# R (mosaic) – Key Commands

**Get going**

install.packages("xyz")

require(xyz) ← starting mosaic, psych, psy, ggplot2, openxlsx, readxl, plspm
(required for each session)

**Recoding, computing, formatting variables & creating new datasets:**

Recoding (new) variables:

dataset$new_variable[dataset$old_variable=="XXX"] <- "new_value"

→ Please note: categorial values require quotes (""), numerical values not

Calculating new variables:

dataset$new_variable <- dataset$variable1 + dataset$variable2

Changing the the variable formats:

dataset$new_variable <- as.numeric(dataset$variable)

→ factor to numeric

(1): dataset$variable <- as.factor(dataset$variable)

(2): levels(dataset$variable) <- c("attribute1", "attribute1")

→ numeric to factor

Creating new datasets (is equal, is not equal etc.):

new_dataset <- dataset[dataset$variable == "value",]

new_dataset <- dataset[dataset$variable != "value",]

new_dataset <- dataset[dataset$variable > value,]

**General command structure**

command(y~x, z)

**Frequencies**

tally(~variable1, data = dataset)

tally(~variable1, format = "percent", data = dataset)

tally(variable1~variable2, data = dataset)

tally(variable1~variable2, format = "percent", data = dataset)

**Median, mean, variance, standard deviation, maximum, minimum**

median(~variable, data = dataset) / mean(~variable, data = dataset)

var(~variable, data = dataset) / sd(~variable, data = dataset)

max(~variable, data = dataset) / min(~variable, data = dataset)

favstats(~variable, data = dataset)

→ result: Min, Q1, median, Q3, Max , mean, standard deviation, n, missing values)





**Chi² test**

xchisq.test(variable1~variable2, data = dataset)

**t-test**

t.test(variable~grouping_variable, data = dataset)

**ANOVA**

summary(aov(variable~grouping_variable, data = dataset))

→ alternative way:

modelxy <- aov(variable~grouping_variable, data = dataset)

summary(modelxy)

**Shapiro-Wilk test**

shapiro.test(dataset$variable)

**Mann-Whitney & Wicoxon test**

wilcox.test(variable~grouping_variable, data = dataset)

(→Mann-Whitney test)

wilcox.test(dataset$variable1, dataset$variable2, paired = T)

(→ Wilcoxon signed rank test (paired sample))

**Correlation**

cor.test(variable1~variable2, data = dataset)

**Linear regression**

summary(lm(dependent_variable~independent_variable, data = dataset))

**Multiple regression**

summary(lm(dependent_variable~ind_variable1 + ind_variable2, data = dataset))

**Principal component analysis** (package: psych)

dataset_new <- cbind(dataset$variable1, dataset$variable2, dataset$variable3…)

colnames(dataset_new) <- c("Item_1", "Item_2")

KMO(dataset_new)

pcaX <- princomp(dataset_new, scores = TRUE, cor = TRUE)

summary(pcaX) plot(pcaX)

principal(dataset_new, nfactors = x, rotate = "varimax")

**Cronbach's alpha** (package: psy)

factorX <- cbind(dataset$variable1, dataset$variable2, dataset$variable3…)

cronbach(factorX)

**Graphs** (package: ggplot2)

→ Please note: formatting parameters have standard setting and can be adjusted

Bar graph – frequencies with 1 variable

```
table <- tally(~variable, format = "proportion",data = dataset)

frame <- as.data.frame(table)

View(frame)

ggplot(frame, aes(variable, Freq)) + geom_bar(position = "dodge", stat = "identity", fill = "#143C78") +
scale_y_continuous(limits = c(min, max)) + geom_text(aes(label = sprintf("%0.2f", frame$Freq)),
position = position_dodge(width = 0.9), vjust = -0.25, colour = "black", size = 2.7)
```

Bar graph – frequencies with 2 variables

```
table <- tally(variable_1~variable_2, format= "proportion",data = dataset)

frame <- as.data.frame(table)

View(frame)

ggplot(frame, aes(variable_1, Freq, fill = variable_2)) + geom_bar(position = "dodge", stat = "identity")
+ scale_fill_manual(values = c("#143C78", "#CDEBFF")) + scale_y_continuous(limits = c(min, max)) +
geom_text(aes(label = sprintf("%0.2f", frame$Freq)), position = position_dodge(width = 0.9), vjust = -
0.25, colour = "black", size = 2.7)
```

→ Please note: the number of colours in "scale_fill_manual" depends on the number of values of variable_2

Pie chart (& stacked bar graph with 1 variable)

```
table <- tally(~variable, format = "proportion", data = dataset)

frame <- as.data.frame(table)

View(frame)

ggplot(frame, aes("", Freq, fill = variable)) + geom_bar(stat = "identity") + scale_fill_manual(values =
c("#143C78", "#CDEBFF")) + geom_text(aes(label = sprintf("%0.2f", frame$Freq)), position =
position_stack(vjust = 0.5), colour = "white", size = 2.7)+ coord_polar("y", start = 0)
```

→ Please note: the number of colours in "scale_fill_manual" depends on the number of values of variable

Stacked bar graph with 2 variables

```
table <- tally(variable_1~variable_2, format= "proportion",data = dataset)

frame <- as.data.frame(table)

View(frame)

ggplot(frame, aes(variable_1, Freq, fill = variable_2)) + geom_bar(stat = "identity") +
scale_fill_manual(values = c("#143C78", "#CDEBFF")) + geom_text(aes(label = sprintf("%0.2f",
frame$Freq)), position = position_stack(vjust = 0.5), colour = "white", size = 2.7)
```

→ Please note: the number of colours in "scale_fill_manual" depends on the number of values of variable_2





Bar graph – means with 1 grouping variable (median, standard deviation etc.)

```
frame <- df_stats(variable~grouping_variable, data = dataset, mean)

View(frame)

ggplot(frame, aes(grouping_variable, mean_variable)) + geom_bar(position = "dodge", stat =
"identity", fill = "#143C78") + scale_y_continuous(limits = c(min,max)) + geom_text(aes(label =
sprintf("%0.2f", frame$mean_variable)), position = position_dodge(width = 0.9), vjust = -0.25, colour =
"black", size = 2.7)
```

Bar graph – means with 2 grouping variables (median, standard deviation etc.)

```
frame <- df_stats(variable~grouping_variable_1 + grouping_variable_2, data = dataset, mean)

View(frame)

ggplot(frame, aes(grouping_variable_2, mean_Variable, fill=grouping_variable_1)) +
geom_bar(position = "dodge",stat = "identity") + scale_y_continuous(limits = c(min, max)) +
scale_fill_manual(values = c("#143C78", "#CDEBFF")) + geom_text(aes(label = sprintf("%0.2f",
frame$mean_Variable)), position = position_dodge(width = 0.9), vjust = -0.25, colour = "black", size =
2.7)
```

→ Please note: the number of colours in "scale_fill_manual" depends on the number of values of grouping_variable_1

Colours:

| | | | |
|---|---|---|---|
| Navy (dark & light) | | Purple (dark & light) | |
| #143C78 | (rgb= 20, 60, 120) | #4E008D | (rgb= 78, 0, 141) |
| #CDEBFF | (rgb= 205, 235, 255) | #9400D3 | (rgb= 148, 0, 211) |
| Maroon (dark & light) | | Green (dark & light) | |
| #660632 | (rgb= 102, 0, 50) | #073F15 | (rgb= 7, 63, 21) |
| #F08ABC | (rgb= 240, 138, 188) | #C7FFD5 | (rgb= 199, 255, 213) |

**Export R results to Excel** (package: openxlsx)

(1.1) Preparing frequencies

```
frame <- data.frame(tally(variable_1~variable_1, data = dataset))
```

Alternative:

```
table <- tally(variable_1~variable_2, format= "proportion", data = dataset)

frame <- as.data.frame(table)
```

(1.2) Preparing mean, median etc.

```
frame <- data.frame(new_variable_name_1 = c(mean(~variable_1, data = dataset),
mean(~variable_2, data = dataset)), new_variable_name_2 = c(mean(~variable_3, data = dataset),
mean(~varaible_4, data = dataset)))
```

Alternative:

```
frame <- df_stats(variable~grouping_variable, data = dataset, mean)
```

(2) Export to Excel

```
write.xlsx(frame, "Desktop/R_operations/name.xlsx", asTable = FALSE)
```

→ Please note: The name of the R frame & the directory / name of the xlsx-file have to be adjusted

**Partial Least Squares Path Modeling** (package: plspm)

**(1) PLSPM – Basic Application**

Building the inner model

```
latent_variable_A = c(0, 0, 0)
latent_variable_B = c(0, 0, 0)
latent_variable_C = c(1, 1, 0)
```

→ "1" = dependent from column

```
pls = rbind(latent_variable_A, latent_variable_B, latent_variable_C)
colnames(pls) = rownames(pls)

pls
innerplot(pls)
```

Building the outer model

```
pls_blocks = list(c("variableA1", "variableA2", "variableA3"), c("variableB1", "variableB2", "variableB3"), c("variableC1", "variableC2", "variableC3"))
pls_modes = rep("A", n)
```

→ Please note: n = number of latent variables

Running plspm & results

```
pls_results = plspm(data, pls, pls_blocks, pls_modes)

pls_results

pls_results$outer_model
plot(pls_results, what = "loadings", arr.width = 0.1)
pls_results$unidim
pls_results$crossloadings

pls_results$inner_model
pls_results$path_coefs
plot(pls_results)
pls_results$inner_summary
```

Bootstrap validation

```
pls_results_val = plspm(data, pls, pls_blocks, modes = pls_modes, boot.val = TRUE, br = 5000)
pls_results_val$boot
```





**(2) PLSPM – Group Comparison**

1st Perform basic PLSPM

2nd Perform group comparison

```
pls_group_1 = dataset[dataset$variable == "attribute_1", ]
pls_group_1_results = plspm(pls_group_1, pls, pls_blocks, modes = pls_modes)
pls_group_2 = dataset[dataset$variable == "attribute_2", ]
pls_group_2_results = plspm(pls_group_2, pls, pls_blocks, modes = pls_modes)

dataset$variable <- as.factor(dataset$variable)
```

→ Please note: this step is required as the grouping variable must be a factor

```
group_comparison_results = plspm.groups(pls_results, dataset$variable, method = "bootstrap")
group_comparison_results
```

**(3) PLSPM – Moderator Variable**

1st Perform basic PLSPM

2nd Calculate product indicators

Definitions:

variable_1 = indicator 1 of influenced variable
moderator_1 = indicator 1 of moderator variable
inter_1 = indicator 1 of interaction/moderating effect

(for categorical moderators (2 groups):

```
moderator_1 = rep(0, number of rows)
moderator_1 [grouping_variable == "attribute"] = 1
dataset$moderator_1 = moderator_1
```

)

```
dataset$inter_1 = dataset$moderator_1 * dataset$variable_1
dataset$inter_2 = dataset$moderator_1 * dataset $variable_2
…
dataset$inter_3 = dataset$moderator_2 * dataset $variable_1
dataset$inter_4 = dataset$moderator_2 * dataset $variable_2
…
```

3rd Perform PLSPM with interacting variable (inter)