**University of Michigan (Fall 2015)**

**HS853: Scientific Methods for Health Sciences: Special Topics**

**Linear Modeling Review**

Contents

[I. Statistical Software – Pros/Cons Comparison 2](#_Toc436067867)

[II. Quality Control 4](#_Toc436067868)

[II. Multiple Linear Regression 12](#_Toc436067869)

[Visualizing the Regression-Model coefficients (effect-sizes) 20](#_Toc436067870)

[(Fixed-Effect) Linear Model Assumptions 21](#_Toc436067871)

[Protocols for rectifying residual plots indicating nonlinearity 24](#_Toc436067872)

[Lack of collinearity 24](#_Toc436067873)

[Heteroskedasticity 24](#_Toc436067874)

[Normality of residuals 26](#_Toc436067875)

[Outliers 27](#_Toc436067876)

[Independence 28](#_Toc436067877)

[III. Linear mixed effects analyses 29](#_Toc436067878)

[Fixed and random effects 29](#_Toc436067879)

[**Scientific Inference** 40](#_Toc436067880)

[**Mixed Effect Model Assumptions** 46](#_Toc436067881)

[Types of Data Analyses 49](#_Toc436067882)

[Mixed effects logistic regression 50](#_Toc436067883)

[IV. Machine Learning Algorithms 56](#_Toc436067884)

[**Prediction** 56](#_Toc436067885)

[Data Modeling/Training 56](#_Toc436067886)

[Appendix 58](#_Toc436067887)

[Example 1: Simulation (subject, day, treatment, observation) 58](#_Toc436067888)

[Example 2: Genotype-phenotype 61](#_Toc436067889)

[**References** 66](#_Toc436067890)

## Statistical Software – Pros/Cons Comparison

|  |  |  |
| --- | --- | --- |
| Statistical Software | Advantages | Disadvantages |
| R | * R is actively maintained (100,000 developers, 15K packages) * Excellent connectivity to various types of data and other systems * Versatile for solving problems in many domains * It’s free, open-source code * Anybody can access/review/extend the source code * R is very stable and reliable * If you change or redistribute the R source code, you have to make those changes available for anybody else to use * R runs anywhere (platform agnostic) * Extensibility: R supports extensions, e.g., for data manipulation, statistical modeling, and graphics * Active and engaged community supports R * Unparalleled question-and-answer (Q&A) websites * R connects with other languages (Java/C/JavaScript/Python/Fortran) & database systems, and other programs, SAS, SPSS, etc. * Other packages have add-ons to connect with R. SPSS has incorporated a link to R, and SAS has protocols to move data and graphics between the two packages | * Mostly scripting language * Steeper learning curve |
| SAS | * Large datasets * Commonly used in business & Government | * Expensive * Somewhat dated programming language * Expensive/proprietary |
| Stata | * Easy statistical analyses | * Mostly classical stats |
| SPSS | * Appropriate for beginners Simple interfaces | * weak in more cutting edge statistical procedures lacking in robust methods and survey methods |
| * <http://www.ats.ucla.edu/stat/mult_pkg/compare_packages.htm> * <https://en.wikipedia.org/wiki/Comparison_of_statistical_packages> | | |

|  |  |
| --- | --- |
| GoogleScholar Research Article Pubs | |
| |  |  |  |  | | --- | --- | --- | --- | | Year | R | SAS | SPSS | | 1995 | 8 | 8620 | 6450 | | 1996 | 2 | 8670 | 7600 | | 1997 | 6 | 10100 | 9930 | | 1998 | 13 | 10900 | 14300 | | 1999 | 26 | 12500 | 24300 | | 2000 | 51 | 16800 | 42300 | | 2001 | 133 | 22700 | 68400 | | 2002 | 286 | 28100 | 88400 | | 2003 | 627 | 40300 | 78600 | | 2004 | 1180 | 51400 | 137000 | | 2005 | 2180 | 58500 | 147000 | | 2006 | 3430 | 64400 | 142000 | | 2007 | 5060 | 62700 | 131000 | | 2008 | 6960 | 59800 | 116000 | | 2009 | 9220 | 52800 | 61400 | | 2010 | 11300 | 43000 | 44500 | | 2011 | 14600 | 32100 | 32000 |   require(ggplot2)  require(reshape)  Data\_R\_SAS\_SPSS\_Pubs <- read.csv('https://umich.instructure.com/files/522067/download?download\_frd=1', header=T)  df <- data.frame(Data\_R\_SAS\_SPSS\_Pubs)  # convert to long format  df <- melt(df , id.vars = 'Year', variable.name = 'Time')  ggplot(data=df, aes(x=Year, y=value, colour=variable, group = variable)) + geom\_line() + geom\_line(size=4) + labs(x='Year', y='Citations') |  |

## Quality Control

**Questions:**

* **Is the data what it’s supposed to (does it represent the study cohort/population)?**
* **How to inspect the quality of the data?**

Data Quality Control (QC) and Quality Assurance (QA) represent important components of all modeling, analytics and visualization that precede all subsequent data processing steps. QC and QA may be performed manually or automatically. Statistical quality control involves quantitative methods for monitoring and controlling a process or data derived from observing a natural phenomenon. For example, is there evidence in the plots below of a change in the mean of these processes?

# simulate data with base value of 100 w/ normally distributed error

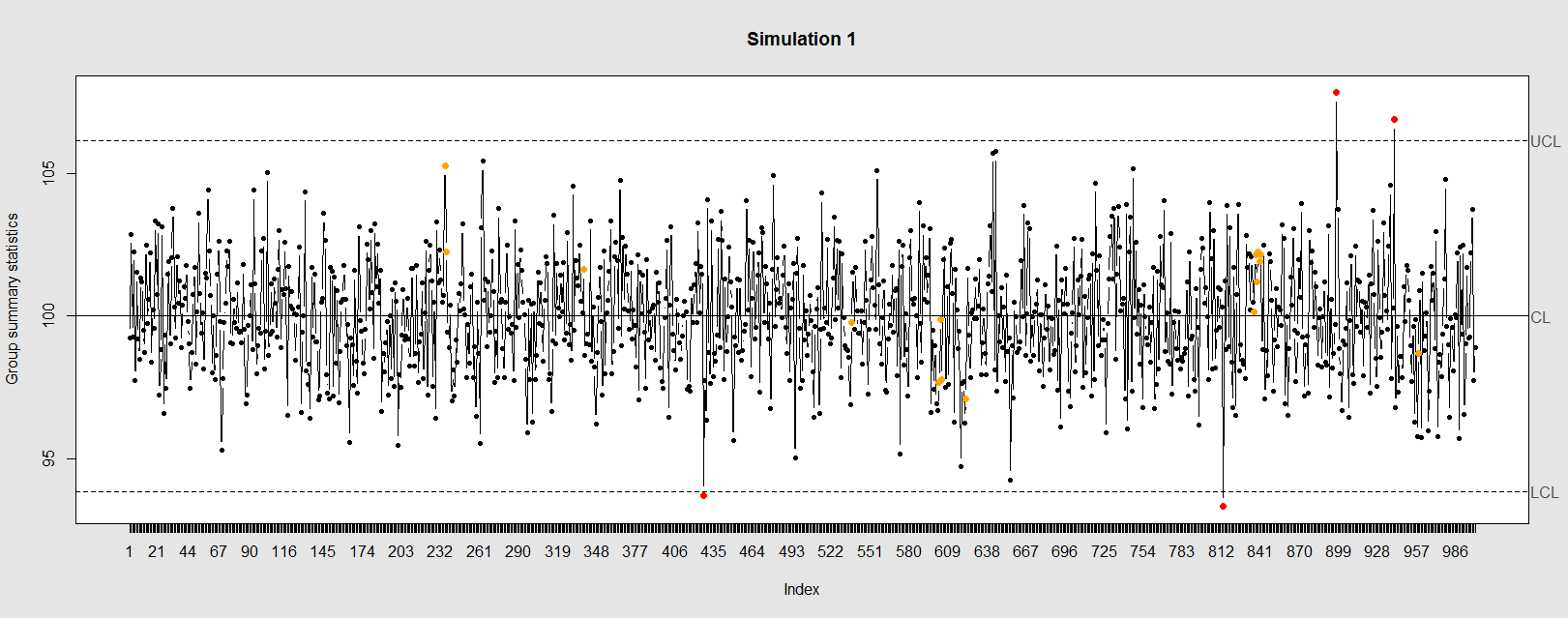
# install.packages("qcc")

library(qcc)

demo.data.1 <- rep(100, 1000) + rnorm(1000, mean=0, sd=2)

qcc(demo.data.1, type="xbar.one", center=100, add.stats=FALSE,

title="Simulation 1", xlab="Index")



Now let’s introduce a trend

# first 800 points have base value of 100 w/ normally distributed error,

# next 100 points have base value of 105 w/ normally distributed error

# last 100 points have base value of 110 w/ normally distributed error

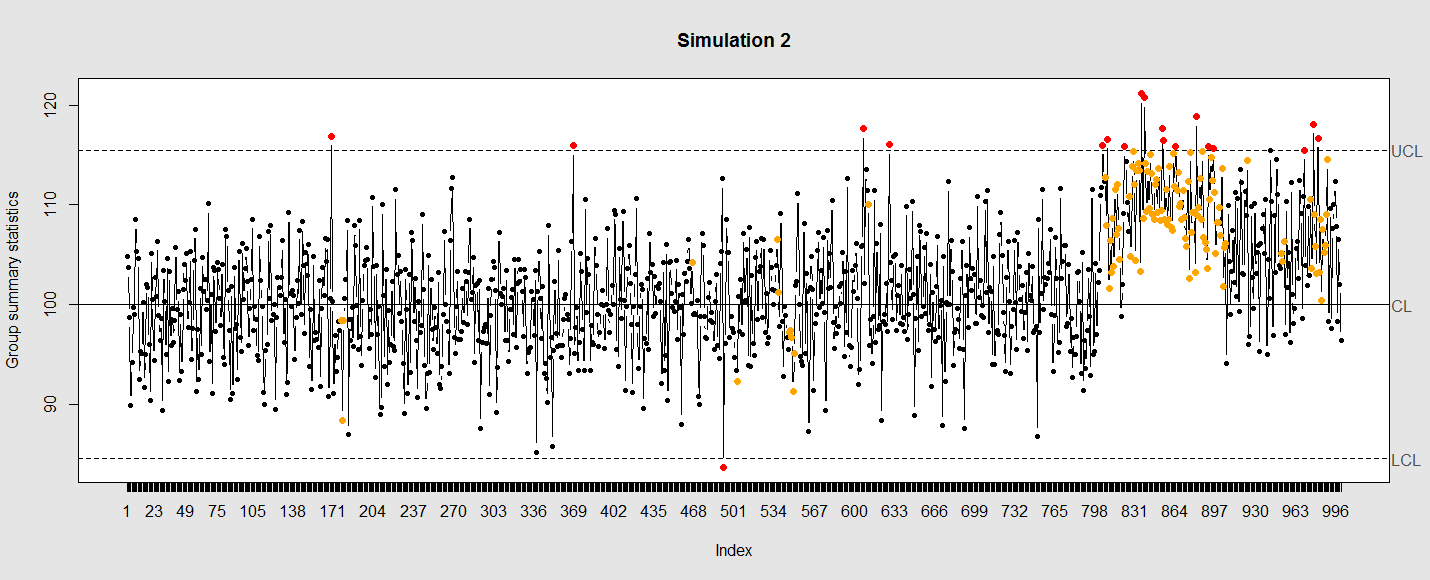
M <- 110

SD=5

demo.data.2 <- c(rep(100, 800), rep(M, 100), rep(100+(M-100)/2, 100)) + rnorm(1000, mean=0, sd=SD)

qcc(demo.data.2, type="xbar.one", center=100, add.stats=FALSE,

title="Simulation 2", xlab="Index")



Our goal is to use statistical quality control to automatically identify issues with the data. The qcc package in R provides methods for statistical quality control – given the data, it identifies candidate points as outliers based on the Shewhart Rules. Color-coding the data also helps point out irregular points.

The Shewhart control charts rules (cf. 1930’s) are based on monitoring events that unlikely when the controlled process is stable. Incidences of such atypical events are alarm signals suggesting that stability of the process may be compromised and the process is changed.

An instance of such an unlikely event is the situation when the upper/lower control limits (UCL or LCL) are exceeded. UCL and LCL are constructed as limits, indicating that the process is under control within them. Additional warning limits (LWL and UWL) are constructed at or . Other rules specifying events having low probability when the process is under control can be constructed:

1. One point exceeds LCL/UCL.
2. Nine points above/below the central line.
3. Six consecutive points show increasing/decreasing trend.
4. Difference of consecutive values alternates in sign for fourteen points.

5. Two out of three points exceed LWL or UWL limits.

6. Four out of five points are above/below the central line and exceed limit.

7. Fifteen points are within limits.

8. Eight consecutive values are beyond limits.

We can define training/testing dataset within qcc by adding the data we want to calibrate it with as the first parameter (demo.data.1), followed by the new data (demo.data.2) representing the test data.

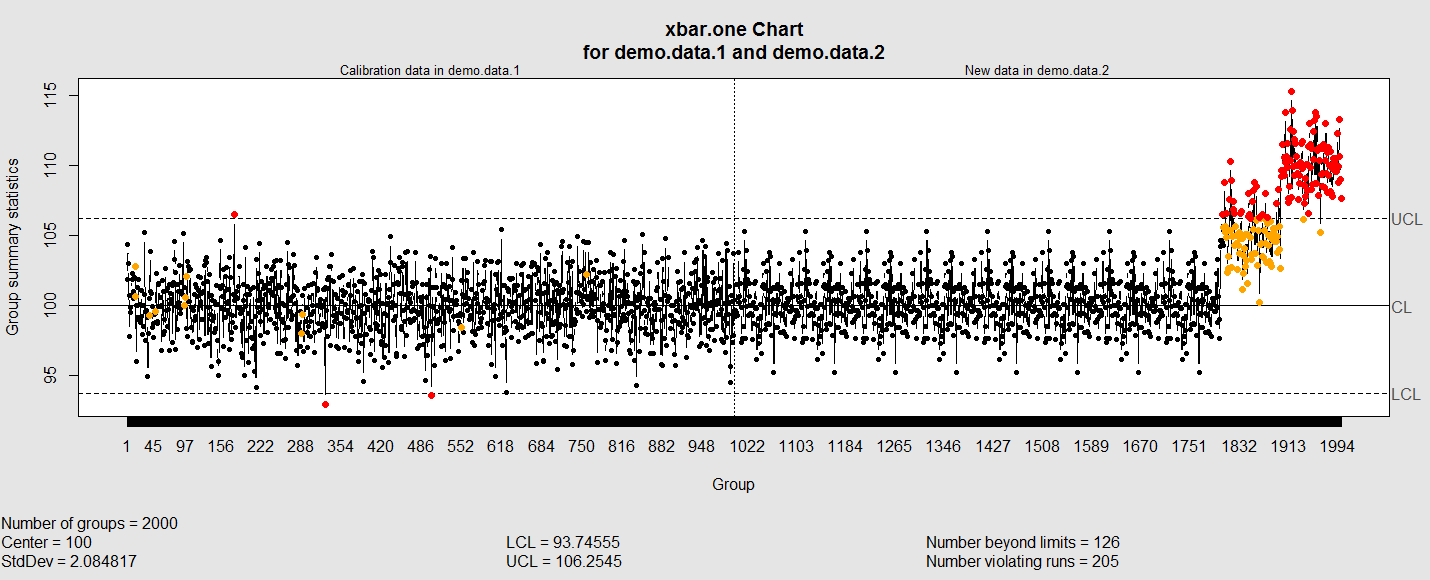
#example using holdout/test sets

demo.data.1 <- rep(100, 1000) + rnorm(1000, mean=0, sd=2)

demo.data.2 <- c(rep(100, 800), rep(105, 100), rep(110, 100)) + rnorm(100, mean=0, sd=2)

MyQC <- qcc(demo.data.1, newdata=demo.data.2, type="xbar.one", center=100, add.stats=FALSE, title="Simulation 1 vs. 2", xlab="Index")

plot(MyQC) # , chart.all=FALSE)



# add warning limits at 2 std. deviations

MyQC2 <- qcc(demo.data.1, newdata=demo.data.2, type="xbar.one", center=100, add.stats=FALSE, title="Second Simulation 1 vs. 2", xlab="Index")

warn.limits <- limits.xbar(MyQC2$center, MyQC2$std.dev, MyQC2$sizes, 0.95)

plot(MyQC2, restore.par = FALSE)

abline(h = warn.limits, lty = 2, lwd=2, col = "blue")

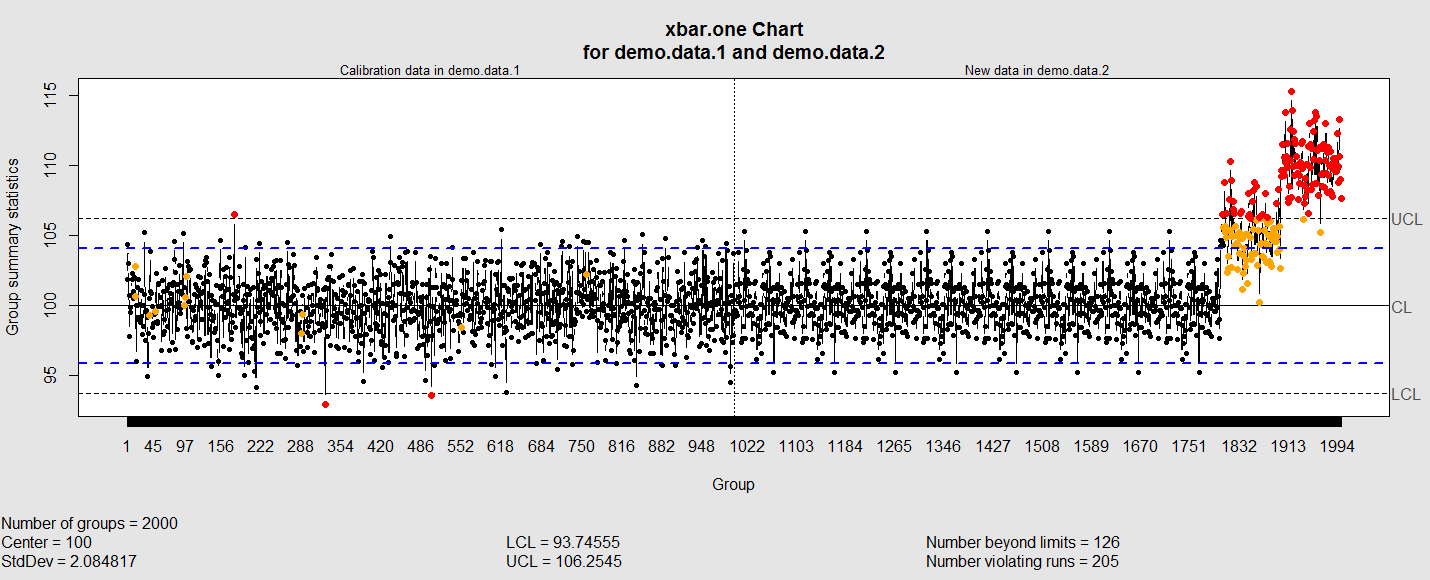
## limits.xbar(center, std.dev, sizes, conf)

Center = sample/group center statistic

Sizes= samples sizes.

std.dev= within group standard deviation.

Conf= a numeric value used to compute control limits, specifying the number of standard deviations (if conf > 1) or the confidence level (if 0 < conf < 1).



Natural processes may have errors that are non-normally distributed. However, using (appropriate) transformations we can often normalize the errors.

We can use thresholds to define zones in the data where each zone represents, say, one standard deviation span of the range of the dataset.

find\_zones <- function(x) {

x.mean <- mean(x)

x.sd <- sd(x)

boundaries <- seq(-3, 3)

# creates a set of zones for each point in x

zones <- sapply(boundaries, function(i) {

i \* rep(x.sd, length(x))

})

zones + x.mean

}

head(find\_zones(demo.data.2))

evaluate\_zones <- function(x) {

zones <- find\_zones(x)

colnames(zones) <- paste("zone", -3:3, sep="\_")

x.zones <- rowSums(x > zones) - 3

x.zones

}

evaluate\_zones(demo.data.2)

find\_violations <- function(x.zones, i) {

values <- x.zones[max(i-8, 1):i]

# rule4 <- ifelse(any(values > 0), 1,

rule4 <- ifelse(all(values > 0), 1,

ifelse(all(values < 0), -1,

0))

values <- x.zones[max(i-5, 1):i]

rule3 <- ifelse(sum(values >= 2) >= 2, 1,

ifelse(sum(values <= -2) >= 2, -1,

0))

values <- x.zones[max(i-3, 1):i]

rule2 <- ifelse(mean(values >= 3) >= 1, 1,

ifelse(mean(values <= -3) >= 1, -1,

0))

#values <- x.zones[]

values <- x.zones[max(i-3, 1):i]

rule1 <- ifelse(any(values > 2), 1,

ifelse(any(values < -2), -1,

0))

c("rule1"=rule1, "rule2"=rule2, "rule3"=rule3, "rule4"=rule4)

}

find\_violations(evaluate\_zones(demo.data.2), 20)

Now we can compute the rules for each point and assign a color to any violations.

library("plyr")

compute\_violations <- function(x, start=1) {

x.zones <- evaluate\_zones(x)

results <- ldply (start:length(x), function(i) {

find\_violations(x.zones, i)

})

results$color <- ifelse(results$rule1!=0, "pink",

ifelse(results$rule2!=0, "red",

ifelse(results$rule3!=0, "orange",

ifelse(results$rule4!=0, "yellow",

"black"))))

results

}

tail(compute\_violations(demo.data.2))

Now let’s make a quality control chart.

plot.qcc <- function(x, holdout) {

my.qcc <- compute\_violations(x, length(x) - holdout)

bands <- find\_zones(x)

plot.data <- x[(length(x) - holdout):length(x)]

plot(plot.data, col= my.qcc$color, type='b', pch=19,

ylim=c(min(bands), max(bands)),

main="QC Chart",

xlab="", ylab="")

for (i in 1:7) {

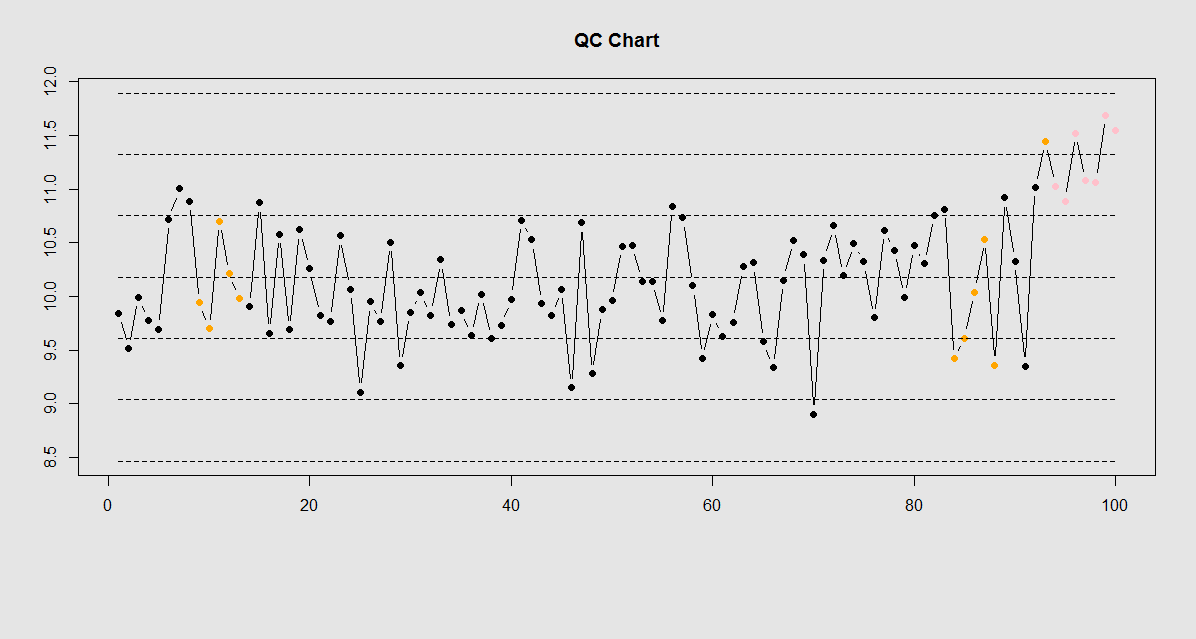
lines(bands[,i], col= my.qcc$color[i], lwd=0.75, lty=2)

}

}

demo.data.4 <- c(rep(10, 90), rep(11, 10)) + rnorm(100, mean=0, sd=0.5)

plot.qcc (demo.data.4, 100)



Let’s use the “Student's Sleep Data” (sleep) data.

library("qcc")

attach(sleep)

q <- qcc.groups(extra, group)

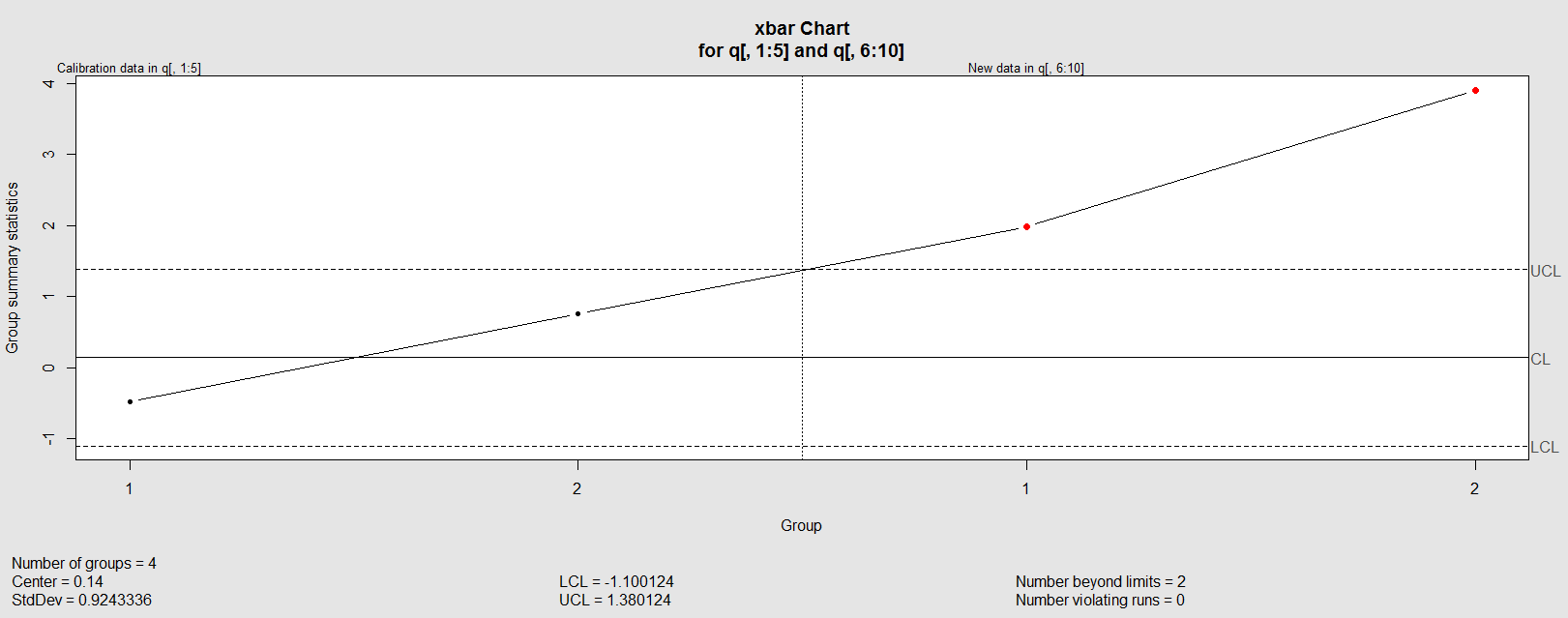
dim(q)

obj\_avg\_test\_1\_2 <- qcc(q[1:2,], type="xbar")

obj\_avg\_test\_1\_5 <- qcc(q[,1:5], type="xbar")

summary(obj\_avg\_test\_1\_5)

obj\_avg\_test\_train <- qcc(q[,1:5], type="xbar", newdata=q[,6:10])



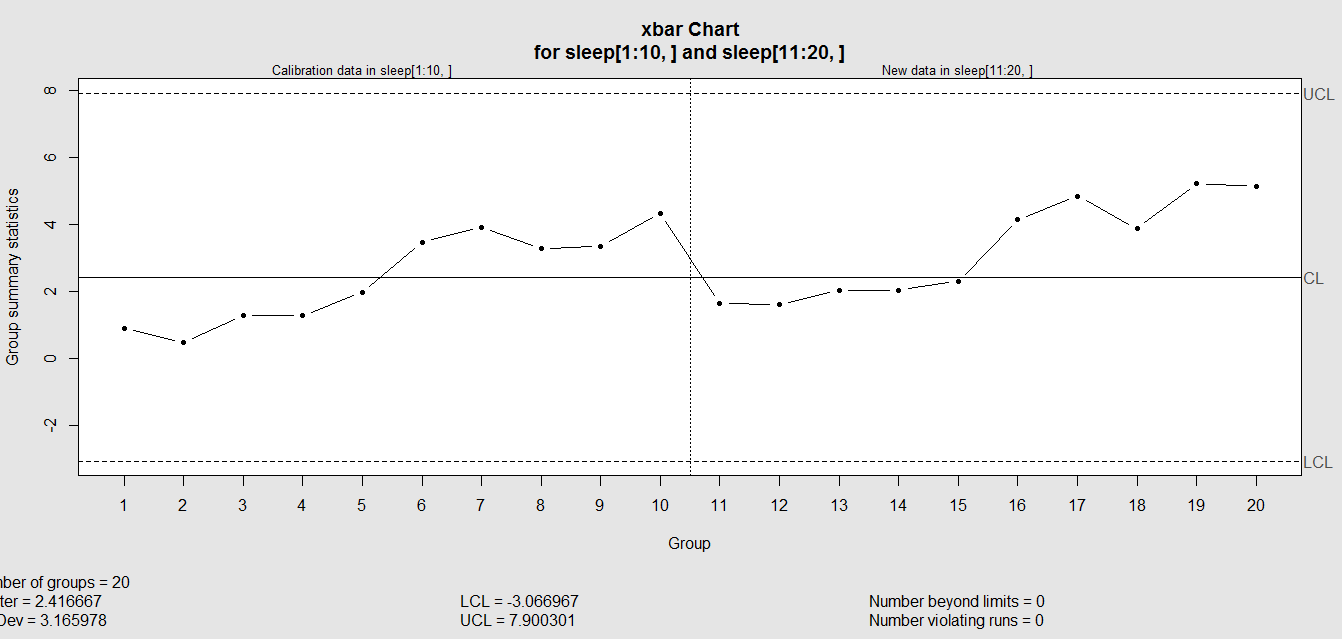
**# How is this different from this?**

obj\_avg\_new <- qcc(q[,1:10], type="xbar")

This control chart has the solid horizontal line (center), the upper and lower control limits (dashed lines), and the sample group statistics (e.g., mean) are drawn as a piece-wise line connecting the points. The bottom of the plot includes summary statistics and the number of points beyond control limits and the number of violating runs. If the process is “in-control”, we can use estimated limits for monitoring prospective (new) data sampled from the same process/protocol. For instance,

obj\_test\_1\_10\_train\_11\_20 <- qcc(sleep[1:10,], type="xbar", newdata=sleep[11:20,])

plots the X chart for training and testing (11-20) sleep data where the statistics and the control limits are based on the first 10 (training) samples.



Now try (range) QCC plot

obj\_R <- qcc(q[,1:5], type="R", newdata=q[,6:10])

A control chart aims to enhance our ability to monitor and track a process proxied by the data. When special causes of variation (random or not) are present the data may be considered “out of control”. Corresponding action may need to be taken to identify, control for, or eliminate such causes. A process is declared to be “controlled” if the plot of all data points are randomly spread out within the control limits. These Lower and Upper Control Limits (LCL, UCL) are usually computed as from the center (e.g., mean). This QCC default limits can be changed using the argument ***nsigmas*** or by specifying the confidence level via the ***confidence.level***  argument.

|  |  |
| --- | --- |
| Control chart variables | |
| "xbar" | Sample means are plotted to control the mean level of a continuous process variable. | |
| "xbar.one" | Sample values from a one–at–a-time data process to control the mean level of a continuous process variable. | |
| "R" | Sample ranges are plotted to control the variability of a continuous process variable. | |
| "S" | Sample standard deviations are plotted to control the variability of a continuous process variable. | |
| **Control charts for attributes** | | |
| "p" | The proportion of nonconforming units is plotted. Control limits are based on the binomial distribution | |
| "np" | The number of nonconforming units is plotted. Control limits  are based on the binomial distribution | |
| "c" | The number of defectives per unit are plotted. This chart assumes that defects of the quality attribute are rare, and the control limits are computed based on the Poisson distribution. | |
| "u" | The average number of defectives per unit is plotted. The Poisson distribution is used to compute control limits, but, unlike the *c* chart, this chart does not require a constant number of units. | |

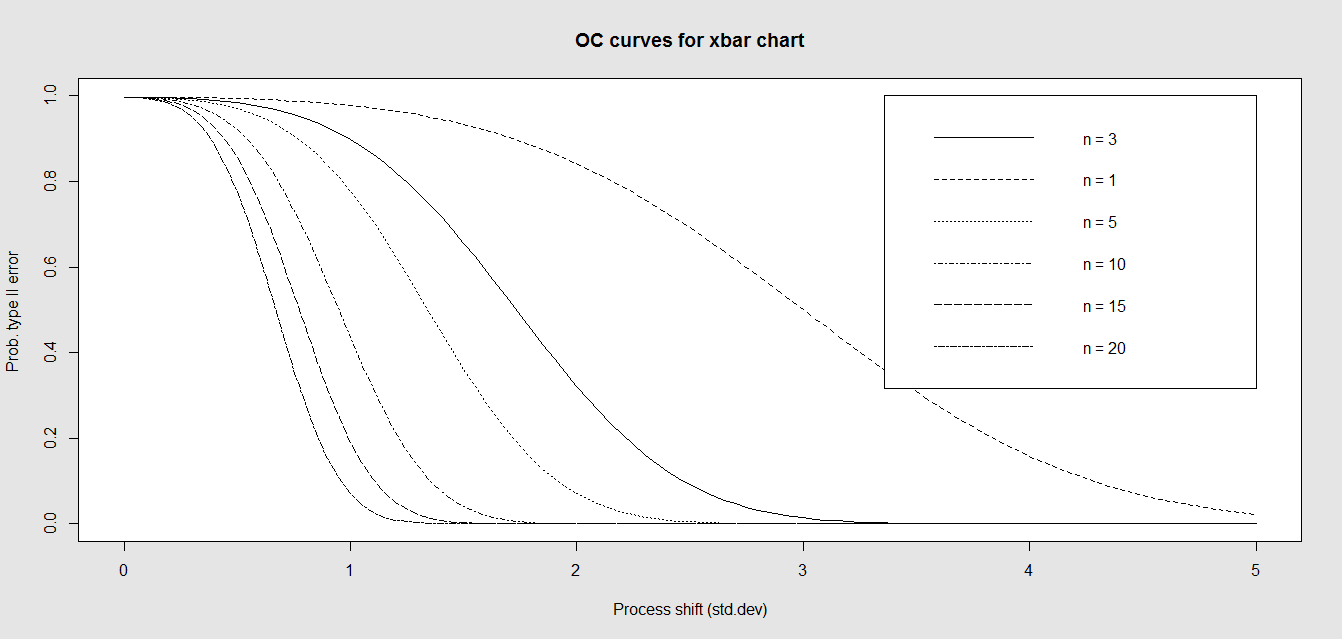
When the process is governed by a Gaussian distribution the limits correspond to a two-tails probability of p=0.0027.

Finally, an operating characteristic (OC) curve shows the probability of not detecting a shift in the process (false-negative, type II error), i.e., the probability of erroneously accepting a process as being “in control”, when in fact, it’s out of control.

# par(mfrow=c(1,1))

oc.curves(obj\_test\_1\_10\_train\_11\_20)

# oc.curves(obj\_test\_1\_10\_train\_11\_20, identify=TRUE) # to manually identify specific OC points



The function ***oc.curves*** returns a matrix or a vector of probabilities values representing the type II errors for different sample-sizes. See help(oc.curves), e.g., identify=TRUE, which allows to interactively identify values on the plot, for all options.

Notice that the OC curve is “S”-shaped. As expected, this is because as the percent of non-conforming values increases, the probability of acceptance decreases. A small sub-sample, instead of inspecting the entire data, may be used to determine the quality of a process. We can accept the data as in-control as long as the process percent nonconforming is below a predefined level.

## II. Multiple Linear Regression

**Questions**:

* Are there (linear) associations between predictors and a response variable(s)?
* When to look for linear relations and what assumptions play role?

Let’s use some of the data included in the Appendix. Snapshots of the first few rows in the data are shown below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Genotype | Race | Subject | Weight | | A | 1 | 1 | 8 | | A | 1 | 2 | 9 | | A | 1 | 3 | 11 | | A | 1 | 4 | 12 | | A | 1 | 5 | 10 | | A | 2 | 1 | 17 | | … |  |  |  | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Index | Subject | Day | Treatment | Obs | | 1 | 13 | Day1 | B | 6.472687 | | 2 | 14 | Day1 | B | 7.01711 | | 3 | 15 | Day1 | B | 6.200715 | | 4 | 16 | Day1 | B | 6.613928 | | 5 | 17 | Day1 | A | 6.829968 | | 6 | 18 | Day1 | A | 7.387583 | | … |  |  |  |  | |

Eyeballing the data may suggest that Race “2” subjects may have higher “Weights” (first dataset), or “Treatment A” yields higher “Observations” (second dataset). However this cursory observation may be incorrect as we may be looking at a small/exceptional sub-sample of the entire population. Data-driven linear modeling allows us to quantify such patterns and compute probability values expressing the strength of the evidence in the data to make such conclusions. In a nutshell, we express relationships of interest (e.g., weight as a function of race, or Observation as a function of treatment) using a linear function as an analytical representation of these inter-variable relations:

Weight ~ Race, Weight ~ Genotype, Obs ~ Treatment, Obs ~ Day, etc.

W = a +b\*R,

This “~” (tilde) notation implies “Weight predicted by Race” or “Observation as a linear function of Treatment”. The “dependent variable” (a measureable response) on the left is predicted by the factor on the right acting as an “independent variable”, covariate, predictor, “explanatory variable”, or “fixed effect”.

Often times interdependencies may not be perfect, deterministic, or rigid (like in the case of predicting the Area of a disk knowing its radius, or predicting the 3D spatial location of a planet having a precise date and time). Weight may not completely and uniquely determined by Race alone, as many different factors play role (genetics, environment, aging, etc.) Even if we can measure all factors we can identify as potentially influencing Weight, there will still be intrinsic random variability into the observed Weight measures, which can’t be control for. This intrinsic random variation may be captured and accounted for using “random” term at the end.

Weight ~ Race + ε ~ D(m=0, s).

Epsilon “ε” represent the error term of predicting Weight by Gender alone and summarize the aggregate impact of all factors aside from Race that impact individual’s Weight, which are experimentally uncontrollable or random. This formula a schematic analytic representation of a linear model that we’re going to estimate, quantify and use for prediction and inference. The right hand side splits the knowledge representation of Weight into 2 complementary components - a “fixed effect” for Race, which we understand and expect, and a “random effect” (“ε”) what we don’t know well. (W ~ R represents the “structural” or “systematic” part of the linear model and “ε” stands for the “random” or “probabilistic” part of the model.

**R Experiment (See Appendix)**

mydata1 <- data.frame(

Subject = c(13, 14, 15, 16, 17, 18),

Day = c("Day1", "Day1", "Day1", "Day2", "Day2", "Day2"),

Treatment = c("B", "B", "B", "A", "A", "A"),

Obs = c(6.472687, 7.017110, 6.200715, 6.613928, 6.829968, 7.387583)

)

We construct an R frame object [[1]](#footnote-1) concatenating 3 data-elements for 6 subjects, and saving it into “mydata1”.

mydata1

Subject Day Treatment Obs

1 13 Day1 B 6.472687

2 14 Day1 B 7.017110

3 15 Day1 B 6.200715

4 16 Day2 A 6.613928

5 17 Day2 A 6.829968

6 18 Day2 A 7.387583

Using the linear model Obs ~ Treatment + ε, we can invoke the linear modeling function

lm() [[2]](#footnote-2). The “ε” term is implicit in all models so it need not be specified.

lm.1 <- lm(Obs ~ Treatment, mydata1)

The assignment operator “<-” stores the linear model result in object lm.1. For these data (the data object is “mydata1”), this model expresses Obs as a function of Treatment. To inspect the result of the linear model use the “summarize” function summary():

AIC(lm.1); BIC(lm.1)

summary(lm.1)

The result is:

Call:

lm(formula = Obs ~ Treatment, data = mydata1)

Residuals:

1 2 3 4 5 6

-0.10342 0.44100 -0.37540 0.03782 -0.27881 0.27881

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 7.1088 0.2507 28.350 9.21e-06 \*\*\*

TreatmentB -0.5327 0.3071 -1.734 0.158

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3546 on 4 degrees of freedom

Multiple R-squared: 0.4293, Adjusted R-squared: 0.2866

F-statistic: 3.008 on 1 and 4 DF, p-value: 0.1579

The report shows:

* The model analytical formula specified by the lm() call.
* The residuals (errors, discrepancies between observed and model-predicted outcomes).
* The coefficients of the fixed effects (predictors, explanatory variables).
* And finally, the overall model quality (describing the ability of the model to describe the linear relations in these data). “Multiple R-squared” refers to the R2 statistic that measures the “variance explained by the model” or “variance accounted for by the model”. 0≤R2≤1, in our result, R2 = 0.4293, which is good, but not great. In essence, 42.93% of the data variability may be explained by this specific (best linear fit) model. In this case, the model solely relies on “Treatment” (fixed effect) to explain Obs (outcome). So, the R2 reflects how much of the Obs variance is accounted for by different Treatments (A or B).

Models with high *R2* values are preferred subject to 2 conditions:

* What is considered a high *R2* value is relative and depends on study/application.
* When the study phenomenon is highly deterministic, *R2* values can be approach 1.

Higher number of explanatory variable and higher model-complexity tend to yield higher *R2* values, but are more difficult to interpret.

The “Adjusted R-squared” value is a modified *R2* value normalizes the total variance “explained” by the model by the number of fixed effects included in the model. Larger number of predictors will lower the “Adjusted R-squared”. The adjusted *R2adj* = 0.2866, but in general, *R2adj* can be much lower when the model includes many fixed effects.

The statistical test of “model significance” is quantified by the output “F-statistic: 3.008 on 1 and 4 DF, p-value: 0.1579”. Under a null hypothesis where the model captures little or no information about the process, the probability value quantifies the data-driven evidence to reject the null and accept an alternative stating that this model does capture useful patterns in the data. Specifically, the p-value represents the conditional probability under the condition that the null hypothesis is true. In this case,

* The null hypothesis is Ho: “Treatment has no effect on Obs”.
* Alternative research Hypothesis is H1: “Treatment has effect on Obs”.

This p-value is 0.1579, and the linear model fails to reject the null hypothesis. This leaves open the possibility that Treatment may have no effect on Obs (just based on using these 6 data points).

However, if the p-value was much smaller, and assuming Ho were true, then this data would be less likely to be observed in reality (hence we would interpret small p-values as showing that the alternative hypothesis “Treatment affects Obs” as more likely and the model result is “statistically significant”).

There is a distinction between the overall “model-significance” (as quantified by the p-value at the very bottom of the output, which considers all effects together) and the p-value of individual effects (coefficients table including significance for each predictor).

The model F‐statistic and the degrees of freedom are in 1-1 correspondence with the p-value – the latter is computed as the right tail probability for an *F(df1,df2)* distribution corresponding to the F-statistics (critical value). To report the overall model inference in this case, we can state:

*“Using a simple linear model of Obs as a function of Treatment, the model was not significant (F(1,4)=3.001, p>0.15), which indicates that “Treatment” may not be a critical explanatory factor describing the “Obs” outcome.”*

To examine the coefficient table for (lm.1 model).

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 7.1088 0.2507 28.350 9.21e-06 \*\*\*

TreatmentB -0.5327 0.3071 -1.734 0.158

The overall model p-value (0.1579) is similar to the p‐value quantifying the significance of the “Treatment” factor to explain Obs. This is because the model had only one fixed effect (“Treatment” itself). So, the significance of the overall model is the same as the significance for this coefficient (subject to rounding error).

In the presence of multiple fixed effects, the overall model significance will be different from the significance of each coefficient corresponding to a specific covariate.

Why does he report shows “TreatmentB” not “Treatment”? The estimate of the “(Intercept)” is 7.1088. Let’s look at the mean Obs values within each of the 2 Treatment groups (A and B):

# Model: lm.1 <- lm(Obs ~ Treatment, mydata1)

mean(mydata1[mydata1$Treatment=="A",]$Obs)

> 7.108776

mean(mydata1[mydata1$Treatment=="B",]$Obs)

> 6.57611

# in general, subsetting [[3]](#footnote-3) can be accomplished by:

subset\_data <- mydata1[ which(mydata1$Treatment=='A' & mydata1$Day=='Day2'),]

The mean of the Obs values for {Treatment=A} is the same as the “Intercept” term.

The estimate for “TreatmentB” is negative (-0.5327). Note that:

7.108776 - 0.5327 = 6.57611,

which is the mean of the “TreatmentB” cohort. Thus,

the estimate for “(Intercept)” represents for the “TreatmentA” category, and

the estimate for “TreatmentB” represents the *difference* between the “Treatment” “A” and “B” categories.

Analytically, the linear models represent “linear” associations, as in the plot below:

mydata2 <- data.frame(

Subject = c(13, 14, 15, 16, 17, 18),

Day = c("Day1", "Day1", "Day1", "Day1", "Day1", "Day1"),

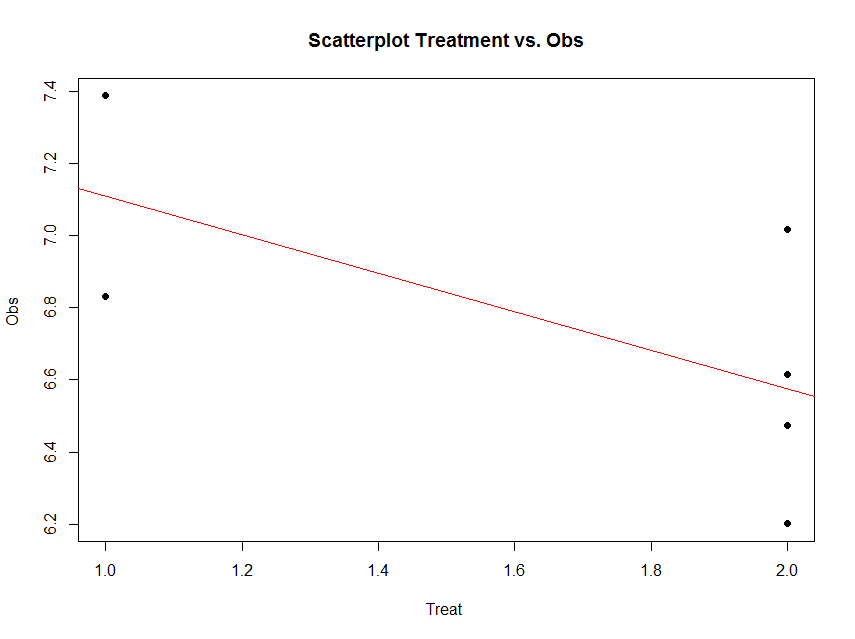
Treatment = c(2, 2, 2, 2, 1, 1),

Obs = c(6.472687, 7.017110, 6.200715, 6.613928, 6.829968, 7.387583)

)

plot(mydata2$Treatment, mydata2$Obs, main="Scatterplot Treatment vs. Obs",   
   xlab="Treatment", ylab="Obs ", pch=19)

# Add fit lines  
abline(lm(mydata2$Obs~mydata2$Treatment), col="red")



The linear model represents the difference between Treatments “A” and “B” as a slope. Going From Treatment “A” to “B” we go down -0.5327 (in terms of the units measuring the “Obs” variable), which is exactly the “TreatmentB” coefficient, relative to Treatment “A”.

Treatment “A” acts as baseline (center of the local coordinate system) and is represented by the “(Intercept)”. The difference between Treatments “A” and “B” is expressed as a slope heading down from 7.108776 by 0.5327. The p-values in the coefficient table correspond to the significance that each coefficient (intercept or Treatment) is “non-trivial”.

In our case, the Intercept is significant (10-5), but the Treatment change (from A to B) is not (0.15). By default, the lm() function takes lexicographical ordering to determine which level (A or B) of the predictor (Treatment) comes first to label Treatment “A” as the intercept and “B” as the slope. Categorical differences like Treatment “A” and “B” can be expressed as slopes because difference between two categories is directly correlated with the slope between the two categories.

The advantage of representing difference between two categories as lines crossing the two categories is that it allows us to use the same computational principles for numeric and categorical variables. That is the same interpretations can be made for Treatment as for continuous (e.g., age) or discrete (e.g., dosage) covariates.

0

5

0

5

0

5

For example, using the MBL Data [[4]](#footnote-4), we can examine player’s Weight as a function of Age. Save the data table as a text file “01a\_data.txt” and load it in RStudio:

# data <- read.table('C:\\Users\\Dinov\\Desktop\\01a\_data.txt',as.is=T, header=T)

# Data: <https://umich.instructure.com/courses/38100/files/folder/data>

data <- read.table('https://umich.instructure.com/courses/38100/files/folder/data/01a\_data.txt',as.is=T, header=T)

attach(data)

# Weight = Age + ε

df.2 = data.frame(Age, Weight)

lm.2 = lm(Weight ~ Age, df.2)

summary(lm.2)

Call:

lm(formula = Weight ~ Age, data = df.2)

Residuals:

Min 1Q Median 3Q Max

-52.479 -14.489 -0.814 13.400 89.915

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 179.6684 4.3418 41.381 < 2e-16 \*\*\*

Age 0.7672 0.1494 5.135 3.37e-07 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20.75 on 1032 degrees of freedom

Multiple R-squared: 0.02492, Adjusted R-squared: 0.02397

F-statistic: 26.37 on 1 and 1032 DF, p-value: 3.368e-07

The significance of the intercept is not very interesting as it just specifies the baseline. However, the p-value in each (predictor) row evaluates whether the corresponding coefficient is significantly non-trivial. Trivial coefficients can be removed from the model. The intercept (179.6684) represents the predicted Weight for a player at Age 0, which doesn’t make sense for Baseball players, but it captures the constant impact of Age on Weight.

The interesting part is the coefficient of “Age” (0.7672), which is a significant “predictor” of Weight (). Thus, for every increase of age by 1 year, The Weight is expected to go up by 0.7 to 0.8 lbs.

The regression line in the scatterplot represents the model-predicted mean Weight-gain with Age. The y-intercept (179.6684), for Age=0, and slope (0.7672) of the line represents the corresponding (Intercept and Age) coefficients of the model output.

plot(Age, Weight, main="Scatterplot Age vs. Weight", xlab="Age", ylab="Weight", pch=19)

abline(lm(Weight ~ Age), col="red")



Interpreting Model Intercepts

The model estimates of the intercept may not always be useful. For instance, if we subtract the mean Age from each player’s Age:

df.2$Age.centered = df.2$Age - mean(df.2$Age)

lm.2a = lm(Weight ~ Age.centered, df.2)

summary(lm.2a)

This creates a new column in the data.frame (df.2) called “Age.centered” representing the Age variable with the mean subtracted from it. This is the resulting coefficient table from running a linear model analysis of this “centered” data:

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 201.7166 0.6452 312.648 < 2e-16 \*\*\*

Age.centered 0.7672 0.1494 5.135 3.37e-07 \*\*\*

Residual standard error: 20.75 on 1032 degrees of freedom

Multiple R-squared: 0.02492, Adjusted R-squared: 0.02397

F-statistic: 26.37 on 1 and 1032 DF, p-value: 3.368e-07

The intercept estimate has changed from 179.6684 (predicted Player Weight at birth, Age=0) to 201.7166 (predicted player Weight at mean Age). However, the slope and its significance (0.7672, 3.37x10-07) are unchanged and neither is the significance of the full model

(3.368x10-07). So, centering the Age does not impact the nature of the linear model, it just changed the intercept that now points to the Weight at the mean Age.

As we have additional information for each MLB player, e.g., “Team”, “Position”, “Height”, “Weight”, “Age”) we can include additional factors into the linear model (lm.2a) to increase its explanatory power (R2 = 0.02492). We can start by adding

Weight ~ Age + Height + ε.

lm.3 = lm(Weight ~ Age.centered + Height, df.2)

summary(lm.3)

Call:

lm(formula = Weight ~ Age.centered + Height, data = df.2)

Residuals:

Min 1Q Median 3Q Max

-50.818 -12.148 -0.344 10.720 74.175

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -164.0141 17.2897 -9.486 < 2e-16 \*\*\*

Age.centered 0.9624 0.1252 7.690 3.43e-14 \*\*\*

Height 4.9626 0.2345 21.163 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 17.33 on 1031 degrees of freedom

Multiple R-squared: **0.3202**, Adjusted R-squared: 0.3189

F-statistic: 242.8 on 2 and 1031 DF, p-value: < 2.2e-16

And them adding additional factors:

Weight ~ Age + Height + Position + Team + ε.

lm.4 = lm(Weight ~ Age.centered + Height+ Position + Team, df.2)

summary(lm.4)

Call:

lm(formula = Weight ~ Age.centered + Height + Position + Team,

data = df.2)

Residuals:

Min 1Q Median 3Q Max

-48.692 -10.909 -0.778 9.858 73.649

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -139.4055 18.8556 -7.393 3.04e-13 \*\*\*

Age.centered 0.8906 0.1259 7.075 2.82e-12 \*\*\*

Height 4.7175 0.2563 18.405 < 2e-16 \*\*\*

PositionDesignated\_Hitter 8.9037 4.4533 1.999 0.045842 \*

PositionFirst\_Baseman 2.4237 3.0058 0.806 0.420236

PositionOutfielder -6.2636 2.2784 -2.749 0.006084 \*\*

PositionRelief\_Pitcher -7.7695 2.1959 -3.538 0.000421 \*\*\*

PositionSecond\_Baseman -13.0843 2.9638 -4.415 1.12e-05 \*\*\*

PositionShortstop -16.9562 3.0406 -5.577 3.16e-08 \*\*\*

PositionStarting\_Pitcher -7.3599 2.2976 -3.203 0.001402 \*\*

PositionThird\_Baseman -4.6035 3.1689 -1.453 0.146613

TeamARZ 7.1881 4.2590 1.688 0.091777 .

TeamATL -1.5631 3.9757 -0.393 0.694278

TeamBAL -5.3128 4.0193 -1.322 0.186533

TeamBOS -0.2838 4.0034 -0.071 0.943492

TeamCHC 0.4026 3.9949 0.101 0.919749

TeamCIN 2.1051 3.9934 0.527 0.598211

TeamCLE -1.3160 4.0356 -0.326 0.744423

TeamCOL -3.7836 4.0287 -0.939 0.347881

TeamCWS 4.2944 4.1022 1.047 0.295413

TeamDET 2.3024 3.9725 0.580 0.562326

TeamFLA 2.6985 4.1336 0.653 0.514028

TeamHOU -0.6808 4.0634 -0.168 0.866976

TeamKC -4.7664 4.0242 -1.184 0.236525

TeamLA 2.8598 4.0817 0.701 0.483686

TeamMIN 2.1269 4.0947 0.519 0.603579

TeamMLW 4.2897 4.0243 1.066 0.286706

TeamNYM -1.9736 3.9493 -0.500 0.617370

TeamNYY 1.7483 4.1234 0.424 0.671655

TeamOAK -0.5464 3.9672 -0.138 0.890474

TeamPHI -6.8486 3.9949 -1.714 0.086778 .

TeamPIT 4.3023 4.0210 1.070 0.284890

TeamSD 2.6133 4.0915 0.639 0.523148

TeamSEA -0.9147 4.0516 -0.226 0.821436

TeamSF 0.8411 4.0520 0.208 0.835593

TeamSTL -1.1341 4.1193 -0.275 0.783132

TeamTB -2.6616 4.0944 -0.650 0.515798

TeamTEX -0.7695 4.0283 -0.191 0.848556

TeamTOR 1.3943 4.0681 0.343 0.731871

TeamWAS -1.7555 4.0038 -0.438 0.661142

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 16.78 on 994 degrees of freedom

Multiple R-squared: 0.3858, Adjusted R-squared: 0.3617

F-statistic: 16.01 on 39 and 994 DF, p-value: < 2.2e-16

Notice the linear model coefficients, the corresponding p-values, and the overall model quality, at the bottom of the output change. Are these changed between the 2 models?

* new model (Weight ~ Age + Height + Position + Team + ε.) vs.
* old model (Weight ~ Age + Height + ε)?

These examples illustrate “multiple linear regression” as a linear modeling technique involving 1 response (dependent) variable expressed as a (affine) function of multiple predictor (independent) variables.

Visualizing the Regression-Model coefficients (effect-sizes):

#library("arm")

# data <- read.table('E:\\Ivo.dir\\Research\\UMichigan\\Education\_Teaching\_Curricula\\2015\_2016\\HS\_853\_Fall\_2015\\Modules\_docx\\data\\01a\_data.txt',as.is=T, header=T)

data <- read.table('01a\_data.txt',as.is=T, header=T)

attach(data)

df.2 = data.frame(Age, Weight, Height, Position, Team)

lm.2 = lm(Weight ~ Age + Height+ Position + Team, df.2)

lm.3 = lm(Weight ~ Age\*Height+ Position\*Team, df.2)

lm.4 = lm(Weight ~ Age\*Team + Position\*Height, df.2)

par (mfrow=c(1,1))

# coefplot(lm.2, xlim=c(-2, 2), intercept=TRUE)

coefplot(lm.2, vertical=FALSE, col.pts="green")

coefplot(lm.3, vertical=FALSE, add=TRUE, col.pts="red", offset=0.2)

coefplot(lm.3a, vertical=FALSE, add=TRUE, col.pts="black", offset=0.4)

coefplot(lm.4, vertical=FALSE, add=TRUE, col.pts="blue", offset=0.6)



### (Fixed-Effect) Linear Model Assumptions

The linear modeling approach (above) is referred to as a *linear*  model, as opposed to another type of model (e.g., non-linear, exponential, etc.) for 2 reasons. First, functional analytic representation of the model involves linear terms of the predictive variables. Second, the model assumptions dictate linear relationships among the covariates and between covariates and response.

**Linearity**: The response variable (Y=Weight, in our case) can be expressed via a linear combination formula of the independent variable (e.g., Height, Age, etc.) If it assumption is invalid, the residual plot may include patterns (e.g., quadratic curve or non-linear trend) indicating another type of model may be appropriate. For instance, the residuals may include a pair of lines when we have dichotomous categorical data. Similarly, the QQ Normal Probability Plot may not be perfectly linear (e.g., S-shaped), which again indicates the distribution of the residuals (observed minus fitted (predicted) values) may not be IID Normal (Gaussian distributed).

This plot shows the age/pitch relationship along with the depiction of the residuals:

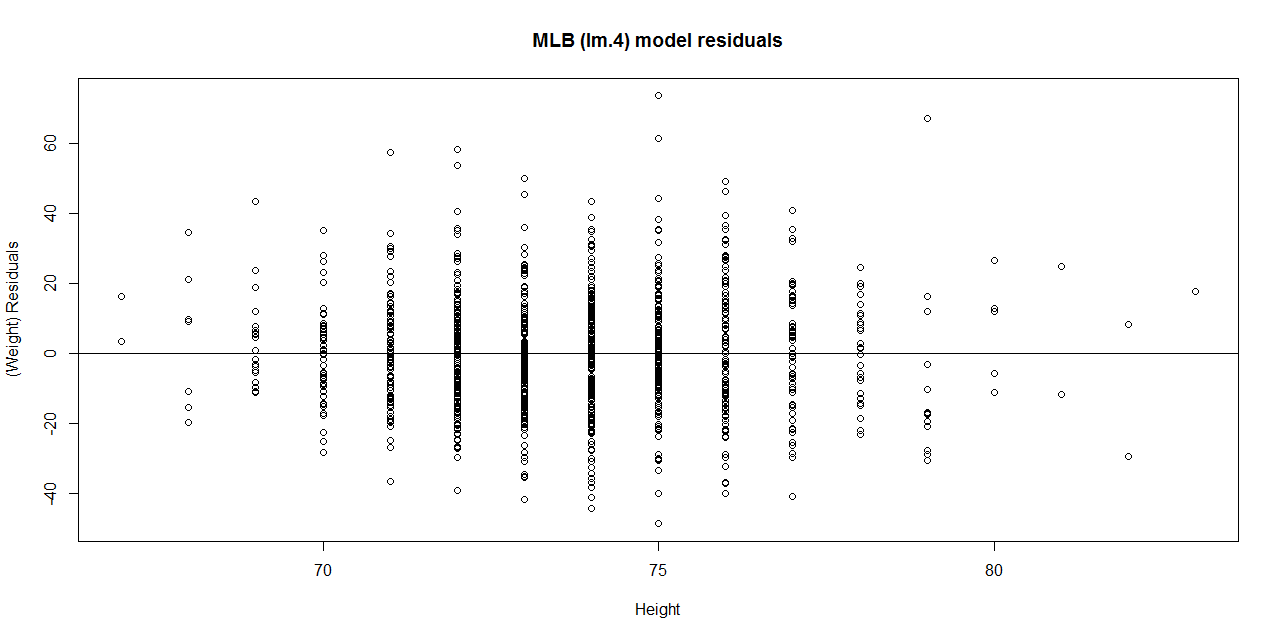
# Weight ~ Age + Height + Position + Team + ε

# lm.4 = lm(Weight ~ Age.centered + Height+ Position + Team, df.2)

lm.4.res = resid(lm.4)

plot(Height, lm.4.res, ylab="(Weight) Residuals", xlab="Height", main="MLB (lm.4) model residuals")

abline(0, 0)



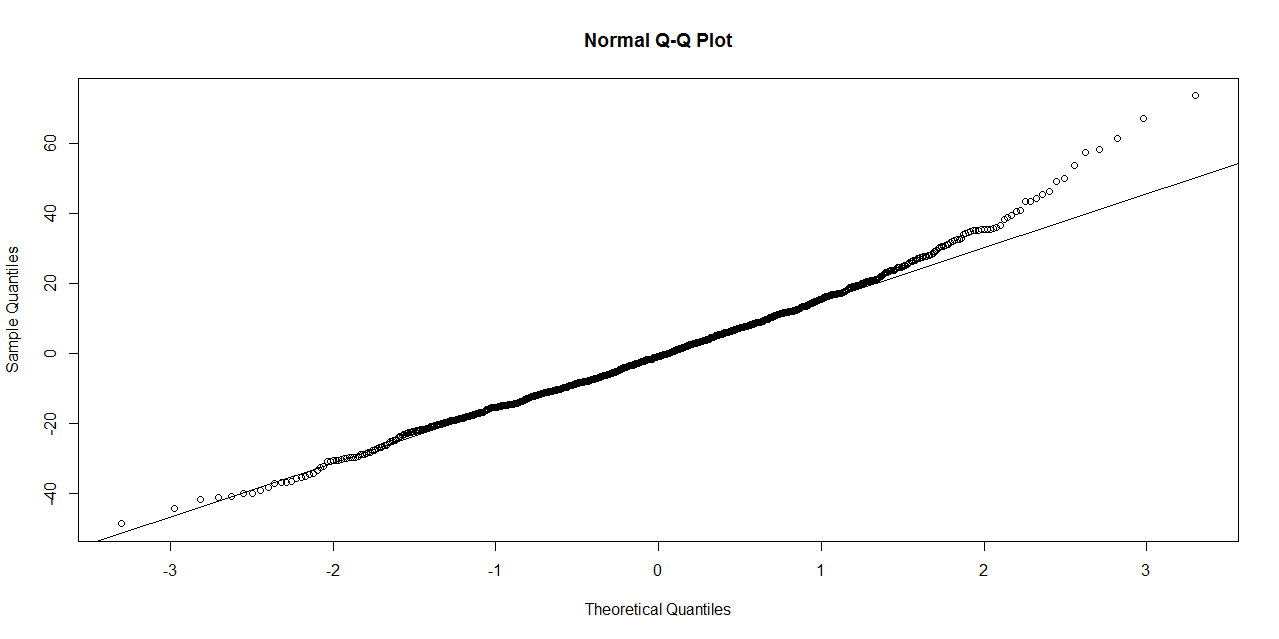
# Weight ~ Age + Height + Position + Team + ε

# lm.4 = lm(Weight ~ Age.centered + Height+ Position + Team, df.2)

# lm.4.res = resid(lm.4)

qqnorm(lm.4.res) # A quantile normal plot - good for checking normality

qqline(lm.4.res)



Mind that now, for illustrative purposes, we change the model to (lm.2: Weight ~ Age)!

# lm.2 = lm(Weight ~ Age, df.2)

# calculate residuals and predicted values

lm.2.res = resid(lm.2)

residuals <- signif(lm.2.res, 5)

predicted <- predict(lm.2)

plot(Age, Weight, main="Visualization of the Residuals of Model: Weight ~ Age", xlab="Age", ylab="Weight", pch=19)

abline(lm(Weight ~ Age), col="red")

# plot distances between points and the regression line

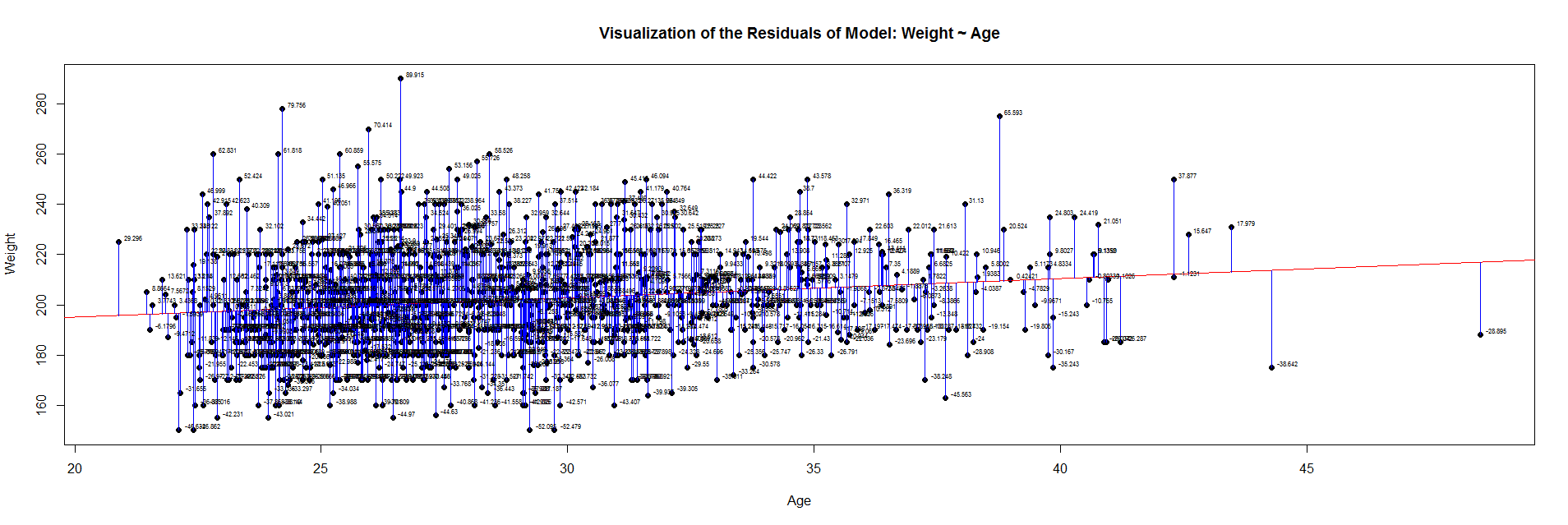
segments(Age, Weight, Age, predicted, col="blue")

# add residual-value labels to (Age) points

# install.packages("calibrate")

# library("calibrate", lib.loc="~/R/win-library/3.1")

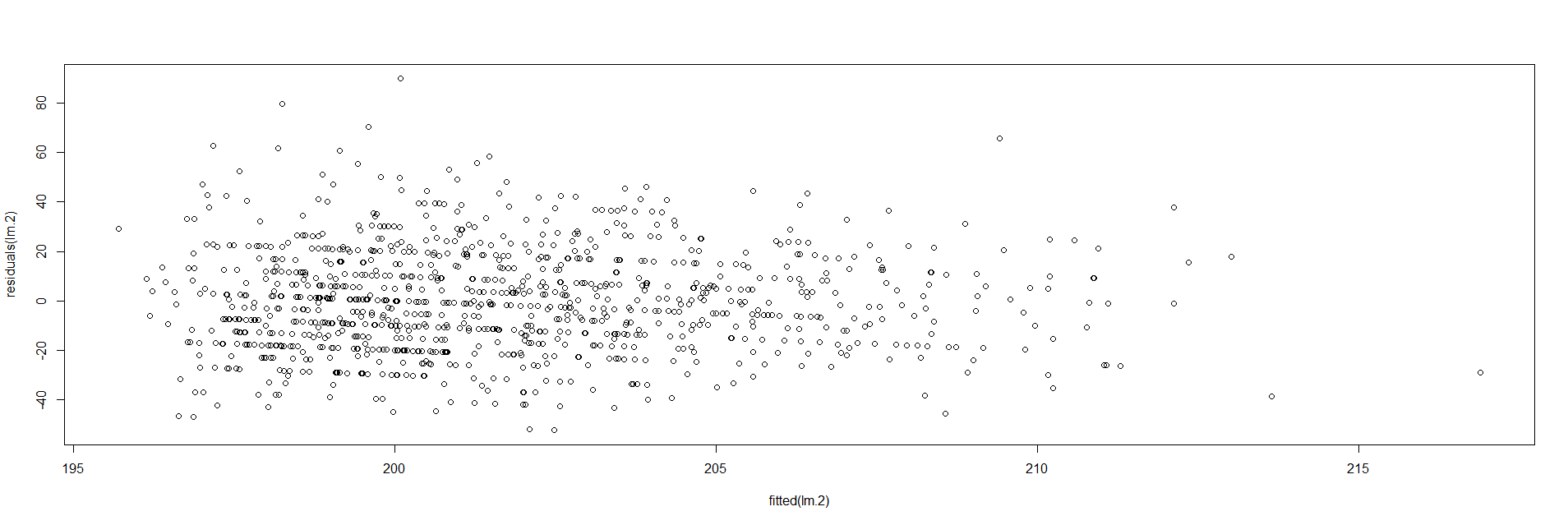
library(calibrate)

textxy(Age, Weight, residuals, cx=0.7)

The **blue** lines indicate the (magnitude of the) residuals and represent the deviations of the observed data points from the model-predicted values (fitted values). The sum of all residuals (positive and negative) is zero, ensured by the least-squares method for estimating the linear model parameters (intercept and slope).

An alternative view is to plot the fitted values (predicted means) on the horizontal line and the residuals, as deviations, on the vertical line.

plot(fitted(lm.2),residuals(lm.2))



Protocols for rectifying residual plots indicating nonlinearity**:**

Model may be excluding important fixed effects that interact with fixed effects already accounted for in the model. If new fixed effects are added, the pattern in the residual plot may disappear.

* Variable transformation – Perform a nonlinear transformation of the response, e.g., log, or reciprocal transform. See this SOCR Activity [[5]](#footnote-5).
* Perform a nonlinear transformation to explanatory variables. For instance, if Age is related to Weight in a U‐shaped way (e.g., quadratic relation), then the model could include Age and Age2 as predictors.
* If residual plots include stripes, then there may be categorical variables playing role, which may necessitate alternative class of models, such as logistic or multinomial models.

Lack of collinearity: A pair of (linearly) correlated predictors are also called collinear. Suppose Age and Height are correlated (which is certainly to be expected!), then including both as predictors of Weight presents a collinearity problem. In the presence of significant variable collinearity, the interpretation of the model becomes challenging. Depending on which correlated predictors are included in the model, the fixed effects may (incorrectly) become significant or insignificant. Significant findings for these correlated or collinear fixed effects is difficult to interpret as there may be trading-off between the “explanatory power” of the pair of covariates. Multiple predictors that are very similar (linearly correlated) inhibit our ability to decide what plays a big role and what has only marginal impact on the response.

Collinearity may be avoided early during the study-design/data-collection phase of the study to collect fewer fixed effects known to not be linearly correlated. Alternatively, we can selectively include/exclude predictors (e.g., only include in model the most meaningful independent variable and drop the others). Dimensionality reduction methods (e.g., Principal/Independent Component Analyses) also identify (fuse) linearly correlated variables into factors (representing linear combinations transforming several correlated variables into one component variable, which can be used as new fixed effect).

Heteroskedasticity (lack of homoscedasticity). If the variance of the data is (approximately) stable across the range of the predicted values, then the process is homoscedastic (equal variance assumption). Otherwise, when the homoscedasticity criterion is violated, the process is called heteroskedastic (unequal variances).

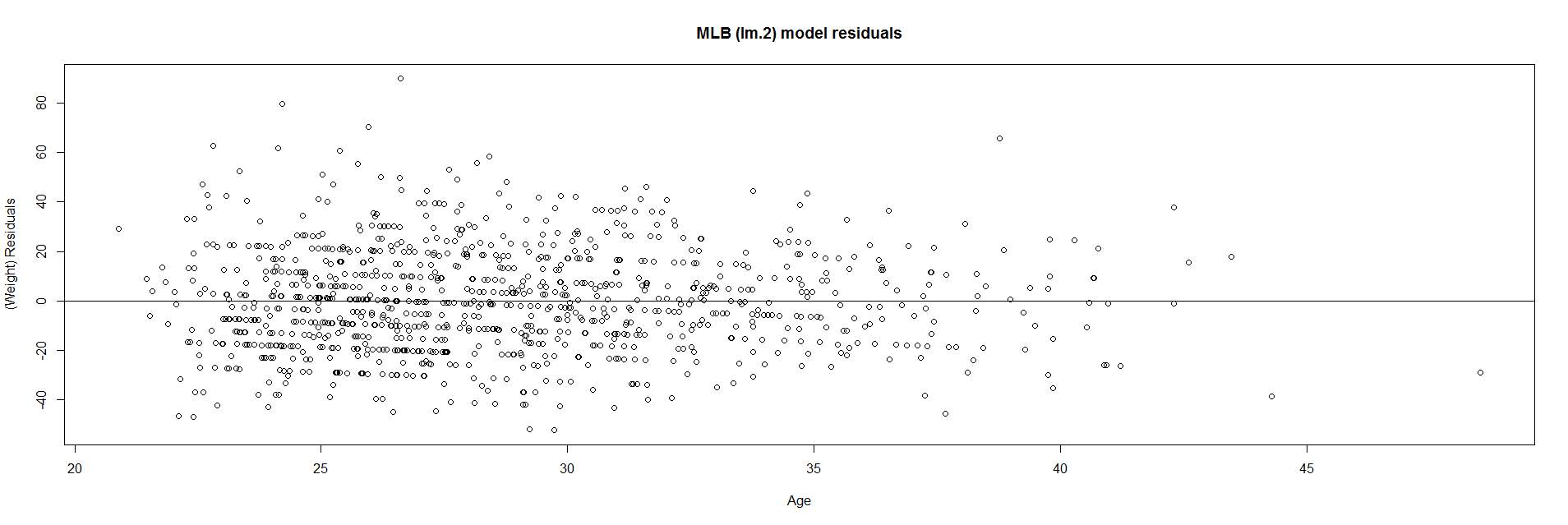
Satisfying the homoscedasticity assumption requires the model residuals to roughly have a similar amount of deviation from the predicted values. This can be checked by examining the residual plot.

# lm.2 = lm(Weight ~ Age, df.2)

lm.2.res = resid(lm.2)

plot(Age, lm.2.res, ylab="(Weight) Residuals", xlab="Age", main="MLB (lm.2) model residuals")

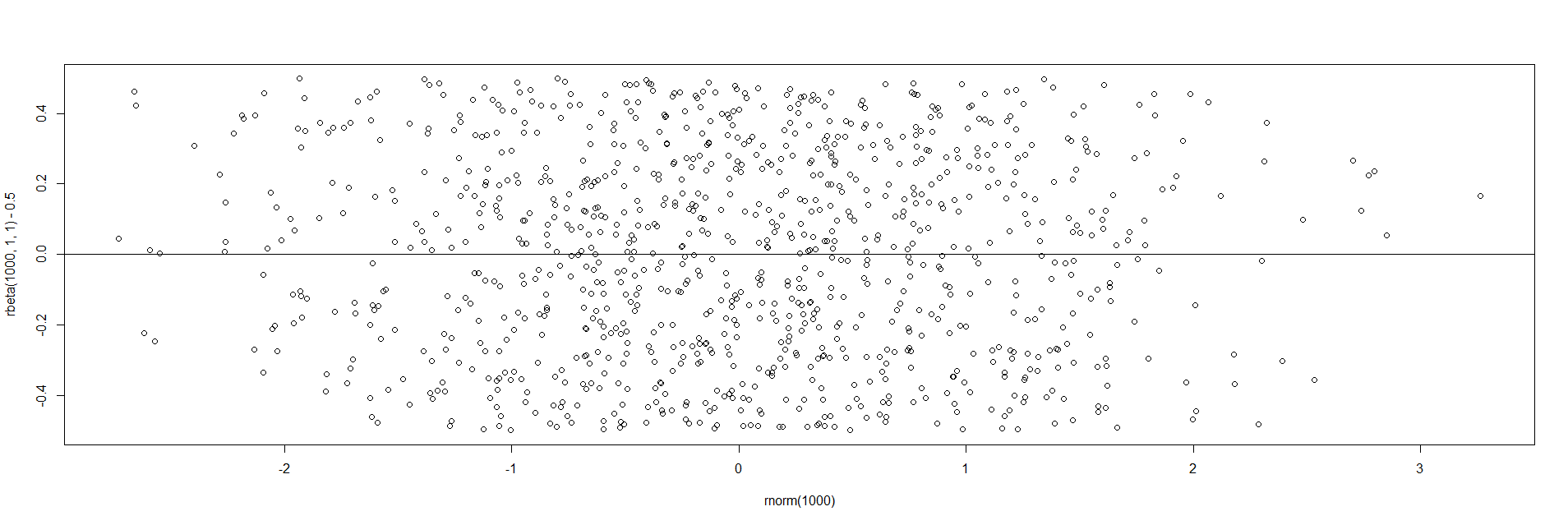
abline(0, 0)



Overall, these data are mostly homoscedastic. A good residual plot essentially looks random blob cluster. For instance, using real (simulated) random data, we can see the expectation for the residuals under the linear model assumptions:

plot(rnorm(1000), rbeta(1000, 1,1)-0.5)

abline(0, 0)



This creates two sets of 1,000 random numbers (x and y axes representing Normal(mean=0,SD=1) and Beta(1,1) distributions, respectively [[6]](#footnote-6)). Redoing this simulation multiple times and recreating the residual plots will \*not\* change the appearance of the scatter!

However there are many situations where the residual plots may show clear heteroskedasticity:

set.seed(11) # see the random generator with some integer

n <- 256 # define sample size

X <- (1:n)/n # The set of the predictor X values (uniform steps of 1/n from 0 to 1)

E <- rnorm(n, sd=1) # A set of \*normally distributed\* random noise E values

i <- order(runif(n, max=dnorm(E))) # Reorder E, by putting larger errors at the end, on average

Y <- 1 + 5 \* X + E[rev(i)] # Simulate new (observed responses) Y values, X plus "error" `E`.

lm.5 <- lm(Y ~ X) # Regress `Y` against `X`.

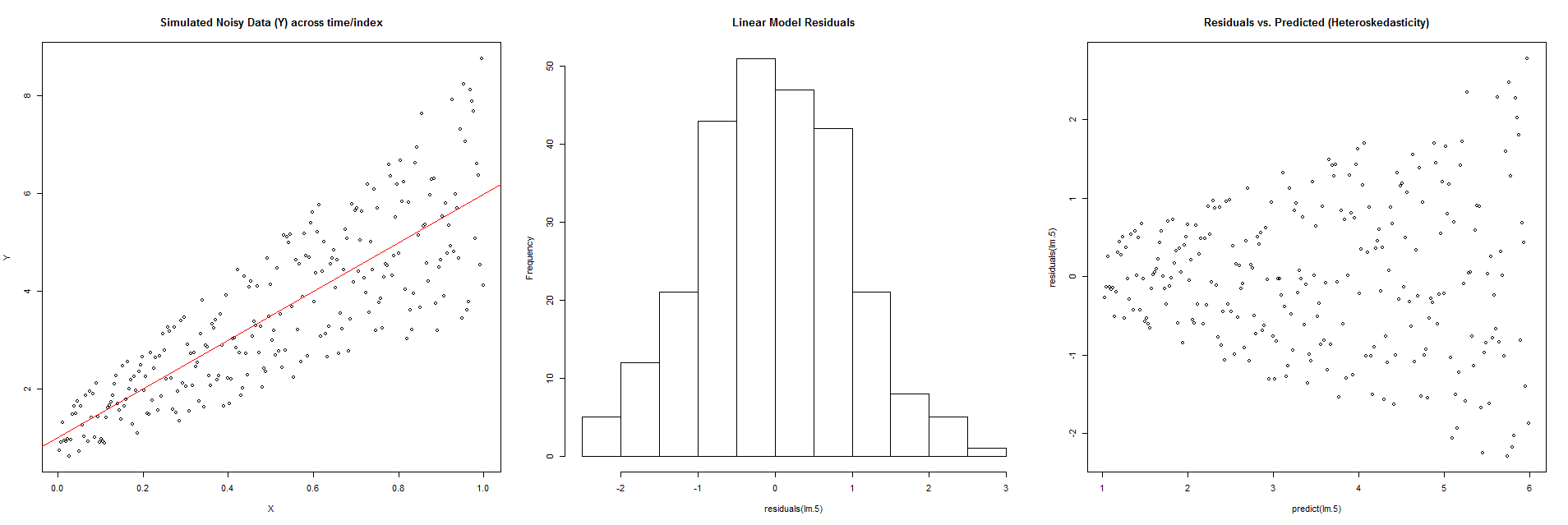
par(mfrow=c(1,3)) # Set up 1 row of 3 plots for drawing the graphs

plot(X,Y, main="Simulated Noisy Data (Y) across time/index", cex=0.8)

abline(coef(lm.5), col="Red")

hist(residuals(lm.5), main="Linear Model Residuals")

plot(predict(lm.5), residuals(lm.5), cex=0.8, main="Residuals vs. Predicted (Heteroskedasticity)")



In this example, larger fitted values correspond with larger residuals (the opposite may be true as well). Transforming the data often helps resolve such Heteroskedasticity.

Normality of residuals. The normality of residuals assumption (parametric assumptions) is also important. Linear models may be robust and gracefully handle certain some violations of the normality assumption.

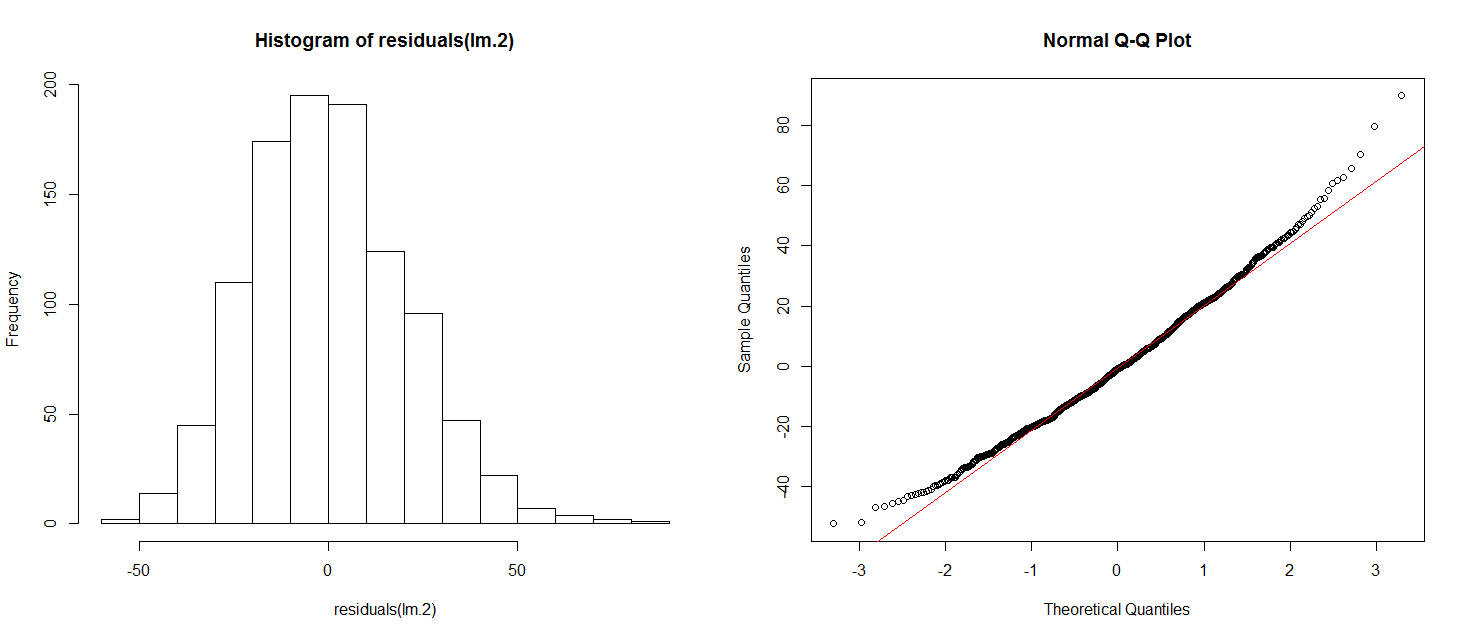
# lm.2 = lm(Weight ~ Age, df.2)

par(mfrow=c(1,2))

hist(residuals(lm.2))

qqnorm(residuals(lm.2))

qqline(lm.2.res, col="red")



The histogram (left) is relatively bell-shaped and the Q-Q normal probability plot (right) indicates the residuals are mostly on a straight line (suggesting errors (observed-predicted) are similar to a normal distribution). Thus, there is no strong evidence suggesting possible violation of the normality assumption.

Outliers (influential data points). Data should not include extreme influential points, otherwise the linear model may be heavily biased. Outliers (influential data points) can drastically change the interpretation of model results and inference, similarly to variable collinearity.

The R function *dfbeta()* computes some of the regression (leave-one-out deletion) diagnostics for linear models and allows us to check for outliers.

# lm.2 = lm(Weight ~ Age, df.2)

df.results <- dfbeta (lm.2)

# lm.2 = lm(Weight ~ Age, df.2)

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 179.6684 4.3418 41.381 < 2e-16 \*\*\*

Age 0.7672 0.1494 5.135 3.37e-07 \*\*\*

head(df.results)

(Intercept) Age

1 -0.1654198377 0.0051723556

2 -0.0690980186 0.0026986682

3 -0.0139731950 0.0007125283

4 -0.0284523604 0.0010963970

5 0.1803531368 -0.0069199855

6 0.0001719468 -0.0008892050

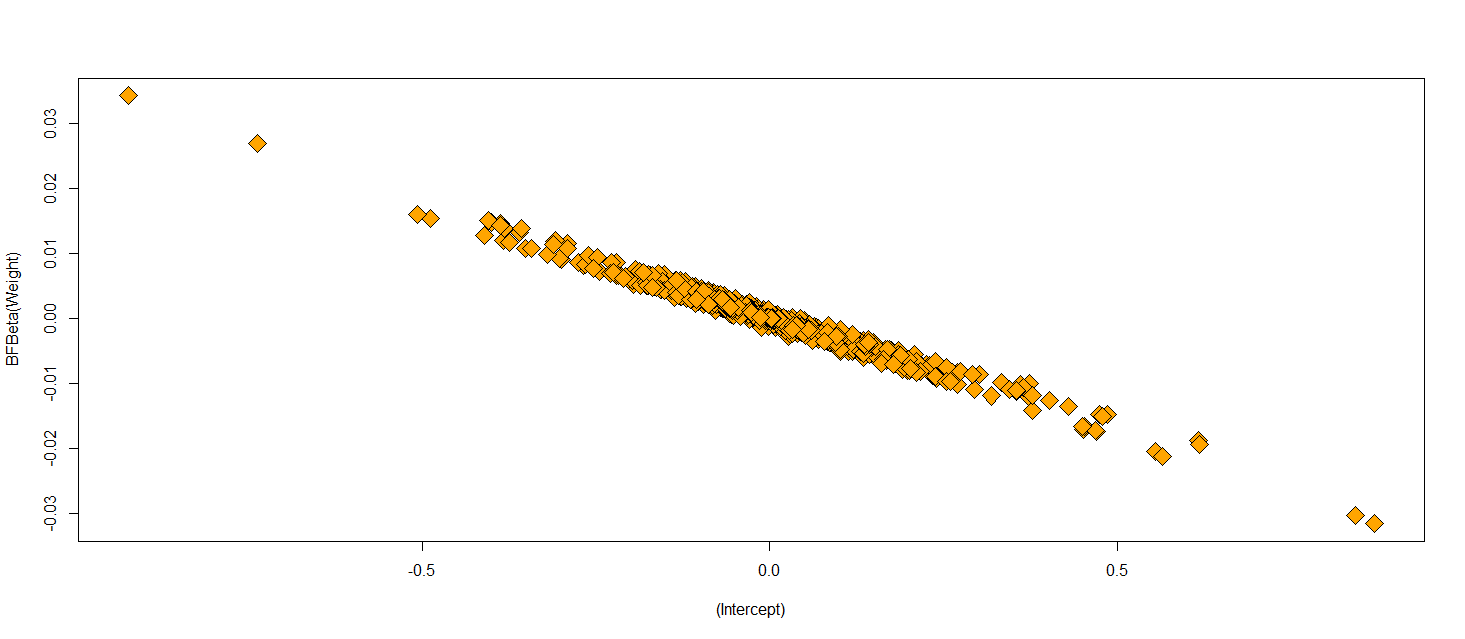
For each coefficient of the lm.2 model (intercept, Age, etc.), dfbeta gives the DFbeta values. The first row means that the coefficient for Age (0.7672) has to be adjusted by 0.0051723556 if data point 1 is excluded. In other words, the Age coefficient of the model without the first data point would be -­‐0.7723724 (=0.7672 + 0.0051723556). If the slope is negative, DFbeta values are subtracted, if the slope is positive, DFbeta values are added.

Interpretation of large or a small DFbeta values is similarly “open” as the interpretation of p-values (case-specific). However, if the DFbeta value changes the sign of the coefficient slope, then this data point is an outlier candidate (influential point) that needs special attention. Excluding that 1 point would change the interpretation of the model results.

For small to medium size () data, |DFbeta| > 1 is generally suspicious (outlier). For larger datasets, a rule of thumb criterion is .

par(mfrow=c(1,1))

plot(df.results, pch=23, bg= 'orange', cex=2, ylab="BFBeta(Weight)")



Exclude outliers and reporting results using the reduced data set is not an optimal solution. Running and reporting both the analyses with/without the influential points may be a better strategy. We can exclude influential points when there evidence that these represent technical errors.

Independence. The independence linear model assumption is the most important limitation (which plays role in most statistical tests). Each data point is supposed to be independent from all the others (e.g., observations come from different subjects).

Violations of the independence assumption make the interpretation of the model results impractical. All assumptions are important, however the independence assumption is critical. Violating independence may inflate the chance of finding spurious effects, bias the results and generate meaningless p‐values.

Independence is mostly a question of the experimental design which is tightly intertwined with the subsequent statistical analyses. When the study design demands collection of more data per subject, repeated measure designs, the data will have (time) dependencies and the appropriate statistical methodologies for interrogating such data involve mixed linear models.

## III. Linear mixed effects analyses

**Questions**:

* What happens if data are not independent and identically distributed (IIDs)?
* How to model multiple observations for the case/subject (across time, conditions, etc.)?

### Fixed and random effects

Linear models express relationships between data elements (variables) in terms of a (linear) function. For example, we can model weight as a function of height [[7]](#footnote-7).

Weight ~ Height + ε.

Here, “height” a fixed effect, and ε was our “error term” representing the deviations between the model predictions and observations (of weight) due to “random” factors that we cannot control experimentally. This tem (ε) is the “probabilistic” or “stochastic” part of the model. Let’s try to unpack “ε” and add complexity to it. In mixed (fixed and random effect) models, everything in the “systematic” part of your model works just like with linear models. If we change the random aspect of our model this leaves the systematic part (height) unchanged.

Suppose we’re looking at the Baseball data [[8]](#footnote-8) and try to identify a relationship that looks like this:

Weight ~ position + ε

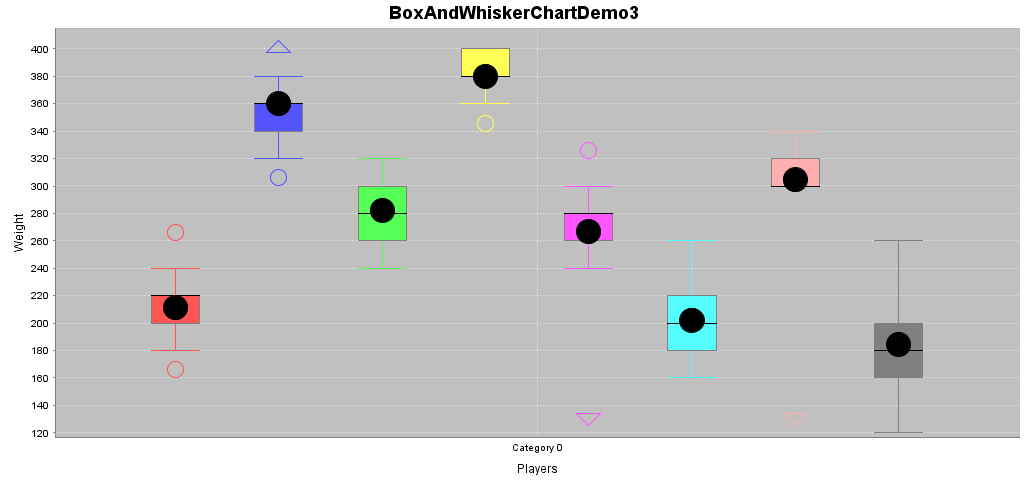
Position (player [field position](http://en.wikipedia.org/wiki/Baseball_field)) is treated as a categorical factor with several levels (e.g., Catcher, First-Baseman, etc.) On top of that, we also have an additional fixed effect, Height, and so our bivariate linear model looks more like this:

Weight ~ Height + position + ε

This model expansion is nice, but it complicates a bit the data analytics and scientific inference. If the study design involved taking multiple measures per player, say across time/age, each player would yield multiple position, height and weight responses. According to the assumptions of the linear model, this would violate the independence assumption as multiple responses from the same subject cannot be regarded as independent from one another. Every player has a slightly different weight, and this is going to be an idiosyncratic factor that affects all responses from the same player, thus rendering these different responses inter-dependent (within player) instead of independent, as required by the model assumptions.

A way to resolve this model assumption violation is to add a **random effect** for players. This allows us to account for inter-independences by assuming a different “**baseline**” weight value for each player. For example, player 1 may have a mean weight 200 pounds across different times, and player 2 may have a mean weight of 350 pounds. Here’s a visual depiction of how this looks like:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TeamA\_Player1 | TeamA\_Player2 | TeamA\_Player3 | TeamA\_Player4 | TeamA\_Player5 | TeamB\_Player1 | TeamB\_Player2 | TeamB\_Player3 |
| 160 | 340 | 240 | 340 | 180 | 200 | 240 | 180 |
| 180 | 340 | 240 | 400 | 240 | 200 | 320 | 120 |
| 220 | 320 | 320 | 400 | 260 | 180 | 340 | 160 |
| 240 | 300 | 300 | 380 | 320 | 160 | 300 | 160 |
| 200 | 380 | 300 | 380 | 280 | 260 | 300 | 260 |
| 220 | 360 | 280 | 360 | 280 | 180 | 300 | 180 |
| 260 | 360 | 320 | 400 | 260 | 240 | 320 | 180 |
| 200 | 360 | 280 | 380 | 300 | 220 | 320 | 200 |
| 220 | 480 | 260 | 380 | 280 | 180 | 300 | 220 |





SOCR Charts generated these plots (<http://www.socr.umich.edu/html/cha/>). The same data may similarly be plotted using R:

data <- read.table('C:\\Users\\Dinov\\Desktop\\01a\_data.txt',as.is=T, header=T)

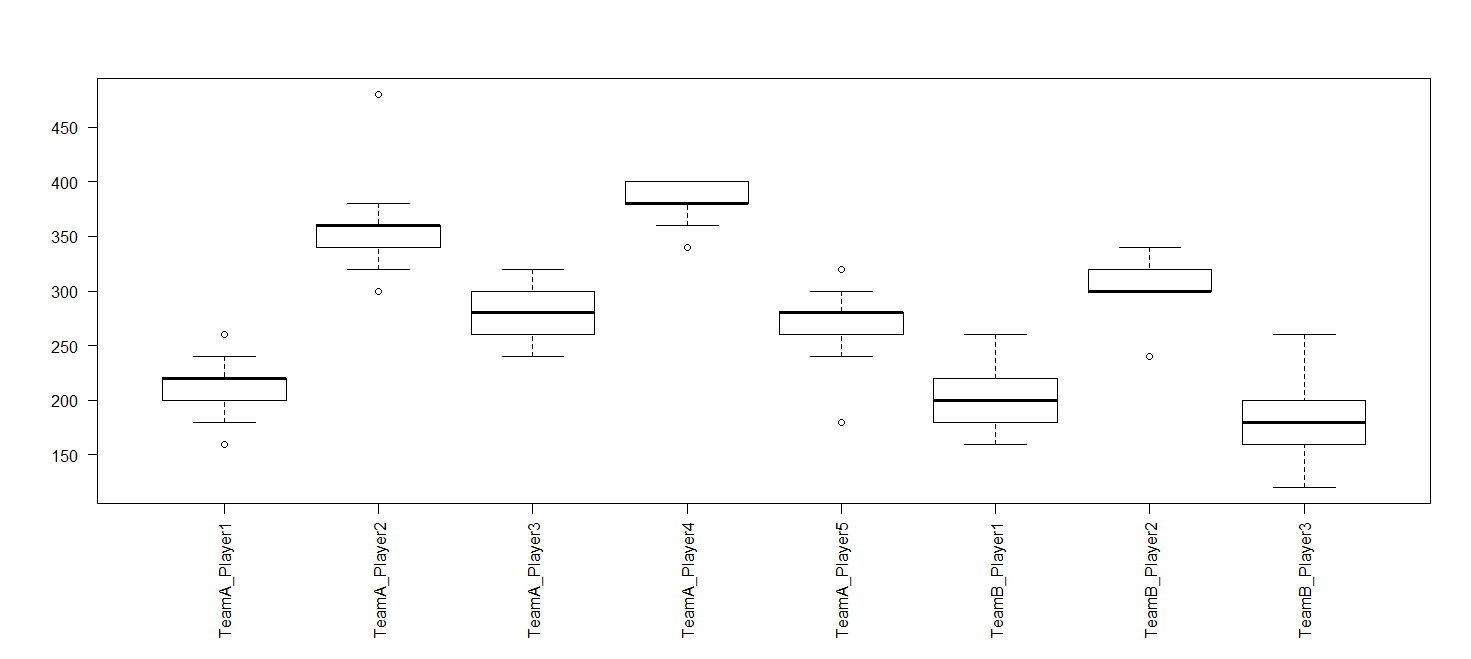
# data <- read.table('https://umich.instructure.com/files/330381/download?download\_frd=1&verifier=HpfmjfMFaMsk7rIpfPx0tmz960oTW7JA8ZonGvVC',as.is=T, header=T)

attach(data)

boxplot(TeamA\_Player1, TeamA\_Player2, TeamA\_Player3, TeamA\_Player4, TeamA\_Player5, TeamB\_Player1, TeamB\_Player2, TeamB\_Player3, col=c("white","lightgray"))

boxplot(data, las = 2)

boxplot(data, las = 2, par(mar = c(8, 5, 4, 2)+ 0.1))



We can model these individual differences by assuming different ***random intercepts*** for each player. In other words, each player will be assigned a different intercept value, and the mixed model estimates these intercept values.

The (fixed effects) linear models include several fixed effects and a general error term “ε”. Linear modeling segregates the world into things that we understand as systematic, i.e., fixed, effects or the explanatory variables, and things that we cannot control for or poorly understand (error, ε). Statistical inference requires that the unsystematic part (ε) of the model, does not have any interesting structure or pattern – it should represent random white noise with common (iid) across-the-board characteristics.

In mixed modeling, we add one or more terms to the fixed effects to account for random effects. These random effects essentially generalize the model (make it more applicable in situations when the error term does have structure) – that is, the random effect models pull out structure from the error term “ε”. For instance, in the baseball weights example, we add a random effect for “player”, and this characterizes idiosyncratic variation that is due to individual bodybuild differences. The intertwining of fixed and random effects is what makes these models mixed-effect models.

Our updated formula looks like this:

M1: Weight ~ Height + position + ε1

M2: Weight ~ Height + position + (1|player) + ε2

“(1|player)” is the R notation for random player effects. It assumes an intercept that’s different for each player”, and “1” stands for the constant-term or intercept. This formula explicitly states that the model should account for (inter-player dependencies) multiple responses per player (say over time) – i.e., account for different baselines for each player. This effectively resolves the conflict with weight-dependences within players that stem from multiple weight observations.

The mixed effect model still contains a general error term “ε”, as even though it accounts for individual by-player variation, there’s still going to be “random” differences between different body-size measurements (e.g., weight) from the same player (noise).

The player position represents an additional source of non-independence that may need to be accounted for. Similarly to the case of by-player variation, we may expect by-position variation. For instance, there might be some special body-size demands for different positions that may lead to overall higher/lower weight. The weight measurements for different players may similarly be affected by this random factor due to game-position-specific idiosyncrasies. Thus, the different responses to one position may not always be independent. There may be some similarities in multiple weight measurements for the same play-position – even if these come from different players. Disregarding these interdependencies, we may violate the linear model independence assumptions. Below is an exemplary visual representation of the by-position variability in weight.

data <- read.table('https://umich.instructure.com/files/330381/download?download\_frd=1&verifier=HpfmjfMFaMsk7rIpfPx0tmz960oTW7JA8ZonGvVC',as.is=T, header=T)

library("reshape2")

# melting by "Position". `melt is from the reshape2 package.

# do ?melt for help

data.m <- melt(data[,-c(1:2)], id.var = "Position")

#

require(ggplot2)

ggplot(data = data.m, aes(x=variable, y=value)) + geom\_boxplot(aes(fill=Position))

ggplot(data = data.m, aes(x=Position, y=value)) +

geom\_boxplot() + facet\_wrap(~variable,ncol = 4)

p <- ggplot(data = data.m, aes(x=variable, y=value))

p <- p + geom\_boxplot(aes(fill = Position))

# to color the points replace group with color=Position

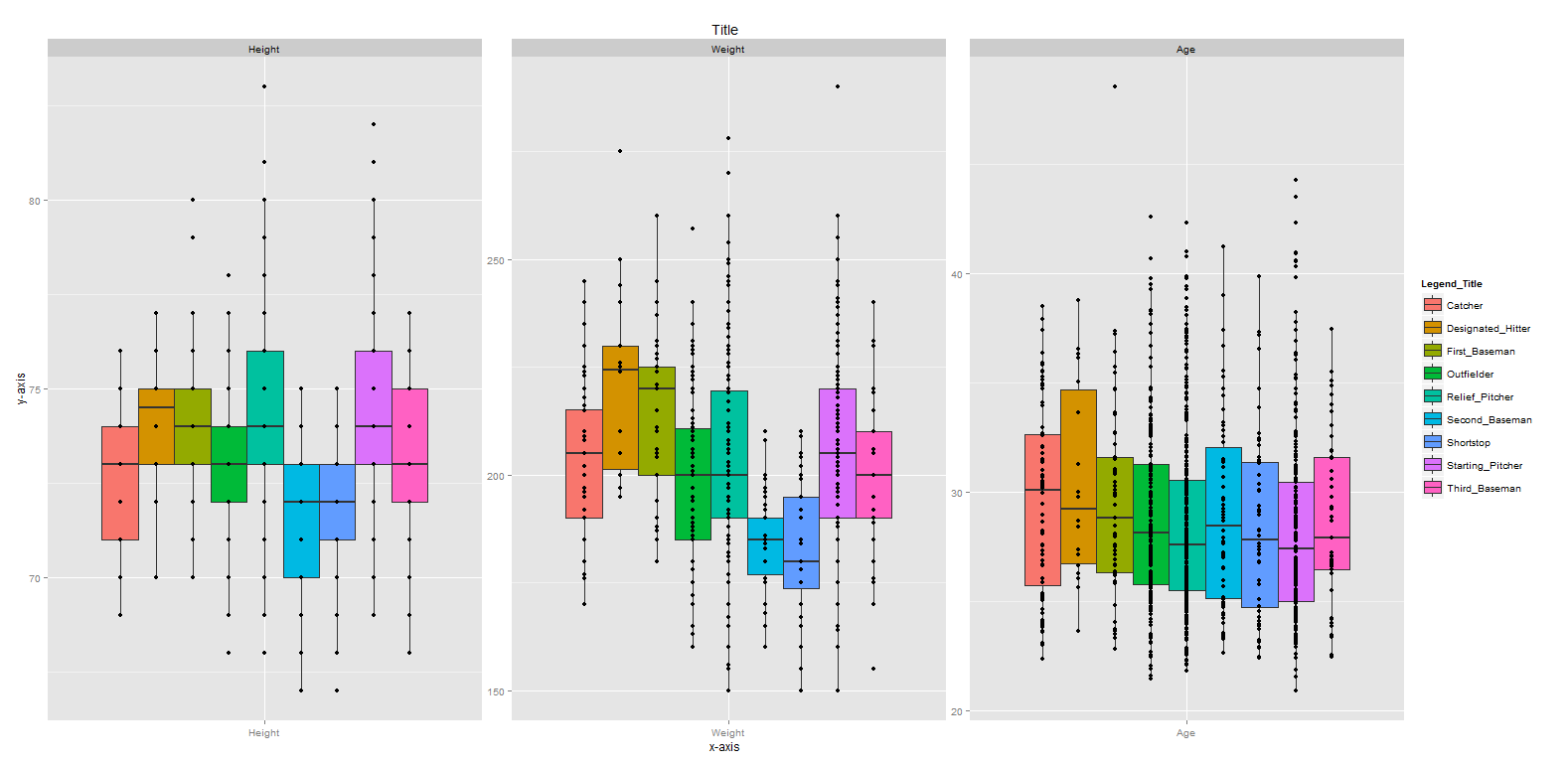
p <- p + geom\_point(aes(y=value, group=Position), position = position\_dodge(width=0.75))

p <- p + facet\_wrap( ~ variable, scales="free")

p <- p + xlab("x-axis") + ylab("y-axis") + ggtitle("Title")

p <- p + guides(fill=guide\_legend(title="Legend\_Title"))

p



The variation between positions may be different from the variation between players!

To add an additional random effect for position we expand the model:

Weight ~ Height + position + (1|player) + (1|position) + ε.

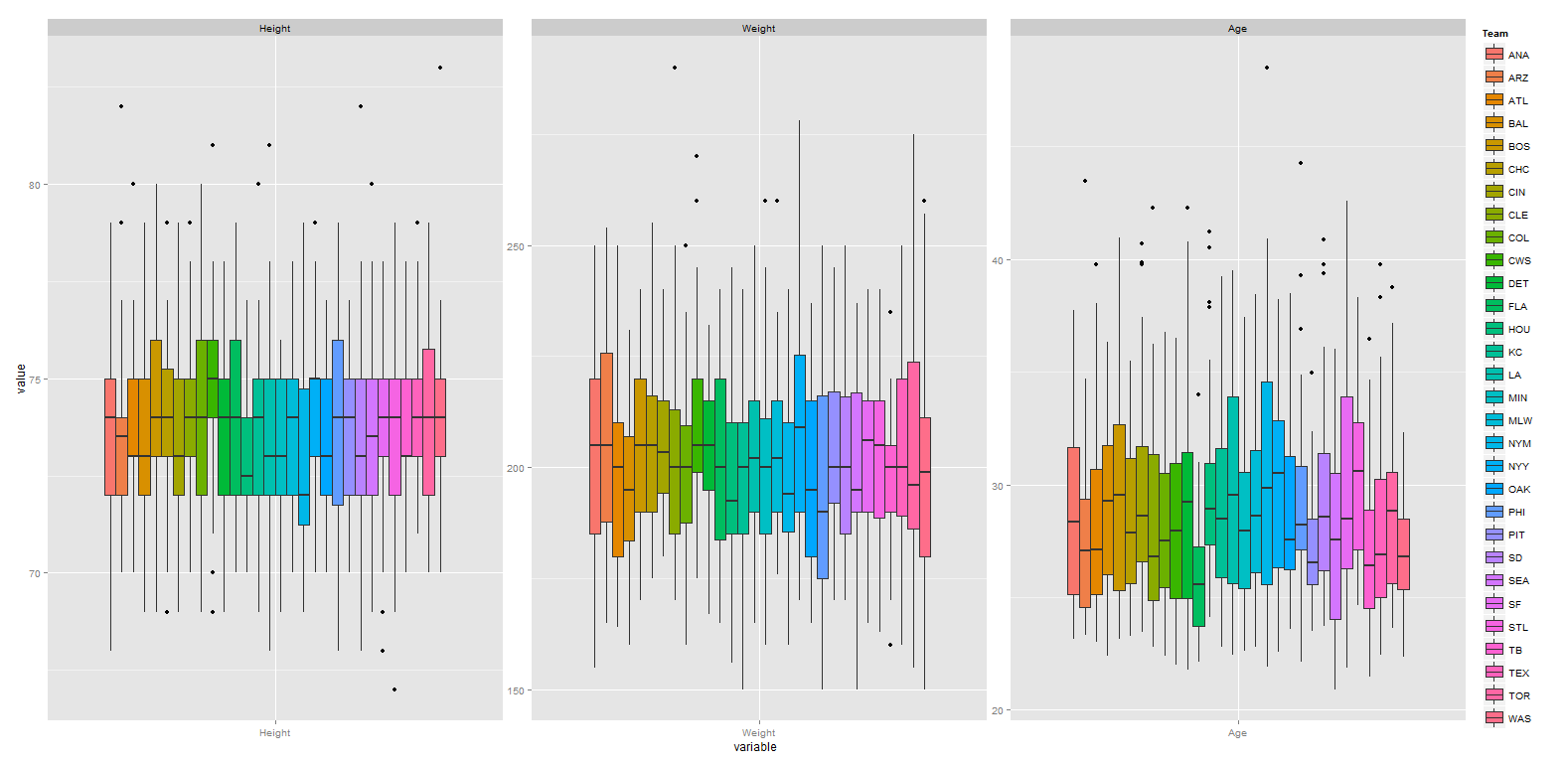
Note that a similar argument may be made for “*Team*”, there could be error-term dependencies or patterns reflecting the between-team random effects on weight – some teams may be bulkier than others. This is really well documented in sports (e.g., top notch Serbian and Spanish water polo teams have a vastly different physiques).

data.m2 <- melt(data[,-c(1,3)], id.var = "Team")

# require(ggplot2)

p <- ggplot(data = data.m2, aes(x=variable, y=value)) + geom\_boxplot(aes(fill=Team))

p + facet\_wrap( ~ variable, scales="free")



Thus, in addition to modeling different intercepts for different players, we may include, as necessary, different intercepts for different positions or teams. This “resolves” dependencies in the linear model, potentially accounting for multiple responses per player, position or team, and we correctly representing the model by-player and by-position variation in overall weight.

Prior to the advent of mixed linear models, researchers used the averages of all multiple records. For example, in nursing researchers may average the vital signs (e.g., heart rates, temperatures, etc.) of their patients over time or condition for a patient-based analysis where each data point is assumed to be derived from one subject, assuring independence. Then researchers may also average over subjects for a diagnostic condition type analysis, where each data point comes from one condition. There are advantages and disadvantages of this averaging approach.

Whereas traditional analyses based on averaging are in principle correct, mixed models provide more flexibility and take the complete data into account. In a patient-based analysis (averaging over conditions/Dx), we basically disregard the by-condition variability. Conversely, in a condition-based analysis, we may disregard by-patient variation. A mixed-effect model would account for both sources of variation in a single analysis.

Let’s apply this understanding of the linear mixed effects modeling via R.

R commands for mixed-effect modeling

(See appendix for complete R script) Open RStudio [[9]](#footnote-9) and install the R package *lme4* [[10]](#footnote-10):

install.packages(“lme4”)

library(lme4)

This makes available the function **lmer()**, which is the mixed model equivalent of the function **lm()** in the fixed-effect model. Load some data in RStudio [[11]](#footnote-11),[[12]](#footnote-12).

data = read.csv(file.choose( ))

attach(data)

You can also try the R QCC package (for quality control, see the first section of this tutorial):

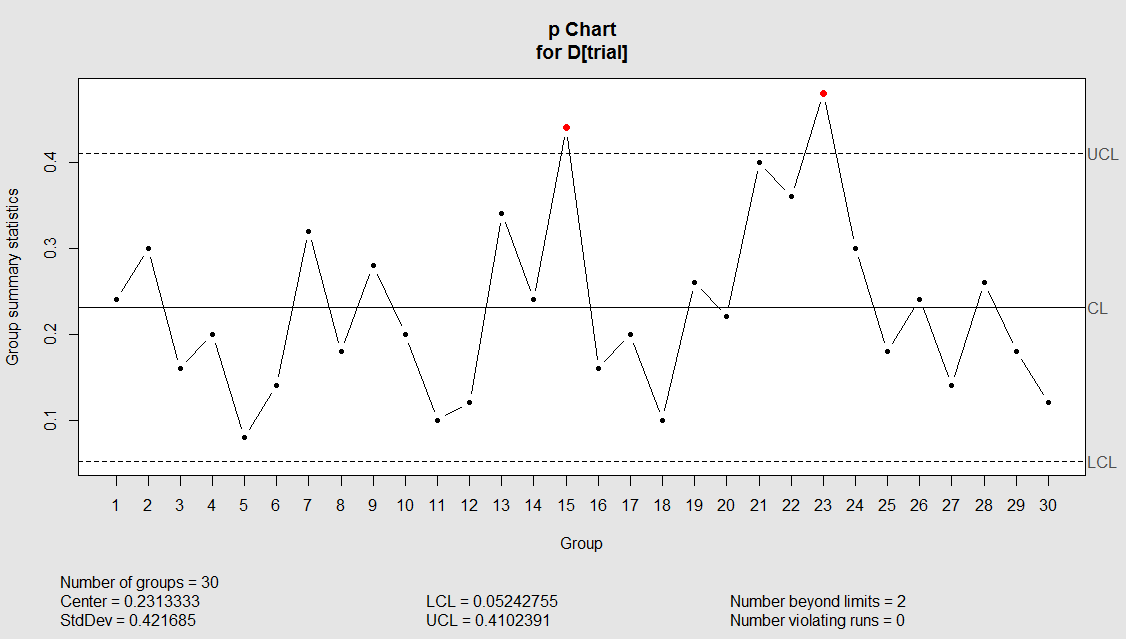
install.packages("qcc")

library("qcc", lib.loc="~/R/win-library/3.1")

data(orangejuice)

attach(orangejuice)

qcc(D[trial], sizes=size[trial], type="p")



Test the data-import and get a summary by using head(), tail(), summary(), str(), colnames(), or whatever commands you commonly use to get an overview of a dataset. Also, it is always good to check for missing values:

which(is.na(Team)==T)

If there are missing values, there are no problems for the mixed-effect modeling.

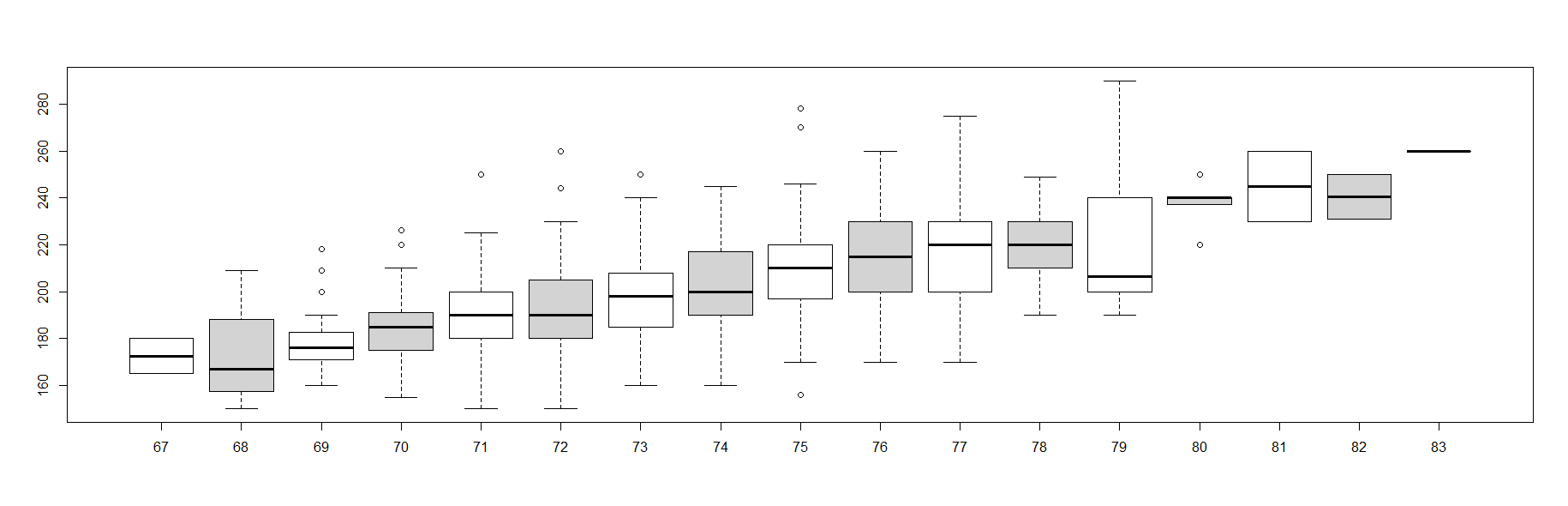
Using the MLB Baseball data [[13]](#footnote-13) we can explore relationship between variables (e.g., Height and Weight) by means of boxplots:

# data <- read.table('E:\\Ivo.dir\\Research\\UMichigan\\Education\_Teaching\_Curricula\\2015\_2016\\HS\_853\_Fall\_2015\\Modules\_docx\\data\\01a\_data.txt',as.is=T, header=T)

data <- read.table('data.txt',as.is=T, header=T)

boxplot(Weight ~ Height, col=c("white","lightgray"))

boxplot(Weight ~ Height\*team, col=c("white","lightgray"))



Is the thick median line in the middle of the boxplot lower for the light than the heavier players?

To construct the mixed model try in the command below …

lmer(Weight ~ Height, data=data)

You will get an error that should look like this:

Error: No random effects terms specified in formula

This is because the model needsat least 1 random effect however we only specified a single fixed effect (Height). So, let’s add random intercepts for subjects and items (remember that items are called “scenarios” here):

model.lmer <- lmer(Weight ~ Height + (1|Team) + (1|Position), data= data)

This command creates a model that includes a fixed effect “Height” to predict Weight, controlling for by-team and by-position variability. The model is saved in an object model.lmer. To see the model type in *model.lmer* – this will print the output in the shell. Note that in the classical fixed effect model, lm(), we need to use summary() to get this output.

This is the full output:

summary(model.lmer)

Linear mixed model fit by REML ['lmerMod']

Formula: Weight ~ Height + (1 | Team) + (1 | Position)

Data: data

REML criterion at convergence: 8841.4

Scaled residuals:

Min 1Q Median 3Q Max

-2.9396 -0.6885 -0.0222 0.6010 4.1358

Random effects:

Groups Name Variance Std.Dev.

Team (Intercept) 1.285 1.133

Position (Intercept) 56.218 7.498

Residual 295.652 17.195

Number of obs: 1034, groups: Team, 30; Position, 9

Fixed effects:

Estimate Std. Error t value

(Intercept) -142.8215 19.0160 -7.511

Height 4.6973 0.2571 18.271

Correlation of Fixed Effects:

(Intr)

Height -0.991

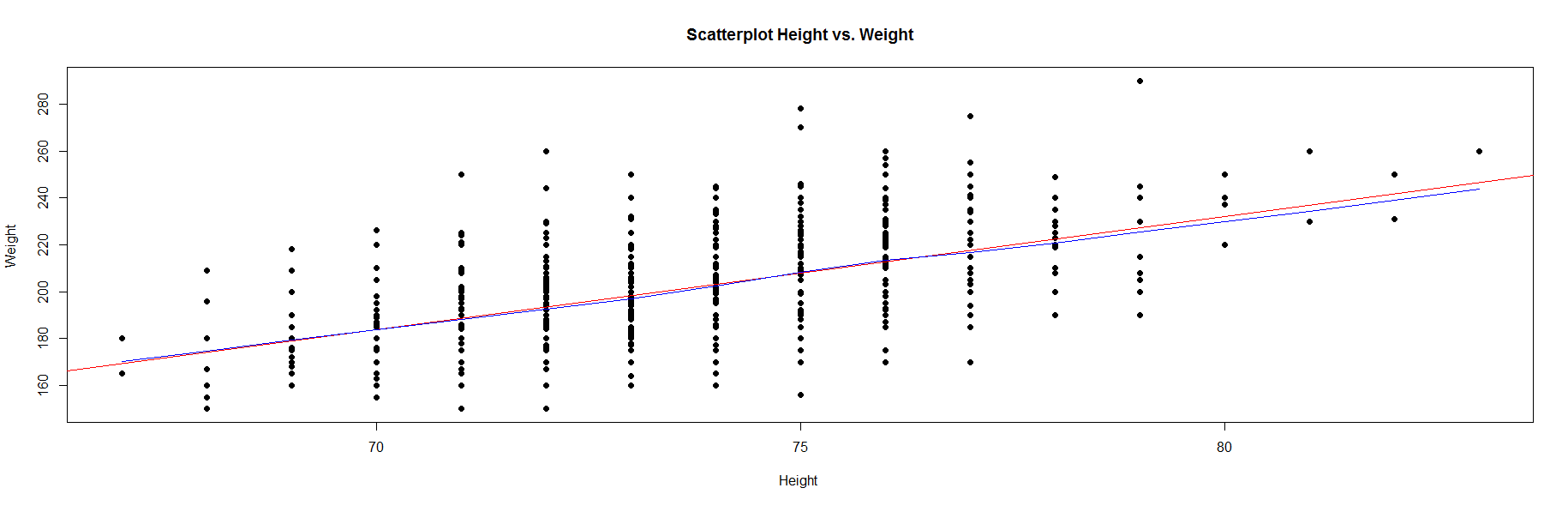
The output starts with the user-specified model and data. Then, there’s the restricted maximum likelihood (ReML) criterion, summary of the model residuals (ε), followed by random and fixed effects.

The *standard deviation column* represents a measure of the variability for each random effect included in the model. “Team” has much less variability than Position. The Residual row represents the variability that’s not due to either Team or Position. This is the ε term – random deviations from the predicted values that are not due to Team or Position. This reflects the fact that body-size has some factors that affect Weight that are outside of the observations we have in this experiment. Similarly, to the fixed effect models, the fixed effects output in this mixed model output includes estimates, SE and T-statistic.

The coefficient of “Height” (4.6973) is the slope for this variable. Finally, the Correlation of the Fixed Effects (H vs. Intercept) is reported to be negative! The output under "*correlation of fixed effects*" has a different interpretation from the intuitive meaning. It is not about the correlation of the variables, but about the “expected correlation of the regression coefficients”. This may, or may not, be related to multicollinearity. In this case, suggest that if we redo the experiment again a decrease of the Height coefficient will increase the intercept, and vice-versa increase of the Height coefficient is expected to will drive down the intercept term (which is currently =-142.8215).

plot(Height, Weight, main="Scatterplot Height vs. Weight", xlab="Height ", ylab="Weight ", pch=19)

# Add fit lines  
abline(lm(Weight~Height), col="red") # regression line (W~H)   
lines(lowess(Height,Weight), col="blue") # lowess line (H,W), (locally weighted scatterplot smoothing)



**Inference on LMER coefficients**: The Satterthwaite approximation, implemented in the **lmerTest** package, overloads the **lmer** function, using exactly the same model, but the **summary**() will include approximate degrees of freedom and p-values for all predictors.

# install.packages("lmerTest")

library(lmerTest)

# re-fit model

model.lmer <- lmer(Weight ~ Height + (1|Team) + (1|Position), data= data, REML = FALSE)

coeffs <- data.frame(coef(summary(model.lmer)))

# get Satterthwaite-approximated degrees of freedom

coeffs$df.Satt <- coef(summary(model.lmer))[, 3]

# get approximate p-values

coeffs$p.Satt <- coef(summary(model.lmer))[, 5]

coeffs

Estimate Std..Error df t.value Pr...t.. df.Satt

(Intercept) -143.26916 18.9698712 977.2059 -7.552458 9.792167e-14 977.2059

Height 4.70294 0.2567663 1030.9565 18.316031 0.000000e+00 1030.9565

p.Satt

(Intercept) 9.792167e-14

Height 0.000000e+00

For categorical variables, like in the fixed effect model, lm(), the mixed effect model, lmer(), takes whatever comes first in the alphabet to be the reference level. “A” comes before “C”, so the slope representing the change from “A” to “C” for a categorical variable (e.g., Team) is in that direction. If the interpretation requires the reference category to be “C”, rather than “A”, then we need to change how we interpret the sign of the slope coefficient. However, interpretation of standard errors and significance would remain the unchanged (as there are sign agnostic).

As the earlier model (model.lmer <- lmer(Weight ~ Height + (1|Team) + (1|Position), data= data)) did not account for age, the intercept may be biased between different age groups. If there are age effects – e.g., the distribution of weights may be bimodal (for younger and older players) – than the average weight may not be representative at all. Think of a study of mammals including 10 bipedal (humans) and 10 quadrupedal (dogs) where the mean number of legs of a mammal would be 3 ((10\*2+10\*4)/20=3), which is not informative and actually incorrect for mammals.

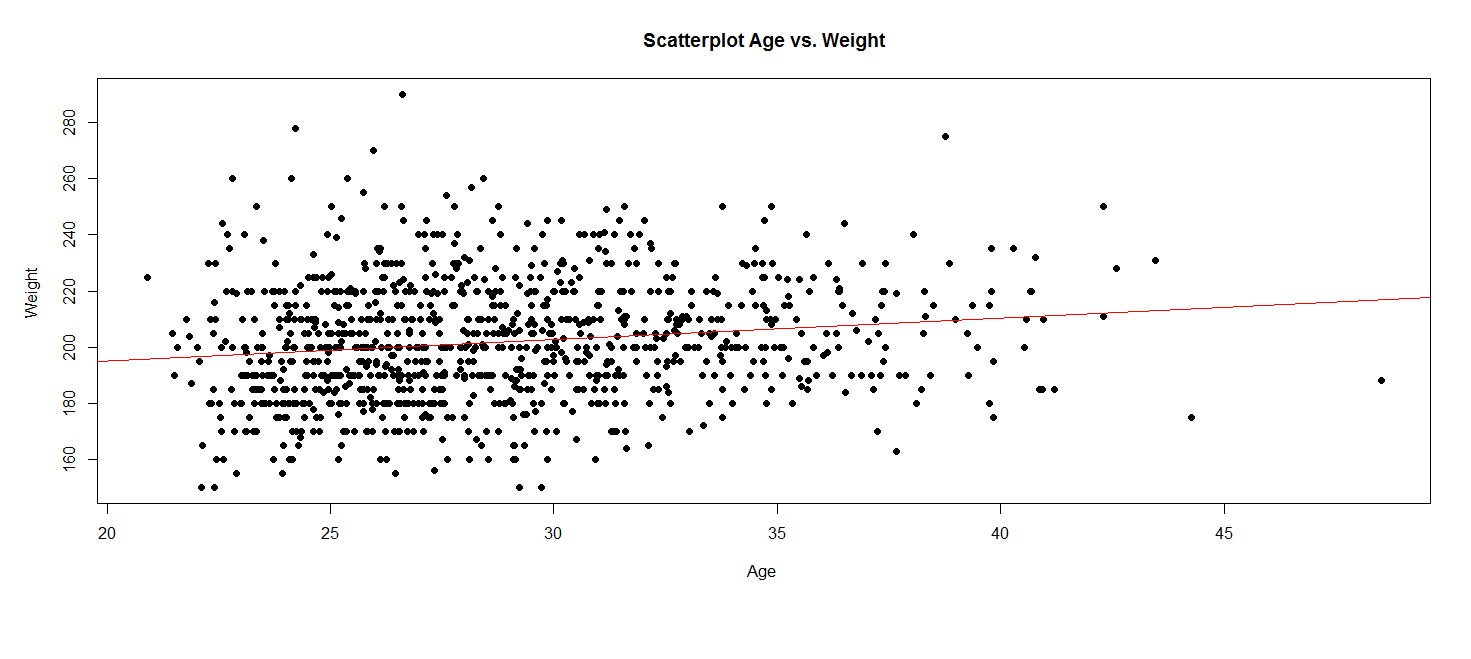
Adding Age as an additional fixed effect:

**lmer.model.2 = lmer(Weight ~ Height + Age + (1|Team) + (1|Position), data=data)**

This expands the original model object **lmer.model** with an age predictor. “Age” is added as a fixed effect because the relationship between Age and Weight is systematic and predictable (not random). Older players are expected to have higher weights.

plot(Age, Weight, main="Scatterplot Age vs. Weight", xlab="Age ", ylab="Weight ", pch=19)

abline(lm(Weight~Age), col="red") # regression line (W~A)



Note that Age is different from the random effects for Team and Position, where the relationship between these and Weight is more unpredictable or stochastic.

Look at the residuals of lmer.model.2 output:

summary(lmer.model.2)

Linear mixed model fit by REML ['lmerMod']

Formula: Weight ~ Height + Age + (1 | Team) + (1 | Position)

Data: data

REML criterion at convergence: 8792.2

Scaled residuals:

Min 1Q Median 3Q Max

-2.9434 -0.6454 -0.0414 0.6022 4.4715

Random effects:

Groups Name Variance Std.Dev.

Team (Intercept) 1.136 1.066

Position (Intercept) 47.935 6.924

Residual 281.805 16.787

Number of obs: 1034, groups: Team, 30; Position, 9

Fixed effects:

Estimate Std. Error t value

(Intercept) -176.4537 19.0913 -9.243

Height 4.8043 0.2512 19.125

Age 0.8888 0.1222 7.273

Correlation of Fixed Effects:

(Intr) Height

Height -0.974

Age -0.240 0.056

Note that compared to our earlier model without the fixed effect for Age, the variation that’s associated with the random effects for “Team” and “Position” are reduced. This is because the variation due to Age may be *confounded* with the variation that’s due to Team/Position. As the initial model didn’t know about Age, its predictions were relatively more varying, producing larger residuals.

Exploring the fixed effect coefficients:

Fixed effects:

Estimate Std. Error t value

(Intercept) -176.4537 19.0913 -9.243

Height 4.8043 0.2512 19.125

Age 0.8888 0.1222 7.273

We see that the impact of Age on Weight is about 0.9, with an intercept of -176 (pounds). The coefficient for the fixed effect of Height changed slightly from 4.6973 to 4.8.

### **Scientific Inference**

Reporting LME models requires explicit quantitative estimation of probability values expressing the statistical significance of these estimates. Interpretation of p-values for mixed models is different from their counterparts in fixed-effects linear models. The Likelihood Ratio Test provides one approach to obtain the p-values. Likelihood is the (conditional) probability of seeing the observed data given a model. The likelihood ratio test compares the (ratio of) two likelihoods corresponding to two alternative models. For instance, suppose we are interested in quantifying the significance of a factor A. The model without the factor A (the null model), will be compared to the model with the factor A.

Suppose we are comparing the following 2 models aiming to assess the significance of the impact of Age on Weight.

m1: Weight ~ Height + Age

m2: Weight ~ Height

A significant difference between “m2” and “m1” would yield that Age matters as a predictor of Weight. Similarly, to estimate the effect of the Height, you would have to do a similar comparison:

m1’: Weight ~ Height + Age

m2’: Weight ~ Age

In both cases, we compared a full model (with the fixed effects in question) against a reduced model without the variable of interest explicitly modeled. A fixed effect would be significant if the ratio between the likelihoods of these two models is significant. In R, we start by constructing the null model:

**lmer.model.0 <- lmer(Weight ~ Height + (1|Team) + (1|Position), data=data, REML=FALSE)**

Note the added the argument REML=FALSE that changes the calculation of the likelihood estimator, which is required when comparing models using the likelihood ratio test. Then, we re-do the full model, also with REML=FALSE:

**lmer.model.1 <- lmer(Weight ~ Height + Age + (1|Team) + (1|Position), data=data, REML=FALSE)**

To compare the two models (the full model with the effect in question with the null model without the effect), we perform the likelihood ratio test using the anova() function:

**anova(lmer.model.0, lmer.model.1)**

The output of this call is:

Data: data

Models:

lmer.model.0: Weight ~ Height + (1 | Team) + (1 | Position)

lmer.model.1: Weight ~ Height + Age + (1 | Team) + (1 | Position)

Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)

lmer.model.0 5 8854.3 8879.0 -4422.1 8844.3

lmer.model.1 6 8804.5 8834.1 -4396.2 8792.5 51.795 1 6.16e-13 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

The report starts with the formulas of the two models, computes a Chi-Square statistics with associated degrees of freedom (2-1), and reports the p-value representing the strength of the data-driven evidence to reject the null-model (and hence to accept the significance of the factor of interest, Age, as impactful on Weight). The qualitative interpretation of these quantitative results may be stated as:

“… Age affects Weight (2(1)= 51.79, p< 10-12), increasing it (annually) by about 0.89 lb ± 0.1222 (standard error) …”

We are using a Chi-Square test because Wilk’s Theorem [[14]](#footnote-14) states that approaches a Chi-Square distribution with degrees of freedom equal to the number of parameters that differ between the models (in this case, only “Age”, so ).

This likelihood-based inference approach is somewhat different from the classical t-tests, ANOVA and linear model inference. Instead of obtaining a p-value directly from the MLE model, we need to run a second-order analysis (in this case ANOVA on the 2 models) to compare them and derive a p-value.

**Note that the “Height” predictor was present in both models (null and full models) as we aimed to assess the impact of Age using the likelihood ratio test. In this test, we think of the covariate “Height” as a control variable and of “Age” as the test variable.**

What happens if we compare the following two models:

m0: Weight ~ 1 # intercept only constant model estimating the mean only

m1: Weight ~ Height + Age # Intercept and 2 covariates model

Suppose the likelihood-ration test suggest significant differences between the 2 models, this would suggest that “m1” (full) and “m0” (null) are significantly different from one another. However, it would not attribute this difference to “Height” or “Age” alone. We may not be able to tease out conclusively which one of the two variables (or both) was crucial.

How about the possibility of having an Age-by-Height interaction effect? For instance, “Height” effect on Weight may be modulated through “Age”. If such inter-dependence between two factors (an interaction) is present or suspected, it’s impact may be assessed the same way:

full model: Weight ~ Height \* Age

reduced model: Weight ~ Height + Age

Interactions between two factors are commonly specified with a “\*” rather than a “+”. Comparing these models via the likelihood ratio test and the anova() function, yields a p-value quantifying the significance of the interaction term (Height \* Age). A significant result implies that Height and Age are strongly inter-dependent on each other. An insignificant results implies that there is no strong evidence in these data for inter-dependence.

**Note**:

lm.1 <- lm(y ~ x:z) # includes only the interaction (colon, “:”) between x & z,

# but not the original variables x and z

lm.2 <- lm(y ~ x + z + x:z) # this is a complete model for y using x & z

#In regressions with a lot of variables, the product (\*) notation provides a shortcut to minimize typing.

# We can indicate this using a simple multiplication symbol (\*):

lm.3 <- lm(y ~ x\*z) # equivalent to model lm.2, but shorter.

Using this code, R will automatically include both variables in the regression in addition to the interaction between the two

We can experiment with computing alternative likelihoods to contrast different models using these, or other, dataset. For instance, compare “Height\*Age” vs. “Height + Age” vs. simply “1” (intercept only model). We need to specify REML=FALSE in the models to be able to use the log-likelihood ratio test.

**Interpreting effect-sizes for random effects** (random slopes vs. random intercepts)

Let’s inspect the coefficients of the mixed model by Team and by Position:

coef(**lmer.model.1**)

$Team

(Intercept) Height Age

ANA -176.9872 4.810485 0.8900941

ARZ -176.2895 4.810485 0.8900941

ATL -177.2084 4.810485 0.8900941

BAL -177.6145 4.810485 0.8900941

BOS -177.0276 4.810485 0.8900941

CHC -176.9496 4.810485 0.8900941

CIN -176.7425 4.810485 0.8900941

CLE -177.1519 4.810485 0.8900941

COL -177.4466 4.810485 0.8900941

CWS -176.4970 4.810485 0.8900941

DET -176.7084 4.810485 0.8900941

FLA -176.7071 4.810485 0.8900941

HOU -177.0758 4.810485 0.8900941

KC -177.5616 4.810485 0.8900941

LA -176.6696 4.810485 0.8900941

MIN -176.7251 4.810485 0.8900941

MLW -176.4952 4.810485 0.8900941

NYM -177.2486 4.810485 0.8900941

NYY -176.7923 4.810485 0.8900941

OAK -177.0576 4.810485 0.8900941

PHI -177.8202 4.810485 0.8900941

PIT -176.4746 4.810485 0.8900941

SD -176.6908 4.810485 0.8900941

SEA -177.1092 4.810485 0.8900941

SF -176.8983 4.810485 0.8900941

STL -177.1223 4.810485 0.8900941

TB -177.2853 4.810485 0.8900941

TEX -177.0813 4.810485 0.8900941

TOR -176.8177 4.810485 0.8900941

WAS -177.2275 4.810485 0.8900941

$Position

(Intercept) Height Age

Catcher -172.2235 4.810485 0.8900941

Designated\_Hitter -166.9709 4.810485 0.8900941

First\_Baseman -169.9896 4.810485 0.8900941

Outfielder -177.8433 4.810485 0.8900941

Relief\_Pitcher -179.5896 4.810485 0.8900941

Second\_Baseman -183.9341 4.810485 0.8900941

Shortstop -186.9628 4.810485 0.8900941

Starting\_Pitcher -179.2067 4.810485 0.8900941

Third\_Baseman -176.1245 4.810485 0.8900941

attr(,"class")

[1] "coef.mer"

As expected from a mixed-effect model, each Team and each Position is assigned a different intercept, as we specified the model with “(1|Team)” and “(1|Position)” to take into account by-Team and by-Position variability:

**lmer.model.1 <- lmer(Weight ~ Height + Age + (1|Team) + (1|Position), data=data, REML=FALSE)**

The fixed effects (**Height + Age**) are all the same for all Team and Positions, as this model is a random intercept model where we account for baseline-differences in Weight assuming whatever the effect of Height is, it’s the same for all Teams and Positions.

This assumption may not always be valid – some Positions may demand more or less Weight. The Weight effect may be different for different Positions. Similarly, the Weight effect may be different for different Teams (recall the Water Polo National teams – Spain vs. Serbia).

Thus, we may need is a *random slope model*, where Team and Positions may have differing intercepts as well as different slopes for the Weight. In R this can be coded as:

**lmer.model.2 = lmer(Weight ~ Height + Age + (1+Height|Team) + (1+Height|Position), data=data, REML=FALSE)**

This new model only includes more complicated random effects. In this case, the notation “(1+Height|Team)” means that the model expects differing baseline-levels of Weight (the intercept, represented by 1) **and** differing levels of “Height”. The same is true for Position.

Examining the coefficients of this updated model:

coef(**lmer.model.2**)

Generates the following output:

$Team

(Intercept) Height Age

ANA -184.5752 4.913632 0.8799488

ARZ -157.6363 4.564806 0.8799488

ATL -185.3436 4.923583 0.8799488

BAL -193.6560 5.031218 0.8799488

BOS -183.8410 4.904126 0.8799488

CHC -181.9705 4.879904 0.8799488

CIN -176.1931 4.805094 0.8799488

CLE -195.5968 5.056349 0.8799488

COL -193.8297 5.033467 0.8799488

CWS -174.9203 4.788613 0.8799488

DET -168.4951 4.705414 0.8799488

FLA -180.6757 4.863139 0.8799488

HOU -191.5622 5.004105 0.8799488

KC -204.9527 5.177497 0.8799488

LA -166.8079 4.683568 0.8799488

MIN -182.1954 4.882816 0.8799488

MLW -164.4339 4.652826 0.8799488

NYM -184.1068 4.907567 0.8799488

NYY -186.9822 4.944800 0.8799488

OAK -184.9647 4.918675 0.8799488

PHI -208.3698 5.221744 0.8799488

PIT -176.5423 4.809616 0.8799488

SD -169.8045 4.722369 0.8799488

SEA -183.9884 4.906034 0.8799488

SF -180.9398 4.866558 0.8799488

STL -183.5529 4.900395 0.8799488

TB -186.5983 4.939829 0.8799488

TEX -182.5549 4.887472 0.8799488

TOR -193.1088 5.024132 0.8799488

WAS -203.4900 5.158557 0.8799488

$Position

(Intercept) Height Age

Catcher -209.6130 5.323756 0.8799488

Designated\_Hitter -236.8530 5.766876 0.8799488

First\_Baseman -219.8348 5.490036 0.8799488

Outfielder -180.9668 4.857761 0.8799488

Relief\_Pitcher -172.6381 4.722276 0.8799488

Second\_Baseman -141.1982 4.210837 0.8799488

Shortstop -127.4526 3.987233 0.8799488

Starting\_Pitcher -173.4682 4.735780 0.8799488

Third\_Baseman -191.4820 5.028815 0.8799488

attr(,"class")

[1] "coef.mer"

**Note that the columns containing the by-team and by-position coefficients for the effect of “Height” is different for each Team and each Position**. The fact that this column values are always positive (quite similar to one another) implies that despite variation between teams/positions there is consistency in how Height impacts Weight. In other words, the Weight of all Players tends to go UP with Height, which is to be expected, albeit for some positions Weight yay go UP slightly more than for other positions (e.g., Designated\_Hitter=5.766876 vs. Shortstop=3.987233). On the other hand, the coefficients for Age remain static across positions and teams – why? Because the model (**lmer.model.2**) didn’t specify random slopes for the by-Position or by-Team effect of Age.

Next, we need to address the significance of these effects (i.e., compute the corresponding p-values using the likelihood ration test). We keep our model from above (**lmer.model.2**) and compare it to a new null model (**lmer.model.0**) using the likelihood ratio test:

**lmer.model.0 = lmer(Weight ~ Age + (1+Height|Team) + (1+Height|Position), data=data, REML=FALSE)**

**lmer.model.2 = lmer(Weight ~ Height + Age + (1+Height|Team) + (1+Height|Position), data=data, REML=FALSE)**

The null model (**lmer.model.0**) needs to have the same random effects structure as the full model (**lmer.model.2**). That is, all random slope parameters included in the full model must be present in the null model.

Let’s now do the likelihood ratio test:

anova(**lmer.model.0**, **lmer.model.2**)

Models:

lmer.model.0: Weight ~ Age + (1 + Height | Team) + (1 + Height | Position)

lmer.model.2: Weight ~ Height + Age + (1 + Height | Team) + (1 + Height | Position)

Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)

lmer.model.0 9 8876.4 8920.8 -4429.2 8858.4

lmer.model.2 10 8809.3 8858.7 -4394.6 8789.3 69.082 1 < 2.2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Model differences are statistically significant. In general, do we need random effects, and if so, which random slopes to include in the model? It’s a bit easier to work with random effect models that only include random intercept. However, it may be best to always include the random slopes in the models, as if these are insignificant, we can determine that and reduce the model. In general, we can expect that the effect of an experimental manipulation (e.g., player position, team, etc.) may not be static across items (e.g., locations). There is some evidence from simulation studies that mixed models excluding random slopes may be too liberal - their Type I error rate may be higher than expected (i.e., we can find significant effects which are actually due to chance alone).

The full model (**lmer.model.2**) is focused on baseball players’ Weight. As Age is not a variable we are interested in studying, albeit it may affect Weight, we need to control for Age. Thus **lmer.model.2** has random slopes for the effect of **Height** (by Team and Position) but not the effect of Age. In other words, we only modeled by-Team and by-Position variability in how Height affects Weight.

**lmer.model.2 = lmer(Weight ~ Height + Age + (1+Height|Team) + (1+Height|Position), data=data, REML=FALSE)**

### **Mixed Effect Model Assumptions**

The same conditions we have in the fixed effect multivariate linear model apply to mixed and random effect models – co-linearity, influential data points, homoscedasticity, lack of normality. These assumptions can be checked by creating residual plots, histogram plots of the residuals or a Q-Q normal probability plots.

The fixed effect independence condition is relaxed in mixed/random effect models as this was the main motivation for mixed models – to resolve dependencies in the data. Mixed effect models still require independence, e.g., when ignoring independent and including just a fixed effect for a variable of interest. For instance, working with a model that does not include a random effect “Player”, then we have multiple Weight responses per Player. This would violate the LME model independence assumption. Careful selection of fixed effects and random effects is necessary to resolve potential dependencies in the data.

The function **dfbeta() can’t be used for assessing influential data points in mixed effects linear models** the way it can for fixed effect models. To check for influential points in mixed effect models the package **influence.ME** [[15]](#footnote-15) or a **leave-one-out validation** can be employed.

For example we can define a vector of size equal to the number of rows in the data. Iterating over each row (i), we estimate a new mixed model excluding the current row index (data[-i,]). The function fixef() extracts the coefficients of interest, which can be adapted to the specific analysis. Running fixef() on the linear model yields the position of the relevant coefficient. For example, position “1” refers to the intercept (which is always the first coefficient mentioned in the coefficient table) and position “2” reflects the effect of “Height” appears second in the list of coefficients.

df <- as.data.frame(data)

all.res=numeric(nrow(df))

for(i in 1:nrow(df))

{ # Generic

# myfullmodel=lmer(response~predictor+ (1+predictor|randomeffect))

# results[i]=fixef(myfullmodel)[parameter position index]

fullmodel=lmer(Weight~Height+ (1+Height|Team), data=data[-i,])

results[i]=fixef(fullmodel)[2]

echo ("Row = ", i)

}

**Comments**

Fixed effects represent explanatory predictors that are expected to have a systematic and predictable influence on the data (response). Whereas random effects represent covariates expected to have a non-systematic, idiosyncratic, unpredictable, or “random” influence on the response variable. Examples of such random effects in experimental studies include “subject/patient/player/unit” and “Age”, as we generally have no control over idiosyncrasies of individual subjects or their age at time of observation.

Often fixed effects are expected to exhaust the population of interest, or the levels of a factor. In the MLB study the factor “Team” may not exhaust the space as there are other teams/leagues. However, for MLB at a fixed time, the “Team” factor may be fully exhaustive. Same with Height. Random effects represent sub-samples from the population of interest and may not “exhaust” the population as more players or teams could be included in the study. The levels of random factor may only represent a small sub-subset of all levels of the factor.

**Hands-on Activity**:

Use these cancer data (<http://www.ats.ucla.edu/stat/data/hdp.csv>), representing cancer phenotypes and predictors (e.g., "IL6", "CRP", "LengthofStay", "Experience") and outcome measures (e.g., remission) collected on patients, nested within doctors (DID) and within hospitals (HID). To fit a mixed model (<http://www.ats.ucla.edu/stat/r/dae/melogit.htm>) and examine *remissions* as cancer outcomes.

This lung cancer dataset includes a variety of outcomes collected on patients, nested within doctors, who are in turn nested within hospitals. Doctor level variables include *experience*.

hdp <- read.csv("http://www.ats.ucla.edu/stat/data/hdp.csv")

hdp <- within(hdp, {

Married <- factor(Married, levels = 0:1, labels = c("no", "yes"))

DID <- factor(DID)

HID <- factor(HID)

})

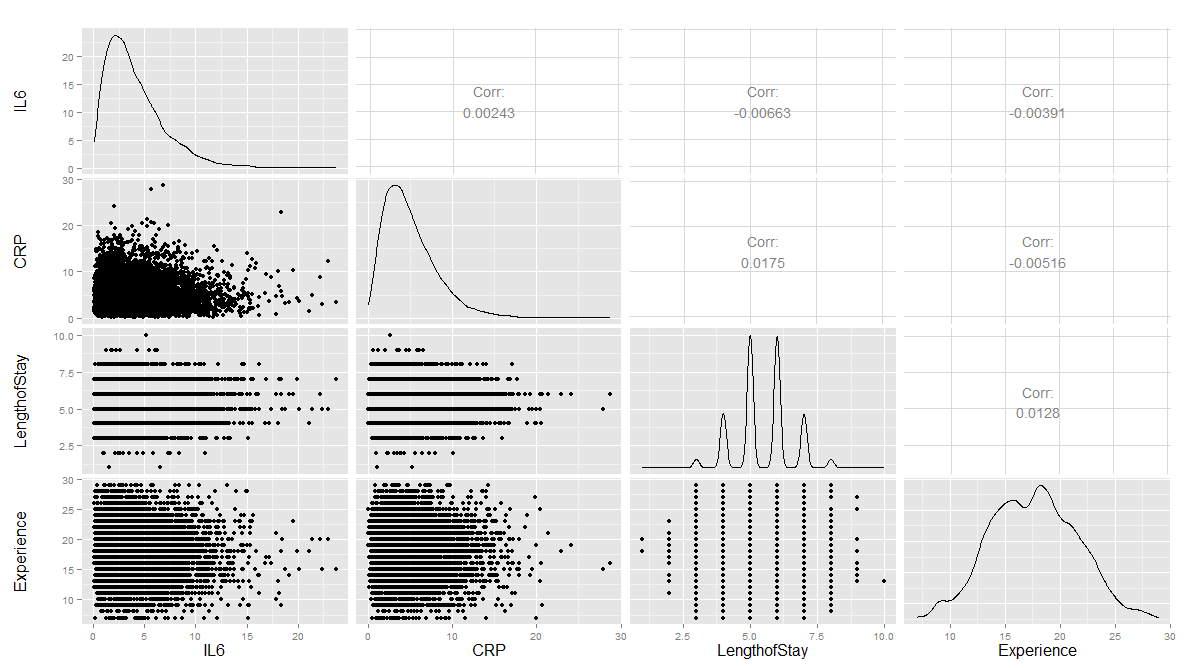
Plot several continuous predictor variables to examine the distributions and catch coding errors (e.g., if values range from 0 to 7, but we see a 999), and explore the relationship among our variables.

# install.packages("ggally")

# library(GGally)

# library("ggplot2")

# ggpairs (hdp[, c("IL6", "CRP", "LengthofStay", "Experience")])



Are there strong linear relations among the continuous variables? Examine CancerStage and LengthofStay closer. The area of bubbles are proportional to the number of observations with the corresponding values.

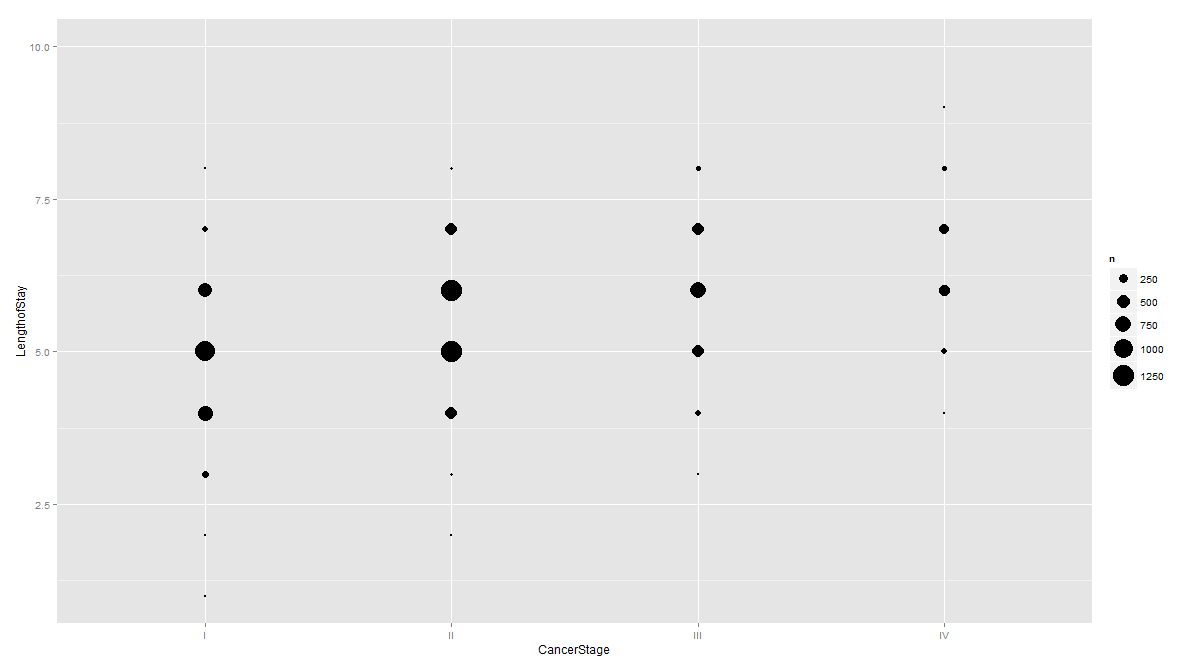
Violin plots may be used for continuous predictors. We can render all raw data separated by CancerStage. To reduce overlaying, we can add some random noise (along the x axis) or alternatively set the alpha opacity level.

Note that IL6 and CRP have skewed distributions indicating that we use a square root scale on the y axes. The distributions appear normal and symmetric with long right tails, even after square root transformation.

ggplot(hdp, aes(x = CancerStage, y = LengthofStay)) +

stat\_sum(aes(size = ..n.., group = 1)) +

scale\_size\_area(max\_size=10)



# install.packages("reshape")

# library(reshape)

tmp <- melt(hdp[, c("CancerStage", "IL6", "CRP")], id.vars="CancerStage")

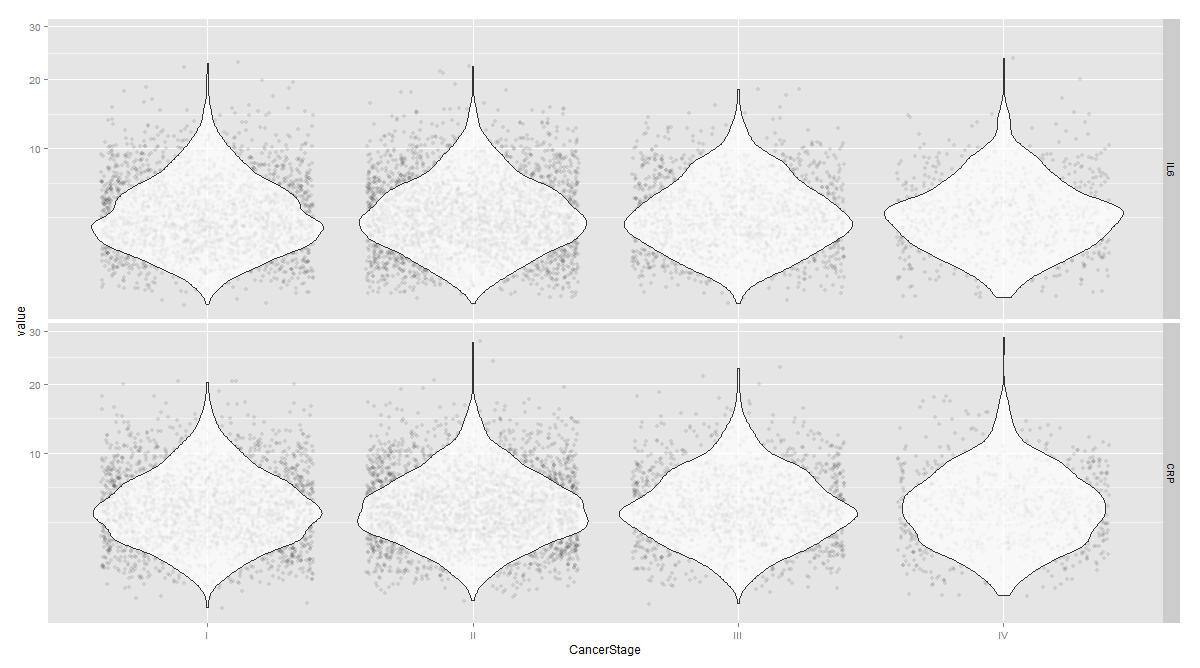
ggplot(tmp, aes(x = CancerStage, y = value)) +

geom\_jitter(alpha = .1) +

geom\_violin(alpha = .75) +

facet\_grid(variable ~ .) +

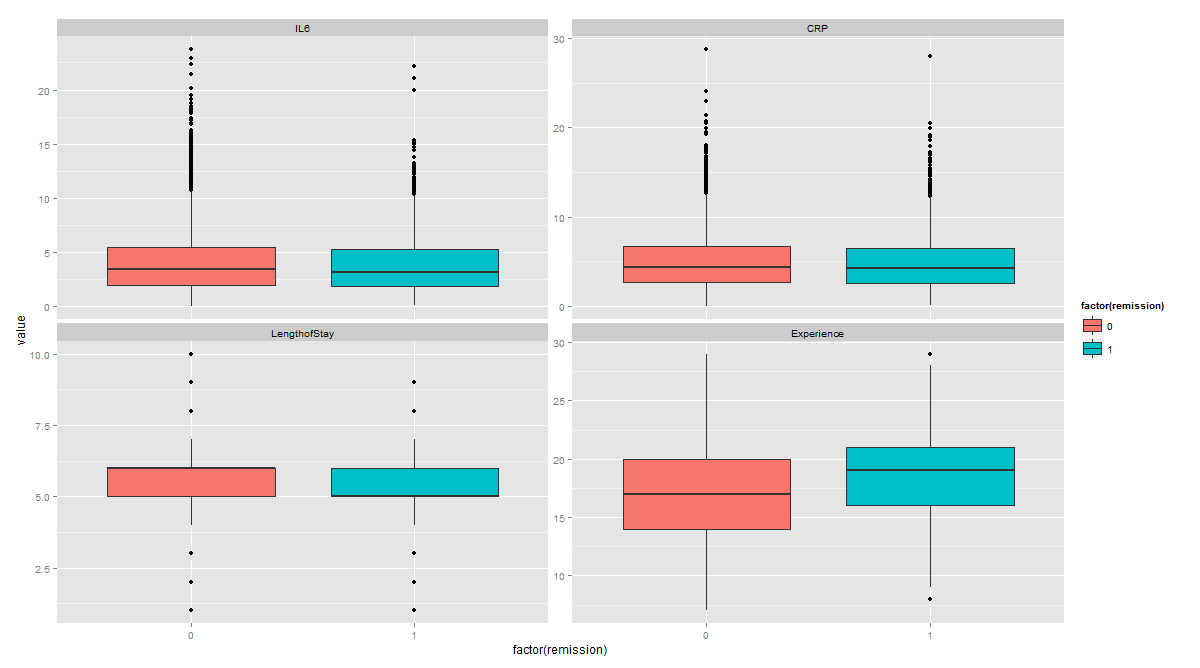
scale\_y\_sqrt()



The distribution of continuous variables at each level of the binary outcome to provide a better depiction of the change of the binary variables over levels of continuous variables.

tmp <- melt(hdp[, c("remission", "IL6", "CRP", "LengthofStay", "Experience")], id.vars="remission")

ggplot(tmp, aes(factor(remission), y = value, fill=factor(remission)))+ geom\_boxplot() + facet\_wrap(~variable, scales="free\_y")



## Types of Data Analyses [[16]](#footnote-16)

* Mixed effects logistic regression, the focus of this page.
* Mixed effects probit regression is very similar to mixed effects logistic regression, but it uses the normal CDF instead of the logistic CDF. Both model binary outcomes and can include fixed and random effects. (Note: This link function, aka Probit link, defined in the 1930’s by biologists studying the dosage-cure rate link, refers to the “probability unit”. It’s kind of the inverse CDF, of the model: if , then Probit link = .
* Fixed effects logistic regression is limited in this case because it may ignore necessary random effects and/or non-independence in the data.
* Fixed effects probit regression is limited in this case because it may ignore necessary random effects and/or non-independence in the data.
* Logistic regression with clustered standard errors. These can adjust for non-independence but does not allow for random effects.
* Probit regression with clustered standard errors. These can adjust for non-independence but does not allow for random effects.

## Mixed effects logistic regression

The glmer model can be used to estimate a mixed effects logistic regression model with Il6, CRP, and LengthofStay as patient level continuous predictors, CancerStage as a patient level categorical predictor (I, II, III, or IV), Experience as a doctor level continuous predictor, and a random intercept by DID, doctor ID.

Estimating and interpreting generalized linear mixed models (GLMMs, of which mixed effects logistic regression is one) can be quite challenging.

# estimate the model and store results in m

# library("lme4")

m1 <- glmer(remission ~ IL6 + CRP + CancerStage + LengthofStay + Experience +

(1 | DID), data = hdp, family = binomial, control = glmerControl(optimizer = "bobyqa"), nAGQ = 10)

# print the mod results without correlations among fixed effects

print(m1, corr = FALSE)

Generalized linear mixed model fit by maximum likelihood

(Adaptive Gauss-Hermite Quadrature, nAGQ = 10) [glmerMod]

Family: binomial ( logit )

# This part conforms the estimates (based on an adaptive Gaussian Hermite approximation of the likelihood) using 10 integrations. More integration points improves the approximation (convergnce to the ML estimates), however, increase the computational requirements

# To avoid a warning of nonconvergence, we specify a different optimizer with the argument control=glmerControl(optimizer="bobyqa"). Although the model will produce nearly identical results without the new argument, we prefer to use models without such warnings.

Formula:

remission ~ IL6 + CRP + CancerStage + LengthofStay + Experience +

(1 | DID)

Data: hdp

AIC BIC logLik deviance df.resid

7397.276 7460.733 -3689.638 7379.276 8516

Random effects:

Groups Name Std.Dev.

DID (Intercept) 2.015

Number of obs: 8525, groups: DID, 407

# This section gives the basic information to compare models, and lists the random effect estimates. This represents the estimated variability in the intercept on the logit scale. When there are other random effects, e.g., random slopes, they are incldued here.

# The total number of observations, and the number of level 2 observations, the total number of patients (8,525) and doctors (407) are reported.

Fixed Effects:

(Intercept) IL6 CRP CancerStageII

-2.05271 -0.05677 -0.02148 -0.41393

CancerStageIII CancerStageIV LengthofStay Experience

-1.00346 -2.33703 -0.12118 0.12009

# The part incldues a table of the fixed effects estimates. The estimates represent the regression coefficients, which are raw/unstandardized on the logit scale.

#The estimates are followed by their standard errors (SEs). As is common in GLMs, the SEs are obtained by inverting the observed information matrix (negative second derivative matrix). However, for GLMMs, this is again an approximation. The approximations of the coefficient estimates likely stabilize faster than do those for the SEs. Thus if you are using fewer integration points, the estimates may be reasonable, but the approximation of the SEs may be less accurate. The Wald tests, *Estimate/SE*, rely on asymptotic theory, here referring to as the highest level unit size converges to infinity, these tests will be normally distributed, and from that, p values (the probability of obtaining the observed estimate or more extreme, given the true estimate is 0).

To obtain confidence intervals (CIs) using the SE estimates.

se <- sqrt(diag(vcov(m1)))

# table of estimates with 95% CI, ***fixef*** = Extract fixed-effects estimates

(tab <- cbind(Est = fixef(m1), LL = fixef(m1) - 1.96 \* se, UL = fixef(m1) + 1.96 \* se))

Est LL UL

(Intercept) -2.05270650 -3.09434022 -1.011072788

IL6 -0.05677184 -0.07934785 -0.034195828

CRP -0.02148295 -0.04151100 -0.001454894

CancerStageII -0.41393353 -0.56243063 -0.265436433

CancerStageIII -1.00346481 -1.19609924 -0.810830385

CancerStageIV -2.33703403 -2.64682910 -2.027238952

LengthofStay -0.12118216 -0.18710346 -0.055260857

Experience 0.12008900 0.06628364 0.173894365

Instead of coefficients on the logit scale, we can report the add odds ratios by exponentiating the estimates and CIs.

exp(tab)

Est LL UL

(Intercept) 0.12838695 0.04530489 0.3638285

IL6 0.94480962 0.92371856 0.9663822

CRP 0.97874617 0.95933878 0.9985462

CancerStageII 0.66104489 0.56982235 0.7668712

CancerStageIII 0.36660701 0.30237139 0.4444888

CancerStageIV 0.09661377 0.07087560 0.1316986

LengthofStay 0.88587258 0.82935793 0.9462383

Experience 1.12759721 1.06852976 1.1899299

r1 <- ranef(m1) # Extract the modes of the random effects

r1.order <- r1[order(r1$DID),]

# install.packages("lattice")

dotplot(ranef(m1,condVar=TRUE), lattice.options=list(layout=c(1,2)))

# Inference

VarCorr(m1); anova(m1); intervals(m1)

**Example of Logistic Regression:**

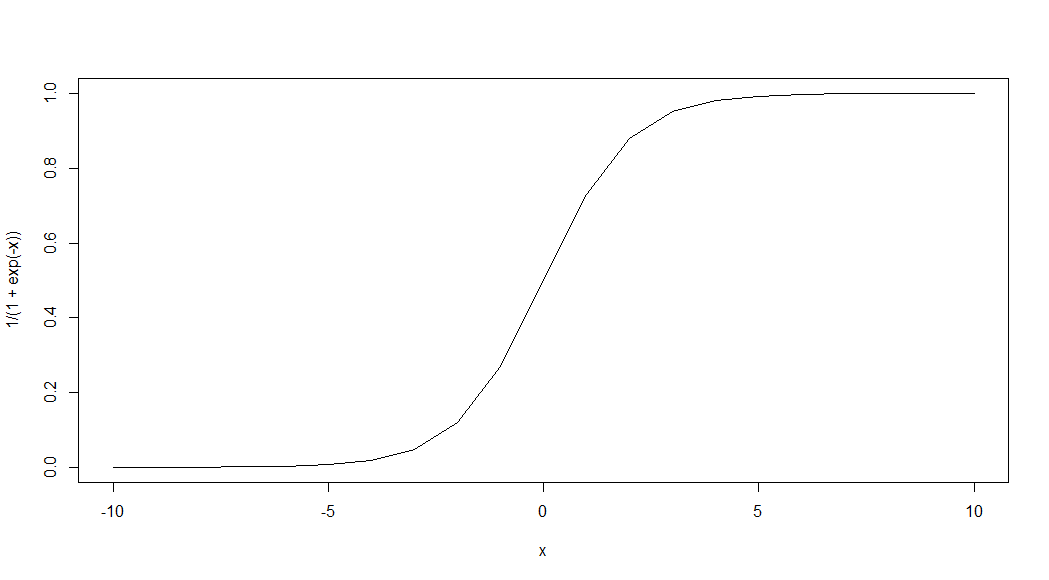
1. Logistic curve:

,

where and represent *probability* and *quantitative-predictor* values, respectively.

x <- seq(-10,10,1)

plot(x,1/(1+exp(-x)), type="l")



The point of this logistic curve is that:

,

which is the **log-odds [[17]](#footnote-17)**  (when y is the probability of an event of interest)!!!

1. Logistic regression equation model to estimate the probability of specific outcomes:

,

where the coefficients (intercept) and , , are estimated using GLM according to a maximum likelihood approach. Using this model allows us to estimate the probability of the dependent (outcome) variable (CO), i.e., surviving surgery, given the observed values of the predictors , .

Probability of surviving a heart transplant based on surgeon’s experience. A group of 20 patients undergo heart transplantation with different surgeons having experience in the range {0(least), 2…, 10(most)}, representing 100’s of operating/surgery hours. How does the surgeon’s experience affect the probability of the patient survival?

The data is shown below and represents each patient and the outcome of the surgery (1=survival) or (0=death).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Surgeon’s  Experience | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 3.5 | 4 | 4.5 | 5 | 5.5 | 6 | 6.5 | 7 | 8 | 8.5 | 9 | 9.5 | 10 | 10 |
| Clinical  Outcome | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

mydata <- read.csv("https://umich.instructure.com/files/405273/download?download\_frd=1&verifier=AOny2eq3wF7WqWAV5YsT6e9zakRTEbZpcuNYRdtM") # 01\_HeartSurgerySurvivalData.csv

# estimates a logistic regression model for the clinical outcome (CO), survival, using the **glm**

# (generalized linear model) function.

# convert Surgeon’s Experience (SE) to a factor to indicate it should be treated as a categorical variable.

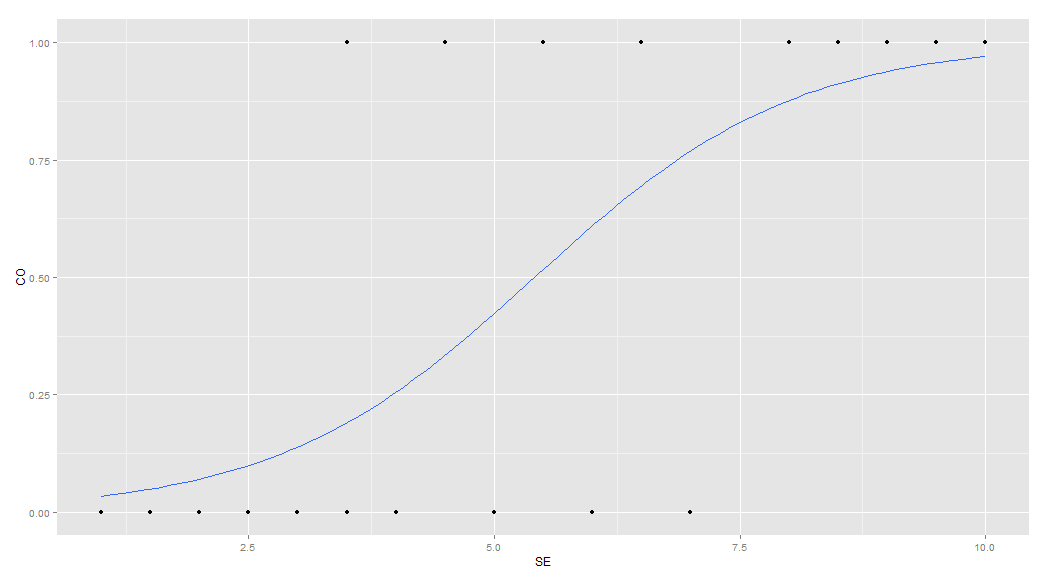
# mydata$rank <- factor(mydata$SE)

mylogit <- glm(CO ~ SE, data = mydata, family = "binomial")

# library(ggplot2)

ggplot(mydata, aes(x=SE, y=CO)) + geom\_point() +

stat\_smooth(method="glm", family="binomial", se=FALSE)



Graph of a logistic regression curve showing probability of surviving the surgery versus surgeon’s experience.

The graph shows the probability of the clinical outcome, survival, (Y-axis) versus the surgeon’s experience (X-axis), with the logistic regression curve fitted to the data.

mylogit <- glm(CO ~ SE, data = mydata, family = "binomial")

summary(mylogit)

The logistic regression analysis gives the following output.

Estimate Std. Error z value Pr(>|z|) Wald

(Intercept) -4.1030 1.7629 -2.327 0.0199 \*

SE 0.7583 0.3139 2.416 **0.0157** \*

The output indicates that surgeon’s experience (SE) is significantly associated with the probability of surviving the surgery (0.0157, Wald test). The output also provides the **coefficients** for:

and .

These coefficients can then be used in the logistic regression equation model to estimate the probability of surviving the heart surgery:

* Probability of surviving heart surgery
* For example, for a patient who is operated by a surgeon with 200 hours of operating experience (), we plug in the value 2 in the equation to get an estimated probability of survival, **p=0.07**:

**SE=2**

CO =1/(1+exp(-(-4.1030+0.7583\*SE)))

> CO

[1] **0.07001884**

* Similarly, a patient undergoing heart surgery with a doctor that has 400 operating hours experience (), the estimated probability of survival is **p=0.26**:

**SE=4**; CO =1/(1+exp(-(-4.1030+0.7583\*SE))); CO

> CO

[1] **0.2554411**

The table below shows the probability of surviving surgery for several values of surgeons’ experience.

|  |  |
| --- | --- |
| Surgeon’s Experience | Probability of patient survival (Clinical Outcome) |
| 1 | 0.034 |
| 2 | 0.07 |
| 3 | 0.14 |
| 4 | 0.26 |
| 5 | 0.423 |

The output from the logistic regression analysis gives a p-value of , which is based on the Wald z-score. In addition to the Wald method, we can calculate the p-value for logistic regression using the **Likelihood Ratio Test (LRT)**, which for these data give 0.0006476922.

Estimate Std. Error z value Pr(>|z|) Wald

SE 0.7583 0.3139 2.416 **0.0157** \*

The logit of a number given by the formula: , representing the **log-odds** (of survival in this case).

confint(mylogit)

*So, why exponentiating the coefficients*? Because,

🡺 🡺 (odds-ratio, OR)

> **exp**(coef(mylogit)) # exponentiated logit model coefficients

(Intercept) SE

0.01652254 **2.13474149** ## == exp(0.7583456)

> coef(mylogit) # raw logit model coefficients

(Intercept) SE

-4.1030298 **0.7583456**

exp(cbind(**OR** = coef(mylogit), confint(mylogit)))

**OR** 2.5 % 97.5 %

(Intercept) 0.01652254 0.0001825743 0.277290

**SE 2.13474149 1.3083794719 4.839986**

with(mylogit, df.null - df.residual)

Finally, the **LRT (likelihood-ratio test)** **p-value** can be obtained using:

with(mylogit, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail = FALSE))

[1] 0.0006476922

The LRT tells us that our model as a whole fits significantly better than an empty model. The deviance residual is -2\*log likelihood, and to see the model's log likelihood:

logLik(mylogit)

'log Lik.' -8.046117 (df=2)

**Side-note**: The LRT compares the data fit of two models. For instance, removing predictor variables from a model will reduce model quality (i.e., a model will have a lower log likelihood). To statistically assess whether the observed difference in model fit is significant, the LRT compares the difference of the log likelihoods of the two models. When this difference is statistically significant, the full model (the one with more variables) is a better fit to the data, compared to the reduced model. LRT is computed from the log likelihoods of the models:

where and are the reduced and the full models, respectively, and denote the likelihoods of the 2 models, and and represent the log likelihood (natural log of the model likelihood.

The distribution of the LRT is chi-squared with degrees of freedom equal to the number of parameters that are reduced (i.e., the number of variables removed from the model). In our case, , as we have an intercept and one predictor (SE), and the null model is empty (no parameters).

## IV. Machine Learning Algorithms

**Questions**:

* How can we tie human intuition and computer-generated results to obtain reliable, effective, and efficient decision-support system (that facilitates, forecasting)?
* Niels Born – “*It is difficult to make predictions, especially about the future”* …
* Can we unsupervisely classify the data?

### **Prediction**

For most of the machine learning algorithms (including first-order linear regression), we:

* ﬁrst generate the model using training data, and then
* predict values for test/new data.

Predictions are made using the R **predict** function. (type **?predict.name**), where **name** is the function-name corresponding to the algorithm. The ﬁrst argument of predict often represents the variable storing the model and the second argument is a matrix or data frame of test data that the model needs to be applied to. Calling predict can be done in 2 ways: type **predict** or type of **predict.name**.

**Example**:

#mydata <- read.table('https://umich.instructure.com/files/330381/download?download\_frd=1&verifier=HpfmjfMFaMsk7rIpfPx0tmz960oTW7JA8ZonGvVC',as.is=T, header=T) # 01a\_data.txt

# mydata <- read.table('data.txt',as.is=T, header=T)

# (1) First, there are different approaches to split the data (partition the data) into

# training and testing sets.

## TRAINING: 75% of the sample size

sample\_size <- floor(0.75 \* nrow(mydata))

## set the seed to make your partition reproductible

set.seed(1234)

train\_ind <- sample(seq\_len(nrow(mydata)), size = sample\_size)

train <- mydata[train\_ind, ]

# TESTING DATA

test <- mydata[-train\_ind, ]

lin.mod <- lm(Weight ~ Height\*Team, **data=train**)

predicted.values <- predict(lin.mod, **newdata=test**)

### Data Modeling/Training

**Logistic Regression**:

glm\_model <-glm(ifelse(Weight > 200,1,0) ~ Height\*Team, family=binomial(link="logit"), **data=train**)

**K-Means Clustering**

train.1 <- cbind(train$Height, train$Weight, train$Age)

test.1 <- cbind(test$Height, test$Weight, test$Age)

Weight.1 <- ifelse(train$Weight > 200,1,0)

head(train.1)

kmeans\_model <- kmeans(**train.1**, 3)

plot(train.1, col = kmeans\_model$cluster)

points(kmeans\_model$centers, col = 1:2, pch = 8, cex = 2)

**k-Nearest Neighbor Classiﬁcation**

# install.packages("class")

library("class")

knn\_model <- knn(train=train.1, test=test.1, cl=as.factor(Weight.1), k=5)

plot(knn\_model)

summary(knn\_model)

**Naïve Bayes Classifier**

install.packages("e1071")

library("e1071")

nbc\_model <- naiveBayes(Weight ~ Height\*Age, data=train.1)

**Decision Trees (CART)**

#install.packages("e1071")

library("rpart")

cart\_model <- rpart(Weight ~ Height+Age, data= as.data.frame(train.1), method="class")

plot(cart\_model)

text(cart\_model)

**AdaBoost**

install.packages("ada")

# X be the matrix of features, and labels be a vector of 0-1 class labels.

library("ada")

boost\_model <- ada(x= cbind(train$Height, train$Weight, train$Age), y= Weight.1)

plot(boost\_model)

boost\_model

**Support Vector Machines (SVM)**

#install.packages("e1071")

library("rpart")

svm\_model <- svm(x= cbind(train$Height, train$Weight, train$Age), y=as.factor(Weight.1),

kernel ="radial")

summary(svm\_model)

## Appendix

### Example 1: Simulation (subject, day, treatment, observation)

This model is accounts for:

Response = Obs

Fixed effects:

Treatment (fixed)

Day (fixed)

Treatment\*Day interaction

Random Effects:

Subject nested within Treatment (random)

Day crossed with "Subject within Treatment" (random)

mydata <- data.frame(

Subject = c(13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 29, 30, 31, 32, 33,

34, 35, 36, 37, 38, 39, 40, 62, 63, 64, 65, 13, 14, 15, 16, 17, 18,

19, 20, 21, 22, 23, 24, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,

40, 62, 63, 64, 65, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,

29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 62, 63, 64, 65),

Day = c(rep(c("Day1", "Day3", "Day6"), each=28)),

Treatment = c(rep(c("B", "A", "C", "B", "C", "A", "A", "B", "A", "C", "B", "C",

"A", "A", "B", "A", "C", "B", "C", "A", "A"), each = 4)),

Obs = c(6.472687, 7.017110, 6.200715, 6.613928, 6.829968, 7.387583, 7.367293,

8.018853, 7.527408, 6.746739, 7.296910, 6.983360, 6.816621, 6.571689,

5.911261, 6.954988, 7.624122, 7.669865, 7.676225, 7.263593, 7.704737,

7.328716, 7.295610, 5.964180, 6.880814, 6.926342, 6.926342, 7.562293,

6.677607, 7.023526, 6.441864, 7.020875, 7.478931, 7.495336, 7.427709,

7.633020, 7.382091, 7.359731, 7.285889, 7.496863, 6.632403, 6.171196,

6.306012, 7.253833, 7.594852, 6.915225, 7.220147, 7.298227, 7.573612,

7.366550, 7.560513, 7.289078, 7.287802, 7.155336, 7.394452, 7.465383,

6.976048, 7.222966, 6.584153, 7.013223, 7.569905, 7.459185, 7.504068,

7.801867, 7.598728, 7.475841, 7.511873, 7.518384, 6.618589, 5.854754,

6.125749, 6.962720, 7.540600, 7.379861, 7.344189, 7.362815, 7.805802,

7.764172, 7.789844, 7.616437, NA, NA, NA, NA))

install.packages("lme4")

library("lme4", lib.loc="~/R/win-library/3.1")

m1 <- lmer(Obs ~ Treatment \* Day + (1 | Subject), mydata)

m1

Linear mixed model fit by REML ['lmerMod']

Formula: Obs ~ Treatment \* Day + (1 | Subject)

Data: mydata

REML criterion at convergence: 56.8669

Random effects:

Groups Name Std.Dev.

Subject (Intercept) 0.2163

Residual 0.2602

Number of obs: 80, groups: Subject, 28

Fixed Effects:

(Intercept) TreatmentB TreatmentC

7.1827 -0.6129 0.1658

DayDay3 DayDay6 TreatmentB:DayDay3

0.2446 0.4507 -0.1235

TreatmentC:DayDay3 TreatmentB:DayDay6 TreatmentC:DayDay6

-0.2740 -0.3508 -0.3327

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | Subject | Day | Treatment | Obs |
| 1 | 13 | Day1 | B | 6.472687 |
| 2 | 14 | Day1 | B | 7.01711 |
| 3 | 15 | Day1 | B | 6.200715 |
| 4 | 16 | Day1 | B | 6.613928 |
| 5 | 17 | Day1 | A | 6.829968 |
| 6 | 18 | Day1 | A | 7.387583 |
| 7 | 19 | Day1 | A | 7.367293 |
| 8 | 20 | Day1 | A | 8.018853 |
| 9 | 21 | Day1 | C | 7.527408 |
| 10 | 22 | Day1 | C | 6.746739 |
| 11 | 23 | Day1 | C | 7.29691 |
| 12 | 24 | Day1 | C | 6.98336 |
| 13 | 29 | Day1 | B | 6.816621 |
| 14 | 30 | Day1 | B | 6.571689 |
| 15 | 31 | Day1 | B | 5.911261 |
| 16 | 32 | Day1 | B | 6.954988 |
| 17 | 33 | Day1 | C | 7.624122 |
| 18 | 34 | Day1 | C | 7.669865 |
| 19 | 35 | Day1 | C | 7.676225 |
| 20 | 36 | Day1 | C | 7.263593 |
| 21 | 37 | Day1 | A | 7.704737 |
| 22 | 38 | Day1 | A | 7.328716 |
| 23 | 39 | Day1 | A | 7.29561 |
| 24 | 40 | Day1 | A | 5.96418 |
| 25 | 62 | Day1 | A | 6.880814 |
| 26 | 63 | Day1 | A | 6.926342 |
| 27 | 64 | Day1 | A | 6.926342 |
| 28 | 65 | Day1 | A | 7.562293 |
| 29 | 13 | Day3 | B | 6.677607 |
| 30 | 14 | Day3 | B | 7.023526 |
| 31 | 15 | Day3 | B | 6.441864 |
| 32 | 16 | Day3 | B | 7.020875 |
| 33 | 17 | Day3 | A | 7.478931 |
| 34 | 18 | Day3 | A | 7.495336 |
| 35 | 19 | Day3 | A | 7.427709 |
| 36 | 20 | Day3 | A | 7.63302 |
| 37 | 21 | Day3 | C | 7.382091 |
| 38 | 22 | Day3 | C | 7.359731 |
| 39 | 23 | Day3 | C | 7.285889 |
| 40 | 24 | Day3 | C | 7.496863 |
| 41 | 29 | Day3 | B | 6.632403 |
| 42 | 30 | Day3 | B | 6.171196 |
| 43 | 31 | Day3 | B | 6.306012 |
| 44 | 32 | Day3 | B | 7.253833 |
| 45 | 33 | Day3 | C | 7.594852 |
| 46 | 34 | Day3 | C | 6.915225 |
| 47 | 35 | Day3 | C | 7.220147 |
| 48 | 36 | Day3 | C | 7.298227 |
| 49 | 37 | Day3 | A | 7.573612 |
| 50 | 38 | Day3 | A | 7.36655 |
| 51 | 39 | Day3 | A | 7.560513 |
| 52 | 40 | Day3 | A | 7.289078 |
| 53 | 62 | Day3 | A | 7.287802 |
| 54 | 63 | Day3 | A | 7.155336 |
| 55 | 64 | Day3 | A | 7.394452 |
| 56 | 65 | Day3 | A | 7.465383 |
| 57 | 13 | Day6 | B | 6.976048 |
| 58 | 14 | Day6 | B | 7.222966 |
| 59 | 15 | Day6 | B | 6.584153 |
| 60 | 16 | Day6 | B | 7.013223 |
| 61 | 17 | Day6 | A | 7.569905 |
| 62 | 18 | Day6 | A | 7.459185 |
| 63 | 19 | Day6 | A | 7.504068 |
| 64 | 20 | Day6 | A | 7.801867 |
| 65 | 21 | Day6 | C | 7.598728 |
| 66 | 22 | Day6 | C | 7.475841 |
| 67 | 23 | Day6 | C | 7.511873 |
| 68 | 24 | Day6 | C | 7.518384 |
| 69 | 29 | Day6 | B | 6.618589 |
| 70 | 30 | Day6 | B | 5.854754 |
| 71 | 31 | Day6 | B | 6.125749 |
| 72 | 32 | Day6 | B | 6.96272 |
| 73 | 33 | Day6 | C | 7.5406 |
| 74 | 34 | Day6 | C | 7.379861 |
| 75 | 35 | Day6 | C | 7.344189 |
| 76 | 36 | Day6 | C | 7.362815 |
| 77 | 37 | Day6 | A | 7.805802 |
| 78 | 38 | Day6 | A | 7.764172 |
| 79 | 39 | Day6 | A | 7.789844 |
| 80 | 40 | Day6 | A | 7.616437 |
| 81 | 62 | Day6 | A | NA |
| 82 | 63 | Day6 | A | NA |
| 83 | 64 | Day6 | A | NA |
| 84 | 65 | Day6 | A | NA |

### Example 2: Genotype-phenotype

Save this data file as a tab-separated TXT file:

|  |  |  |  |
| --- | --- | --- | --- |
| Genotype | Race | Subject | Weight |
| A | 1 | 1 | 8 |
| A | 1 | 2 | 9 |
| A | 1 | 3 | 11 |
| A | 1 | 4 | 12 |
| A | 1 | 5 | 10 |
| A | 2 | 1 | 17 |
| A | 2 | 2 | 17 |
| A | 2 | 3 | 16 |
| A | 2 | 4 | 15 |
| A | 2 | 5 | 19 |
| A | 2 | 6 | 18 |
| A | 2 | 7 | 18 |
| A | 2 | 8 | 18 |
| A | 2 | 9 | 24 |
| A | 3 | 1 | 12 |
| A | 3 | 2 | 12 |
| A | 3 | 3 | 16 |
| A | 3 | 4 | 15 |
| A | 3 | 5 | 15 |
| A | 3 | 6 | 14 |
| A | 4 | 1 | 17 |
| A | 4 | 2 | 20 |
| A | 4 | 3 | 20 |
| A | 4 | 4 | 19 |
| A | 4 | 5 | 19 |
| A | 4 | 6 | 18 |
| A | 4 | 7 | 20 |
| A | 4 | 8 | 19 |
| A | 4 | 9 | 19 |
| B | 5 | 1 | 9 |
| B | 5 | 2 | 12 |
| B | 5 | 3 | 13 |
| B | 5 | 4 | 16 |
| B | 5 | 5 | 14 |
| B | 5 | 6 | 14 |
| B | 6 | 1 | 10 |
| B | 6 | 2 | 10 |
| B | 6 | 3 | 9 |
| B | 6 | 4 | 8 |
| B | 6 | 5 | 13 |
| B | 6 | 6 | 9 |
| B | 6 | 7 | 11 |
| B | 7 | 1 | 12 |
| B | 7 | 2 | 16 |
| B | 7 | 3 | 17 |
| B | 7 | 4 | 15 |
| B | 7 | 5 | 15 |
| B | 7 | 6 | 15 |
| B | 8 | 1 | 9 |
| B | 8 | 2 | 6 |
| B | 8 | 3 | 8 |
| B | 8 | 4 | 8 |
| B | 8 | 5 | 13 |
| B | 8 | 6 | 9 |
| B | 8 | 7 | 9 |
| B | 8 | 8 | 10 |

# data <- read.table('C:\\Users\\Dinov\\Desktop\\data.txt',as.is=T, header=T)

data <- read.table('data.txt',as.is=T, header=T)

names(data)

attach(data)

table(Genotype, Race)

table(Race, Subject)

# for demonstration, construct a balanced data set

# 5 subjects for each race

data\_balance <- data[data$Subject <=5,]

# create factors

data\_balance$g <- as.factor(data\_balance$Genotype)

data\_balance$t <- as.factor(data\_balance$Race)

data\_balance$s <- as.factor(data\_balance$Subject)

# fit the ANOVA

anova.model <- lm(Weight ~ g+t, data= data\_balance)

# get the ANOVA table

anova(anova.model)

# note that all F tests use MSE, subjects within race as denominator

# will need to hand-calculate test for genotypes

# Random effects modeling estimation using REML

library(lme4)

lme.model <- lmer(Weight~g+(1|t),data= data\_balance)

summary(lme.model)

anova(lme.model)

# various extractor functions:

fixef(lme.model) # estimates of fixed effects

vcov(lme.model) # VC matrix for the fixed effects

VarCorr(lme.model) # estimated variance(-covariance) for random effects

ranef(lme.model) # predictions of random effects

coef(lme.model) # fixed effects + pred's of random effects

fitted(lme.model) # conditional means for each obs (X bhat + Z uhat)

resid(lme.model) # conditional residuals (Y - fitted)

# REML is the default method for estimating variance components.

# If want to use ML, can specify that

lmer(Weight~g+(1|t),REML=F, data= data\_balance)

# Now, to get p-values for the effects, or construct a ci

# inference on Subject weight focus is the difference between the two genotypes

# which is Cb for C = [0, 1] using the default R parameterization

# following assumes lme4 library loaded, data frame is d

# uses full data set (unbalanced)

# also assumes g and t are factors identifying genotypes and trays

# fit the model

data$g <- as.factor(data$Genotype)

data$t <- as.factor(data$Race)

data$s <- as.factor(data$Subject)

# See R DATA TYPES: <http://www.statmethods.net/input/datatypes.html>

model.lmer <- lmer(Weight ~ g + (1|t), data= data)

# get a confidence interval for g

# slower, obvious programing

nsim <- 10

gdiff <- rep(NA, nsim)

for (i in 1:nsim) {

data$y <- simulate(model.lmer) # param bootstrap data set

model.lmer.1 <- lmer(unlist(data$y) ~ g + (1|t), data= data)

# we need to turn the list of simulated values (y) into an atomic vector with unlist()

# both data frames and models objects (e.g., produced by lm()) are lists

# <http://adv-r.had.co.nz/Data-structures.html>

gdiff[i] <- fixef(model.lmer.1)[2] # keep only the est diff

}

quantile(gdiff, c(0.025, 0.975))

# print 95% CI of the coefficient/effect-size of genotype(g)

# print model summary

summary(model.lmer)

# two ways to speed up CI construction and increase simulations to 1K

# use apply to avoid the for loop

# use refit() to avoid the setup time before fitting the LME

yall <- simulate(model.lmer, nsim=1000)

gdiff <- apply(yall, 2, function(y) {fixef(refit(model.lmer, y))[2]})

quantile(gdiff, c(0.025, 0.975))

# To get both the estimate and the SE

# which are in the fixed effect table returned by summary()

# use the extractor functions fixef() and vcov()

yall <- simulate(model.lmer, nsim=1000)

gdt <- apply(yall, 2, function(y) {

model.lmer.2 <- refit(model.lmer, y);

c(fixef(model.lmer.2)[2], sqrt(vcov(model.lmer.2)[2,2]) )

})

gdt <- t(gdt) # because result of apply is 2 rows, nsim cols

# Using gdt, we can obtain a parametric bootstrap-t interval

# Hypothesis testing: here we need to simulate under **H0: g = 0**

# To find the t statistic and the F statistic

# t statistic is in the @coefs table

# F statistic is a value in the anova() output

# if more than one test, need to subscript (unless you want all F's)

# need to reference as a list to avoid R issues

# Complete model with genotypes

model.lmer <- lmer(Weight ~ g + (1|t), data= data)

model.lmer.0 <- lmer(Weight ~ (1|t), data=data)

yall <- simulate(model.lmer.0, nsim=1000)

# NB: simulate data under model.lmer.0

gdt0 <- apply(yall, 2, function(y) {

model.lmer.2 <- refit(model.lmer, y); # but analyze under alt. model (model.lmer)

c(summary(model.lmer.2), anova(model.lmer.2)$'F value' )

})

gdt0 <- t(gdt0)

# obs t = 1.438179, p-value using t statistic:

coef(summary(model.lmer))

mean(abs(gdt0[,1]$sim\_1$residuals) >= 1.438179)

# p-value = 0.125

# obs F = 2.0684

anova(model.lmer)

mean(gdt0[,2]$sim\_2$residuals >= 2.0684)

# p-value = 0.01785714

**Baseball data**

<http://wiki.socr.umich.edu/index.php/SOCR_Data_MLB_HeightsWeights>

data <- read.table('E:\\Ivo.dir\\Research\\UMichigan\\Education\_Teaching\_Curricula\\2015\_2016\\HS\_853\_Fall\_2015\\Modules\_docx\\01a\_data.txt',as.is=T, header=T)

boxplot(Weight ~ Position, data = data, xlab = "Position", ylab = "Weight",

main = "MLB Weight Distribution by Position")

boxplot(Weight ~ Team, data = data, xlab = "Team", ylab = "Weight",

main = "MLB Team Weight Distributions")

### **References**

Bates, D.M., Maechler, M., & Bolker, B. (2012). lme4: Linear mixed-effects models using S4 classes. R package version 0.999999-0.

Baayen, R.H. (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics Using R*. Cambridge: Cambridge University Press.

Baayen, R.H., Davidson, D.J., Bates, D.M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59, 390-412.

Barr, D.J., Levy, R., Scheepers, C., & Tilly, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, 68, 255–278.

Bolker, B. M., Brooks, M. E., Clark, C. J., Geange, S. W., Poulsen, J. R., Stevens, M. H. H., & White, J. S. S. (2009). Generalized linear mixed models: a practical guide for ecology and evolution. *Trends in Ecology & Evolution*, 24(3), 127-135.

Wike, E.L., & Church, J.D. (1976). Comments on Clark’s “The language-as-fixed-effect fallacy”. *Journal of Verbal Learning & Verbal Behavior*, 15, 249-255.

Winter, B. (2013). Linear models and linear mixed effects models in R with linguistic applications. arXiv:1308.5499. [http://arxiv.org/pdf/1308.5499.pdf]

1. <http://www.statmethods.net/input/datatypes.html> [↑](#footnote-ref-1)
2. <http://www.statmethods.net/advstats/glm.html> [↑](#footnote-ref-2)
3. <http://www.statmethods.net/management/subset.html> [↑](#footnote-ref-3)
4. <http://wiki.socr.umich.edu/index.php/SOCR_Data_MLB_HeightsWeights#Data_Table> [↑](#footnote-ref-4)
5. <http://wiki.socr.umich.edu/index.php/SOCR_EduMaterials_Activities_PowerTransformFamily_Graphs> [↑](#footnote-ref-5)
6. <http://socr.umich.edu/html/dist/> [↑](#footnote-ref-6)
7. <http://wiki.socr.umich.edu/index.php/SOCR_Data_Dinov_020108_HeightsWeights> [↑](#footnote-ref-7)
8. <http://wiki.socr.umich.edu/index.php/SOCR_Data_MLB_HeightsWeights> [↑](#footnote-ref-8)
9. <http://www.rstudio.com/products/rstudio/download/> [↑](#footnote-ref-9)
10. <http://cran.r-project.org/web/packages/lme4/lme4.pdf> [↑](#footnote-ref-10)
11. <http://www.r-tutor.com/r-introduction/data-frame/data-import> [↑](#footnote-ref-11)
12. <http://www.statmethods.net/input/importingdata.html> [↑](#footnote-ref-12)
13. <http://wiki.socr.umich.edu/index.php/SOCR_Data_MLB_HeightsWeights> [↑](#footnote-ref-13)
14. <http://en.wikipedia.org/wiki/Likelihood-ratio_test> [↑](#footnote-ref-14)
15. <http://cran.r-project.org/web/packages/influence.ME/influence.ME.pdf> [↑](#footnote-ref-15)
16. <http://www.ats.ucla.edu/stat/mult_pkg/glmm.htm> [↑](#footnote-ref-16)
17. <http://wiki.socr.umich.edu/index.php/SMHS_OR_RR> [↑](#footnote-ref-17)