

Transitioning from SPSS to R

Daryn Cushnie-Sparrow — R-Ladies London


July 7, 2017

Introduction

For many of us, SPSS has been something we've used for statistical analysis since early in our experience with statistics. Running tests in SPSS might be second-nature to you, and there are lots of resources online for how to do things you may be unclear about. The comfort of SPSS can make it daunting to explore other options; however, SPSS has limitations, and you may have stumbled upon an interesting statistical test that can be run in R and not in SPSS. Alternatively, you may have decided that SPSS licenses are very expensive and inconvenient, and you're interested in other options. I have recently been making the jump from SPSS to R, so I'm hoping to share some of what I've learned that might ease your transition.

Initial Setup


We're going to set up an RStudio project that we'll be working in today. Here are the steps you'll need to follow:

1. **Download and install R**
 - [Windows](#)
 - [Mac](#)
2. **Download and install RStudio**
 - [Here](#) - Scroll down and find your operating software
3. **Create a new RStudio project in a directory that works for you**
 - Open RStudio
 - Click the  in the top right corner
 - Choose "New Project"... "New Directory"... Then specify the appropriate path and name your folder something like 'SPSStoR'
 - [These](#) instructions might be helpful if you are stuck
4. **Download the .Rmd and .csv files provided into your RStudio project directory**
 - [Here's the Google drive link](#)

So if my path when I was creating my directory looked like this:

New Project

[Back](#) **Create New Project**



Directory name:

Create project as subdirectory of:

[Browse...](#)

☐ Use packrat with this project

☐ Open in new session


[Create Project](#) [Cancel](#)

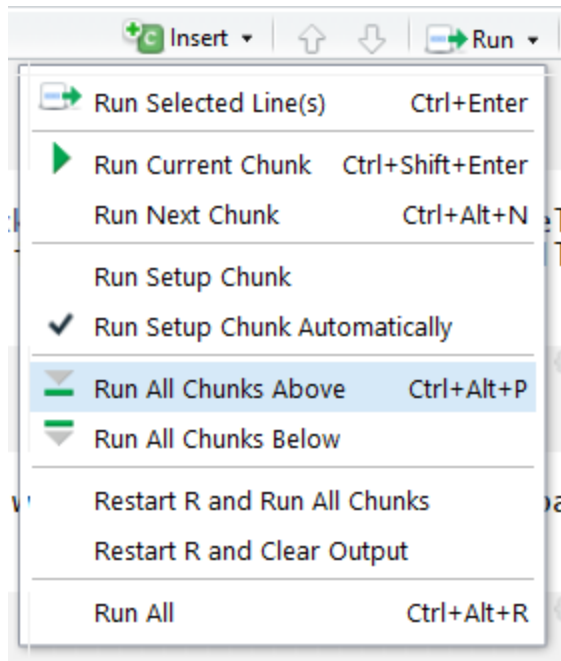
Then your folder should look like this once you've downloaded the files.

This PC > Desktop > SPSStoR

Name	Date modified	Type
.Rproj.user	2017-07-06 8:39 A...	File folder
OnOffRLadies.csv	2017-06-24 11:16 ...	Microsoft Excel (
SPSStoR.Rmd	2017-07-06 8:37 A...	RMD File
SPSStoR.Rproj	2017-07-06 8:39 A...	R Project

5. **Open the .Rmd file in your RStudio project**

- Click the  in the top left
- Select the .Rmd you downloaded into the same directory
- 6. **In the .Rmd file, put your cursor on the line just before the section title 'Reading Data into R'**
- Then click "Run All Chunks Above"



Packrat Setup

SPSS is a one-piece statistical package. All the functionalities and features available to you are included in the installation. R has many useful base functions, but there are tons of packages available from CRAN that allow you to do more things or do them better. When you are working on a large project, you can start to have a lot of packages you're using, and if you have several of these projects going at once, you might have different packages in each project and different versions. Updating your packages in one project can break your old code in another. Packrat can help you solve this issue. It's a dependency management package that makes your projects isolated, portable and reproducible by keeping your packages in a private library for each project and automatically restores all those packages when you re-open your project or even open it on a different computer. At the beginning of any project, initialize Packrat by calling `packrat::init`.

```
install.packages("packrat")  
packrat::init()
```

Now, any packages you install will be part of your packrat library. We're going to need a few packages from the tidyverse, so we'll just grab the whole thing. We'll also need a few other packages. I've listed them all below. Run this block to get your packages installed.

```
install.packages("tidyverse")  
install.packages("Hmisc")  
install.packages("car")  
install.packages("lmtest")  
install.packages("ez")  
install.packages("psych")  
install.packages("ggplot2")  
packrat::snapshot()
```

We can check to see if everything we call in our project is available in the packrat library by calling `packrat::status()`. It will also tell us if it detects package calls in the script that are not in the library, and it will tell us if we have packages in our library that aren't called anywhere in the script.

```
packrat::status()
```

We'll need to output a couple files. Wherever you set up your project when you opened R Studio will be your working directory. Save your path to a variable so you don't have to change it each time it comes up.

```
path <- getwd()
```

Reading Data into R

There are lots of ways to read files into R. `read.table` and its wrappers are helpful functions that are part of base R. I like using the `readr` package from the tidyverse. Let's read in the data that we're going to be using today. I've saved this file as a `.csv` - easily used by both Excel and R. We already installed `readr` when we installed the tidyverse earlier, so now we need to load it using `library`.

```
library(readr)
```

```
OnOffMaster <- read_csv(file.path(path, "OnOffRLadies.csv"),  
  col_names = TRUE)
```

Our `.csv` will have come in as a data frame. Let's take a look at what we have. You can do this in a few ways. Run each of these lines separately (using `Ctrl+Enter` or `Command+Enter`) and look at the different ways R will show you your data frame.

```
str(OnOffMaster)
```

```
head(OnOffMaster)
```

```
tail(OnOffMaster, n = 10)
```

```
library(dplyr)
```

```
glimpse(OnOffMaster)
```

You can see that this data frame has 62 observations of 26 variables. This data contains measurements of the voice quality of individuals with Parkinson's disease (PD) while they are on and off their dopamine medication. We have data on 51 people with PD and on 11 controls. For the controls, we only have measurements for them off medication since they don't normally take dopamine medication. The `ON` and `DIFF` (difference score) columns are entirely missing for these 11 observations. Missing values in R have the value `NA`. This is different than in SPSS, where missing values show up as a single period.

Variable Information * `VAS` stands for "visual-analogue scale" and these numbers represent the person's voice quality as perceived by unfamiliar listeners. * `UPDRS` is the Unified Parkinson's Disease Rating Scale, which is commonly used to assess symptoms of PD. * `LED` is a person's Levodopa Equivalent Dose, which we use to compare individuals' dopamine medication, since people often use different combinations of medications which have different potency. * `Shimmer`, `HNRR`, `CPPS` and `AVQI` are all acoustic measures of voice quality. We also have a couple grouping variables, stored as factors. * `PDGroup` separates individuals with PD from controls. * `VASGroup` divides people based on their perceived voice quality off-medication. Individuals with poor voice quality off-medication are coded as `Low`, and those with better voice quality are coded as `High`. Controls are coded as `Con`. * `DurationGroup` divides people based on how long they've had PD. They are binned into `Early`, `Early-Mid`, `Mid-Late` and `Late`.

Datasets in SPSS vs. Data Frames in R

In SPSS, we use datasets in the form of a spreadsheet. SPSS datasets are saved as a .sav, store variables in columns and cases in rows. In R, we have several data structures that allow us to handle our data differently. Typically, you'll store your data in a data frame. A data frame is similar to an SPSS dataset, with variables stored in columns and observations in rows. R allows you to pick out particular variables or observations in a variety of ways, such as indices, dollar notation or functions like dplyr's select. If you've ever agonized over restructuring your data in SPSS/Excel by copy and pasting columns, this flexibility will be something you'll appreciate. Let's take a look at how you can select variables and observations from a data frame using indices and dollar notation.

```
# Prints a vector of the whole 7th column
OnOffMaster[,7]
# Same as the above - R defaults to the column if no comma is present
OnOffMaster[7]
# Prints a vector of all measurements on the 10th observation
OnOffMaster[10, ]
# Prints the value of the 9th variable on the 50th observation
OnOffMaster[50, 9]
# Prints a vector containing the VAS_OFF column of OnOffMaster
OnOffMaster$VAS_OFF
```

These are helpful ways to select information from a data frame. You'll probably want to use dollar notation over indices in most cases, because if you delete or rearrange columns, what was once your 5th variable could now be something different.

The dplyr package offers some great data manipulation tools. With select, you can use helper functions like starts_with, ends_with, contains and more. You can also just provide a list of variables you'd like it to select, or you can put a minus sign in front to tell it to remove that item. I had some key columns that delineated each condition (OFF, ON, and DIFF). These columns are unnecessary, and just provided visual separation in Excel when I was working with a larger version of this dataset. Let's remove them using dplyr's select. We'll start working with a different data frame, just in case we ever want to access to original without reading it back in.

```
library(dplyr)
OnOff <- select(OnOffMaster, -starts_with("Cond"))
glimpse(OnOff)
```

We can see that all 3 of those key columns are now gone, since had "Cond" at the start. We now have 23 variables, so we know we haven't deleted anything else by mistake. Another thing we need to make sure we do is factor all our variables that aren't measurements. You can do this automatically when reading it in using the base R function read.csv by specifying stringsAsFactors = FALSE, but I prefer to do this manually so I can specify my levels. If you don't specify your levels, readr will list the levels alphabetically. The default for ordering is ordered = FALSE, meaning that the variable is categorical with no ordinance. If you have an ordinal variable, you'll need to specify ordered = TRUE.

```
OnOff$Participant <- factor(OnOff$Participant)
OnOff$PDGroup <- factor(OnOff$PDGroup, levels = c("PD", "Con"))
OnOff$VASGroup <- factor(OnOff$VASGroup, levels = c("Low", "High",
```

```
"Con"))
OnOff$DurationGroup <- factor(OnOff$DurationGroup,
                             levels = c("Early", "Early-Mid",
                                         "Mid-Late", "Late", "Con"))
```

In addition to selecting variables, sometimes you might want to choose particular observations. You can do this with dplyr's filter. For some analyses, I might only want to include people with PD. Let's pick those ones and leave out all the controls.

```
pds <- filter(OnOff, PDGroup == "PD")
glimpse(pds)
```

The result has 23 variables, but only 51 observations. There are some analyses I want to do that include only people with PD, but I also don't want to use any of the difference scores. We could do this by using select on the data frame we just made above. Another way we could do this is using the %>% operator, called a 'pipe'. Piping functions together is helpful if you're doing several manipulations.

```
PDsOnOff <-
OnOff %>%                                # Specify what data to use
select(-ends_with("DIFF")) %>%          # Skip anything that ends with
"DIFF"
filter(PDGroup == "PD")                  # Only take PD observations
```

Writing an SPSS file

If you wanted to compare SPSS and R or you had a test you wanted to run in SPSS, you'll need to save that data as a .sav. You can do this using the haven package from the tidyverse.

```
library(haven)
write_sav(OnOff, file.path(path, "OnOffData.sav"))
```

Check your project directory to see if you successfully wrote the file here. One reason you might want to use R before creating an SPSS file would be that importing variable names from an Excel document in SPSS is more challenging. R reads column names easily and allows you to specify variable types in batches, which haven feeds into the .sav. When you open this file in SPSS, your factors will show as nominal with ordered factors as ordinal. It will also convert your factor levels to numerical values so that SPSS knows how to handle them, but you will store your labels as part of the value, so your SPSS output will show your label names instead.

Correlation

Bivariate correlations are a helpful tool in most fields, allowing us to quickly get a sense of association between variables. Base R allows us to do a bivariate correlation easily using cor. Let's see if there are associations between our voice quality measures. We'll compare across PD and control, so we'll only test in the off-state. I only want to include UPDRS, VAS, shimmer, HNR, CPPS and AVQI. Put those into a data frame named "bivars".

```
# Exercise 1:
# Select UPDRS, VAS, shimmer, HNR, CPPS and AVQI, only in the off
state.
# Save this to a data frame called bivars.
```

Now that we have our data frame, let's use `cor` to run Pearson correlations. We want missing values to be omitted on a pairwise basis, which it will find in the UPDRS column for controls.

```
cor(bivars, method = "pearson", use = "pairwise.complete.obs")
```

This was convenient, but notice that this output only includes the *r* values, not *p* values. Most of the time, we're going to want both of those. Lots of packages and functions exist that can do this. I like `rcorr` from the `Hmisc` package. `rcorr` only accepts a matrix, so we'll need to use `as.matrix` in the `rcorr` call to coerce our data frame to a matrix.

```
library(Hmisc)
```

```
bivarsOut <- rcorr(as.matrix(bivars), type = "pearson")
```

The output from this is 3 tables. The first includes the *r* values, the second includes the number of pairwise observations used to analyze each pair of variables, and the third includes the *p* values for each correlation. Using `str` on this output will reveal that this is a list of 3 matrices. If you want to convert this output into a data frame, you'll need to select individual elements. You can do this with dollar notation.

```
bivarsOut$r
```

```
bivarsOut$n
```

```
bivarsOut$p
```

I stumbled upon a handy custom function [here](#) that flattens `rcorr` output into a nice data frame including only the *r* and *p* values.

```
flattenCorrMatrix <- function(cormat, pmat) {  
  ut <- upper.tri(cormat)  
  data.frame(  
    row = rownames(cormat)[row(cormat)[ut]],  
    column = rownames(cormat)[col(cormat)[ut]],  
    cor = (cormat)[ut],  
    p = pmat[ut]  
  )  
}
```

```
bivarsflat <- flattenCorrMatrix(bivarsOut$r, bivarsOut$p)
```

I personally don't like scientific notation. I prefer long decimals until I'm ready to do final rounding. You can turn off scientific notation using options. All this does is change how R prints out values within that session - not how it calculates behind the scenes. Look at our flattened correlation results again now to see the difference.

```
options(scipen = 999)
```

If you wanted to change it back, you can just run `options(scipen = 0)`.

We often default to Pearson to take a quick look at associations, but if we want to report results, we should check that parametric testing is appropriate. Let's check the normality of our variables using the Shapiro-Wilk test.

```
shapiro.test(bivars$UPDRS_OFF)
```

```
shapiro.test(bivars$VAS_OFF)
```

```
shapiro.test(bivars$Shimmer_OFF)
```

```
shapiro.test(bivars$HNR_OFF)
```

```
shapiro.test(bivars$CPPS_OFF)
```

```
shapiro.test(bivars$AVQI_OFF)
```

If we had run the Shapiro-Wilk as a part of SPSS' Explore procedure, the output would look like this:

Tests of Normality

	Kolmogorov-Smirnov ^a			Shapiro-Wilk	
	Statistic	df	Sig.	Statistic	df
UPDRS_OFF	.090	51	.200 [*]	.973	51
VAS_OFF	.057	62	.200 [*]	.984	62
Shimmer_OFF	.190	62	.000	.863	62
HNR_OFF	.109	62	.066	.978	62
CPPS_OFF	.120	62	.028	.973	62
AVQI_OFF	.118	62	.032	.958	62
UPDRS_ON	.106	51	.200 [*]	.970	51
VAS_ON	.089	51	.200 [*]	.972	51
Shimmer_ON	.241	51	.000	.820	51
HNR_ON	.102	51	.200 [*]	.951	51
CPPS_ON	.082	51	.200 [*]	.980	51
AVQI_ON	.119	51	.069	.967	51
UPDRS_DIFF	.159	51	.002	.925	51
VAS_DIFF	.112	51	.154	.970	51
Shimmer_DIFF	.188	51	.000	.910	51
HNR_DIFF	.115	51	.088	.933	51
CPPS_DIFF	.078	51	.200 [*]	.965	51
AVQI_DIFF	.115	51	.088	.939	51

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

Testing for Normality in SPSS' Explore

The results show that shimmer and AVQI are non-normal. In this case, we would need to use a Spearman correlation instead. Let's recalculate our correlations using Spearman instead of Pearson. Then we'll flatten it and pick out the ones that include Shimmer and AVQI.

```
spearOut <- rcorr(as.matrix(bivars), type = "spearman")
```

Now that we have that output, let's flatten it again using that handy custom function.

```
# Exercise 2:
```

```
# Use flattenCorrMatrix() to flatten our Spearman rcorr output
```

```
# into a data frame.
```

Here I've just pulled out the relevant correlations from each flattened matrix.

```
spear <- rbind(spearFlat[2:3,], spearFlat[6,],  
              spearFlat[9,], spearFlat[11:15,])
```

```
pear <- rbind(bivarsflat[1,], bivarsflat[4:5,], bivarsflat[7:8,],  
             bivarsflat[10,])
```

Your turn! Test the association between UPDRS and HNR, on medication. First, test for normality using the Shapiro-Wilk test. Then run rcorr. Save the results to 'ex3'. Use flattenCorrMatrix to tidy the output, and save that to 'ex3table'.

```
# Exercise 3:
```

```
# Run bivariate correlation between UPDRS and HNR, on medication.
```

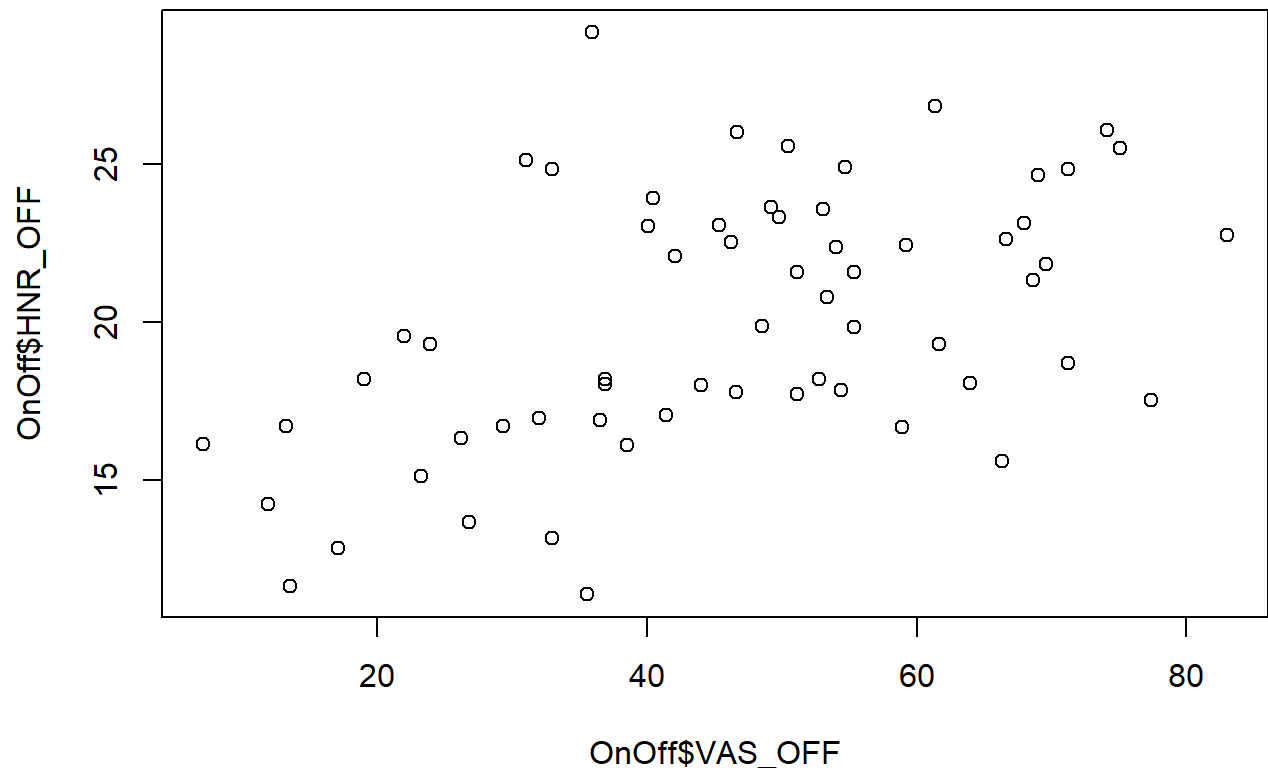
Simple linear regression

We're just going to touch on regression today, mostly as it pertains to visualization correlation. Regression is a *huge* area in R, and there are lots of resources that can help you figure out what regression can do for you. I've included some resources at the end in the **Resources** section to get you started.

We've got a .49 correlation between VAS and HNR - let's take a deeper look at that. How is an individual's perceived voice quality affected by their harmonic-to-noise ratio? This would give us a clue about how strongly related this acoustic measure is to our perception of quality.

One of the assumptions of linear regression is a roughly linear relationship between the variables. Let's quickly visualize it with base graphics.

```
plot(OnOff$VAS_OFF, OnOff$HNR_OFF)
```



You can see a roughly linear positive trend, so we can proceed. Other assumptions need to be tested after we set up our model, so let's do that next.

```
# Regression formula syntax is y ~ x, or y as a function of x.
# x is our independent variable (HNR in this case),
# and y is our dependent variable (VAS in this case).
```

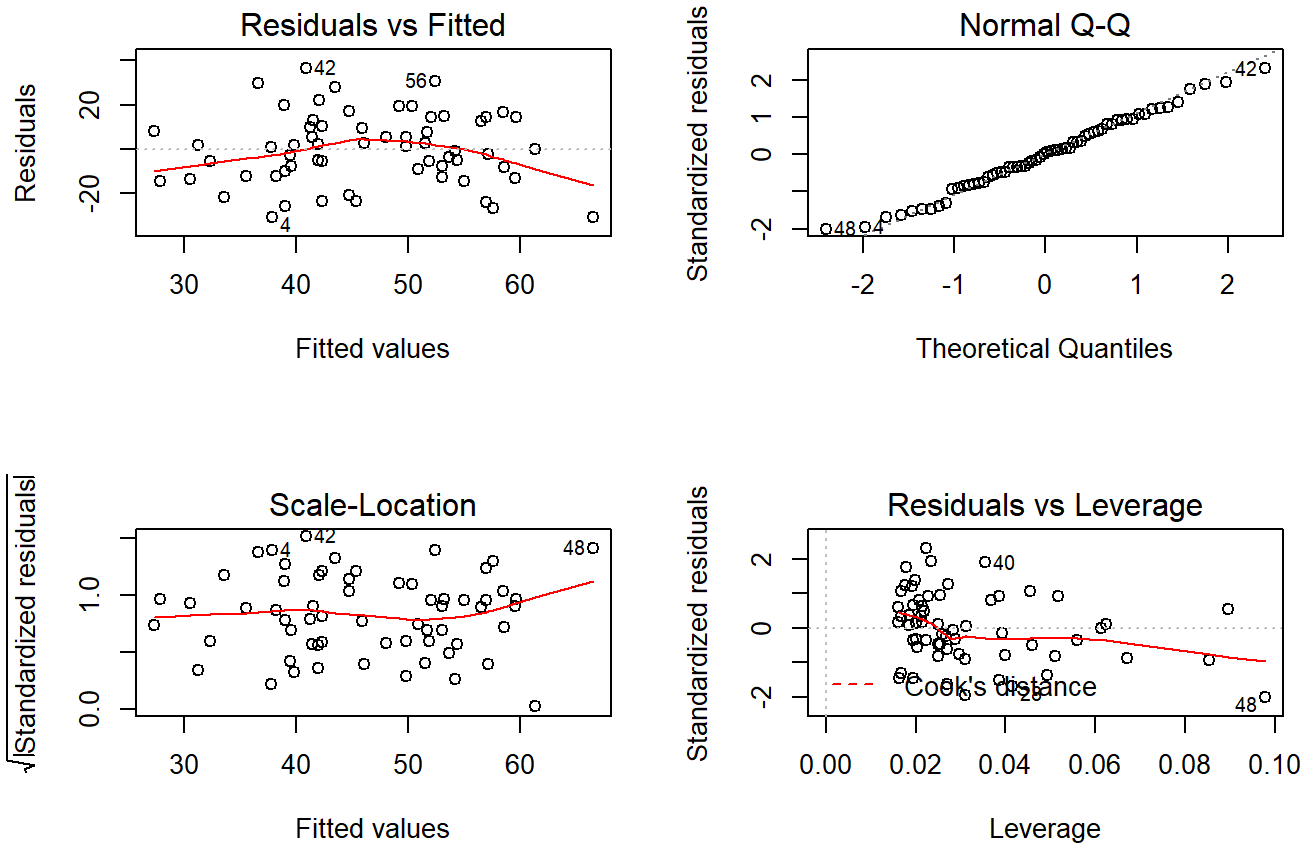
```
vasHNRmodel <- lm(VAS_OFF ~ HNR_OFF, data = OnOff)
```

We're going to test 3 very important assumptions: independence of observations, homogeneity of variance (homoscedasticity) and normality of residuals. We can test independence with the Durbin-Watson test, `dwtest` from the `lmtest` package. We want Durbin-Watson to be non-significant, with a test statistic close to 2. Homogeneity of variance and normality of residuals are both assessed visually by calling `plot` on a model. The 2 plots on the left both allow us to assess homogeneity of variance. If homoscedasticity is upheld, the line should be approximately flat. The top right plot (a Q-Q plot) shows us the normality of the residuals. The points should be very close to the line if the residuals are normal.

```
library(lmtest)
# Durbin-Watson to check independence of observations/autocorrelation
dwtest(vasHNRmodel)
```

```
# Homogeneity of variance (2 left plots) & normality of residuals
# (top-right)
```

```
par(mfrow=c(2,2))
plot(vasHNRmodel)
```



Now that we know our assumptions are upheld, let's take a look at a summary of our model

```
summary(vasHNRmodel)
```

We see that R-squared is equal to 0.24, so we can say that about 24% of the variance in perceived voice quality is predicted by HNR.

If we did our regression in SPSS, our output would look like this:

Model Summary^b

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Durbin-Watson
1	.494 ^a	.244	.231	15.99529	2.110

a. Predictors: (Constant), HNR_OFF

b. Dependent Variable: VAS_OFF

ANOVA^a

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	4943.978	1	4943.978	19.324	.000 ^b
	Residual	15350.962	60	255.849		
	Total	20294.940	61			

a. Dependent Variable: VAS_OFF

b. Predictors: (Constant), HNR_OFF

Coefficients^a

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	95.0% Confidence Interval for	
		B	Std. Error	Beta			Lower Bound	Upper Bound
1	(Constant)	2.222	10.246		.217	.829	-18.272	22.716
	HNR_OFF	2.202	.501	.494	4.396	.000	1.200	3.204

a. Dependent Variable: VAS_OFF

Residuals Statistics^a

	Minimum	Maximum	Mean	Std. Deviation	N
Predicted Value	27.3000	66.4656	46.3676	9.00271	62
Residual	-30.68518	36.48457	.00000	15.86364	62
Std. Predicted Value	-2.118	2.232	.000	1.000	62
Std. Residual	-1.918	2.281	.000	.992	62

a. Dependent Variable: VAS_OFF

SPSS Linear Regression Output

Information in the **Model Summary** table is found at the bottom of the summary of our model, plus Durbin-Watson which we called separately using `dwtest`. If we wanted to get the information in the **ANOVA** table, we can call a summary on an `aov` on our model.

```
summary(aov(vasHNRmodel))
```

The **Coefficients** table is similar to our summary. What we're missing are standardized

coefficients and confidence intervals. We can get the standardized coefficient by calling the `lm.beta` function in the `QuantPsyc` package on our model. Confidence intervals can be calculated using `confint`.

```
# Confidence interval
confint(vasHNRmodel, 'HNR_OFF', level = 0.95)
confint(vasHNRmodel, '(Intercept)', level = 0.95)
```

A word of caution: If you load in the `QuantPsyc` package, it will load `MASS` by default. `MASS` has a `select` function that will mask `dplyr`'s `select`. You'll notice this when you next try to use `select`, because you'll get error messages about incorrect arguments. To correct this problem, you can reassign `select` like this.

```
select <- dplyr::select
```

`R` doesn't give statistics on the residuals by default, but this can be done. You should be aware that `SPSS` calculates standardized residuals based by dividing residuals by their root mean square error. You can look into this by looking at the [SPSS Algorithm Manual](#). `rstandard` will give you the `SPSS` Studentized residuals.

```
# Row 1: predicted values
row1 <- c(
  min(vasHNRmodel$fitted.values),
  max(vasHNRmodel$fitted.values),
  mean(vasHNRmodel$fitted.values),
  sd(vasHNRmodel$fitted.values),
  nobs(vasHNRmodel))

# Row 2: residuals
row2 <- c(
  min(vasHNRmodel$residuals),
  max(vasHNRmodel$residuals),
  mean(vasHNRmodel$residuals),
  sd(vasHNRmodel$residuals),
  nobs(vasHNRmodel))

# Row 3: standardized predicted values
stanfit <- (vasHNRmodel$fitted.values -
  mean(vasHNRmodel$fitted.values)) /
  sd(vasHNRmodel$fitted.values)
row3 <- c(
  min(stanfit),
  max(stanfit),
  mean(stanfit),
  sd(stanfit),
  nobs(vasHNRmodel))

# Row 4: standardized residuals
stanr <-
  vasHNRmodel$residuals / sqrt(sum(vasHNRmodel$residuals^2) / vasHNRmodel$df.residual)
row4 <- c(
  min(stanr),
```

```

max(stanr),
mean(stanr),
sd(stanr),
nobs(vasHNRmodel))

```

```

# Binding these into a data frame
residstats <- rbind(row1, row2, row3, row4)
colnames(residstats) <- c("Min", "Max", "Mean", "SD", "N")
rownames(residstats) <- c("Predicted Value", "Residuals", "Std.
Predicted", "Std. Residual")

```

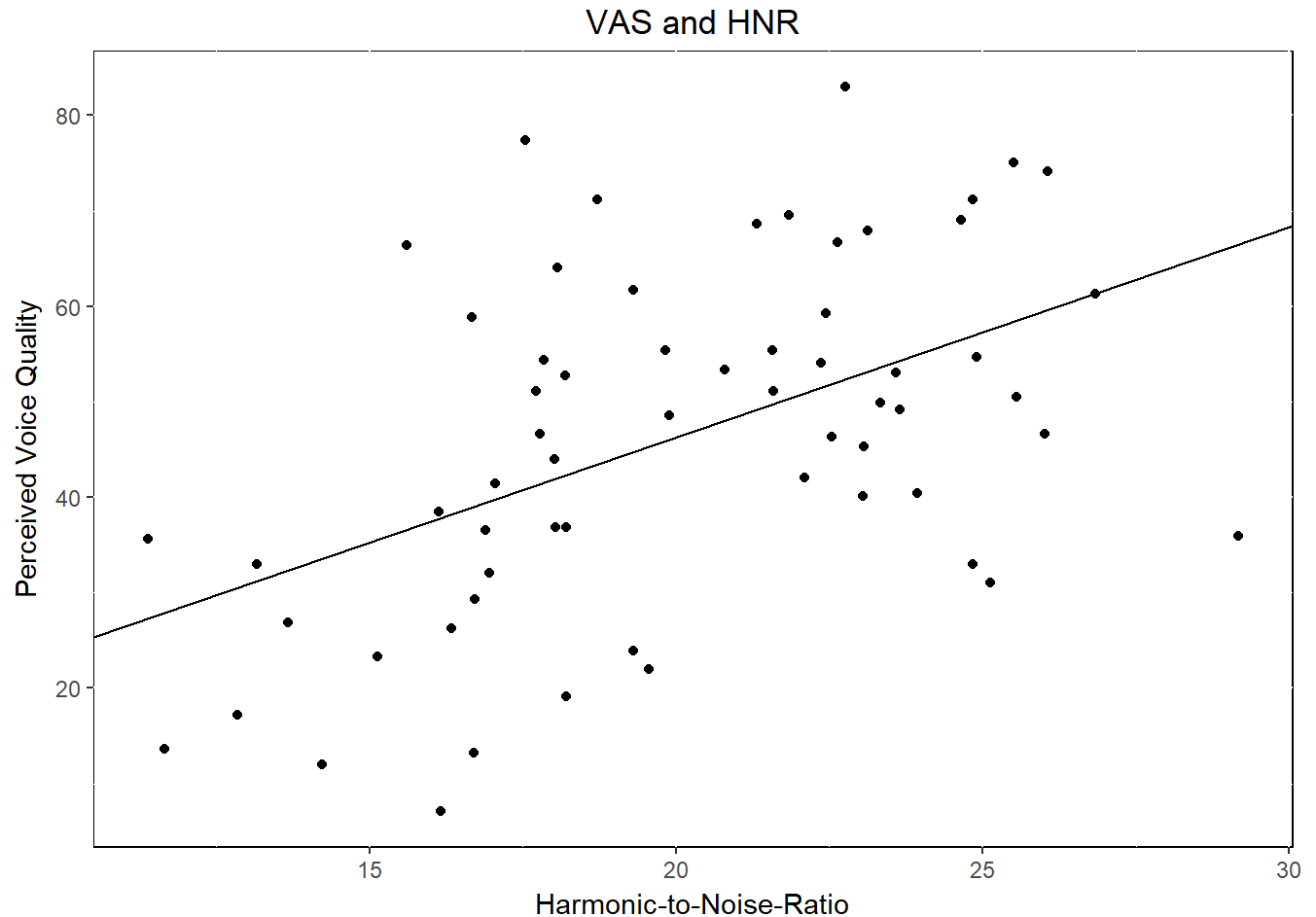
Now you've got all the same information SPSS provided in its output.

Our coefficients will allow us to add a trendline onto our plot of VAS and HNR. Let's plot that with ggplot. Note that in ggplot, you'll have to manually pick out the coefficients for the intercept and slope. The coefficient estimate for your intercept (top left of the coefficients table) is your intercept, and the estimate for your independent variable (HNR_OFF for us) is your slope.

```

library(ggplot2)
ggplot(OnOff, aes(x=HNR_OFF, y=VAS_OFF, na.rm=TRUE)) +
  geom_point() +
  geom_abline(intercept=2.222, slope=2.202) +
  labs(x="Harmonic-to-Noise-Ratio", y="Perceived Voice Quality",
       title="VAS and HNR") +
  theme(panel.background=element_rect(fill="white",
                                       colour="black")) +
  theme(plot.title = element_text(hjust = 0.5))

```



Here's a nice, simple visualization of the relationship between these two variables. For more on regression, check out the **Resources** section at the bottom.

Differences between two groups

Evaluating if two groups are significantly different on a given metric is a very common goal for many of us. We can do this in several ways, including t-tests, their non-parametric equivalents and ANOVA. Let's go through a few of these methods and compare the output we get to what we'd expect in SPSS.

Independent t-test

An independent t-test tests if two independent groups are significantly different on a measure. Similar to regression, we have 3 major assumptions to verify. We need independence of observations, a normal distribution of the dependent variable, and homogeneity of variance. Unlike regression, we don't need to test for independence of observations here - this is a study design characteristic. You should choose two groups that are not related. We do need to test for the other two assumptions: normality with the Shapiro-Wilk and homogeneity of variance with

Levene's test.

Let's do an independent t-test comparing PDs and controls on VAS (perceived voice quality). Our dependent variable will be VAS_OFF, and our independent variable will be PDGroup, our variable that specifies whether a person has PD or not. First we'll need to test for normal distribution using the Shapiro-Wilk. We've done this before, when we did correlations above. Go ahead and test the normality of VAS_OFF.

```
# Exercise 4: Testing normality of VAS_OFF with Shapiro-Wilk
```

Now we'll need to test homogeneity of variance with leveneTest, from the car package.

```
library(car)
```

```
leveneTest(OnOff$VAS_OFF, OnOff$PDGroup, center = mean)
```

Levene's test gives a p value > .05, so the assumption of homogeneity of variances is met and we can proceed with our t-test. If we did this in SPSS, our output would look like this:

Independent Samples Test							
		Levene's Test for Equality of Variances					
		F	Sig.	t	df	Sig. (2-tailed)	t-test t
VAS_OFF	Equal variances assumed	1.727	.194	-1.904	60	.062	-1
	Equal variances not assumed			-2.232	17.851	.039	-1

Independent t-test SPSS Output

Notice that SPSS runs Levene's test as part of its procedure, and the output table includes two rows: equal variances assumed, and equal variances not assumed. This is determined by the result of Levene's test. If homogeneity of variance is upheld, you can use Student's t-test. If Levene's test is significant, you must use a Welch t-test. In R, the default for t.test is to calculate a Welch t-test. If you're looking for a Student's t-test, be sure to specify var.equal = TRUE.

```
t.test(OnOff$VAS_OFF ~ OnOff$PDGroup, var.equal = TRUE)
```

```
t.test(OnOff$VAS_OFF ~ OnOff$PDGroup)
```

The output from our t-test includes almost all of the same information as the SPSS output. All we're missing is mean difference and the standard error of the difference. Mean difference is relatively easy to calculate - standard error of the difference is a bit more complex but here it is! Once again, these funky formulas are coming from the SPSS algorithm manual (linked above).

```
# Mean difference - the same for Student and Welch tests
```

```
pds <- filter(OnOff, OnOff$PDGroup == "PD")
```

```
controls <- filter(OnOff, OnOff$PDGroup == "Control")
```

```
meandiff <- mean(pds$VAS_OFF) - mean(controls$VAS_OFF)
```

```
npd <- sum(!is.na(pds$VAS_OFF))
```

```
ncon <- sum(!is.na(controls$VAS_OFF))
```

```
# Standard error of the difference: unpooled - for Welch's t-test
```

```
errordiffunpooled <- sqrt((var(pds$VAS_OFF)/npd) +  
                           (var(controls$VAS_OFF)/ncon))
```

```
# Standard error of the difference: pooled variance - for Student's  
t-test
```



```
poolvar <- ((npd-1)*var(pds$VAS_OFF) +
            (ncon-1)*var(controls$VAS_OFF))/(npd + ncon -2)
errordiffpooled <- sqrt(poolvar)*sqrt(1/npd+1/ncon)
```

Let's practice an independent t-test. Compare PDs and controls on HNR using an independent t-test. Test your assumptions of normality and homogeneity of variance, and then do the t-test variant that is appropriate based on your homogeneity of variance result.

```
# Exercise 5: Independent t-test comparing HNR_OFF across PDGroup
```

Mann-Whitney

If our assumption of normality is violated, we need to use a nonparametric test. The nonparametric equivalent of an independent t-test is a Mann-Whitney (or Mann-Whitney-Wilcoxon). Here's an example of a Mann-Whitney. We've specified exact = TRUE to allow us to compare to SPSS results.

```
shapiro.test(OnOff$Shimmer_OFF)
wilcox.test(OnOff$Shimmer_OFF ~ OnOff$PDGroup, paired = FALSE, exact = TRUE)
```

In SPSS, there are two ways to run a Mann-Whitney. Going through the legacy procedure and opting for an 'exact' significance instead of only asymptotic produces the following output:

Mann-Whitney Test

Ranks				
	PDGroup	N	Mean Rank	Sum of Ranks
Shimmer_OFF	Con	11	25.36	279.00
	PD	51	32.82	1674.00
	Total	62		

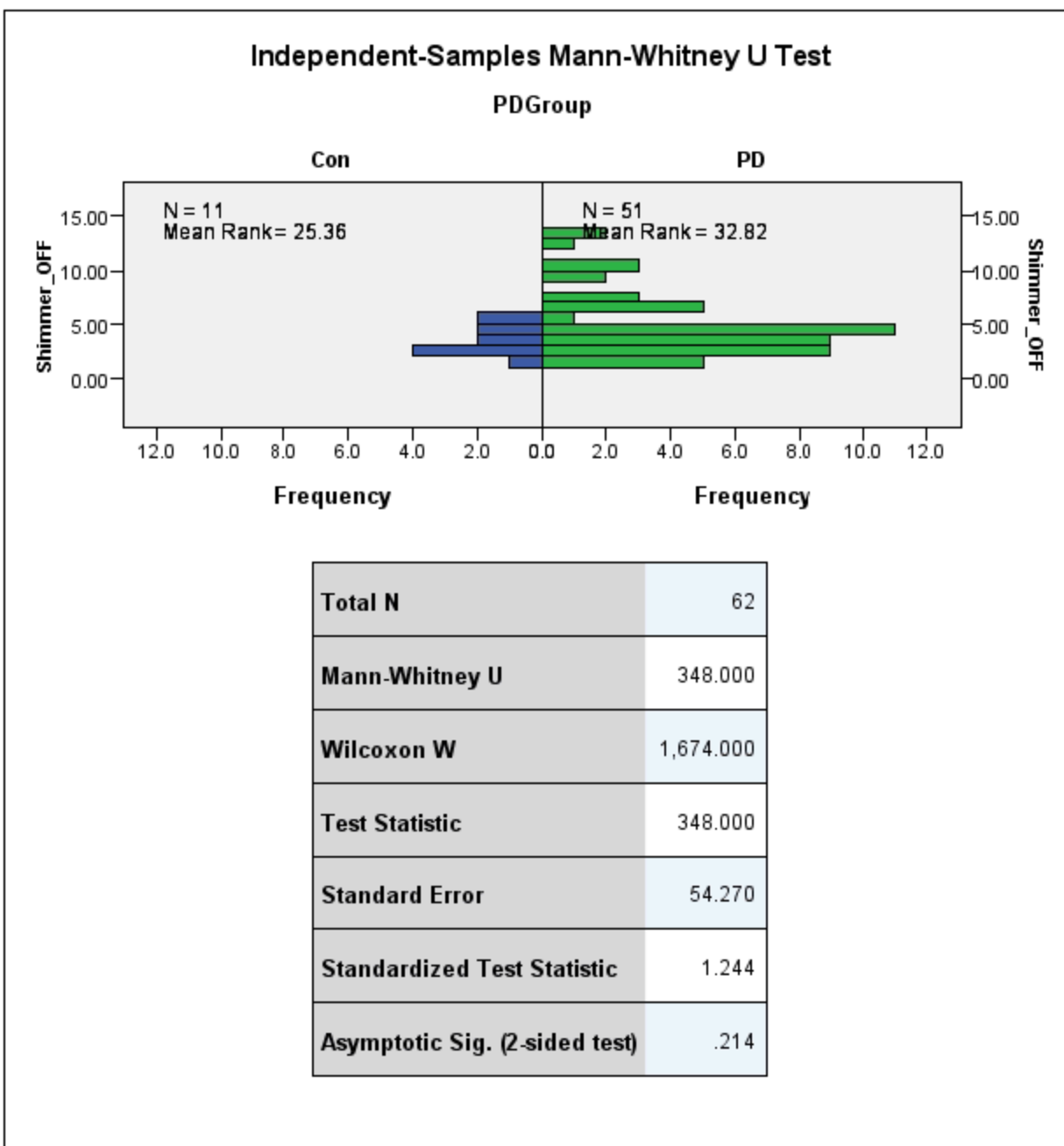
Test Statistics^a

	Shimmer_OF F
Mann-Whitney U	213.000
Wilcoxon W	279.000
Z	-1.244
Asymp. Sig. (2-tailed)	.214
Exact Sig. (2-tailed)	.220
Exact Sig. (1-tailed)	.110
Point Probability	.003

a. Grouping Variable: PDGroup

Mann-Whitney legacy procedure

This output matches our SPSS significance, but the test statistic is much different. This is a result of the order in which the groups were compared. If we flip the factor levels of 'PDGroup', we'll get a test statistic of 213 instead of 348. Note that what R calls W when we specify paired = FALSE is Mann-Whitney's U - confusing! This is discussed further [here](#). Using the new procedure, exact significance is not provided, so the p-value will not match this output.



Mann-Whitney new procedure

The test statistic here matches, but the p-value is off (because it's asymptotic). The asymptotic p-values differ in R and SPSS. As of yet, I haven't uncovered a way to fully reconcile these differences. This is another SPSS-R quirk on my to-do list! *Note:* Be aware that when using a

Mann-Whitney, you need to be sure that the shape of the distributions matches - otherwise, you cannot compare medians, only mean ranks. In this case, based on the distributions, we've only compared mean ranks. This is true in many cases - discussed more [here](#).

Dependent t-test

For comparisons among related groups in different conditions, a dependent t-test is needed. Dependent t-tests in R are very similar to their independent counterparts. We need to test for normality using Shapiro-Wilk again. If normality is met, we can go ahead and run our t-test. The syntax for this is the same except that we need to specify `paired = TRUE`. Let's compare the CPPS of PDs when they're on and off their medication. We made a data frame with only PD observations on and off medication using `select` and `filter` earlier called 'PDsOnOff', so we'll use that instead of our usual 'OnOff' data frame.

```
t.test(PDsOnOff$CPPS_OFF, PDsOnOff$CPPS_ON, paired = TRUE)
```

If we had done this in SPSS, we'd get the following output:

Paired Samples Test						
		Paired Differences				
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference	
					Lower	Upper
Pair 1	CPPS_OFF - CPPS_ON	-.22590	2.92116	.40904	-1.04749	.59568
						t
						-.552

SPSS dependent t-test

We get mostly the same information in our `t.test` output. SPSS also provides us the mean, standard deviation and mean standard error. We can get these by feeding the difference scores to `psych`'s `describe`.

```
diff <- PDsOnOff$CPPS_OFF - PDsOnOff$CPPS_ON
library(psych)
describe(diff)
```

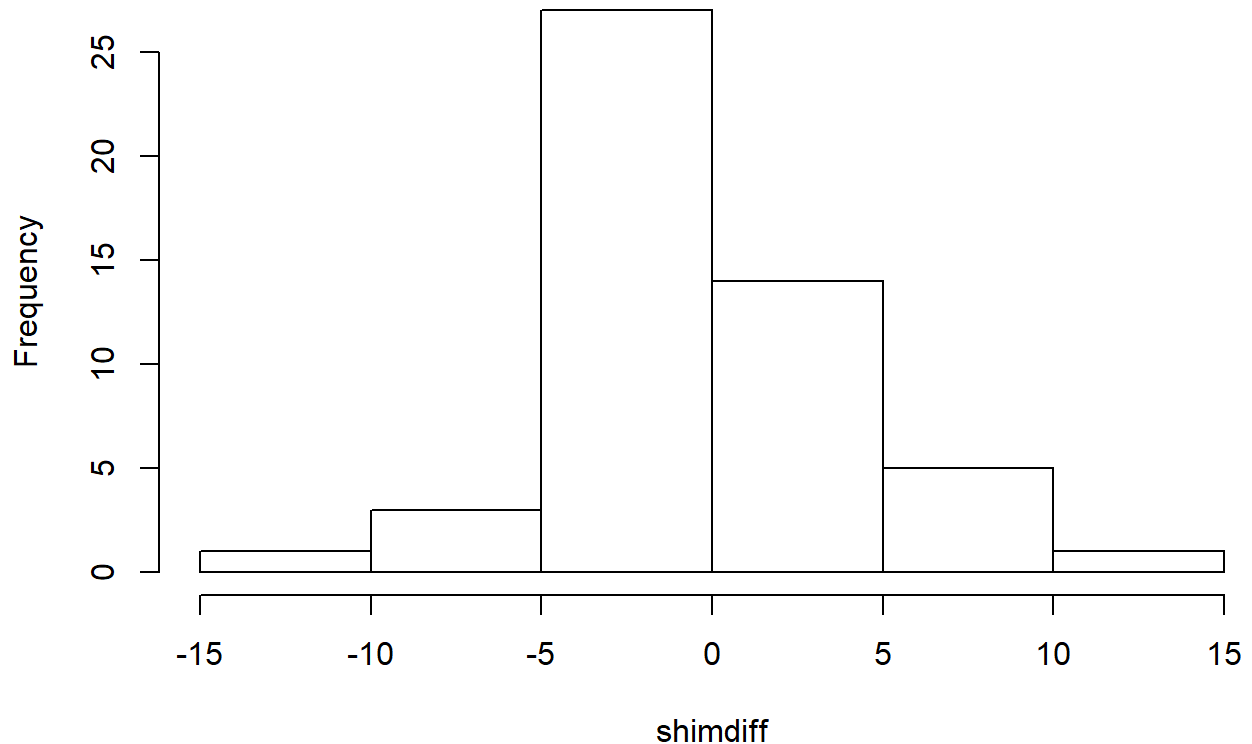
Wilcoxon Signed-Rank Test

Let's look at the Wilcoxon signed-rank test, the nonparametric counterpart to a dependent t-test. This has similar syntax to the `wilcox.test` call we used with the Mann-Whitney.

```
# Notice that the order in which you specify the groups
# changes the test statistic. SPSS uses the latter order.
```

```
wilcox.test(PDsOnOff$Shimmer_OFF, PDsOnOff$Shimmer_ON, paired = TRUE,
exact = TRUE)
wilcox.test(PDsOnOff$Shimmer_ON, PDsOnOff$Shimmer_OFF, paired = TRUE,
exact = TRUE)
shimdiff <- PDsOnOff$Shimmer_ON - PDsOnOff$Shimmer_OFF
hist(shimdiff)
```

Histogram of shimdiff



With a Wilcoxon signed-rank test, we have the same order effect on the test statistic as we saw with the Mann-Whitney. When comparing your results to SPSS results for sanity-checking purposes, try to remember that if you flip your groups (by changing your factor levels or by changing the order in the formula) you might solve your issue! We've called hist to quickly assess our distributional assumption in the Wilcoxon, which is that it should be approximately symmetrical. A fair amount of leeway is okay here, since symmetrical data is hard to come by, so I'd say this is symmetrical enough.

Here's the SPSS output from the legacy dialog to match the Wilcoxon above:

Wilcoxon Signed Ranks Test

Ranks

		N	Mean Rank	Sum of Ranks
Shimmer_ON - Shimmer_OFF	Negative Ranks	31 ^a	24.65	764.00
	Positive Ranks	20 ^b	28.10	562.00
	Ties	0 ^c		
	Total	51		

a. Shimmer_ON < Shimmer_OFF

b. Shimmer_ON > Shimmer_OFF

c. Shimmer_ON = Shimmer_OFF

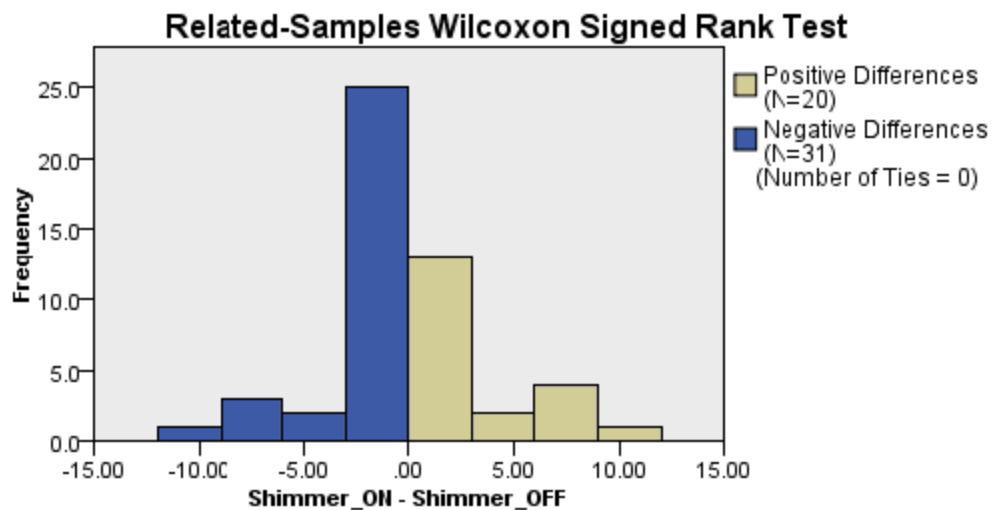
Test Statistics^a

Shimmer_ON - Shimmer_OFF	
F	
Z	-.947 ^b
Asymp. Sig. (2-tailed)	.344
Exact Sig. (2-tailed)	.349
Exact Sig. (1-tailed)	.175
Point Probability	.002

a. Wilcoxon Signed Ranks Test

b. Based on positive ranks.

SPSS Wilcoxon signed-rank: legacy



Total N	51
Test Statistic	562.000
Standard Error	106.684
Standardized Test Statistic	-.947
Asymptotic Sig. (2-sided test)	.344

SPSS Wilcoxon signed-rank: new

Again, we see that the asymptotic p-value (exact = FALSE) is different between SPSS and R, but if you specify an exact significance, you'll get the same results.

Let's see if there's a significant difference between PDs on and off medication on HNR.

```
# Exercise 5: Test for a significant difference between PDs on HNR
# depending on their medication state.
# Test your assumptions and then conduct the appropriate test.
```

ANOVA

ANOVA is a popular procedure in SPSS for many study designs, but it's less popular in R. This will be reflected by a couple issues we're going to encounter and discuss as we go through a few ANOVA designs as examples.

One-way ANOVA

One of the tests we did on this data was a one-way ANOVA comparing people's perceived voice quality based on their VASGroup. You might recall from above in the variable definitions that VASGroup divides people based on their voice quality off-medication into 3 groups: low voice quality, high voice quality and control. We'll need to start by testing normality and homogeneity using Shapiro-Wilk and Levene's.

```
shapiro.test(OnOff$VAS_OFF)
```

```
leveneTest(OnOff$VAS_OFF, OnOff$VASGroup, center = mean)
```

Our assumptions are upheld, so we'll go ahead and construct our model. There are a few ways we can do an ANOVA in R. I'll demonstrate 3 ways, which we can compare to SPSS and discuss. Before you run your ANOVA, you need to set your contrasts to match SPSS. This is discussed [here](#).

```
options(contrasts = c("contr.sum", "contr.poly"))
```

Using aov

```
# Specify the model using lm()
```

```
vasmod <- lm(VAS_OFF ~ VASGroup, data = OnOff)
```

```
vasaov <- aov(vasmod)
```

```
summary(vasaov)
```

Using Anova from the car package

```
# Specify the model using lm(), just like with aov()
```

```
vasmod <- lm(VAS_OFF ~ VASGroup, data = OnOff)
```

```
library(car)
```

```
vasAnova <- Anova(vasmod, type = 3)
```

```
vasAnova
```

Using ezANOVA from the ez package

```
library(ez)
```

```
vasEZ <- ezANOVA(data = OnOff, dv = VAS_OFF,  
                 between = VASGroup,  
                 wid = Participant, type = 3,  
                 detailed = TRUE,  
                 return_aov = TRUE  
                 )
```

```
vasEZ
```

Here's our SPSS output for this ANOVA:

		ANOVA				
		Sum of Squares	df	Mean Square	F	Sig.
VAS_OFF	Between Groups	13029.809	2	6514.904	52.907	.000
	Within Groups	7265.132	59	123.138		
	Total	20294.940	61			

SPSS one-way ANOVA

If we look at our sums of squares and F-values, we can see that for this test, aov and Anova results both give the same results, but ezANOVA provides different results. We'll just focus on

aov and Anova. In the Anova call, we specify type = 3. This is because SPSS uses Type III sums of squares and R defaults to Type I. But we didn't specify type in the aov call. In some cases, Type I and Type III sums of squares will be the same. This will depend on characteristics of your individual model.

Post-hoc testing There are some downsides to using Anova over aov. For example, aov is easier to do post-hoc testing with. If you want to do Tukey HSD, you need an aov, as neither Anova nor ezANOVA support Tukey. You can do pairwise t-tests with any option though, and you can specify p value adjustment methods like Bonferroni and Holm.

```
TukeyHSD(vasaov)
```

```
pairwise.t.test(OnOff$VAS_OFF, OnOff$VASGroup, p.adjust = "bonf")
```

Here's what SPSS gives us for post-hoc testing:

Multiple Comparisons							
Dependent Variable		(I) VASGroup	(J) VASGroup	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval Lower Bound
VAS_OFF	Tukey HSD	Con	High	-4.25566	4.01495	.542	-13.9086
			Low	26.26662 [*]	3.99129	.000	16.6706
		High	Con	4.25566	4.01495	.542	-5.3973
			Low	30.52228 [*]	3.10831	.000	23.0491
		Low	Con	-26.26662 [*]	3.99129	.000	-35.8627
			High	-30.52228 [*]	3.10831	.000	-37.9954
Bonferroni	Con	High	-4.25566	4.01495	.880	-14.1491	5.6378
		Low	26.26662 [*]	3.99129	.000	16.4315	36.1018
	High	Con	4.25566	4.01495	.880	-5.6378	14.1491
		Low	30.52228 [*]	3.10831	.000	22.8629	38.1816
	Low	Con	-26.26662 [*]	3.99129	.000	-36.1018	-16.4315
		High	-30.52228 [*]	3.10831	.000	-38.1816	-22.8629

Both post-hoc test options give us the same results as R here.

One-way repeated measures ANOVA

You can't do a one-way repeated measures ANOVA with this data, so we're going to import some data just for demonstration. We'll be importing WorldPhones, which contains information on the number of telephones in the world from 1951 to 1961.

```
data(WorldPhones)
```

```
head(WorldPhones)
```

This data has the years as the observations, rather than as variables. We're going to flip this using the transpose function t.

```
phonesT <- t(WorldPhones)
```

```
phonesT <- data.frame(phonesT)
```

I'm not really interested in looking at every year - I'd rather just look at 1951, 1956 and 1961

since those are the mid and endpoints.

```
# Exercise 6: Select 1951, 1956 and 1961 from phonesT.  
# Tip: look at phonesT with head() to see what the exact  
# variable names are. Name this new data frame "phones_wide"
```

Let's rename those columns for ease, and turn those row names into their own column - we're going to need the countries in their own variable for analysis.

```
colnames(phones_wide) <- c("1951", "1956", "1961")  
countries <- rownames(phones_wide)  
phones_wide <- cbind(countries, phones_wide)
```

If you were at the last R-Ladies meetup with Monica, you'll remember talking about "tidy" data.

For our larger dataset, we left it pretty wide because of the nature of the data. For this one, we're going to want to gather this into a long dataset. For more on long and wide data, look [here](#) or check out the 'Cleaning data in R' course on DataCamp.

```
library(tidyr)  
phones <- gather(phones_wide, Year, Amount, "1951", "1956", "1961")
```

```
# Factoring our categorical variable and case identifier
```

```
phones$Year <- factor(phones$Year)  
phones$countries <- factor(phones$countries)
```

That data's looking nice and tidy. We're ready to do our analysis, but first let's write it to a .sav so we could compare it in SPSS. We're actually going to write our wide version, because SPSS prefers wide data.

```
# Exercise 7: Write our phones_wide data frame to a .sav using haven  
# Hint: we did this earlier in the 'Writing an SPSS file' section
```

We need to test our assumptions of sphericity and normality. We can test normality in advance with Shapiro-Wilk.

```
library(car)  
shapiro.test(phones$Amount)
```

Let's look at a few ways to run this ANOVA.

Multivariate model This method is actually a multivariate design, we'll just be using the univariate table from it. This method and the rationale for using it are by Paul Gribble and you can read about it [here](#). This method will run a Mauchly's test as part of its procedure, so we can go ahead and construct our model.

A multivariate model requires a matrix, not a data frame. A matrix has to have all the same class, so we need to remove our column containing country names. This matrix needs to be in wide format, so we'll use our pre-tidied version. We also will need to define a factor including only our factor levels.

```
# Getting our data into a matrix, not a data frame  
phones_grib <- select(phones_wide, -countries)  
phones_grib <- as.matrix(phones_grib)
```

```
# Defining a multivariate model  
mod <- lm(phones_grib ~ 1)
```

```
# Defining our design factor  
year <- factor(c("1951", "1956", "1961"))
```

```
# Specifying the model
```

```
modAnova <- Anova(mod, idata = data.frame(year),
  idesign = ~year, type = "III")
```

```
# Multivariate = FALSE just tells R to skip printing
# the multivariate tables.
```

```
summary(modAnova, multivariate = FALSE)
```

Using aov This method is simple, but we can't run Mauchly's test of Sphericity using this method.

```
phonesaov <- aov(Amount ~ Year + Error(countries/Year), data =
phones)
```

```
summary(phonesaov)
```

Using ezANOVA This is the simplest method for this type of ANOVA. Mauchly's test is included in the function.

```
phonesEZ <- ezANOVA(data = phones, dv = Amount, wid = countries,
  within = Year, type = 3, detailed = TRUE,
  return_aov = TRUE)
```

```
phonesEZ$ANOVA
```

```
phonesEZ$`Mauchly's Test for Sphericity`
```

Here's our SPSS output so we can compare:

Tests of Within-Subjects Effects

Measure: Amount

Source		Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Year	Sphericity Assumed	325930593.4	2	162965296.7	3.804	.053	.388
	Greenhouse-Geisser	325930593.4	1.003	324841710.4	3.804	.099	.388
	Huynh-Feldt	325930593.4	1.005	324190705.6	3.804	.099	.388
	Lower-bound	325930593.4	1.000	325930593.4	3.804	.099	.388
Error(Year)	Sphericity Assumed	514092321.2	12	42841026.77			
	Greenhouse-Geisser	514092321.2	6.020	85395803.22			
	Huynh-Feldt	514092321.2	6.032	85224664.26			
	Lower-bound	514092321.2	6.000	85682053.54			

SPSS repeated measures ANOVA

Mauchly's Test of Sphericity^a

Measure: Amount

					Epsilon ^b		
Within Subjects Effect	Mauchly's W	Approx. Chi-Square	df	Sig.	Greenhouse-Geisser	Huynh-Feldt	Lower-bound
Year	.007	25.042	2	.000	.502	.503	.50

Mauchly's test

Our Mauchly results for ezANOVA and our multivariate method match up nicely with SPSS output. Similarly, the ANOVA results for all 3 of these methods matches SPSS. aov's greatest flaw is that it doesn't support Mauchly's test, but none of these methods allow for easy post-hoc

testing.

Post-hoc testing For all of these methods, we'll need to do pairwise t-tests as our post-hoc tests. Let's look at a Bonferroni correction here.

```
pairwise.t.test(phones$Amount, phones$Year, p.adjust = "bonf")
pairwise.t.test(phones$Amount, phones$Year, p.adjust = "none")
```

Pairwise Comparisons

Measure: Amount

(I) Year	(J) Year	Mean Difference (I-J)	Std. Error	Sig. ^a	95% Confidence Interval for Difference ^a	
					Lower Bound	Upper Bound
1	2	-3957.857	2053.939	.307	-10710.088	2794.374
	3	-9600.857	4920.455	.297	-25776.631	6574.917
2	1	3957.857	2053.939	.307	-2794.374	10710.088
	3	-5643.000	2879.469	.293	-15109.126	3823.126
3	1	9600.857	4920.455	.297	-6574.917	25776.631
	2	5643.000	2879.469	.293	-3823.126	15109.126

Based on estimated marginal means

a. Adjustment for multiple comparisons: Bonferroni.

SPSS post-hoc testing

Note that these results are bunch different than SPSS' pairwise results - R yields much higher p values even without the Bonferroni correction. This is not a difference I've been able to reconcile yet.

Two-way ANOVA

Doing a two-way ANOVA will be similar to the one-way ANOVA we did above, but gives us an opportunity to discuss interaction. Let's do an ANOVA comparing perceived voice quality among PDs as a function of VASGroup and DurationGroup. DurationGroup bins individuals into 4 groups based on their disease duration. With a two-way ANOVA, we need to test for normality and homogeneity of variance, like we did with a one-way.

```
shapiro.test(pds$VAS_OFF)
leveneTest(pds$VAS_OFF, pds$VASGroup, center = mean)
leveneTest(pds$VAS_OFF, pds$DurationGroup, center = mean)
```

Using aov Without drop1, aov's results do not match SPSS. Adding drop1 changes the term deletion to align it with SPSS. Read more about this [here](#).

```
twowayaov <- aov(VAS_OFF ~ VASGroup*DurationGroup, data = pds)
drop1(twowayaov, .~., test = "F")
```

Using Anova

```
options(contrasts = c("contr.sum", "contr.poly"))
twowayanova <- Anova(lm(VAS_OFF ~ VASGroup*DurationGroup, data=pds),
type=3)
```

Using ezANOVA

```

twowayEZ <- ezANOVA(data = pds, dv = VAS_OFF, wid = Participant,
                    between = .(VASGroup, DurationGroup), type =
3,
                    detailed = TRUE,
                    return_aov = TRUE
)
twowayEZ$ANOVA

```

Here's what SPSS gives us:

Tests of Between-Subjects Effects

Dependent Variable: VAS_OFF

Source	Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared	Noncent. Paramete
Corrected Model	12192.066 ^a	7	1741.724	15.409	.000	.715	107.8
Intercept	98587.523	1	98587.523	872.223	.000	.953	872.2
VASGroup	11748.545	1	11748.545	103.942	.000	.707	103.9
DurationGroup	271.069	3	90.356	.799	.501	.053	2.3
VASGroup * DurationGroup	61.536	3	20.512	.181	.908	.013	.5
Error	4860.299	43	113.030				
Total	117419.487	51					
Corrected Total	17052.365	50					

a. R Squared = .715 (Adjusted R Squared = .669)

b. Computed using alpha = .05

SPSS two-way ANOVA

All 3 of our methods align with SPSS results.

Post-hoc testing We need to do post-hoc testing on DurationGroup, since we have 4 levels.

```
pairwise.t.test(pds$VAS_OFF, pds$DurationGroup, p.adjust = "bonf")
```

```
TukeyHSD(twowayaov)
```

Multiple Comparisons

Dependent Variable: VAS_OFF

	(I) DurationGroup	(J) DurationGroup	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
Tukey HSD	Early	EarlyMid	-2.2923	4.18244	.947	-13.4696	8.88
		MidLate	-5.4323	4.01835	.536	-16.1710	5.30
		Late	-3.4569	4.28358	.851	-14.9044	7.99
	EarlyMid	Early	2.2923	4.18244	.947	-8.8849	13.46
		MidLate	-3.1399	4.18244	.876	-14.3172	8.03
		Late	-1.1646	4.43787	.994	-13.0244	10.69
	MidLate	Early	5.4323	4.01835	.536	-5.3065	16.17
		EarlyMid	3.1399	4.18244	.876	-8.0373	14.31
		Late	1.9754	4.28358	.967	-9.4722	13.42
	Late	Early	3.4569	4.28358	.851	-7.9906	14.90
		EarlyMid	1.1646	4.43787	.994	-10.6953	13.02
		MidLate	-1.9754	4.28358	.967	-13.4229	9.47
Bonferroni	Early	EarlyMid	-2.2923	4.18244	1.000	-13.8603	9.27
		MidLate	-5.4323	4.01835	1.000	-16.5464	5.68
		Late	-3.4569	4.28358	1.000	-15.3046	8.39
	EarlyMid	Early	2.2923	4.18244	1.000	-9.2756	13.86
		MidLate	-3.1399	4.18244	1.000	-14.7079	8.42
		Late	-1.1646	4.43787	1.000	-13.4390	11.10
	MidLate	Early	5.4323	4.01835	1.000	-5.6819	16.54
		EarlyMid	3.1399	4.18244	1.000	-8.4280	14.70
		Late	1.9754	4.28358	1.000	-9.8723	13.82
	Late	Early	3.4569	4.28358	1.000	-8.3908	15.30
		EarlyMid	1.1646	4.43787	1.000	-11.1099	13.43
		MidLate	-1.9754	4.28358	1.000	-13.8231	9.87

Based on observed means.

The error term is Mean Square(Error) = 113.030.

SPSS two-way post-hoc tests

Our Bonferroni results match SPSS, but once again, Tukey results are different. There is likely a difference in how it's being calculated which could be uncovered with a little (a lot) more digging. That's a problem for another day!

ANOVA Practice

Try a one-way ANOVA on your own. Compare CPPS_OFF across DurationGroup. Don't forget

to test your assumptions of normality and homogeneity of variance. Use Tukey's HSD as your post-hoc test.

```
# Exercise 8: one-way ANOVA examining CPPS across DurationGroup
# Test assumptions of normality and homogeneity of variance, and
# use Tukey for post-hoc testing.
```

Conclusion

I hope this meetup was helpful in some ways! The transition from SPSS to R isn't the simplest, and there are certainly quirks of each that can make things more complex - mostly if you're looking to match SPSS and R results exactly. Results provided by SPSS and R are both credible, even when they differ. The reason I've tried to match results here is that I recognize many are hesitant to switch to R, feeling like you know what SPSS does and what those results mean to you. I hope I've been able to demonstrate that for many tests, these results do align well. In other cases, mismatches are not necessarily because one of the two is doing something *incorrect*; there are lots of parameters and decisions made behind the scenes and there can be multiple ways to meet similar statistical goals with credibility.

Resources

If you're stuck, you can reach out to me - I probably won't have an answer for you off-hand, but I'm certainly willing to look into things with you. [*dcushni@uwo.ca*](mailto:dcushni@uwo.ca)

- <http://www.r-tutor.com/elementary-statistics>
- <http://tutorials.iq.harvard.edu/R/Rstatistics/Rstatistics.html>
- <http://r-statistics.co/Assumptions-of-Linear-Regression.html>
- <http://www.statmethods.net/stats/anova.html>
- <http://myowelt.blogspot.ca/2008/05/obtaining-same-anova-results-in-r-as-in.html>
- <http://www.sthda.com/english/wiki/correlation-matrix-a-quick-start-guide-to-analyze-format-and-visualize-a-correlation-matrix-using-r-software>
- <https://psychwire.wordpress.com/2011/07/10/migrating-from-spssexcel-to-r/>
- <https://www.r-bloggers.com/example-2014-6-comparing-medians-and-the-wilcoxon-rank-sum-test/>
- <http://rmhogervorst.nl/cleancode/blog/2016/02/20/from-spss-to-r-part1.html>
- <http://gribblelab.org/stats/index.html>
- ftp://public.dhe.ibm.com/software/analytics/spss/documentation/statistics/24.0/en/client/Mannuals/IBM_SPSS_Statistics_Algorithms.pdf