alpha)")
```



Momentum in Manager Returns (Sorted by past 12 Month Alpha)