BIG DATA REPORT

TIANZHIXI YIN, DAMIAN STANSBURY AND DONGYANG KUANG

ABSTRACT. In this project, we were handed the historical registrar data of the University of Wyoming. We tried to analyze the data and find meaningful results using statistical methods. We also tried to integrate our methods with an existing database management system so that other people can utilize our analytic approach easily to tackle their own problems.

1. Data Analysis

All our analyses were done in the statistical computing language R. We used the integrated development environment called RStudio (Version 0.98.507). The R packages we use include:

- ggplot2
- truncreg
- gclus
- ISLR
- tree
- randomForest
- MASS
- gbm
- maps

You need to install these packages before you can call them in R:

```
install.packages("ggplot2")
library(ggplot2)
```

We tried different kinds of regression models, random forest algorithm and many methods of data visualization. Some examples of our program can be seen in the appendix.

1.1. **Examples.**

Figure 1 shows our boxplot of the GPA in different degrees, our goal is to make it easy for others to plot a boxplot for any level, college or major. Figure 2 is an example of GPA between difference majors. We use average GPA as the horizontal axis and number of students as the vertical axis. We plot all the majors here so it is not easy to read. We will make it possible for others to select the majors they want to be plotted. Figure 3 is an example finding the high schools this university might want to invest in for recruitment. This plot only includes the high schools that send 40 to 100 students to this university. We want the users of our program to be able to change the axis limits. Figure 4 is an example of regression lines, in which we can find the relationship between variables, for instance UW GPA and ACT score. We hope to allow users to choose their own predictor, response variable and the categories they want to compare, like the two majors in this plot. Figure 5 shows the GPA distribution of

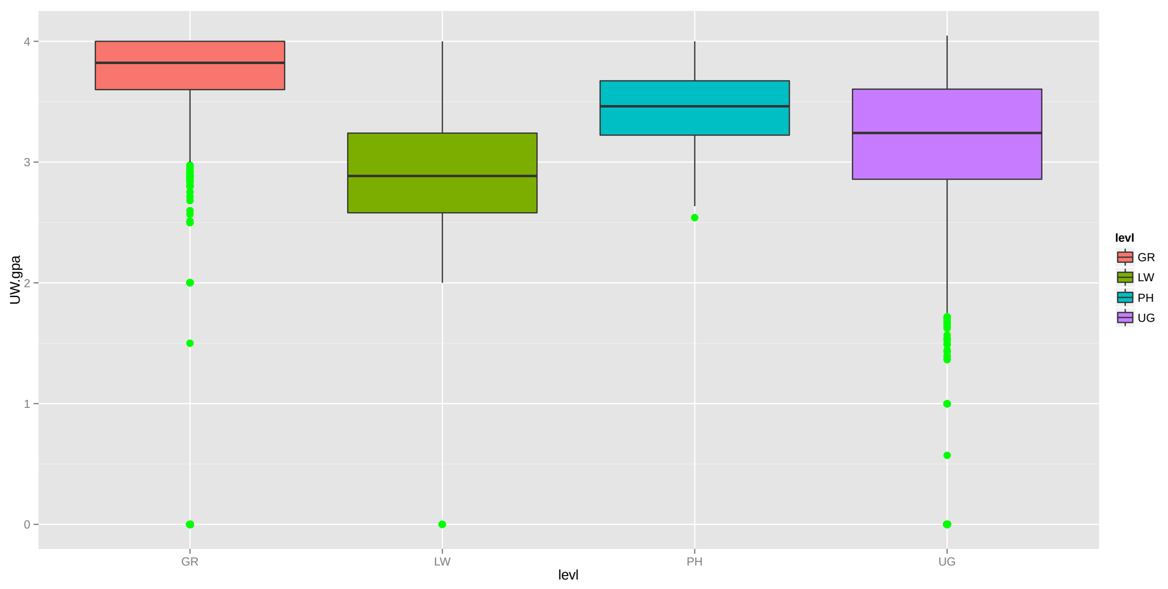


FIGURE 1. Boxplot For GPA in Different Levels.

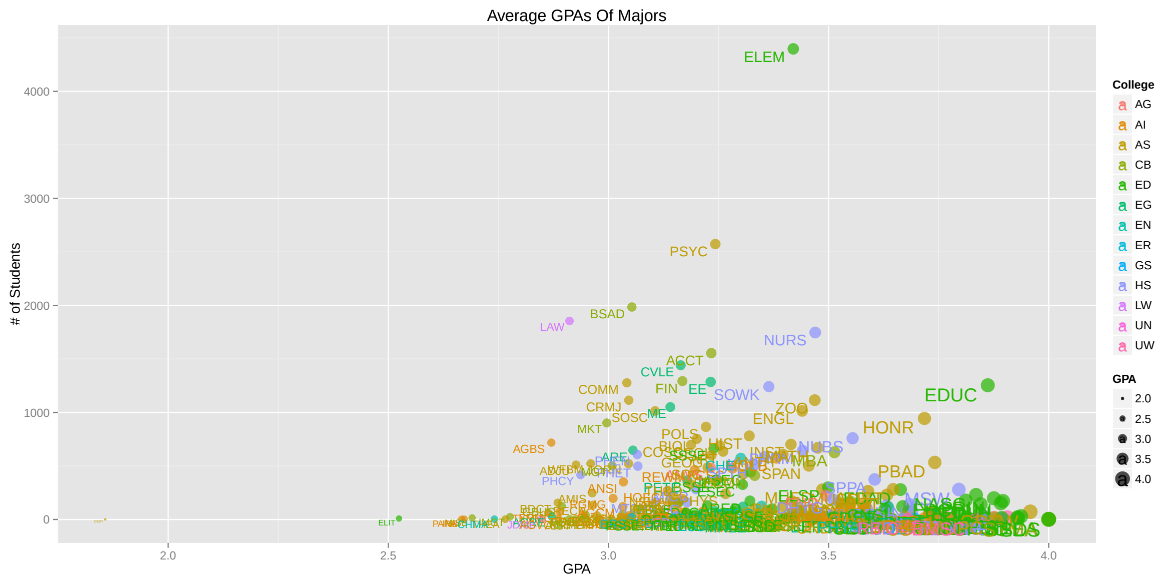


FIGURE 2. Average GPA and # of Students by Majors.

students with and without financial aid. Our hope is that other people can select any subjects they want to compare, for example high schools, majors, states etc. to make a plot like this.

Therefore our main task is to integrate our results in R with a powerful database. Our users can just type in simple structured queries and accomplish the statistical analyses that normal SQL could never do.

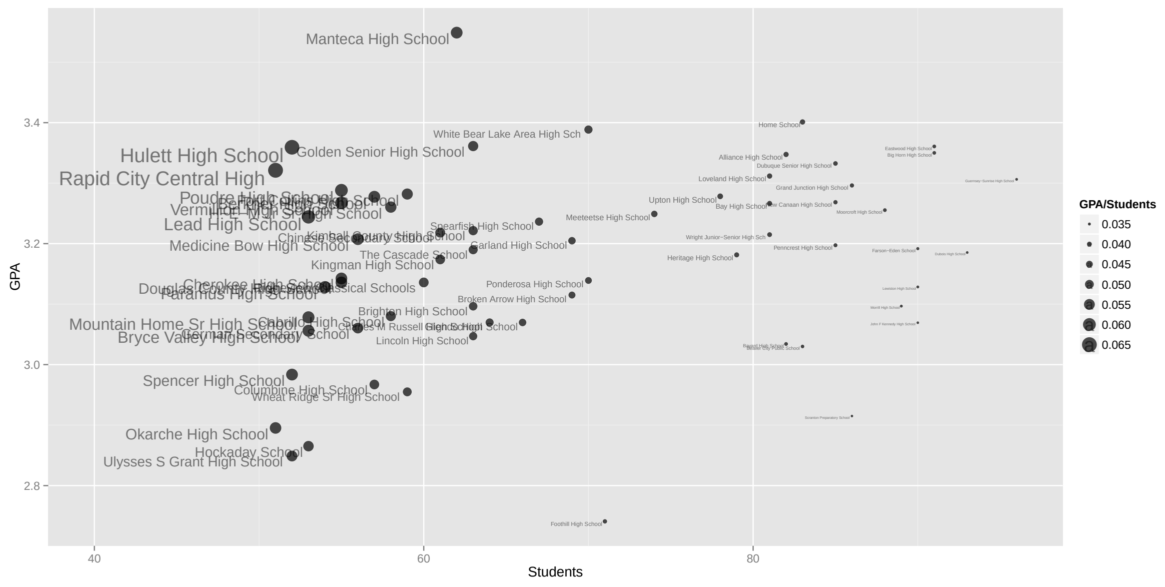


FIGURE 3. Average GPA and # of Students by High School.

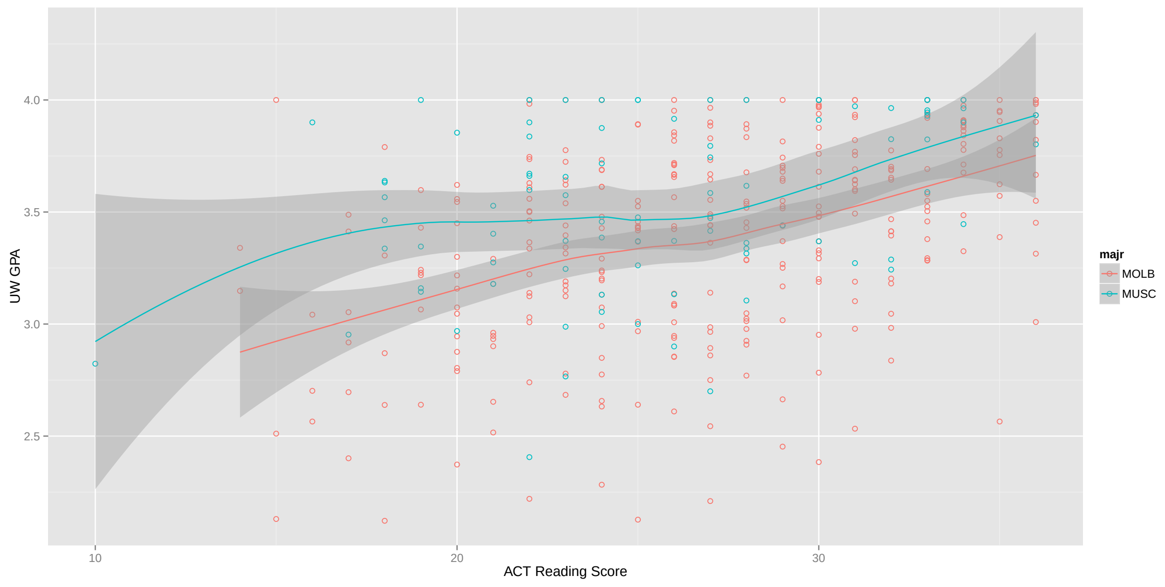


FIGURE 4. ACT Score and UW GPA.

2. POSTGRESQL + PL/R

We are running the system under a Windows environment. All the sql script is attached together with this report. Note: you should get R installed first.

2.1. **Installation.** The full description about the installation can be found at *http : //www.bostongis.com/PrinterFriendly.aspx?content_name = postgresql_plr_tut01*

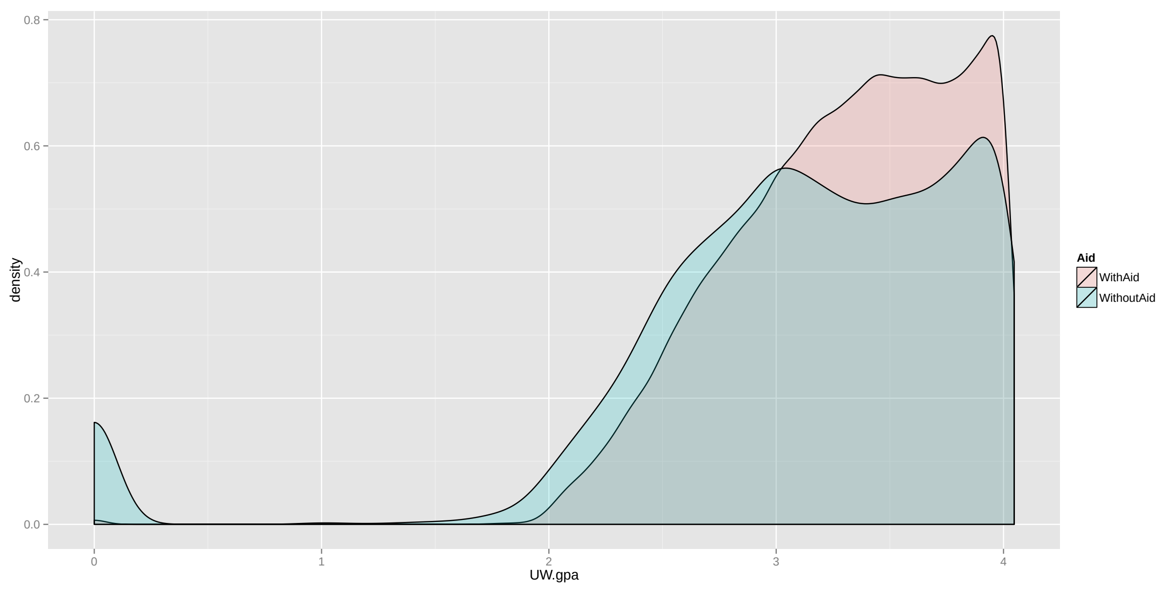


FIGURE 5. Performance With or Without Financial Aid.

2.2. **Importing the data as tables.** Once the installation is complete and the basic database is set(our name for the database is ”test1”), we can start creating corresponding tables to import the source data:

1.Click the database ”test1” inside the object browser and then click the ”SQL” button on the main menu to open the sql window to type in queries.

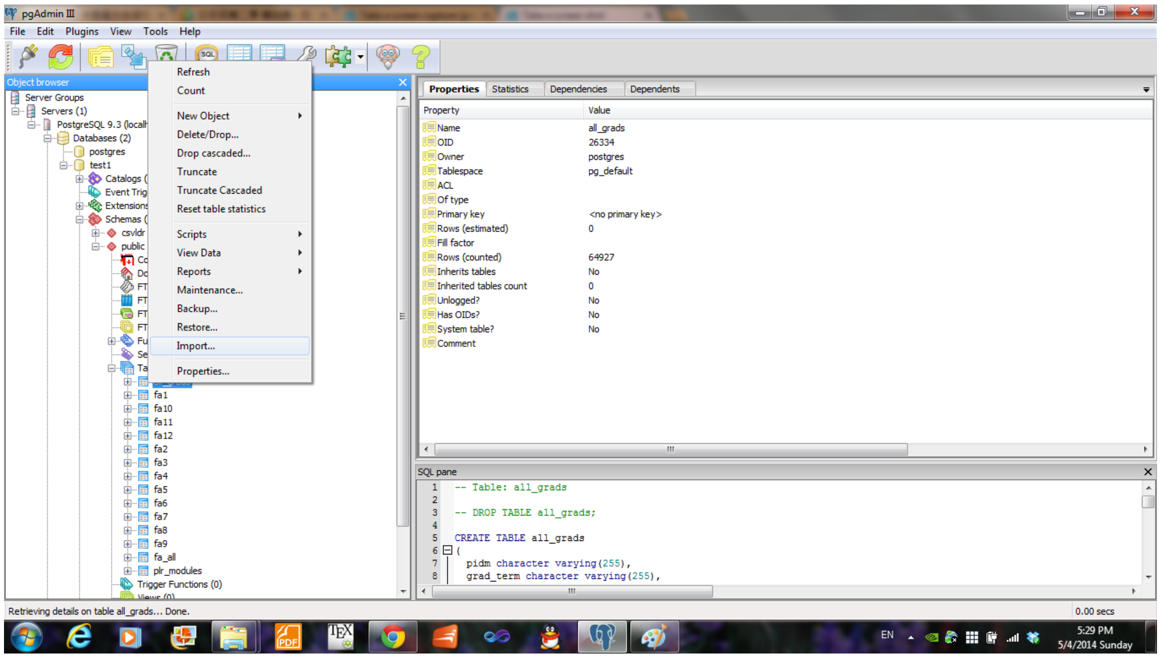
2.Run the ”create_table_allgrads.sql” in the window. When the query is done, an empty table is available in the object browser :”test1/schemas/public/Tables”. Right click on the newly created table, and select the import option.

3. Follow the instructions in the import window, you can now import the source data as a table easily.

4. Run ”create_table_fa.sql” in another sql window to create the table for the financial aid information and import the source data as stated above. Note: You may need to change the name of the table in the sql file each time if you want the 12 financial aid files as different tables.

5. You can also create the corresponding table by just right clicking the ”Table” object inside the object window. There will be a ”New table...” option.

2.2.1. *Problems met.* It did not go smoothly when we first imported the source data due to bad entries in them. The good thing is the database will tell you which row and column in the source data it can not import, so you can directly go to that entry and ”correct” it.



Property	Value
Name	all_grads
OID	26334
Owner	postgres
Tablespace	pg_default
ACL	
Of type	
Primary key	<no primary key>
Rows (estimated)	0
Fill factor	
Rows (counted)	64927
Inherits tables	No
Inherited tables count	0
Unlogged?	No
Has OIDs?	No
System table?	No
Comment	

```
-- Table: all_grads

-- DROP TABLE all_grads;

CREATE TABLE all_grads
(
  pidm character varying(255),
  grad_term character varying(255),
```

FIGURE 6. Import the data

2.3. **Writing PL/R functions.** Our function will be in the object "Function", all the functions can be combined with SQL clauses and used like the general sql functions.To create these functions, just run the corresponding sql files in the sql window.

2.3.1. *Loading R modules.* Since we are using raw R functions as the body of our wrapper functions in the database, we require some R packages or libraries pre-loaded when we connect to the database. See PL/R manual or our slides for details.

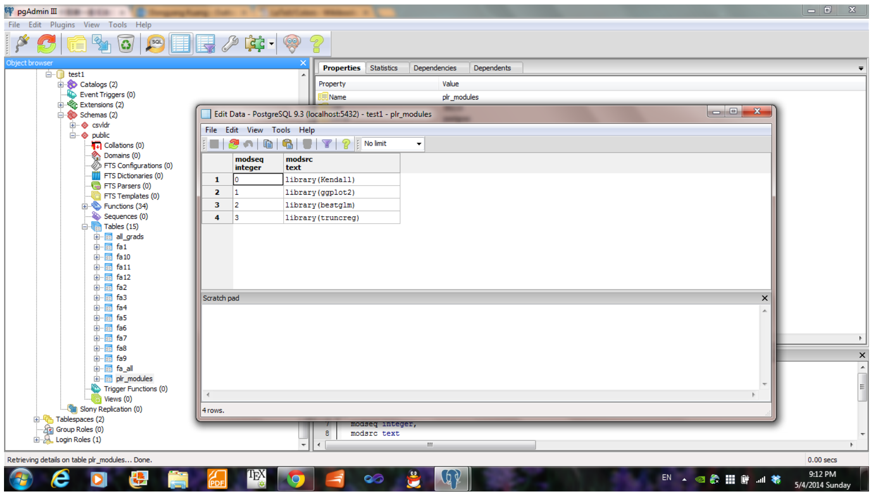
- Create the table to contain the library by running:
  CREATE TABLE plr_modules (
  modseq int4,
  modsrc text
  );
  (You just need to run it once, then the table will be seen in the object browser.)

- Insert into the table libraries you want:
  INSERT INTO plr_modules
  VALUES (0, 'library(Kendall)');

- Refresh the status by running: SELECT * from reload_plr_modules();

You should now see the "library Kendall" is in the table to use. To use the function in the library, you will have to write your own wrapper function.

2.3.2. *Current functions.* Currently, we have integrated several R functions(prototypes) into the data base:


pgAdmin III
Object browser
Properties
Statistics
Dependencies
Dependents
Edit Data - PostgreSQL 9.3 (localhost:5432) - test1 - plr_modules
No limit
Scratch pad
4 rows.
Retrieving details on table plr_modules... Done.

	modseq integer	modsrc text
1	0	library(Kendall)
2	1	library(ggplot2)
3	2	library(bestglm)
4	3	library(truncreg)

FIGURE 7. table of the R library

- plr_plot(float[],float[]): Scatter plot of two variables;
- plr_qqplot(float[],float[]): Quantile-Quantile plot of two columns;
- boxplot(float[]): Takes a numerical valued column and generate a boxplot;
- hist(float []): Takes a numerical valued column, generates a histogram, and shows frequencies of each subinterval;
- plr_qplot(float[], float[]): With proper parameters, it is a function that can make lots of fancy plots.By default, it generates a histogram.
- median(float []): An AGGREGATE function: takes a numerical valued column and shows the median;
- quantile(float [],float): An AGGREGATE function: takes a numerical valued column and a float type number and shows the corresponding quantile of the column;
- iqr(float[]): An AGGREGATE function: calculates the internal quantile range of a certain variable;
- plr_corr_Kendall(float[],float[]): Takes two numerical valued columns and calculates Kendall's correlation between them.
- lm_plr(float[],float[]): Takes two numerical valued columns and calculates the parameters of the linear regression (first argument vs. second argument).
- glm_plr(float[],float[]): Version of general linear regression;

- truncreg_plr(float[],float[]): Version of truncated linear regression;

2.3.3. *Before you use our functions.*

a) Before you run those linear regression functions, make sure you have created the type "lm_type" in the database(the script is in lm_plr.sql), so please run lm_plr.sql first before you create other linear regression functions;

b) To use the non-aggregate function, please use "array_agg(column_name)". If the function's input is type "float[]"; e.g. SELECT median(uw_gpa) FROM all_grads, while you should use SELECT lm_plr(array_agg(uw_gpa),array_agg(first_term_gpa) FROM all_grads.

c) For a function that plots, it won't actually plot until you exit out of the sql window. The plot is by default saved in "D:/" in pdf format unless you change it somewhere else by modifying the sql script for that function.

d) The functions we present here are actually simpler versions of the functions in R, restricting the inputs and outputs. The intention was for users to use them without much knowledge in R. If users know enough about these functions in R, he or she can actually add parameters or see other output by altering parameters directly in the sql scripts creating those functions.

2.3.4. *Problems met.* The biggest problem we met in this part is the datatype-mapping between R and PostgreSQL. We had to understand the input and output structures in R, which varied depending on different functions. We also had to consider how to treat the output from the database so that R functions can take them as input legally. For some the functions, in order to make them work, we restricted the R output and had to create corresponding composite types to hold those outputs. For example, we created the lm_type and used it in our wrapper function to hold the coefficient part from the output of corresponding R function.

2.4. **Future Works.** Currently we just have very simple R functions integrated into the data base. On one side, we will try to migrate as many R functions as possible; on the other hand, we will try to make the integrated functions more powerful by carefully studying the data types and their mapping between our database and R.

Appendix A. Code Listings

Listing 1. Truncated Regression Model in R.

```
fit <- truncreg(UW.gpa~levl, data=GPA, point = 4.047,
direction = "right")
summary(fit)
```

Listing 2. Multicollinearity Plot in R.

```
dta <- subset(GPA,SAT_Total!=0&ACT_Total!=0&hsgpa!=0,
select=c("SAT_Total","ACT_Total","hsgpa"))
dta.r <- abs(cor(dta,use="pairwise.complete.obs"))
dta.col <- dmat.color(dta.r)
dta.o <- order.single(dta.r)
windows()
cpairs(dta, dta.o, panel.colors=dta.col, gap=.5,
       main="Multicollinearity, Colored by Correlation" )
```

Listing 3. Classification Tree in R.

```
Perfect$ACT_Total<-Perfect$ACT_Reading+Perfect$ACT_Math+
Perfect$ACT_English+Perfect$ACT_Composite+Perfect$ACT_Sci_Reasoning
Perfect = subset(Perfect,ACT_Composite!=0&hsgpa!=0,
select=c("UW.gpa","levl","marital","sex","race","ACT_Composite","hsgpa"))
attach(Perfect)
High=ifelse(UW.gpa<=3.5,"No","Yes")
Perfect = data.frame(Perfect, High)
tree.Perfect=tree(High~.-UW.gpa,data=Perfect)
summary(tree.Perfect)
windows()
plot(tree.Perfect)
text(tree.Perfect,pretty=0)

boost.Perfect=gbm(UW.gpa~.,data=Perfect[train,],
distribution="gaussian",n.trees=10000,shrinkage=0.01,
interaction.depth=4)
windows()
summary(boost.Perfect)
windows()
plot(boost.Perfect,i="hsgpa")
windows()
plot(boost.Perfect,i="ACT_Composite")
windows()
```

```
plot(boost.Perfect,i="levl")
```

Listing 4. Code Examples For Plotting in R.

```
p <- ggplot(Level, aes(levl, UW.gpa))
windows()
p + geom_boxplot(aes(fill = levl),outlier.colour = "green",
outlier.size = 3)

intv = c(as.vector(as.data.frame(table(GPA["grad.term"]))[,1]))
cnts = c(as.vector(as.data.frame(table(GPA["grad.term"]))[,2]))
windows()
barplot(cnts, space=0, names = intv, xlab='Academic Term',
ylab = 'Counts',
        main='# of Students Graduated in Each Term')

qplot(grad.term, UW.gpa, data = avegradtermGPA,color=levl,
geom = c("point", "line"),
      xlab="Academic Term", ylab="Average GPA",
      main="UW Average GPA Over Time")

m <- ggplot(collGPA, aes(x = UW.gpa))
m <- m + geom_histogram(binwidth = 0.1,aes(fill = ..count..))
m <- m + aes(y = ..density..)+
  geom_density(colour="Red") + xlim(0, 4)
m + facet_wrap(~ coll)

ggplot(lowmajrperc,aes(x=reorder(majr,Perc),y=Perc,color="red"))+
  geom_point(size=8)+coord_flip()+
  theme(axis.text.y = element_text(color="black"))+
  theme(axis.text.x = element_text(color="black"))+
  theme(legend.position="none")+xlab("")+ylab("")+
  ggtitle("Percentage of Low Performance Students in Majors")

p <- qplot(Students, GPA, data = hschlexample,
size = GPA/Students, alpha = I(0.7),label=hschl) +
xlim(5, 20)+ ylim(3.5, 4)
p + geom_text(hjust=1.1, vjust=1.1,angle = 25,
alpha = I(0.5))+xlab("Number of Students")+ylab("Average GPA")
```

Listing 5. PL/R Example of Boxplot.

```
create or replace function boxplot(data float[])
returns text as
```

```
'
pdf("D:/boxplot.pdf");
boxplot(data);
dev.off;
print("done");
'
language 'plr' strict;

select boxplot(array_agg(uw_gpa)) from all_grads;
```

Listing 6. PL/R Example of Generalized Linear Model.

```
CREATE OR REPLACE FUNCTION glm_plr(y float8[], x float8[])
RETURNS SETOF lm_type AS
'
     m1<- glm(y~x)
     m1_s<- summary(m1)$coef
     temp_m1<- data.frame(rownames(m1_s), m1_s)
     return(temp_m1)
'
LANGUAGE 'plr';
```

Listing 7. PL/R Example of Finding Quantiles.

```
DROP AGGREGATE IF EXISTS quantile(float8, float8);
CREATE OR REPLACE FUNCTION r_quantile(float8[])
    returns float8 as
    '
    binsz <- arg1[1]
    thedata <- arg1[2:length(arg1)]
    quantile(thedata, binsz, na.rm=TRUE)
    ' language 'plr' strict ;
CREATE OR REPLACE FUNCTION q_array_accum(float8[], float8, float8)
    returns float8[] as '
    select case when $1 is NULL then array[$3] else $1 || $2 end
    ' language sql;
CREATE AGGREGATE quantile (float8, float8) (
        sfunc = q_array_accum,
        stype = float8[],
        finalfunc = r_quantile);
```